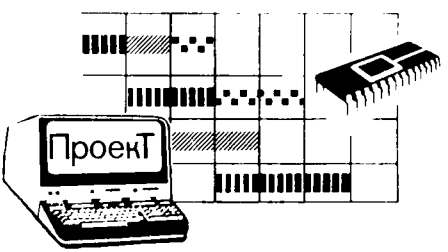
Ю.Н. Арсеньев
В.М. Журавлев

ПРОЕКТИРОВАНИЕ СИСТЕМ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ НА МИКРОПРОЦЕССОРНЫХ СРЕДСТВАХ



Допущено Государственным комитетом СССР по народному образованию в качестве учебного пособия для студентов вузов, обучающихся по специальности «Вычислительные машины, комплексы, системы и сети».

Москва
«Высшая школа» 1991

ББК 32.973.2
А85
УДК 681.326.32

Рецензенты:

кафедра вычислительной техники Ленинградского института точной механики и оптики (зав. кафедрой — д-р техн. наук, проф. Г. И. Новиков); д-р техн. наук, проф. Н. М. Соломатин (МГТУ им. Н. Э. Баумана)

Арсеньев Ю. Н., Журавлев В. М.

А85 Проектирование систем логического управления на микропроцессорных средствах: Учеб. пособие для вузов по спец. «Вычисл. машины, комплексы, системы и сети». — М.: Высш. шк., 1991. — 319 с.: ил.

ISBN 5-06-001765-6

Рассмотрены вопросы проектирования систем логического управления на базе микропроцессорных средств.

Дан анализ методов повышения надежности и отказоустойчивости проектируемых систем за счет структурной, временной, информационной и программной избыточности. Изложение материала иллюстрируется практическими примерами.

А $\frac{2404090000(4309000000)-368}{001(01)-91}$ 168—91

ББК 32.973.2
6Ф7.3

ISBN 5-06-001765-6

ПРЕДИСЛОВИЕ

Интенсификация современного производства и ускорение научно-технического прогресса в значительной мере определяются степенью автоматизации разнообразных технологических и производственных процессов на базе широкого применения новейших средств микропроцессорной и вычислительной техники и уровнем подготовки инженерных кадров в стране. В соответствии с этим в учебных программах подготовки инженеров по таким специальностям, как, например, «Автоматика и управление в технических системах», «Вычислительные машины, комплексы, сети и системы», «Электропривод и автоматизация промышленных установок» и др., уделяется серьезное внимание изучению микропроцессорных средств, а также их проектированию и применению в системах автоматизации, управления и обработки информации различного назначения.

В пособии изложены вопросы выбора и проектирования управляющих логических устройств и систем управления на основе микропроцессорных средств, обеспечения их надежности и отказоустойчивости при функционировании. От выпущенных ранее учебных пособий, например [1—10], данное пособие отличается освещением методов проектирования и обеспечения надежности довольно широкого класса систем управления — систем логического управления (СЛУ), реализуемых в виде типовых проектных решений (аппликаций) на средних и больших интегральных микросхемах, микропроцессорах и микроЭВМ и выполняющих заданные алгоритмы или классы функций программным и (или) аппаратным путем.

В основу настоящего учебного пособия положены материалы лекций, лабораторных и практических занятий, разработанных авторами, а также результаты их научных исследований. Изложение теоретического материала сопровождается примерами, контрольными вопросами.

Материал книги распределяется между авторами следующим образом: введение, главы 1, 2, 6 написаны совместно, главы 3—5, 7, 8 и приложения — Ю. Н. Арсеньевым.

Авторы выражают глубокую благодарность рецензентам книги — коллективу кафедры вычислительной техники Ленинградского института точной механики и оптики (зав. кафедрой — проф. Г. И. Новиков), проф. МГТУ им. Н. Э. Баумана Н. М. Соломатину. Их замечания и доброжелательные советы несомненно способствовали улучшению содержания книги.

Замечания и отзывы о содержании книги можно направлять по адресу: *101430, Москва, ГСП-4, Неглинная ул., 29/14, издательство «Высшая школа».*

Авторы

СПИСОК ОСНОВНЫХ СОКРАЩЕНИЙ

АН — автомат надежности
АК — адаптер-контроллер
АВО — адаптивный восстанавливающий орган
А/Д — адрес / данные
АЛУ — арифметико-логическое устройство
АСУ — автоматизированная система управления
АСТПП — автоматизированная система технологической подготовки производства
АСУП — автоматизированная система управления предприятием
АСНИКИ — автоматизированная система научных исследований и комплексных испытаний
АСУ ГПС — автоматизированная система управления гибкой производственной системой
АсН — ассоциативный накопитель
АЦП — аналого-цифровой преобразователь
БГ — билогический граф
БИС — большая интегральная схема
БКС — базовая конфигурация структуры
БМУ — блок микропрограммного управления
БС — базовая система
БСА — бортовая система автоматики
БУП — блок управления переключением
БФ — булева функция
ВБР — вероятность безотказной работы
ВО — восстанавливающий орган
ВСФР — вспомогательные средства функционального резервирования
ГАП — гибкое автоматизированное производство
ГПК — гибкий производственный комплекс
ГПС — гибкая производственная система
ГПМ — гибкий производственный модуль
ГР — гибридное резервирование
ГСА — граф-схема алгоритма
ГТИ — генератор тактовых импульсов
ДМХ — демультиплексор
ДР — детектор рассогласования
ДШМО — дешифратор микроопераций
ДЦП — дискретно-цифровой преобразователь
ЗУ — запоминающее устройство, память
ЗУП — память программ
ЗЭ — запоминающий элемент
ЗЯ — запоминающая ячейка
ИАВО — индивидуальный АВО
ИЧ — интерфейсная часть
КА — корректирующий автомат
КВУ — контроллер внешнего устройства
КП — комплекс программ
КР — комбинированное резервирование
КТС — комплекс технических средств
КТМС — комплекс телемеханических средств
ЛГТВ — локальный генератор тактовых воздействий
ЛСА — логические схемы алгоритмов
ЛССВ — локальные средства самовосстановления
ЛСФД — локальные средства функционального диагностирования
МА — микропрограммный автомат
МВО — мультиплексорный ВО
МО — математическое обеспечение
МП — микропроцессор

ММП — микропрограммируемый микропроцессор
МПУУ — микропрограммное устройство управления
МПК — микропроцессорный комплект БИС
МПС — микропроцессорные средства (системы)
МПСУ — микропроцессорные системы управления
МПЛМ — масочная ПЛМ
МР — мажоритарное резервирование
МРО — магистральный решающий орган
МФМ — многофункциональный модуль
МФЛМ — многофункциональный логический модуль
МХ — мультиплексор
МЭ — мажоритарный элемент
МЭВМ — микроЭВМ
НМД (НМЛ) — накопитель на магнитном диске (ленте)
НС — нагруженная сеть
ОЗУ — оперативная память
ОМП — однокристальный микропроцессор
ОМЭВМ — однокристальная микроЭВМ
ОС — однородная среда
ОСПАП — операторные схемы параллельных алгоритмов с памятью
ОСТД — общие средства тестового диагностирования
ОУ — объект управления
ОФ — остаточные функции
ОЧ — операционная часть
ОШ — общая шина
ПГР — параллельно-гибридное резервирование
ПДП — прямой доступ к памяти
ПГСА — параллельные граф-схемы алгоритмов
ПЗУ — постоянная память
ППЗУ — перепрограммируемая память
ПИ — программируемый интерфейс
ПИ, ПИД — пропорционально-интегральный, пропорционально-интегрально-дифференциальный законы регулирования
ПК — программируемый контроллер
ПЛК — программируемый логический контроллер
ПЛОС — перфоленточная операционная система
ПЛМ — программируемая логическая матрица
ПМ — программный модуль
ПМВ — программируемая матрица вентилей
ПМЛ — программируемая матрица логики
ПМП — память микропрограмм
ПМХ — программируемый мультиплексор
ППЛМ — перепрограммируемая ПЛМ
Пар. МАВО (Пос. МАВО) — параллельный (последовательный) магистральный АВО
ПП — плата памяти
ПР — постоянное резервирование
ПРК — программируемый регулирующий контроллер
ПО — программное обеспечение
ПСВ — подсистема самовосстановления
ПСД — подсистема самодиагностики
РБ — резервируемый блок
РЗ — резервирование замещением
РгАМК — регистр адреса микрокоманд
РгМК — регистр микрокоманд
РКС — релейно-контактная схема
РМВ — реальный масштаб времени
РМК — ремиконт (регулирующий микропроцессорный контроллер)
РОН — регистры общего назначения
РП — регистр признаков
САПР — система автоматизированного проектирования
СБИС — сверхБИС
СБФ — система булевых функций
СВА — схема вычисления адреса
СВГ — система взаимосвязанных графов
СВК — схемы встроенного контроля
СВМПСУ — самовосстанавливающаяся МПСУ
СИС — средняя интегральная схема
СЛУ — система логического управления
СОН — система обеспечения надежности

СР — скользящее резервирование

СЧПУ — система числового программного управления

ТКК — типовая конфигурация контроллера

УА — управляющий автомат

УАИ — устройство анализа и индикации

УВВ — устройство ввода — вывода

УВС — указатель вершины стека

УОИО — устройство обнаружения и исправления ошибок

УПД — устройство прямого доступа

УП — управляющая память

УС — управляющее слово, указатель стека

УСО — устройство связи с объектом

УУ — устройство управления

ФКР — формирователь контрольных разрядов

ФР — фиксированное резервирование

ФУ — функциональное устройство

ЦАП — цифро-аналоговый преобразователь

ЦДП — цифро-дискретный преобразователь

ЦП — центральный процессор

ЦУА — цифровой управляющий автомат

ЧПУ — числовое программное управление

ША — шина адреса

ШД — шина данных

ШУ — шина управления

ЭА — электроавтоматика

ЭК — элемент коммутации

ЭНОЗУ — энергонезависимое ОЗУ

ЭП — элемент памяти

ВВЕДЕНИЕ

Ускорение научно-технического прогресса тесно связано с автоматизацией и комплексной механизацией производства, расширением номенклатуры приборов, оборудования и средств автоматизации, созданием разнообразных систем автоматического управления и контроля на базе новейших средних (СИС) и больших (БИС) интегральных схем, микропроцессоров (МП), микропроцессорных комплектов больших интегральных схем (МПК БИС) и микроЭВМ (МЭВМ).

Одной из наиболее обширных сфер применения СИС и БИС, МП и МЭВМ является область построения цифровых управляющих автоматов (ЦУА) и систем логического управления (СЛУ) для дискретной промышленной автоматики (электроавтоматика, станки с ЧПУ, обрабатывающие центры, манипуляторы и др.). Характерным признаком структуры этих систем является применение: а) двоичных датчиков и исполнительных механизмов в качестве источников входных и приемников выходных сигналов; б) языков логических функций и конечных автоматов для описания алгоритмов функционирования данных систем.

Проектирование СЛУ и ЦУА традиционно разбивается на архитектурное (или системное), логическое и техническое [22, 27, 44].

Системное проектирование включает в себя выбор архитектуры проектируемого автомата (совокупности составных блоков, их входов и выходов, связей между блоками), а также проверку правильности взаимодействия выбранных блоков между собой и с внешней средой в различных режимах его работы. Критерии оптимальности архитектуры автомата могут быть весьма различными и зависят от его типа и области применения.

Логическое проектирование предусматривает разработку схем автомата в целом и его блоков, с той или иной степенью подробности описывающих их структуру. Конечным результатом логического проектирования является разработка функциональных схем.

На этапе технического проектирования производятся компоновка функциональных элементов в модули, их размещение в конструктивах, трассировка печатных плат и др.

Наиболее сложен и ответствен этап логического проектирования СЛУ, на котором производятся: а) описание архитектуры и алгоритма функционирования автомата на одном из «первичных» языков, применяемых в различных отраслях промышленности, учиты-

вающих специфику задания на проектирование и сложившегося языка проектировщиков, с последующим преобразованием их в базовые и автоматные языки, обеспечивающие различные формальные преобразования; б) внутри- и межблочные преобразования автомата с целью оптимизации используемых ресурсов памяти, композиции и декомпозиции блоков и др.; в) реализация памяти и комбинационной части; г) обеспечение надежности автомата, его устойчивости и безотказности.

Появление микропроцессорных средств обеспечило гибкую программную и программно-аппаратную реализацию алгоритмов функционирования СЛУ из-за модификации программ и незначительных изменений в интерфейсах, а также различной пространственно-временной организации вычислительных процессов, отвечающей заданным критериям качества.

Темпы выпуска объема и номенклатуры микропроцессорных средств стремительно возрастают. Появляются и активно используются на практике 16- и 32-разрядные МП и МЭВМ, программируемые логические и регулирующие контроллеры и системы ЧПУ, специализированные и универсальные контроллеры ввода — вывода и др. Все это требует от разработчиков глубоких знаний о специфике функционирования имеющихся и перспективных микропроцессорных средств, о таком их применении, которое обеспечивает эффективное и надежное функционирование объектов управления различного назначения в реальных производственных условиях.

Несмотря на относительно высокий уровень надежности БИС, МП и МЭВМ, этот уровень оказывается недостаточным при создании ответственных систем управления, обеспечивающих безотказное правильное функционирование при ошибках в программном обеспечении, при сбоях или при выходе отдельных компонентов систем из строя и предотвращающих тем самым экономический ущерб или угрозу здоровью и жизни человека.

Разработка высокопроизводительных и надежных структур СЛУ на микропроцессорных средствах предусматривает объединение отдельных МП и МЭВМ в единую систему на принципах децентрализованного управления, построения избыточных модульных структур, более живучих и гибких в эксплуатации и обслуживании при сравнительно простом программном обеспечении.

В пособии рассматривается комплекс вопросов, связанных с конструктивно-технологическими особенностями микропроцессорной элементной базы, проблемами рационального выбора типов МП и МЭВМ, проектирования структур МПСУ, аппаратной и программной реализации требуемых алгоритмов и классов систем, обеспечения высокого качества их функционирования и реального применения на практике.

ГЛАВА 1
ЗАДАЧИ, МЕТОДЫ И СРЕДСТВА ПРОЕКТИРОВАНИЯ СИСТЕМ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ

Среди разновидностей систем управления системы логического управления занимают важное место. К их числу относятся ЭВМ, системы связи и искусственного интеллекта, устройства промышленной автоматики, числового программного управления (ЧПУ) и др. Проектирование систем логического управления связывается с решением двух взаимосвязанных задач: выбором компактного описания алгоритма функционирования системы на одном из имеющихся языков и эффективной (по каким-либо критериям) реализации этого описания программным и (или) аппаратным путем с удовлетворением предъявляемых к системе требований по стоимости, быстродействию, надежности, энергопотреблению и т. д.

§ 1.1. ОСНОВНЫЕ ПРИНЦИПЫ УПРАВЛЕНИЯ И СПОСОБЫ ПОСТРОЕНИЯ СТРУКТУР СИСТЕМ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ

Системы логического управления характеризуются, как правило, большим числом входов, на которые поступают внешние воздействия и сигналы с объекта управления, большим числом выходов, с которых на объект управления поступают управляющие воздействия, а также сложностью реализуемых в СЛУ алгоритмов. Для их описания широко применяются классические автоматные языки и многочисленные языки, учитывающие особенности функционирования реальных систем управления, их будущих структурных или схемных реализаций и другие свойства.

В общем случае систему логического управления можно представить композицией (рис. 1.1, *а*) управляющего устройства (УУ), или управляющего автомата (УА), и объекта управления (ОУ). Для осуществления функций контроля и управления заданным объектом управления применяется ряд датчиков (сигнализаторов), обеспечивающих выработку для УУ сигналов о состоянии ОУ, и исполнительных механизмов, обеспечивающих прием от УА сигналов управления, или управляющих команд.

Управляющий автомат реализует алгоритм функционирования системы управления, определяющий последовательность шагов (тех или иных операций) по управлению конкретным объектом.

Во время своей работы УА в соответствии с заданным алгоритмом функционирования вырабатывает последовательность сигналов управления, воздействующих на ОУ. Применительно к ЭВМ, например, система УА—ОУ может быть представлена моделью в виде управляющего и операционного автоматов, причем последний из них соответствует ОУ.

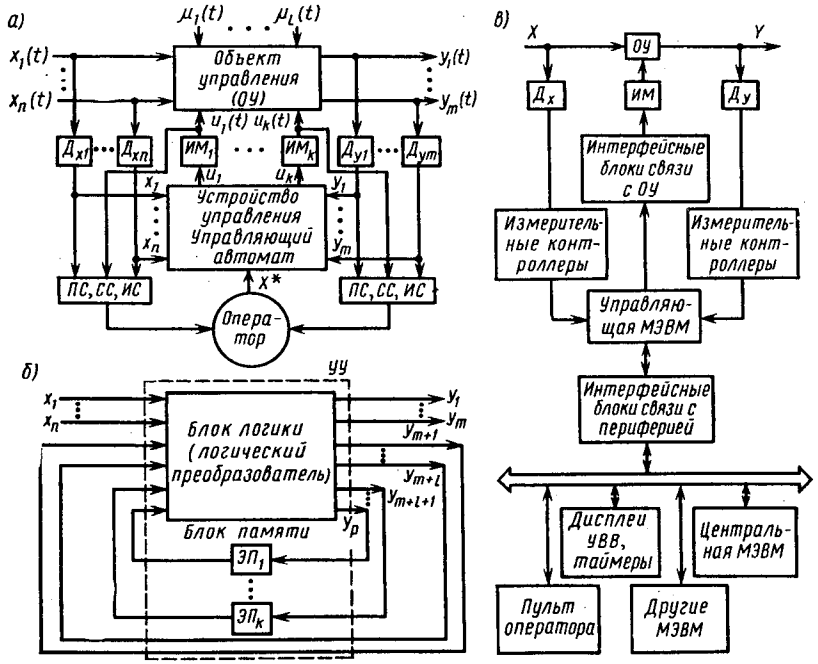


Рис. 1.1. Структура системы управления:

а — общий вид; *б* — структура управляющего автомата; *в* — структура системы управления на микроЭВМ; $X=\{x_1, ..., x_n\}$ — множество входных контролируемых воздействий; $Y=\{y_1, ..., y_m\}$ — множество выходных контролируемых воздействий; $U=\{u_1, ..., u_m\}$ — множество управляющих воздействий; $M=\{\mu_1(t), ..., \mu_l(t)\}$ — множество неконтролируемых внешних возмущений; X^* — цель управления; $Д_{x_1}, ..., Д_{x_n}$ ($Д_{y_1}, ..., Д_{y_m}$ —датчики и преобразователи входных (выходных) параметров ОУ; $ИМ_1, ..., ИМ_k$ — исполнительные механизмы, органы, устройства; ПС, СС, ИС — преобразовательные, сигнализирующие, измерительные средства; УВВ — устройства ввода — вывода

Для большинства автоматизируемых объектов характерны цикличность, нестационарность, независимость протекающих в них процессов, а также переменность технологии и режимов работы технологического оборудования. Цикличность процессов, режимов пуска, останова и автоматической работы технологического оборудования обусловливает применение функций логического управле-

ния, нестационарность обусловливает применение адаптивного управления, переменность технологии обусловливает применение гибкой перестройки системы управления.

В зависимости от способа построения УА и топологии ОУ различают сосредоточенные, распределенные или смешанные системы с централизованным, децентрализованным или комбинированным принципом управления. В общем случае управляющий автомат реализуется структурой из блока логики (логического преобразователя, или комбинационной схемы) и блока памяти (запоминающего устройства), из которых получаются все другие разновидности схем (рис. 1.1, *б*). Блок памяти в отличие от логического преобразователя реализует временны́е логические функции (систему временны́х логических функций) на основе триггеров, задержек и различного рода ЗУ [22]. Большинство триггеров (синхронные, асинхронные, полисинхронные, стробируемые, с управлением по фронту или срезу и др.), а также более сложные устройства (например, регистры и счетчики) могут быть реализованы на основе БИС ЗУ. Это позволяет существенно повышать регулярность структуры УА, его быстродействие, надежность, диагностируемость, контролепригодность.

Математической моделью СЛУ является автомат A вида

$$A=\langle \mathbf{X},\ \mathbf{Y},\ \mathbf{Q},\ \delta,\ w,\ q_0\rangle,$$

где $\mathbf{X}=\{x_1, ..., x_n\}$ — множество входных сигналов; $\mathbf{Y}=\{y_1, ..., y_m\}$ — множество выходных сигналов; $\mathbf{Q}=\{q_0, q_1, ..., q_l\}$ — множество состояний; $\delta: \mathbf{X}\times\mathbf{Q}\to\mathbf{Q}$ — функция переходов, ставящая некоторым парам «состояние — входной сигнал» (q_i, x_j) состояние автомата $q_s=\delta(q_i, x_j)$, $q_s\in\mathbf{Q}$; w: $\mathbf{X}\times\mathbf{Q}\to\mathbf{Y}$ — функция выходов, ставящая некоторым парам «состояние — входной сигнал» (q_i, x_j) выходные сигналы автомата $w_s=w(q_i, x_j)$; q_0 — начальное состояние автомата.

Понятие состояния в определении автомата введено в связи с необходимостью описания поведения систем, выходы которых зависят не только от состояния входов в данный момент времени, но и от некоторой предыстории (от ранее поступавших на входы системы сигналов).

Рассмотрим специфику построения и функций, реализуемых СЛУ, на примере управления технологическими процессами промышленного производства (АСУ ТП).

В целом система управления технологическими процессами должна обеспечивать:

регулирование параметров, программное управление заданиями регуляторов и др.;

логическое управление оборудованием и циклическими процессами;

контроль правильности ввода—вывода данных, их преобразование с осуществлением функций сигнализации, индикации и регистрации.

Алгоритмы логического управления встречаются в АСУ ТП в задачах:

— контроля хода технологического процесса в процессе первичной обработки данных о работе технологического агрегата или узла по показаниям датчиков; выявления фактов отступления от технологических режимов и заданных параметров;

— управления технологическими процессами при двухпозиционных агрегатах и механизмах по логическим законам с изменением уставок, аварийных ограничений, дистанционного включения и отключения исполнительных устройств; автоматизированной реализации технологических регламентов в течение определенного интервала времени; индикации данных и логических условий в удобной для оператора форме; обработки данных, полученных от периферийных устройств, и др.

Реализация перечисленных задач управления АСУ ТП может быть обеспечена рядом технических средств, в том числе:

а) в централизованных АСУ ТП — пультом управления с УВК, содержащим микроЭВМ, устройства внешней памяти, устройства и линии связи с объектом, аналоговые и цифровые регуляторы и устройства ввода—вывода данных, причем данную аппаратуру располагают обычно не у технологических объектов, а в отдельном помещении; датчики и исполнительные устройства располагаются непосредственно на объекте управления;

б) в децентрализованных АСУ ТП — группой локальных микроЭВМ или микроконтроллеров, расположенных по месту, у технологических объектов, с реализацией распределенных функций контроля хода технологического процесса или управления тем или иным технологическим узлом или объектом.

В общем случае с помощью микропроцессорной техники могут быть реализованы все перечисленные функции АСУ ТП.

Появление микропроцессорных средств сделало целесообразным переход от централизованных систем управления и контроля к децентрализованным распределенным системам, обеспечивающим более высокую гибкость решения задач, надежность, живучесть благодаря возможности реконфигурации и др. Необходимость децентрализации во многом определяется логикой развития самих объектов управления, однако до появления микропроцессорных средств обработки и передачи данных практическая реализация децентрализованных систем была ограничена.

Построение распределенных систем контроля, регулирования, управления и обработки данных базируется на принципах функциональной и топологической децентрализации. *Функциональная* децентрализация системы управления подразумевает разбиение процесса на такие подпроцессы, которые слабо взаимосвязаны друг с

другом. *Топологическая* децентрализация предполагает пространственное распределение датчиков, регуляторов и других исполнительных устройств, а также МПСУ (микроЭВМ, микроконтроллеров) и другой аппаратуры обработки данных. Как правило, оптимальная функциональная децентрализация не совпадает с топологическим оптимумом, в связи с чем приходится принимать различные компромиссные решения.

Проектирование распределенных МПСУ можно вести на основе иерархического подхода с описанием системы на определенном уровне абстракции с выделением функциональной, структурной и параметрической моделей $<\mathbf{F}, \mathbf{S}, \mathbf{P}>$. *Функциональная* модель **F** задается четверкой $<\mathbf{Y}, \mathbf{D}, \Sigma, \mathbf{И}>$, где **Y** — управляющая сеть (например, логическая сеть Петри), при помощи которой задаются функции системы; **D** — информационная сеть, являющаяся обобщением информационного графа; Σ — функция назначения, задающая отображение множества переходов из **Y** в множество операций из **D**; **И** — интерпретация, задающая семантику операций и первоначальное содержимое памяти. *Структурная* модель **S** представляет собой ориентированную сеть, задающую распределение операций модели **F** по реальным подсистемам системы, рассматриваемой на определенном уровне абстракции. *Параметрическая* модель **P** является вектором параметров, характеризующим определенный аспект поведения и структуры системы.

При проектировании распределенных МПСУ наиболее важен этап структурного синтеза с введением структуры в определенную функциональную модель. Он может осуществляться либо подходом «сверху» с декомпозицией системы, описанной при помощи функциональной модели, на подсистемы, либо подходом «снизу», заключающимся в назначении функций по элементам данной реальной структуры, описанной при помощи структурной модели, и определении функциональной модели. Каждый из этих подходов имеет свою наиболее целесообразную сферу применения, и оба активно применяются при проектировании.

В целом выбор конкретных технических комплексов микропроцессорных систем (МП, МЭВМ или микроконтроллеров) для АСУ ТП определяется классом реализуемых алгоритмов и требованиями к динамическим характеристикам системы и представляет собой сложную плохо формализуемую задачу, связанную с учетом большого числа факторов.

На рис. 1.1, *в* приведена обобщенная структура системы управления с использованием микропроцессорных средств и дополнительных средств регулирования, периферийного оборудования, вычислительной техники и др. УУ в данной системе включает в себя управляющую микроЭВМ или вычислительную систему в общем случае. Алгоритмы управления и регулирования технологических процессов реализуются в виде программ, хранящихся в памяти микроЭВМ. Интерфейсные блоки обеспечивают связь с объектом

управления и разнообразным периферийным оборудованием — пультом оператора, временными средствами (таймерами), дисплеями, автоматизированными рабочими местами, другими микроЭВМ и ЭВМ более высокого ранга и др. Измерительные контроллеры преобразуют и выдают в микроЭВМ показания датчиков о состоянии ОУ и внешней среды. В состав интерфейсных блоков связи и измерительных контроллеров могут входить свои микропроцессоры, позволяющие выполнять трудоемкие операции ввода—вывода и предварительной обработки информации, выдавая в ЭВМ более высокого ранга интегрированную информацию.

При достаточно сложных объектах управления они разбиваются на отдельные функциональные части, для каждой из которых создается своя микропроцессорная система управления, аналогичная приведенной выше и обеспечивающая связь с другими микроЭВМ или с центральной ЭВМ при решении поставленных перед нею задач. Центральная ЭВМ или микроЭВМ в такой распределенной системе обычно решает задачи координации взаимодействия локальных управляющих микроЭВМ по управлению сложным объектом, диспетчеризации, оперативно-календарного планирования и управления и др.

Описание проектируемых систем и их алгоритмов функционирования осуществляется с помощью разнообразных языков проектирования. Для систем логического управления выбор языка проектирования имеет большое значение, позволяя переходить от неформализованного описания поведения СЛУ к формальному заданию, оптимизировать ее структуру, вести конструктивный диалог с технологами и т. д.

§ 1.2. МЕТОДЫ И СРЕДСТВА ПРОЕКТИРОВАНИЯ СИСТЕМ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ

Проектирование систем логического управления разнообразными объектами промышленной автоматики, обеспечивающих их эффективную согласованную работу в соответствии с заданным алгоритмом функционирования, всегда считалось одной из важнейших задач автоматизации производственных процессов.

Как указывалось выше, процесс создания ЦУА и СЛУ подразделяется на ряд этапов (архитектурное, логическое, техническое). Важнейший из них — этап логического проектирования или синтеза, на котором по формальному заданию условий работы СЛУ находится логическая структура (сеть или схема логических элементов), удовлетворяющая этим условиям с учетом заданных критериев качества функционирования. При оптимизации СЛУ испсльзуются комбинаторные процедуры, связанные с понятием «дерево поиска» и обширным перебором промежуточных решений, их оцениванием, отбраковкой одних и выбором других, определяющих перспективные направления последующего поиска. Данные комби-

наторные процедуры весьма трудоемки и требуют применения современных ЭВМ [29].

Развитие новых методов синтеза ЦУА и СЛУ с использованием различной элементной базы ведется в следующих направлениях:

1) совершенствования методов структурного синтеза с применением: а) микросхем средней и малой степени интеграции, а также заказных и полузаказных БИС для реализации асинхронных и синхронных СЛУ и ЦУА на основе «жесткой» аппаратной логики; б) проблемно-ориентированных настраиваемых БИС; однородных вычислительных сред и структур; ПЛМ и ЗУ (ПЗУ, ППЗУ, ОЗУ, РПЗУ); нескоммутированных БИС логики и базовых матричных кристаллов;

2) совершенствования программно-вычислительных методов синтеза микропроцессорных СЛУ с применением: а) однопроцессорных МЭВМ, МПК БИС и программируемых логических контроллеров для программной реализации логических алгоритмов на базе методов микропрограммирования, резидентных интерпретаторов, предварительного преобразования исходных алгоритмов, непосредственного программирования; б) сосредоточенных и распределенных мультимикропроцессорных и мультимикромашинных средств для аппаратно-программной реализации методов распараллеливания и распределения алгоритмов логического управления.

В развитие методов структурного синтеза синхронных и асинхронных ЦУА на основе жесткой аппаратной логики большой вклад внесли советские и зарубежные ученые М. А. Гаврилов, В. М. Глушков, В. А. Горбатов, А. Д. Закревский, А. В. Каляев, В. Г. Лазарев, В. И. Варшавский, С. И. Баранов, Д. А. Поспелов, А. Н. Мелихов, Э. А. Якубайтис, С. В. Яблонский, К. Шеннон, А. Гилл и др.

Наиболее актуальным и перспективным направлением в области синтеза ЦУА на «жесткой» логике является разработка методов автоматизации проектирования внутренней структуры заказных БИС, ЦУА и устройств управления для высокопроизводительных ЭВМ.

Особенности синтеза ЦУА на основе проблемно-ориентированных настраиваемых БИС состоят в том, что средством реализации алгоритмов функционирования проектируемых систем являются БИС, которые могут настраиваться на реализацию любого конкретного логического алгоритма из заданного класса с помощью специальных дополнительных технологических или электрических воздействий на эти БИС. Процесс настройки таких БИС заключается в создании или, напротив, в разрушении некоторых связей в их исходной регулярной внутренней структуре.

Большой вклад в развитие методов синтеза СЛУ и ЦУА на однородных вычислительных средах и структурах внесли Э. В. Евреинов, Ю. Г. Косарев, О. Л. Бандман, А. В. Каляев, В. Г. Лазарев, И. В. Прангишвили, В. А. Скоробогатов и др. Однако на сегодня

основное практическое применение и развитие однородные вычислительные среды и структуры получили в основном в области построения высокопроизводительных многопроцессорных вычислительных систем с программируемой архитектурой [12, 13] и управляющих вычислительных комплексов типа ПС2000, ПС3000 [17].

К другим типам программно-настраиваемых БИС относятся БИС программируемых логических матриц (ПЛМ), ПЗУ, ППЗУ и некоторые другие. Методы синтеза ЦУА и СЛУ на основе этих разновидностей БИС глубоко разработаны в трудах С. И. Баранова, А. Д. Закревского, Б. М. Кагана, В. А. Склярова, Э. А. Якубайтиса, Е. И. Пупырева, Ю. И. Попова и др.

Менее разработаны в теоретическом плане методы синтеза СЛУ и ЦУА на БИС нескоммутированной логики, внутри которых содержится определенный набор готовых, но электрически не скоммутированных между собой типовых логических, триггерных, усилительных и других ячеек. Настройка такого вида проблемно-ориентированных БИС на реализацию конкретного ЦУА заключается в нанесении на поверхность БИС необходимого числа дополнительных слоев металлизации с целью получения соединений между имеющимися в БИС типовыми схемными фрагментами в соответствии с ранее разработанной функциональной схемой синтезируемого устройства.

Появление микропроцессорной элементной базы обусловило переход от методов структурной теории автоматов к программно-вычислительным методам, широко используемым при синтезе микропроцессорных систем логического управления. В развитие программно-вычислительных методов реализации алгоритмов логического управления на основе микропроцессорных средств большой вклад внесли работы Л. Г. Алексенко, Л. А. Амбарцумяна, Е. П. Балашова, В. В. Девяткова, В. А. Горбатова, Г. И. Иванова, Б. М. Кагана, О. П. Кузнецова, Д. В. Пузанкова, А. Н. Мелихова, В. Г. Першеева, В. Б. Смолова, В. В. Сташина, А. Тейза, Р. П. Чапцова, М. И. Шамрова и др.

Основная задача исследований в этом направлении состоит в разработке методов смешанной структурно-вычислительной реализации алгоритмов логического управления, оптимизации и автоматизации синтеза однопроцессорных СЛУ, распараллеливания алгоритмов логического управления и их мультимикропроцессорной реализации в распределенных СЛУ и др.

Проектирование СЛУ на основе новейших БИС, СИС и микропроцессорных средств обусловливает необходимость разработки новых подходов к синтезу и оптимизации реализуемых структур систем с учетом новых критериев качества функционирования, рационального соотношения между программной и аппаратурной реализацией, соблюдения множества технических требований и ограничений, присущих условиям конкретного применения и эксплуатации.

Одним из таких подходов к проектированию СЛУ является направление обобщенных типовых реализаций (аппликаций), позволяющих значительно повысить гибкость и надежность работы си-

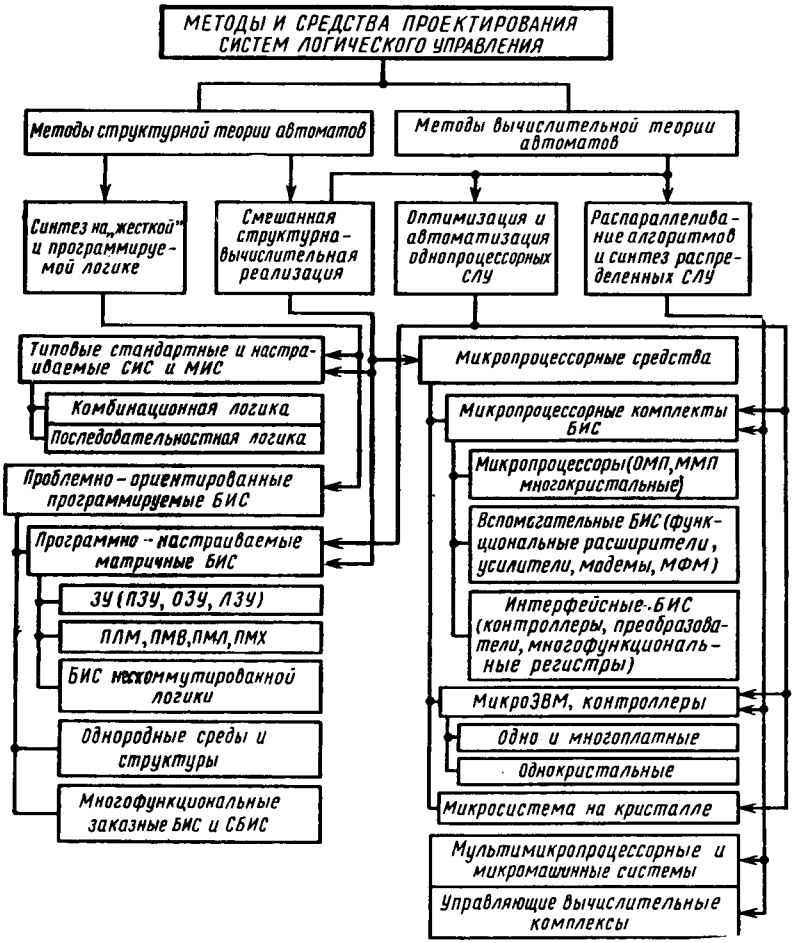


Рис. 1.2. Классификация основных методов и средств, применяемых при проектировании систем логического управления

стем, их диагностируемость, контроле- и ремонтопригодность и др. В развитие методов этого направления внесли вклад советские и зарубежные ученые А. А. Амбарцумян, В. И. Варшавский, В. Н. Захаров, Дж. Патил, А. И. Потехин, В. Г. Лазарев, В. П. Чистов, С. А. Юдицкий и др.

Применение разнообразных вышеперечисленных средств при синтезе СЛУ и ЦУА обусловливает появление новых типов струк-

тур, рационально сочетающих программные и аппаратные средства с удовлетворением многочисленным и противоречивым критериям оптимизации. При программной реализации задача оптимизации обычно заключается в экономии памяти и времени вычислений,

Таблица 1.1. **Характерные области применения ЦУА и СЛУ**

№ п/п	Название области применения	Основные выполняемые функции	Примеры характерных объектов автоматизации
1	Системы программного управления, в том числе системы ЧПУ	Управление по жесткой программе, предварительно записанной в память	Системы связи, коммутации. Металлообрабатывающие станки, станочные линии, кузнечно-прессовое оборудование, полиграфические машины, санитарно-технические установки
2	АСУ технологическими процессами, установками	Контроль, измерение, регистрация, логическое управление, регулирование, аварийная защита, расчет уставок регуляторов	Весовые дозаторы, печи, сортовые и проволочные станы, гибкие производственные модули механообработки, цементные заводы, бетонорастворные узлы, хлебозаводы и др.
3	АСУ гибкими участками и цехами	То же, что в п. 2, и диспетчерское управление при двухступенчатой структуре	Гибкое автоматизированное производство сборки, механической обработки, заводы-автоматы
4	Мелиорация	То же, что в п. 2, и диспетчерское управление при двух- или трехступенчатой структуре	Объекты регулирования и управления водовыпуска, насосные станции различного назначения
5	Научное приборостроение	То же, что в п. 2	Системы сбора и обработки информации, бортовая автоматика, контрольно-измерительная аппаратура
6	ЭВМ, САПР	Управление, логическая обработка и преобразование данных, организация ввода — вывода. индикация, сигнализация	Управляющие и операционные автоматы

уменьшении сложности интерпретации параллельных процессов в виде последовательных и т. д. При аппаратурной реализации задача оптимизации состоит в сокращении числа корпусов микросхем, повышении быстродействия, надежности, живучести и др.

Если используемые средства обеспечивают программную реализацию требуемых функций с обслуживанием запросов, поступающих извне в реальном масштабе времени, то возникают задачи подсчета среднего времени обслуживания, определения приоритета

и др., обычные для теории массового обслуживания. Если же эти условия не выполняются, то осуществляется перевод ряда функций на аппаратурную реализацию с учетом предъявляемых к СЛУ и ЦУА требований и ограничений.

На рис. 1.2 приведен перечень основных методов и средств для проектирования ЦУА и СЛУ для наиболее характерных областей применения (табл. 1.1). Разработчик СЛУ располагает различными средствами для их реализации и ориентируется, как правило, на использование универсальных и проблемно-ориентированных БИС (микропроцессорных комплектов, ПЛМ, ЗУ, нескоммутированных логических матриц, универсальных и специализированных настраиваемых модулей и др.), а также программируемых контроллеров, микроЭВМ или управляющих вычислительных комплексов.

Рассмотрим более подробно конструктивно-технологические особенности и функциональные возможности современной элементной базы, применяемой при проектировании СЛУ.

ГЛАВА 2

ОСОБЕННОСТИ ФУНКЦИОНИРОВАНИЯ СХЕМ СРЕДНЕЙ И БОЛЬШОЙ СТЕПЕНИ ИНТЕГРАЦИИ

При построении СЛУ широко используются разнообразные типовые СИС и БИС, которые позволяют реализовать аппаратным (схемотехническим) путем сложные функции и алгоритмы [24, 34, 35 и др.]. К числу таких СИС относятся арифметико-логические, регистровые, счетные и триггерные устройства, а также сумматоры, умножители, мультиплексоры, демультиплексоры, шифраторы, дешифраторы, компараторы, схемы сквозного переноса и др. Рассмотрим особенности их построения и функционирования.

§ 2.1. ЛОГИЧЕСКИЕ СХЕМЫ СРЕДНЕЙ СТЕПЕНИ ИНТЕГРАЦИИ

По функциональному назначению типовые СИС подразделяются на комбинационные и последовательностные схемы. К *комбинационным* схемам относятся схемы, не содержащие в своей структуре элементов памяти (триггеры). Их состояние определяется комбинацией входных сигналов и не зависит от предыдущего состояния. К их числу относятся разнообразные сумматоры (по модулю два, полусумматоры, полные одно- и многоразрядные и др.), умножители, компараторы, мультиплексоры, дешифраторы, шифраторы, преобразователи кодов и т. д.

К *последовательностным* СИС относятся регистры, счетчики, распределители импульсов и др. Они строятся на основе триггеров и имеют ту особенность, что их состояние зависит не только от сигналов, воздействующих на входы в данный момент времени, но и от

предыдущих состояний. Принципы действия и способы построения этих устройств широко рассмотрены в литературе [34, 35].

Менее известны в практике проектирования СЛУ такие комбинационные СИС, как мультиплексоры, обладающие широкими функциональными возможностями и объединяющие в себе свойства дешифратора и переключателя или селектора. В простейшем случае под мультиплексором (МХ) понимают управляемый многопозиционный переключатель или коммутатор. Из-за их способности выбора (селекции) определенного канала МХ называют еще селекторами, а иногда селекторами-мультиплексорами и др.

В общем случае мультиплексор представляет собой логическое устройство, реализованное в виде СИС, содержащее q управляющих входов $u_1, u_2, ..., u_q$, 2^q информационных входов $D_0, D_1, ..., D_{2^q-1}$, один общий или несколько раздельных прямых и (или) инверсных стробирующих входов s_k ($k=1, 2, ...$) и один или несколько прямых и (или) инверсных выходов y_i ($i=1, 2, ...$). При подаче на управляющие $u_i|i=\overline{1, q}$ входы некоторой комбинации двоичных сигналов $\tilde{u}_1\tilde{u}_2...\tilde{u}_q$ ($\tilde{u}_i \in \{0, 1\}$) и на стробирующие входы соответствующих сигналов низкого или высокого уровня ($s_k \in \{0, 1\}$) к выходу y_i мультиплексора подключается только тот информационный вход $D_j|j=\overline{0, 2^q-1}$, порядковый номер которого соответствует весу двоичной комбинации управляющих сигналов. В качестве примера на рис. 2.1 приведены условное обозначение и функциональная схема мультиплексора с тремя управляющими входами $u_i|i=\overline{1, 3}$, восемью информационными входами $D_j|j=\overline{0, 7}$, одним инверсным стробирующим входом s_1 и двумя выходами: прямым y и инверсным $\bar{y}$. Таблица состояний данного мультиплексора, поясняющая принцип его функционирования, приведена в табл. 2.1.

Мультиплексоры в корпусе СИС могут быть реализованы одиночными, сдвоенными и строенными, иметь один общий выход на 2^q входов или несколько раздельных выходов для каждой пары соответствующих входов при наличии одного или нескольких общих или раздельных стробирующих входов, управляемых уровнем логического нуля (лог. 0) или логической единицы (лог. 1).

На основе КМДП-структур можно реализовать так называемые «передающие вентили» [34]. Такие устройства являются двунаправленными, и для них не имеет значения, какой из выводов используется в качестве входа, а какой — в качестве выхода. Таким образом, при наличии мультиплексора, реализованного на базе КМДП-технологии, при использовании его выхода в качестве информационного входа, а информационных входов — в качестве выходов из мультиплексора можно получить другой логический элемент — демультиплексор (ДМХ). В общем случае *демультиплексор* реализуется на основе СИС мультиплексора с КМДП-структурой, содержит q управляющих входов $u_i|i=\overline{1, q}$, один или несколько стробирующих входов, один информационный вход D и 2^q

выходов, причем к информационному входу подключается только тот выход, порядковый номер которого (от 0 до 2^q—1) соответствует весу двоичной комбинации управляющих сигналов. Причем для отдельных типов СИС мультиплексоров — демультиплексоров информационные сигналы могут быть как цифровыми, так и аналоговыми.

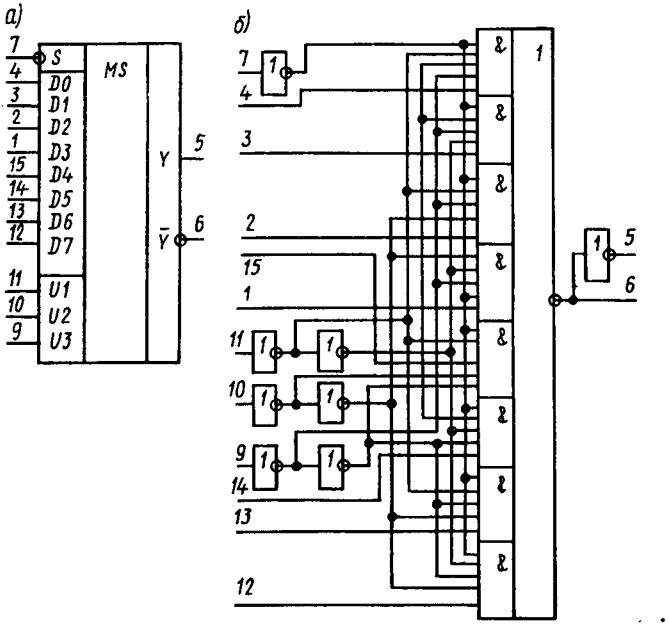


Рис. 2.1. Селектор-мультиплексор данных на 8 каналов со стробированием:
а — условное обозначение; *б* — функциональная схема

Структура одного из аналоговых МХ приведена на рис. 2.2, *а*. Здесь ключи *К1—К4* являются «передающими вентилями» с управляющим входом *С*, обращение к которым производится по конкретному адресу с помощью схем логики выбора. Такой аналоговый МХ может являться и ДМХ, т. е. сигнал может быть подан на выход и снят с избранного входа, или выполнять функции цифрового МХ—ДМХ одновременно. На рис. 2.2, *б* в качестве примера приведено условное обозначение СИС аналогового МХ с управляющими входами V_i. Если в этой СИС сигнал $V_i=0$, то входной сигнал A_i передается на выход B_i $(i=\overline{1, 4})$.

Практическая реализация цифрового МХ—ДМХ выполнена, например, на базе СИС 564КП1, 564КП2, строенного мажоритарно-мультиплексорного элемента — на базе СИС 564ИК1.

Мультиплексоры серии 155 (К155КП1, К155КП2, К155КП5, К155КП7), выполненные по ТТЛ-технологии, могут выполнять роль универсальных кодопреобразователей, реализующих булевы функции вида

Таблица 2.1. **Таблица логических состояний мультиплексора-селектора 8 каналов на один**

Входы											Выходы	
управляющие			информационные								строб	прямой	инверсный
a_1	a_2	a_3	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	s	y	$\bar{y}$
×	×	×	×	×	×	×	×	×	×	×	В	Н	В
Н	Н	Н	В	×	×	×	×	×	×	×	Н	В	Н
Н	Н	Н	Н	×	×	×	×	×	×	×	Н	Н	В
В	Н	Н	×	В	×	×	×	×	×	×	Н	В	Н
В	Н	Н	×	Н	×	×	×	×	×	×	Н	Н	В
Н	В	Н	×	×	В	×	×	×	×	×	Н	В	Н
Н	В	Н	×	×	Н	×	×	×	×	×	Н	Н	В
В	В	Н	×	×	×	В	×	×	×	×	Н	В	Н
В	В	Н	×	×	×	Н	×	×	×	×	Н	Н	В
Н	Н	В	×	×	×	×	В	×	×	×	Н	В	Н
Н	Н	В	×	×	×	×	Н	×	×	×	Н	Н	В
В	Н	В	×	×	×	×	×	В	×	×	Н	В	Н
В	Н	В	×	×	×	×	×	Н	×	×	Н	Н	В
Н	В	В	×	×	×	×	×	×	В	×	Н	В	Н
Н	В	В	×	×	×	×	×	×	Н	×	Н	Н	В
В	В	В	×	×	×	×	×	×	×	В	Н	В	Н
В	В	В	×	×	×	×	×	×	×	Н	Н	Н	В

Примечание. В — высокий уровень; Н — низкий уровень; × — безразличное состояние.

$$y=\bar{x}_s\tilde{x}_l \bigvee_{l=0}^{2^q-1} \bigwedge_{j=1}^{q} \tilde{x}_1 \ldots \tilde{x}_j \ldots \tilde{x}_q. \qquad (1.1)$$

где $\bigvee_{l=0}^{2^q-1} \bigwedge_{j=1}^{q} \tilde{x}_1, \ldots, \tilde{x}_j, \ldots, \tilde{x}_q$ — совершенная дизъюнктивно-нормальная форма (СДНФ) булевой функции от q переменных ($q=2, 3, 4$); $\bar{x}_s$ — стробирующий сигнал; $\tilde{x}_i \in \{0, 1\}$ — любой сигнал, подключаемый к входу D_j.

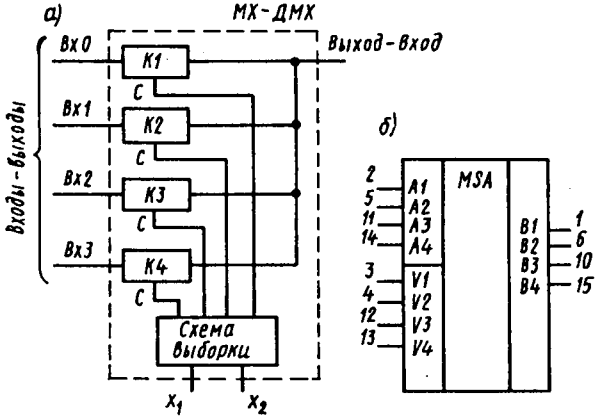


Рис. 2.2. Цифровой (аналоговый) мультиплексор-демультиплексор (МХ—ДМХ):

а — структура МХ—ДМХ; *б* — условное обозначение СИС типа К590КП12; Вх0—Вх3 — цифровые (аналоговые) входы — выходы; x_1, x_2 — цифровые адресные входы; *К1—К4* — ключи (передающие вентили); *А1 ... А4* (*В1 ... В4*) — аналоговые входы (выходы); *V1 ... V4* — управляющие цифровые входы; *С* — управляющий вход

Более сложные логические функции реализуют СИС, сочетающие в себе свойства мультиплексоров, демультиплексоров и дешифраторов (например, К155ИД4, К555ИД4, К555ИД7П). С помощью селектора К531КП11П, например, можно реализовать функцию четырехполюсного двухпозиционного переключателя логических сигналов двух каналов в один с тремя состояниями. Такие схемы во многом предпочтительнее механических переключателей, так как коммутация всех каналов производится одновременно и практически мгновенно. Отметим, что на аналогичном принципе работают, например, коммутаторы К555КП1, К555КП12 и др.

Перечень наиболее распространенных разновидностей СИС мультиплексоров с указанием их конструктивных параметров, а также условные обозначения, структурные схемы и таблицы состояний сведены для удобства ознакомления в приложении I.

В целом СИС мультиплексоров и их разновидности могут эффективно применяться при аппаратной реализации определенных

классов систем булевых функций и цифровых автоматов, как это проиллюстрировано ниже в § 3.1.

В практике проектирования логических устройств кроме типовых СИС активно применяются специализированные многофункциональные логические модули (МФЛМ), различающиеся используемой элементной базой, числом настроечных входов, способом настройки и другими особенностями. Элементный состав, специфика построения структуры МФЛМ и синтеза логических устройств на их основе разнообразны и ориентируются на реализацию определенных классов задач и функций. Более подробно эти вопросы рассмотрены, например, в [16, 26].

Другим направлением реализации МФЛМ с перестраиваемой структурой являются *однородные вычислительные среды* (ОС), отличающиеся большим множеством типов элементов, способов их настройки, сложностью реализуемых функций и другими признаками.

Однородные среды, как правило, реализуются в виде плоской композиции идентичных ячеек, настраиваемых на выполнение определенных функций с помощью настроечной памяти, запоминающей информацию, поступающую к ней извне во время процесса настройки. Поведение ячейки ОС определяется ее внутренним состоянием, состоянием соседних ячеек и состоянием настроечной памяти. Обработка информации в ОС осуществляется реализуемыми в ней с помощью программной настройки схемами.

Таким образом, в ОС происходит сближение программных и аппаратных методов реализации алгоритмов. Более подробно эти вопросы освещены в ряде работ, например в [12, 13, 21, 25, 28].

§ 2.2. ЛОГИЧЕСКИЕ СХЕМЫ БОЛЬШОЙ СТЕПЕНИ ИНТЕГРАЦИИ

При проектировании систем логического управления на микропроцессорных средствах широко применяются матричные схемы серийных БИС ЗУ, а также разнообразные программируемые БИС — матрицы вентилей, логики и др. Рассмотрим их более подробно.

БИС запоминающих устройств. Из всего многообразия современных ЗУ на практике наиболее широко применяются полупроводниковые БИС ЗУ, успешно конкурирующие с другими типами ЗУ по таким характеристикам, как информационная емкость, быстродействие, потребляемая мощность, надежность, стоимость и др. Например, в МПСУ 70—80% их объема и стоимости занимают ЗУ. Конкретный выбор элементной базы для реализации БИС ЗУ определяется приоритетом тех или иных требований, соответствующих реально достигнутому технологическому уровню [24].

По типу запоминающих элементов (ЗЭ), построенных на основе биполярного или полевого транзистора, БИС ЗУ делятся на два больших класса: а) биполярные структуры, имеющие более высокое быстродействие и использующие транзисторно-транзисторные (ТТЛ), эмиттерно-связанные (ЭСЛ), инжекционные ($И^2Л$, $И^2ЛШ$) и тиристорные ЗЭ; б) МДП-структуры (МОП, КМОП), обладающие большой и сверхбольшой информационной емкостью и меньшим быстродействием по сравнению с биполярными структурами. Несколько отличаются от них ПЗС- и ЦМД-структуры (приборы с зарядовой связью и на цилиндрических магнитных доменах), использующие наряду с полевым эффектом принцип зарядовой связи.

В составе МПСУ могут быть выделены внутренние и внешние ЗУ. *Внутренние ЗУ* обеспечивают хранение данных и программ, используемых при выполнении текущего вычислительного процесса; *внешние ЗУ* — долговременное хранение крупных массивов информации, передаваемых во внутренние ЗУ по мере необходимости и в требуемом объеме. Внутренние ЗУ подразделяются на постоянные, оперативные и логические.

Постоянные ЗУ (ПЗУ) служат, как правило, для хранения постоянных или редко изменяющихся массивов информации (константы, табличные значения различных функций, таблицы распределения данных, типовые подпрограммы вычислений, программы операционных систем, преобразователи кодов, генераторы символов, функций, тестов и др.). Основными требованиями к ПЗУ являются неразрушающее считывание, высокая надежность и энергонезависимость хранения информации. БИС ПЗУ выпускают трех типов (см. приложение, табл. П.II.1): *масочные ПЗУ (ПЗУМ)*, в которых однократная запись информации осуществляется при изготовлении БИС путем нанесения фиксированного рисунка межсоединений, определяемого маской (фотошаблоном); *программируемые ПЗУ (ППЗУ)*, требуемая информация в которые записывается только один раз самим потребителем, например путем выжигания диодов, плавких вставок; *репрограммируемые* или *перепрограммируемые ПЗУ (РПЗУ)*, в которых существует возможность многократной перезаписи информации путем стирания ее электрическими сигналами (ЭППЗУ) или ультрафиолетовым светом (УФПЗУ).

Оперативные ЗУ (ОЗУ) (см. приложение, табл. П.II.2) предназначены для хранения переменной информации, имеют практически одинаковое быстродействие при считывании и записи. В них происходит разрушение хранимой в ОЗУ информации при отключении напряжения питания, что является существенным недостатком при использовании конкретных типов ОЗУ в специализированных системах. Обобщенная типовая структура БИС ЗУ, включающая основной состав блоков для разновидностей ПЗУ и ОЗУ, приведена на рис. 2.3.

Логические ЗУ (ЛЗУ) кроме хранения информации выполняют некоторые логические и арифметические операции и выпу-

скаются в виде *ассоциативных ЗУ* (АЗУ), в которых логические задачи решаются при изменении структуры ЗЭ памяти и внутренних связей, и *многофункциональных ЗУ (МФЗУ)*, в которых логические и арифметические операции решаются при сохранении основной структуры ЗЭ памяти и внутренних связей ЗУ в результате использования только добавочных связей в разрядных цепях [25]. Объединение в АЗУ логических запоминающих функций, например, в 10—100 раз ускоряет поиск, упорядочение и обработку информации, в 2—10 раз экономит ресурсы памяти по сравнению с обычными ЗУ.

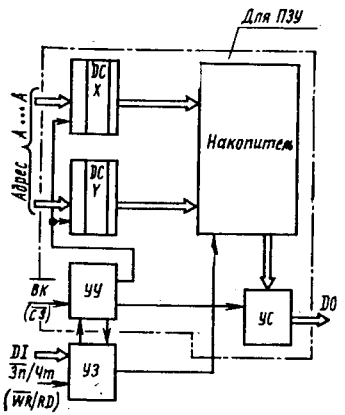


Рис. 2.3. Типовая структура БИС ЗУ (ОЗУ, ППЗУ, РПЗУ): УУ — устройство управления, УЗ (УС) — устройство записи (считывания); DI — DO — входная (выходная) информация; ВК (CS) — сигнал выборки кристалла; Зп/Чт (WR/RD) — сигнал управления записью / считыванием

Полупроводниковые БИС ЗУ (ОЗУ, ПЗУ, ППЗУ, РПЗУ), обычно состоят (рис. 2.3) из накопителя и схем электронного обрамления. Накопитель представляет собой квадратную или прямоугольную матрицу запоминающих элементов (ЗЭ), определяющую информационную емкость БИС ЗУ и занимающую примерно 60—70% всей площади кристалла. В состав схем электронного обрамления ЗУ входят:

адресная часть (строчные и столбцовые дешифраторы), обеспечивающая выбор одного или нескольких ЗЭ в накопителе в соответствии с поданной на адресные входы комбинацией двоичных сигналов;

числовая часть (усилители считывания — записи), обеспечивающая передачу информации от входа БИС ЗУ к выбранному ЗЭ при записи (в ОЗУ) и передачу информации от ЗЭ к выходу БИС ЗУ при считывании (в ОЗУ и ПЗУ);

блок местного управления (БМУ), координирующий работу всех узлов в режимах хранения, записи, считывания, регенерации (динамические ОЗУ) и стирания информации (репрограммируемые ПЗУ). В отдельных типах БИС ЗУ в цепи электронного обрамления входят строчные и столбцовые буферные регистры, адресные мультиплексоры, блоки управления регенерацией и др.

Электрические параметры, характеризующие работу БИС ЗУ, подразделяются на статические и динамические (см. приложение, табл. П.II.3). К *статическим* параметрам БИС ЗУ относятся: параметры, характеризующие совместную работу БИС ЗУ с другими входными и выходными устройствами; предельные параметры, характеризующие устойчивость БИС ЗУ к воздействиям допустимых электрических режимов, после окончания которых БИС ЗУ не бу-

дет повреждена; параметры, характеризующие технологию производства.

Динамические параметры БИС ЗУ определяются временны́ми процессами и отображаются временны́ми диаграммами. Наибольшую общность для разных типов ЗУ имеют временны́е диаграммы работы ОЗУ с произвольной выборкой (см. приложение, рис. П.II.1), так как ими охватываются практически все виды динамических параметров ЗУ. Для конкретных типов БИС ЗУ те или иные

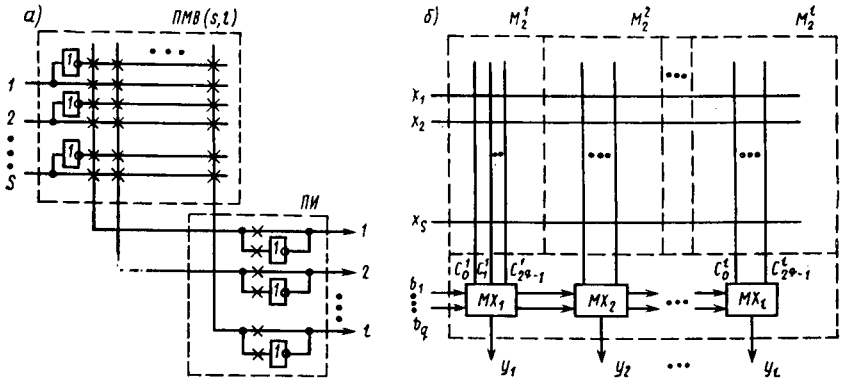


Рис. 2.4. Программируемые БИС:

а — матрица вентилей ПМВ (s, l); *б* — программируемый мультиплексор ПМ (s, l, q); M_2^i — матрица И — ИЛИ; $MX_1, ..., MX_l$ — мультиплексоры; $b_1, ..., b_q$ — управляющие входы; $C_0^i, ..., C_{2^q-1}^i$ — информационные входы; $x_1, ..., x_s$ — внешние информационные входы; $y_1, ..., y_l$ — информационные выходы

задержки между сигналами могут отсутствовать или даже быть другого знака, управляющие сигналы могут быть импульсными или потенциальными. Временны́е параметры, указываемые в ТУ, гарантируются лишь в рабочем диапазоне напряжения источников питания и температуры окружающей среды при определенных значениях сопротивления и емкости нагрузки.

Программируемые матричные БИС. К программируемым матричным БИС относятся матрицы вентилей, мультиплексоров, разнообразной логики.

Простейшую матричную схему рассмотрим на примере программируемой матрицы вентилей (ПМВ) (рис. 2.4, *а*), состоящей из единственной матрицы конъюнкций (М1) и выходных программируемых инверторов (ПИ), позволяющих представлять выходные сигналы в прямом или инверсном виде с реализацией функций — конъюнкции или дизъюнкции входных переменных, а также их отрицания. На одной БИС ПМВ, имеющей s входов и l выходов и условно обозначаемой как ПМВ (s, l), может быть реализовано не более l функций от k переменных ($k \leqslant s$). Например,

известна серийная ПМВ (16, 9), программируемая пользователем.

Программируемые матрицы логики (ПМЛ) представляют собой структуры, состоящие из наборов ПМВ (s, l), рассмотренных выше. К входам и выходам каждой из матриц М1 ПМВ, образующих отдельные секции ПМЛ, могут быть подключены различные логические и (или) запоминающие элементы. Неко-

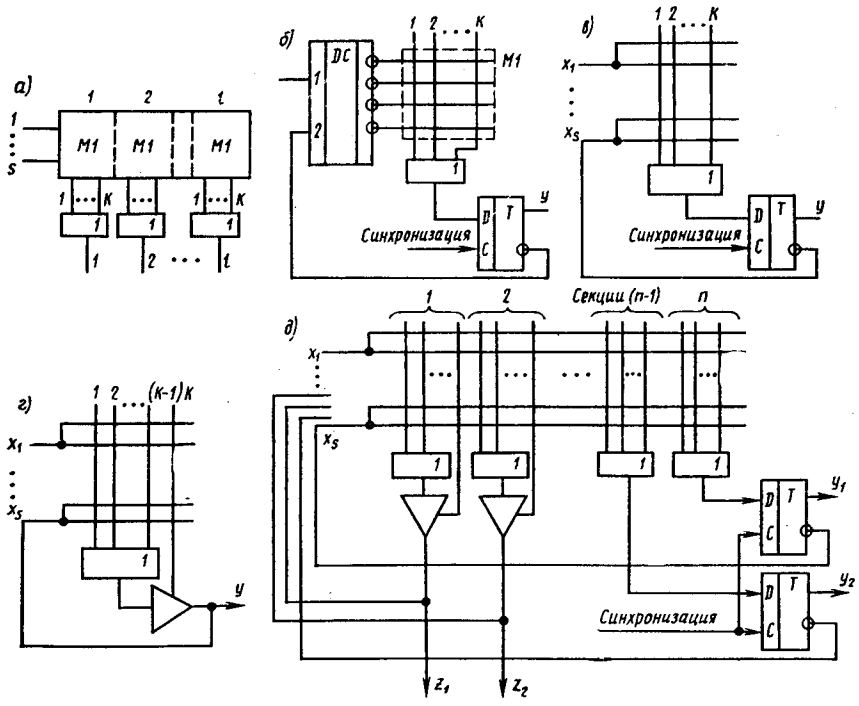


Рис. 2.5. Программируемая матрица логики ПМЛ (s, l):

а — простейшая ПМЛ из l секций с подключением на выходе матриц М1 схемы ИЛИ; *б* — ПМЛ с входными элементами на базе дешифратора и выходными элементами на базе логического элемента f со сложной функцией и синхронного D-триггера; *в*, *г* — выходные элементы ПМЛ на базе схемы ИЛИ и синхронного *D*-триггера или управляемого вентиля; *д* — сложная комбинированная ПМЛ

торые из типовых структур ПМЛ приведены на рис. 2.5. В качестве входных элементов для матриц М1 могут использоваться двухвходовые дешифраторы (рис. 2.5, *б*), в качестве выходных — схемы ИЛИ (рис. 2.5, *а*), схемы ИЛИ и управляемые буферные усилители (рис. 2.5, *г*), схемы ИЛИ и синхронные D-триггеры (рис. 2.5, *в*), логические элементы со сложной функцией и D-триггеры (рис. 2.5, *б*), а также их комбинации (рис. 2.5, *д*).

В зависимости от набора элементов ПМЛ могут быть простыми, состоящими только из l секций ПМВ, или сложными с исполь-

зованием разновидностей перечисленных входных и выходных элементов. В табл. П.II.4 приведены данные о некоторых типах зарубежных ПМЛ.

Программируемые мультиплексоры (ПМ) (рис. 2.4, *б*) состоят из матриц дизъюнкций (М2) и l мультиплексоров МХ1, МХ2, ..., $МХ_l$, имеющих общие управляющие входы $u_1, ..., u_q$, внешние информационные входы $x_1, ..., x_s$ и выходы $y_1, ..., y_l$. Каждый из мультиплексоров $МХ_i$ имеет q собственных управляющих входов, 2^q информационных входов $C^l_0,..., C^l_{2^q-1}$ и один выход y_i. Каждая матрица М2 имеет s входных и 2^q-1 выходных шин, подсоединяемых к соответствующим информационным входам своего мультиплексора, и позволяет реализовать 2^q элементарных дизъюнкций путем ее программирования с пережиганием перемычек. В общем случае матрицы М2 могут реализовать как элементарные дизъюнкции, так и конъюнкции аналогично матрице М1 ПМВ.

Из зарубежных серийных ПМ известен, например, ПМ (10, 4, 3), программируемый пользователем, где $s=10$, $l=4$, $q=3$.

Наиболее распространены на практике БИС программируемых матриц (ПЛМ), представляющие собой разновидность ПЗУ и выпускаемые в виде:

1) однородных ПЛМ с переменной (настраиваемой и перестраиваемой в процессе функционирования) структурой, в которой каждый логический элемент, коммутирующий горизонтальную и вертикальную шину матриц, может быть запрограммирован на включение (лог. 1) и отключение (лог. 0) путем установки в одно из двух состояний соответствующего ему триггера настройки;

2) неоднородных ПЛМ с жесткой структурой и транзисторными или диодными связями, программируемыми изготовителем БИС на пересечениях только тех пар шин матрицы, между которыми требуется передача электрического сигнала; такие ПЛМ принято называть *масочными* ПЛМ (МПЛМ);

3) однородных ПЛМ с жесткой структурой и полностью изготовленной матрицей шин с транзисторными или диодными связями, программирование которых осуществляется путем соответствующей настройки самим пользователем или изготовителем ПЛМ по данным (карте-заказу на программирование), выдаваемым пользователем; такие ПЛМ получили название программируемые пользователем логические матрицы — ППЛМ;

4) *ПЛМ с памятью*, содержащих наряду с полностью изготовленной матрицей шин, программируемой пользователем, память, состоящую из входных, выходных и промежуточных регистров и дополнительных вспомогательных блоков; такие ПЛМ представляют собой микропрограммную управляющую структуру;

5) *специализированных ПЛМ*, изготовляемых в виде однородных модулей или матричных структур с небольшим числом входов,

ориентированных на реализацию систем булевых функций (СБФ) с помощью настроечных элементов;

6) *репрограммируемых ПЛМ*, имеющих возможность многократной электрической записи и ультрафиолетового (рентгеновского) или электрического стирания информации. В табл. П.II.5, П.II.6 приведены характеристики ряда зарубежных и отечественных ПЛМ.

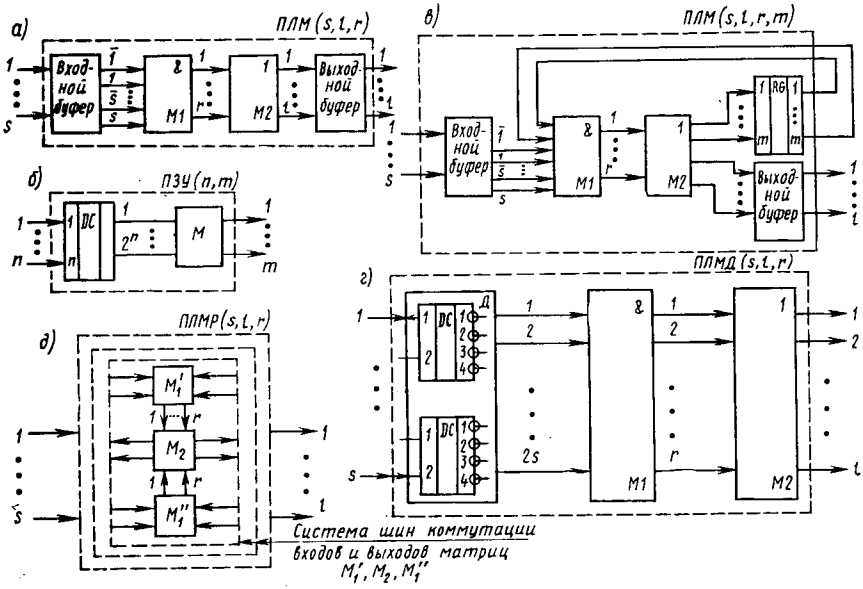


Рис. 2.6. Программируемая логическая матрица:

а — структура ПЛМ комбинационной логики; *б* — структура ПЗУ; *в* — структура ПЛМ с памятью; *г* — структура ПЛМ с дешифраторами на входе; *д* — структура ПЛМ с разрезанием шин; М, М1 — матрица конъюнкторов; М2 — матрица дизъюнкторов; DC — полный дешифратор; RG — регистр D-триггеров; M_1', M_1'' — части матрицы М1

Наибольшее применение получили ПЛМ комбинационной логики (рис. 2.6, *а*), условно обозначаемые как ПЛМ (*s*, *l*, *r*), где *s* — число входов, *l* — число выходов, *r* — число промежуточных шин. ПЛМ (*s*, *l*, *r*) предназначены для реализации любой СБФ и производят над ее входными переменными следующее преобразование:

$$y_j = \bigvee_{k=1}^{r} x_1^{a_{k1}} \ldots x_i^{a_{ki}} \ldots x_s^{a_{ks}} \beta_{kj}, \qquad (1.2)$$

где $y_j | j = \overline{1, l}$ — выходные значения булевых функций; $x_i | i = \overline{1, s}$ — входные переменные СБФ; k $(k = \overline{1, r})$ — общее число конъюнктивных термов (дизъюнкций) в каждой из функций y_j; α_{ki} (β_{kj}) — элементы (программирования) входных (выходных) переменных; $\alpha_{ki} \in \{0, 1, \times\}$; $\beta_{kj} \in \{0, 1\}$; $\times$ — безразличное состояние.

Конкретные значения α_{ki} и β_{kj} устанавливаются при программировании ПЛМ. Величина β_{kj} может принимать значение 1, если реализуемая конъюнкция $x_1, \ldots, x_s$ входит в представление j-й функции, и 0, если она не входит. Переменная x_i может входить в конъюнкцию в прямом или в инверсном виде и может не входить в нее. Если, например, на j-м выходе ПЛМ требуется иметь прямое представление функции y_j, а инверсное представление ее содержит меньшее число термов, то целесообразнее реализовать инверсную функцию, а затем обеспечить ее инвертирование. Этот прием способствует значительному сокращению аппаратурных затрат при реализации многих СБФ, экономии площади ПЛМ.

Конструктивно ПЛМ комбинационной логики (рис. 2.6, *а*) состоит из входного и выходного буферов и матриц М1 и М2, первая из которых формирует r конъюнктивных термов от s переменных (или их отрицаний), а вторая — l дизъюнкций от термов, полученных в матрице М1.

В целом структура ПЛМ комбинационной логики близка к структуре полупроводникового ПЗУ с числом входов n, полным дешифратором на входе и числом выходов $r=2^n$ (рис. 2.6, *б*) и отличается от нее лишь программируемой матрицей М1 с числом входов s и выходов $r \ll 2^s$. Однако наличие программируемой матрицы М1 в ПЛМ в отличие от жесткого входного дешифратора в ПЗУ дает ряд следующих преимуществ: если в ПЗУ для каждой входной комбинации существует соответствующее выходное слово, то в ПЛМ часть входных комбинаций может не выбирать ни одного выходного слова (что равносильно выбору нулевых слов в БИС ЗУ) или выбирать одно и то же выходное слово, а одна и та же входная комбинация может входить в несколько выходных слов.

ПЛМ с памятью условно обозначают как ПЛМ (s, l, r, m). где s, l, r — параметры, аналогичные параметрам ПЛМ комбинационной логики, m — число элементов памяти в цепи обратной связи. Структура ПЛМ с памятью (рис. 2.6, *в*) дополнительно содержит m-разрядный регистр в цепи обратной связи между матрицами М2 и М1, вследствие чего матрица М1 формирует q термов от $s'+m$ переменных (или их отрицаний), а матрица М2 реализует $l+r'$ дизъюнкций от термов, полученных в матрице М1. Регистр выполняют на различных типах триггеров, но наиболее часто на D-триггерах, способствующих сокращению числа выходов матрицы М2. Из этого типа ПЛМ известны, например, серийные БИС управляющей памяти, реализуемые на основе ППЛМ и отличающиеся большими информационно-логическими возможностями. К их числу относятся БИС из К587, К588, К1883 и других серий. Например, БИС КР587РП1 генерирует 14-разрядные коды микрокоманд и применяется для микропрограммного управления некоторыми типами МП, а также для аппаратной реализации автоматов. В ее состав (рис. 2.7) входят: ППЛМ, входные регистры (Рг1 и Рг2), регистр следующего адреса (РгСА), выходной регистр микрокоманды

(РгМК), блок синхронизации (БС), схемы обмена 1 и 2 (ОБ1 и ОБ2) и регистр управления (РгУ). Основой БИС УП является ППЛМ, состоящая из двух подматриц ПЛМ1 и ПЛМ2 и программируемого слоя инверторов (ПИ), наличие которых расширяет

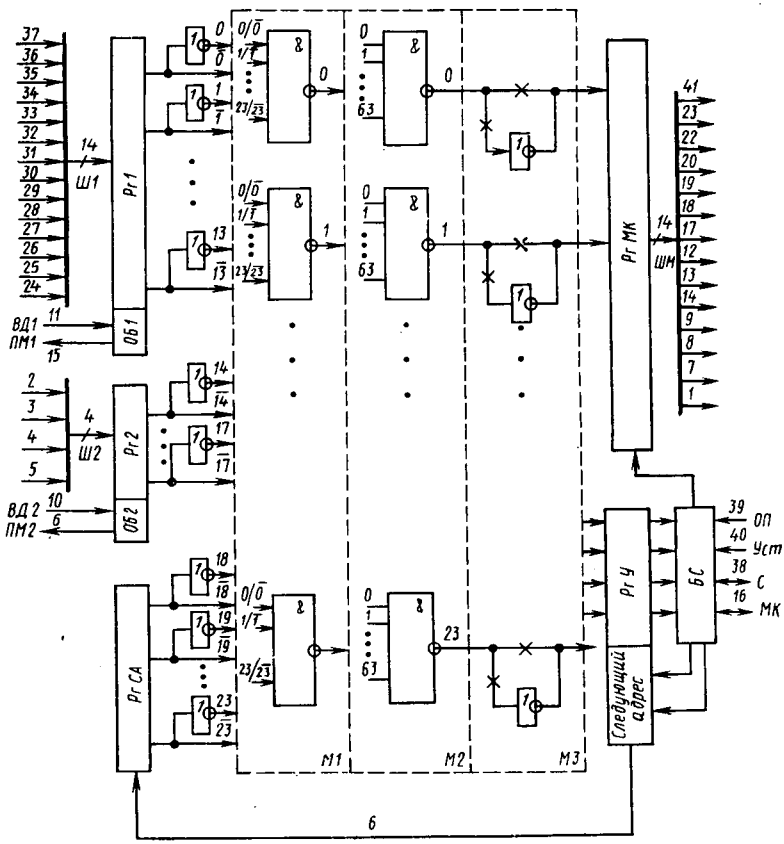


Рис. 2.7. Структурная схема БИС управляющей памяти К587РП1:
Рг1, Рг2 — входные регистры; РгМК — регистр микрокоманд, РгСА — регистр следующего адреса; РгУ — регистр управления; ОБ1, ОБ2 — схемы обмена; М1, М2, М3 — матрицы конъюнкторов, дизъюнкторов и программируемых инверторов

возможности УП. ПЛМ1 эквивалентна 64 схемам И—НЕ на 24 входа каждая, а ПЛМ2 эквивалентна 24 схемам И—НЕ на 64 входа каждая. Связи в подматрицах и программируемом слое инверторов проектируются в зависимости от требуемого набора микрокоманд, выполняются в процессе изготовления БИС и не могут быть изменены в дальнейшем.

Аналогичную структуру и принцип действия имеют БИС УП серий 588, 1883 и др., отличаясь числом триггеров, входов, выходов и конъюнктивных термов.

Выпускается модифицированный вариант ПЛМ (s, l, r) — ПЛМ (z, r), в котором фиксируется лишь два параметра: общее число входов и выходов $z=s+l$ и число промежуточных шин (термов) r. Конкретные значения s и l могут выбираться произвольно при настройке ПЛМ (z, r).

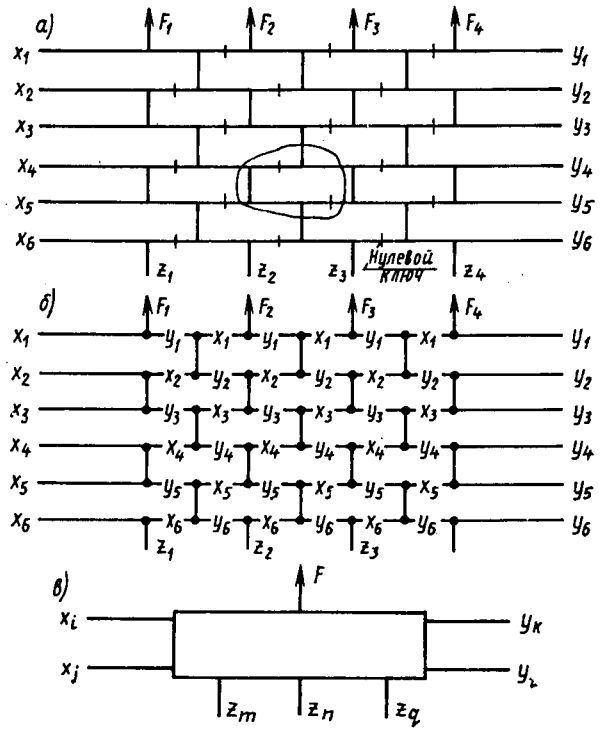


Рис. 2.8. Структура ПЛМ типа «пчелиные соты»:
а — общий вид; *б* — способ настройки; *в* — модуль ПЛМ П/2-3М

Существуют ПЛМ, на входе у которых устанавливается дополнительный блок Д (рис. 2.6, *г*) с различной реализацией (чаще всего двухвходовые полные дешифраторы). Такие ПЛМ условно обозначаются ПЛМД (s, l, r). Они позволяют реализовать булевы функции с числом термов, превышающим число промежуточных шин r, что невозможно достичь на обычных ПЛМ.

Выпускаются ПЛМ, допускающие разрезание шин в требуемом месте и обозначаемые условно ПЛМР (s, l, r). В такой ПЛМ (рис. 2.6, *д*) матрица М1 делится на две подматрицы М1′ и М1″, распо-

лагаемые над матрицей М2 и под ней соответственно, позволяя при необходимости разрезать промежуточные шины в матрице М2 и реализовать на верхней и нижней частях одной и той же промежуточной шины различные конъюнкции входных переменных. Любая горизонтальная шина матрицы М1′ (М1″) может разрезаться в одном месте, и на одну ее часть подается x_i (или $\bar{x}_i$) с левого входа матрицы, а на другую — с правого. Аналогично, для матрицы М2 любая горизонтальная шина разрезается в одном месте и на одной ее части формируется значение функции для левого выхода матрицы, а на другой — для правого. Кроме того, на кристалле БИС предусмотрена специальная система шин, позволяющая соединять выходы одной матрицы с входами другой. Выполнение разрезов шин и организация необходимых связей между входами и выходами различных матриц производятся заводом-изготовителем на этапе настройки ПЛМ.

Известен ряд специализированных ПЛМ типа «пчелиные соты» [28]. Настройка таких ПЛМ (рис. 2.8) осуществляется в процессе функционирования подачей управляющих сигналов, с помощью которых одна и та же ПЛМ в различные моменты времени реализует различные наборы выходных функций. Входы $\{x_i\}$, расположенные слева по периметру схемы, управляют левыми (нулевыми) ключами ПЛМ, а входы, расположенные справа $\{y_i\}$, — правыми (единичными) ключами. Вертикальные входы определяют распространение управляющих сигналов между входами $\{x_i, y_i\}$ и выходами $\{F_i\}$ ПЛМ.

Функция произвольного узла ПЛМ, обведенного на рис. 2.8, имеет вид $f = x_i^{\sigma_i}\varphi_i \vee y_j^{\sigma_j}\varphi_j$, , где $x_l^{\sigma_l}, y_j^{\sigma_j}$ — левый и правый горизонтальные входы узла; φ_i, φ_j — функции, поступающие на вертикальные входы, причем на информационные входы разрешена подача как самих переменных, так и их инверсий $\sigma = \{0, 1\}$; $x_l^0 = \bar{x}_l$; $x_l^1 = x_l$.

В целом простота и технологичность структуры ПЛМ, высокое быстродействие, низкая стоимость обусловливают широкое их применение при проектировании СЛУ.

§ 2.3. МИКРОПРОЦЕССОРНЫЕ КОМПЛЕКТЫ БИС

Применение микропроцессорных средств в практике проектирования систем управления и обработки информации позволяет перейти от «жесткой» аппаратной реализации алгоритмов к «гибкой» программной реализации, резко улучшить информационные, функциональные и надежностные характеристики создаваемых систем при относительно меньшем быстродействии их функционирования [1—6].

Системы управления можно строить на основе следующих микропроцессорных средств: секционированных микропрограммируемых БИС, однокристальных МП, однокристальных, одно- или мно-

гоплатных МЭВМ, микросистем, реализуемых на одной пластине или суперкристалле [1—3, 42].

Секционированные МПК БИС состоят из БИС или СБИС, каждая из которых является частью (секцией) функционального модуля. Секции могут объединяться с такими же или смежными секциями, образуя при этом разнообразные функциональные модули (процессоры, контроллеры и др.). К секционированным МПК БИС относятся серии К(КР)587, К(КР)588, К589, КР1802, КР1804, К582, К583, К584, К1800, U83-К1883.

Однокристальные МП состоят из БИС или СБИС, каждая из которых представляет собой один функциональный модуль вычислительной системы: процессор, память, контроллер и т. п. Однокристальные МП входят в серии К(КР)580, К586, К1801, К1810, К1816, К1815.

Однокристальные микроЭВМ (ОМЭВМ) состоят из СБИС, реализующих функции ЭВМ и имеющих в своем составе МП, ЗУ, УВВ. К однокристальным микроЭВМ относятся К586ВЕ1, К1801ВЕ1, К1814, К1820, КМ1816ВЕ48, КМ1816ВЕ51, К1816ВЕ35 и др.

Микропроцессорная вычислительная система на пластине состоит из одной СБИС, реализующей все основные компоненты и функции МПС. Например, на одной СБИС-пластине реализованы МП К1815ВФ3, цифровой матричный коммутатор КМ1509КП1 и ортогональная регистровая память 1517ИР1.

Микропроцессорные средства — конструктивно и функционально законченные изделия вычислительной и управляющей техники, построенные на основе микропроцессорных интегеральных схем, рассматриваемых как единое целое с точки зрения поставки, приемки, испытаний и эксплуатации, пригодные к применению для построения более сложных микропроцессорных средств или систем.

Микропроцессорный комплект интегральных схем (МПК ИС) включает в себя совокупность микропроцессорных и других микросхем, совместимых по конструкторско-технологическому исполнению и используемых при построении МЭВМ, контроллеров и других комплексов.

МПК ИС условно подразделяются на **универсальные** МПК, применяемые в самых различных средствах вычислительной и управляющей техники, и на **специализированные** МПК, предназначенные для построения только одного типа ЭВМ. К универсальным относятся такие комплекты, как К580, К584, К587, К588, К589, К1800, К1804, К1810, К1816 и др., а к специализированным — К581, К586, К536.

ИС, на которых строится собственно МП, образуют базовый МПК ИС. Микросхемы, необходимые для построения остальных устройств МПСУ (ОЗУ, ПЗУ, УВВ и т. д.), дополняют базовый МПК ИС до расширенного МПК ИС.

Совокупность системных устройств, состоящая из МП, памяти (БИС ОЗУ и ПЗУ), БИС ввода—вывода и при необходимости

пульта управления и источников электропитания, объединенных общей несущей конструкцией, нацеленная на выполнение множества определенных функций, называется **микрокомпьютером** или **микроЭВМ**. Микрокомпьютеры без периферийных УВВ, встраиваемые в технологическое оборудование, называют **контроллерами.**

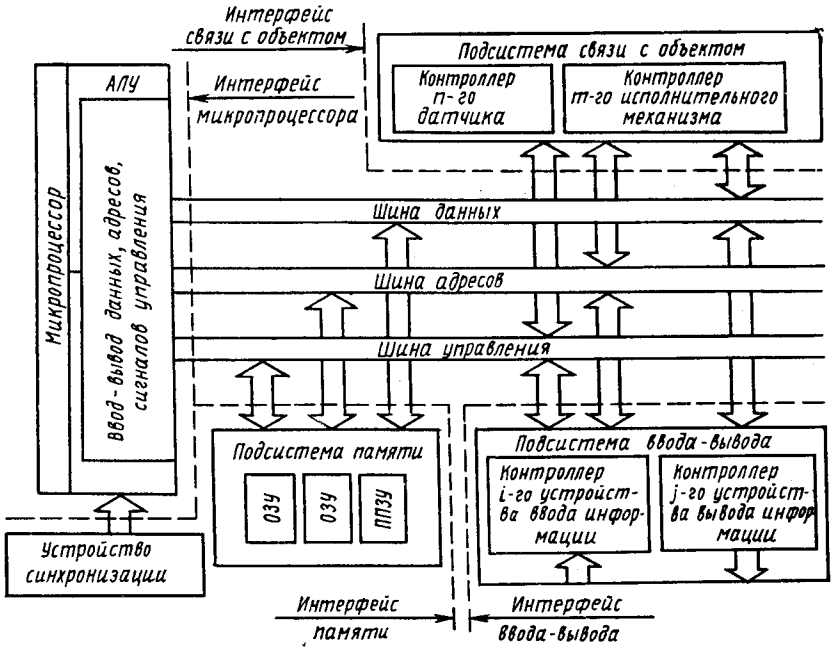


Рис. 2.9. Структура системы управления, построенной на МПК ИС

Специализированную информационную или управляющую систему, построенную на основе МЭВМ или МП и нескольких других БИС, способную выполнять операции над данными, хранимыми в памяти, согласно записанной программе управлять обменом информации с периферийным оборудованием, исполнительными механизмами, средствами измерения и контроля объекта управления, называют **микропроцессорной системой управления** (МПСУ). Система соединительных магистралей, вспомогательной аппаратуры, программных средств и алгоритмов для организации обмена информацией между МП, памятью и УВВ образует **интерфейс** МПСУ или МЭВМ (рис. 2.9).

Физические компоненты и схемы МПСУ или МЭВМ являются их аппаратурой, способной выполнять только ограниченный набор элементарных операций. К аппаратным средствам МПСУ относятся микропроцессорные средства (МПК БИС, однокристальные, од-

ноплатные и многоплатные МЭВМ и контроллеры, а также схемы сопряжения с обслуживаемыми объектами), размещенные на некоторой конструкции и соединенные согласно некоторой принципиальной схеме. Все прочие функциональные возможности МПСУ достигаются программным путем. Программы, написанные для МПСУ, образуют ее **программное обеспечение** (ПО).

Принципы построения МП и систем управления на их основе. Проектирование систем управления на основе МПК БИС предъявляет к разработчику некоторые новые требования [9—11]. Во-первых, эффективное применение новой элементной базы невозможно без знания структурно-функциональной организации и возможностей используемых МПК БИС, без наличия определенного математического обеспечения. Во-вторых, разработчику необходимо овладеть программированием (микропрограммированием), так как при создании МПС основная трудность заключается не в структурно-функциональном проектировании систем, а в разработке программного обеспечения. В-третьих, использование МПК БИС обусловливает реализацию новых принципов распределенной обработки информации, перестраиваемости структур и др.

Анализ структурной организации МП показывает, что существуют фиксированная структура БИС (однокристальные МП) и изменяемая структура БИС (секционированные, разрядно-модульные МП). При построении систем управления на базе МПК БИС широко используется структурный принцип, получивший название трех М (3М) и заключающийся в модульном подходе к построению структур на всех уровнях системы, в магистральном способе обмена информацией между функциональными модулями, в микропрограммном или программируемом управлении функциями модулей и процессом обработки информации.

Модульный подход при проектировании МПСУ способствует созданию рядов микроЭВМ, отличающихся функциональными возможностями и характеристиками, перекрывающих значительный диапазон их применения, стандартизации элементов все более высокого уровня, простому наращиванию вычислительных мощностей и возможности реконфигурации систем.

Магистральный способ обеспечивает обмен информацией между функциональными и конструктивными модулями различного уровня с помощью одной—трех магистралей или более. Использование магистрального способа обмена информацией обеспечивает регулярность структуры МПК БИС и связей между конструктивными модулями МПСУ, двунаправленный обмен данными, стандартизацию интерфейсов, сокращение числа выводов БИС.

Микропрограммное управление обусловливает бóльшую гибкость при организации многофункциональных микропроцессорных модулей, увеличивает их регулярность, обеспечивает параллелизм решения задач, повышает надежность МПСУ благодаря применению серийных БИС ЗУ, упрощает контроль функционирования.

Однокристальные микропроцессоры (ОМП) (рис. 2.10) характеризуются фиксированными разрядностью (8, 16, 32 бит) и системой команд, отличаясь от других типов МП следующими основными особенностями:

на кристалле удается реализовать 8- и 16-разрядное АЛУ с достаточно простыми функциями;

организации исполнения процедур в АЛУ в различных типах МП сходны между собой, отличаясь лишь деталями структуры и количеством регистров общего назначения (РОН);

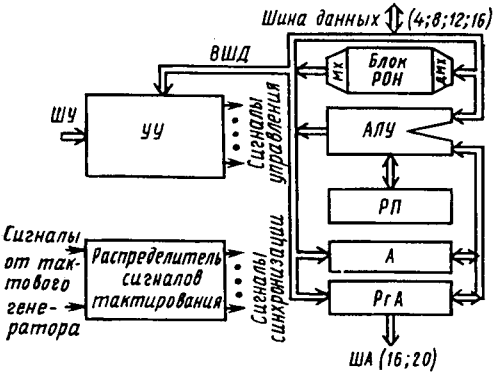


Рис. 2.10. Структура однокристального микропроцессора:

УУ — устройство управления; АЛУ — арифметико-логическое устройство; РП — регистр признаков; А — аккумулятор; РОН — регистры общего назначения; РгА — регистр адреса; ВШД — внутренняя шина данных; МХ (ДМХ) — мультиплексор (демультиплексор); ША — шина адреса

управляющие устройства (УУ) ОМП строятся обычно на основе «жесткой» (схемной) логики, реализуемой в базисе программируемых логических матриц (ПЛМ); если внутренней организацией МП практически мало отличаются друг от друга, то их системы команд сильно различаются по числу команд и по реализуемым ими процедурам;

большинство ОМП имеет 16- или 20-разрядную шину адреса (ША), что позволяет осуществлять прямую адресацию к внешней памяти емкостью 64 К или 1 М байт;

шину управления (ШУ) образуют линии (от 6 до 17), предназначенные для передачи управляющих сигналов, признаков состояния процессора и периферийного оборудования. Обычно по этим линиям передаются: синхросигналы для сопровождения информации при двунаправленной передаче по мультиплексированной внутренней шине данных, сигналы обращения к памяти (чтение или запись), сигналы о состоянии внешних устройств (готовность), запросов на обслуживание от УВВ, разрешение прерывания, запрос и разрешение на прямой доступ к памяти и др.;

системная ШД из-за ограниченного числа внешних выводов МП реализуется с помощью специальных буферных схем, осуществляющих электрическое и логическое разделение внешней и внутренней ШД и функционирующих в режиме мультиплексированных двунаправленных передач;

в состав блока РОН, как правило, входят: счетчик команд, регистр косвенного адреса, индексный регистр, регистр-указатель стека, некоторое число рабочих регистров (от 4 до 16), предназна-

ченных для тиражирования операндов и их временного хранения и др.;

наличие в блоке РОН указателя стека и специального блока памяти, реализующего стековую дисциплину доступа, позволяет: быстро реагировать на запросы прерывания от УВВ и аппаратными средствами осуществлять переход к подпрограммам обслуживания и возврат к прерванной программе; эффективно работать с подпрограммами с помощью достаточно простых схем и однобайтовых команд;

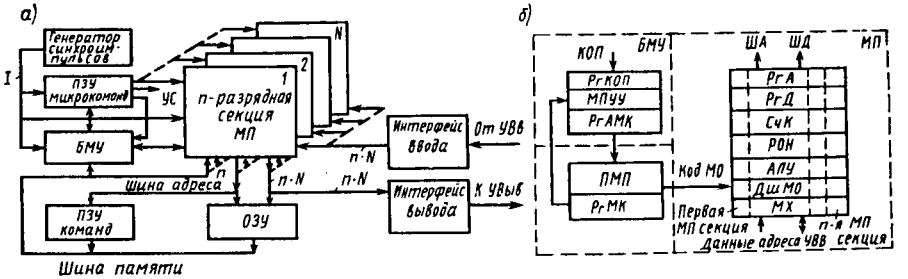


Рис. 2.11. Организация МПС на основе МП с микропрограммным управлением: *а* — структура ММПС; *б* — структура микропрограммируемого МП; БМУ — блок микропрограммного управления; ПМП — память микропрограмм; МПУУ — микропрограммное управляющее устройство; РгАМК — регистр адреса микрокоманд; РгА, РгД — регистры адреса, данных; СчК — счетчик команд; ДшМО — дешифратор микроопераций

наличие в блоке РОН только регистра-указателя стека (УС) при выполнении самого стека в виде некоторой зоны в ОЗУ способствует увеличению емкости стека, числа вложений при обработке подпрограмм, рекурсивных процедур;

процесс выполнения команд управляется на микропрограммном уровне путем координации работы УУ с содержимым регистра признаков (РП), осуществляющим идентификацию состояния МП в каждый момент выполнения команды. Возможность программным способом опрашивать определенные разряды РП позволяет производить условные переходы в программе по значениям соответствующих признаков.

В целом вычислительные и функциональные возможности ОМП определяются характеристиками АЛУ, УУ, емкостью блока РОН и параметрами ША, ШУ, ШД. ОМП ориентированы преимущественно на те области (дискретная автоматика, УВВ, системы связи и передачи данных и др), где наилучшим образом используются их положительные качества (универсальность, наличие готового ПО, низкая стоимость, высокая надежность) и где не требуются перестройка структуры МП, большая разрядность обрабатываемых слов, высокое быстродействие.

МПСУ на базе микропрограммируемых МП (ММП) (рис. 2.11, *а*) характеризуются наращиваемой разрядностью и микропро-

граммным управлением. Они реализуются в виде набора нескольких микропроцессорных секций, блока микропрограммного управления (БМУ), ПЗУ микрокоманд, ПЗУ, ОЗУ, БИС интерфейса ввода—вывода и ряда других вспомогательных БИС и СИС. Включая или исключая эти компоненты, изменяя способы соединения и содержимое ПЗУ микрокоманд, вводя программный уровень управления и различные способы организации интерфейса, проектируют МПСУ с необходимыми функциональными и временными характеристиками. Выполнение команд в микропрограммируемых МПС обеспечивается последовательной выборкой из ПЗУ микрокоманд управляющих слов, каждое из которых содержит информацию об элементарных действиях, подлежащих выполнению в данном такте. Последовательности микрокоманд, обеспечивающие реализацию определенных операций, составляют микропрограммы этих операций.

В структуру многокристального ММП (рис. 2.11, *б*) обычно входят: а) микропроцессорная секция, содержащая регистры адреса (РгА) и данных (РгД), РОН, счетчик команд (СчК) и АЛУ, используемые в качестве операционного устройства, а также дешифратор микроопераций (ДшМО), мультиплексоры (МХ); б) блок микропрограммного управления (БМУ), включающий регистры кода операций (РгКОП), адреса микрокоманд (РгАМК) и микропрограммное устройство управления (МПУУ); в) ПЗУ микрокоманд, включающее память микропрограмм (ПМП) и регистр микрокоманд (РгМК).

Достоинствами многосекционных МП с микропрограммным управлением являются гибкость в расширении разрядности МП, возможность разработки команд, ориентированных на конкретное применение, наличие большого числа независимых шин, что позволяет избежать мультиплексирования и введения дополнительных буферных схем. Вместе с тем выбор и разработка набора команд для такого МП затруднительны, так как процесс микропрограммирования более сложен, чем программирование с фиксированным набором команд. Здесь разработчику микропрограммы необходимо больше знать о структуре МП, временны́х соотношениях, механизмах формирования адресов следующих микрокоманд, тактах передачи кодов, возможных задержках и др. Программное обеспечение для микропрограммирования обычно более слабое или вообще отсутствует, что приводит к трудностям в написании, редактировании и отладке микропрограмм.

§ 2.4. МИКРОЭВМ В СИСТЕМАХ УПРАВЛЕНИЯ

Серийные микроЭВМ. В настоящее время функционирует большое число отечественных микроЭВМ, выпускаемых промышленностью, например:

— МЭВМ ряда «Электроника С5», реализованные на основе МПК К536 и К586;

— МЭВМ ряда «Электроника НЦ», реализованные на основе МПК К587 и К588;

— МЭВМ ряда «Электроника 60», реализованные на основе МПК К581;

— микроЭВМ, реализованные на основе МПК К580 («Электроника К», СМ1800, В7; «Кристалл 60»; КТС ЛИУС-2 и др.);

Таблица 2.2. **Основные характеристики моделей микроЭВМ ряда «Электроника С5»**

Наименование характеристики МЭВМ	Многоплатные модели		Одноплатные модели				Однокристальные модели	
	С5-01	С5-02	С5-11	С5-12	С5-21 С5-21М	С5-41	МС1211	С5-31
Объем ОЗУ, слов	1—6 К	4—10 К	128	128	256	1—4 К	1—2 К	128
Параллельный ввод — вывод, байт	10 входных, 9 выходных	17 входных, 15 выходных	4	4	4	4	8	3
Последовательный ввод-вывод, бит	3	4	—	—	4	4	8	1

Примечание. У всех моделей микроЭВМ быстродействие составляет 10 000 операций в секунду; разрядность — 16, число каналов прерывания — 8.

— специализированные микроЭВМ типа ПС-300, реализуемые на основе специализированных элементов однородных сред и др.

Как правило, МЭВМ одного ряда имеют общие архитектурные принципы построения, математическое обеспечение, единую систему команд, программно-аппаратную совместимость. Более поздние модели МЭВМ целиком используют ПО ранних моделей. Перспективные МЭВМ ориентируются на использование периферийных устройств, совместимых с интерфейсом «Общая шина» и с интерфейсом ЕС ЭВМ, они обладают развитой системой прерывания с быстрой реакцией на прерывание при работе в реальном масштабе времени и способностью образовывать мультимикропроцессорные и многомашинные структуры.

МикроЭВМ ряда «Электроника С5» являются универсальными МЭВМ одно- и многоплатного, а также однокристального исполнения (табл. 2.2). Они выполнены по единой технологии, имеют совместимое математическое обеспечение и единую систему команд. Система математического обеспечения разработана с учетом работы МЭВМ в автономном режиме и в комплексе с большими ЭВМ.

Имеются кросс-ассемблер на базе БЭСМ-6 и ЕС ЭВМ; резидентный ассемблер на уровне автокода; резидентный транслятор с языка Бейсик; библиотека стандартных программ и др. Все машины ряда оснащаются дополнительными многофункциональными модулями (МФМ) (табл. 2.3), что способствует широкому применению в АСУ ТП, аппаратуре передачи данных, телеграфии, мелиорации, системах контроля БИС, приборах для испытаний конструкций и др.

Таблица 2.3. **Перечень дополнительных многофункциональных модулей микроЭВМ «Электроника С5»**

Наименование МФМ	Назначение МФМ
«Электроника С5-2101»	Модуль АЦП
«Электроника С5-2115»	Модуль ЦВВ и управления ГМД
«Электроника С5-2103»	Модуль управления перфоратором, фотосчитывающим устройством и телетайпным аппаратом
«Электроника С5-2105»	Модуль ОЗУ на 16 К 16-разрядных слов
«Электроника С5-2106»	Модуль дисплейного адаптера
«Электроника С5-2108»	Модуль ППЗУ на 4 К 16-разрядных слов
«Электроника С5-2111»	Модуль эмулятора ввода — вывода однокристальной МЭВМ
«Электроника С5-2107»	Модуль пульта программиста

МикроЭВМ ряда «Электроника НЦ» (табл. 2.4) отличаются большим объемом памяти и числом команд, более высоким уровнем машинного языка и быстродействием, полной автономией, применяются и как самостоятельные УВМ, и как элементы вычислительной системы равноправных или соподчиненных МЭВМ. Для связи с УВВ разработана унифицированная магистраль ряда, обеспечивающая совместимость с СМ ЭВМ, одновременную работу нескольких процессоров и прямое управление одного процессора другим. К ней можно подключать устройства перфоленточного ввода и вывода, дисплей, устройства печати. Для расширения памяти МЭВМ разработаны автономные ОЗУ с произвольной выборкой из памяти 18-разрядных слов — «Электроника 64К», «Электроника 256К». С помощью этих ОЗУ можно создавать запоминающие системы емкостью до 1 мегаслова (комплект из 16 блоков для «Электроники 64К»). Кроме того, через радиальный интерфейс предусмотрено подключение внешних ЗУ ряда СМ ЭВМ на магнитных дисках и магнитных лентах. Математическое обеспечение разработано с учетом работы МЭВМ как в автономном режиме, так и в системе, имеются кросс-обеспечение на базе БЭСМ-6 и ЕС ЭВМ, резидентный ассемблер, библиотека стандартных программ, систе-

ма редактирования и отладки, перфоленточная ОС (ПЛОС) и др.

МикроЭВМ ряда «Электроника 60» имеют модульный принцип построения. Все функциональные блоки выполнены в виде конструктивно законченных устройств (модулей), связь между которыми осуществляется через единый канал обмена информацией. МЭВМ «Электроника 60» выпускается в нескольких исполнениях, имеет ряд дополнительных модулей для расширения памяти, интерфейса, связи с разнообразными УВВ, датчиками и преобразователями. Канал ЭВМ обеспечивает реализацию трех типов обмена данными: программный обмен, обмен в режиме ПДП и обмен в режиме прерываний. В состав ПО МЭВМ входят абсолютный загрузчик, основной тест команд, тест прерываний, программы на языках Ассемблер, Бейсик и др.

Таблица 2.4. **Основные характеристики моделей микроЭВМ ряда «Электроника НЦ»**

Наименование характеристики	«Электроника НЦ»					
	НЦ-03Т	НЦ-03Д	НЦ-31	НЦ-04Т	НЦ-80	НЦ-80-01
Быстродействие, тыс. операций в секунду	100	100	130	200	550/250	550/250
Разрядность, бит	16	16	16	16	16/32	16/32
Объем ОЗУ, слов	8 К	16 К	8 К	32 К	128	16 К
Объем ПЗУ, слов	8 К	16 К	32 К	32 К	1024	16 К
Число основных команд (с модификацией)	190	188	280	328	120	120
Число уровней прерывания	4	1	4	2	8	8

В настоящее время МЭВМ «Электроника 60» включают в себя ряд МЭВМ «Электроника 60М», «Электроника 60Т», представляющие собой конфигурируемые пользователем модульные структуры. Помимо центрального процессора они включают в себя набор функциональных блоков (модули дополнительной памяти, контроллеры периферийных устройств и модули межпроцессорной связи). Эти МЭВМ аппаратно- и программно-совместимы и различаются реализацией ЦП, набором команд, быстродействием, степенью интеграции и объемом резидентной памяти. Известны следующие серийные комплекты МЭВМ ряда «Электроника 60»: а) 15ВМ-16-002, являющийся минимальным комплектом, состоящим из ЦП и устройства управления В1, который обычно встраивается в технологическое оборудование; б) 15ВМ-16-004, состоящий из ЦП, устройств управления типа В1 и В2 и источника питания; в) 15ВМ-16-005, представляющий собой автономную вычислительную систе-

му, рассчитанную на одного пользователя, который содержит комплект 15ВМ-16-004, электрическую пишущую машинку «Консул-260», фотосчитыватель и перфоратор ПЛ-150.

Развитием ряда МЭВМ «Электроника 60» являются МЭВМ «Электроника 60-1» и ее модификации «Электроника МС1211», «Электроника МС1212», обладающие более высоким (в 2—3 раза) быстродействием, расширенной системой команд, увеличенным объемом памяти, возможностью мультипрограммного режима работы. Например, МЭВМ МС1211 выпускаются в двух вариантах: МС1211.01, являющийся встраиваемой моделью, и МС1211.02, предназначенный для установки в стойки. Дальнейшим развитием МЭВМ этого ряда является персональный вычислительный комплекс «Электроника МС0585», предназначенный для применения в АСУ ТП, системах автоматизации научных и инженерных расчетов и обработки экономической и статистической информации.

МикроЭВМ СМ1800 (и другие из этого семейства) используются для построения низовых подсистем управления и контроллеров распределенных АСУ ТП и обеспечивают сбор информации с аналоговых и дискретных датчиков, фиксирующих ход процесса; обработку, хранение принятой информации и передачу ее в ЭВМ верхнего ранга; выдачу управляющих воздействий на исполнительные механизмы; связь с оператором посредством видеотерминалов, устройств технологической печати; ввод—вывод информации с помощью перфоленты и гибких дисков. Все функциональные блоки СМ1800 объединяются в единую систему на основе интерфейса И41. Через блок связи с интерфейсом «Общая шина» или через модуль сопряжения по стыку С2 обеспечивается связь с СМ ЭВМ, ЕС ЭВМ или МЭВМ. СМ1800 укомплектована модулями памяти, устройствами связи с объектами управления и УВВ, имеет мощные резидентные и кросс-средства на различных ЭВМ.

В последнее время благодаря успехам интегральной технологии в распоряжении пользователей появились разнообразные одноплатные и однокристальные МЭВМ. Они обладают значительными вычислительными ресурсами, высокой надежностью, низкой стоимостью и широко применяются при построении распределенных систем управления и микропроцессорных контроллеров разнообразного назначения. К их числу относятся, например, одноплатные МЭВМ ряда «Электроника МС1201», выполненные на МПК К1801 и совместимые с МЭВМ ряда «Электроника 60» по системе команд, системной магистрали и конструктивному исполнению; «Электроника МС1110.01», «Электроника НЦ-80», «Электроника С5-31», а также однокристальные МЭВМ серий К1814, К1816, К1820 и др.

Программируемые микроконтроллеры. В автоматизированном производстве широко применяются *программируемые микроконтроллеры* (ПК), представляющие собой специализированные управляющие МЭВМ, работающие в реальном масштабе времени по определенным рабочим программам, размещаемым в ПЗУ. В мире

выпускается свыше 150 типов ПК. Они используются примерно в 35% систем автоматизации технологических процессов и в большинстве случаев реализуют законы программно-логического управления или аналого-цифрового регулирования. Различают ПК трех типов:

— программируемые логические контроллеры (ПЛК), ориентированные на реализацию алгоритмов логического управления, обеспечивающие замену релейных и бесконтактных схем электроавтоматики;

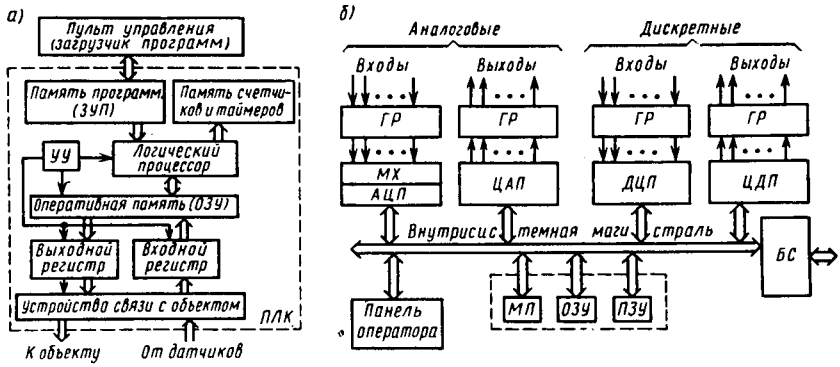


Рис. 2.12. Структура программируемого контроллера (ПК):

а — логического ПК; *б* — регулирующего ПК; УУ — управляющее устройство, МХ — мультиплексор; ГР — гальваническая развязка, АЦП (ЦАП) — аналого-цифровой (цифро-аналоговый) преобразователь; ДЦП (ЦДП) — преобразователь дискретно-цифровой (цифро-дискретный); МП — микропроцессор; ПЗУ (ОЗУ) — постоянная (оперативная) память; БС — блок сопряжения

— программируемые регулирующие микроконтроллеры, или ремиконты, ориентированные на реализацию алгоритмов автоматического регулирования аналоговых и аналого-дискретных технологических процессов, заменяющие различные аналоговые и цифровые регуляторы;

— микроконтроллеры, ориентированные на реализацию специальных алгоритмов управления игровыми автоматами, бытовыми приборами, светофорами, контрольно-измерительной аппаратурой, транспортными механизмами и др.

Программируемые логические контроллеры. ПЛК осуществляют реализацию систем булевых функций в реальном масштабе времени и представляют собой программно-настраиваемую модель цифрового управляющего автомата, ориентированную на определенную область применения. В состав ПЛК (рис. 2.12, *а*) входят МП, управляющее устройство (УУ), память программ (ЗУП), оперативная память данных (ОЗУ), устройства связи с объектом (УСО), пульт настройки системы и загрузки программ. В памяти программ производится запись алгоритма уп-

равления в виде последовательности строк, содержащих булевы функции. В ОЗУ записываются состояния одноразрядных (битовых) входов, выходов и внутренних элементов. Сигналы от датчиков через УСО, осуществляющие нормализацию и масштабирование, поступают на входной регистр, а управляющие команды хранятся в выходном регистре и через УСО воздействуют на объект.

ПЛК функционирует в циклическом режиме: МП в соответствии с программой ЗУП моделирует конкретную релейную схему (булевы уравнения) системы управления, опрашивает все входы, производит логическое сравнение состояний входов и выходов и по результатам сравнения управляет теми или иными исполнительными устройствами. МП последовательно строка за строкой опрашивает ЗУП, производит последовательное вычисление системы булевых функций, заданной в программе, и заносит вычисленные значения в память данных по окончании опроса всего ЗУП. УУ организует обмен данными между входными и выходными регистрами в ОЗУ. Далее процесс опроса памяти и обмена данными периодически повторяется. В целом ПЛК характеризуются следующими особенностями:

рабочие циклы ПЛК, непрерывно повторяемые в режиме управления объектом, состоят из трех сменяющих друг друга фаз: получения данных о текущем состоянии объекта, переработки этих данных в управляющие сигналы и выдачи их на исполнительные устройства;

ориентация на логические, а не на арифметические операции ведет к однообразной последовательной обработке входных данных, применению в составе ПЛК однобитовых МП с системой команд, не содержащей развитые арифметические операции;

в ПЛК используются простые специализированные языки программирования и языки описания алгоритмов управления (например, язык релейно-контактных схем (РКС), язык булевой алгебры, язык символического кодирования и др.), содержащие от единиц до нескольких десятков операторов, включая такие операторы, как вычисление булевых функций, сравнение кодов чисел, выполнение простейших операций над числами, подсчет времени и числа событий и т. п. Программирование при этом может осуществляться самим потребителем, не имеющим специальной подготовки в области программирования;

пульт управления загрузкой программ выполняется в виде автономного и переносного устройства, подключаемого к ПЛК на время только с целью набора требуемых программ с клавиатуры пульта и запоминания их в ЗУП;

наличие в ПЛК сканирующего устройства позволяет непрерывно и поочередно опрашивать все логические входы и выходы, а наличие УСО обеспечивает согласование и гальваническую развязку входных сигналов и усиление выходных сигналов.

По степени сложности реализуемых алгоритмов управления и функциональным возможностям ПЛК условно подразделяются на ПЛК, осуществляющие управление:

несложными объектами по фиксированной программе; ПЛК имеют небольшое число (от 16 до 96) входов и выходов, малую емкость внутренней памяти, не имеют внешнего языка, а алгоритмы управления объектом задаются, как правило, микропрограммным способом;

достаточно сложными, локально работающими объектами; ПЛК имеют несколько сотен входов и выходов, достаточную емкость, ЗУП и ОЗУ, внешний язык задания алгоритмов управления и средства их загрузки; системные средства у таких ПЛК, как правило, отсутствуют:

ответственными, достаточно сложными и даже уникальными объектами; ПЛК могут иметь мультимикропроцессорную структуру, сопрягаться с ЭВМ более высокого ранга, осуществлять обработку данных и выдавать интегральную информацию на верхний уровень управления; такие ПЛК имеют развитый язык задания алгоритмов управления, собственные периферийные УВВ (видеотерминал, печатающее устройство и др.).

Обычно ПЛК применяются для управления электроавтоматикой станочных комплексов с ЧПУ, обрабатывающих центров, промышленных роботов и др. Алгоритмы управления подобными объектами характеризуются частым изменением технологических параметров и необходимостью модификации отдельных программ в процессе функционирования.

Регулирующие микроконтроллеры. Для управления современными объектами с непрерывным характером технологических процессов требуется большое число разнообразных типов регуляторов (аналоговых и цифровых), представляющих собой узкоспециализированные приборы для одноконтурного регулирования по ПИ-, ПИД- и другим, более сложным законам регулирования. Известно множество законов регулирования, в том числе адаптивных, скользящих, беспоисковых и других, но аппаратная реализация их сложна, дорогостоящa и менее надежна.

Появление регулирующих микроконтроллеров (РМК), обладающих значительными логическими и вычислительными возможностями, обеспечило программную реализацию любых алгоритмов регулирования. Один РМК может заменить группу (из 10—60 обычных ПИ- или ПИД-) аналоговых или цифровых регуляторов для 8—64 контуров регулирования. Однако выход его из строя ведет к нарушению регулирования во всех контурах, из-за чего предусматривают дублирование РМК или переход на ручное управление.

Структура РМК (рис. 2.12, *б*) аналогична структуре МЭВМ и содержит УСО или средства ввода—вывода аналоговых и дискретных сигналов. Отличие РМК от универсальной МЭВМ состоит в том, что основные алгоритмы регулирования записываются в па-

мять РМК уже при его изготовлении на заводе-изготовителе и для его применения не требуется стандартных УВВ и средств отладки программ.

Мультиплексоры и ЦАП обеспечивают прием 16 аналоговых сигналов, причем подключение датчиков осуществляется под управлением МП. В ремиконте предусмотрена возможность установки четырех мультиплексоров и АЦП, обеспечивающих прием данных от 64 аналоговых датчиков. На выходе ремиконта не предусмотрено мультиплексирование, и число ЦАП равно числу выходных цепей, наращиваемых группами от 8 до 64 по 8 выходов в группе. Дискретно-цифровые и цифро-дискретные преобразователи (ДЦП и ЦДП) выполняют функции согласования уровней логических сигналов, принятых в ремиконте, с уровнями дискретных сигналов, обеспечивая групповой прием и выдачу от 16 до 128 дискретных сигналов по 16 сигналов в группе. Путем простой перекоммутации осуществляется изменение функций ЦДП на цифро-импульсные с программным управлением скважностью выходных импульсов, выдаваемых для управления исполнительными механизмами с переменной частотой вращения. Все каналы ремиконта могут работать автономно и не иметь перекрестных связей, что эквивалентно многоканальному регулятору. При связи отдельных выходов с соответствующими входами можно получить многосвязные многоуровневые структуры регулирования.

В память программ РМК записывается библиотека программ с числом стандартных алгоритмов, не превышающим 20—25, включающих аналоговое и импульсное ПИД-регулирование, динамическое преобразование (суммирование, умножение и др.), нелинейное преобразование (селектирование, переключение), функции управляющей логики (логическое сложение, умножение и др.). Комбинации этих алгоритмов позволяют строить системы регулирования практически любой сложности.

Панель оператора имеет клавиши с надписями, принятыми в регулировании, что позволяет человеку общаться с РМК в процессе эксплуатации на понятном пользователю языке. Оператор с помощью панели управляет выбором режимов, установкой и изменением задания значений технологических параметров, а также осуществляет выбор конфигурации регулирующего контура, требуемых алгоритмов управления и параметров статической и динамической настройки РМК.

РМК имеют внутреннее ПО и не требуют внешних программных средств (операционной системы, транслятора, ассемблера и др.). Внутреннее ПО РМК состоит из *диспетчера рабочих программ,* координирующего весь процесс вычислений в реальном времени, *рабочих программ*, реализующих алгоритмы управления, программы обслуживания панели оператора, выполняющей приказы оператора и выдающей ему информацию с помощью световых индикаторов и светодиодов, *диагностической программы,* контролирующей

безотказность аппаратных и программных средств, указывающей место неисправности, сигнализирующей об этом оператору или вводящей автоматически резервные блоки взамен отказавших.

Ремиконт управляет 8—16 контурами технологического процесса, работает со стандартными датчиками с унифицированным выходом (0—5 или 4—20 мА) и исполнительными устройствами пропорционального действия или постоянной скорости. Он может работать как в автономном режиме, так и связываться с ЭВМ верхнего уровня и интерактивными средствами представления данных. Пока применение ремиконтов вместо обычных регуляторов экономически оправдано в системах, содержащих свыше 6—8 каналов регулирования. Однако развитие однокристальных МП и МЭВМ, совмещающих на одном кристалле память, АЦП и ЦАП, позволит эффективно применять их и в одноконтурных регуляторах.

Контрольные вопросы

1. В чем состоят достоинства и недостатки микропроцессорной реализации систем управления перед их аппаратной реализацией на основе СИС и БИС? Какие разновидности БИС вы знаете? В чем их особенности? 2. Какие разновидности комбинационных СИС вы знаете? В чем заключается особенность реализации СИС с двусторонней передачей сигналов? Поясните этот принцип на примере работы СИС К564КП1, К564КП2. 3. Перечислите разновидности ЗУ. Какие из них наиболее часто применяются в МПСУ? 4. В чем сходство и отличие в функционировании и построении структур ЗУ (ПЗУ, ППЗУ, ОЗУ)? 5. Какие типы программируемых матричных БИС вы знаете? В чем заключается особенность построения разновидностей ПЛМ? Приведите примеры построения структуры ПЛМ. В чем их отличие от структур ПМВ, ПМЛ? 6. Какие преимущества обеспечивает применение МП и МЭВМ? 7. О чем говорит приставка «микро» в терминах МП и МЭВМ? В чем отличие МЭВМ от мини-ЭВМ, большой ЭВМ? 8. Перечислите основные компоненты структуры МПСУ, состав функций и назначение отдельных компонент. 9. В чем заключается принцип «ЗМ» 10. Какие типы МЭВМ вы знаете? Дайте краткую их характеристику. 11. Какие разновидности программируемых контроллеров вы знаете? В чем их особенности?

ГЛАВА 3

ПРОЕКТИРОВАНИЕ СИСТЕМ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ НА ОСНОВЕ СХЕМ БОЛЬШОЙ И СРЕДНЕЙ СТЕПЕНИ ИНТЕГРАЦИИ

При формализованном проектировании систем логического управления применяются различные математические (поведенческие и структурные) и физические модели, среди которых основной является математическая модель конечного детерминированного автомата, или просто автомата [22, 29]. В зависимости от специфики работы СЛУ различают последовательностные автоматы (автоматы с памятью), комбинационные схемы (автоматы без памя-

ти), перестраиваемые автоматы (с памятью и без памяти) и др. [19, 27], которые могут быть реализованы аппаратным и(или) программным путем.

Рассмотрим некоторые из методов проектирования СЛУ на базе наиболее распространенных СИС и БИС.

§ 3.1. СИНТЕЗ СИСТЕМ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ НА МИКРОСХЕМАХ СРЕДНЕЙ СТЕПЕНИ ИНТЕГРАЦИИ

При проектировании систем логического управления на основе СИС выделяются два основных направления: первое связано с исследованием возможностей применения серийных типовых СИС — сумматоров, дешифраторов, мультиплексоров, счетчиков, регистров и др.; второе — с применением и разработкой универсальных и специализированных многофункциональных логических модулей (МФЛМ) различного назначения.

Учитывая наличие большого числа работ по синтезу СЛУ на основе сумматоров, счетчиков, регистров, специализированных логических модулей, например [19, 21, 26, 35], рассмотрим более подробно вопросы синтеза комбинационных схем на основе мультиплексоров и других элементов.

Построение логических схем на мультиплексорах и вспомогательных элементах обычно ведется в виде древовидных, цепочечных, каскадных структур, отличающихся способами функционального разделения и разложения булевых функций (БФ). Наиболее часто на практике применяется разложение БФ по методу Шеннона, имеющему вид

$$f(x_1, x_2, \ldots, x_n) = \bar{x}_{i1}\bar{x}_{i-2}\ldots\bar{x}_{ik}f_0 \vee \bar{x}_{i}\bar{x}_{i2}\ldots\bar{x}_{ik-1}x_{ik}f_1 \vee \ldots$$
$$\ldots \vee x_{i1}x_{i2}\ldots x_{ik}f_{2^k-1},$$

где $f_0, f_1, \ldots, f_{2^k-1}$ — остаточные функции (ОФ) разложения, которые получаются из функции f путем подстановки констант 0 и 1 вместо переменных множества $\{x_{i1}x_{i2}\ldots x_{ik}\}$: для f_0 имеем $x_{i1} = x_{i2} = \ldots = x_{ik} = 0$; для f_1 имеем $x_{i1} = x_{i2} = \ldots = x_{ik-1} = 0$ $x_{ik} = 1$; для f_{2^k-1} имеем $x_{i1} = x_{i2} = \ldots x_{ik} = 1$.

Разложение БФ является одним из трудоемких этапов проектирования логических схем на мультиплексорах, так как получение экономичного (оптимального) решения связывается с частичным или полным перебором вариантов разложения БФ по определенному числу переменных, причем в зависимости от сложности реализуемых на мультиплексорах булевых функций процесс разложения БФ является многоступенчатым, выполняемым до момента полного сведения получаемых остаточных функций БФ к простейшему виду.

Рассмотрим некоторые из машинно-ориентированных алгоритмов синтеза СЛУ на мультиплексорах с использованием методов разложения БФ по Шеннону и теории чисел.

Алгоритм синтеза СЛУ на мультиплексорах при полном и неполном разложении булевых функций. Для компактности представления булевых функций, записываемых, как правило, в громоздкой двоичной форме, в дальнейшем будем использовать цифровую десятичную форму записи с обозначением ею отдельных термов (конъюнкций) БФ, имеющих место в двоичной форме.

Например, пусть булева функция имеет вид

$$f(x_1,\ldots, x_5)=\bar{x}_1\bar{x}_2\bar{x}_3x_4x_5\vee\bar{x}_1\bar{x}_2x_3\bar{x}_4x_5\vee\bar{x}_1\bar{x}_2x_3x_4\bar{x}_5\vee$$
$$\vee\bar{x}_1x_2\bar{x}_3\bar{x}_4\bar{x}_5\vee\bar{x}_1x_2\bar{x}_3x_4\bar{x}_5\vee\bar{x}_1x_2x_3x_4x_5\vee x_1\bar{x}_2\bar{x}_3\bar{x}_4\bar{x}_5.$$

Тогда в десятичной форме эту БФ можно записать в виде множества $\{r_k\}$:

$$f(x_1,\ x_2,\ldots,x_5)=\{r_k\}=\Sigma(3,\ 5,\ 6,\ 8,\ 10,\ 15,\ 20).$$

С учетом специфики работы мультиплексоров, изложенной в § 1.1, и конструктивных особенностей их реализации с числом управляющих входов q ($q=\overline{2,4}$) и информационных входов, равным 2^q (4, 8, 16), разложение заданной БФ можно вести по двум, трем или четырем переменным. Тогда при построении логической схемы на мультиплексорах эти переменные должны подключаться к управляющим входам, а ОФ разложения — к информационным входам соответствующего МХ. Если образуемые в результате первого шага разложения остаточные функции имеют нетривиальный вид, то процедура разложения каждой из получаемых на очередном шаге ОФ должна повторяться вплоть до момента превращения их в тривиальные, а именно:

$$\Sigma(0),\ \Sigma(1),\ \Sigma(0,\ 1),\ \varnothing\ (\varnothing\text{ — пусто}).$$

Остаточные функции разложения Q_t по последним двум $\{x_{n-1}x_n\}$, трем $\{x_{n-2}x_{n-1}x_n\}$, четырем $\{x_{n-3}x_{n-2}n_{n-1}x_n\}$ переменным из булевой функции $f(x_1,\ x_2,\ldots,x_n)$ могут быть вычислены по формулам

$$Q_t=\bigvee_{\{r_k\}}E\left(\frac{r_k}{2^q}\right),\quad \forall\, r_k: F\left(\frac{r_k}{2^q}\right)=t,$$

где $t=0,\ 1,\ldots,2^q-1$; $E(r_k/2^q)$ — целая часть от деления $r_k/(2q)$; $F(r_k/(2q))$ — остаток от деления $r_k/(2q)$; $\{r_k\}$ — множество термов БФ; q — число переменных, по которым разлагается заданная БФ.

Как отмечалось, при построении логической схемы на МХ, реализующей заданную БФ, возможны два случая: а) $n\leqslant q$; б) $n>q$.

В первом случае БФ реализуется схемой, состоящей из одного мультиплексора, в которой q переменных $(x_{n-q}, \ldots, x_n)$ подключаются к управляющим входам МХ, а на информационные его входы подаются константы 0 (если данный терм в функции отсутствует) или 1 (если он присутствует).

Во втором случае процесс построения логической схемы на мультиплексорах производится по результатам разложения заданной БФ. В результате первого шага разложения исходной БФ $f(x_1, \ldots, x_n)$ по q переменным получается совокупность ОФ, которые зависят уже только от n—q переменных. Последующие шаги

Таблица 3.1

$\{r_k\}$	0	1	3	5	6	6	10	15	20	21	22	25	28	29	30	31
$E(r_k/4)$	0	0	0	1	1	2	2	3	5	5	5	6	7	7	7	7
$F(r_k/4)$	0	1	3	1	2	0	2	3	0	1	2	1	0	1	2	3

разложения уменьшают каждый раз число переменных в ОФ на q, вплоть до получения в процессе разложения ОФ тривиального вида. Таким образом, число шагов разложения БФ соответствует числу ярусов схемы на мультиплексорах с подключением на управляющие входы МХ тех переменных, по которым производилось разложение; на информационные входы МХ последнего яруса подаются одиночные переменные x_i или $\bar{x}_i$, а также сигналы логического нуля (лог. 0) или логической единицы (лог. 1) исходя из вида получаемых тривиальных ОФ:

$$\Sigma(0) \equiv \bar{x}_i; \quad \Sigma(1) \equiv x_i; \quad \Sigma(0, 1) \equiv \text{лог. } 1; \quad \varnothing \equiv \text{лог. } 0.$$

Пример 3.1. Реализовать на мультиплексорах с числом управляющих входов $q=2, 3, 4$ булеву функцию вида

$$f(x_1, \ldots, x_5) = \Sigma(0, 1, 3, 5, 6, 8, 10, 15, 20, 21, 22, 25, 28, 29, 30, 31)$$

и выбрать наилучшую реализацию по критерию минимума аппаратных затрат. В качестве элементной базы использовать МХ 155 серии микросхем.

Согласно приведенному выше алгоритму осуществим разложение заданной БФ по двум, трем и четырем переменным, сводя результаты расчетов в таблицы.

1. Вариант разложения БФ по двум переменным $\{x_4 x_5\}$ приведен в табл. 3.1. Таким образом, на первом шаге разложения БФ получаем следующие ОФ:

$$Q_0 = \Sigma(0, 2, 5, 7); \quad Q_1 = \Sigma(0, 1, 5, 6, 7); \quad Q_2 = \Sigma(1, 2, 5, 7);$$

$$Q_3 = \Sigma\ (0, 3, 7).$$

Разложение БФ продолжим, так как все ОФ имеют нетривиальный вид.

На втором шаге в качестве исходных данных $\{r^1_k\}$ теперь рассматриваются слагаемые каждой из полученных на первом шаге разложения остаточных функций Q_t. Анализируя значения $\{r^1_k\}$, замечаем, что наиболее целесообразно разложение БФ на втором шаге продолжить лишь по двум переменным, так как в противном случае при разложении по трем (и более) переменным схемы получатся менее экономичными (табл. 3.2).

На втором шаге разложения БФ имеем следующие ОФ:

для Q_0 $Q_0^1 = \Sigma(0)$; $Q_1^1 = \Sigma(1)$; $Q_2^1 = \Sigma(0)$; $Q_3^1 = \Sigma(1)$;

для Q_1 $Q_0^1 = \Sigma(0)$; $Q_1^1 = \Sigma(0, 1)$; $Q_2^1 = \Sigma(1)$; $Q_3^1 = \Sigma(1)$;

для Q_2 $Q_0^1 = \varnothing$; $Q_1^1 = \Sigma(0, 1)$; $Q_2^1 = \Sigma(0)$; $Q_3^1 = \Sigma(1)$;

для Q_3 $Q_0^1 = \Sigma(0)$; $Q_1^1 = \varnothing$; $Q_2^1 = \varnothing$; $Q_3^1 = \Sigma(0, 1)$.

Таблица 3.2

Функция	Q_0				Q_1					Q_2				Q_3		
$\{r^1{}_k\}$	0	2	5	7	0	1	5	6	7	1	2	5	7	0	3	7
$E(r^1{}_k/4)$	0	0	1	1	0	0	1	1	1	0	0	1	1	0	0	1
$F(r^1{}_k/4)$	0	2	1	3	0	1	1	2	3	1	2	1	3	0	3	3

Так как ОФ, полученные на втором шаге, являются тривиальными, то процесс разложения БФ заканчивается и ее можно реализовать двухъярусной схемой из МХ с $q=2$. Схемная реализация БФ на МХ типа К155КП2 приведена на рис. 3.1, *а*.

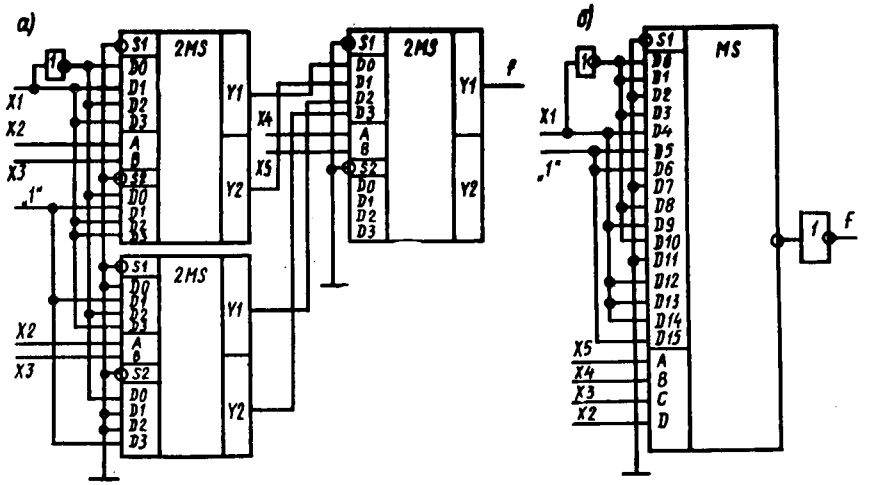


Рис. 3.1. Реализация заданной БФ $f=\Sigma$ (0, 1, 3, 5, 6, 8, 10, 15, 20, 21, 22, 25, 28, 29, 30, 31) схемой из мультиплексоров типа К155КП2 (*а*) и К155КП1 (*б*)

2. Вариант разложения БФ по трем переменным приведен в табл. 3.3.

Таким образом, после первого шага разложения БФ по трем переменным получили следующие ОФ:

$$Q_0 = \Sigma(0, 1);\quad Q_1 = (0, 3);\quad Q_2 = \Sigma(1);\quad Q_3 = \Sigma(0);\quad Q_4 = \Sigma(2, 3);$$

$$Q_5 = \Sigma(0, 2, 3);\quad Q_6 = \Sigma(0, 2, 3);\quad Q_7 = \Sigma(1, 3).$$

Поскольку одна часть ОФ получилась тривиальной (Q_0, Q_2, Q_3), а другая часть (Q_1, Q_4, Q_7) имеет нетривиальный вид, это свидетельствует о нецелесообразности дальнейшего разложения БФ и ее схемной реализации (для окончательной реализации БФ при таком подходе потребуется восемь мультиплексоров).

Таблица 3.3

$\{r_k\}$	0	1	3	5	6	8	10	15	20	21	22	25	28	29	30	31
$E(r_k/8)$	0	0	0	0	0	1	1	1	2	2	2	3	3	3	3	3
$F(r_k/8)$	0	1	3	5	6	0	2	7	4	5	6	1	4	5	6	7

3. Вариант разложения БФ по четырем переменным $\{x_2, x_3, x_4, x_5\}$ приведен в табл. 3.4.

Таким образом, после первого шага разложения БФ по четырем переменным получаем следующие ОФ:

$$Q_0 = Q_1 = Q_3 = Q_8 = Q_{10} = \Sigma(0);\quad Q_4 = Q_9 = Q_{12} = Q_{13} = Q_{14} = \Sigma(1);$$

$$Q_5 = Q_6 = Q_{15} = \Sigma(0, 1);\quad Q_2 = Q_7 = Q_{11} = \varnothing.$$

Поскольку все ОФ получились тривиальными, разложение БФ заканчивается и ее можно реализовать на одном МХ с $q=4$. Схемная реализация заданной БФ на одном мультиплексоре типа К155КП1 приведена на рис. 3.1, *б*.

Таблица 3.4

$\{r_k\}$	0	1	3	5	6	8	10	15	20	21	22	25	28	29	30	31
$E(r_k/16)$	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$F(r_k/16)$	0	1	3	5	6	8	10	15	4	5	6	9	12	13	14	15

В целом анализ способов аппаратной реализации на мультиплексорах различного вида систем БФ показал следующее.

Все множество СБФ условно можно подразделить на два класса: а) случайные (или близкие к ним) СБФ, характеризуемые сложной аналитической записью в виде двоичных таблиц или математических и логических соотношений; б) упорядоченные СБФ с компактной формой записи. Хотя доля упорядоченных СБФ очень мала по сравнению с множеством всех возможных функций, именно они наиболее часто встречаются на практике при синтезе СЛУ. В общем случае эффективность реализации конкретной СБФ зависит от ее вида, способа упорядочения СБФ и вы-

бора типа БИС или СИС для реализации СБФ в качестве элементной базы.

Для реализации на мультиплексорах лучше всего подходят упорядоченные СБФ с числом переменных в конъюнкциях до 9, имеющие «ядро» из двух — четырех переменных, составляющих СДНФ или близкую к ней форму, а остальная часть переменных, входящих в конъюнкции БФ, может отличаться друг от друга.

Процесс построения схем на МХ является многошаговым с необходимостью комбинаторного перебора процедур разложения БФ по различному числу переменных на каждом шаге и построения схемы в виде «дерева» начиная с его «корня» (выходного яруса). Однако технические ограничения серийных МХ с числом управляющих входов $q=\overline{2,4}$ сокращают число вариантов перебора и позволяют эвристически оценить сложность схем с отбрасыванием неэффективных вариантов начиная уже с первых шагов разложения БФ. Структуры из однотипных МХ по сравнению со структурами из МХ различного типа, а также из МХ и других логических элементов являются, как правило, более избыточными.

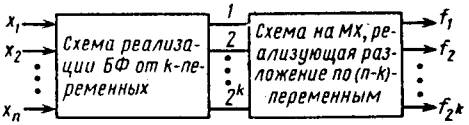


Рис. 3.2. Способ реализации СБФ на мультиплексорах и дополнительной логике при неполном разложении функций

Так, например, значительный выигрыш в аппаратурных затратах получается при неполном разложении БФ и синтезе логической схемы, состоящей из двух частей (рис. 3.2): первой — на МХ, реализующих разложение БФ по k переменным, второй — на других элементах, реализующих ОФ разложения $n-k$ переменных. Реализация второй части на различных типах элементов и микросхем способствует появлению новых методов синтеза логических устройств на основе МХ. Аппаратные затраты на реализацию логических схем с использованием генератора БФ от двух или трех переменных и остальной ее части в виде многоярусной структуры на МХ, например, уже при $7\leqslant n<11$ по сравнению с методом полного разложения БФ дают примерно двойную экономию аппаратурных затрат и увеличение быстродействия схем.

Наиболее экономичные реализации схем на мультиплексорах получаются при минимуме ярусов из мультиплексоров в схеме и расположении МХ в структуре от входа к выходу схемы в порядке убывания значения q. Таким образом, эффективной реализации БФ с 6 переменными можно добиться двухъярусной схемой на МХ с $q=3$ и $q=2$; БФ с 8 переменными — на МХ с $q=4$ и $q=3$; БФ с 9 переменными — на МХ с $q=4$ и $q=4$ и т. д.

Алгоритм синтеза СЛУ на мультиплексорах и элементах И—НЕ, И—ИЛИ—НЕ. Экономичные решения по критерию аппаратурных затрат получаются при синтезе комбинационных схем на

основе мультиплексоров и элементов И—НЕ, И—ИЛИ—НЕ с использованием методов, предложенных в [29].

Задача синтеза логических устройств состоит из двух этапов: а) выбора «наилучшего» способа разбиения множества входных переменных $\mathbf{X}=\{x_1, \ldots, x_n\}$, соответствующего минимальным аппаратным затратам; б) построения схемы в соответствии с найденным разбиением.

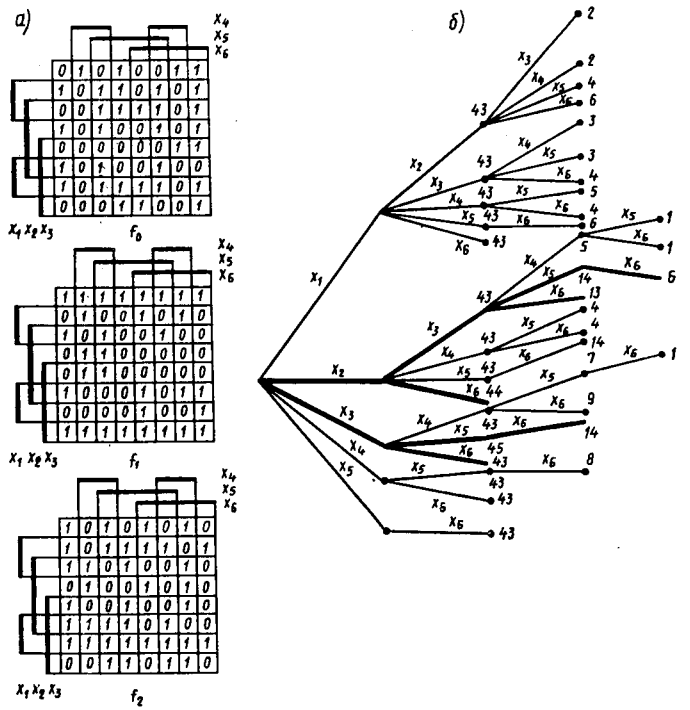


Рис. 3.3. Способ задания СБФ в виде карт Карно (*а*) и логического дерева поиска (*б*) для получения оптимального решения

Из-за громоздкости формализованного описания алгоритмов синтеза дадим содержательную интерпретацию одного из этих методов.

Поскольку процесс выбора «лучшего» разбиения связан с обширным перебором, используется приближенный многошаговый алгоритм. На первом шаге из множества $\mathbf{X}$ отыскивается подмножество $\mathbf{Y}_a$, обладающее максимальным весом. На втором шаге из множества $\mathbf{Y}_a$ отыскивается подмножество $\mathbf{Y}_{a-1}$, обладающее мак-

симальным весом. На последующих шагах аналогичным образом отыскиваются множества $\mathbf{Y}_{a-2}$, $\mathbf{Y}_{a-3}$, ..., $\mathbf{Y}_0$.

Процесс перебора отображается деревом поиска, в котором каждое ребро соответствует некоторой переменной из множества $\mathbf{X}$, а каждая вершина — отличному от нуля весу подмножества, образованного переменными — ребрами, связывающими данную вер-

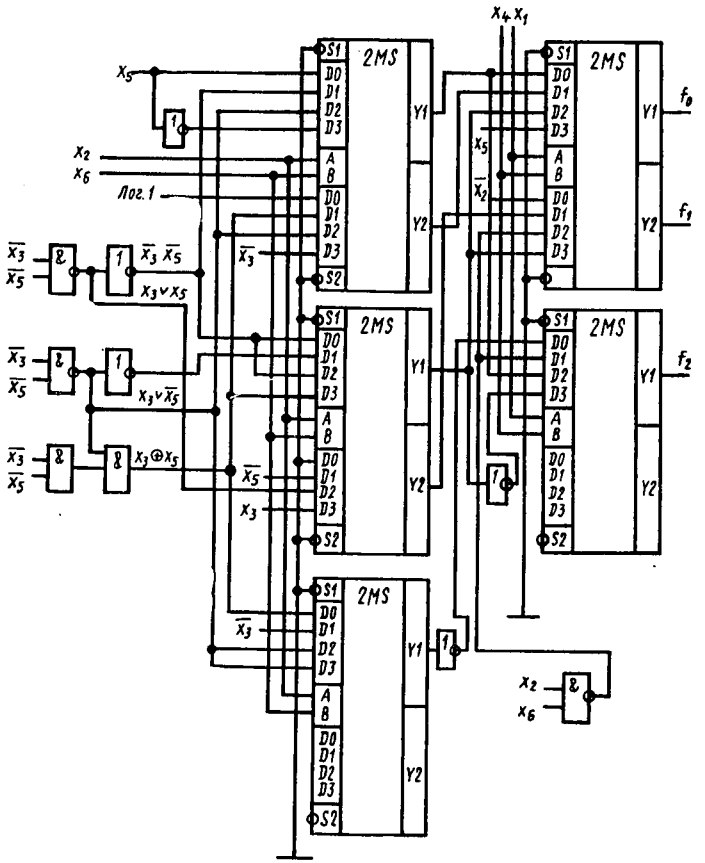


Рис. 3.4. Реализация СБФ $F=\{f_0, f_1, f_2\}$ на базе мультиплексоров К155КП2 и элементов И — НЕ, ИЛИ — НЕ

шину с корнем дерева. Не останавливаясь на правилах обхода логических деревьев поиска, отметим, что по окончании перебора путь, связывающий корень дерева с вершиной яруса, отмеченной максимальным весом, будет представлять множество $\mathbf{Y}_a$.

Для нахождения множества $\mathbf{Y}_{a-1}$ в дереве поиска отсекаются некоторые ветви по следующему правилу: из совокупности путей, связывающих вершины с корнем дерева, сохраняются только те,

для каждого из которых соответствующее ему подмножество из {X} является подмножеством множества $\mathbf{Y}_a$. После этого второй путь, связывающий корень дерева с вершиной яруса, отмеченной максимальным весом, будет представлять множество $\mathbf{Y}_{a-1}$. Такой итеративный процесс отсекания ветвей продолжается до тех пор, пока не будет отыскано множество $\mathbf{Y}_0$.

Схема на МХ синтезируется следующим образом. Сначала для каждой из функций, входящих в СБФ, строится полная древовидная структура, содержащая определенное число МХ в каждом из ярусов. При этом на управляющие входы всех МХ i-го яруса подаются соответствующие переменные. Затем проводится итеративный процесс исключения избыточных МХ (с удалением и добавлением необходимых связей) из ярусов i—1, i—2, ..., 1. Соответствующим образом осуществляется и синтез части схемы на элементах И—НЕ, И—ИЛИ—НЕ.

Пример 3.2. Реализовать на мультиплексорах К155КП2 и элементах И—НЕ, И—ИЛИ—НЕ систему булевых функций $F=\{f_0, f_1, f_2\}$, заданную с помощью карт Карно (рис. 3.3, *а*).

Выбор лучшего разбиения начинается с сокращенного перебора подмножеств из {X} и определения их весов. Дерево поиска приведено на рис. 3.3, *б*, откуда определяем, что $\mathbf{Y}_1=\{x_2, x_3, x_5, x_6\}$. Для нахождения множества $\mathbf{Y}_0$ в дереве поиска отсекаем часть ветвей (оставшиеся ребра на рис. 3.3, *б* изображены утолщенными линиями). Из усеченного дерева поиска определяем, что $\mathbf{Y}_0=\{x_3, x_5\}$.

Часть схемы, реализуемая на элементах И—НЕ, включает следующие булевы функции *:

$$y_1=\bar{x}_3\bar{x}_5;\quad y_2=x_3\vee\bar{x}_5;\quad y_3=x_3\vee\vee x_5;\quad y_4=\bar{x}_3x_5;\quad y_5=x_3\vee x_5$$

В результате применения описанного метода синтеза получается логическая схема (рис. 3.4). Для сравнения заметим, что реализация данной СБФ в базисе серии 155 требует около 18 корпусов микросхем малой степени интеграции, что в 2,5 раза превышает полученную реализацию схемы на МХ и элементах И—НЕ, И—ИЛИ—НЕ.

§ 3.2. СИНТЕЗ СИСТЕМ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ НА ПОСТОЯННЫХ И ПЕРЕПРОГРАММИРУЕМЫХ ЗАПОМИНАЮЩИХ УСТРОЙСТВАХ

Как указывалось в § 1.2, ПЗУ является функциональным элементом с n входами и m выходами, сокращенно ПЗУ (n, m), в котором информация записывается однократно, а считывается многократно и не разрушается в течение всего срока службы. По желанию пользователя в ПЗУ может быть записано множество 2^n логических слов длиной m. ПЗУ широко используются для аппаратной реализации систем булевых функций (СБФ) и обеспечивают значительный выигрыш в быстродействии по сравнению с программной реализацией. Например, программная реализация

* Ниже в выражении переменной y_3 для обозначения сложения по модулю 2 применен знак $\vee\vee$. В данной книге этот знак применен и в других случаях.

БФ на МП К580 составляет десятки микросекунд, в то время как время считывания ПЗУ (n, m) равно десяткам наносекунд.

Если для СБФ вида $y_j=f(x_1, \ldots, x_N)$, $j=\overline{1,M}$; $i=\overline{1,N}$, выполняется соотношение $N\leqslant n$ и $M\leqslant m$, то СБФ может быть реализована на одной БИС ПЗУ. Если $n\geqslant N$, $m<M$, то СБФ реализуется на числе БИС ПЗУ, равном $]M/m[$, где $]a[$ — ближайшее целое, большее a или равное ему, если a — целое число. Если же $N>n$, то применяются различные методы минимизации и декомпозиции функций, позволяющие реализовать СБФ на БИС ПЗУ. При этом необходимое число ПЗУ определяется применяемым эвристическим методом декомпозиции, а минимальной реализации СБФ можно добиться лишь при полном переборе вариантов. Рассмотрим некоторые из этих методов.

Синтез систем булевых функций на БИС ПЗУ. Реализовать СБФ на БИС ПЗУ, например, можно с использованием процедур следующего эвристического алгоритма [16].

1. Разбить исходную СБФ на подфункции, каждая из которых реализуется на одном выходе ПЗУ.
2. Из подмножества полученных подфункций выделить группы, реализуемые одним ПЗУ.
3. Соединить выходы ПЗУ для получения выходов схемы.
4. Составить коды настройки для каждого ПЗУ.

На первом этапе при разбиении исходной СБФ стремятся получить минимум подфункций, зависящих не более чем от n переменных, с целью реализации каждой из них одним ПЗУ. Для получения подфункций образуют термы конъюнкций с одинаковыми признаками. У оставшихся термов признаки получаются попарно различными и каждая конъюнкция является подфункцией.

На втором этапе группируют множества подфункций. Если некоторые из подфункций зависят более чем от n переменных, то производят их дальнейшее разбиение и реализацию некоторых из них на элементах ИЛИ. Группируют остальные подфункции на базе метода таблиц покрытий [29]. В результате выполнения второго этапа определяется схема на ПЗУ, на выходах которых реализуются подфункции, полученные на первом этапе.

На третьем этапе выходы ПЗУ соединяются друг с другом в соответствии с правилом получения необходимых выходов схемы и реализацией заданных функций через дизъюнкции подфункций.

На четвертом этапе определяют коды настройки каждого ПЗУ с уточнением требуемого множества значений выходных функций на всех наборах значений выходных переменных.

Проиллюстрируем применение данного метода на следующем примере.

Пример 3.3. Реализовать на БИС ПЗУ типа К556РТ2 СБФ $\{y_1, \ldots, y_8\}$, описывающую некоторое специализированное устройство:

Таблица 3.5. Разрешенные состояния системы булевых функций

Импликант СБФ	Входные переменные $\tilde{x}_i$ $(i=\overline{1,16})$																Выходы y_j $(j=\overline{1,8})$							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	1	2	3	4	5	6	7	8
1		1			1												1	1				1		
2							1										1							
3		0	0			1				1							1			1	1			
4					1	1	0				0	0	0				1				1			1
5						1	0				1						1				1			
6						1	0					1					1				1			
7						1	0						1				1				1			
8			0	1													1			1	1	1		
9			1	0													1		1		1	1		
10		0	1	1													1		1	1	1	1		
11		1		0													1	1			1	1		
12				1	0	0											1			1		1		
13					0	1					0	0	0				1					1		1
14			0	0	0			0		1								1			1			
15			0	0	0			0						1				1						
16			0	0			1	1										1						

Продолжение табл. 3.5

Импликанг СБФ	Входные переменные $\tilde{x}_i$ $(i = \overline{1, 16})$																Выходы y_j $(j = \overline{1, 8})$							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	1	2	3	4	5	6	7	8
17		0	0	0		1		1										1						1
18		1	0															1						
19			0	0					1									1						
20			1			1													1					
21		0		0	0			0	0	0					1				1					
22		0		0			1		1										1					
23		0		1		1			1										1					
24		0	0			1	0			0			1	0					1					1
25		0	0			0	0		0	0										1				
26		0	0	0	1	0	0													1				1
27		0	0				1			1										1				
28		0	0						1	1										1				
29	0	1	1	0																1				
30		0	0	0	0															1	1		1	1

Продолжение табл. 3.2

Имплицант СБФ	Входные переменные $\tilde{x}_i$ $(i = \overline{1, 16})$																Выходы y_j $(j = \overline{1, 8})$							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	1	2	3	4	5	6	7	8
31		0	0			0	0			0			1	1						1				
32		0	0			0	0			0			1		1					1				
33		0	0			0	0			0			1			1				1				
34		0	0	0				1				0	0	0								1		
35						1	1															1		
36		1				1	0				0	0	0											1
37	0	1	1	1	1	0	0				1													1
38						0	1				0	0	0											1
39				0					1		0	0	0											1
40			0	1							0	0	0											1
41			1	0							0	0	0											1
42		0	1	1							0	0	0											1
43		1		0							0	0	0											1
44		1	1	1		0	1																	1

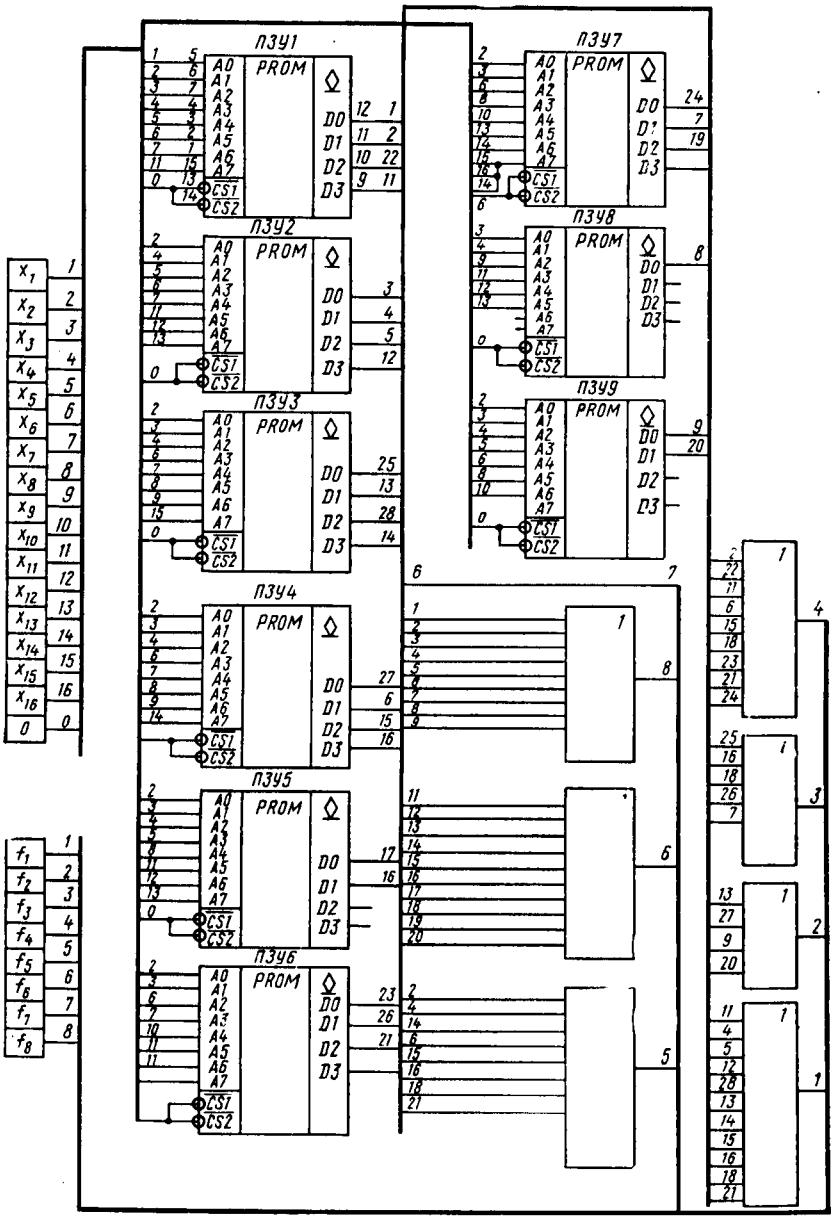


Рис. 3.5. Схема реализации заданной СБФ на ПЗУ К556РТ2

$y_1 = x_2x_5 \vee x_7 \vee \bar{x}_2\bar{x}_3x_6x_{10} \vee x_5x_6\bar{x}_7\bar{x}_{11}\bar{x}_{12}\bar{x}_{13} \vee x_6\bar{x}_7x_{11} \vee x_6\bar{x}_7x_{12} \vee x_6\bar{x}_7x_{13} \vee \bar{x}_3x_4 \vee x_3\bar{x}_4 \vee \bar{x}_2x_3x_4 \vee x_2\bar{x}_4 \vee x_4\bar{x}_5\bar{x}_6 \vee \bar{x}_5x_6\bar{x}_{11}\bar{x}_{12}\bar{x}_{13};$

$y_2 = x_2x_5 \vee \bar{x}_3\bar{x}_4\bar{x}_5\bar{x}_8x_{10} \vee \bar{x}_3\bar{x}_4x_5\bar{x}_8x_{14} \vee \bar{x}_3\bar{x}_4x_7x_8 \vee \bar{x}_2\bar{x}_3\bar{x}_4x_6x_8 \vee x_2\bar{x}_4 \vee x_2\bar{x}_3 \vee \bar{x}_3\bar{x}_4x_9;$

$y_3 = x_3x_6 \vee \bar{x}_2\bar{x}_4\bar{x}_5\bar{x}_8\bar{x}_9\bar{x}_{10}x_{15} \vee \bar{x}_1\bar{x}_4x_7x_9 \vee \bar{x}_2\bar{x}_4x_6x_9 \vee x_3\bar{x}_4 \vee \bar{x}_2x_3x_4 \vee \bar{x}_2\bar{x}_3x_6\bar{x}_7\bar{x}_{10}x_{13}x_{14};$

$y_4 = \bar{x}_2\bar{x}_3\bar{x}_6\bar{x}_7\bar{x}_9\bar{x}_{10} \vee \bar{x}_2\bar{x}_3\bar{x}_4x_5\bar{x}_6\bar{x}_7 \vee \bar{x}_2\bar{x}_3x_7x_{10} \vee \bar{x}_2\bar{x}_3x_9x_{10} \vee \bar{x}_2\bar{x}_3x_6x_{10} \vee \bar{x}_3x_4 \vee \bar{x}_1x_2x_3\bar{x}_4 \vee \bar{x}_2x_3x_4 \vee x_4\bar{x}_5\bar{x}_6 \vee \bar{x}_2\bar{x}_3\bar{x}_4\bar{x}_5 \vee \bar{x}_2\bar{x}_3\bar{x}_6\bar{x}_7\bar{x}_{10}x_{13}x_{14} \vee \bar{x}_2\bar{x}_3\bar{x}_6\bar{x}_7\bar{x}_{10}x_{13}x_{15} \vee \bar{x}_2\bar{x}_3\bar{x}_6\bar{x}_7\bar{x}_{10}x_{13}x_{16};$

$y_5 = \bar{x}_2\bar{x}_3x_6x_{10} \vee x_5x_6\bar{x}_7\bar{x}_{11}\bar{x}_{12}\bar{x}_{13} \vee x_6\bar{x}_7x_{11} \vee x_6\bar{x}_7x_{12} \vee x_6\bar{x}_7x_{13} \vee \bar{x}_3x_4 \vee x_3\bar{x}_4 \vee \bar{x}_2x_3x_4 \vee x_2\bar{x}_4 \vee \bar{x}_2\bar{x}_3\bar{x}_4x_5;$

$y_6 = x_2x_5 \vee \bar{x}_3\bar{x}_4\bar{x}_5\bar{x}_8x_{10} \vee \bar{x}_3x_4 \vee x_3\bar{x}_4 \vee \bar{x}_2x_3x_4 \vee x_2\bar{x}_4 \vee x_4\bar{x}_5\bar{x}_6 \vee \bar{x}_5x_6\bar{x}_{11}\bar{x}_{12}\bar{x}_{13} \vee \bar{x}_2\bar{x}_3\bar{x}_4x_8\bar{x}_{11}\bar{x}_{12}\bar{x}_{13};$

$y_7 = \bar{x}_2\bar{x}_3\bar{x}_4x_5;$

$y_8 = x_2x_5\bar{x}_7\bar{x}_{11}\bar{x}_{12}\bar{x}_{13} \vee \bar{x}_1x_2x_3x_4x_5\bar{x}_6\bar{x}_7x_{11} \vee \bar{x}_2\bar{x}_3\bar{x}_4x_5\bar{x}_6\bar{x}_7 \vee \bar{x}_6x_7\bar{x}_{11}\bar{x}_{12}\bar{x}_{13} \vee \bar{x}_2\bar{x}_3\bar{x}_4x_6x_8 \vee x_5x_6\bar{x}_7\bar{x}_{11}\bar{x}_{12}\bar{x}_{13} \vee \bar{x}_4x_9\bar{x}_{11}\bar{x}_{12}\bar{x}_{13} \vee \bar{x}_3x_4\bar{x}_{11}\bar{x}_{12}\bar{x}_{13} \vee \bar{x}_3\bar{x}_4\bar{x}_{11}\bar{x}_{12}\bar{x}_{13} \vee \bar{x}_2\bar{x}_3x_4\bar{x}_{11}\bar{x}_{12}\bar{x}_{13} \vee x_2\bar{x}_4\bar{x}_{11}\bar{x}_{12}\bar{x}_{13} \vee x_2x_3x_4\bar{x}_6x_7 \vee \bar{x}_5x_6\bar{x}_{11}\bar{x}_{12}\bar{x}_{13} \vee \bar{x}_2\bar{x}_3x_6\bar{x}_7\bar{x}_{10}x_{13}\bar{x}_{14}.$

Преобразуем для наглядности двоичную запись данной СБФ в компактную таблицу разрешенных состояний с указанием состояний входных и выходных переменных и номеров термов, обозначив $\bar{x}_i$ через 0, y_i и x_i через 1 ($i=\overline{1,16}$, $j=\overline{1,8}$) (табл. 3.5).

Один из вариантов реализации СБФ на БИС ПЗУ, полученный по вышеописанному методу, приведен на рис. 3.5.

Синтез микропрограммных автоматов на ПЗУ и ППЗУ. Синтез СЛУ на основе принципа микропрограммного управления с использованием ПЗУ или ППЗУ позволяет повышать регулярность структуры проектируемых устройств, их надежность и диагностируемость. Так, например, типовая обобщенная структура микропрограммного автомата (рис. 3.6, *а*), состоящая из регистра микрокоманд (РМК), дешифратора (Дш), матриц внешних микроопераций (М1), логических условий (М2) и формирования кода последующих микрокоманд (М3), может быть реализована на одной или нескольких БИС ПЗУ или ППЗУ.

В общем случае задача синтеза микропрограммного автомата (МА) на БИС ПЗУ (рис. 3.6, *б*) сводится к записи каким-либо образом в ячейки ЗУ алгоритма функционирования этого автомата. В ячейках ЗУ может содержаться информация о значениях переменных $\{z_{A1}, \ldots, z_{An}\}$, $\{z_{P1}, \ldots, z_{Pm}\}$, $\{z_{B1}, \ldots, z_{Bk}\}$, которая выдается с выхода ПЗУ после подачи сигнала считывания в виде набора слов $\langle \delta_{A1}, \ldots, \delta_{An}, \delta_{P1}, \ldots, \delta_{Pm}, \delta_{B1}, \ldots, \delta_{Bk} \rangle$, где

$$\delta_{Av} = \begin{cases} 1, & \text{если должен быть выполнен оператор } A_v, \\ 0 & \text{в противном случае;} \end{cases}$$

$$\delta_{Pu} = \begin{cases} 1, & \text{если должно быть проверено логическое условие } P_u, \\ 0 & \text{в противном случае;} \end{cases}$$

$$\delta_{Bi} = \begin{cases} 1, & \text{если должен быть возбужден } i\text{-й элемент памяти,} \\ 0 & \text{в противном случае.} \end{cases}$$

В состав схемы вычисления адреса (СВА), реализуемого на ПЗУ МА, могут входить также дешифраторы внешних микроопераций $\{z_{A1}, \ldots, z_{An}\}$, логических условий $\{z_{P1}, \ldots, z_{Pm}\}$ и внутренних

состояний $\{z_{B1}, ..., z_{Bk}\}$, особенно при ограниченном числе входов и выходов серийных БИС ПЗУ. Рассмотрим особенности синтеза микропрограммных автоматов на ППЗУ на ряде конкретных примеров.

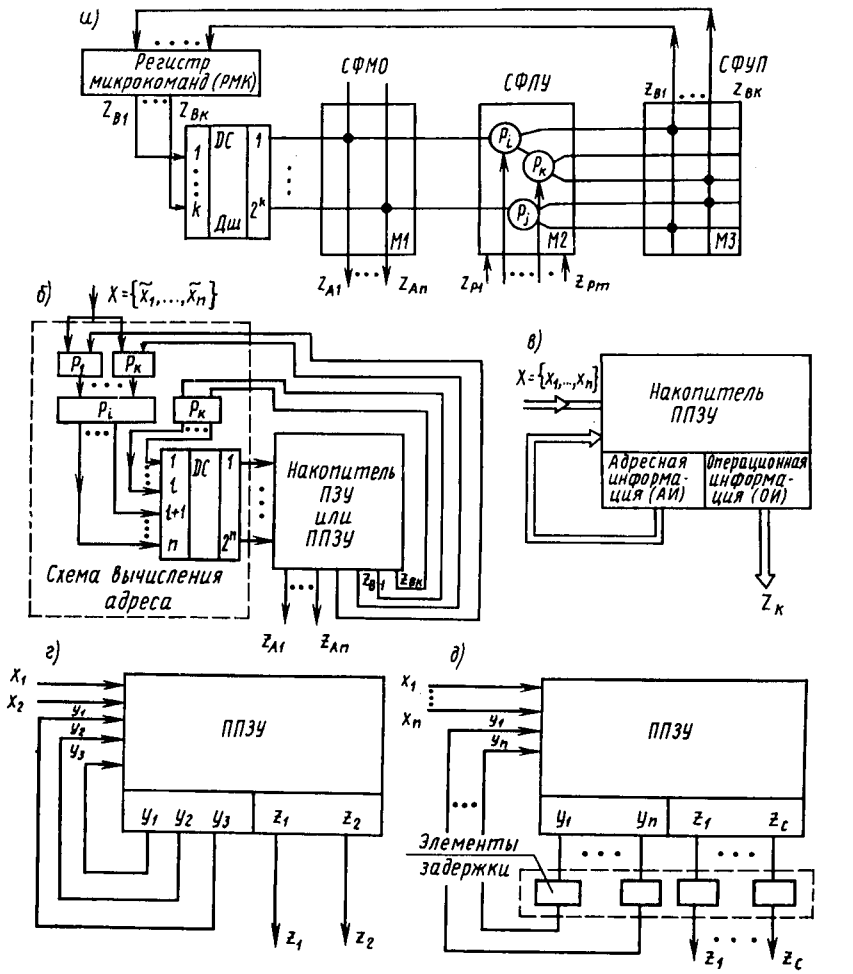


Рис. 3.6. Типовые схемы реализации микропрограммных автоматов (МА) на ПЗУ и ППЗУ:

а — обобщенная структура МА; *б* — способ реализации МА на ППЗУ; *в* — *д* — реализация МА на ППЗУ при вырожденной СВА (*в*) и конкретных условиях (*г*, *д*) для примера 3.5, СФМО, СФЛУ, СФУП — схемы формирования микроопераций, логических условий, условных переходов соответственно

Пусть требуется реализовать на ППЗУ микропрограммный автомат, алгоритм функционирования которого описывается таблицей переходов (табл. 3.6), в которой введены следующие обозна-

чения: $Q=\{q_1, \ldots, q_4\}$ — множество внутренних состояний; $X=\{x_1x_2\}$ — множество входных сигналов; $Z=\{z_1, z_2, z_3\}$ — множество выходных сигналов; $\delta: X \times Q$ — функции переходов, $w: Q \times X$ — функции выходов, значения которых выражаются содержимым клеток таблицы переходов.

Если представить фрагмент таблицы переходов МА применительно к входу x_1 в виде микропрограммы, записываемой в ППЗУ, то функциям переходов δ будет соответствовать адресная часть, а функциям выходов w — операционная часть микрокоманды (табл. 3.7). Следовательно, в типовой структуре МА на ППЗУ (рис. 3.6, *б*) ППЗУ используется для запоминания таблицы переходов автомата, при этом каждое кодовое слово ППЗУ состоит из двух полей: поля, определяемого функцией переходов δ и представляющего следующее состояние; поля, определяемого функцией выходов w и представляющего выходной сигнал $z_{Aj} | j=\overline{1, n}$. Пер-

Таблица 3.6

Состояние Q	Входные сигналы	
	x_1	x_2
q_1	q_1/z_2	q_1/z_1
q_2	q_3/z_1	q_3/z_2
q_3	$q_3/—$	q_2/z_1
q_4	q_2/z_3	$q_4/—$

Таблица 3.7

Микрокоманда для входа x_1	
адресная часть δ_i	операционная часть w_i
q_1	z_2
q_3	z_1
q_3	—
q_2	z_3

вая часть кодового слова вместе с входным сигналом x_j образует сигнал обратной связи, поступающей на схему вычисления адреса (СВА), определяющую следующий адрес. Вторая часть кодового слова используется в качестве выходного сигнала управления.

Выбор способа хранения таблицы переходов в ППЗУ влияет на сложность СВА, которая особенно возрастает при минимизации кодов адресов ЗУ.

Однако с точки зрения повышения скорости обработки информации и регулярности структуры ЗУ лучше применять избыточные ППЗУ с упрощенной СВА. При введении в адресное слово всех строк таблицы переходов и каждого из входных сигналов в отдельности СВА полностью вырождается, а типовая структура МА реализуется непосредственно на ППЗУ (рис. 3.6, *в*). Однако при непосредственной реализации на ППЗУ функционирование МА может оказаться неустойчивым.

Пример 3.4. Необходимо исследовать правильность функционирования МА, реализуемого на ППЗУ и задаваемого кодированной таблицей переходов (табл. 3.8),

Если заданную таблицу переходов поместить в ППЗУ, то запись будет иметь вид, приведенный в табл. 3.9. В функционировании такого МА могут встретиться нежелательные ситуации, когда при сохранении значений входных сигналов могут изменяться адреса, вызывая постоянную смену выходных сигналов. Пусть, например, выполняется условие $\tilde{x}_1\tilde{x}_2=01$, $\tilde{y}_1\tilde{y}_2=00$ (поз. 5). При этом адрес в ППЗУ станет 0100, а кодовое слово будет 0111. В этом слове два левых разряда

Таблица 3.8

Код состояний $\tilde{y}_1\tilde{y}_2$	Коды входных сигналов $(\tilde{x}_1\tilde{x}_2)$			
	00	01	10	11
00	00/11	01/11	01/11	00/00
01	10/—	11/01	11/01	01/10
10	11/—	10/—	00/00	01/10
11	11/—	11/00	10/10	11/11

равны 01, поэтому при входной комбинации $x_1x_2=01$ они поступают по цепи обратной связи на вход ППЗУ (рис. 3.6, *в*) и адрес изменится на 0101. Это вызовет появление следующего кодового слова 1101 (поз. 6) и установку нового адреса 0111 (поз. 8), который приведет к появлению нового кодового слова 1100. После этого установившийся адрес 0111 приведет к сохранению кодового слова 1100 и появлению устойчивого выходного сигнала 00.

Таблица 3.9

№ поз.	Адрес $\tilde{x}_1\tilde{x}_2\tilde{y}_1\tilde{y}_2$	Кодовое слово: адрес (последующее состояние q_i)	Кодовое слово: выходной сигнал z_i	№ поз.	Адрес $\tilde{x}_1\tilde{x}_2\tilde{y}_1\tilde{y}_2$	Кодовое слово: адрес (последующее состояние q_i)	Кодовое слово: выходной сигнал z_i
1	0000	00	11	9	1000	01	11
2	0001	10	—	10	1001	11	01
3	0010	11	—	11	1010	00	00
4	0011	11	—	12	1011	10	10
5	0100	11	11	13	1100	00	00
6	0101	01	01	14	1101	01	10
7	0110	10	—	15	1110	01	10
8	0111	11	00	16	1111	11	11

Пусть имеет место поз. 9, когда адрес ППЗУ равен 1000 и выходное кодовое слово равно 0111. Адресная часть этого слова 01 появляется на входе ППЗУ с установкой нового адреса 1001, который опять меняет кодовое слово на 1101 (поз. 10). Затем установится адрес 1011 (поз. 12) с появлением слова 1010. Оно вызывает переход в поз. 11, затем в поз. 9, и далее перечисленные действия повторяются по циклу.

Кроме отмеченных недостатков, в данной реализации МА на ППЗУ при установке адресов возможны ошибки в срабатывании выходов из-за критических

переходов. Пусть, например, адрес автомата соответствует поз. 15, в которой кодовое слово равно 0110. Оно должно вызывать появление адреса 1101 (поз. 14), однако из-за разброса в изменении значений отдельных адресных линий могут

Таблица 3.10. **Алгоритм функционирования микропрограммного автомата**

№ состояния	Входные сигналы $\{x_1, x_2\}$			
	00	01	10	11
1	**1**/00	3/10	5/00	**1**/10
2	1/00	2/10	2/11	3/01
3	**3**/11	4/00	5/00	**3**/01
4	**4**/01	**4**/00	2/11	1/10
5	4/01	4/00	5/00	1/10

Примечание. Цифры, выделенные полужирным шрифтом, означают устойчивые состояния автомата.

появиться адреса как 1100 (поз. 13), при котором будет выдан выходной устойчивый сигнал 00, так и 1111 (поз. 16), при котором будет выдан противоположный выходной сигнал 11. Следовательно, из-за критических состязаний сигналов вместо правильного кодового слова 1101 (поз. 14) могут считываться другие кодовые слова 1100 (поз. 13) или 1111 (поз. 16) с ошибочными выходными сигналами.

Таблица 3.11

№ состояния	Коды состояний $\tilde{y}_1\tilde{y}_2\tilde{y}_3$	Коды входных сигналов $\tilde{x}_1\tilde{x}_2$			
		00	01	10	11
1	(000)	000/00	011/10	100/00	000/10
2	(011)	000/00	011/10	011/11	101/01
3	(101)	101/11	110/00	100/00	101/01
4	(110)	110/01	110/00	011/11	000/10
5	(100)	110/01	110/00	100/00	000/10
6	(001)	000/00	011/10	—/—	101/01
7	(010)	000/00	011/10	011/10	000/10
8	(111)	—/—	110/00	011/11	101/01

Для устранения этого недостатка следует применять специальное распределение состояний, обеспечивающее минимум числа переменных, длины кодового слова и размерности ППЗУ при максимальном быстродействии выборки из ППЗУ.

Пример 3.5. Пусть алгоритм функционирования МА задается корректной таблицей переходов (табл. 3.10). Тогда кодированная таблица переходов, составленная с учетом распределения состояний, обеспечивающего отсутствие критических состязаний сигналов при изменениях входных переменных $\tilde{x}_1\tilde{x}_2$, соответствует табл. 3.11, а таблица записи алгоритма функционирования автомата в ППЗУ — табл. 3.12.

В структуре МА на ППЗУ (рис. 3.6, *г*) в этом случае уже не возникают неустойчивые выходные сигналы при произвольной дисциплине смены входных

сигналов. Кроме того, при однократном изменении входного сигнала обязательно происходит только одно изменение адреса, включая случаи одновременного изменения сигналов на нескольких адресных линиях.

С целью устранения возможных всплесков выходных сигналов, вызываемых статическими и динамическими шумами, на выходах ППЗУ следует устанавливать элементы задержки (рис. 3.6, *д*).

Таблица 3.12. **Запись алгоритма функционирования автомата перепрограммируемой памяти**

Адрес $\sim x_1 \sim x_2 \sim y_1 \sim y_2 y_3$	Кодовое слово		Адрес $\sim x_1 \sim x_2 \sim y_1 \sim y_2 y_3$	Кодовое слово	
	адресная часть	операционная часть		адресная часть	операционная часть
00000	000	00	10000	100	00
00001	000	00	10001	—	—
00010	000	00	10010	011	11
00011	000	00	10011	011	11
00100	110	01	10100	100	00
00101	101	11	10101	100	00
00110	110	01	10110	011	11
00111	—	—	10111	011	11
01000	011	10	11000	000	10
01001	011	10	11001	101	01
01010	011	10	11010	000	10
01011	011	10	11011	101	01
01100	110	00	11100	000	10
01101	110	00	11101	101	01
01110	110	00	11110	000	10
01111	110	00	11111	101	01

Минимальный промежуток в смене адресов (или входных сигналов) T_{min} должен быть равен

$$T_{min} \approx \tau_{з\,max} + \tau_{в},$$

где $\tau_в$ — время выборки слова из ППЗУ; $\tau_{з\,max}$ — время задержки наиболее инерционного элемента задержки.

В целом МА, реализуемый на ППЗУ, характеризуется регулярностью структуры, высоким быстродействием, малыми аппаратурными затратами и корректным функционированием при состязаниях входных или адресных сигналов, кратковременных всплесках выходных сигналов, вызываемых статическими и динамическими шумами.

§ 3.3. СИНТЕЗ СИСТЕМ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ НА БИС ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ МАТРИЦ И ЗАПОМИНАЮЩИХ УСТРОЙСТВ

Большие интегральные схемы ПЛМ и ПЗУ на сегодня являются распространенными элементами, применяемыми при проектировании цифровых управляющих автоматов и систем логического управления разнообразного назначения. Рассмотрим основные методы синтеза логических устройств на их основе.

Реализация комбинационных схем на БИС ПЗУ и ПЛМ. Комбинационные схемы, описываемые разнообразными видами случайных или упорядоченных СБФ, могут быть эффективно реализованы на БИС ПЗУ или ПЛМ. Рассмотрим основные типовые схемотехнические решения по реализации таких СБФ с выводом приближенных оценок сложности.

Для реализации на ПЗУ или ПЛМ упорядоченных СБФ вида $y_j=f(x_1,\ldots,x_N)$, $j=\overline{1,M}$, с числом термов в каждой функции, равным K ($K\ll 2^N$), в общем случае требуется не менее $\log_2(2^{N\cdot M})=N\cdot M$ элементов настройки, занимающих основную площадь кристалла БИС. Данный параметр может быть использован в качестве критерия стоимости реализации СБФ или приближенной оценки аппаратурных затрат W.

При реализации упорядоченных СБФ на БИС ПЗУ и ПЛМ аппаратурные затраты W соответственно составят:

для ПЗУ $W=C1+2^NM$;

для ПЛМ $W=C2+2NK+MK=C2+K(2N+M)$, где $C1$ и $C2$ — некоторые константы, причем $C1<C2$.

Отсюда следует, что при реализации упорядоченных СБФ $W_{\text{ПЛМ}}<W_{\text{ПЗУ}}$.

Для случайной или близких к ней СБФ может оказаться, что число термов в каждой функции $K\approx 2^N$, тогда при реализации случайной СБФ на БИС ПЛМ и ПЗУ имеем

$$W_{\text{ПЗУ}}=C1+2^NM;\quad W_{\text{ПЛМ}}=C2+2N2^N+M2^N=C2+2^N\ (2N+M).$$

Отсюда следует, что при реализации разновидностей случайных СБФ имеем $W_{\text{ПЛМ}}>W_{\text{ПЗУ}}$.

Таким образом, полученные оценки аппаратурных затрат W показывают, что реализацию упорядоченных СБФ целесообразнее вести на БИС ПЛМ, а случайных СБФ — на БИС ПЗУ.

Реализация СБФ вида $y_j=f(x_1,\ldots,x_N)$, $j=\overline{1,M}$, с числом термов в каждой функции, равным K, условно обозначаемой как СБФ (N, M, K), на БИС ПЗУ с параметрами ПЗУ (n, m) и БИС ПЛМ с параметрами ПЛМ (s, l, r) может вестись следующим образом.

Если выбранные ПЛМ (s, l, r) или ПЗУ (n, m) и реализуемая СБФ (N, M, K) имеют такие параметры, что $N\leqslant s$ ($N\leqslant n$), $M\leqslant l$ ($M\leqslant m$), $K\leqslant r$, то требуемая СБФ может легко реализоваться на одной БИС ПЛМ или ПЗУ. Для этого в БИС ПЛМ необходимо провести программирование соответствующих элементов настройки в матрицах М1 и М2 без использования программируемых инверторов в матрице М3, так как сокращать число термов K нет необходимости. Аналогично в БИС ПЗУ осуществляется программирование структуры путем пережигания и непережигания соответствующих перемычек в матрице ПЗУ в соответствии с вхождением конъюнкций в состав СБФ.

Если выбранные ПЛМ (s, l, r) или ПЗУ (n, m) и реализуемая СБФ (N, M, K) таковы, что $N>s$ ($N>n$), $M>l$ ($M>m$), $K>r$, то для реализации требуемой СБФ потребуется несколько БИС ПЛМ или ПЗУ, соединяемых между собой определенным образом.

Соединение БИС ПЛМ (ПЗУ) друг с другом для реализации СБФ с параметрами N, M, K, превышающими соответствующие параметры исходных ПЛМ (s, l, r) или ПЗУ (n, m), называют *расширением* ПЛМ (ПЗУ) по *входам* $N>s$ ($N>n$), *выходам* $M>l$ ($M>m$) и *термам* $K>r$ соответственно.

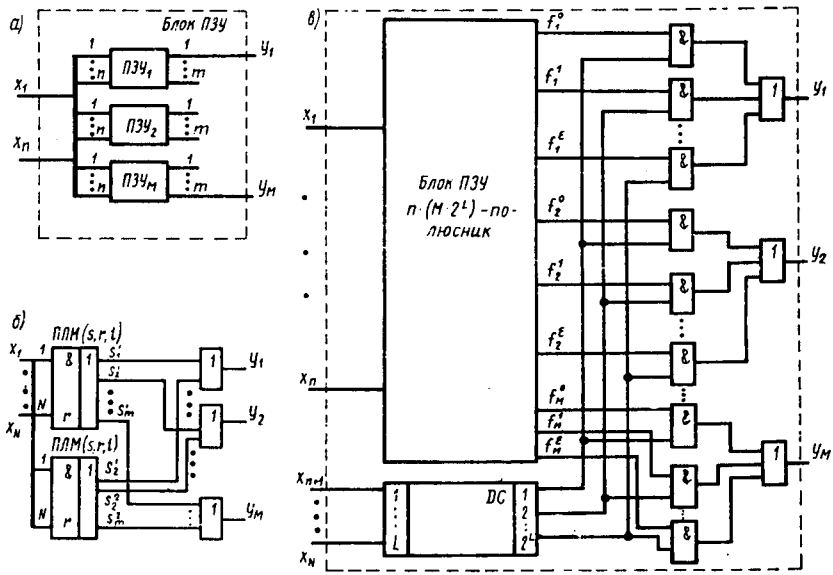


Рис. 3.7. Способы расширения и реализации СБФ на ПЗУ и ПЛМ:
а — расширение по выходам; *б* — расширение по термам; *в* — способ реализации СБФ N, M, K на БИС ПЗУ (n, m) при $N>n$, $M>m$

Расширение ПЛМ (ПЗУ) по выходам при $M>l$ ($M>m$) организуется достаточно просто путем объединения соответствующих входов ПЛМ (s, l, r) или ПЗУ (n, m). Матрицы М1 у всех q ПЛМ ($q=]M/l[$) при расширении по выходам q раз совпадают, а матрицы М2 всех q ПЛМ позволяют реализовать систему ql функций от s переменных с r термами.

Аналогично при реализации СБФ на БИС ПЗУ при $M>m$ используется q ПЗУ ($q=]M/m[$), объединяющих одноименные входы ПЗУ и образующих блок ПЗУ согласно рис. 3.7, *а*.

При расширении ПЛМ (s, l, r) по термам ($K>r$) необходимо объединить как входы, так и выходы q_2 ПЛМ ($q_2=]K/r[$). Следует отметить, что соединение одноименных входов и выходов q_2

ПЛМ между собой осуществляется с помощью проводного ИЛИ в предположении, что используемые ПЛМ обладают такой возможностью. При недопустимости соединения выходов ПЛМ подобным способом они объединяются с помощью элементов ИЛИ, ПМВ, ПМЛ и др. [18]. На N входов каждой ПЛМ (s, l, r), $N \leqslant s$, подаются значения входных переменных из множества $\{x_1, ..., x_N\}$, на M выходах этих ПЛМ, $M \leqslant l$, формируются значения выходных переменных из множества $\{y_1, ..., y_M\}$. Число же требуемых промежуточных шин в группе из q_2 ПЛМ с учетом того, что одноименные входные и выходные шины во всех ПЛМ объединены, определяется значением $q_2 r \geqslant K$.

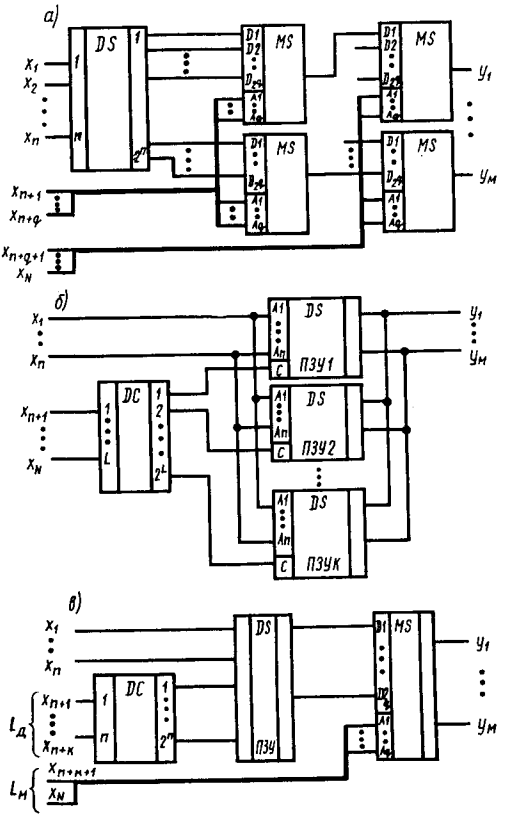


Рис. 3.8. Типовые реализации СБФ (N, M, K) на БИС ПЗУ:

а — на ПЗУ с двухъярусным мультиплексированием выходов ПЗУ; *б* — на ПЗУ с использованием входа «Выборка кристалла» (CS) и дешифратора; *в* — на ПЗУ, дешифраторах и мультиплексорах

Реализация СБФ на ПЛМ (s, l, r) при $K > r$ в общем случае предполагает совместную минимизацию СБФ для уменьшения K, являющуюся сложной задачей особенно при большом числе входных переменных. Большой эффект от совместной минимизации СБФ получают при учете возможности инвертирования функций, однако из-за громоздкости решения задачи в этом случае обычно применяют приближенные эвристические алгоритмы. Известен ряд методов синтеза СБФ в предположении, что совместная минимизация проведена, но полученная величина $K > r$. Методы основываются на перегруппировке конъюнкций СБФ в совокупности конъюнкций при общем числе термов в каждой совокупности не более K, учитывают возможность дальнейшей совместной минимизации и получения общей типовой структуры реализации, приведенной на рис. 3.7, *б*.

Сложнее осуществлять реализацию на ПЛМ (s, l, r) или ПЗУ (n, m) СБФ с числом входных переменных N, превышающим число входов БИС, т. е. $N>s$ ($N>n$). В этом случае используются методы, основанные на дизъюнктивном разложении функций по Шенному (см. § 3.1).

Ниже приводится ряд типовых структур по реализации СБФ на основе БИС ПЗУ и серийных СИС.

Если количество избыточных переменных в заданной СБФ равно $L=N-n$, то все функции этой СБФ разлагаются сначала по L переменным ($\tilde{x}_{n+1}\tilde{x}_{n+2}\dots\tilde{x}_N$). В результате разложения появляется совокупность $M\cdot 2^L$ функций от n переменных, которые могут быть реализованы на $A1=M\cdot 2^L$ БИС ПЗУ. Полностью реализация СБФ (N, M, K) выполняется на основе $M\cdot 2^L$ БИС ПЗУ и дешифратора, имеющего L входов (рис. 3.7, *в*) или группы мультиплексоров.

При реализации СБФ на базе БИС ПЗУ и мультиплексоров на управляющие входы последних подают L входных переменных ($x_{n+1}\dots x_N$). Для полной реализации СБФ (N, M, K) потребуется M мультиплексоров с L управляющими входами. Если число управляющих входов (l_M) мультиплексора $l_M<L$, то используют схему с многоярусным мультиплексированием (рис. 3.8, *а*). В такой структуре аппаратные затраты составляют

$$W=\beta A1+\gamma P1,$$

где $A1$ ($P1$) — количество БИС ПЗУ (мультиплексоров), β, γ — их весовые коэффициенты (например, стоимость).

Упростить реализацию СБФ (N, M, K) можно благодаря наличию у БИС ПЗУ входа «Выборка кристалла» (ВК), трехстабильного выхода и других особенностей. В типовой структуре (рис. 3.8, *б*) входные переменные $\{x_1,\dots,x_n\}$ поступают на входы всех ПЗУ, а остальные L переменных $\{x_{n+1},\dots,x_N\}$, по которым производится разложение, поступают на дешифратор, выходы которого подсоединяются ко входам ВК ПЗУ. Одноименные выходы всех ПЗУ объединяются с реализацией функции проводное ИЛИ. Если количество входов в дешифраторе $l_д<L$, то может использоваться многоярусная дешифрация.

Количество БИС ПЗУ в такой структуре оценивается величиной $A2=2^L\cdot]M/m[$, а общие аппаратные затраты W составляют:

$$W=\alpha Д2+\beta A2,$$

где $Д2$ — количество дешифраторов, α — их весовой коэффициент.

При определенных соотношениях СБФ (N, M, K) и ПЗУ (n, m) можно получить экономичную реализацию СБФ с использованием смешанной структуры (рис. 3.8, *в*), объединяющей в себе БИС ПЗУ, дешифраторы и мультиплексоры.

В целом при реализации разнообразных СБФ на базе СИС и БИС ПЗУ или ПЛМ следует предварительно оценивать выбор

вариантов типовых структур с учетом вида задания и класса реализуемых функций, требуемого объема аппаратурных затрат и др.

Эффективность реализации СБФ на типовых структурах БИС ПЛМ существенно зависит от способов представления, упрощения и минимизации СБФ с учетом функциональной связности переменных, возможности программирования функций в прямом и инверсном виде, технологии и особенностей изготовления БИС. Более подробно методы синтеза СБФ на БИС ПЛМ и рекомендации по их использованию приведены в [19].

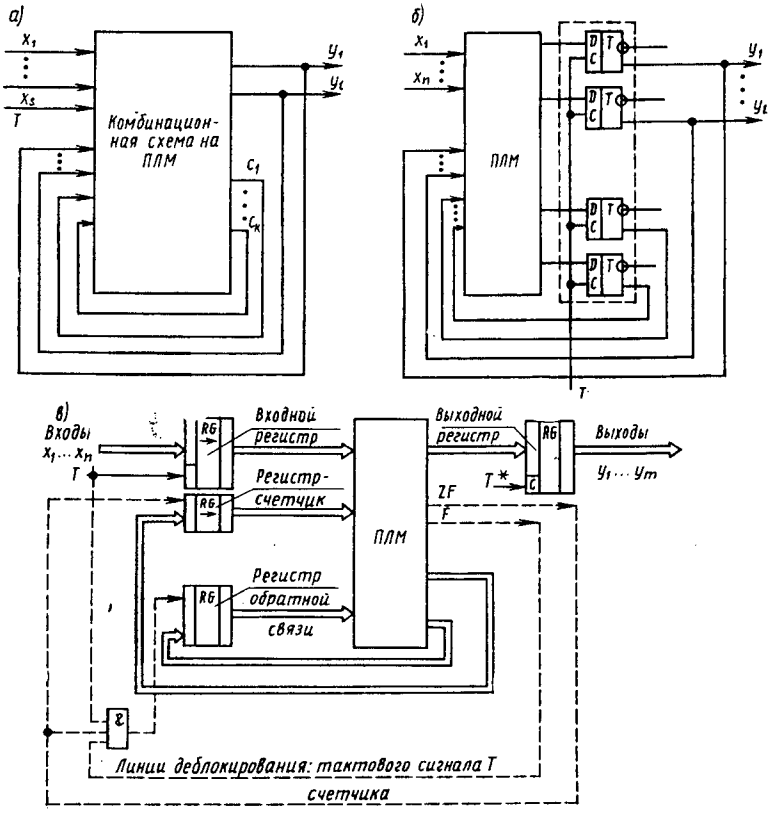


Рис. 3.9. Реализация управляющего автомата на БИС ПЛМ:
а — на комбинационной ПЛМ с управляющим входом *Т*; *б* — на ПЛМ и регистре из *D*-триггеров; *в* — на ПЛМ и тактируемых регистрах

Синтез управляющих автоматов на основе ПЛМ и ПЗУ. Реализация на ПЛМ управляющих автоматов (УА) или схем с памятью в отличие от комбинационных схем требует применения таких методов, которые учитывают высокое быстродействие ПЛМ и обеспечивают устойчивость проектируемых устройств ко всем ви-

дам состязаний сигналов при произвольных конечных соотношениях естественных задержек в ПЛМ и других элементах структуры.

Как известно, в схеме асинхронного УА возможны логические и функциональные состязания. К функциональным относятся состязания между входными сигналами (входные состязания), между промежуточными сигналами (гонки) и между входными и промежуточными сигналами (существенные состязания) [22, 29, 44, 45].

Простейшую реализацию УА получают путем соединения некоторого числа выходов ПЛМ с ее входами (рис. 3.9, *а*). Устранение входных состязаний в такой реализации обеспечивается единичной дисциплиной изменения входных сигналов при смене состояний входов автомата. Устранение гонок производится за счет противогоночного или другого специального кодирования внутренних состояний, логических состязаний — за счет введения избыточных конъюнкций в систему ДНФ, описывающую автомат. Состязания между элементами памяти обычно не влияют на устойчивость поведения УА, поэтому кодирование внутренних состояний в основном преследует цель получения наиболее простой логической структуры. Однако существенные состязания в общем случае устранить нельзя, не обеспечив задержку изменения промежуточных сигналов относительно входных на определенную величину, например, путем введения управляющих сигналов, реализуемых с помощью искусственных задержек или элементов памяти (ЭП), устанавливаемых в цепи обратной связи УА.

При отсутствии ограничений на число изменяемых входных сигналов в процессе смены входных состояний можно использовать структуру автомата с управляющим сигналом T (рис. 3.9, *а*), обеспечивающим устойчивое функционирование УА на ПЛМ с обратными связями без включения в них специальных ЭП или с включением регистра из D-триггеров (рис. 3.9, *б*). Сигнал T меняет свое значение 0 на 1 с определенной задержкой относительно окончания изменения набора входных сигналов. Изменение промежуточных и выходных сигналов допускается лишь в интервале времени, когда сигнал $T=1$. Устойчивость к входным и существенным состязаниям достигается путем выбора подходящего способа кодирования (противогоночного, соседнего, единичного и др.) при определенных условиях существования входных сигналов.

Реализации условий устойчивости с устранением дребезга входных сигналов можно добиться схемой УА на ПЛМ (рис. 3.9, *в*) с добавлением входного и выходного регистров, регистра-счетчика и регистра обратной связи при следующем принципе ее функционирования: при реализованном условии перехода формируется дополнительный сигнал $ZF=1$, поступающий на вход счетчика. В каждом следующем такте состояние счетчика увеличива-

ется при выполнении очередного условия перехода и повторной генерации сигнала $ZF=1$. При нереализации условия перехода в процессе считывания, например из-за дребезга входных состояний или другой причины, сигнал $ZF=0$.

Желаемый переход происходит, если счетчик находится в конечном состоянии, при этом сигнал $F=1$ и условие перехода, проверяемое сигналом ZF, выполнено. Новое состояние перехода заносится в регистр обратной связи при помощи следующего тактового сигнала.

Такой принцип работы УА позволяет сократить процесс считывания для переходов, которые необходимо реализовать за ограниченное время. Для отдельных выходных переменных счетчика либо указывается условие перехода, равное 1, либо предусматривается деблокирование тактового сигнала сигналом $F=1$.

В целом при использовании данной схемы в структурной таблице УА необходимо дополнительно указывать условия перехода, устойчивости и выдачи информации, а для устранения последствий дребезга на входах ПЛМ применять счетчик.

§ 3.4. СИНТЕЗ ПОДСИСТЕМ ПАМЯТИ МПСУ НА БИС ЗАПОМИНАЮЩИХ УСТРОЙСТВ

Полупроводниковые БИС ЗУ широко применяются в ЭВМ, специализированных вычислителях, периферийных устройствах, микропроцессорных системах, устройствах промышленной автоматики. Проектируемые на основе БИС ЗУ устройства подразделяются на *системы ЗУ* (устройства памяти, включающие в себя разные по функциональному назначению блоки ЗУ); *блоки ЗУ* (законченные конструктивно и однородные по функциональному назначению устройства, обеспечивающие заданный информационный объем и быстродействие, состоящие из накопителя, блока местного управления, интерфейса и схем контроля); *модули ЗУ* (части блока ЗУ, позволяющие получить оптимальные характеристики по информационной емкости и быстродействию для данного типа БИС).

Модули ЗУ представляют собой функционально законченные устройства, обеспечивающие заданный информационный объем и быстродействие, а при необходимости и наращивание объема ЗУ (по адресам и разрядам).

При построении модуля на БИС ЗУ необходимо выбрать тип БИС ЗУ и способ организации накопителя (соотношения числа слов и разрядов), влияющего на основные характеристики модуля ЗУ: емкость, быстродействие, энергопотребление, надежность и др. При объединении БИС ЗУ необходимо предусматривать установку соответствующих схем согласования нагрузок по входам и выходам.

Известны три способа увеличения информационной емкости накопителя модуля ЗУ (рис. 3.10): увеличение разрядности слов; увеличение количества слов; увеличение разрядности и количества слов [24].

Основными факторами, влияющими на структуру проектируемых модулей памяти, являются входные и выходные характеристики используемых БИС ЗУ и согласующих схем, а также их временные характеристики.

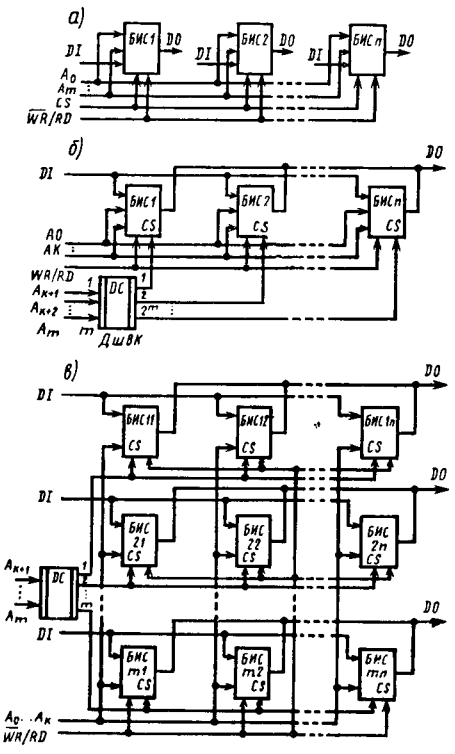


Рис. 3.10. Способы наращивания информационной емкости модулей ОЗУ:
а — по разрядности; *б* — по количеству слов; *в* — по разрядности и количеству слов

Нагрузка на согласующие схемы управления модуля памяти, в котором накопитель выполнен на биполярных БИС ЗУ, определяется входными токами логических «0» и «1» и входными емкостями. Если накопитель реализуется на БИС ЗУ с МОП-структурой, то входными токами, определяемыми токами утечки, обычно пренебрегают.

Количество БИС ЗУ, требуемое для построения накопителя модуля ЗУ, определяется вне зависимости от способа его построения как

$$Q_{\text{нк}} = (N/N_m)(n/n_m), \qquad (3.1)$$

где $Q_{\text{нк}}$ — количество БИС ЗУ, требуемое для организации накопителя модуля; $N(N_m)$ — количество слов (адресов) в накопителе модуля ЗУ (БИС ЗУ); $n(n_m)$ — число разрядов в накопителе (БИС ЗУ).

Объединение БИС ЗУ по входу. Эта операция производится с учетом коэффициентов объединения по любой цепи накопителя (адресной, информационной и управления), под которыми понимают число одноименных цепей накопителя модуля ЗУ, которые следует подключить к одному выходу соответствующей согласующей схемы. Тогда коэффициент объединения по адресным цепям ($K_{\text{а}}$) и цепям записи-считывания ($K_{\text{зп-сч}}$) равен числу БИС ЗУ в накопителе модуля, т. е.

$$K_{\text{а}} = K_{\text{зп-сч}} = Q_{\text{нк}} = (N/N_m)/n\,(n_m).$$

Коэффицент объединения по информационным входным цепям ($K_и$) и число выходов S дешифратора выбора кристаллов (ДшВК) определяются как

$$K_и = S = N/N_m. \tag{3.2}$$

Коэффициент объединения по цепям выбора кристалла ($K_{в.к}$) равен

$$K_{в.к} = n/n_m. \tag{3.3}$$

Дешифратор выбора кристаллов обычно управляется адресами и синхронным сигналом при необходимости.

Нагрузка по одной входной цепи накопителя, например адресной, определяется как

$$I_а^0 = K_а I_{вх.а}^0; \quad I_а^1 = K_а I_{вх.а}^1; \quad C_а = K_а C_{вх.а} + C_{ма}, \tag{3.4}$$

где $I_а^0$, $I_а^1$ — токи нагрузки лог. 0 и лог. 1 соответствующего согласующего элемента; $I_{вх.а}^0$, $I_{вх.а}^1$ — входные токи лог. 0 и лог. 1 для одного адресного входа БИС ЗУ; $C_а$ ($C_{вх.а}$) — входная емкость одной входной адресной цепи накопителя модуля (одного адресного входа БИС ЗУ); $C_{ма}$ — монтажная емкость по одной адресной цепи накопителя модуля (в расчетах емкость монтажа по любой цепи БИС обычно принимается $C = 15$ пф) [24].

Нагрузка по другим входным цепям накопителя (ВК, записи-считывания и информационным входам БИС ЗУ) определяется по формуле (3.4) с учетом своих коэффициентов объединения, а также значений входных токов и емкостей.

Если нагрузка по любому входу накопителя больше допустимой на согласующую схему этой цепи, то требуется разделить одноименные цепи накопителя на группы и поставить дополнительные согласующие схемы. Число этих схем определяется по каждой цепи отдельно. Например, для адресной цепи

$$m_{аC} = \left\rceil \frac{C_а}{C_{н.согл} - C_{ма}} \right\lceil; \tag{3.5}$$

$$m_а I_{вых}^0 = \left\rceil \frac{I_а^0}{I_{вых.согл}^0} \right\lceil; \quad m_а I_{вых}^1 = \left\rceil \frac{I_а^1}{I_{вых.согл}^1} \right\lceil, \tag{3.6}$$

где $C_{н.согл}$ — допустимая емкость нагрузки согласующей схемы; $I_{вых.согл}^0$ ($I_{вых.согл}^1$) — допустимый выходной ток лог. 0 (лог. 1) согласующей схемы.

При этом число групп $m_а$ выбирается равным максимальному значению, определенному из выражений (3.5), (3.6).

В каждой группе накопителя число БИС $Q_а'$, объединенных по адресному входу, определяется по формуле $Q_а' = K_а/m_а$. Для ос-

тальных входных цепей накопителя (информационной, ВК, Зп-Сч) значение m определяется аналогично выражениям (3.5), (3.6). Выбранные согласующие элементы должны обеспечивать заданные уровни входных сигналов БИС ЗУ.

Если в параметрах БИС ЗУ указываются требования по фронтам входных сигналов, то число групп согласующих схем определяется кроме расчета по выражениям (3.5), (3.6) и по условию

$$m'_{aC} = I^1_{\text{вых.согл}} t_\phi / (C_a U^1), \qquad (3.7)$$

где t_ϕ — длительность фронта входного импульса.

Окончательное число согласующих схем по входным цепям накопителя модуля ЗУ ($m_{нк}$) можно определить как

$$m_{нк} = m_a \log_2 N + m_{зп\text{-}сч} + m_и n_m + m_{вк} S,$$

где $\log_2 N$ — число адресных входов БИС ЗУ накопителя.

Выходные нагрузочные характеристики модуля ЗУ определяются по схемотехнической реализации выходов БИС ОЗУ: в биполярных структурах — это выходные каскады ТТЛ-типа, трехстабильный выход или выход с открытым коллектором (ОК); в структурах на МОП-транзисторах — выход с тремя состояниями.

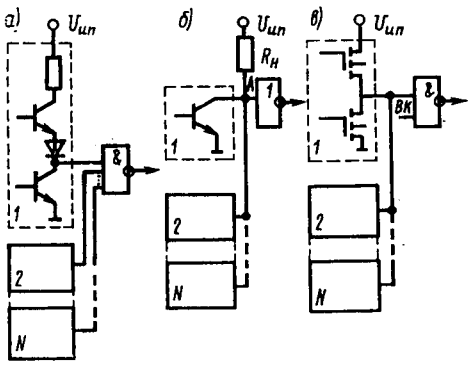


Рис. 3.11. Схема объединения выходов БИС ЗУ:
а — ТТЛ-выход; *б* — выход с ОК; *в* — трехстабильный выход

Объединение БИС ЗУ по выходу. Для увеличения информационной емкости выходы БИС ЗУ объединяют по проводному ИЛИ или по логическому ИЛИ.

Объединение выходных каскадов БИС ЗУ по проводному ИЛИ производится в том случае, если БИС ЗУ выполнена с открытым коллектором (ОК) или с тремя состояниями (рис. 3.11, *б*, *в*), и по логическому ИЛИ, если БИС ЗУ имеет выход ТТЛ-типа (рис. 3.11, *а*).

В любой момент времени можно выбрать только одну из объединенных микросхем. При объединении БИС ЗУ по логическому ИЛИ расчет выходных нагрузочных характеристик производится аналогично расчету схем на логических ТТЛ-схемах.

При объединении БИС ЗУ по проводному ИЛИ (рис. 3.11, *б*) нагрузка определяется числом микросхем, объединенных в одну точку. Коэффициент объединения по выходу P определяется как

$$P = N / N_m. \qquad (3.8)$$

Емкость нагрузки на выходе БИС ЗУ с ОК в точке A определяется как

$$C_{н}=(P-1)C_{вых}+rC_{вх}+C_{м}, \qquad (3.9)$$

где $C_{вых}$ — выходная емкость одной БИС ЗУ; $C_{вх}$ — емкость входной цепи нагрузки; r — число входных цепей нагрузки; $C_{м}$ — монтажная емкость ($C_{м}=15$ пФ).

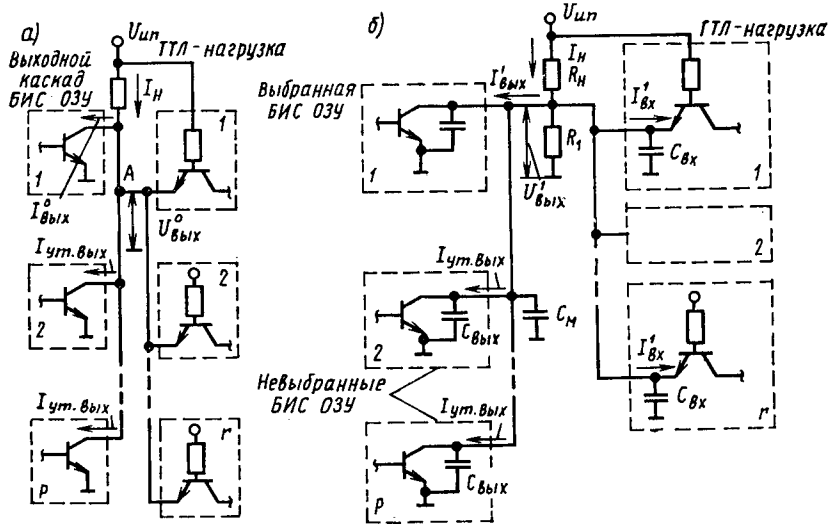


Рис. 3.12. Выходные цепи БИС ЗУ с ОК:
а — режим лог. 0; *б* — режим лог. 1

При построении модуля на БИС ЗУ должны выполняться условия

$$C_{н}\leqslant C_{н.доп};\quad I^{1}_{вых}\leqslant I^{1}_{вых.доп};\quad I^{0}_{вых}\leqslant I^{0}_{вых.доп}. \qquad (3.10)$$

Если условия (3.10) не выполняются, то БИС ЗУ, объединяемые по выходу, делят на группы исходя из расчета:

$$Z_{C}=\left\rceil\frac{C_{н}}{C_{н.доп}}\right\lceil;\quad Z_{I^{1}_{вых}}=\left\rceil\frac{I^{1}_{вых}}{I^{1}_{вых.доп}}\right\lceil;\quad Z_{I^{0}_{вых}}=\left\rceil\frac{I^{0}_{вых}}{I^{0}_{вых.доп}}\right\lceil. \qquad (3.11)$$

При этом выбранное Z должно быть равно максимальному значению, определяемому из условий (3.11). Отдельные группы объединяются через логическое ИЛИ.

В схемах БИС ОЗУ с ОК (рис. 3.12, *а*) сопротивление нагрузки $R_{н}$ зависит от минимального и максимального значений уровней напряжения согласования на выходе БИС, определяясь как

$$R_{\text{н max}} = \frac{U_{\text{и.п min}} - U^1_{\text{min}}}{I^1_{\text{вых}} + rI^1_{\text{вх}} + (P-1)\,I_{\text{ут.вых}}}; \tag{3.12}$$

$$R_{\text{н min}} = \frac{U_{\text{и.п max}} - U^0_{\text{max}}}{I^0_{\text{вых}} + rI^0_{\text{вх}} + (P-1)\,I_{\text{ут.вых}}}, \tag{3.13}$$

где $I_{\text{ут.вых}}$ — выходной ток невыбранной схемы; $U_{\text{и.п max}}$, $U_{\text{и.п min}}$ — напряжение питания БИС ОЗУ (максимальное и минимальное).

Так как член $(P-1)I_{\text{ут. вых}}$ мал по сравнению с другими, то им при расчете обычно пренебрегают.

Величина резистора $R_{\text{н}}$ должна лежать в пределах

$$R_{\text{н min}} \leqslant R_{\text{н}} \leqslant R_{\text{н max}}. \tag{3.14}$$

Если из расчета получено, что $R_{\text{н}} < 1$ кОм, то для обеспечения нормального режима работы ТТЛ—ИС, необходимо включить дополнительный резистор R_1 (рис. 3.12, *б*), сопротивление которого определяется следующим образом.

Исходя из уровней лог. 1 для ТТЛ-схем, имеем

$$U^1_{\text{min}} < U_{\text{A}} < U^1_{\text{max}}, \tag{3.15}$$

где U_{A} — напряжение между землей и точкой А.

Считая, что $R^1_{\text{экв}}$ — суммарная величина параллельно включенных резисторов R_1, $R^1_{\text{вх}}/r$ (входного сопротивления ТТЛ-схем в режиме лог. 1) и $R_{\text{ут}}/(P-1)$ (выходного сопротивления невыбранных БИС ЗУ), имеем

$$\frac{1}{R^1_{\text{экв}}} = \frac{1}{R_1} + \frac{1}{R_{\text{вых}}} + \frac{r}{R^1_{\text{вх}}} + \frac{P-1}{R_{\text{ут}}}. \tag{3.16}$$

Отсюда

$$\frac{U^1_{\text{min}}}{U_{\text{и.п min}} - U^1_{\text{min}}} R_{\text{н}} \leqslant R^1_{\text{экв}} \leqslant \frac{U^1_{\text{max}}}{U_{\text{и.п max}} - U^1_{\text{max}}} R_{\text{н}}.$$

Для большинства модулей ЗУ на БИС с ОК считают, что

$$R^1_{\text{экв}} \approx R_1.$$

Выбор значений $R_{\text{н}}$ и R_1 оказывает влияние на быстродействие модуля ЗУ, поэтому к информационным входным сигналам предъявляется следующее требование по длительности $t_{\text{и}}$ их существования:

$$t_{\text{и}} = 2{,}3R_{\text{экв}}C_{\text{н}} \leqslant t_{\text{и.доп}}, \tag{3.17}$$

где

$$R_{\text{экв}} = R^1_{\text{экв}}R_{\text{н}}/(R_{\text{экв}} + R_{\text{н}}).$$

Если условие (3.17) не выполняется, значение $R_{\text{н}}$ из (3.14) выбрано минимальным, а R_1 из (3.16) — максимальным, то не-

обходимо уменьшить число БИС ЗУ, объединяемых по выходу. В этом случае, считая, что $C_{н} \approx PC_{вых}$, получим

$$P \leqslant t_{и.доп}/(2{,}3R_{экв}C_{вых}). \qquad (3.18)$$

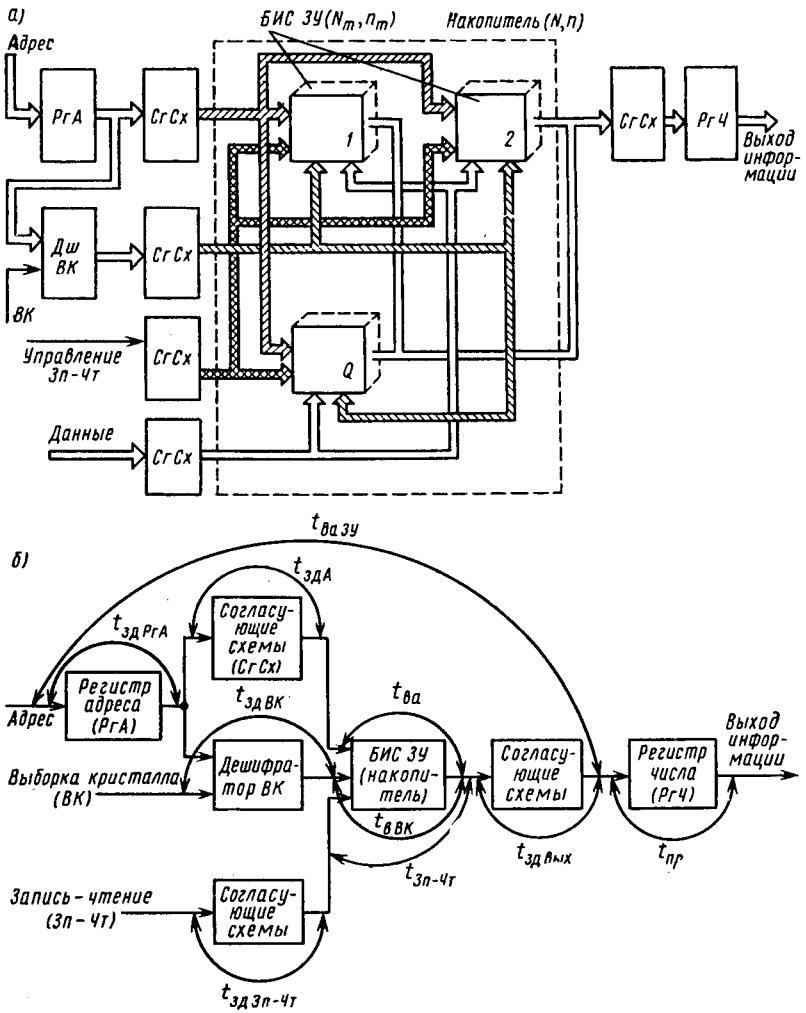


Рис. 3.13. Модуль памяти на БИС ОЗУ:
а — структурная схема модуля; *б* — определение времени выборки

При выборе из (3.14) значения $R_{н} \geqslant 1$ кОм необходимо проверить выполнение условия (3.17), считая $R_{экв} \approx R_{н}$. Если оно не выполняется, то необходимо уменьшить число БИС ЗУ, объединяемых по выходу согласно (3.18).

После синтеза структуры модуля ЗУ определяют его временны́е характеристики с учетом основных режимов функционирования БИС ЗУ (считывания, записи и хранения информации), а также величины задержек в схемах управления и согласования.

Расчет временны́х характеристик. Расчет временны́х характеристик модуля памяти проведем применительно к БИС ЗУ и временны́м диаграммам, приведенным на рис. 3.13, в П II.1 [24].

Для режима считывания обычно определяют время выборки адреса ЗУ ($t_{\text{в.а ОЗУ}}$) и время цикла считывания ($t_{\text{ц.сч ЗУ}}$). С учетом типа конкретных БИС ЗУ время выборки адреса определяется следующим образом:

а) если сигнал считывания подается одновременно с адресом постоянным уровнем, а сигнал ВК — импульсный, то

$$t_{\text{ва}} = t_{\text{зд Рг.А}} + t_{\text{зд ВК}} + t_{\text{с ВКа}} + t_{\text{с ВК}} + t_{\text{зд.вых}} + t_{\text{зд.а}}; \qquad (3.19)$$

б) если сигналы считывания и ВК — импульсные, то для расчета принимаются максимальные значения задержек соответствующих сигналов из выражения (3.19):

$$t_{\text{ва}} = t_{\text{зд max Рг.А}} + t_{\text{зд max ВК}} + t_{\text{с max ВКа}} + t_{\text{зд max вых}} + t_{\text{с max ВК}} + t_{\text{зд max а}}; \qquad (3.20)$$

в) если сигналы считывания и ВК подаются постоянным уровнем, то

$$t_{\text{ва ОЗУ}} = t_{\text{зд Рг.А}} + t_{\text{зд.а}} + t_{\text{ва}} + t_{\text{зд.вых}}. \qquad (3.21)$$

Время цикла считывания определяется следующим образом:

а) при импульсном характере сигнала ВК

$$t_{\text{ц.сч}} = \tau_{\text{ВК.д.сч}} + t_{\text{сх.а ВК}} + t_{\text{зд.а}} + t_{\text{с ВКа}}, \qquad (3.22)$$

где

$$\tau_{\text{ВК д.сч}} = \tau_{\text{ВК}} + t_{\text{зд ВК}} + t_{\text{зд.вых}} + t_{\text{пр}}; \qquad (3.23)$$

б) при подаче сигналов считывания и ВК постоянным уровнем

$$t_{\text{ц.сч ОЗУ}} = t_{\text{ва ОЗУ}}, \text{ причем } t_{\text{ва ОЗУ}} > t_{\text{сх.и.а}} > t_{\text{пр}}. \qquad (3.24)$$

Кроме этих временны́х параметров требуется также определять моменты подачи и длительность управляющих сигналов ВК, Зп-Сч и других с учетом задержек согласующих схем модуля ОЗУ. Так, время сдвига сигнала ВК на входе модуля ЗУ определяется как

$$t_{\text{с ВК ОЗУ}} = t_{\text{зд ВК}} + t_{\text{с ВКа}}. \qquad (3.25)$$

Для режима записи информации в ЗУ, характеризующегося наличием на входах БИС ЗУ сигналов записи (Зп) и ВК, время цикла записи ($t_{\text{ц.зп}}$) определяется следующим образом:

а) при постоянном уровне сигнала ВК

$$t_{\text{ц.зп ОЗУ}} = t_{\text{с.зп.а}} + \tau_{\text{зп}} + t_{\text{сх.а.зп}} + t_{\text{зд.зп}} + t_{\text{зд.а}}; \qquad (3.26)$$

б) при импульсном сигнале ВК и постоянном уровне сигнала Зп

$$t_{\text{ц.зп ОЗУ}} = t_{\text{с ВКа}} + \tau_{\text{ВК д.зп}} + t_{\text{сх.а ВК}} + t_{\text{зд.а}}, \qquad (3.27)$$

где $\tau_{\text{ВК д.зп}} = \tau_{\text{ВК}} + \tau_{\text{зд ВК}}$.

Так как $\tau_{\text{ВК д.сч}} > \tau_{\text{ВК д.зп}}$, то обычно выбирают $\tau_{\text{ВК д}} = \tau_{\text{ВК д.сч}}$;

в) при импульсном характере сигналов ВК и Зп время цикла определяется максимальными задержками сигналов.

Энергопотребление модуля ЗУ складывается из энергопотребления схем управления и накопителя:

$$P_{\text{пот ЗУ}} = P_{\text{упр}} + P_{\text{нк}} = P_{\text{Рг.А}} + P_{\text{Рг.Ч}} + P_{\text{дш}} + P_{\text{согл}} + QP_{\text{пот}}. \qquad (3.28)$$

В БИС ЗУ, выполненных, например, по КМОП-технологии, энергопотребление в режимах хранения, записи или считывания разное, поэтому

$$P_{\text{нк}} = P_{\text{хр}}\left(Q - \frac{n}{n_m}\right) + P_{\text{пот}}\frac{n}{n_m}\frac{\tau_{\text{ВК}}}{t_{\text{ц}}} + P_{\text{хр}}\frac{n}{n_m}\frac{t_{\text{ц}} - \tau_{\text{ВК}}}{t_{\text{ц}}}, \qquad (3.29)$$

где $\tau_{\text{ВК}}/t_{\text{ц}}$ — максимальное значение из двух отношений

$$\tau_{\text{ВК}}/t_{\text{ц.сч}} \text{ или } \tau_{\text{ВК}}/t_{\text{ц.зп}}.$$

Алгоритм синтеза модуля памяти на БИС ЗУ. В целом модули памяти МПСУ на основе БИС ЗУ (рис. 3.13) синтезируются с учетом следующего алгоритма:

1. Исходя из технических требований, предъявляемых к синтезируемому модулю (по емкости, быстродействию, энергопотреблению, надежностным параметрам), выбирают конкретный тип БИС ЗУ из множества возможных или допустимых к применению.

2. По выражениям (3.1)—(3.3) определяют требуемое число БИС ЗУ, коэффициенты объединения по выходным цепям ($K_{\text{а}}$, $K_{\text{зп-сч}}$, $K_{\text{вк}}$, $K_{\text{и}}$) и число выходов S дешифратора ВК.

3. По выражению (3.4) с учетом соответствующих параметров определяют значения токовых и емкостных нагрузок по входным цепям модуля (адресным, ВК, записи-считывания, информационным).

4. По полученным в п. 3 результатам выбирают тип согласующих схем.

5. Согласно выражениям (3.5)—(3.7), БИС ЗУ разделяют на группы так, чтобы значения нагрузки каждой группы БИС не превышали допустимых выходных нагрузок соответствующей согласующей схемы.

6. По выражению (3.8) определяют число БИС, объединяемых по выходным цепям.

7. Определяют нагрузку по выходным цепям: для БИС, имеющих трехстабильный выход, — по выражению (3.9); для БИС, имеющих выход с ОК, — согласно выражениям (3.9), (3.12)—

(3.14); для биполярных БИС с объединением их по логическому ИЛИ — согласно параметрам микросхем.

8. Исходя из полученных в п. 7 результатов по выражению (3.11) определяют число БИС, объединяемых по выходам в группы.

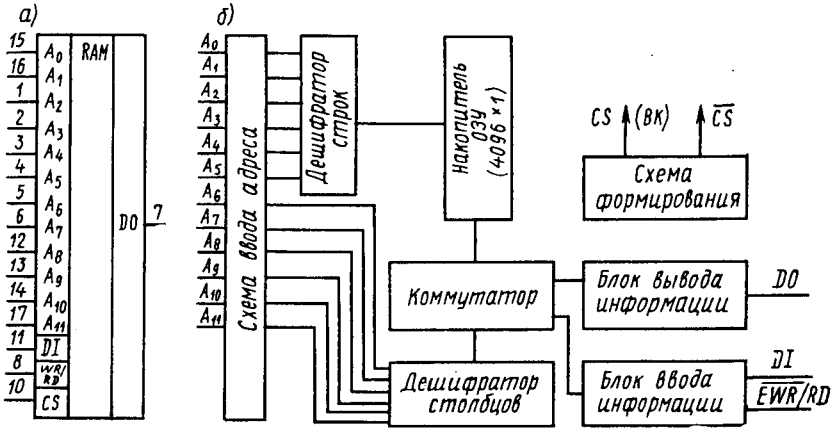


Рис. 3.14. БИС ОЗУ типа КР537РУ3А:
а — условное обозначение; *б* — функциональная схема

9. Определяют тип согласующих схем по выходным цепям для групп БИС ЗУ с их объединением через логическое ИЛИ.

10. По выражениям (3.17)—(3.27) определяют временные характеристики модуля ЗУ.

Таблица 3.13. **Особенности функционирования КР537РУ3**

Входы микросхем				Выход	Режим работы
$\overline{CS}$	*EWR/RD*	$A_0 \ldots A_{11}$	*DI*	*DO*	
1	×	×	×	R_{off}	Хранение
0	0	*A*	1	»	Запись 1
0	0	*A*	0	»	Запись 0
0	1	*A*	×	Данные в прямом коде	Считывание

Примечание. × — безразличное состояние, $A = \{0, 1\}$, R_{off} — выход в высокоимпедансном состоянии.

11. С учетом проведенных расчетов определяют окончательное число управляющих и согласующих схем.

12. По выражениям (3.28), (3.29) рассчитывают потребляемую мощность модуля ЗУ.

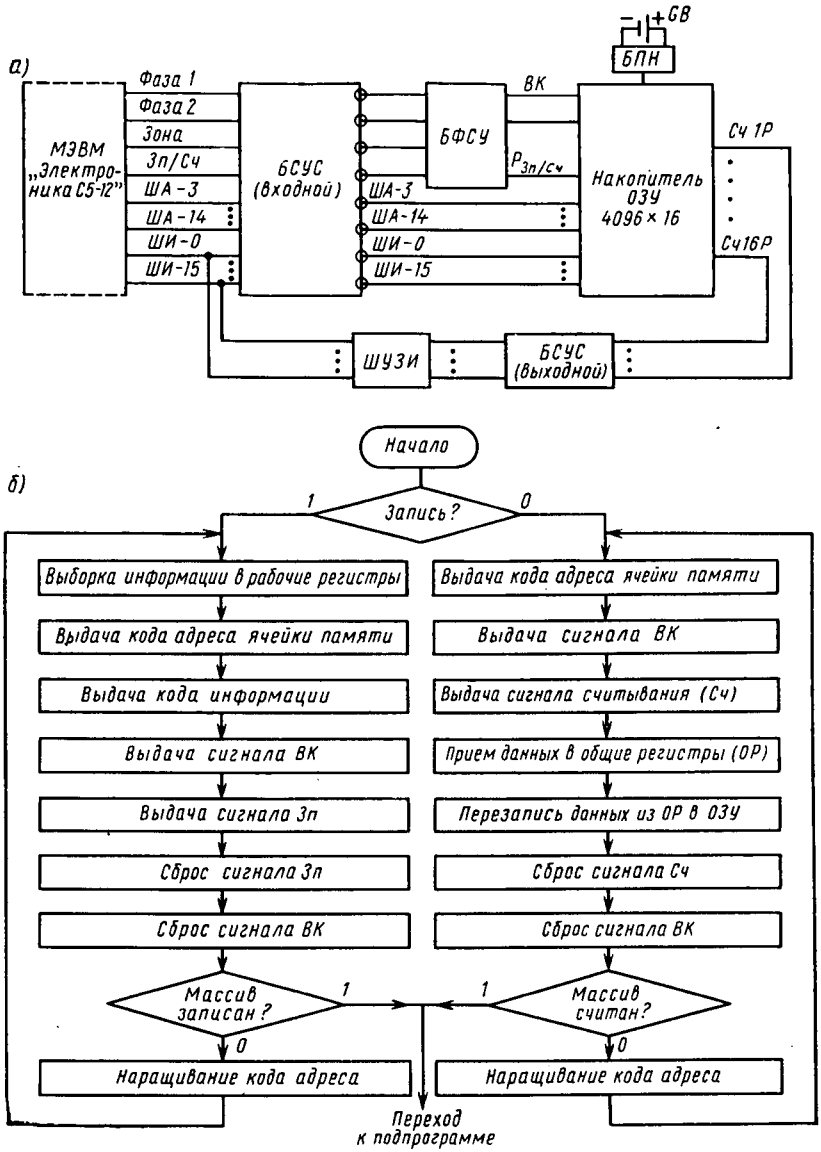


Рис. 3.15. Модуль энергонезависимой памяти на БИС ОЗУ КР537РУ3А:
а — структурная схема модуля; *б* — алгоритм работы ЭНОЗУ

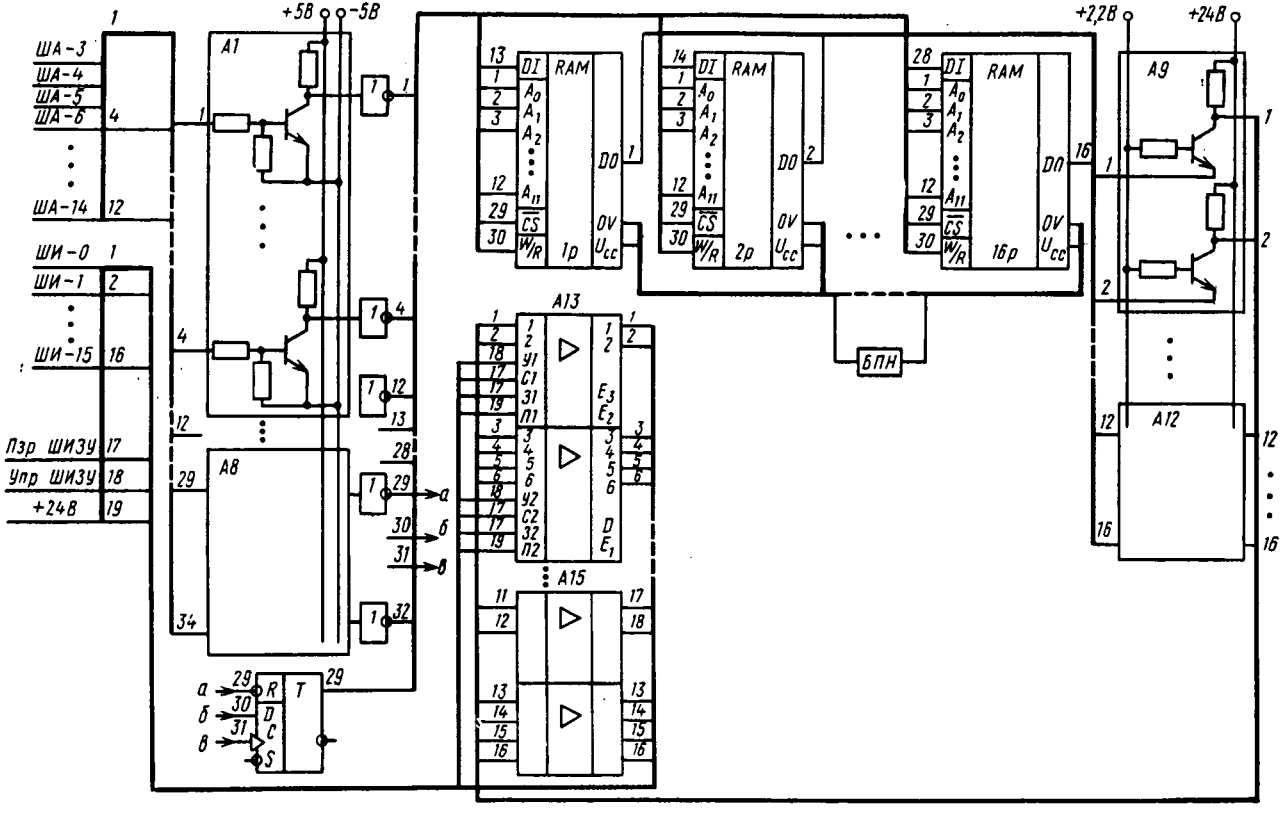


Рис. 3.16. Упрощенная схема модуля ЭНОЗУ на БИС ОЗУ КР537РУ3А

13. Если полученные характеристики модуля не соответствуют заданным, то выбирают другой тип БИС ЗУ или другие типы управляющих и согласующих микросхем.

Заметим, что применение одноразрядных БИС ЗУ позволяет легко обнаруживать и устранять ошибки с помощью дополнительных специализированных или универсальных средств (см. § 6.4). Любая неисправность в одноразрядной БИС ЗУ эквивалентна появлению независимых отказов лишь в одном разряде выходных слов, в то время как отказ многоразрядной БИС ЗУ может привести к появлению любого количества ошибок в группе разрядов выходных слов одновременно.

Пример 3.6. Для непосредственного сопряжения с микроЭВМ «Электроника С5-12» синтезировать энергонезависимый модуль ОЗУ 4К×16 на базе БИС ОЗУ типа К537РУ3, выполненных на КМОП-структурах с организацией 4К×1 (рис. 3.14) и трехстабильным выходом. Принцип действия БИС К537РУ3 поясняется табл. 3.13.

Структурно-функциональная схема энергонезависимого модуля ОЗУ (ЭНОЗУ) (рис. 3.15, *а*) содержит: входной и выходной блоки согласования уровней сигналов (БСУС) с микроЭВМ, блок формирования сигналов управления (БФСУ), шинный усилитель считывания с запоминанием информации (ШУЗИ); блок переключения напряжения (БПН). Алгоритм функционирования ЭНОЗУ (рис. 3.15, *б*) включает в себя последовательность операций по организации режимов записи и считывания информации. Опуская описание принципа действия и построения блоков БСУС, реализованного на микросхемах К125НТ1 и К155ЛН1, и ШУЗИ, реализованного на БИС К536УИ2, отметим следующее:

1. Расчет параметров модуля ЭНОЗУ на К537РУ3, выполненный по приведенному выше алгоритму, показал, что модуль можно реализовать на шестнадцати БИС ОЗУ К537РУ3, обеспечить заданную информационную емкость путем наращивания разрядности без деления на группы и использования дополнительных согласующих схем, так как их функции выполняют входной и выходной БСУС.

2. Связь ЭНОЗУ с микроЭВМ «Электроника С5-12» может быть осуществлена как непосредственно (I вариант), так и через модуль цифровых входов-выходов (II вариант).

В качестве примера на рис. 3.16 приведена упрощенная принципиальная электрическая схема ЭНОЗУ для первого варианта.

Контрольные вопросы

1. Какие основные модели применяются при проектировании СЛУ? **2.** Каковы особенности синтеза СБФ на мультиплексорах? Как получить экономичную реализацию СБФ на МХ? **3.** В чем заключается специфика синтеза автоматов на ПЗУ и ППЗУ? Какие особенности встречаются при синтезе автоматов, реализуемых на основе ППЗУ, и как обеспечить корректность их функционирования? **4.** В чем состоит сложность реализации автоматов на программируемых БИС? Какие типовые решения применяются при этом? **5.** В чем состоит особенность проектирования модулей памяти на основе БИС ЗУ?

ГЛАВА 4
ПРОЕКТИРОВАНИЕ ВСТРАИВАЕМЫХ СИСТЕМ УПРАВЛЕНИЯ НА БАЗЕ МПК БИС

В общем случае процесс проектирования систем управления включает в себя ряд этапов, на каждом из которых используется совокупность моделей, отражающих структуру, поведение, цель или ресурсы проектирования с различной степенью детализации. В зависимости от сложности реализуемых задач и применяемых для этого микропроцессорных средств можно выделить ряд уровней функциональных и технических структур МПСУ, имеющих взаимно-однозначное соответствие (рис. 4.1) [4]. При этом на каждом из уровней детализации структур принципиально необходимыми становятся процедуры выбора наилучших решений, удовлетворяющих заданным ограничениям и критериям качества.

§ 4.1. ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ СИСТЕМ УПРАВЛЕНИЯ НА БАЗЕ МИКРОПРОЦЕССОРНЫХ СРЕДСТВ

На основе микропроцессорных средств сегодня создаются разнообразные системы управления. По особенностям требований, предъявляемым к МПСУ со стороны пользователей, их, например, можно подразделить на:

а) системы управления со встроенными МП, жестко запрограммированными на выполнение определенных функций (МПСУ автомобилем, медицинские, электро- и радиоизмерительные приборы, бытовые аппараты и др.);

б) системы управления средней сложности, программируемые самим пользователем, с возможностью модификации исходных программ в процессе эксплуатации (МПСУ технологическими процессами, станочными комплексами, роботами и др.);

в) системы сбора и ускоренной обработки больших массивов информации с распараллеливанием алгоритмов и вычислительных процессов и реализацией их на основе многопроцессорных структур, позволяющих управлять сложными объектами в реальном масштабе времени (к таким системам относятся, в частности, новые многопроцессорные системы с программируемой архитектурой, универсальной коммутацией, распределенной памятью и многофункциональными МП [12], в систему команд которых входят сложные арифметические, функциональные, логические, дифференциальные, интегральные, матричные, тензорные и другие виды операторов);

г) специализированные управляющие вычислительные комплексы (типа ПС-2000, ПС-3000), позволяющие обрабатывать боль-

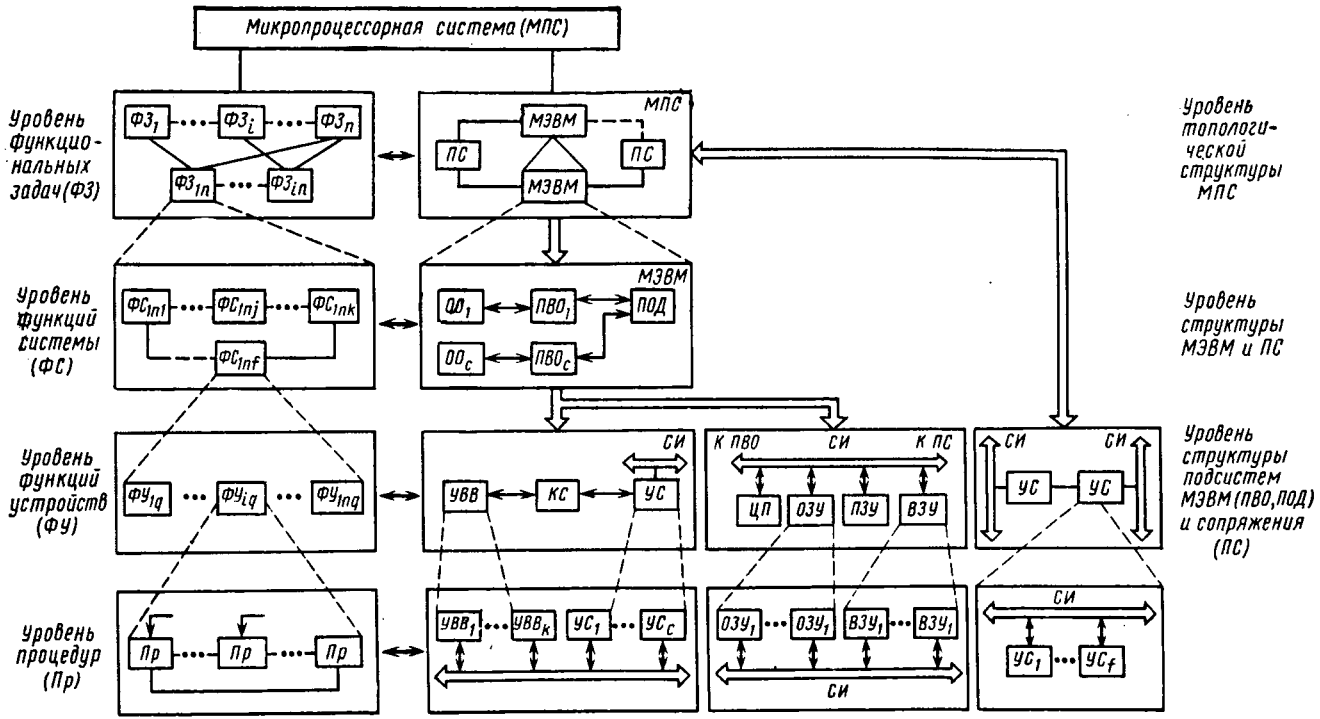


Рис. 4.1. Уровни иерархии функциональной и технической структур МПС

шие массивы информации с быстродействием 20—200 млн. операций в секунду, формировать модели сложных систем в реальном масштабе времени путем распараллеливания и одновременного выполнения сложных операций (быстрого преобразования Фурье, цифровой фильтрации сигналов, вычисления корреляционных функций, векторно-матричных преобразований и др.).

Последние два класса систем представляют собой отдельные перспективные направления в реализации сложных систем управления и обработки информации на базе новых микропроцессорных средств, поэтому далее они не обсуждаются.

Ниже рассмотрим особенности проектирования систем логического управления на основе МПК БИС. Заметим, что СЛУ на базе МПК БИС разрабатываются в основном для специализированных областей при массовом их использовании с получением выигрыша в эффективности, экономичности, быстродействии и других качественных показателях проектируемых систем по сравнению с серийными МЭВМ и в то же время их быстродействие приемлемо для объекта управления.

В общем случае процесс разработки системы управления на основе МПК БИС включает в себя:

а) ознакомление с техническими характеристиками и принятие решения об использовании того или иного типа МП, способа сопряжения МП с УСО, памятью и УВВ, выбора требуемого состава оборудования;

б) декомпозицию функций с целью реализации их на программном и(или) аппаратном уровне; наиболее важными критериями, определяющими границу аппаратной и программной реализации заданных функций, служат параметры быстродействия используемых МП и экономической эффективности реализуемых функций;

в) составление алгоритмов и согласование интерфейсов;

г) программирование и отладку целевых программ на кросс-ЭВМ параллельно с разработкой макета и их последующая стыковка;

д) комплексную проверку и отладку МПСУ с проверкой соответствия полученных решений поставленным требованиям;

е) разработку проекта системы управления по принятым в данной отрасли нормативным материалам.

Проектирование аппаратных средств состоит в разработке подробной логической схемы каждого модуля МПС (модулей управления, памяти, интерфейса ввода — вывода) с построением временных диаграмм важнейших управляющих сигналов; в выборе устройств, необходимых для реализации МПСУ, с учетом их стоимости, быстродействия и энергопотребления; в конструировании каждого модуля на макете (тестовой плате); в проверке схем путем прослеживания сигналов и сравнения их с временными диаграммами; в сборке модулей в законченную систему (прототип)

с проверкой аппаратных неисправностей в прототипе до и после загрузки команды.

Создание прикладных программных средств состоит в разбиении сложной программы на подпрограммы для упрощения ее отладки и модификации; в разработке блок-схем для каждого программного модуля; кодировании программных модулей на одном из языков программирования; в отладке каждого программного модуля с обнаружением логических ошибок, их исправлением и повторной проверкой программ; в объединении всех программных модулей в законченный программный комплекс и в его окончательной отладке. Обобщенный алгоритм процесса проектирования и разработки аппаратных и программных средств МПСУ с применением МПК БИС приведен на рис. 4.2.

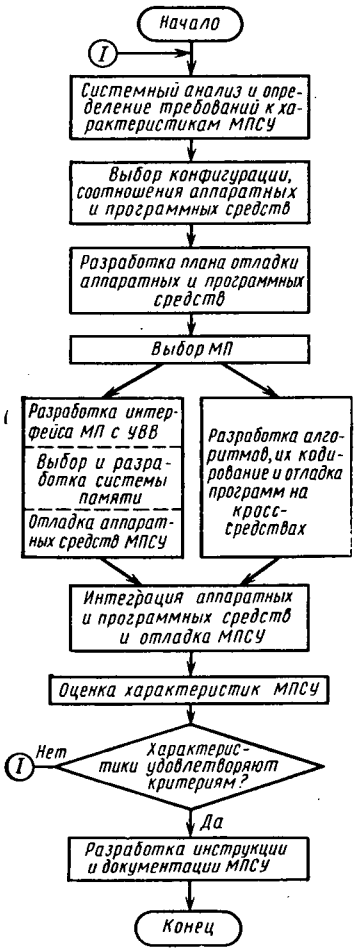


Рис. 4.2. Блок-схема алгоритма разработки МПСУ

Наиболее ответственным этапом при проектировании систем управления на МПК БИС является выбор типа микропроцессора, от которого в дальнейшем во многом зависит эффективность разрабатываемой системы.

§ 4.2. ОСОБЕННОСТИ ВЫБОРА МИКРОПРОЦЕССОРНЫХ СРЕДСТВ ДЛЯ ПРОЕКТИРОВАНИЯ ВСТРАИВАЕМЫХ СИСТЕМ УПРАВЛЕНИЯ

При проектировании МПСУ, выдвигается множество противоречивых критериев качества, технических требований и ограничений — стоимость, надежность, быстродействие, энергомощностные, весогабаритные ограничения и др. Поиск рационального решения, удовлетворяющего заданным критериям, требованиям и ограничениям, может производиться на основе: а) применения единого интегрального критерия Φ и решения задачи синтеза МПСУ с поиском экстремума этого критерия на основе свертки; б) использования методов выбора рациональной структуры МПСУ по многим критериям без свертки (например, метод зондирования пространства параметров).

Известны три основных метода получения свертки [30]:

а) метод нахождения «весов» $b_1, \ldots, b_k$, учитывающий предпочтительность или «важность» критериев $\Phi_1, \Phi_2, \ldots, \Phi_k$ с учетом условия нормирования весов $\sum_{i=1}^{k} b_i = 1$ и сводящий все критерии качества к одному усредненному критерию вида

$$\Phi(A) = b_1\Phi_1(A) + b_2\Phi_2(A) + \ldots + b_k\Phi_k(A); \qquad (4.1)$$

б) метод отождествления обобщенного критерия Φ с одним из критериев качества Φ_i, считающимся наиболее важным, и заменой всех остальных критериев ограничениями;

в) метод последовательных уступок, выбор оптимального варианта по которому зависит от порядка нумерации критериев и величины уступок по каждому из них.

Каждый из перечисленных методов свертки использует дополнительные данные (назначение коэффициентов важности, выбор главного критерия, порядок следования критериев и величины уступок по каждому из них), которые оценить трудно, а при неточном их выборе можно получить неэффективное решение.

На сегодня нет формального метода для выбора предпочтительного j-го варианта системы управления, синтезируемой на основе МПК БИС, позволяющего аналитически связать ее эффективность Q_j с частными критериями $\{K_i | i = \overline{1, m}\}$. Вид этой зависимости обычно неизвестен, и отыскать его трудно и не всегда возможно. Поэтому на практике предпочтительный вариант МПСУ отыскивают путем:

а) выделения одного критерия K_i^* из множества возможных $\{K_i | i = \overline{1, m}\}$ и перевода остальных частных критериев $\{K_i \setminus K_i^*\}$ в разряд ограничений. При этом для конкретного применения выбирают вариант МПСУ, величина K_i^* у которого в максимальной степени удовлетворяет техническим требованиям и ограничениям;

б) формирования по определенному решающему правилу обобщенного критерия эффективности из ограниченного числа частных критериев;

в) определения некоторой функции, тем или иным способом формально объединяющей частные критерии $\{K_i\}$ и позволяющей при сравнении ряда вариантов использовать условия неравенства критериев в виде $K_i^1 \geqslant K_i^2$. Если хотя бы для одного номера i неравенство выполняется строго, то этот вариант предпочтительнее других вариантов.

К известным методам, пригодным для оценки МПСУ с использованием частных критериев, на этапе выбора МПК БИС относятся методы, использующие аддитивные и мультипликативные кри-

терии Q_j вида (4.2). Они позволяют объединять не сами частные критерии, имеющие различные физическую природу и размерность, а их отношение к некоторой нормирующей величине: экстремаль-

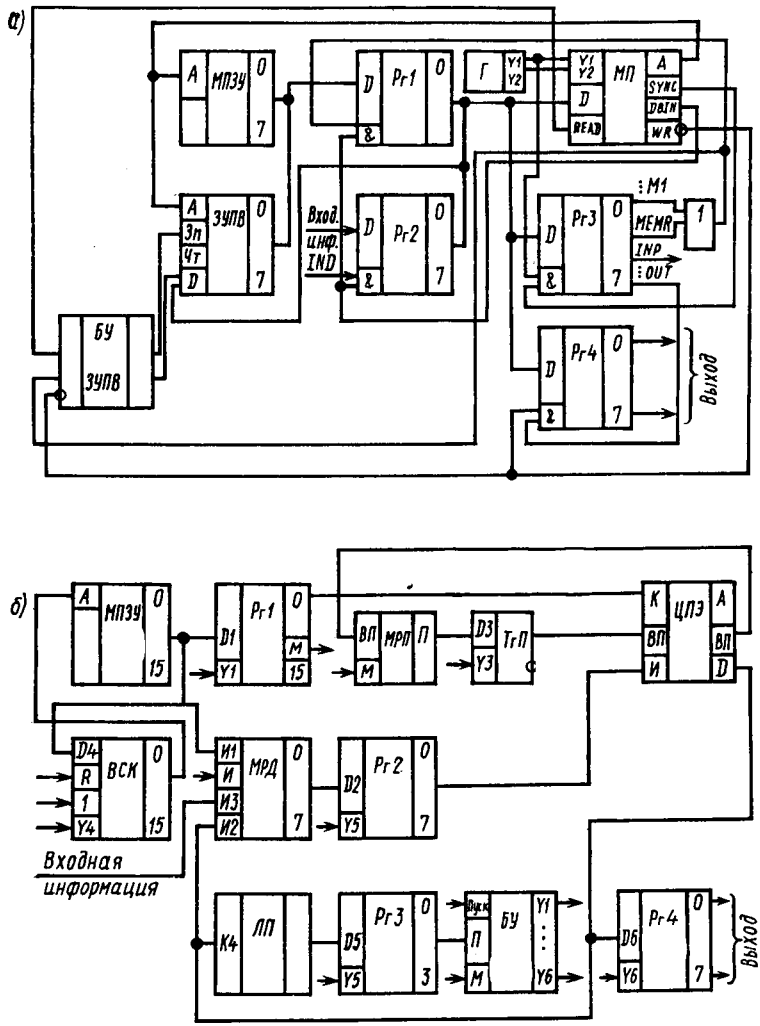


Рис. 4.3. Базовые конфигурации структуры МПСУ
а — на базе МП К580; *б* — на базе МП К582

ному значению частного критерия из множества рассматриваемых; разности между максимальным и минимальным значениями частного критерия; отклонению частных критериев от экстремальных значений и др.:

$$Q_j = \sum_{(i)} b_i \Phi_i(K) \text{ или } Q_j = \prod_i K_i^{b_i}, \qquad (4.2)$$

где b_i — весовой коэффициент значимости i-го частного критерия K ($i = \overline{1, n}$; $j = \overline{1, m}$).

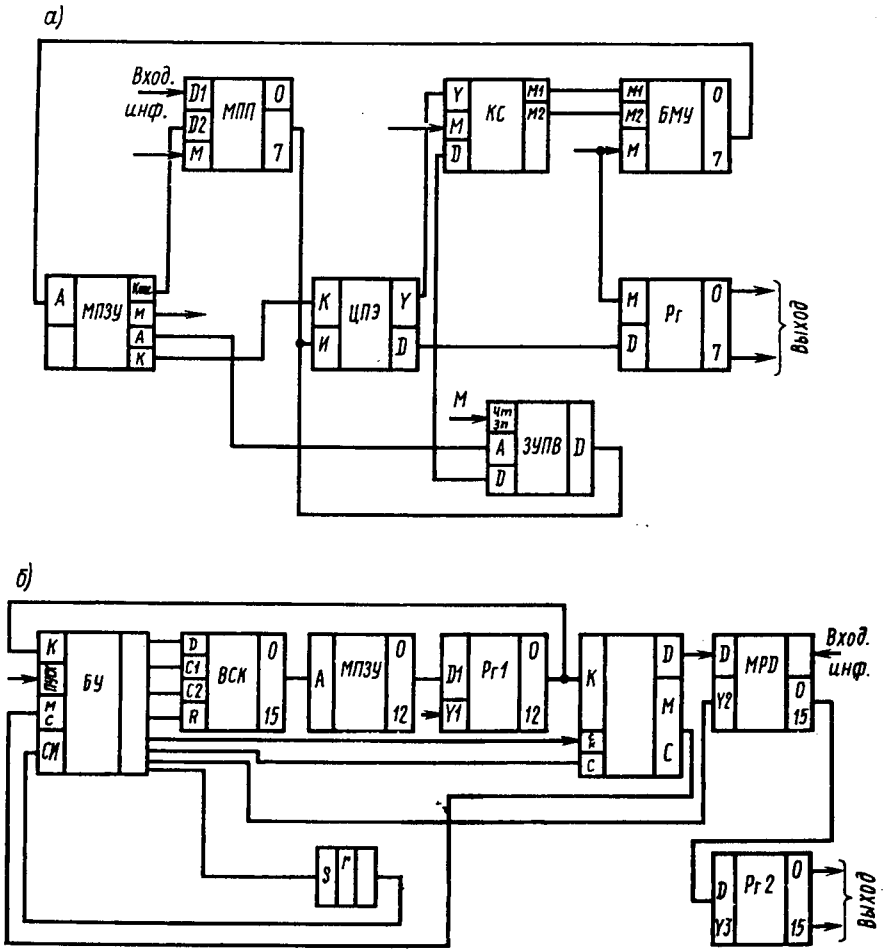


Рис. 4.4. Базовые конфигурации структуры МПСУ:
а — на базе МП К584; *б* — на базе МП К588

Однако определение весовых коэффициентов b_i в этих методах сопряжено с большими трудностями и обычно сводится к методам экспертной оценки. В целом данные методы позволяют сравнивать только сами МПК, включающие в себя базовые БИС, без

учета дополнительных СИС и МИС, обеспечивающих работоспособность МПСУ, что может привести к выбору неэффективного решения и большим программно-аппаратным, энергомощностным или временны́м затратам. Целесообразнее оценивать и сравнивать

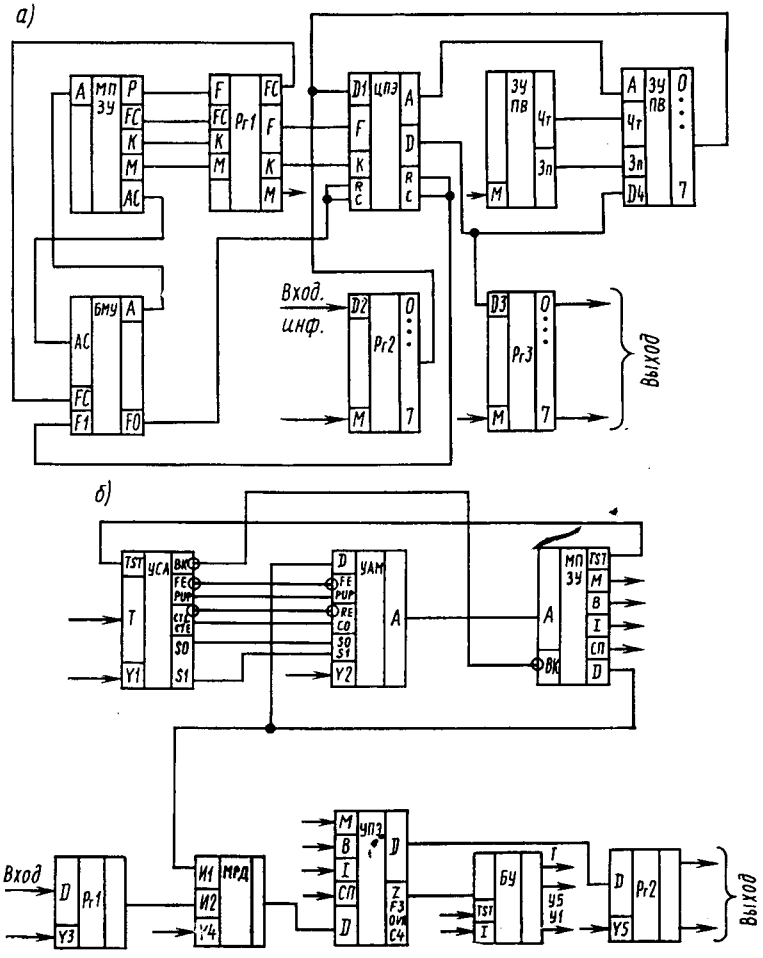


Рис. 4.5. Базовые конфигурации структуры МПСУ:
а — на базе МП К589; *б* — на базе МП КР1804

не сами МПК БИС, а базовые конфигурации структур (БКС) МПСУ, реализуемых на их основе. БКС включают в себя как типовые схемы процессоров и ЗУ, приводимые обычно в технической документации и реализующие программно (микропрограммно) заданные алгоритмы, так и дополнительные, соответствующие конкретному применению типовые схемы ввода — вывода. На рис.

4.3—4.7 приведены БКС МПСУ для восьми типов серийных МПК БИС с обеспечением для них равных возможностей в разрядности данных, представлении информации на входе и выходе, функциональной полноте систем команд (микрокоманд).

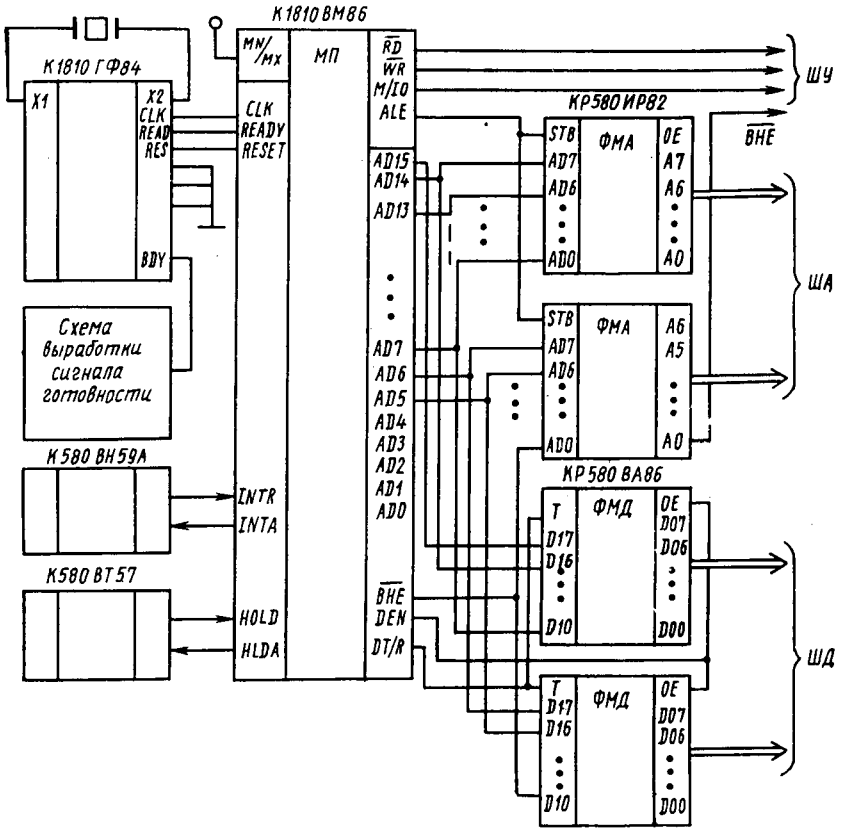


Рис. 4.6. Базовая конфигурация структуры МПСУ на базе МП К1810

Выбор и оценка БКС МПСУ на основе существующих разновидностей МПК БИС производятся следующим путем. На первом этапе исходя из цели и назначения проектируемой МПСУ, вида и характера реализуемого алгоритма, сформулированных на основе требований технического задания или условий эксплуатации, частных критериев эффективности, числовых текущих K_{ji} и эталонных значений $R_{эi}$, технических ограничений K_{jo}, из множества возможных для применения БКС $\{M_i | i=\overline{1,n}\}$ выделяется подмножество $M' \subseteq M$ предпочтительных вариантов БКС МПСУ. На втором этапе из выделенного подмножества БКС определяет-

ся МПК БИС, наиболее предпочтительный по условиям конкретного применения.

Если в качестве нормирующей величины при этом использовать эталонные значения $K_{эi}$, то векторы K_{ji} и K_{jo} преобразуются к виду

$$P_{j0}=\langle P_{j1},\ldots, P_{jn}\rangle;\quad P_0=\langle P_{10},\ldots, P_{n0}\rangle, \tag{4.3}$$

где $P_{ji}=K_{ji}/K_{эi}$; $P_{oi}=K_{oi}/K_{эi}$, если увеличение частного критерия K_{ji} (K_{oi}) улучшает качественные показатели, или $P_{ji}=K_{эi}/K_{ji}$;

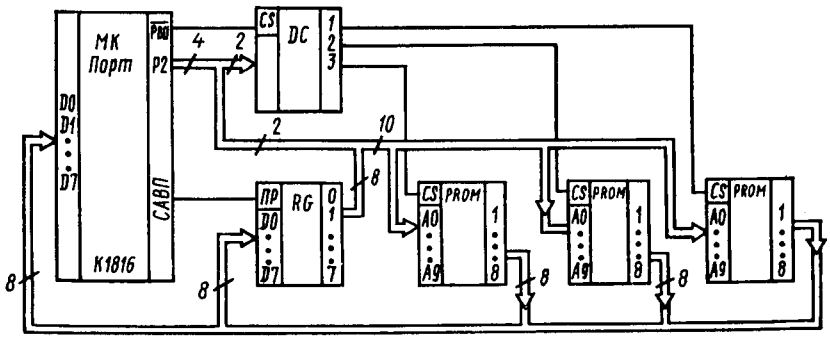


Рис. 4.7. Базовая конфигурация структуры МПСУ на базе МП К1816

$P_{oi}=K_{эi}/K_{oi}$ в противном случае.

Среди множества векторов отыскиваются векторы и, следовательно, БКС МПСУ, наиболее близкие к эталонной конфигурации с $P_э=1$. Мерой близости при этом может служить одна из норм вида

$$Q=\|P_э-P_j\|_r=\sqrt[r]{\sum_{i=1}^{n}(1-P_{ji})^r}, \tag{4.4}$$

где $r=\overline{1, N}$.

Окончательный выбор наиболее целесообразной БКС МПСУ при равенстве мер близости Q можно вести по критерию условной цены C (4.5), характеризующей общие затраты на достижение требуемого соответствия между текущими и эталонными значениями критериев эффективности:

$$C=\sum_{k=1}^{n} C_k Z_{l_k},\quad l=\overline{1, n}, \tag{4.5}$$

где Z_{l_k} — интегральный показатель качества МПСУ, реализованной на МПК БИС, C_k — коэффициент подсчета цены единицы соответствующего k-го параметра (стоимости, энергопотребления, объема программы и др.).

В целом степень объективности выбора БКС и оценки их параметров зависит от обоснованности выбора состава частных критериев, их текущих K_{ji} и эталонных $K_{эi}$ значений для каждого из вариантов.

В процессе проектирования МПСУ на базе МПК БИС стремятся обеспечить максимально эффективную реализацию заданных алгоритмов, что можно оценить с помощью таких параметров, как время реализации T алгоритма или его «критических» участков T_i, число управляющих слов N, объем постоянной памяти V, связность алгоритма и др. Если алгоритмы представляются в виде формализованного описания на языках логических, графических схем-алгоритмов (ЛСА) или (ГСА) с выделением операторных A_i и логических p_i вершин [18], то время выполнения реализуемого алгоритма T оценивают величиной

$$T=\sum_{l=1}^{n} w_l(z)\,t_l(z), \tag{4.6}$$

где $w_l(z)$, $t_l(z)$ — соответственно вероятность и время выполнения каждой z-цепочки ЛСА или ветви ГСА с конфигурацией, зависящей от общего числа ветвей в структуре алгоритма.

Число управляющих слов (УС) N, требуемое для реализации заданного алгоритма, оценивается величиной

$$N=\sum_{i=1}^{r} b_i A_i+\sum_{j=1}^{s} a_j p_j, \tag{4.7}$$

где b_i, a_j — количество элементарных управляющих слов, требуемое для выполнения каждого оператора A_i или логического условия p_j, входящих в ГСА ($i=\overline{1,r}$; $j=\overline{1,s}$).

Объем ПЗУ V для конкретной БКС МПСУ определяется величиной

$$V=dN, \tag{4.8}$$

где d — ориентировочная разрядность управляющего слова МПСУ.

Связность алгоритма характеризуется максимальным числом промежуточных результатов, подлежащих одновременному хранению в ОЗУ за время реализации конкретного алгоритма. По ее значению можно ориентировочно определять требуемое число РОН или примерный объем ОЗУ.

Пример 4.1. Выбрать наилучшую структуру из шести вариантов БКС МПСУ (рис. 4.3—4.5) применительно к областям I—III, заданным множеством эталонных значений $K_{эI}$, $K_{эII}$, $K_{эIII}$ соответственно, сведенным в табл. 4.1.

Выбор наилучшего варианта БКС МПСУ проводится по вышеописанной методике по двенадцати параметрам. Результаты расчета оценок Q_1 и Q_2 и относительных показателей числовых параметров, вычисленных по (4.3), (4.4), сведены в табл. 4.1.

Таблица 4.1. Расчетные данные по выбору предпочтительных структур МПС на базе МПК БИС

Серия МПК БИС	Область применения и параметры	Числовые параметры МПК и их относительные показатели: число источников питания	энергопотребление, Вт	наличие регистра состояния	число внутренних регистров	температура эксплуатации, °С (max)	площадь печатной платы, мм²	аппаратные затраты БКС	длина умножения (число чисел)	длина деления (число чисел)	время на умножение, мкс	время на деление, мкс	разрядность управляющ. слова, бит	Q_1 (см. (4.4))	Q_2 (см. (4.4))	место предпочтительности в ряду
Эталонные значения параметров для областей применения																
Эталонный МПК	$K_{эI}$ I	1	10	1	12	80	20000	50	10	14	100	100	16			
	$K_{эII}$ II	1	2	1	16	60	15000	100	2	2	30	30	8			
	$K_{эIII}$ III	3	5	1	7	125	10000	14	50	50	500	500	8			
К580	K_{it}	3	2,8	1	7	85	13400	14	41	33	278	346	8			
	I	2	0,72	0	0,42	0,06	0,33	0,44	3,1	1,35	1,78	2,46	0,5	13,16	5,07	6
	II	2	0,4	0	0,56	0,41	0,1	0,72	19,5	15,5	8,2	10,5	0	57,8	28,3	6
	III	0	0,44	0	0	0,32	0,34	0	0,18	0,34	0,44	0,3	0	2,36	0,91	1
К582	K_{jt}	1	8	0	8	80	35100	58	10	14	224	228	16			
	I	0	0,2	1	0,33	0	0,75	1,24	0	0	1,2	1,28	0	6,02	2,64	3
	II	0	3	1	0,5	0,33	1,34	0,12	4	6	6,4	6,6	1	30,3	14,26	5
	III	0,66	0,6	1	0,14	0,36	2,5	3	0,8	0,72	0,55	0,54	1	11,87	4,46	4
К584	K_{jt}	1	2,9	1	8	70	20900	21	2	3	25,5	27	60			
	I	0	0,71	0	0,33	0,12	0,04	0,18	0,80	0,79	0,74	0,73	2,7	7,13	2,66	4
	II	0	0,45	0	0,5	0,16	0,09	0,59	0	0,5	0,15	0,1	6,5	9,3	6,59	1
	III	0,66	0,42	0	0,14	0,44	1,0	0,46	0,96	0,94	0,95	0,94	6,5	13,41	6,91	6

Продолжение табл. 4.1

Серия МПК БИС	Область применения и параметры	Числовые параметры МПК и их относительные показатели														
		число источников питания	энергопотребление, Вт	наличие регистра состояния	число внутренних регистров	температура эксплуатации, °С (max)	площадь печатной платы, мм²	аппаратные затраты БКС	длина умножения (число чисел)	длина деления (число чисел)	время на умножение, мкс	время на деление, мкс	разрядность управляющ. слова, бит	Q_1 (см. (4.4))	Q_2 (см. (4.4))	место предпочтительности в ряду
		Эталонные значения параметров для областей применения														
К585	K_{ji}	1	15	0	11	85	15000	24	10	13	26	30	18			
	I	0	0,5	1	0,08	0,06	0,25	0,1	0	0,07	0,74	0,7	0,12	3,62	1,54	1
	II	0	6,5	1	0,31	0,41	0	0,55	4	5,5	0	0,13	1,25	19,65	9,5	3
	III	0,66	2	1	0,57	0,32	0,5	0,6	0,8	0,74	0,94	0,94	1,25	10,32	3,31	3
К588	K_{ji}	1	2	1	16	85	17300	28	12	15	272	285	16			
	I	0	0,8	0	0,33	0,06	0,13	0,08	0,2	0,07	1,7	1,85	0	5,22	2,49	2
	II	0	0	0	0	0,41	0,15	0,46	5	6,5	8,0	8,5	1	30,02	12,19	4
	III	0,66	0,6	0	1,2	0,32	0,73	0,92	0,76	0,7	0,45	0,43	1	7,77	2,48	2
К1804	K_{ji}	1	13	0	16	125	15000	29	2	3	2,21	2,34	41			
	I	0	0,3	1	0,33	0,56	0,25	0,26	0,8	0,78	0,97	0,97	1,5	7,72	3,2	5
	II	0	5,5	1	0	1	0	0,63	0	0,5	0,92	0,9	4,1	14,5	7,16	2
	III	0,66	1,6	1	1,2	0	0,5	0,32	0,96	0,94	0,9	0,9	4,1	13,08	5,1	5

§ 4.3. ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ ВСТРАИВАЕМЫХ МПСУ НА МИКРОПРОГРАММИРУЕМЫХ И ОДНОКРИСТАЛЬНЫХ МИКРОПРОЦЕССОРАХ

Известно, что любой алгоритм, отличающийся составом функциональных операторов $A=\{A_1, \ldots, A_k\}$ и набором логических условий $Z=\{z_1, \ldots, z_l\}$, может быть программно реализован с помощью различных систем команд (микрокоманд) G, присущих конкретным типам микропроцессоров и являющихся функционально полными относительно заданного алгоритма или класса алгоритмов $\{E\}$. При этом каждой системе команд G соответствует некоторая i-я архитектура МПСУ, характеризующаяся определенными параметрами (аппаратными затратами R_i, временем реализации заданного алгоритма T_i, стоимостью C_i, энергопотреблением P_i, надежностью H_i и др.). Эти параметры должны удовлетворять заданным техническим ограничениям ($R_i \leqslant R_{\text{доп}}$; $T_i \leqslant T_{\text{доп}}$; $C_i \leqslant C_{\text{доп}}$). Обычно при проектировании систем управления на базе МПК БИС ставится задача обеспечить: а) заданное (минимальное) время реализации алгоритма при минимальных (заданных) аппаратных затратах; б) минимальную стоимость МПСУ при заданных ограничениях на затраты оборудования и время реализации алгоритма.

Рассмотрим, как решается эта задача при проектировании встраиваемых МПСУ на основе МПК БИС с использованием микропрограммируемых и однокристальных МП. Условимся, что этапы изучения и описания объекта и устройства управления уже выполнены с уточнением размерности системы управления, детализацией структуры и алгоритма функционирования.

Тогда исходными данными для разработчика МПСУ являются:

а) совокупность БКС МПСУ, каждая из которых характеризуется соответствующим набором элементов и связей между ними $\{F_0\}$, а также системой микрокоманд $\{G_0\}$ и множеством технических ограничений $\{R_0, T_0, C_0, P_0, H_0, \ldots\}$;

б) алгоритм $A_i \in \{E\}$, записанный на одном из формализованных языков и подлежащий реализации на проектируемой МПСУ за время T_0 (при полной реализации всего алгоритма) или за время T_{0i} (при реализации отдельных частей этого алгоритма);

в) допустимые значения параметров ограничений, обусловливаемые спецификой конкретного применения $\{R_{\text{доп}}, T_{\text{доп}}, C_{\text{доп}}, P_{\text{доп}}, H_{\text{доп}}, \ldots\}$.

Обычно задача выбора рациональной структуры МПСУ возникает при несоответствии одного или нескольких параметров конкретной архитектуры МПСУ заданным техническим ограничениям. Чаще всего это происходит при недостатке быстродействия:

$$T_{0l} > T_{\text{доп}\,i}, \quad \sum_{i \in N} T_{0l} > T_{\text{доп}\,i},$$

где $N=\overline{1,n}$ — число операторов A_i и логических условий, входящее в состав реализуемого алгоритма.

Повысить быстродействие выполняемого алгоритма $\{E\}$ в конкретной МПСУ с параметрами R_0, T_0, G_0, Q_0, ... можно либо введением дополнительных микрокоманд $G=\{q_k\}$, $k=\overline{1,m}$ (q_k не принадлежит $\{G_0\}$), либо введением дополнительных схемных элементов или связей $F=\{f_k\}$, $k=\overline{1,n}$ (f_k не принадлежит $\{F_0\}$), либо введением того и другого одновременно $Q=G\vee F$, $Q=\{q_k\vee \vee f_k\}$. Введение каждой дополнительной микрокоманды q_k или нового дополнительного элемента схемы или связи f_k приводит к сокращению времени выполнения отдельного оператора или участка алгоритма $A_i\in\{E\}$ на величину t_{ik} за счет роста аппаратных затрат на величину r_k. Данное условие математически можно записать следующим образом:

$$T_i^*=T_{0i}-\sum_{k=1}^{m} t_{ik}X_k;\quad R=R_0+\sum_{k=1}^{m} r_iX_k,$$

$t_{ik}=$ { t_{ik}, если элемент f_k или микрокоманда q_k участвуют в выполнении оператора A_i;
0 в противном случае;

$X_k\in\{1, 0\}$ — целочисленная (булева) переменная;

$X_k=$ { 1, если дополнительный элемент q_k f_k включается в множество $\{Q\}$,
0 в противном случае.

При использовании *микропрограммируемых МП с вертикальным микропрограммированием* повышения быстродействия добиваются путем введения дополнительных микрокоманд q_k (q_k не принадлежит $\{G_0\}$, $k=\overline{1,m}$) с соответствующим ростом аппаратных затрат на величину r_k и сокращением времени выполнения оператора A_i заданного алгоритма $\{E\}$ на величину t_{ik}. Заметим, что введение дополнительных микрокоманд q_k способствует сокращению числа управляющих слов L_0 на величину l_k, т. е.

$$L=L_0-\sum_{k=1}^{m} l_kX_k,$$

где L_0 — число управляющих слов базовой конфигурации МПСУ.

При использовании микропрограммируемых *МП с горизонтальным микропрограммированием* повышения быстродействия добиваются путем введения дополнительных элементов q_k, блоков или связей f_k с одновременным добавлением соответствующих управляющих полей в базовую структуру микрокоманды. Здесь, ана-

логично ММП с вертикальным микропрограммированием, введение f_k, обусловливая увеличение аппаратных затрат на величину r_k, способствует сокращению времени выполнения оператора $A_i \in \{E\}$ на величину t_{ik}, влияя одновременно на длину D микрокоманды и объем управляющей памяти V, т. е.

$$D = D_0 + \sum_{l=1}^{m} d_k X_k; \quad V = V_0 + \sum_{l=1}^{m} v_k X_k,$$

где D_0, V_0 — длина микрокоманд и объем управляющей памяти в БКС МПСУ.

При использовании *однокристальных МП (ОМП)* повышения быстродействия добиваются путем введения дополнительных операционных (схемных) элементов f_k, позволяющих относительно быстро выполнять «трудные» операции A_i (деление, умножение, вычисление специальных функций и др.). Здесь каждый элемент f_k подключается к МП и обслуживается, как обычное УВВ. При этом введение каждого дополнительного элемента f_k способствует увеличению аппаратных затрат на величину r_k и сокращению времени выполнения операций A_i на величину t_{ik}.

Исходными величинами при определении рациональной структуры на базе ОМП являются: время выполнения «трудной» операции A_i, подлежащей ускорению, — T_i; время выполнения ее с помощью дополнительного элемента f_k — T_i^k; число команд, необходимых для передачи как операндов из МП в элемент f_k, — $\eta^k_{\text{вв } i}$, так и результата из элемента f_k в МП, — $\eta^k_{\text{выв } i}$.

Тогда выигрыш во времени выполнения операции A_i за счет элемента f_k может быть определен как

$$t_{ik} = T_i - (T_i^k - T^k_{\text{вв}\,i}) = T_i T_i^k + t_0 (\eta^k_{\text{вв}\,i} + \eta^k_{\text{выв}\,i}),$$

где t_0 — длительность машинного такта МПСУ при работе с УВВ.

Выигрыш в сокращении числа управляющих слов на величину l_k благодаря введению элемента f_k оценивается величиной

$$l_k = \eta_i - (\eta^k_{\text{вв}\,i} + \eta^k_{\text{выв}\,i}),$$

где η_i — число УС, требующееся для реализации операции A_i.

Изменение объема управляющей памяти для ОМП при разрядности $n=8$ оценивается величиной, равной $V_k = 8l_k$.

§ 4.4. ПРОЕКТИРОВАНИЕ АППАРАТНЫХ СРЕДСТВ МПСУ, РЕАЛИЗУЕМЫХ НА ОСНОВЕ МПК БИС

При проектировании аппаратных средств МПСУ на основе МПК БИС широко применяются проверенные на практике типовые решения, касающиеся реализации шин (магистралей), подсистем памяти и ввода — вывода. Достаточно полное их освещение проведено в ряде источников, например в [2, 5, 14, 20, 24 и др.].

В связи с многообразием имеющихся типов МП, УВВ, аппаратных средств, способов реализации отдельных компонентов МПСУ рассмотрим лишь некоторые обобщенные решения.

При проектировании раздельных микропроцессорных шин адреса (ША), данных (ШД), управления (ШУ) следует учитывать допустимую величину токовой нагрузки для каждой линии и при превышении ею допустимого значения выходной нагрузки МП устанавливать буферные элементы или шинные формирователи. При

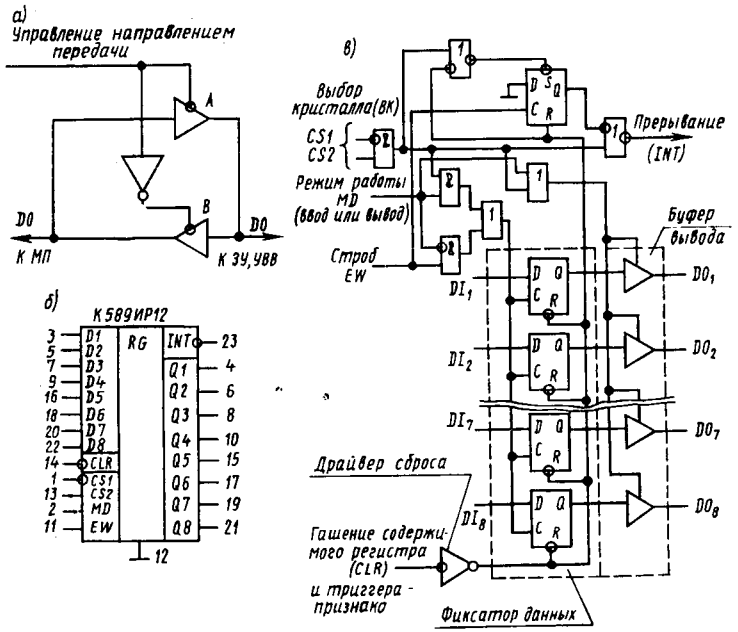


Рис. 4.8. Структура и обозначение буферов и фиксаторов:
а — двунаправленный буфер; *б, в* — условное обозначение и структура регистра К589ИР12

выборе буфера учитывать особенности его реализации (неинвертирующий, инвертирующий, с открытым коллектором, с тремя состояниями и др.), применять многовходовые трехстабильные буферы или схемы с открытым коллектором.

Для МП с общей магистралью следует осуществлять предварительную фиксацию логического состояния мультиплексированных адресных линий путем применения трехстабильных буферов-фиксаторов с управляемым импульсным запуском и выдачей адресных данных по фронту или спаду соответствующего сигнала управления; обеспечивать согласование нагрузки по току для входов выбранного фиксатора и мультиплексируемых выходов МП.

Для шины данных, обеспечивающей работу МП в режиме ввода и вывода информации, использовать двунаправленные трехстабильные буферы (рис. 4.8, *а*), имеющие вход «Управление направлением передачи» буферированных данных. В качестве буферов ШД использовать микросхемы К589ШФ16, К589ШФ26, К589ИР12 (рис. 4.8, *б*, *в*) и др.

Для шины управления МП с разделением функций доступа к памяти и периферийным УВВ учитывать условия формирования сигналов управления при выполнении каждой из функций МПСУ; рассчитывать допустимую нагрузку, устанавливая буферы на линиях управления при необходимости. Для ШУ МП с совмещением функций доступа к памяти и УВВ соответствующие сигналы управления формировать в комбинации с сигналами запись-чтения, сигналом достоверности адреса памяти, тактовым сигналом и др., что способствует устранению ложной записи информации в память МПСУ или уточнению состояния всех сигналов на ШУ.

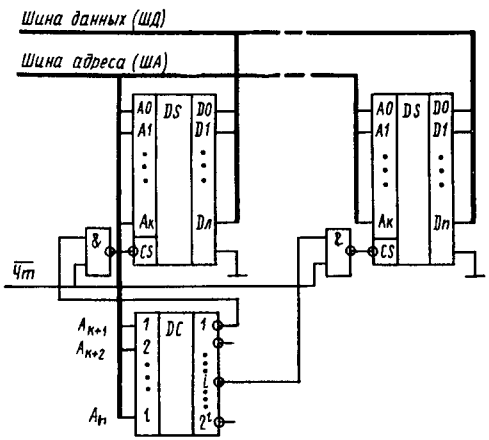


Рис. 4.9. Организация интерфейса модулей БИС ПЗУ (ППЗУ)

При проектировании генераторов тактовых импульсов (ГТИ) следует использовать специализированные микросхемы, выполняющие функции ГТИ, с выбором характеристик и подключением кварцевых резонаторов в соответствии с техническими рекомендациями.

При неправильном функционировании или частых сбоях МПСУ в первую очередь проверять соблюдение правильности значений подаваемых уровней напряжения и выполнения всеми последовательностями тактовых импульсов необходимых временны́х соотношений.

При проектировании подсистемы памяти МПСУ, компонуемой из модулей БИС ОЗУ и ПЗУ, необходимо составить схему распределения памяти, в соответствии с которой осуществить синтез модулей памяти на основе БИС ОЗУ и ПЗУ (см. § 3.4). При проектировании интерфейса модуля памяти из быстродействующих БИС ПЗУ, для которых время доступа к данным меньше, чем время, выделяемое для этого МП, адресные выводы и выводы выдачи данных для всех БИС ПЗУ поразрядно соединить с соответствующими линиями ША и ШД, а входы выборки кристалла соответ-

ствующих БИС ПЗУ — с выходом двухвходовой схемы (И — НЕ) (рис. 4.9), на которую воздействуют сигналы, поступающие с линии управления чтением из памяти и с выхода дешифратора адреса. При проектировании интерфейса памяти из медленно действующих БИС ПЗУ, для которых время доступа к данным больше, чем время, выделяемое для этого МП, рассмотреть возможность замедления нормальной работы МП путем перевода его в состояние ожидания или изменения частоты ГТИ для правильного обмена информацией (например, для БИС МПК 1804).

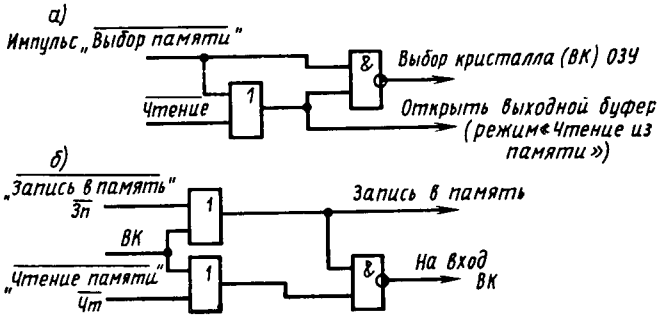


Рис. 4.10. Получение сигналов управления для ОЗУ:
а — в режиме чтения; *б* — в режиме записи

При проектировании интерфейса модуля оперативной памяти из статических БИС ОЗУ, характеризующихся наличием раздельных входов и выходов и отсутствием внутренних схем интерфейса, на выходах БИС ОЗУ устанавливать трехстабильные буферы для отключения выходов ОЗУ от ШД в режимах записи и хранения информации. Управляющие сигналы ОЗУ и буферов формировать согласно схеме рис. 4.10. При наличии в БИС ОЗУ буферных схем, формирующих сигналы управления с отключением выходов ОЗУ от ШД в режимах записи и хранения, поразрядно соединить входные и выходные выводы устройства памяти и подключить их к ШД.

В режиме записи данных в память на адресные линии ОЗУ подать код конкретного адреса памяти; на линии ввода данных в ОЗУ подать данные, на линию разрешения записи в память подать сигнал, устанавливающий соответствующий (низкий или высокий) уровень напряжения, на линию выбора кристалла подать соответствующего уровня сигнал. Если в оперативной памяти используются БИС ОЗУ, характеризующиеся меньшей разрядностью и отсутствием в БИС выходных буферов, то следует сформировать сигналы управления разрешением записи и выбора кристалла, линии подачи адреса и ввода данных ОЗУ соединить с соответствующими линиями ША и ШД, выходы ОЗУ подключить к входам трехстабильного

буфера, выходы буфера подключить к соответствующим входам этого же БИС ОЗУ для ввода данных.

В режиме считывания данных из модуля памяти на адресные линии ОЗУ подать адрес памяти; на линии управления разрешением чтения из памяти и выбора кристалла подать сигналы соответствующего уровня; открыть выходные буферы ОЗУ и выдать данные на ШД; после операции чтения данных из ОЗУ линию разрешения чтения в системной ШУ перевести в нерабочее состояние, а буфер на выходе ОЗУ — в высокоимпедансное состояние.

При проектировании интерфейса модуля оперативной памяти из статических БИС ОЗУ с общим входом и выходом необходимо: а) в режиме записи данных в ОЗУ внутренние выходные линии блокировать с помощью внутренних схем ОЗУ для перевода их в высокоимпедансное состояние путем одновременной подачи управляющих сигналов соответствующего уровня как на линию разрешения записи, так и на линию выбора кристаллов; б) в режиме считывания данных из ОЗУ сформировать управляющие сигналы по схеме (рис. 4.10).

При проектировании подсистем ввода—вывода МПСУ в качестве аппаратных средств используются:

1. Для простых малоскоростных типов УВВ (переключателей, индикаторов, клавишных пультов и др.) — *фиксаторы*, реализуемые на основе D-, RS-триггеров; *одновибраторы, формирователи* сигналов управления автономными УВВ, не находящимися под управлением МП (малоскоростные механические или электромеханические УВВ, высокоскоростные генераторы сигналов, аппаратура связи и др.).

2. Для УВВ с повышенными сложностью и скоростью функционирования — *дешифраторы*, *селекторы-мультиплексоры*, обеспечивающие управление модулями памяти, портами УВВ, а также преобразование входных кодов в стандартные выходные коды (BCD, ISO, ASCII); *счетчики* с различными функциональными возможностями и режимами функционирования (отсчет в прямом и обратном направлениях, параллельный синхронный вывод информации, очистка или предварительная установка с трехстабильным выходом и др.); *сдвиговые регистры*, расширяющие функциональные возможности ввода — вывода, тактирование, организацию задержек, преобразование данных из параллельного формата в последовательный и наоборот.

3. Для более сложных типов УВВ с повышенными и высокими скоростями функционирования — *порты ввода — вывода* (многорежимные буферные регистры), управляющие выбором конкретных УВВ с установлением требуемой конфигурации структуры и прямой связи с двунаправленными шинами при использовании синхронных фиксаторов, трехстабильных буферов, триггеров, фиксирующих сигналы «Готово», «Открыто», «Очистка» и др; *универсальные программируемые синхронно-асинхронные приемопередатчики,* управ-

ляющие последовательными УВВ с выполнением ряда типовых операций и функций (генерации бита четности, проверки четности, добавления битов «старт» и «стоп» для маркировки начала и конца передач, выработки сигналов управления стандартными интерфейсами); *программируемые параллельные интерфейсы*, осуществляющие организацию режимов прерывания, ПДП, прямой связи с системными шинами, обмен данными в параллельном формате практически с любым УВВ при наличии в структуре буферов и фиксаторов, регистров управления, схем выработки сигналов состояния и синхронизации.

При проектировании интерфейса между МП и простыми УВВ используются следующие типовые решения:

а) в устройствах ввода типа кнопочных переключателей применяются управляемые трехстабильные буферы, RS-триггеры (рис. 4.11, *а*, *б*);

б) для многопозиционных переключателей без собственных средств кодирования информации о положении каждый из выводов переключателей подключается к ШД через свой трехстабильный буфер; при наличии в переключателях средств кодирования информации о положении переключателя для связи с МП устанавливается только один трехстабильный буфер;

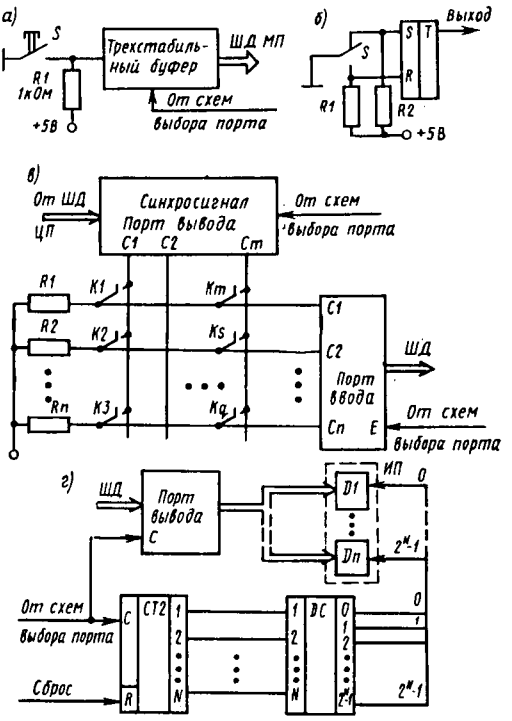


Рис. 4.11. Организация интерфейса МП и простых УВВ:

а, б — для одно- и двухпозиционных переключателей, *в* — для матричных $m \times n$ клавиатур *г* — для многосегментных цифро-буквенных индикаторных устройств

в) для переключателей, объединенных в клавиатуру, применяются: для клавиатур до 16 клавиш — независимое подсоединение выводов переключателей; для более мощных клавиатур — ТТЛ-кодеры на базе СИС; для клавиатур большого объема — МОП-кодеры на базе ПЗУ (ППЗУ); для матричных $m \times n$-клавиатур (рис. 4.11, *в*) — последовательное (или одновременное) заземление каждой линии (всех линий) строк или столбцов через порты ввода и вывода соответственно.

В *индикаторных устройствах вывода* применяют: а) для управления светодиодами транзисторы или предоконечные усилители с целью нормального функционирования портов ввода — вывода; программно-реализованные процедуры задержки или таймеры, управляющие генерацией импульсов тока заданной длительности и скважности; б) для многосегментных цифро-буквенных индикаторов — БИС ПЗУ, выполняющие функции генераторов символов русского и латинского алфавитов, цифр и знаков; в) при кратковременном формировании импульсов управления индикаторами обычно используют один порт вывода (рис. 4.11, *г*).

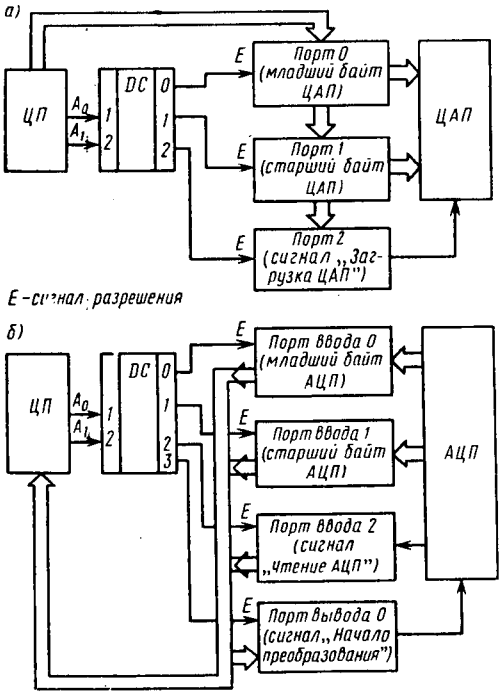


Рис. 4.12. Организация интерфейса МП с ЦАП и АЦП:
а — интерфейс ЦАП; *б* — интерфейс АЦП

При проектировании интерфейса устройств связи с объектом (УСО) используют два следующих типовых решения:

1. Организацию интерфейса МП с ЦАП, преобразующими цифровые входные данные в выходные аналоговые сигналы для управления разнообразными УВВ (двигателями, реле, графопостроителями и др.), осуществляют следующим путем:

а) при использовании ЦАП с разрядностью адресных входов, меньшей разрядности МП, применяют несколько ЦАП при параллельной пересылке информации в адресуемые буфера ЦАП и отпирании их по одному управляющему сигналу;

б) при использовании ЦАП с разрядностью адресных входов, превышающей разрядность МП, загрузку ЦАП проводят как одну операцию с отпиранием адресных входных буферов ЦАП по одному управляющему сигналу или раздельно (двумя байтами) с передачей старшего и младшего байтов через различные выходные порты и отпиранием каждого из буферов по своему сигналу управления (рис. 4.12, *а*).

2. Организацию интерфейса МП с АЦП, преобразующими аналоговые входные сигналы от датчиков, первичных преобразователей

и других источников в цифровые и имеющими в своей структуре трехстабильные выходы и раздельные входы для управления каждым байтом, осуществляют путем считывания данных с АЦП за одну или две операции ввода (рис. 4.12, *б*).

В связи с многообразием типов УВВ согласованное функционирование МП и УВВ обеспечивается использованием унифицированных интерфейсов (см. табл. 5.1), отличающихся структурой управления, системой шин, типом сигналов и др.

Контрольные вопросы

1. В чем состоит сложность синтеза систем логического управления на основе МПК БИС в условиях многокритериальности? 2. Какие методы используются для выбора предпочтительных типов МПК БИС? 3. В чем проявляются особенности алгоритма проектирования МПСУ на базе МПК БИС? 4. В чем заключается сложность определения рациональной архитектуры МПСУ при использовании различных типов МП и каким образом она решается? Какие основные параметры учитываются при этом? 5. Перечислите основные типовые решения, рекомендуемые для проектирования следующих компонентов встраиваемых МПСУ на МПК БИС: магистралей, подсистемы памяти, подсистемы ввода — вывода.

ГЛАВА 5

ПРОЕКТИРОВАНИЕ СИСТЕМ УПРАВЛЕНИЯ НА ОСНОВЕ МИКРОЭВМ

Проектирование систем управления на основе законченных серийных МЭВМ исключает многие трудности, связанные с сопряжением и размещением МПК БИС, позволяя разработчику целиком сосредоточить свое внимание на разработке ПО и оптимизации структуры системы в целом. При этом особенно важен выбор рациональной структуры КТС МПСУ, наиболее полно соответствующей требованиям технического задания, специфике решаемых задач, информационной базы и условий эксплуатации.

§ 5.1. ОСОБЕННОСТИ ОПТИМИЗАЦИИ СТРУКТУРЫ МПСУ И ВЫБОРА МИКРОПРОЦЕССОРНЫХ КОМПЛЕКСОВ ТЕХНИЧЕСКИХ СРЕДСТВ

Известен ряд подходов к оптимизации и выбору рациональной структуры КТС, сводящихся к задачам линейного программирования, определению эффективности того или иного варианта системы по каким-либо частным критериям и др., что в целом приводит лишь к частичной оптимизации структуры систем [23, 30]. Предпочтителен комплексный подход к выбору и оценке структур КТС, позволяющий выявить рациональную структуру МПСУ из множества возможных или ограниченных условиями применения вариантов

структур. Рассмотрим особенности применения микроЭВМ в системах управления и основных подходов к выбору структур МПСУ.

Особенности применения микроЭВМ в системах управления. Использование микроЭВМ в системах управления имеет отличительные особенности по сравнению с использованием ее в качестве универсальной МЭВМ в системах обработки информации и научно-технических расчетов, где она ориентируется в первую очередь на диалог с пользователем и обрабатывает данные по запросу пользователя. Поэтому в универсальной МЭВМ через блоки сопряжения подключаются разнообразные УВВ: алфавитно-цифровые и графические дисплеи, печатающие устройства, графопостроители, кодировщики, функциональные клавиатуры, а также устройства внешней памяти для хранения программ и данных на перфолентах, гибких магнитных дисках, магнитных лентах.

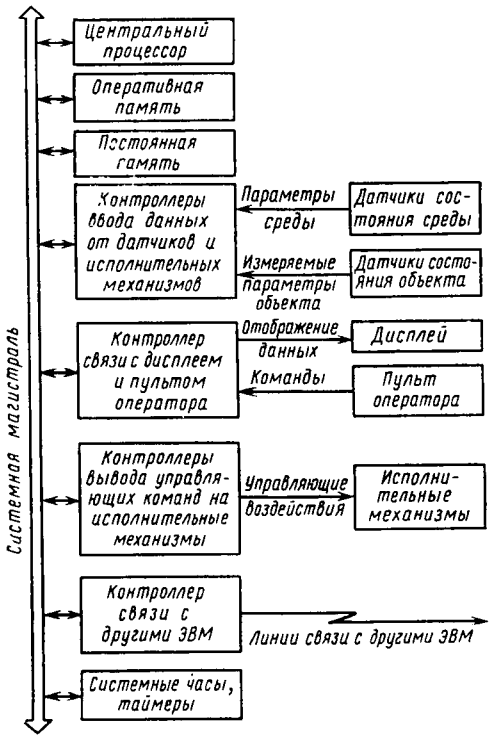


Рис. 5.1. Структурная схема системы управления на базе микроЭВМ

Основная же задача управляющей МЭВМ состоит в том, чтобы на основании информации, получаемой от датчиков, и других логических условий вычислить и передать на исполнительные механизмы управляющие воздействия. Обычно управляющие МЭВМ встраиваются в технологическое оборудование, настраиваются на конкретную область применения и работают по уже готовым программам, хранимым в ПЗУ или ППЗУ. В состав управляющей МЭВМ (рис. 5.1) обязательно входят контроллеры для приема информации от датчиков состояния окружающей среды и объекта управления, а также для передачи управляющих воздействий на исполнительные механизмы. Так как часть датчиков состояния среды и объекта может выдавать аналоговую информацию в виде сигналов, которые могут принимать произвольные значения, то осуществляется аналого-цифровое или дискретно-цифровое преобразование с использованием АЦП и ДЦП. Для передачи управляющих воздействий на испол-

нительные механизмы информация из цифрового кода с помощью ЦАП или ЦДП преобразуется контроллером соответственно в аналоговый или дискретный сигнал. Для связи с дисплеем и пультом оператора применяется свой контроллер. Однако в отличие от универсальной алфавитно-цифровой клавиатуры пульт оператора содержит в основном специальные клавиши, выполняющие определенные функции. Кроме того, для связи с другими управляющими МЭВМ и ЭВМ более высокого уровня применяется контроллер связи с ЭВМ.

Отличительной чертой функционирования управляющих микроЭВМ является выполнение ими всех операций в реальном масштабе времени. Вычисление управляющих воздействий за время больше требуемого здесь приравнивается к получению неправильного результата, поскольку МЭВМ должна оперативно управлять объектом.

В целом при работе микроЭВМ в составе системы управления осуществляется решение следующих задач:

прием информации о текущем состоянии системы, окружающей среды и объекта;

расчет в реальном масштабе времени управляющих воздействий и передача их на исполнительные механизмы;

отображение информации о состоянии системы и объекта оператору на дисплее и других вспомогательных устройствах;

прием и обработка команд оператора по изменению условий процесса управления;

взаимообмен информацией с другими ЭВМ и др.

Каждая из этих задач решается с помощью своей программы, выполняемой в определенное время по мере необходимости, с учетом присвоенного приоритета. Например, при поступлении сообщения о наступлении момента опроса датчиков выполнение текущей программы может прекращаться, запускаться программа опроса датчиков, затем программа расчета управляющих воздействий на исполнительные механизмы и последующее возвращение к прерванной программе. При этом к программному обеспечению МЭВМ в системах управления предъявляются повышенные требования по надежности и отказоустойчивости, так как от его стабильной работы зависят эффективная работа МПСУ, экономические и другие потери.

Особенности оптимизации систем управления на базе микроЭВМ. В общем случае любые системы по степени их сложности подразделяются на *простые* (с небольшим числом состояний, легко поддающимся описанию), *сложные* (с разнообразием внутренних связей и возможностью их описания) и *большие* (с разнообразным числом связей и своеобразием отношений между элементами, таким, что нельзя все их выявить и проанализировать).

МПСУ по их функциональным свойствам можно отнести к

классу сложных систем, поэтому для их оптимизации применяют как однокритериальные способы («свернутые», «скалярные»), так и многокритериальные (векторные) и др. (один из них см. § 4.2). Однако наиболее важными и подходящими для МПСУ являются способы формирования доминирующих решений (множеств Парето) и последовательного выбора уступок.

Метод формирования множеств доминирующих решений требует больших затрат: если известно N вариантов реализации сложной системы $T_1, \ldots, T_N$ и можно вычислить M показателей, по которым решено оценивать варианты N_m $(m=\overline{1, M})$, то сравнение между собой любых двух вариантов позволит определить, превосходят ли показатели одного варианта $N_1, \ldots, N_m$ соответствующие показатели другого. Если это выполняется, то вариант с лучшими показателями может рассматриваться как доминирующий, если же такой определенности нет (один вариант лучше по одним показателям, другой — по другим показателям или оба равноценны), то необходимо продолжать сравнение применительно к новым сочетаниям вариантов. Таким образом, для установления для данного варианта n $(1 \leqslant n < N)$ хотя бы одного доминирующего варианта необходимо исследовать все сочетания из N вариантов по два, т. е. $N(N-1)/2$.

На практике пользуются методом поэтапной векторной оптимизации, относящейся к категории векторных по числу применяемых критериев качества (КК), пользуясь понятием «нехудших» вариантов.

При определении «нехудших» вариантов сложных систем используются ограничения трех типов: 1) требования конструктивно-технологической совместимости подсистем (тип 1); 2) ограничения, накладываемые на ряд параметров, значения которых должны быть не больше (не меньше) значений ограничений на параметры, задаваемые большой системой (тип 2); 3) ограничения (тип 3), накладываемые на ряд параметров подсистемы (когда увеличение или уменьшение значений параметров относительно соответствующих значений ограничений на параметры, накладываемые большой системой, не приводит к улучшению качества большой системы в целом). Помимо перечисленных ограничений может быть задана неравнозначность параметров сложной подсистемы, которая также влияет на выбор числа оптимальных вариантов.

Задача векторной оптимизации сложной подсистемы формулируется следующим образом. Пусть имеется N вариантов сложной подсистемы с номерами $n=\overline{1, N}$, входящей в большую систему. Каждый вариант характеризуется вектором частных критериев качества q_{nm}, $m=\overline{1, M}$. Тогда каждому варианту сложной подсистемы соответствует вектор качества $\overline{Q}_n$ (ВК). Со стороны большой подсистемы накладываются три типа ограничений, перечисленные выше. В случае неравнозначности параметров сложной подсистемы мо-

жет быть задана матрица учета весов параметров $\|S\| = (S_{nm})$, члены которой $0 \leqslant S_{nm} \leqslant 1$.

В целом методика поэтапной векторной оптимизации сложной подсистемы состоит в следующем:

1) исключаются из рассмотрения варианты подсистемы, не обеспечивающие конструктивно-технологическую совместимость;

2) исключаются из рассмотрения варианты подсистемы, если хотя бы для одного параметра которых не выполняются ограничения типа 2;

3) для вариантов сложной подсистемы, ограниченных по типу 3, при выполнении условий $q_{nm} > q_m$ при ограничении «не больше» и $q_{nm} < q_m$ при условии «не меньше» присваивается $q_{nm} = q_m$;

4) если после выполнения п. 1) — 3) существуют параметры, значения которых одинаковы для всех оставшихся в рассмотрении вариантов, то они исключаются из дальнейшего рассмотрения;

5) в случае исключения вариантов сложной подсистемы в п. 1) и 2) изменяются номера оставшихся вариантов; в аналогичных ситуациях в п. 4 изменяют номера параметров, учитывая исключенные из рассмотрения;

6) на каждом этапе векторной оптимизации параметры нормируются таким образом, чтобы $0 \leqslant \bar{q}_{nm} \leqslant 1$, $\bar{q}_{nm} = q_{nm}/\max q_{nm}$ в том случае, если качество сложной подсистемы улучшается при увеличении параметра q_{nm}, и $\bar{q}_{nm} = \min q_{nm}/q_{nm}$ в том случае, если качество ее ухудшается;

7) для всех рассмотренных на данном этапе векторной оптимизации вариантов сложной подсистемы вычисляются значения критерия $\bar{O}_n$ оптимизации, значения которого при равнозначности параметров равно

$$\bar{Q}_n = \sum_{m=1}^{M} q_{nm}. \qquad (5.1)$$

В случае неравнозначности параметров сравниваемых вариантов сложных подсистем в (5.1) включается матрица учета весов параметров, задавамая большой системой $\|S\| = (S_{nm})$, где $0 \leqslant S_{nm} \leqslant 1$;

8) последовательно начиная с варианта, имеющего минимальное значение $\bar{Q}_n$, просматриваются «худшие» варианты сложной подсистемы; определяется наихудший вариант, имеющий $\bar{q}_{nm} = 1$, исключение которого из числа рассматриваемых повлечет изменение величин $\bar{q}_{nm}$ остальных вариантов;

9) все «худшие» варианты из дальнейшего рассмотрения исключаются;

10) если остается только один вариант, то он считается лучшим; если несколько вариантов имеют одинаковое значение $\bar{Q}_n$, то это множество «нехудших» вариантов, лучший вариант из которых по данной методике определить нельзя, а если различные значения, то необходимо перейти к п. 6).

Как указывалось выше, обобщенный критерий оптимизации сложных систем, каковыми являются МПСУ, зависит от большого числа частных критериев качества, имеющих различный физический смысл, и требует процедур сравнительного анализа при выборе наиболее эффективного варианта. При учете важности того или иного КК в процессе определения обобщенного критерия оптимизации используются экспертные знания, например при определении значений «весов» (коэффициентов неравнозначности) КК. Однако даже самые квалифицированные эксперты из-за субъективности оценки значений КК вносят существенную погрешность в результаты анализа. Повышения объективности оценки можно добиться, например, применением методов многокритериальной оптимизации сложных систем с использованием экспертных знаний для ранжирования частных КК в порядке убывающей важности.

Данные экспертов могут описываться различным образом: а) на естественном языке в форме высказываний вида «критерий K_i слабо (сильно) значимее критерия K_j»; б) отношений предпочтительности $K_i \succ K_j$; в) числовых «весовых» оценок; г) математическим языком на основе теории нечетких множеств и др.

Одним из новых является метод, в котором цели и ограничения, накладываемые на систему, рассматриваются как нечеткие множества в пространстве решений, что позволяет не делать различий между ними при формировании решений.

Нечеткие множества решений имеют вид (5.2), и для каждого этапа решения $\tilde{F} \in F$ имеем: $\tilde{F} = \{f, \mu_F(f)\}$, где $f \in F$ при известных нечетких целях $\tilde{C}$, имеющих функцию принадлежности $\mu_{\tilde{C}}(f) : F \to \{0, 1\}$, и нечетких ограничениях $\tilde{D}$, имеющих функцию принадлежности $\mu^1(f) : F \to \{0, 1\}$. При заданных $\tilde{C}$ и $\tilde{D}$ $\mu_F(f)$ определяется на основе анализа значений критериев качества решений и конкретно сформулированных в техническом задании ограничений как

$$\mu_{\tilde{F}}(f) = \min \{ [\mu_{\tilde{C}}(f_1)^{\alpha_1} \ldots \mu_{\tilde{C}}(f_i)^{\alpha_i} \ldots \mu_{\tilde{C}}(f_n)^{\alpha_n}], \\ [\mu_{\tilde{D}}(f_1^{\alpha_1} \ldots \mu_{\tilde{D}}(f_i)^{\alpha_i} \ldots \mu_{\tilde{D}}(f_n)^{\alpha_n}] \}, \qquad (5.2)$$

где α_i — оценка важности i-го критерия (цели и ограничения).

При выборе оптимального решения с помощью соответствующих α стремятся сделать функцию $\mu_{\tilde{F}}(f)$ зависимой от наиболее важных критериев. Достоверность экспертной информации определяется методом получения данных от эксперта, для чего рекомендуется: 1) предложить эксперту оценить важность критериев, сравнивая их между собой с составлением бинарной матрицы; 2) восстанавливать по бинарной матрице важностей относительные важности критериев с помощью формализованных алгоритмов.

В целом погрешность данных методов значительно ниже погрешности методов, базирующихся на использовании экспертных знаний для определения значений «весов» частных критериев качест-

ва. Невысокие требования, предъявляемые к квалификации экспертов (ранжировать параметры в порядке убывающей важности значительно легче, чем определять значения их «весов»), существенно упрощает формирование и обеспечение работы экспертных групп.

Ниже приведем один из возможных алгоритмов выбора структуры МПСУ, создаваемой на основе микропроцессорных КТС. Алгоритм заключается в следующем.

1. Составить морфологическую матрицу решений по всем выделенным функциональным блокам сложной системы МПСУ и на ее основе выделить набор допустимых условиями эксплуатации решений.

2. Из набора имеющихся или предлагаемых к применению комплексов технических средств МПСУ составить таблицу с перечнем составных элементов $\{s_l | l=\overline{1, N}\}$ отдельных КТС и указанием их сопоставимых количественных характеристик (стоимости, габаритов, электропотребления, надежности, наличия комплектных устройств, периферийного оборудования, внешней памяти и др.).

3. Сформировать множество конкурирующих структур КТС $\{S_i | i=\overline{1, n}\}$ с перечислением составных элементов $\{s_l | l=\overline{1, N}\}$ для каждой структуры $S_k=\{S_1, S_2, ..., S_l, ..., s_N\}$, $k\in\overline{1, n}$.

4. Провести отбор частных критериев $\{K_j | j=\overline{1, m}\}$, по которым оценивается эффективность конкурирующих структур S_i с уточнением или расчетом их количественных и качественных показателей: стоимости, быстродействия МП, объема основной памяти, условных габаритов, энергопотребления, наличия ПО, средств программирования, комплектных устройств, внешней памяти, периферийных УВВ, необходимости разработки нестандартных устройств, комфортности и др.

5. По числу выделенных частных критериев $\{K_j\}$ составить матрицу бинарных предпочтений $A=\|j\times j\|$ с занесением результатов (1 или 0 в зависимости от приоритета критерия строки перед критерием столбца и попарного сопоставления каждого из этих критериев друг с другом), определяемых экспертным путем.

6. Произвести подсчет цены C_j (числа всех единиц в строке) каждого частного критерия K_j и общую цену $\sum_{j=1}^{m} C_j$.

7. Определить «вес» V_j каждого критерия K_j:

$$V_j = C_j \Big/ \sum_{j=1}^{m} C_j,$$

причем $\sum_{j=1}^{m} V_j = 1$.

8. Составить матрицу $B\|j\times i\|$ с занесением в нее конкретных значений параметров «Критерии (K_j) — структуры (S_i)» и подсчета оценок (α_{ji}) по каждой из структур S_i с помощью конкретных единиц измерения (для количественных критериев) или оценок эксперта, определенных каким-либо образом по шкале желательности Харрингтона (для качественных критериев).

9. На основании множества выделенных структур $\{S_i\}$ выделить гипотетическую эталонную структуру $S_э$ с параметрами $\alpha_{эi}$, являющимися наилучшими из набора показателей конкурирующих структур S_i по каждому из критериев K_j.

10. Преобразовать матрицу «Критерии K_j — структуры S_i» с введением относительных оценок P_{ji} по каждой структуре, рассчитываемых как

$P=\alpha_{ji}/\alpha_{эi}$, если увеличение частного критерия K_j улучшает качественные показатели S_i;

$P=\alpha_{эi}/\alpha_{ji}$ в противном случае.

11. Для каждой выделенной структуры S_i вычислить обобщенную скалярную оценку качества Q_i, определяемую по выражению (5.3):

$$Q_i = \left\{ \sum_{j=1}^{m} [(1 - P_{ji})(1 - V_j)]^p \right\}^{1/p}, \qquad (5.3)$$

где p — целое конечное число, равное 1, 2, ..., N.

При $p=1$, 2 скалярная оценка Q_i имеет вид выражения (5.4):

$$Q_1 = \sum_{j=1}^{m} [(1 - P_{ji})(1 - V_j)]; \quad Q_2 = \sqrt{\sum_{j=1}^{m} [(1 - P_{ji})(1 - V_j)]^2}. \qquad (5.4)$$

12. Рациональную или близкую к ней структуру S_{rat} определить по величине $Q_{\text{opt}} = Q_{\min}$.

В последнее время у нас в стране и за рубежом по проблемам проектирования МПСУ, оптимизации структур и выбора комплексов технических средств создаются разнообразные экспертные системы, позволяющие использовать знания специалистов и принимать самостоятельные решения на уровне эксперта-профессионала. Более подробно эти вопросы рассмотрены в [23, 30 и др.].

§ 5.2. ИНТЕРФЕЙСЫ МИКРОПРОЦЕССОРНЫХ СИСТЕМ, ИХ ТИПОВЫЕ СТРУКТУРЫ, ПРИМЕНЕНИЕ И ОРГАНИЗАЦИЯ

При проектировании МПСУ важное место занимает решение проблемы интерфейса, связанной с необходимостью реализации сопряжения как внутренних компонентов МПСУ (МП, подсистем памяти и ввода — вывода), так и подключения внешних периферийных устройств разнообразного назначения и принципа действия, а также объединения МП и МЭВМ в мультимикропроцессорные системы, многомашинные комплексы и сети на основе единых принципов и протоколов обмена информацией [48].

Согласно ГОСТ 15971—84, под *интерфейсом* понимается совокупность правил, устанавливающих единые принципы взаимодействия устройств ЭВМ, некоторый протокол однозначного сопряжения используемых МЭВМ и МП и периферийных устройств различной степени сложности, обеспечивающих информационную, функциональную, электрическую и конструктивную совместимость внутренних и внешних устройств МПСУ на базе комплекса аппаратных и программных средств и унифицированных линий связи.

Известно несколько классификаций интерфейсов различного назначения по ряду отличительных признаков. Например, в ГОСТ 26.016—81 интерфейсы классифицированы по следующим четырем признакам: а) способу соединения компонентов системы (магистральный, радиальный, цепочечный, смешанный); б) способу передачи данных (параллельный, последовательный, параллельно-последовательный); в) режиму передачи данных (односторонняя пе-

редача, двусторонняя одновременная или поочередная); г) принципу обмена информацией (асинхронный, синхронный). Однако более полной систематизация интерфейсов получается при выделении совокупности следующих признаков: функционального назначения; логической и функциональной организации; физической реализации [48].

В соответствии с функциональным назначением интерфейсы подразделяются на: машинные (или системные), интерфейсы для однопроцессорных МЭВМ; интерфейсы периферийных УВВ; интерфейсы мультимикропроцессорных систем; интерфейсы распределенных вычислительных систем (вычислительных и локальных сетей, распределенных систем управления).

Машинные интерфейсы обеспечивают организацию связи между составными компонентами МЭВМ, их сопряжение и взаимодействие с внешней средой. В зависимости от классов МЭВМ различают: интерфейсы ввода — вывода с разделением информационных каналов к памяти и УВВ; интерфейсы ввода — вывода с объединенным информационным каналом (типа «общая шина»); интерфейсы одноплатных МЭВМ с объединенным информационным каналом к УВВ и ОЗУ, ориентированные на внутриплатное и внутрисхемное применение при сопряжениях МПК БИС, а также функциональных узлов СБИС микроЭВМ.

Интерфейсы периферийных УВВ обеспечивают сопряжение процессоров, контроллеров с разнообразными типами периферийного оборудования, исполнительными механизмами, измерительными приборами, аппаратурой передачи данных. Они подразделяются на группы интерфейсов *радиальной структуры*, применяемых в основном для сопряжения контроллеров с исполнительными механизмами с организацией схем сопряжения «точка — точка», и *магистральной структуры*, применяемых на уровне связи с объектом управления и программируемых контроллеров с организацией схем многоточечного подключения многочисленных измерительных приборов, преобразователей информации, датчиков, пультов операторов и др.

Интерфейсы мультимикропроцессорных систем представляют собой в основном магистральные системы сопряжения с объединением в единый комплекс нескольких процессоров, модулей ОЗУ, контроллеров внешней памяти. Они подразделяются на две группы в соответствии со структурой шин адреса и данных: с раздельными и мультиплексными шинами — и представляют собой внутриблочную систему сопряжения магистральной структуры с высокой пропускной способностью, аппаратной реализацией функций селекции и координации.

Интерфейсы распределенных вычислительных систем обеспечивают интеграцию средств обработки информации, размещенных на значительном расстоянии, и ориентируются на использование в системах различного назначения. Они подразделяются на группы интерфейсов малых локальных и локальных сетей (с длиной магист-

рали от десятков метров до нескольких километров); интерфейсов распределенных систем управления; территориально и географически распределенных сетей ЭВМ (с длиной линии более десятка километров).

По функциональной организации интерфейсы классифицируются по информационному и управляющему каналам с выделением основного и дополнительного признаков, расширяя в целом классификацию, принятую в ГОСТ 26.016—81. Например, по *информационному каналу* классификационными признаками являются:

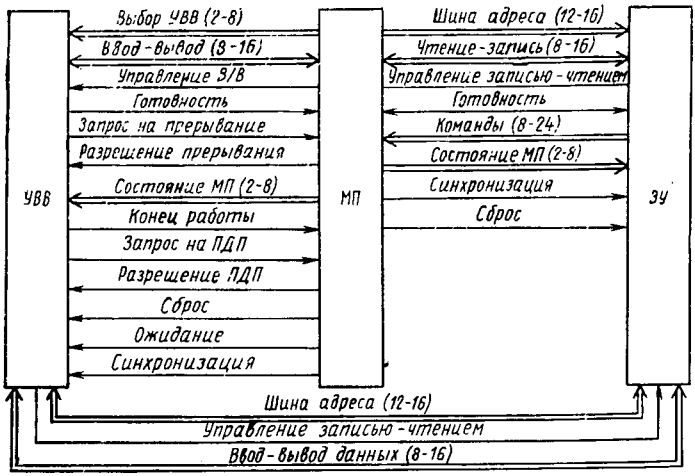


Рис. 5.2. Организация связи МП с ЗУ и УВВ

1) структура шин (магистральная или цепочечная, замкнутая и разомкнутая, комбинированная); 2) способы обмена данными (однобитовый последовательный, побайтовый (1,2,3,4-байтовый) параллельный); 3) режимы передачи (симплексный, двунаправленный дуплексный и полудуплексный); 4) виды совмещения шин (полное разделение, полное или частичное совмещение шин). По *управляющему каналу* классификационными признаками интерфейсов являются: 1) структура управления (централизованная, децентрализованная); 2) принципы селекции (последовательные цепочечный опрос или адресное сканирование, параллельные сравнение приоритета или адресное сканирование, временная селекция); 3) принципы обмена данными (асинхронный с одно- или двухпроводной обратной связью, синхронный без обратной связи).

По конструктивному исполнению интерфейсы подразделяются на: 1) *межблочные,* обеспечивающие сопряжение компонентов на уровне прибора, блока, стойки, шкафа; 2) *внутриблочные* — на уровне плат, субблоков; *внутриплатные* — на уровне СИС, БИС, СБИС; *внутрикорпусные.*

Рассмотрим некоторые из разновидностей перечисленных интерфейсов.

Системные интерфейсы. В структуре связей МЭВМ формируется *внутренний интерфейс*, объединяющий БИС процессора, модулей ОЗУ, ПЗУ, управления вводом — выводом, и *внешний интерфейс*, обеспечивающий сопряжение между внутренней шиной и периферийными устройствами.

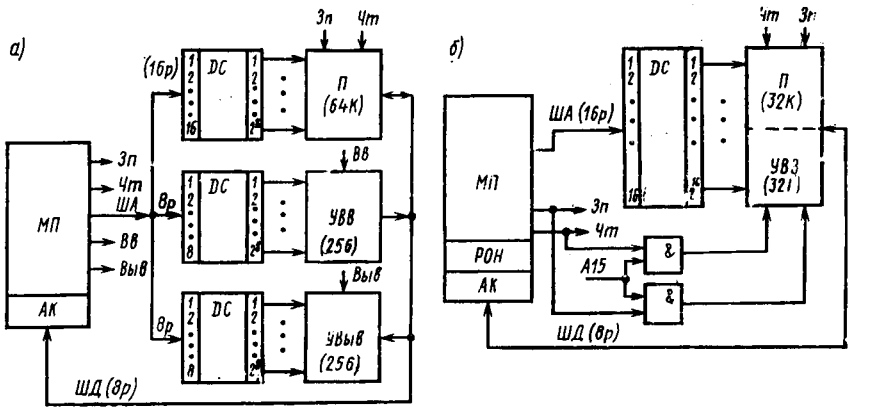


Рис. 5.3. Организация интерфейса МПС с памятью и УВВ:

Организация связи МП с ЗУ и УВВ (рис. 5.2) требует довольно большого числа внешних выводов БИС (от 70 до 120 в зависимости от разрядности), многошинной организации его структуры. Сокращения числа внешних выводов, задействованных под интерфейс МП, обычно добиваются мультиплексированием шин: *временным* — с использованием одной и той же шины для передачи функционально отличной информации полным словом или по частям в разные моменты времени; *пространственным* — с использованием одной и той же шины для передачи в обоих направлениях; *пространственно-временным* — при сочетании первых двух методов и использовании дополнительных интерфейсных средств. Для сопряжения МП с ЗУ и УВВ, как правило, используются совмещенный или раздельный способы организации интерфейса (рис. 5.3), имеющие свои достоинства и недостатки [9].

Внутренний интерфейс подсистемы памяти МПСУ на МПК БИС обеспечивает: электрическую совместимость шин МП и БИС ЗУ; выработку адресов ячеек памяти; взаимодействие с управляющими сигналами МП и синхронизацию процедур записи и считывания данных с операциями на шинах. Особенности структурной и функциональной организации интерфейсов памяти зависят от специфики организации подсистем памяти, типов используемых БИС ЗУ, параметров системной шины и др. Общепринятым считается интер-

фейс памяти на основе БИС ЗУ с раздельными ША и ШД и управляющими сигналами «ВК» и «Чт/Зп». Подключение же такой памяти к МП с совмещенной ША и ШД требует применения дополнительной аппаратуры.

Появление МП с повышенной разрядностью (16, 32 бит) потребовало переориентации применения типовых модулей памяти, созданных ранее в расчете на 8-разрядные МПСУ, и построения стандартизированных интерфейсов памяти, логически перестраиваемых для работы в МПСУ с различной разрядностью МП.

Внутренние интерфейсы БИС микропроцессорных функциональных модулей (МФМ) ввода — вывода обладают большим разнообразием и приближаются по своим функциональным возможностям к интерфейсу МП. Основной особенностью внутриплатного интерфейса БИС УВВ является наличие малоразрядной (2—4 бит) ША, предназначенной для адресации внутренних регистров (иногда эта ША отсутствует).

Кроме ША и ШД для сопряжения с МП в БИС УВВ предусматриваются входы — «ВК» (1—3 бит), «Чт/Зп», «Разрешение ПДП», «Начальная установка», «Сброс»; выходы — сигналы о готовности к обмену, запросы прерывания (1—2 бит), запрос на ПДП и др.

Обмен информацией между МП и УВВ может производиться в режимах: программно-управляемой передачи данных (по инициативе и под управлением МП); передачи данных в режиме прерывания (по инициативе УВВ и под управлением МП) или в режиме ПДП (по инициативе УВВ и под управлением либо самого УВВ, либо специального контроллера ПДП).

Реализация режима ПДП с использованием специализированного контроллера позволяет все интерфейсные функции БИС УВВ сводить в основном к посылке запроса на ПДП, осуществлять процесс адресации ОЗУ и передачи данных. Если в МФМ УВВ контроллер ПДП входит в состав межплатного интерфейса данного модуля, то тогда в МФМ УВВ должны быть: двунаправленные ША и ШД, двунаправленные линии «Чт/Зп», выходные линии запроса ПДП и подтверждения ПДП, входная линия разрешения ПДП. Это обеспечивает непосредственную адресацию ячеек ОЗУ (или другого МФМ УВВ) и управление передачей данных в режиме ПДП. При необходимости работы нескольких УВВ в режимах ПДП или прерывания требуется организация приоритетного обслуживания запросов на ПДП или запросов на прерывания, реализация которых существенно влияет на основные характеристики интерфейсной системы.

Упрощения подключения дополнительных устройств к уже разработанным МПСУ, сопряжения в единую систему отдельных МП добиваются стандартизацией микропроцессорных системных шин, а также решений по организации внутри- и межплатных интерфейсных систем. К подобным зарубежным проектам системных шин МПСУ, например, относятся проекты Versa-bus, Futurebus, IEEE

Р-796 (Multibus), AMS-bus, Eurobus, VME-bus и др. Стандартизированная внутриплатная интерфейсная шина Microbus, например, ориентируется на создание 8-разрядных МПС, использующих раздельные параллельные или совмещенные шины ША и ШД. Увеличение разрядности ШД до 16 бит и ША до 24—32 бит требует применения либо дорогих 64-контактных корпусов БИС, либо осуществления операции мультиплексирования адреса — данных (А/Д), как это принято, например, в проекте MOTEL. Такое решение упрощает сопряжение МП с совмещенной шиной А/Д, стимулируя разработку совместимых с ними БИС ЗУ и УВВ. К стандартам на шины межплатных интерфейсов относятся, например, интерфейсы S-100, Q-bus, Multibus, STD-bus и др. Основные характеристики некоторых из интерфейсов МПСУ приведены в табл. 5.1.

Взаимодействие МЭВМ с внешней средой осуществляется посредством внешних интерфейсов, реализуемых в виде контроллеров внешних устройств (КВУ) на базе БИС (или МФМ) УВВ и подключающихся либо к периферийным устройствам, либо к объекту управления, а также к линиям связи и сигнальным линиям шин других типов интерфейсов (мультимикропроцессорных систем, сетей ЭВМ и пр.).

Существует множество подходов к проектированию КВУ, начиная от использования простейших цифровых входов — выходов общего назначения с реализацией функций интерфейса программными средствами МП и кончая специализированными машинно-ориентированными КВУ с аппаратной реализацией всех интерфейсных функций. В последнее время КВУ широко реализуются на базе однокристальных МЭВМ и микроконтроллеров, обладающих ограниченным числом команд с ориентацией их на управление вводом — выводом, совместимым интерфейсом с МЭВМ более высокого уровня и внутренней памятью (ОЗУ и ПЗУ) небольшого объема, достаточного для формирования требуемых интерфейсных функций по сопряжению со стандартными УВВ.

Функциональная организация интерфейсов определяется их назначением, информацией, характеризующей данные и алгоритмы взаимодействия элементов интерфейса (типы и формы представления данных, перечень интерфейсных операций, адресов, команд, данных о состоянии), а также классом алгоритмов и технико-экономическими требованиями. В целом она характеризует способы построения и порядок взаимодействия интерфейсных блоков.

Алгоритмы взаимодействия элементов интерфейса подразделяются на классы алгоритмов: а) установления и ликвидации информационного канала связи; б) обмена информацией; в) преобразования информации; г) сервисные алгоритмы. Реализуются эти алгоритмы с помощью набора различных интерфейсных операций (табл. 5.2). Наиболее распространенные из них функции приема — передачи информации и ее синхронизации, инициирования запроса и приема разрешения на прерывание или захват магистрали,

Таблица 5.1. **Основные характеристики интерфейсов МПС**

Характеристики	Типы интерфейсов												
	Cuber-bus	Microbus	Eurobus	Modus	Multibus P 796	Q-bus	STD-bus P 961	S-100 P 696	Z-bus	Versa-bus	P-689	VME-bus	Futurebus
Число контактов разъема	72	74	64	—	86	72	56	100	96	132	96	—	—
Разрядность шины адреса	18, М	16	18, М	16, М	16	16, М	16	16	32, М	32	32, М	16, 24, 32	32
Разрядность шины данных	16, М	16	18, М	16, М	16	16, М	8	16	32, М	32	32, М	8, 16, 32	8, 16, 32
Число линий:													
управляющих	19	20	7	16	15	16	22	35	28	18	—	—	—
вспомогательных	13	2	9		8	2			7	9	—	—	—
зарезервированных		2			7	13			2	8	—	—	—
факультативных			9			8		25		10	6	—	—
питания	10	10	16	2	24	16	10	6	21	20	—	—	—
Управление магистралью:													
централизованное	Да	Нет	Да	Да	Да	Да		Да	Нет	Да	—	—	—
приоритетная цепочка	Да	Да	Нет		Да	Да	Да	Нет	Да	Да	—	—	—
параллельное с опросом	Нет	Да	Нет	Да	Да	Нет		Нет	Нет	Нет	—	—	—
последовательное с опросом	Нет	Нет	Нет		Нет	Нет		Нет	Да	Нет	—	—	—
выделенные линии	Да	Нет	Да		Нет	Нет		Да	Нет	Да	—	—	—

Продолжение табл. 5.1

Характеристики	Типы интерфейсов												
	Cuber-bus	Microbus	Eurobus	Modus	Multibus P 796	Q-bus	STD-bus P 961	S-100 P 696	Z-bus	Versa-bus	P-689	VME-bus	Futurebus
Управление прерываниями:													
централизованное	Да	—	Да	—	Нет	Да	Нет	Да	Нет	Да	—	—	—
приоритетная цепочка	Да	—	Нет	—	Нет	Да	Нет	—	Да	Да	—	—	—
параллельное с опросом	Нет	—	Нет	Да	Нет	Нет	Нет	—	Нет	Нет	—	—	—
последовательное с опросом	Нет	—	Нет	Да	Нет	Нет	Нет	—	Да	Нет	—	—	—
выделенные линии	Да	—	Нет		Да	Нет	Нет	Да	Нет	Да	—	—	—
Реакция на запрос магистрали	—	—	—	400	100	A	—	—	50	30	—	—	—
Реакция на прерывание, нс	—	—	—	800	50	A	—	—	50		—	—	—
Число адресуемых портов ввода — вывода	2^{16}	—	—	50	256	4096	—	256	—	2^{32}	2^{32}	—	—
Мультипроцессорный режим	Да	Да	Да	Да	Да	Да	Да	Да	Да	Да	Да	—	—

Примечание. В таблице обозначено: М — мультиплексированный режим; А — асинхронный режим,

Таблица 5.2. **Перечень основных интерфейсных функций, выполняемых в МПСУ**

Класс реализуемых алгоритмов взаимодействия	Интерфейсные операции f_i (i=1, 17)																	
	Прерывание	захват шины	адресация	синхронизация обмена	запись слова	запись блока	чтение слова	чтение блока	чтение вектора	чтение — модификация — запись	преобразование формата	перекодирование	принудительное освобождение шины	контроль данных	контроль функционирования	системный сброс	контроль (авария) питания
Установление и ликвидация связи на информационном канале	+	+	+	+	+	+	+	+	+	+			+		+		
Обмен информацией				+	+	+	+	+	+	+			+		+		
Преобразование информации											+	+		+	+		
Сервисные алгоритмы													+	+	+	+	+

Примечание. Знаком «+» обозначено вхождение интерфейсных операций f_i в классы реализуемых алгоритмов.

приема запросов, выделения и идентификации приоритетного запроса, передачи разрешения на захват магистрали или на прерывание, общего управления интерфейсом. Все эти и другие интерфейсные функции нашли, например, широкое отражение в интерфейсах МЭК 625—1 (ГОСТ 26.003—80), КАМАК и др.

Независимо от назначения и области применения интерфейсы характеризуются рядом технических показателей. Наиболее важные из них — вместимость, пропускная способность, максимальная длина линий связи, надежность и стоимость.

Вместимость определяет максимально возможное число устройств, которое может быть подключено к интерфейсу без использования дополнительных средств его расширения. Вместимость зависит от принятой системы адресации, нагрузочной способности приемопередающих элементов и особенностей выполнения линий связи.

Пропускная способность определяет время передачи единицы информации между устройствами интерфейса. Она определяется сложностью алгоритмов взаимодействия (временем установления связи и ликвидации информационного канала, процесса обмена информацией), набором и быстродействием исполнения интерфейсных операций, реализующих эти алгоритмы, разрядностью информационного канала, степенью совмещения во времени процессов информационного взаимодействия. В целом пропускная способность интерфейсной системы γ зависит от пропускной способности между парой устройств γ':

$$\gamma' = \frac{K\Phi_д}{T_у + K\dfrac{\Phi_л}{r} T_ц + T_л},$$

где K — коэффициент учета режима передачи ($K=1$ для мультиплексного режима передачи; $K=n$ для селекторного (монопольного) режима обмена информацией, n — средняя длина передаваемого массива слов данных); $\Phi_д$ — формат слова передаваемой информации; r — разрядность информационного канала, $T_у$ ($T_л$) — время установления (ликвидации) связи по информационному каналу; $T_ц$ — время цикла передачи одного r-разрядного слова информации.

Максимальная длина линий связи определяет максимально возможную удаленность приемопередающих элементов информации, зависящую от их электрических характеристик, способа выполнения линий связи, метода синхронизации обмена информацией.

Надежность интерфейса зависит от типа структурной организации связей, разрядности и длины информационных каналов, показателей надежности составных элементов и может быть рассчитана с помощью приближенных методов.

Стоимость интерфейса определяется расходами на его разработку, изготовление, внедрение и эксплуатацию. Приблизительно стои-

мость интерфейса оценивается затратами на контроллеры, интерфейсные блоки, источники питания, конструктивы и линии связи. Затраты на источники питания и конструктивы зависят от вместимости интерфейса, его топологии, конструктивных особенностей выполнения интерфейсных элементов. Затраты на линии связи определяются в основном суммарной длиной связи, причем для распределенных систем эти затраты нередко соизмеримы со стоимостью оборудования. Наиболее экономичны структуры с магистральной и последовательной (без обратной связи) топологией.

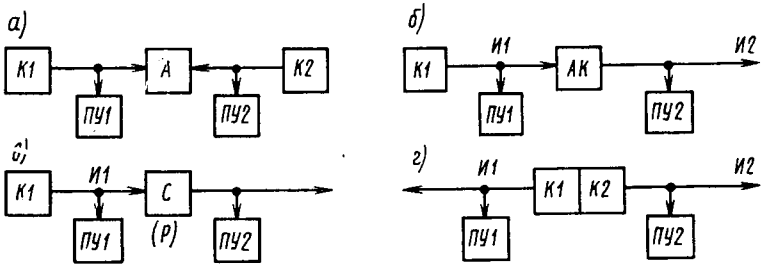


Рис. 5.4. Способы сопряжения объединяемых интерфейсов:

а — с взаимным соподчинением; *б*, *в* — с иерархическим подчинением и косвенным методом доступа; *г* — с централизованным подчинением и прямым методом доступа; И1, И2 — первый и второй интерфейсы; К1, К2 — контроллеры; ПУ1, ПУ2 — периферийные устройства; А — адаптер, АК — адаптер-контроллер; С(Р) — согласователь (расширитель)

В целом описание функциональной и структурной организации интерфейса в виде наборов интерфейсных функций, а также структур и связей позволяет осуществлять синтез интерфейсных элементов с учетом специфики используемой элементной базы, технических требований и ограничений.

Структурная организация интерфейса зависит от выбора номенклатуры и типа составляющих его элементов, а также от организации связей между ними. Различают следующие типы интефейсных элементов: а) *интерфейсная карта* — при использовании в качестве операционных блоков функционально законченных пассивных устройств (ОЗУ, ПЗУ, перфораторов и др.); б) *интерфейсный контроллер* — при использовании в качестве операционных блоков функционально законченных активных устройств (процессор, устройство с режимом ПДП), требующих управления элементами интерфейса; в) *интерфейсный адаптер* — при использовании в качестве операционного блока конкретного интерфейса другого интерфейса, одинакового ранга с первым; г) *интерфейсный адаптер-контроллер* — при использовании в качестве операционного блока другого интерфейса, но более низкого уровня иерархии по сравнению с первым.

Для пояснения сказанного на рис. 5.4 приведены возможные способы обеспечения совместимости двух сопрягаемых интерфейсов

и соответствующие им технические средства. Различают три способа управления объединяемых интерфейсов: со взаимным соподчинением (рис. 5.4, *а*), с иерархическим подчинением (рис 5.4, *б*, *в*) и с централизованным подчинением (рис. 5.4, *г*).

При реализации способа взаимного соподчинения техническим средством объединения двух однотипных независимых интерфейсов является адаптер, выполняющий функции двустороннего коммутатора с обеспечением попеременного взаимного доступа устройств одной системы сопряжения к ресурсам другой системы.

При реализации способа иерархического подчинения предполагается подключение ведомого интерфейса к ведущему через контроллер ведомого. Так как для ведущего интерфейса контроллер ведомого интерфейса является устройством ввода — вывода, то его называют контроллером или *адаптером-контроллером (АК)*. Такой способ обеспечения совместимости, например, наиболее часто применяется при сопряжении системных интерфейсов «Общая шина», И-41 с интерфейсами периферийного оборудования типов КАМАК, 625-1 и др.

Контроллер в такой системе выполняет функции взаимодействия с интерфейсом верхнего уровня (прием и интерпретация команд, преобразование адресов и данных, выделение и обслуживание наиболее приоритетных запросов от подчиненных ему устройств и др.).

При реализации способа иерархического подчинения для однотипных и частично совместимых машинных интерфейсов применяют и непосредственный доступ к ведущему интерфейсу со стороны ведомого. Устройства, обеспечивающие непосредственный доступ к совмещаемым интерфейсам, называют *согласователями* или *расширителями интерфейсов*. Примером могут служить расширители интерфейсов 2К, «Общая шина» и др.

При реализации способа централизованного подчинения, являющегося универсальным и гибким в применении, интерфейсный контроллер обеспечивает выход на разнотипные системы сопряжения. Он, как правило, имеет буферную память и выполняет функции логического и физического управления интерфейсами.

§ 5.3. ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ ИНТЕРФЕЙСОВ ДЛЯ МИКРОПРОЦЕССОРНЫХ СИСТЕМ УПРАВЛЕНИЯ

Проектирование интерфейсов МПСУ ведется, как правило, на базе новейших МПК БИС или ОМЭВМ с выполнением ряда этапов, основными из которых являются этапы внешнего, операционного и логического синтеза.

На этапе *внешнего проектирования* определяется структурная и функциональная организация интерфейса, уточняются алгоритмы взаимодействия интерфейсных элементов, состав и назначение сигнальных линий интерфейса, выбираются критерии для оценки эф-

фективности интерфейсов и элементной базы для их реализации. На этапе *операционного* проектирования разрабатываются алгоритмы выполнения интерфейсных операций, уточняются форматы и способы кодирования данных, управляющей и адресной информации, информации о состоянии, структуры интерфейсных элементов на уровне операционных схем. На этапе *логического* проектирования разрабатываются структура и порядок функционирования интерфейсных элементов на уровне логических схем и электрических принципиальных схем.

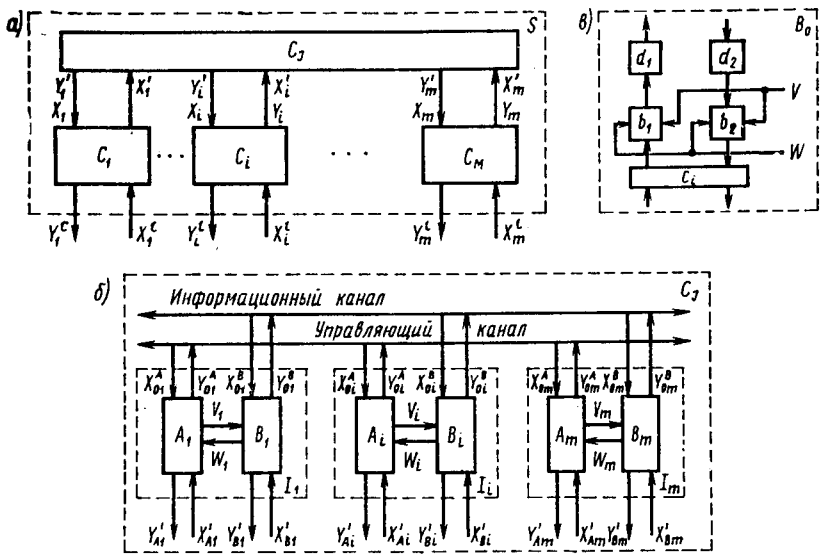


Рис. 5.5. Структурная организация интерфейса МПСУ:
а — общий вид структуры интерфейса; *б* — состав интерфейсной системы; *в* — структура операционного автомата; C_i — операционные (согласующие) блоки; C_J — интерфейсная система; B_i, A_i — операционный и управляющий автоматы блока C_J

Так как интерфейсы существенно влияют на основные технико-экономические показатели МПСУ, то при их выборе и разработке необходимо применять формализованные методы исследования и описания. Наиболее известные средства представления интерфейсов: 1) автоматные модели; 2) языки программирования; 3) комбинированные модели с использованием автоматных моделей и языков программирования [48].

В наиболее перспективных комбинированных моделях функциональная организация и процессы взаимодействия интерфейса описываются на языках программирования, а схемотехническая структура его составных элементов представляется моделью конечного автомата (рис. 5.5).

Более детально модель интерфейса можно представить композицией интерфейсных блоков I_i, состоящих из соответствующих управляющего A_i и операционного B_i автоматов ($i=\overline{1, m}$), описывающих соответственно управляющий и информационный каналы интерфейса, и согласующих блоков C_i, обеспечивающих связь автоматов A_i, B_i с сопрягаемым функциональным устройством (ФУ) и преобразующих унифицированные наборы информационных $\{X'_{B_i}, Y'_{B_i}$ и управляющих $\{X'_{A_i}, Y'_{A_i}\}$ сигналов с автоматов B_i и A_i в наборы $\{X_{B_i}, Y_{B_i}\}$ и $\{X_{A_i}, Y_{A_i}\}$ сигналов с сопрягаемого ФУ.

Управляющий автомат A_i координирует функционирование операционного автомата B_i посредством обмена набором управляющих сигналов v и логических условий w. Автомат A состоит из набора операционных элементов (регистров, дешифраторов, шифраторов, коммутаторов) и связей между ними. Автомат B выполняет функции хранения, преобразования и коммутации информации между функциональными устройствами. Основу элементов B_0, входящих в автомат B, составляют входной $X_0=\{b_2, d_2\}$ и выходной $Y_0=\{b_1, d_1\}$ каналы (рис. 5.5, *в*), где b_2, b_1 — входной и выходной наборы операционных элементов, d_2, d_1 — входной и выходной преобразователи электрических сигналов. Функции преобразователей b_2, b_1 заключаются в выполнении отображений α и β соответственно, т. е.

$\alpha: X_0 \to Y_0'$; $\beta: X_0' \to Y_0$. Функции преобразователей d_1, d_2 состоят в обеспечении электрической совместимости.

В целом блок I_i представляет собой унифицированную часть модуля с функциональной организацией, не зависящей от особенностей сопрягаемого функционального устройства и определяемой набором интерфейсных функций. Основная функция согласующего блока C_i состоит в обеспечении информационной и электрической совместимости блока I_i с конкретным ФУ. Если рассматриваемый модуль является, например, интерфейсным адаптером, то блок C_i превращается в интерфейсный блок $I'=\{A', B'\}$. Базовый интерфейсный блок I служит основой создания контроллера и набора интерфейсных блоков, пригодного для широкого класса различных функциональных устройств МПСУ. Это достигается путем введения определенной избыточности и возможности перестройки или перепрограммирования базового интерфейсного блока.

Проектирование интерфейсного блока I включает в себя этапы: определения полного набора интерфейсных функций; построения операционного автомата B и описания набора сигналов $\{w, v\}$; составления алгоритма функционирования; синтеза функциональной схемы управляющего автомата A и выбора средств его реализации; определения условий совместимости блоков I и C_i (набора связей и временных соотношений между сигналами). Блок C_i разрабатывается в соответствии с условиями, определяемыми набором

связей от блока I, и сигналами от i-го функционального устройства.

Особенно перспективно для реализации интерфейсных модулей применять МП и ОМЭВМ, например К1820, К1814 и др., причем наиболее типовыми функциями при этом являются перекодирование, контроль, преобразование формата данных и др.

Типовые задачи реализации интерфейсов на микропроцессорных средствах могут быть сформулированы следующим образом.

1. Известны назначение МПСУ, ее обобщенный критерий оптимизации $F=F(T)$, вектор основных технических характеристик $T=(\beta, \gamma, L_{max}, H, C)$, определен набор функциональных элементов для ее реализации. Требуется синтезировать оптимальный интерфейс, отвечающий заданному критерию эффективности функционирования МПСУ, т. е.

$$F=\text{extr}\,F(\beta, \gamma, L_{max}, H, C)$$

при $\beta \geqslant \beta_0$, $\gamma \geqslant \gamma_0$, $L_{max} \geqslant L'_{max}$, $H \geqslant H_0$, $C \leqslant C_0$, где β_0, γ_0, L'_{max}, H_0, C_0 — допустимые значения основных технических характеристик интерфейса (вместимости, пропускной способности, максимальной длины линии связи, надежности и стоимости соответственно).

Так как решение данной задачи в общем виде получить сложно, то на практике обычно проводят целенаправленный выбор наиболее рациональных вариантов функциональной и структурной организации интерфейса.

2. Имеется успешно функционирующая МПСУ с типовым интерфейсом, известными функциональной и структурной организацией и техническими характеристиками. Требуется к этой МПСУ подключить еще одно или несколько функциональных устройств с известными характеристиками, не имеющих средств сопряжения с данным интерфейсом.

Решение данной задачи сводится к разработке или выбору интерфейсного элемента, обеспечивающего подключение дополнительного функционального устройства к имеющейся МПСУ без нарушения ее работоспособности. При этом производится: а) уточнение требуемых характеристик интерфейсного элемента, набора интерфейсных функций и способа их реализации; б) выбор приемопередающих элементов с требуемыми электрическими и временны́ми характеристиками; в) реализация интерфейсного элемента в заданном конструктиве с последующей проверкой работоспособности всей модифицированной системы и оценкой эффективности исполнения интерфейсного элемента.

3. Известны назначение и ориентировочная область применения интерфейса. Требуется определить его функциональные и структурные характеристики, условия электрической и конструктивной совместимости и разработать структуру интерфейсных элементов, которая удовлетворяет заданному вектору основных технических характеристик $T=(\beta, \gamma, L_{max}, H, C)$ или оптимизирует одну или несколько из этих характеристик при фиксированных значениях

остальных показателей. При этом часто указывают дополнительные требования по достижению определенных функциональных возможностей интерфейса, его инвариантности к элементному базису и др.

Данная задача также плохо формализуема и требует применения процедур выбора предпочтительного варианта из множества допустимых при сравнительном анализе существующих интерфейсов аналогичного класса.

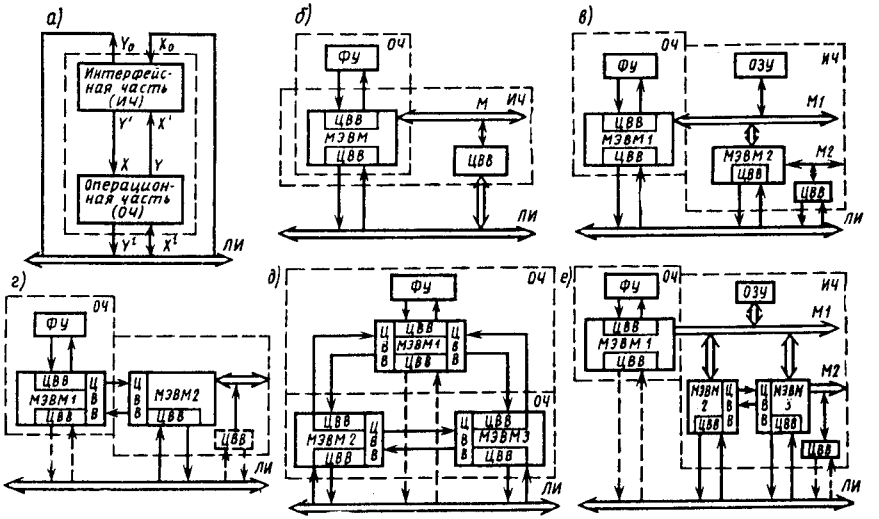


Рис. 5.6. Представление сопрягаемого с МПСУ функционального устройства (ФУ) в виде операционной (ОЧ) и интерфейсной части (ИЧ) и типовые конфигурации контроллера на базе ОМЭВМ:

а — структура ФУ; *б*—*е* — варианты ТКК, реализуемых на основе ОМЭВМ; ЛИ — линии интерфейса; М — магистраль; ЦВВ — цифровые входы — выходы

Появление дешевых одноплатных и однокристальных МЭВМ с развитой структурой программируемого ввода — вывода позволяет эффективно применять их для синтеза интерфейсных контроллеров, реализующих сопряжение разнообразных МПСУ с конкретными типами ФУ. Укрупненный алгоритм синтеза интерфейсных контроллеров на базе ОМЭВМ, в частности ОМЭВМ типа «Электроника С5», включает в себя:

1. Функциональное устройство, подключаемое к МПСУ и представляемое в виде (рис. 5.6, *а*): операционной части (ОЧ), определяющей конкретное функциональное назначение устройства, и интерфейсной части (ИЧ), обеспечивающей его сопряжение с заданной МПСУ. Определить для них требуемые временные характерис-

тики (среднее время пребывания заявки и ожидания обслуживания) исходя из методов теории массового обслуживания.

2. В зависимости от типа подключаемого устройства (активное или пассивное) сформировать подмножество интерфейсных функций Φ_j, подлежащих реализации в ИЧ ФУ, и составить для каждой из них диаграмму состояний.

3. Провести анализ диаграмм состояний по каждой из интерфейсных функций Φ_j и установить программно-аппаратные соглашения: при невозможности программной реализации всей или части диаграммы состояний каждой из интерфейсных функций Φ_j из-за недостаточного быстродействия ОМЭВМ часть Φ_j ($\Phi_j = \Phi_j^{\text{А}} \vee \vee \Phi_j^{\text{П}} \vee \Phi_j^{\text{ПА}}$) реализовать чисто аппаратным $\Phi_j^{\text{А}}$ или программно-аппаратным путем $\Phi_j^{\text{ПА}}$ с введением некоторых дополнительных ресурсов программируемого ввода — вывода и памяти МЭВМ.

4. Определить ресурсы (R) ОМЭВМ, требующиеся для программной реализации подмножества интерфейсных функций:

$$R = (N_{\text{цвв}},\ N_{\text{пр}},\ M,\ T),$$

где $N_{\text{цвв}}$ — количество требуемых цифровых входов — выходов (ЦВВ); $N_{\text{пр}}$ — количество требуемых входов прерывания; M — требуемый объем памяти; T — требуемое быстродействие МЭВМ.

Требуемое число ЦВВ определяется выражением $N_{\text{цвв}} = |N_{\text{и}} \vee \vee N_{\text{о}} \vee N_{\text{в}} \vee N_{\text{л}}|$, где $N_{\text{и}}$, $N_{\text{о}}$, $N_{\text{в}}$, $N_{\text{л}}$ — количество необходимых связей с сигнальными линиями магистрали интерфейса, операционной частью устройства, между интерфейсными функциями, с внешней дополнительной логикой соответственно.

Реализация интерфейса ФУ с заданной МПСУ производится с помощью ряда типовых конфигураций контроллеров (ТКК), при этом:

а) если ОМЭВМ может одновременно управлять ОЧ и ИЧ функционального устройства и все подмножество интерфейсных функций Φ_j может быть реализовано программным путем ($\Phi_j = \Phi_j^{\text{П}}$) с помощью ресурсов одной ОМЭВМ, то используется ТКК (рис. 5.6, *б*). Здесь точки сопряжения ОЧ и ИЧ, а также связи между Φ_j могут быть выделены лишь на программном уровне, т. е. $N_{\text{о}} = N_{\text{в}} = N_{\text{л}} = \varnothing$;

б) если реализация подмножества интерфейсных функций $\Phi_j = \Phi_j^{\text{П}} \vee \Phi_j^{\text{ПА}}$ возлагается на дополнительную ОМЭВМ2, то можно использовать две разновидности типовых конфигураций контроллеров. В первой ТКК (рис 5.6. *в*) связь с ОЧ устройства осуществляется через общее ОЗУ по магистральному интерфейсу ОМЭВМ на микропрограммно-программном уровне. Прием и выдача информации от ФУ производится с помощью либо ОМЭВМ2, либо ОМЭВМ1 под управлением ОМЭВМ2.

Во второй ТКК (рис. 5.6, *г*) связь с ОЧ ФУ осуществляется посредством ЦВВ ОМЭВМ1 и ОМЭВМ2.

Прием и выдачу информации от функционального устройства предпочтительнее возложить на ОМЭВМ1 под управлением ОМЭВМ2 из-за уменьшения N_0;

в) если же все подмножество интерфейсных Φ_j функций не удается реализовать на двух ОМЭВМ по каким-либо причинам (нецелесообразности или невозможности совмещения алгоритмов выполнения интерфейсных функций, снижения быстродействия до недопустимых пределов и др.), то применяют ТКК двух разновидностей, аналогичные вышерассмотренным, но из трех ОМЭВМ (рис. 5.6, *д*, *е*). В этих ТКК программа реализации интерфейсных функций возлагается на ОМЭВМ2 и ОМЭВМ3, прием и выдача информации осуществляются ОМЭВМ1 под управлением ОМЭВМ2 и (или) ОМЭВМ3, а связь ИЧ и ОЧ функционального устройства производится на микропрограммно-программном уровне через общее ОЗУ по магистральному интерфейсу или с помощью ЦВВ ОМЭВМ2 и ОМЭВМ3.

При недостатке быстродействия используемых ОМЭВМ для программной реализации Φ_j в состав ТКК (рис. 5.6, *б*—*е*) вводится аппаратный блок, связанный в общем случае с интерфейсными ОМЭВМ, операционной ОМЭВМ и линиями интерфейса.

Количество требуемых входов прерывания $N_{\text{пр}}$ определяется после окончательного выбора ТКК, реализующей $\Phi_j^{\text{ПА}}$, составления блок-схемы программы и тщательного распределения приоритетов по каждой из причин прерывания. Ориентировочно считают, что $N_{\text{пр}} \geqslant m$, где m — число сигналов, требующих немедленной реакции.

Требуемый объем памяти M можно приблизительно оценить на этапе схемно-программных соглашений согласно соотношению

$$M \geqslant (3 \div 4) N L_{\text{ср}} / \Phi_{\text{я}},$$

где N — число условных и операторных вершин в граф-схеме алгоритма выполнения диаграмм состояний интерфейсных функций $\Phi_j^{\text{ПА}}$; $L_{\text{ср}}$ — средняя длина в битах используемых в программе команд, работающих преимущественно с программируемым вводом — выводом МЭВМ; $\Phi_{\text{я}}$ — формат ячейки памяти, бит.

Окончательно оценить величину M можно после написания программы реализации $\Phi_j^{\text{ПА}}$ с учетом выбранной ТКК конфигурации контроллера.

Требуемое быстродействие T МЭВМ определяется как ее пригодностью для программной реализации каждой интерфейсной функции Φ_j с временем $T^{\text{П}}{}_{i\max} \leqslant t_i$ ($T^{\text{П}}{}_{i\max}$ — максимальное время реализации i-го перехода Φ_j), так и значением среднего времени пребывания заявки $t_{\text{и}}$ в ИЧ ФУ ($t_{\text{и}} \leqslant t^* - t_0$, где $t^*(t_0)$ — допустимое (среднее) время пребывания заявки в ФУ). При невыполнении заданных ограничений производится выбор новой, более быстродействующей ОМЭВМ или отказ от программной реализации Φ_j с переходом к аппаратным средствам.

§ 5.4. ОПТИМИЗАЦИЯ И РАСПРЕДЕЛЕНИЕ РЕСУРСОВ МПСУ

При решении многих задач, например обработки изображений, преобразования и упорядочения структур данных, логического и функционального программирования, управления базами данных, создания специализированных машин и др., возникает необходимость оптимизации ресурсов памяти и производительности МПСУ вследствие сложности программной реализации алгоритмов подобных задач, требующих довольно большой памяти и затрат времени. Оптимизация ограниченных ресурсов памяти и производительности МПСУ, реализуемых на базе серийных МЭВМ, предусматривает установление соответствия структур данных структурам памяти и наоборот с целью размещения в заданном объеме памяти максимально возможного объема информации.

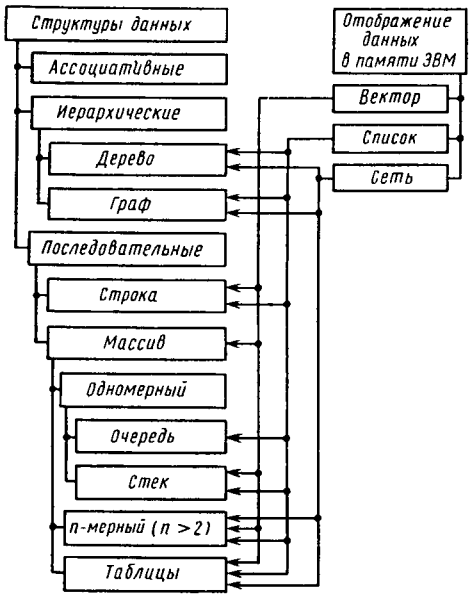


Рис. 5.7. Структуры данных и их отображение в памяти ЭВМ

Рассмотрим вопросы оптимального вложения структур данных в структуры памяти МЭВМ, выбора соответствующих методов управления памятью и использования ресурсов МЭВМ с точки зрения быстродействия и стоимости решения задач.

Структуры данных и их отображение в памяти МЭВМ. Известно, что структуры данных имеют основополагающее значение в программировании и их правильный выбор существенно влияет на качество ПО МПСУ, на эффективность доступа к данным и на величину требуемого для их отображения объема памяти (рис. 5.7).

В общем случае МПСУ осуществляют обработку информации о поведении объектов управления в соответствии с заданным алгоритмом функционирования. Совокупность информации, описывающую некоторое свойство конкретного объекта, называют *логической записью* или просто *записью,* а всю совокупность подобных записей — *информационным массивом.* Организация информационного массива, обеспечивающая определенные связи и отношения между данными, называется *структурой данных.*

Различают три уровня представления данных в МЭВМ: логический уровень, уровень хранения и физический уровень.

На *логическом уровне* работают с *логическими структурами данных*, отражающими реальные отношения между объектами и их характеристиками и указывающими, в каком виде данные представляются пользователю системы. На этом уровне устанавливается перечень *признаков*, полностью характеризующих описываемый класс объектов. Совокупность признаков и их взаимосвязь определяют *внутреннюю структуру логической записи*. Вместе с тем на логическом уровне представления данных не учитываются особенности технического и программного обеспечения МПСУ (тип МЭВМ, тип ЗУ, язык программирования и др.).

На *уровне хранения* оперируют со *структурами хранения* — представлениями логической структуры данных в памяти МЭВМ. При разработке или выборе структуры хранения учитываются тип ОЗУ, возможности применяемого языка программирования, устанавливаются тип и формат данных, определяются способ поддержания логической структуры и объем памяти, необходимый для размещения данных. Одна и та же логическая структура данных может быть реализована в ОЗУ различными структурами хранения, каждая из которых представляет определенный способ доступа к данным и манипулирования ими. От правильности выбора структуры хранения зависит эффективность обработки данных (быстрый поиск нужных данных, возможность обновления записей и их корректировки без разрушения логической структуры, минимальный расход памяти МЭВМ). Поддержание структур хранения осуществляется программными средствами, поэтому при их разработке следует учитывать возможности языка программирования, на котором составляются программы работы с данными.

На *физическом уровне представления данных* происходит оперирование с *физическими структурами данных*, осуществляется реализация структур хранения непосредственно в памяти конкретной МЭВМ с учетом типа и объема памяти, способа адресации, метода и времени доступа к данным, способа обмена данными и др.

В целом структуры данных делятся на *линейные* (массив, стек, очередь, таблица) и *нелинейные* (деревья, графы, многосвязные списки и списковые структуры). В нелинейных структурах в отличие от линейных связь между элементами структуры (записями) определяется отношениями подчинения или какими-либо логическими условиями. Ряд структур данных имеет фиксированный размер и после их создания не допускает обновления (включения или исключения) записей, а допускает лишь их корректировку. Другие структуры данных имеют *переменный размер*, позволяют обновлять записи с возможностью динамического изменения информационного массива. Различные структуры данных обеспечивают и разную дисциплину доступа к своим элементам: в одних структурах доступ возможен к любому элементу, в других — к строго определенному. Ограничение в доступе сопровождается увеличением времени поиска нужных элементов.

В памяти МЭВМ данные могут иметь последовательное или связанное представление. При *последовательном представлении* данные в памяти размещаются в соседних последовательно расположенных ячейках, при этом логическая структура поддерживается физическим порядком следования данных. Совокупность записей, размещенных в последовательно расположенных ячейках ОЗУ, называют *последовательным списком*. В этом случае для хранения ин-

формационного массива выделяется объем памяти, соответствующий максимальному размеру массива, с последовательным логическим порядком записей и размещении вновь появляющихся записей в конце блока на свободном участке памяти. Если число новых записей оказывается больше свободного числа зарезервированных ячеек, то их не удается разместить в памяти, а если меньше, то память остается недоиспользованной. В целом последовательное представление данных обычно используют при реализации линейных структур данных и возможности вычисления предельного размера информационного массива.

При работе с данными, которые непрерывно обновляются и корректируются, представление данных в виде последовательного списка ведет к неэффективному использованию памяти, потере машинного времени на перезапись массива.

При *связанном представлении данных* в каждой записи предусматривается дополнительное поле, в котором размещается *указатель* (ссылка). Физический порядок следования записей в этом случае может не соответствовать логическому порядку. В машинной памяти записи располагаются в любых свободных ячейках ОЗУ и связываются между собой указателями, указывающими на место расположения записи, логически следующей за данной записью. Поэтому указатель часто интерпретируют как адрес ячейки памяти, в которой хранится следующая запись.

Структуры хранения, базирующиеся на связанном представлении данных, называют *связанными списками.* Если каждая запись содержит лишь один указатель, то список называют *односвязным,* если несколько указателей, то *многосвязным.*

Связанное представление данных в памяти МЭВМ используется для хранения нелинейных структур данных, а также для линейных структур при невозможности вычисления предельного размера информационного массива, а следовательно, и требуемого объема памяти. В целом при связанном представлении требуется дополнительный расход памяти под размещение указателей.

При необходимости продвижения по связанному списку в обоих направлениях в каждый элемент списка вводят дополнительный указатель, задающий продвижение по списку в обратном направлении. Такой список называется *двунаправленным.* В поле указателя заносится адрес ячейки с записью, логически предшествующей данной записи. Головная ячейка памяти содержит в этом случае указатели на первую и последнюю ячейки списка. Поиск в двунаправленном списке возможен как с начала, так и с конца. При обновлении записи в таком списке происходит изменение прямых и обратных указателей. В целом наличие обратного указателя позволяет упростить алгоритм изменения указателей, так как обратный указатель удаляемой записи хранит адрес ячейки с логически предшествующей записью. В односвязном списке этот адрес необходимо определить с помощью дополнительных процедур, что удлиняет процессы поиска и ведения информационного массива.

Для реализации связанного представления данных язык программирования должен располагать определенными средствами (иметь данные типа «указатель»). Если же такое средство в язы-

ке отсутствует, то связанное представление может моделироваться с помощью структуры массива.

Дадим пояснения по основным структурам данных.

Линейные структуры данных и их хранение. *Массив* — это линейная структура данных фиксированного размера, реализуемая с использованием последовательного представления данных. Каждый элемент массива идентифицируется одним или несколькими индексами (целыми числами, значения которых определяют позицию соответствующего элемента в массиве и используются для доступа к этому элементу). В зависимости от числа индексов различают одномерные и многомерные массивы.

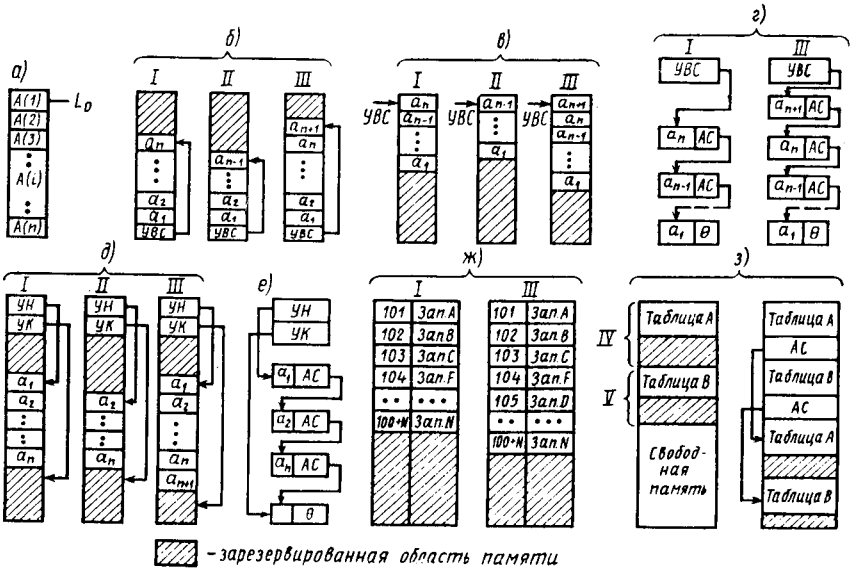


Рис. 5.8. Линейные структуры данных и их представление в памяти МЭВМ:
а — вектор; *б* — стек при последовательном представлении; *в* — стек с неизменным указателем; *г* — стек при связанном представлении; *д(е)* — очередь при последовательном (связанном) представлении данных; *ж* — последовательное представление таблицы; *з* — представление в памяти двух динамически изменяющихся таблиц; *I* — исходное состояние стека, очереди или таблицы; *II* — исключение элемента; *III* — включение элемента; *IV* (*V*) — блок памяти таблицы А (В); L_0 — адрес базы вектора; УВС — указатель вершины стека; УН (УК) — указатель начала (конца); АС — указатель связи; θ — указатель отсутствия элементов в данном направлении

Одномерный массив называется *вектором*. Вектор $A=\{A(1), A(2), \ldots, A(N)\}$ — это последовательность элементов (записей), размещенных в смежных ячейках памяти. Единственный индекс вектора указывает позицию каждого элемента в последовательности. В целом вектор определяется *адресом базы* (адресом первого байта, выделенного для первого элемента вектора), размером элементов и их числом или размером элементов и диапазоном изменения индекса (рис. 5.8, *а*). Если обозначить через L_0 адрес перво-

го байта в блоке памяти, выделенном для хранения вектора, через C — число байтов, выделенное для хранения каждого элемента, то адрес A_i любого i-го элемента определится как

$$\text{loc}(A_i) = L_0 + C(i-1),$$

где знак loc (от англ. location) — определение местоположения. Представление вектора в памяти не зависит от того, как он описывается в языке программирования. При любом описании оно будет одинаковым.

Двумерный массив называется *матрицей*. Каждый элемент матрицы определяется двумя индексами. Если массив является многомерным, то он может быть представлен эквивалентным одномерным массивом, т. е. вектором. Для n-мерного массива указывают число размерностей, а также верхнюю и нижнюю границы диапазона изменения индекса.

Стек — это линейная структура переменного размера, позволяющая в отличие от массива включать и исключать элементы и обеспечивающая динамическое изменение объема данных в стеке во время выполнения программы. Особенность стековой структуры состоит в том, что доступ к элементам, их включение и исключение возможны только с одного конца структуры — с вершины стека. Поэтому первым из стека выбирается тот элемент, который включается в стек последним. Данные в такой структуре обрабатываются по принципу «последним пришел — первым ушел» (структура типа LIFO).

Структура стека является структурой данных с ограниченным доступом, так как доступ разрешается только к элементу, находящемуся в вершине стека и называемому *текущим*. Информацию о позиции текущего элемента стека хранит *указатель вершины стека* (УВС), размещаемый обычно в головной ячейке стека.

При хранении стеков может использоваться как последовательное, так и связанное представление. При последовательном представлении необходимо знать предельный размер стека, под который отводится соответствующий блок памяти. Внутри этого блока памяти стек растет и сокращается. Первая ячейка содержит УВС, причем если стек пуст, то УВС указывает на себя, а при включении каждого нового элемента УВС увеличивается на единицу (рис. 5.8, *б*). Может быть и такая организация стека, что значение УВС остается неизменным, а доступ осуществляется всегда к одной и той же ячейке блока памяти, зарезервированного под стек. На эту ячейку устанавливается УВС, а все остальные элементы стека перемешаются вниз или вверх соответственно внутри блока памяти (рис. 5.8, *в*). Недостаток последовательного представления стека заключается в недоиспользовании части зарезервированной памяти или в опасности ее переполнения.

При связанном представлении стека все его элементы разбросаны по памяти и связаны между собой указателями. УВС указы-

вает на ячейку с верхним элементом стека, при включении или исключении элементов значение УВС изменяется. При связанном представлении данных стек может расти неограниченно (рис. 5.8, *г*).

Структура стека удобна при необходимости быстрого выполнения операций включения и исключения без оценки содержательного смысла. Она широко применяется при реализации вложенных подпрограмм, многоуровневых прерываний, решении рекурсивных задач.

Очередь — это линейная структура переменного размера. Исключение элементов из очереди допускается с начала очереди, а включение элементов в очередь — с конца очереди, при этом обслуживание производится по принципу «первым пришел — первым ушел» (структура типа FIFO). Доступ к элементам очереди осуществляется по указателю начала (УН), указывающему на элемент очереди, который первым будет исключаться, и указателю конца (УК), устанавливающемуся на свободную ячейку памяти, следующую за последней записью в очереди.

Аналогично стеку для реализации очереди в памяти МЭВМ используется последовательное и связанное представление данных (рис. 5.8, *д*, *е*).

Структура очереди используется, например, при моделировании систем с разделением времени и др. Принятый принцип организации обслуживания очереди во многом определяет эффективность функционирования моделируемых систем.

Таблица — это линейная структура данных, каждый элемент которой характеризуется определенным значением ключа и доступ к элементам которой осуществляется по ключу.

При хранении таблиц в памяти МЭВМ может использоваться как последовательное, так и связанное представление данных. При последовательном представлении таблица хранится в виде последовательного списка. Записи таблицы располагаются одна за другой в зарезервированном блоке памяти. Такие таблицы легко составлять и дополнять с размещением новых записей в конце таблицы за минимальное время. Однако поиск в таких таблицах длителен, так как требуются последовательный просмотр всех записей начиная с первой и анализ значений их ключевых полей до момента нахождения нужной записи или выдачи сигнала отсутствия нужной записи после полного просмотра всей таблицы.

Упорядочение записей таблицы по какому-либо принципу (например, по возрастанию значения ключа или по частоте обращения к записям) и хранение их в виде упорядоченного последовательного списка способствуют существенному ускорению поиска, но усложняют их ведение. Например, включение в упорядоченный последовательный список новой табличной записи требует определения места, которое должна занять новая запись по значению ее ключа, освобождения соответствующей ячейки памяти с перезаписью части массива и передвижением всех записей на одну ячейку (рис.

5.8, *ж*). Такой способ хранения таблиц особенно удобен при заранее известном предельном размере таблицы, часто повторяющихся процедурах поиска нужных данных, но при редких процедурах включения и исключения данных.

При связанном представлении записи каждой таблицы размещаются последовательно в зарезервированных блоках памяти. По мере роста таблиц эти блоки заполняются. Если какой-то блок памяти заполняется полностью, то для этой таблицы выделяется новый блок памяти, который связывается указателем с ранее заполненным блоком (рис. 5.8, *з*). Такая структура хранения удобна для размещения в памяти табличных структур с заранее не известным числом элементов.

При хранении таблиц, все записи которых имеют разные значения ключей K_i, часто используют способ прямого доступа к каждой записи таблицы. Для этого находится функция преобразования (или расстановки) $f(K_i)$, принимающая для любого $0<i\leqslant N$ целое значение от 0 до m, которое рассматривается как адрес ячейки памяти, в которой размещается запись с ключом K_i. Доступ к любой записи осуществляется путем непосредственного вычисления по значению ключа адреса хранения этой записи. Время поиска в таких таблицах минимально и определяется в основном временем вычисления $f(K_i)$.

Нелинейные структуры данных подробно рассмотрены в [7] и в настоящей книге не рассматриваются.

На практике наиболее распространенными преобразованиями структур данных являются: *копирование* (размножение некоторого элемента массива в нескольких экземплярах); *маскирование* (запрет обработки некоторого элемента массива); *перестановка* или *перегруппировка* (сдвиг, сортировка, слияние элементов массива).

Программная реализация подобных алгоритмов упорядочения информации требует довольно больших ресурсов памяти и затрат времени. Так как МЭВМ имеют ограниченные ресурсы памяти и производительности, то для них целесообразнее приспосабливать и перестраивать структуры памяти под структуры данных путем создания нелинейной структуры памяти и перестраивания ее в зависимости от объема и вида структуры данных.

На построение эффективных нелинейных структур памяти МЭВМ влияет выбор соответствующего метода управления памятью. Под *управлением памятью* понимают выбор способа стратегии распределения памяти и его реализации, контроля за состоянием каждой ячейки памяти, управления освобождением памяти и динамикой распределения конкретных областей памяти под определенные процессы. Обычно любая стратегия распределения памяти реализуется при тесном взаимодействии технических и программных средств.

Основные стратегии распределения памяти в МПСУ. Выбор стратегии более полного использования ресурсов памяти в МЭВМ ведет к усложнению операционной системы (ОС), управляющей эти-

ми ресурсами, дополнительным затратам программных и аппаратных средств, вводимых для хранения различного рода таблиц и содержимого регистров, связанных с прерываниями, формированием адресов и др. Если для ЭВМ с большой памятью эти затраты малозаметны, то в МЭВМ ОС и различные таблицы могут занять значительную часть памяти, в связи с чем стремятся:

— максимально возможную часть функций ОС по управлению памятью реализовать на отдельном МП со своим ОЗУ, объем которого должен быть достаточен для реализации этих функций;

— ряд массивов (таблицы констант и пр.), описываемых соответствующими математическими выражениями с заданной точностью, вычислять, а не хранить в памяти;

— применять преобразование адресов и данных по соответствующим законам, обеспечивающим сжатие и упаковку информации при записи в память и эффективную ее выборку при считывании с учетом требуемой степени достоверности.

При выборе стратегии получения максимальной производительности МЭВМ и эффективного распределения ресурсов памяти стремятся:

— максимально возможную часть функций ОС по управлению памятью реализовать на базе специализированных БИС;

— учитывать специфику наборов операций в алгоритмах и возможность эффективной их реализации на многофункциональных, ассоциативных и других типах БИС ЗУ;

— исключать непроизводительные затраты времени, связанные с организацией режимов работы памяти (например, регенерацию в динамических БИС ЗУ и др.).

Если эти задачи решаются на этапе построения встроенных МПСУ, реализуемых на МПК БИС, то обычно выбирают наиболее предпочтительные модули БИС по критериям минимума стоимости и времени реализации алгоритмов, выделяют из множества МПК БИС набор базовых модулей, совокупность операций которых максимально способствует эффективной реализации фрагментов заданного алгоритма при минимальном объеме операций. Однако не всегда возможно приспособить структуру МПСУ под классы решаемых задач, а структуры данных — под структуры имеющихся микропроцессорных средств.

Если перечисленные задачи решаются на этапе построения МПСУ, реализуемых на базе серийных микроЭВМ, то для эффективной реализации определенных классов алгоритмов стремятся оптимально распределить ресурсы серийной МЭВМ без каких-либо изменений ее архитектуры и структуры или модифицировать ее с подключением минимума дополнительных типовых модулей.

Если на выбранной МЭВМ невозможно реализовать заданный алгоритм за приемлемое время, то производят ее модификацию с заменой отдельных модулей, входящих в ее состав, другими серийными модулями, удовлетворяющими техническим ограничениям,

стоимости, временным и емкостным параметрам. Если таких модулей не окажется, то переходят к анализу следующей серийной МЭВМ или ее модификации.

Среди всех проанализированных МЭВМ выбирают ту, у которой лучшие критерии по времени реализации алгоритмов и степени распределения ресурсов при выполнении других технических ограничений. Если же среди всех серийных МЭВМ не оказывается подходящей, то переходят к разработке новой архитектуры и структуры МПСУ или МЭВМ. Из-за ограниченности объема основной памяти МЭВМ может возникнуть необходимость размещения некоторого объема информации во внешней памяти, что решается для каждой задачи в отдельности с учетом приоритетов реализуемых фрагментов алгоритма.

Контрольные вопросы

1. Перечислите основные этапы методики оптимизации сложных систем и выбора рациональной технической структуры МПСУ. 2. Что такое интерфейс МПСУ? Каковы отличительные признаки интерфейсов, способы их функциональной и структурной организации? 3. Какими показателями характеризуются интерфейсы? 4. В чем состоит формализация синтеза интерфейсов МПСУ и каким образом эта задача может быть решена? 5. В чем заключается проблема оптимизации ресурсов МЭВМ и какими путями она может быть решена?

ГЛАВА 6

МЕТОДЫ ПОВЫШЕНИЯ НАДЕЖНОСТИ ПРИ ПРОЕКТИРОВАНИИ МПСУ

Эксплуатация микропроцессорных систем управления в реальных производственных условиях и реальном масштабе времени требует обеспечения живучести, долговечности, надежности и отказоустойчивости их функционирования, так как отказы или сбои технических средств и ошибки в программном обеспечении могут привести к тяжелым экономическим и другим нежелательным последствиям. Однако возможность быстрого обнаружения и локализации отказов и сбоев в функционировании систем управления требует введения дополнительных программно-аппаратных средств, зависящих от степени надежности МПСУ для конкретных объектов и уровней их сложности.

§ 6.1. ОБЩИЕ СВЕДЕНИЯ

Для повышения надежности систем управления, проектируемых на микропроцессорных средствах, на практике применяют методы структурной (аппаратной), информационной и программной избыточности.

Структурная избыточность — наиболее распространенный способ, применяемый в системах управления. Она предполагает вклю-

чение в состав системы избыточных элементов, позволяющих скомпенсировать отказы отдельных элементов устройства или системы в целом и обеспечить надежное их функционирование. В целом увеличение надежности введением структурной избыточности неразрывно связано с увеличением стоимости проектируемых устройств и представляет собой сложную оптимизационную проблему, связанную с правильным выбором технико-экономических критериев функционирования.

Требуемого уровня надежности можно достичь и введением *программной избыточности,* причем ее можно вводить в любую уже созданную систему, не нарушая существующей конфигурации технических средств. Однако наиболее эффективно программная избыточность проявляется лишь в комплексе с аппаратной избыточностью.

Информационная избыточность исходных и промежуточных данных, обрабатываемых программными и аппаратными комплексами, существенно влияет на качество нормального функционирования и время восстановления данных с заданной степенью достоверности. При этом для сохранения важных данных обычно применяют дублирование или утроение с соответствующей дисциплиной контроля их сохранности и периодическим обновлением, а для менее важных данных — помехозащищенные коды, способные обнаруживать и (или) исправлять некоторые искажения информации.

Как показывает практика, при работе МПСУ большая часть ошибок (около 75%) вызывается сбоями, возникающими в результате флуктуаций напряжений источников питания, удара, внешних помех, статических зарядов и др. Сбои могут иметь тяжелые последствия, вызывая аварийные ситуации, «тяжелый останов», нарушение хода технологического процесса и др.

В МПСУ различают следующие типы сбоев: сбои памяти; сбои МП; сбои канала; сбои периферийных УВВ; ошибки в ПО.

Сбои памяти в МПСУ устраняются с помощью аппаратных средств с коррекцией одиночных ошибок памяти на базе различных корректирующих кодов или встроенных средств самоконтроля.

Сбои МП устраняются с помощью следующих методов:

— программного метода контрольных точек, заключающегося в периодическом запоминании информации о текущем состоянии вычислительного процесса с целью продолжения работы после сбоя с той точки, в которой произошло запоминание информации;

— программного метода повторения команд, заключающегося в восстановлении всей необходимой информации для повторения команды, которая не была выполнена из-за сбоя, и повторном ее выполнении;

— аппаратно-микропрограммного метода, суть которого заключается в аппаратной поддержке некоторой части операционной системы на микрокомандном уровне при реализации метода повторения команд.

Сбои канала в МПСУ, возникающие при выполнении команд ввода — вывода, устраняются с помощью аппаратно-микропрограммных средств и повторного выполнения команд ввода — вывода или прерывания с записью слова состояния канала и дополнительной информации, уточняющей место и условия, соответствующие ошибке. В этой ситуации восстановление выполняется на программном уровне с помощью специальных программ для соответствующего устройства с восстановлением исходного положения носителя.

Сбои периферийных УВВ в МПСУ обычно устраняются с помощью программных средств, обеспечивающих анализ типа ошибок, уточнение состояния УВВ при сбое и исправление сбойной ситуации путем многократного повторения начальной программы, при выполнении которой произошел сбой. Число повторений зависит от типа ошибок и периферийного оборудования.

Известно, что *ошибки в программном обеспечении* (ПО) вносятся при его проектировании (61—64%) и программировании (36—39%) [31]. Основная масса ошибок в ПО обычно возникает: при переходе в аварийные ветви алгоритмов, на стыках программных модулей при сочетаниях редких событий, не учитываемых при отладке программ; при несоответствии времени исполнения команд динамическим характеристикам системы; при нехватке памяти ОЗУ в условиях ее динамического распределения в режиме максимальной загрузки; при перераспределении функций между МЭВМ при горячем резервировании и др. В целом наличие ошибок в программах МПСУ может привести как к искажениям вычислительного процесса и данных, вызывающим полное или кратковременное прекращение функционирования системы, так и к искажениям (сбоям), быстро устраняющимся и практически не влияющим на результаты предыдущих вычислений.

Рассмотрим подробнее особенности принципов обеспечения надежности и отказоустойчивости, применяемых при проектировании МПСУ.

Методы структурного резервирования. В соответствии с ГОСТ 13377—75 основными методами структурного резервирования являются: резервирование замещением (РЗ), или динамическое резервирование, постоянное резервирование (ПР), или статическое резервирование, и комбинированное резервирование (КР). РЗ различается способом подключения избыточного оборудования (фиксированный или скользящий), состоянием резервного оборудования (нагруженное или «горячее», облегченное, ненагруженное или «холодное», комбинированное).

Фиксированное резервирование (ФР) устройств характеризуется однозначным соответствием одного (или группы) резервного блока (резервных блоков) только одному рабочему, или основному, блоку, при этом резервный блок может заместить рабочий блок в случае отказа последнего. Данный метод резервирования широко используется на практике с применением различной кратности замещения (2, 3, 4 и более).

Скользящее резервирование (СР) устройств характеризуется таким способом замены группы основных блоков группой резервных, когда любой из резервных блоков может быть подключен вместо любого отказавшего рабочего блока. При этом избыточное оборудование используется эффективнее, чем при ФР.

Постоянное резервирование (ПР) характеризуется построением избыточных структур как при использовании специальных восстанавливающих органов (ВО) адаптивного или неадаптивного типа, так и без них. Для адаптивных ВО функция восстановления работоспособности устройств изменяется по мере накопления отказов в избыточной структуре, для неадаптивных ВО она остается неизменной в течение всего времени работы устройств. На практике наибольшее распространение получил мажоритарный метод резервирования (см. § 6.2).

Комбинированное резервирование (КР) характеризуется введением структурной избыточности на базе комбинации методов РЗ и ПР. Оно обеспечивает более высокую надежность и живучесть проектируемых систем и базируется на трех общих принципах: гибридном, итеративном и смешанном.

КР на основе гибридного принципа характеризуется использованием комбинаций двух методов резервирования в одном избыточном блоке. Комбинация ПР с ВО и ФР известна под названием гибридного резервирования, а комбинация ПР с ВО и СР — параллельно-гибридного резервирования.

КР на основе итеративного принципа характеризуется использованием различных способов резервирования уже зарезервированных различными методами блоков, вложенных друг в друга, а также применением различных способов подключения резервного оборудования.

КР на основе смешанного принципа характеризуется применением разновидности методов резервирования для основного и для резервного оборудования с учетом надежности блоков сопряжения, требующихся для преобразования избыточных сигналов предыдущего устройства во входные сигналы последующего устройства.

Из сравнительного анализа применения в МПСУ методов структурной избыточности можно отметить следующее.

При построении надежных МПСУ необходимо учитывать основные особенности микропроцессорной элементной базы: элементная единица — отдельные БИС и СБИС со сложным функциональным содержанием специализированного и универсального назначения; магистральный способ обмена информацией; невозможность непосредственного электрического объединения выводов разных модулей; отсутствие резервирования внутри корпуса (на кристалле БИС).

Метод ФР требует больших аппаратурных затрат, поэтому он не всегда приемлем, особенно при проектировании систем в условиях весогабаритных ограничений. В этом случае более выгоден метод СР, являющийся особенно привлекательным для создания специализированных МПСУ с развитым ПО, программной локализацией отказавших блоков и подключением резервных.

Методы ПР широко применяются в МПСУ, однако при больших интервалах времени трудно удовлетворить заданным требованиям надежности функционирования систем.

Построение надежных МПСУ общепромышленного назначения может быть достигнуто при использовании метода ПР на основе адаптивных восстанавливающих органов (АВО). Данный метод легко адаптируется к структурной организации системы; обладает широкими возможностями выбора функций восстановления в са-

мом АВО, использования существующего программного обеспечения с минимумом доработок.

Методы КР широко применяются при проектировании МПСУ ответственного назначения (бортовые и специализированные ЭВМ со средствами самопроверки и последующего самовосстановления, орбитальные космические станции и корабли и др.).

Особенности обеспечения надежности подсистем памяти МПСУ на основе методов избыточности. Известно, что до 80—95% всего объема МПСУ занимают ЗУ (подсистема памяти и разнообразные триггеры, регистры, счетчики и др.), характеризующиеся тождественным преобразованием «вход — выход». Учитывая данную специфику ЗУ, целесообразно использовать их аппаратные средства для обработки информации и для исправления отказов и сбоев, возникающих в аппаратуре, за счет резервов по быстродействию. Этого, например, можно достичь простым сравнением по модулю два (mod 2) значений выхода Y и входа X для определения в ЗУ вектора ошибки $E=\{X\}\vee\vee\{Y\}$.

Хранящаяся информация в ЗУ может искажаться, приводя к ошибкам по причине производственных *отказов* (обнаруживаемых *дефектов*), устойчивых эксплуатационных отказов (возникающих отказов и неисправностей) или функциональных эксплуатационных *отказов (сбоев)*. Отказ ЗУ характеризуется неравенством нулю составляющих векторов ошибки ($E_{0i}\neq 0$, $E_{1i}\neq 0$), сохраняющихся в течение всего времени работы устройства. Сбой ЗУ характеризуется:

а) кратностью ошибки σ_1, определяемой числом единиц в векторе ошибки:

$$\sigma_1 = |E| = |\{E_1, E_2, \ldots, E_N\}|;$$

б) степенью асимметрии ω, определяемой из отношения

$$\omega = \frac{|E_0| \vee |E_1|}{|E|},$$

где $E_0(E_1)$ — количество единиц в векторе ошибки E, вызванных переходами $1\to 0$ $(0\to 1)$;

в) длительностью сбоя, определяемой как

$$T_{сб} = \sum_{i=1}^{N} t_i, \quad \text{при условии, что } E(t_i)\neq E(t_{i+1})\neq \ldots \neq E(t_{i+N}).$$

Основной причиной сбоев в работе БИС ЗУ, особенно динамических, является чувствительность БИС к облучению светом или радиацией. Интенсивность сбоев ЗУ значительно повышается с увеличением высоты уровня эксплуатации приборов, при этом сбоям подвержены не только динамические, но и статические БИС ЗУ.

Частота появления постоянных отказов ЭП, возникающих в результате физической неисправности при эксплуатации, как правило, на порядок ниже интенсивности сбоев ЭП и составляет 10^{-5}—10^{-7} $ч^{-1}$. С ростом емкости БИС ЗУ вероятность отказов отдельных ЭП увеличивается.

Отказы ЭП в БИС ЗУ возникают последовательно во времени, имеют тенденцию к накоплению, причем интервал времени между двумя последовательными отказами, как правило, достаточен для определения местоположения ошибки и фиксации ее в дополнительной памяти до появления следующей ошибки. Знание местоположения ошибки из-за отказа существенно упрощает реализацию декодирующих схем, исправляющих многократные ошибки.

Анализ надежности систем памяти на БИС ЗУ, использующих дублирование, межоритарное резервирование, коды, исправляющие одиночные и двойные ошибки, с учетом значений времени наработки, емкости памяти, интенсивности отказов, сложности дополнительных схем обрамления показал, что наиболее эффективным методом введения избыточности является применение корректирующих кодов [19, 24].

Представляя МПСУ известной в теории автоматов моделью, включающей в себя исходный автомат и автомат с ошибками (рис. 6.1), можно синтезировать надежные микропроцессорные устройства, обеспечивающие исправление и обнаружение ошибок, вызванных дефектами элементов и сбоями, при введении незначительной избыточности аппаратурных средств.

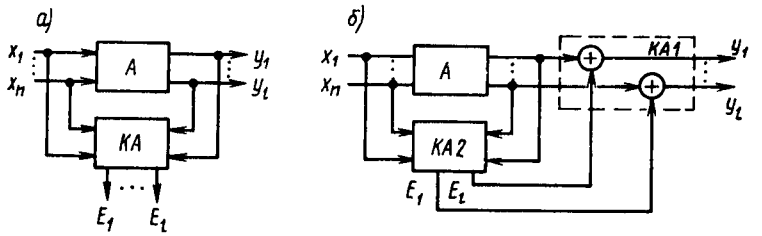


Рис. 6.1. Представление МПСУ автоматной моделью:
а — модель автомата с обнаружением ошибок; *б* — модель автомата с обнаружением и исправлением ошибок; *А* — исходный автомат; КА — корректирующий автомат; + — схема по mod 2

Коррекция ошибок в исходном автомате А может осуществляться как с помощью одного автомата (КА), обнаруживающего ошибки (рис. 6.1, *а*), так и двух автоматов КА1 и КА2 (рис. 6.1, *б*), обнаруживающего и исправляющего ошибки соответственно. Известно, что оптимальные затраты на реализацию КА должны составлять примерно 1,5—10% от затрат на аппаратурную реализацию исходного автомата А.

В заключение заметим, что суммарные аппаратные затраты, отводимые на реализацию корректирующих кодов, обеспечивающих коррекцию ошибок с максимально возможной кратностью, соизмеряются со сложностью реализации схем на мажоритарной логике, обеспечивающих лишь исправление ошибок малой кратности ($\sigma_1 < 2$) на одноименных выходах.

В целом учет специфики функционирования БИС ЗУ, способов введения в состав ЗУ разнообразных видов избыточности позволяет существенно повысить надежность и помехоустойчивость подсистем памяти МПСУ и обеспечивает:

а) при использовании корректирующих кодов и временной избыточности — исправление многократных ошибок, вызванных дефектами в изготовлении элементов и одиночными сбоями;

б) при использовании кодов, обнаруживающих ошибки, и временно́й избыточности — исправление ошибок, вызванных одиночным дефектом и сбоями;

в) при использовании аппаратной и временно́й избыточности — корректирование ошибок любой кратности при введении избыточности, обратно пропорциональной числу разрядов кодового слова;

г) при использовании временно́й избыточности — определение вектора ошибки любой кратности (операцией сравнения по mod 2) и корректирование сбоев.

Рассмотрим более подробно применение перечисленных способов введения избыточности в структуре ЗУ МПСУ.

§ 6.2. СИНТЕЗ МПСУ С ПРИМЕНЕНИЕМ АППАРАТУРНОЙ ИЗБЫТОЧНОСТИ

Вопросы проектирования надежных МПСУ обычно рассматриваются с учетом влияния использования МП на надежность систем управления; соответствия МП, его структуры, программного и математического обеспечения конкретным условиям применения; надежности самого МП как элемента системы, способов его диагностики, контроля, резервирования и др.

В последнее время разрабатываются высоконадежные *многопроцессорные,* или *мультимикропроцессорные,* системы управления (ММПСУ), устойчивые к отказам, обладающие свойством «постепенно́й деградации» структуры при возникновении отказов с реконфигурацией ее за счет введения дополнительных программно-аппаратных средств обнаружения отказавших модулей, изоляции этих модулей в системе и применения таких операционных систем, которые при отказах модулей обеспечивают сохранение данных, обрабатываемых системой в реальном масштабе времени, и позволяют завершить решение не законченных на момент отказа задач.

Известно, что одновременная обработка информации на n параллельно работающих МП позволяет увеличить производительность МПСУ в $\log_2 n$ раз, а живучесть всей системы — на два-три порядка. В то же время применение ММПСУ ведет к снижению эффективности их функционирования из-за увеличения системных издержек, связанных с усложнением схем управления и защиты памяти, снижением быстродействия, сложностью контроля и локализации отказов, возможностью ошибок и их размножения из-за межпроцессорного обмена. В целом ММПСУ, обеспечивающая длительную непрерывную работу на основе структурного резервирования с восстановлением резерва после устранения отказа, должна обладать следующими свойствами:

— при возникновении отказа или сбоя работа системы не должна прекращаться;

— все отказы и сбои должны обнаруживаться системой контроля до того, как они вызовут нарушение работы системы;

— отказавшая аппаратура должна автоматически заменяться резервной;

— при появлении сбоя или отказа должна производиться попытка повтора операции, в случае неудачи он должен рассматриваться как устойчивый отказ;

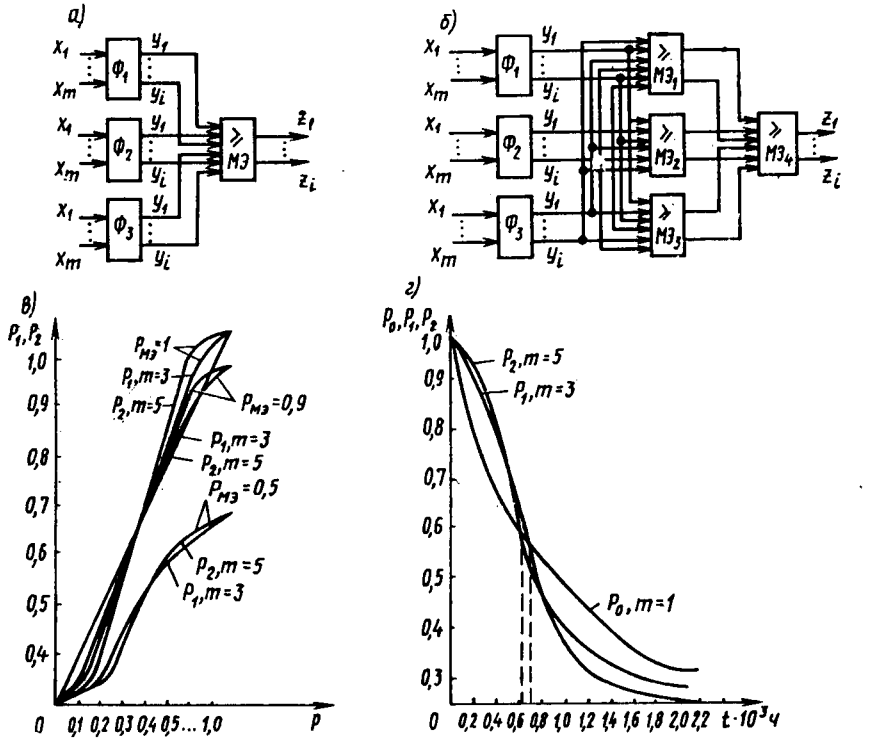


Рис. 6.2. Мажоритарные структуры МПС с одно- и многократной связью: *а*, *б* — структуры с одно- и многократной связью, *в* — зависимость $P_i = f(P)$ при $P_{МЭ}$ = const и $m_1 = 3$, $m_2 = 5$; *г* — зависимость $P_i = f(t)$ при $m = 3$; $P_2 = f(t)$ при $m = 5$, $\lambda = 10^{-6}$ ч$^{-1}$ $n_A = 1000$; Ф — функциональный модуль, МЭ — мажоритарный элемент

— при автоматической замене отказавшей аппаратуры не должно происходить потери информации;

— при устранении отказа не должно происходить нарушений в выполнении программы;

— восстановление резерва не должно вызывать остановки работы системы.

Особенности проектирования МПСУ с мажоритарным резервированием. Постоянное резервирование МПСУ с ВО

неадаптивного типа. Распространенными на практике структурами с аппаратурной избыточностью являются мажоритарные структуры, реализуемые по принципу большинства с числом блоков m ($m=3, 5, 7$) и однократной или многократной связью (рис. 6.2).

Вероятность безотказной работы (ВБР) мажоритарной структуры с однократной связью (рис. 6.2, *а*) при отказе не более чем $(m-1)/2$ блоков определяется как

$$P_j = P_{\text{МЭ}} \sum_{i=0}^{(m-1)/2} \binom{m}{i} P^{m-i}(1-P),$$

где $P_{\text{МЭ}}$ — ВБР мажоритарного элемента (МЭ); P — ВБР неизбыточного функционального блока Φ_i ($i=\overline{1, m}$).

Например, ВБР избыточной структуры с однократной связью

$$P_1 = [P^3 + 3P^2(1-P)]P_{\text{МЭ}} = (3P^2 - 2P^3)P_{\text{МЭ}} \text{ при } m=3;$$

$$P_2 = P_{\text{МЭ}} \sum_{i=0}^{2} \binom{5}{i} P^{5-i}(1-P)^i = P_{\text{МЭ}}(6P^5 - 15P^4 + 10P^3)$$

при $m=5$.

Анализ зависимостей $P_j = f(P)$ при $P_{\text{МЭ}} = \text{const}$ и $m_1=3$, $m_2=5$ (рис. 6.2, *в*) показывает, что в избыточных структурах с однократной связью выигрыш в надежности по сравнению с неизбыточной структурой возможен лишь при наличии высоконадежного МЭ ($P_{\text{МЭ}} \approx 1$) и ВБР неизбыточного Φ_i, равной $P > 0{,}5$. Однако ВБР всей структуры не может превзойти ВБР МЭ, а при его отказе выходит из строя вся структура в целом.

Коэффициент выигрыша в надежности γ избыточной структуры с однократной связью по сравнению с неизбыточной структурой

$$\gamma_{P1} = \frac{P_1}{P} = \frac{P_{\text{МЭ}}(3P^2 - 2P^3)}{P} = P_{\text{МЭ}}(3P - 2P^2) \text{ при } m=3; \qquad (6.1)$$

$$\gamma_{P2} = \frac{P_2}{P} = \frac{P_{\text{МЭ}}(6P^5 - 15P^4 + 10P^3)}{P} = P_{\text{МЭ}}(6P^4 - 15P^3 + 10P^2) \qquad (6.2)$$

при $m=5$.

Взяв производную $d\gamma_P/dP$ от выражений (6.1), (6.2) и приравняв ее нулю, найдем корень соответствующего уравнения, после чего, подставив его в (6.1), (6.2), найдем максимальный выигрыш в надежности избыточной структуры.

Расчеты показывают, что максимальный выигрыш в надежности в таких структурах равен:

при $m=3$ $\gamma P_{1\max} = 1{,}125 P_{\text{МЭ}}$, определяясь из области параметров $1 \leqslant \gamma_{P1} \leqslant 1{,}125$ при $0{,}88 < P_{\text{МЭ}} \leqslant 1$, $0{,}5 \leqslant P \leqslant 1$;

при $m=5$ $\gamma_{P2\max}=1{,}197$ $P_{\text{МЭ}}$, определяясь из области параметров $1\leqslant\gamma_{P2}\leqslant 1{,}197$ при $0{,}88\leqslant P_{\text{МЭ}}\leqslant 1$ и при $0{,}5\leqslant P\leqslant 1$.

ВБР мажоритарной структуры с многократной связью (рис. 6.2, *б*) определяется как

$$P_j=\sum_{i=0}^{(m-1)/2}\binom{m}{i}P^{m-i}(1-P)^i+\sum_{k=0}^{(m-1)/2}\binom{m}{k}P_{\text{МЭ}}^{n-k}(1-P_{\text{МЭ}})^k.$$

Если обозначить через n_A количество элементов в неизбыточной структуре (функциональном блоке), а через $n_{\text{м}}$ количество элементов в МЭ, то относительная сложность α избыточных структур с одно- и многократными связями определяется так:

при $m=3$ $\alpha_1=\dfrac{n_A}{2n_A+n_{\text{м}}}$; $\alpha_2=\dfrac{n_A}{2n_A+4n_{\text{м}}}$;

при $m=5$ $\alpha_1=\dfrac{n_A}{4n_A+n_{\text{м}}}$; $\alpha_2=\dfrac{n_A}{4n_A+6n_{\text{м}}}$;

Анализ ВБР избыточных мажоритарных структур $P_1=f(t)$ при $m=3$, $P_2=f(t)$ при $m=5$ и неизбыточной $P_0=f(t)$ структуры при использовании реальных значений $\lambda=10^{-6}$ ч$^{-1}$, $n_A=1000$ показывает (рис. 6.2, *г*), что выигрыш в безотказности избыточных структур можно получить лишь на начальном отрезке времени $(0-t_0)$ их функционирования, где $t_0=700$ ч. Далее надежность их падает и становится меньше надежности неизбыточной системы. Средняя наработка на отказ этих систем составляет $T_1=0{,}83T_0$; $T_2=0{,}78T_0$, $T_0=1/(n_A\lambda)$, что свидетельствует о применимости подобных мажоритарных структур МПСУ лишь для специализированных применений с небольшим периодом функционирования.

Особенности проектирования МПСУ с мажоритарным резервированием на основе МПК БИС. В общем случае проектирование МПСУ с мажоритарным резервированием на уровне как отдельных БИС, так и функциональных модулей из набора БИС осуществляется с использованием магистральной организации (рис. 6.3, *а*), в которой набор S_i БИС $\{S_i|i=\overline{1,\ k}\}$ связывается друг с другом с помощью совокупности $L_j\{L_j|j=\overline{1,\ m}\}$ магистралей. В такой неизбыточной структуре МПСУ функционирование каждого модуля в любой момент времени можно представить в виде двух соотношений:

а) $S_i\rightarrow L_j\rightarrow S_v$, если осуществляется передача информации по магистрали L_j из модуля S_i в модуль S_v;

б) $S_i\rightarrow S_i$, если происходит обработка информации внутри модуля без выдачи ее вовне.

Повышения надежности МПСУ достигают введением набора

магистральных решающих органов (МРО) $\{M_j\}$, реализующих выходную функцию вида $F_j=f[L_j^{(1)}, L_j^{(2)}, ..., L_j^{(R)}]$, $j=\overline{1, m}$, и распределяющих R идентичных сигналов по избыточным магистралям $L_m^{(R)}$ (рис. 6.3, *б*). В такой избыточной структуре операция $S_i \to L_j \to S_v$ теперь расширяется путем восстановления избыточной информации в МРО и приобретает вид $S_v \to L_j \to M_j \to S_v$. При этом исправляются ошибки входного набора МРО, вид и количество которых определяются функцией F_j. К этим ошибкам относятся ошибки как магистралей, так и модулей. Однако при длительном выполнении операций вида $S_i \to S_i$ в некоторых модулях могут накапливаться и запоминаться случайные ошибки из-за сбоев, отказов, нарушений в синхронизации, что ведет к увеличению вероятности отказа всей системы. В этом случае в структуре МПСУ вводится ряд промежуточных восстановлений в определенные моменты времени $\tau_1 < \tau_2 < ... < \tau_n < T$ по схеме $S_i \to L_j \to M_j \to S_v$.

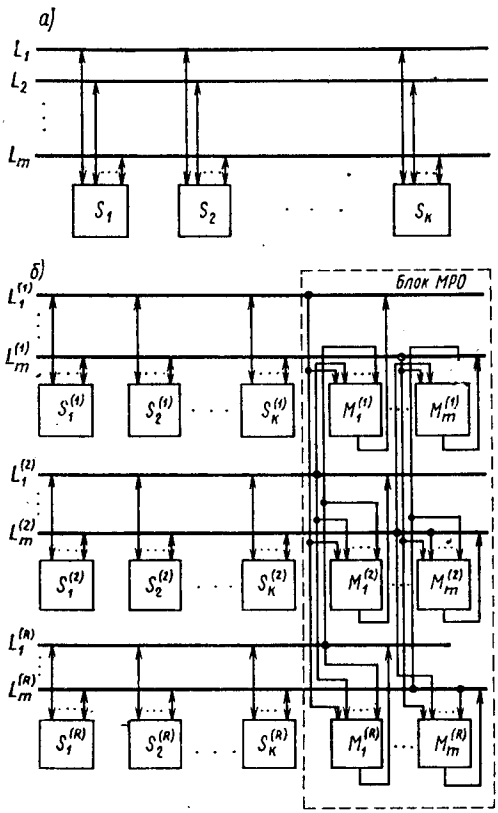


Рис. 6.3. Структуры МПС с мажоритарным резервированием на основе МПК БИС и магистральных решающих органов (МРО): *а* — неизбыточная структура; *б* — избыточная структура

Например, если в трехканальной МПСУ в модуле $S_i^{(1)}$ в момент t_1 возник сбой, вызвавший искажение хода вычислительного процесса, а в модуле $S_i^{(2)}$ сбой возник в момент $t_2 > t_1$, то к моменту T в восстанавливаемом наборе $\{S_i^{(1)}, S_i^{(2)}, S_i^{(3)}\}$ будет содержаться уже две ошибки, что при наличии мажоритарного МРО приведет к отказу МПСУ. Если же ввести два промежуточных восстановления в моменты $t_1 < \tau < t_2$ и $t_2 < \tau < T$, то в восстанавливаемых наборах будет содержаться только по одной ошибке, в результате чего МРО выдаст правильное решение и МПСУ не откажет.

Таким образом, построение МПСУ с использованием МРО обеспечивает возможность пространственно-временно́го разделе-

ния отказов. Свойство пространственного разделения отказов здесь проявляется в том, что отказы отдельных МП ведут к появлению ошибок на различных выходных шинах. Свойство же временно́го разделения отказов проявляется в том, что даже по отношению к одному выходу различные отказы проявляются в разные моменты времени. В отличие от принципов традиционного мажоритарного резервирования при данном подходе наиболее полно используются преимущества БИС, позволяющие усложнить алгоритм восстановления ошибочной информации и повысить эффективность этой процедуры. Кроме того, сочетание аппаратурной избыточности МПСУ с микропрограммным управлением позволяет создавать статически и динамически перестраиваемые системы на уровне функциональных узлов и отдельных МП и обеспечивать работоспособность структуры даже при наличии во всех МП отказавших узлов. Функции МРО при этом могут выполнять универсальные коммутирующие процессоры, входящие в состав МПК БИС.

В целом процесс проектирования МПСУ с мажоритарным резервированием на основе МПК БИС состоит в синтезе блока МРО согласно его месту включения и числу используемых магистралей; в определении оптимальных правил и моментов пространственно-временно́го восстановления избыточной информации в МРО; в определении надежности систем, достигаемой не введением в типовую структуру элементов восстановления ошибок, а реконфигурацией входящих в нее средств и соответствующей организацией вычислительного процесса.

Особенности проектирования МПСУ с комбинированным резервированием. Как указывалось выше, КР обеспечивает высокую надежность и живучесть проектируемых систем при гибком сочетании различных методов РЗ и ПР на основе гибридного, итеративного и смешанного принципов. Выбор того или иного принципа КР зависит от важности и ответственности того или иного устройства в системе, ее назначения, длительности эксплуатации, размера экономических, социальных, информационных потерь, возникающих при нарушении безопасности, нормального функционирования и других факторов.

Наиболее распространенным принципом КР является гибридный принцип с двумя его разновидностями — гибридным (ГР) и параллельно-гибридным (ПГР) резервированием, представляющим собой сочетание методов ПР с ВО, ФР или СР соответственно. Устройства с КР на основе гибридного принципа различаются: типом ВО; стратегией подключения резервных блоков; методами борьбы со сбоями; способами резервирования автомата надежности (АН), включающего в себя ряд специализированных блоков. Рассмотрим некоторые из способов их построения.

Известна типовая структура системы с ГР (рис. 6.4, *а*), состоящая из ядра, включающего в себя нечетное число блоков, ра-

ботающих на ВО, и нескольких резервных блоков (от 1 до 3), каждый из которых в общем случае может заменить любой из блоков ядра. Блоки, образующие ядро, работают на ВО. Специализированный блок-детектор рассогласования (ДР) сравнивает сигнал с ВО и сигналы с каждого из блоков ядра. При их несовпадении ДР выдает сигнал в схему переключения (П), которая отключает отказавший блок и подключает вместо него резервный блок без перерыва в функционировании системы.

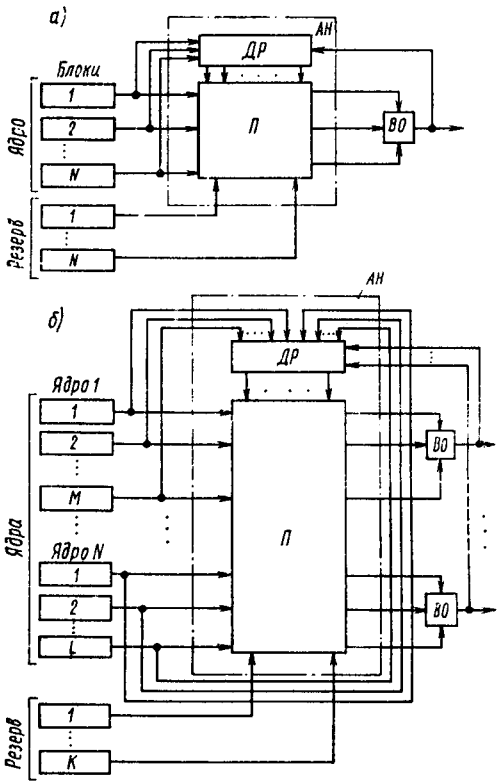


Рис. 6.4. Построение МПС с комбинированным резервированием на основе гибридного принципа

а — структура с гибридным резервированием, автоматом надежности и одним ВО; *б* — структуры с параллельно-гибридным резервированием и несколькими ВО

В состав типовой структуры системы с ПГР (рис. 6.4, *б*) входят N ядер, состоящих из M (трех и более) рабочих блоков и работающих параллельно, а также ряд резервных блоков (от одного до K). Эти ядра вводятся, например, для предотвращения перерывов в работе всей системы при отказе одного из блоков. Резервные блоки в этом случае заменяют любой из блоков каждого из N ядер и используются при этом более эффективно.

В системах с ГР и ПГР применяются следующие методы борьбы со сбоями:

а) каждый выходной элемент ДР подключается к специальному счетчику числа сигналов несовпадения, выдаваемых с ДР, до определенного порога (на практике от двух до четырех) с последующей выдачей сигнала на подключение резервного блока вместо отказавшего;

б) схема переключения П осуществляет замену рабочих блоков на резервные по сигналам несовпадения, поступающим от ДР. После того как все резервные блоки уже использованы и поступает очередной сигнал от ДР, вновь подключается блок, отказавший первым, затем вторым и т. д. Это позволяет при нали-

чии сбоев в работе основных блоков снова через цикл включить их в работу;

в) комбинация первых двух методов при различных модификациях подсчета сигналов несовпадения и постановки в очередь на замену. Безотказность наиболее слабого звена в системе с ГР и ПГР — автомата надежности, включающего в себя детектор рассогласования ДР и схему переключения П, обеспечивается методами ПР — дублированием, утроением, учетверением и выше, а также мажорированием или их комбинацией со специальными методами.

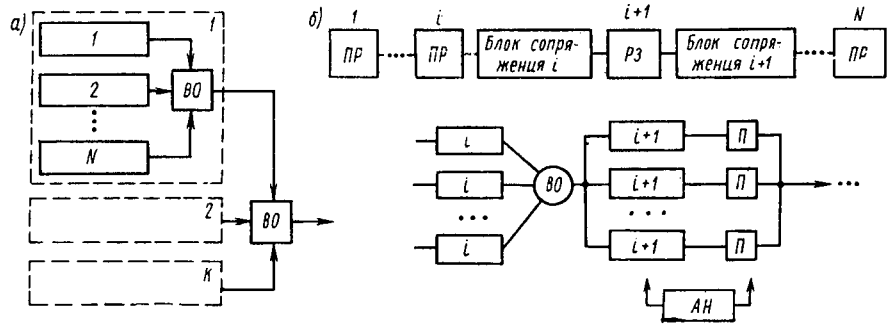


Рис. 6.5. Построение МПС с комбинированным резервированием:
а — на основе итеративного принципа; *б* — на основе смешанного принципа; П — схема переключения; АН — автомат надежности

На рис. 6.5 представлены некоторые методы КР на основе итеративного принципа: комбинации ПР и РЗ, а также более сложные методы (комбинации ГР, мажоритарного резервирования (МР) и др.). На рис. 6.5, *а* представлена структура с КР на основе итеративного принципа, в котором использован метод МР (МР), т. е. для повышения надежности каждого блока используется мажоритарный метод, а сами мажорируемые блоки также соединяются по мажоритарной схеме. Известны надежные структуры, использующие итеративный принцип резервирования РЗ (РЗ), СР (СР), структуры с МР (СР) — резервированием и др. [5, 11].

В структуре устройства с КР на основе смешанного принципа (рис. 6.5, *б*) для повышения надежности i-го блока используется ПР с ВО, а для $(i+1)$-го блока применяется РЗ. Поэтому между ними необходима установка блока сопряжения, например ВО. Блок сопряжения $(i+1)$-го блока включает в себя схемы переключения П. Отказавший блок в $(i+1)$-м ядре обнаруживается и отключается с помощью автомата надежности АН. Такой подход применен, например, в орбитальной космической станции, в которой основное оборудование системы стабилизации задубли-

ровано, а для регистра, содержащего критические параметры системы, применено тройное резервирование. Аналогичные системы военного и космического назначения также широко используют КР на основе смешанного или итеративного принципа, не допуская перерывов в работе бортовых ЭВМ, особенно при обработке ими рабочих программ.

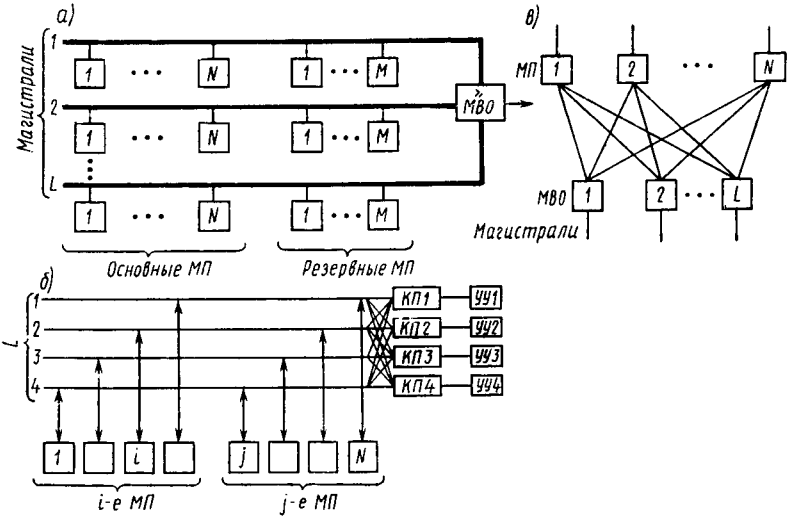


Рис. 6.6. Построение МПС с гибридным резервированием и мультиплексорными восстанавливающими органами (МВО):
а — структура МПС с *L*-магистралями и одним МВО МПС, *б* — структура МПС с коммутирующими процессорами в качестве МВО; *в* — фрагмент структуры МПС при *L* магистралях и *N* МП

На рис. 6.6, *а* приведена структура избыточной МПСУ с использованием специального мультиплексорного ВО и одного из методов ГР, которая состоит из *L* магистралей, к каждой из которых подключается *N* основных и *M* резервных однотипных МП, выполняющих разные функции. Резервные МП могут замещать каждый из рабочих МП, подключенных к соответствующей магистрали на основе метода СР.

Подключение $(N+M)$ МП к одной магистрали существенно упрощает схему переключения неисправных МП и ввода в работу резервных МП при эффективном использовании оборудования. Все *L* магистралей подключаются к одному МВО, выполняющему функции сравнения «*n* из *L*» с высокой степенью надежности.

Анализ вариантов построения избыточных структур МПСУ с ГР при различных кратностях (два, три и более) и способах резервирования МП (фиксированный или скользящий), количестве используемых магистралей, числе типов МП, применяемых в

структурах или в пределах одной магистрали, а также числе порогов срабатывания МВО показал следующее [5, 33]:

— максимальной надежностью обладают структуры однородных МПСУ с ГР, в которых применяются однотипные МП, ВО выполняют функцию «2 из 3» или «2 из 4» при кратности резервирования n, равной 6 ($n=N/S$, где N — общее число однотипных МП в избыточной структуре, S — число решаемых этими МП задач);

— при заданной кратности резервирования разнотипных МП в структуре целесообразнее увеличивать число магистралей, а не число МП в пределах одной магистрали, резервируемых по способу РЗ;

— размещение ВО на входах и выходах МП или только на выходах МП резко снижает надежность избыточных МПСУ (надежность ВО обычно ниже надежности МП);

— при кратности резервирования МП больше трех и допустимом быстродействии целесообразно использовать МВО на базе коммутирующих процессоров типа К583КП1.

В качестве примера на рис. 6.6, *б* приведена структура надежной избыточной МПСУ с ГР, где $n=4$, $L=4$, $M=0$ и каждый МВО работает на свою магистраль. При передаче информации из i-го МП в j-й МП происходит обращение к МВО, который восстанавливает информацию и выдает правильные сигналы в одну из L магистралей. МВО состоит из двух частей: а) собственно ВО, выполняющего функцию восстановления «2 из 4» на базе МП типа К583КП1; б) устройства управления ВО (при централизованном принципе управления они отсутствуют, а микропрограммы восстановления хранятся в ПЗУ). При L магистралях и N МП фрагмент структуры МПСУ с ГР имеет вид рис. 6.6, *в*. Такие структуры МПСУ с ГР позволяют осуществлять программное отключение отказавших блоков и восстановление искаженной информации.

Особенности проектирования МПСУ с использованием адаптивных восстанавливающих органов. Одним из перспективных для практического применения методов синтеза надежных систем управления является метод построения избыточных структур МПСУ с использованием серийных МЭВМ и адаптивных ВО (АВО). В таких структурах серийные МЭВМ могут выступать в качестве резервируемых блоков (РБ) с различной степенью кратности, а АВО может быть реализован программно (на базе МП или ОМЭВМ) или аппаратно (на базе СИС и БИС) и обеспечивать работу РБ системы в нагруженном, облегченном и ненагруженном режимах.

Применение избыточных МПСУ с использованием АВО требует минимума дополнительных программно-аппаратных средств без переработки имеющегося ПО серийных МЭВМ, что очень важно для пользователей.

Основными функциями АВО независимо от способа его реализации являются:

— вычисление пороговой функции (голосование);

— сравнение информации, поступившей из РБ, с восстановленной ее версией, полученной после голосования;

— подсчет числа несовпадений при сравнениях;

— сравнение числа несовпадений с заданным порогом;

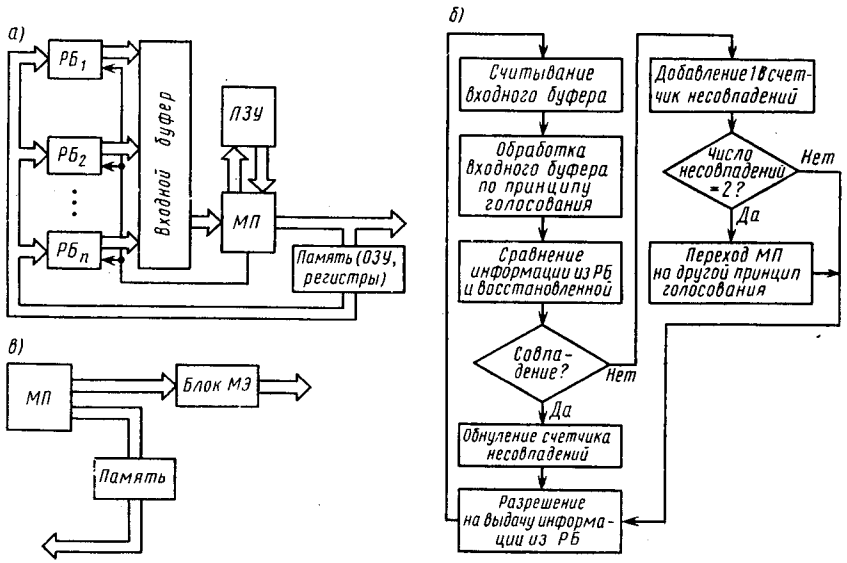


Рис. 6.7. Построение адаптивных ВО на основе МПК БИС:
а — структурная схема; *б* — алгоритм работы; *в* — модификация АВО

— изменение пороговой функции при отказе РБ;

— запуск программы восстановления информации при сбое РБ;

— поддержание синхронной работы РБ;

— хранение промежуточных результатов вычислений.

Анализ вида операций и средств, необходимых для реализации перечисленных функций, показал, что АВО можно создать на МПК БИС любой серии или даже однокристальной МЭВМ, обеспечивая при этом идентификацию отказов и сбоев с минимальными аппаратными затратами и простым программным обеспечением, реализуемым в машинных кодах конкретных МП или МЭВМ.

Наиболее простая структура АВО на МП (рис. 6.7, *а*) содержит: входной буфер, принимающий информацию из $РБ_1, ..., РБ_n$; МП, обрабатывающий по программе, хранимой в ПЗУ команд, информацию входного буфера, выполняющий все функции АВО

и формирующий выходную информацию; ОЗУ, хранящее обработанную МП информацию.

Алгоритм работы данного АВО (рис. 6.7, *б*) состоит в следующем. После формирования выходной информации МП выдает в РБ синхронизирующий сигнал, разрешающий следующую выдачу

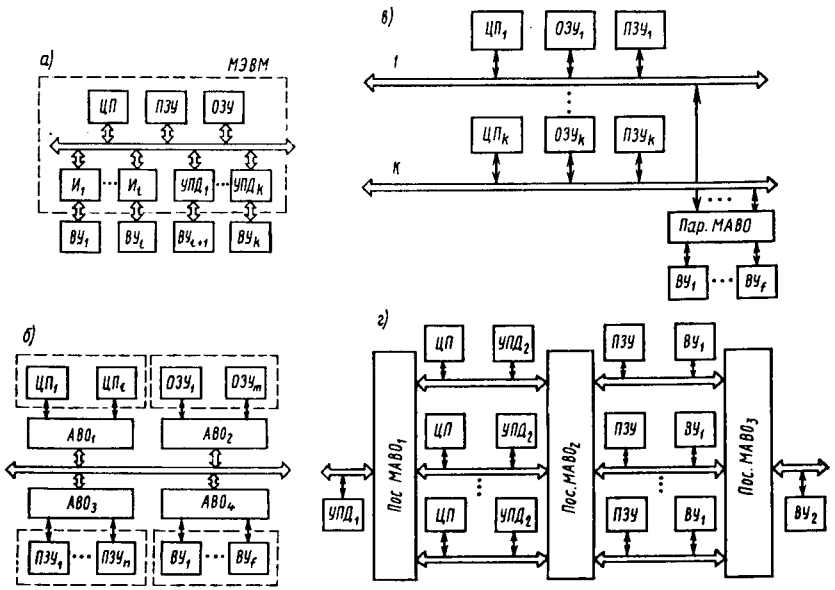


Рис. 6.8. Построение МПСУ с использованием АВО:

а — неизбыточная структура МЭВМ; *б* — избыточная структура МПСУ с использованием индивидуального АВО на каждую функциональную группу; *в* — избыточная структура МПСУ с одним параллельным магистральным АВО; *г* — избыточная структура МПСУ с тремя последовательными магистральными АВО

данных из всех РБ. Если же МП обнаруживает ошибку в информации какого-либо РБ, он посылает во все РБ сигнал о проведении процедуры восстановления. По окончании процесса восстановления МП снова проверяет совпадение информации из РБ с восстановленной. Если из какого-либо РБ вновь выдается неверная информация, то он считается неисправным и отключается, а МП переходит на новый принцип голосования.

Для программной реализации наиболее удобен алгоритм адаптации, в котором порог голосования на единицу меньше числа неотказавших блоков. Если время работы голосования недопустимо велико, то используют структуру (рис. 6.7, *в*) с установкой блока МЭ на выходе. Минимальных затрат аппаратуры при такой структуре АВО требует алгоритм адаптации «2 из $3+n$—3»→ →«2 из $3+n$—4»→ ...→«2 из 3»→1, и его следует считать наибо-

лее рациональным для реализации на МП с установкой блока МЭ на выходе.

Применение подобных структур АВО на МП по сравнению с известными решениями АВО позволяет сократить аппаратурные затраты от 2,6 раза при 4-кратном резервировании до 3,6 раза при 8-кратном резервировании систем.

Существует три варианта включения АВО в магистральную структуру МПСУ: а) применение индивидуального АВО (ИАВО) для каждой резервируемой группы блоков в отдельности; б) применение параллельного магистрального АВО (Пар. МАВО), являющегося единым для МПСУ и подключаемого к магистрали параллельно; в) применение последовательного магистрального АВО (Пос. МАВО), встраиваемого в магистраль.

Если рассмотреть магистральную структуру МЭВМ (рис. 6.8, *а*), то в ней можно выделить ЦП, ОЗУ, ПЗУ и несколько УВВ ($ВУ_1, ..., ВУ_l, ВУ_{l+1}, ..., ВУ_k$), которые подключаются к каналу МЭВМ как через пассивные интерфейсы $И_1, ..., И_l$, так и через устройства прямого доступа к памяти $УПД_1, ..., УПД_k$. При резервировании МЭВМ с помощью ИАВО (рис. 6.8, *б*) общее число АВО определяется числом резервируемых групп (ЦП, ОЗУ, ПЗУ и УВВ), при этом магистраль не резервируется. Недостатками такой структуры являются: большое число требуемых ИАВО из-за невозможности объединения в одном РБ устройств различного функционального назначения; необходимость изменения интерфейса МП и ЗУ, а также конструкции серийных МЭВМ. Такую структуру удобно применять только при резервировании внешних устройств, когда АВО заменяет схемы интерфейса.

Если кроме УВВ требуется зарезервировать остальную часть МПСУ, выступающую в качестве единого РБ, то необходимо резервировать и магистраль. Структура параллельного магистрального АВО, позволяющая это делать (рис. 6.8, *в*), не нарушает конструкции МЭВМ, но при большом числе разнотипных УВВ здесь требуются специализированные АВО.

Перечисленные недостатки устраняются при использовании структуры последовательных магистральных АВО, встраиваемых непосредственно в магистраль МПСУ (рис. 6.8, *г*). В этом случае все устройства МПСУ подразделяют на активные (требующие управления элементами интерфейса и других компонентов) и пассивные и включают АВО между ними. Кроме того, необходимо учитывать специфику работы УВВ, возможность функционирования их в нагруженном режиме, способ синхронизации, уровень надежности отдельных УВВ. Например, в структуре рис. 6.8, *г* ряд УВВ (активное устройство $УПД_1$ и пассивное устройство $ВУ_2$) не резервируется, все другие активные устройства (ЦП и $УПД_2$) подключаются к одной магистрали, а пассивные устройства $ВУ_1$ и платы ПЗУ — к другой магистрали. При любом числе устройств в МПСУ данная структура потребует не более трех МАВО, при-

чем: а) если обмен между $УПД_1$ и $ВУ_2$ не предусматривается, то их можно подключить к одной магистрали, а из $АВО_1$ и $АВО_3$ оставить наиболее быстродействующий; б) если требуется увеличить уровень резервирования, то достаточно подключить пассивные устройства к магистрали активных устройств; в) если все пассивные устройства подключить к магистрали активных устройств, то можно из системы исключить $АВО_2$, а оставить $АВО_1$ и $АВО_3$ либо вообще один из них.

В целом преимущество данного варианта структуры МПСУ перед другими состоит в том, что при использовании трех АВО все УВВ можно подключать через свои стандартные интерфейсы, менять уровень резервирования, легко расширять и модифицировать систему. Однако в ней должны выдерживаться более жесткие требования по быстродействию, особенно при временном мультиплексировании магистралей.

В общем случае проектирование МПСУ с использованием АВО состоит в следующем:

1. Анализируется структура исходной нерезервированной МПСУ, определяется число активных и пассивных устройств, возможности их работы в резерве (облегченный, ненагруженный или нагруженный режимы), учитывая разрядность шин магистрали и допустимые скорости обмена устройств с магистралью.

2. Выбирается место включения АВО в МПСУ, после чего система разбивается на резервируемые блоки.

3. Выбирают структуру АВО и тип МП, учитывая наиболее важные характеристики: разрядность, возможность ее наращивания, наличие раздельных шин ввода и вывода, микропрограммируемость, быстродействие, минимум дополнительных аппаратных затрат (для применения в АВО для МП не требуются режимы прерывания и ПДП, стековая организация).

4. Выбирают способ синхронизации резервированной МПСУ, действующей при отсутствии ошибок в РБ и при возникновении в них сбоев. Обычно при отсутствии ошибок синхронизация РБ производится с помощью программно-аппаратных средств путем посылки каждым активным РБ в АВО сигнала о начале очередного цикла голосования; выполнение каждого нового цикла голосования проводится лишь после получения от АВО подтверждения об окончании предыдущего цикла. При возникновении сбоев синхронизация РБ производится программой восстановления.

5. Рассчитывают надежность спроектированной резервированной МПСУ и сравнивают ее с нерезервированной МПСУ по ряду показателей, например по коэффициенту улучшения надежности МПСУ ($K_{уп}$), коэффициенту повышения наработки ($K_{пн}$) и др.:

$$K_{ун}=\frac{1-P_н(t)}{1-P_р(t)}=\frac{Q_н(t)}{Q_р(t)}; \quad K_{п.н}=\frac{T_р}{T_н}, \qquad (6.3)$$

где $P_н(t)$, $P_р(t)$, $Q_н(t)$, $Q_р(t)$ — вероятности безошибочной (ошибочной) работы нерезервированной и резервированной систем; $T_р$, $T_н$ — время наработки на отказ резервированной и нерезервированной систем.

Пример 6.1. В АСУ ГПС механообработки обычно применяется двухуровневая структура, на верхнем уровне которой находится мини-ЭВМ СМ1420, устанавливаемая в помещении диспетчера ГПС, на нижнем уровне — СЧПУ, выполненные на базе МЭВМ «Электроника 60М» и устанавливаемые по месту на станках с ЧПУ. В связи с высоким уровнем электрических помех в цехе необходимо рассмотреть варианты повышения надежности СЧПУ, имеющих связь с СМ-1420 через адаптеры связи.

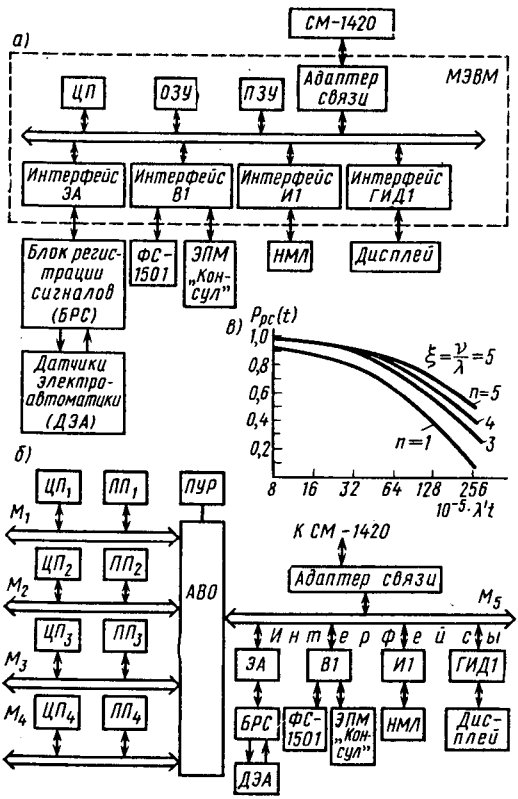


Рис. 6.9. Построение надежной МПСУ на основе АВО:

а — неизбыточная МПСУ; *б* — избыточная структура МПСУ с Пос. МАВО; *в* — зависимости $P_{рс}(t)$ с различной кратностью резервирования

В состав СЧПУ входят: ЦП; платы памяти (ПП), включающие ОЗУ и ПЗУ; интерфейс электроавтоматики (ЭА) для связи датчиков положения, состояний и перемещений исполнительных устройств; УВВ — фотосчитыватель ФС-1501, ЭПМ «Консул-260», подключаемые через интерфейс В1; НМЛ, подключаемый через интерфейс И1, а также дисплей, подключаемый через интерфейс ГИД1. Таким образом, исходная нерезервированная СЧПУ ГПС имеет вид структуры (рис. 6.9, *а*).

Рассмотрим, как наиболее эффективно повысить надежность МЭВМ нижнего уровня.

Активными устройствами в МЭВМ являются ЦП и ГИД1. Резервировать можно лишь ЦП и ПП, так как остальные УВВ служат только для загрузки программ и контроля за ходом технологического и вычислительного процесса, их трудно резервировать и синхронизировать, а такие УВВ, как дисплей или ЭПМ, вообще не могут функционировать в нагруженном режиме из-за работы на них оператора. Датчики и блок регистрации данных выполняются вместе с интерфейсом связи и не резервируются. В МЭВМ «Электроника 60М» применяется мультиплексированная выдача адресов и данных с временем передачи около 150 нс, определяющим минимальное время голосования.

Относительно выбора способа резервирования МЭВМ отметим следующее.

1. Дублирование МЭВМ связано с введением в структуру СЧПУ дополни-

тельного устройства переключения резерва, с разработкой сложного программного обеспечения. Кроме того, на время обнаружения неисправностей ход вычислительного процесса в дублированной МПСУ приостанавливается, что недопустимо при реальной эксплуатации ГПС.

2. Применение троирования МЭВМ («2 из 3») не эффективно из-за того, что при отказе одного из РБ, хотя система и функционирует верно, но оператору об этом ничего не известно, вплоть до момента отказа всей системы. Устранение данного недостатка требует введения программных процедур или схем контроля за нормальным функционированием резервируемых блоков.

3. Применение четырехкратного резервирования МЭВМ предпочтительнее троирования МЭВМ вследствие того, что при отказе одного из РБ он тут же заменяется на резервный без снижения достоверности выдаваемой информации. Для АСУ ГПС, являющейся обслуживаемой системой с наработкой на отказ МЭВМ примерно в 1000 ч, достаточно четырехкратного резервирования МЭВМ с использованием АВО: за это время оператор может заменить отказавшую МЭВМ, а с помощью пульта управления резервированием (ПУР) реконфигурировать систему.

4. Относительно места включения АВО в резервируемые МЭВМ отметим, что при резервировании ЦП, ОЗУ и ПЗУ можно организовать от одной до трех резервных групп с включением соответственно от одного до трех АВО (Пос. МАВО). Так как обмен между ГИД1 и интерфейсами УВВ (И1, В1, ЭА) не предусмотрен, то ГИД1 и нерезервируемые пассивные УВВ можно подключить к одной магистрали, исключая из системы один из АВО. Если резервировать ЦП и ПП отдельно, то конструктивно потребуется $2n+1$ каркасов, если ЦП и ПП объединить в один РБ, то нужно $n+1$ каркасов.

Окончательно выбирается резервированная структура СЧПУ (рис. 6,9 *б*), в которой используются четыре РБ, состоящих из ЦП и ПП, а все остальные устройства не резервируются. Полная реализация АВО выполнена на базе МП КР580ИК80А и других совместимых СИС и БИС. Программа функционирования АВО содержит 80 команд и занимает около 180 байт памяти.

Расчет надежностных характеристик резервированной СЧПУ при различной кратности резервирования ($n=1$, 3, 4, 5) и соотношении сбоев (ν) и отказов (λ) $\xi=\nu/\lambda=5\div10$ показал, что четырехкратное резервирование наиболее эффективно и почти не отличается от пятикратного резервирования (рис. 6.9, *в*). Коэффициент улучшения надежности резервированной системы по сравнению с нерезервированной системой составляет: 1,5—4 для четырехкратно резервированной; 1,7—4 для пятикратно резервированной; 1,2—3,5 для трехкратно резервированной системы.

§ 6.3. СИНТЕЗ ПОДСИСТЕМ ПАМЯТИ МПСУ С ПРИМЕНЕНИЕМ КОДОВОЙ И ВРЕМЕННОЙ ИЗБЫТОЧНОСТИ

Разнообразие ситуаций, возникающих при функционировании подсистем памяти МПСУ в условиях отказов и сбоев, может быть описано различными моделями каналов хранения информации [19]: а) при отсутствии или неиспользовании сведений о местоположении ошибок в ЗУ в качестве модели берут типовой двоичный канал с ошибками, в котором блоки записи и считывания совмещаются с кодером и декодером (рис. 6.10, *а*); б) при наличии сведений об ошибках при записи и считывании используют модель канала без ошибок (рис. 6.10, *б*); в) при известном местоположении отказавших разрядов, но неизвестном их истинном состоянии при считывании и декодировании применяют известную модель канала со стираниями (рис. 6.10, *в*); г) при наличии сведений о местоположении и состояниях отказавших разрядов до

записи информации применяют модель канала с дефектами (рис. 6.10, *г*); д) при возможности одновременного появления ошибок, местоположение которых неизвестно (сбоев), и ошибок с известным местоположением (дефектов) может быть применена модель канала с дефектами и сбоями (рис. 6.10, *д*) [19, 37].

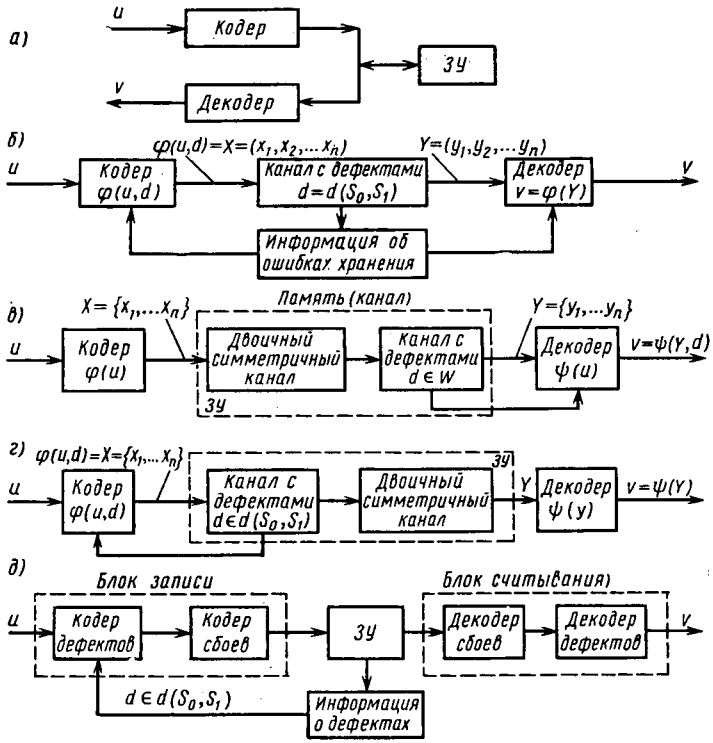


Рис. 6.10. Представление ЗУ разновидностями моделей каналов хранения:

а — модель ЗУ при неиспользовании сведений об ошибках; *б*, *в*, *г* — модели ЗУ с ошибками или дефектами и ошибками при записи и (или) считывании; *д* — модель ЗУ с дефектами и сбоями

В общем случае функционирование ЗУ может быть описано конечным автоматом вида [19]

$$A=\langle X,\ Y,\ Q,\ \varphi(X,\ Q),\ g(X,\ Q)\rangle, \qquad (6.4)$$

где Q — множество состояний $(q^j_0,\ q^j_1, ..., q^j_{N-1} \mid j=\overline{1,\ 2^N})$; $q=\{0,\ 1\}$, X — множество возможных в ЗУ операций W^1, W^0, R, S, где W^1, W^0 — запись лог. 1 (0), R — чтение, S — хранение; $Y=\{y_0,\ y_1, ..., y_{N-1}\}$ — множество выходов; $\varphi(X,\ Q)$ — функция переходов; $g(X,\ Q)$ — функция выходов, причем

$$g(X, Q)=\begin{cases} q_i{}^j, \text{ если } X=R \ (i \text{ — адрес чтения}), \\ \Theta, \text{ если } X\neq R \text{ (значение } \Theta \text{ различно для каждого типа ЗУ}). \end{cases}$$

Для защиты ЗУ от ошибок широко применяются различные корректирующие (n, k)-коды. В последнее время создан ряд новых кодов, предназначенных специально для коррекции ошибок в ЗУ — линейные (групповые), аддитивные, групповые и негрупповые аддитивные, совершенные, модульные коды, коды для исправления ошибок и дефектов и др. [19]. Совершенные коды, например, могут содержать при заданном кодовом расстоянии d минимум избыточных разрядов, но тем не менее требуют повышенной сложности реализации кодирующих и декодирующих устройств. Отсюда следует, что при выборе кодов необходимо учитывать все факторы, способствующие минимальной суммарной избыточной реализации.

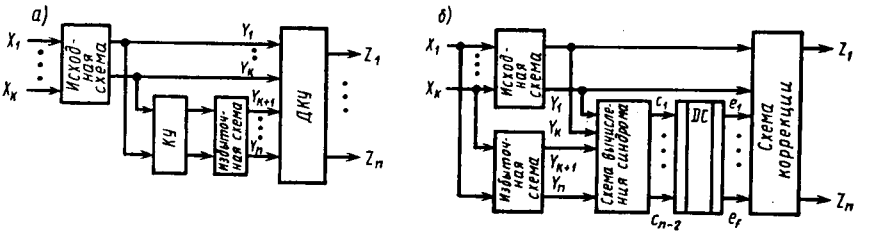


Рис. 6.11. Типовые структуры, реализующие корректирующие групповые коды с обнаружением:

а — с обнаружением ошибок; *б* — с исправлением ошибок; КУ (ДКУ) — кодирующее (декодирующее) устройство

Обобщенные типовые структуры, рекомендуемые для аппаратной реализации корректирующих групповых кодов, обычно содержат (рис. 6.11, *а, б*): исходную неизбыточную схему; кодирующее устройство (КУ), вычисляющее значения избыточных разрядов, или схему вычисления синдрома кода; схему корректора или декодирующее устройство (ДКУ), обеспечивающее реализацию заданного множества выходных сигналов исходного устройства с исправлением в них ошибок, кратность которых не превышает исправляющей способности используемого корректирующего кода; дешифратор.

Сложность реализации отдельных схем ДКУ зависит от выбора (n, k)-кода и кратности исправляемых ошибок. Сложность реализации схемы вычисления синдрома кода зависит от числа единиц, содержащихся как в проверочной матрице $H(\eta_н)$, так и в порождающей матрице $G(\eta_о)$. При наилучшем выборе (n, k)-кода соблюдается условие, что

$$\eta_н \geqslant r(d-r)+n; \quad \eta_о \geqslant rd.$$

Здесь n — число символов кода, k — число информационных разрядов, $(n-k)$ — число проверочных разрядов, d — заданное кодовое расстояние, σ — кратность ошибок в кодовых комбинациях.

Сложность схемы дешифратора в общем случае определяется кратностью исправляемых ошибок. Число выходов дешифратора равняется $\sum_{i=1}^{\sigma} \binom{r}{i}$, а его общая сложность $n_{\text{дш}}$ составляет

$$n_{\text{дш}}=(n-r-1)\sum_{i=1}^{\sigma}\binom{r}{i}+(n-r).$$

Сложность схемы корректора определяется числом сумматоров по mod 2.

Применение для ЗУ разрядностью k корректирующих кодов, обнаруживающих или исправляющих ошибки максимально возможной кратности (т. е. $\sigma=k$), требует введения в них следующего числа проверочных разрядов; $r\geqslant 2k$ для кода, обнаруживающего ошибки кратности k; б) $r\geqslant 4k$ для кода, исправляющего ошибки кратности k.

Синтез МПСУ с использованием кодов, исправляющих ошибки, и временной избыточности. Из-за электрических помех, воздействий α-частиц и других факторов в ЗУ возникают случайные ошибки с изменением двоичных символов ($0\to 1$, $1\to 0$), местоположение которых неизвестно, а их число зависит от интеграции элементов. Математическая модель такого ЗУ может иметь вид рис. 6.10, *в*, а связь выхода Y_i и входа X_i канала описывается соотношением

$$Y_i=\begin{cases}\xi_i, & \text{если } i\in S_0,\\ 1\vee\vee\xi_i, & \text{если } i\in S_1,\\ X_i\vee\vee\xi_i, & \text{если } i\in\overline{S_0\vee S_1},\end{cases}\qquad(6.5)$$

где ξ_i — независимые случайные величины (сбои), $\xi_i\in\{0, 1\}$ с вероятностями $q=1-p$ и p соответственно; S_0, S_1 — подмножества, характеризующие отказы ячеек ЗУ в виде переходов $1\to 0$ и $0\to 1$, $i=\overline{1, n}$.

Для повышения надежности ЗУ, например, широко используются модифицированные коды Хэмминга, исправляющие одиночные и обнаруживающие двойные ошибки. Применение в системах памяти таких корректирующих кодов позволяет увеличить ВБР ЗУ на 5—6 порядков. При высоких требованиях к надежности ЗУ необходимы более мощные средства, чем код Хэмминга, с несколько большей временной и аппаратной избыточностью. Например, реализация в ЗУ возможности исправления двух случайных ошибок уже требует примерно двойного роста избыточности и усложнения схемы декодирования.

В общем случае синтез ЗУ с кодовой избыточностью заключается в расширении его алфавита состояний до образования (n, k)-корректирующего кода с заданной обнаруживающей или исправляющей способностью и организации связи выхода ЗУ с его входом в соответствии с соотношением (6.5).

Если вероятность перехода $0 \to 1$ очень мала ($p=0$), то при записи информации в ЗУ можно применить алгоритм с использованием корректирующего кода, обнаруживающего ошибки, который позволяет при считывании автоматически исправлять искаженные кодовые слова. Если же $p \neq 0$, то необходимо применение кодов, исправляющих ошибки. При условии, что вероятность одновременного возникновения двух (и более) ошибок в n-разрядном считываемом слове из ячейки ЗУ с адресом A_i близка к нулю, рекомендуется применять типовую структуру (рис. 6.12, *а*), работающую следующим образом: при считывании информации из ЗУ выходное n-разрядное слово $\hat{y}$ подается в схему вычисления синдрома (вектора ошибки) $S=YH^T$ (H — проверочная матрица $(n-r)\times n$-используемого кода). При возникновении случайных ошибок оказывается, что синдрома S достаточно для их классификации, определения местоположения и исправления σ-кратной ошибки $S=YH^T=(Y-X)H^T=EH^T$, где $E=(E_1, E_2, \ldots, E_n)$ — вектор ошибок, а $XH^T=0$.

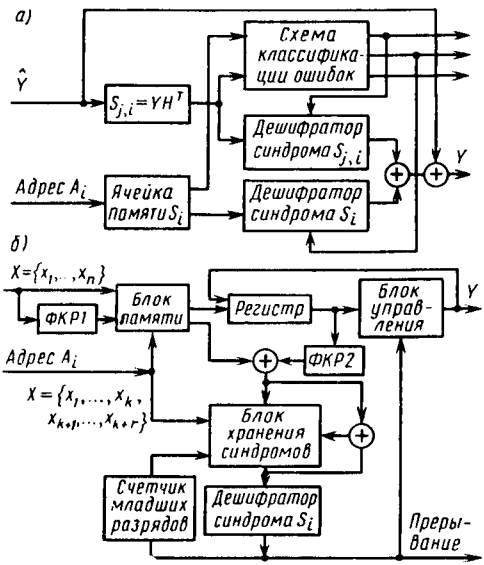


Рис. 6.12. Структурные схемы ЗУ с кодовой избыточностью:

а — схема ЗУ с коррекцией одиночной ошибки в кодовых словах; *б* — схема ЗУ с исправлением одиночной ошибки и многократных дефектов

Если ошибка обнаруживается в первый раз, то соответствующий синдром S_i запоминается в какой-либо ячейке ЗУ. При последующем обнаружении ошибки в той же позиции ($S_{j,i}=S_i$) ошибка классифицируется как отказ. Если появляется ошибка в другой позиции ($S_{j,i}\neq S_i$), то синдром, соответствующий этой ошибке, определяется как $S_{j,i}\vee\vee S_i=S_j$. Итоговая последовательность ошибок, непосредственно используемая для коррекции считанного слова Y ЗУ, равна $E_{ji}\vee\vee E_i=E_j$. Сложение по mod 2 полученного вектора ошибки E_j с искаженным выходным словом

$\hat{Y}$ и хранимым в памяти вектора E_i позволяет скорректировать значение выходного слова: $\hat{y} \vee \vee E_j \vee \vee E_i = Y$.

Пример 6.2. Пусть кодирование записываемой в ЗУ информации выполняется кодом (7,4), исправляющим одиночную ошибку, с проверочной матрицей

$$H = \left\| \begin{matrix} 0\,0\,0\,1\,1\,1\,1 \\ 0\,1\,1\,0\,0\,1\,1 \\ 1\,0\,1\,0\,1\,0\,1 \end{matrix} \right\|.$$

Допустим, что в записанном в ячейку ЗУ с адресом A_i слове $X_1 = (01\bar{1}1100)$ появился отказ в третьем разряде с переходом 1→0. Тогда при его считывании получим слово $\hat{Y} = (0101100)$. Синдром, соответствующий этой ошибке, имеет вид $S_i = 011$, что позволяет исправить одиночную ошибку в третьем разряде. Этот синдром запоминается.

Допустим, что при следующем обращении к ЗУ произошел случайный сбой информации в шестом разряде. Тогда при считывании кодового слова X_1 получим $\hat{Y} = (0101110)$.

При вычислении синдрома ошибки определим $S_{j,i} = 101$. Если бы использовали синдром $S_{j,i} = 101$, то пятый разряд кодового слова был бы ложно скорректирован. Для правильной коррекции двойной ошибки в 3-м и 6-м разрядах кодового слова определим синдром $S_j = S_{j,i} \vee \vee S_i = 011 \vee \vee 101 = 110$. Совместное применение синдромов $S_i = 011$ и $S_{ji} = 110$ позволяет правильно исправить двукратную ошибку в кодовом слове ЗУ.

Для исправления одиночных случайных ошибок ($\sigma_1 = 1$) и многократных дефектов ($\sigma_2 = n$) рекомендуется применять типовую структуру ЗУ (рис. 6.12, *б*), содержащую формирователи контрольных разрядов ФКР1 (при записи) и ФКР2 (при считывании), блок памяти, регистр временно́го хранения, схемы сложения по mod 2, блок хранения синдромов, дешифратор синдрома S_i, блок исправления ошибок и счетчик младших разрядов адреса A_i. Данная схема ЗУ работает следующим образом. При поступлении на вход ЗУ информационного слова $X_i = (x_1, ..., x_k)$ в ФКР1 вычисляются контрольные разряды $x_{k+1}, ..., x_{k+r}$ и слово $\{x_1, ..., x_k, x_{k+1}, ..., x_{k+r}\}$ записывается в блок памяти по адресу A_i. В режиме считывания данных слово из k информационных разрядов $y_1, ..., y_k$ поступает в регистр временного хранения и в ФКР2. Вычисленные в ФКР2 контрольные разряды сравниваются по mod 2 с принятыми

$$S_i = (\hat{y}_{k+1}, \hat{y}_{k+2}, ..., \hat{y}_{k+r})_i \vee \vee (y_{k+1}, y_{k+2}, ..., y_{k+r})_i.$$

На блок хранения синдромов поступают старшие разряды адреса A_i, а младшие разряды адреса определяет счетчик младших разрядов A_i по выходу дешифратора синдрома S_i. Если ошибок нет ($S_i = 0$), то цикл чтения данных заканчивается. Если же $S_i \neq 0$, то с помощью дешифратора синдрома S_i, блока исправления ошибок, регистра и ФКР2 последовательно исправляются σ_2-кратные дефекты до тех пор, пока не будет $S_i = 0$.

Относительная сложность реализации ЗУ по данному методу согласно рис. 6.2 составляет

$$\alpha = \frac{n_A}{n_{KA}} = \frac{M^2 k}{(M+\sigma_2) r}.$$

Величина σ_2 определяется исходя из заданных требований надежности и при организации емкости ЗУ, равной MLk, σ_2=const. Доля ЗУ, необходимого для коррекции ошибок при использовании кодовой и временной избыточности, зависит от емкости БИС ЗУ (M) и отношения k/r числа информационных разрядов кодовой последовательности k к количеству проверочных разрядов r. Известно, что величина k/r больше для кодов, обнаруживающих ошибки, по сравнению с кодами, исправляющими ошибки.

Синтез ЗУ МПСУ с использованием кодов, обнаруживающих ошибки, и временной избыточности. При отсутствии в ЗУ МПСУ неисправностей вектор ошибки E в выходном слове ЗУ можно определить путем сравнения по mod 2 каждого записываемого $X=\{x_1, ..., x_{N-1}\}$ и считываемого $Y=\{y_1, ..., y_{N-1}\}$ слова без вычисления и дешифрирования синдрома, т. е. $E_i=\{X\}\vee\vee\{Y\}$.

Таблица 6.1. **Состояния запоминающей ячейки при наличии ошибок**

$d(S_0, S_1)$	D_3	e_i
0	0	$\tilde{e}_i$
0	1	e_{0i}
1	0	e_{1i}
1	1	$\tilde{e}_i$

Учитывая специфику работы двухстабильных запоминающих ячеек (ЗЯ), являющихся основой многих ЗУ, составляющие вектора ошибки E могут принимать одно значение из множества $e_i=\{e_{0i}, e_{1i}, \tilde{e}_i\}$. Если обозначить характер неисправности ячейки через $d(S_0, S_1)\in\{0, 1\}$, двоичный сигнал, записываемый в ячейку, — через D_3, то $e_i=d(S_0, S_1)\vee\vee D_3$ (табл. 6.1) и значения e_{0i} и e_{1i} получаются при несовпадении неисправности ЗЯ и вида D_3, а значение $\tilde{e}_i$ — при их совпадении ($\tilde{e}_i=d(S_0, S_1)\vee\vee D_3=0$).

Рассмотрим некоторые из наиболее эффективных способов обнаружения ошибок в работе ЗУ и построения надежных МПСУ на их основе.

Если при записи данных в ЗУ соблюдается условие отсутствия составляющих $\tilde{e}_i$, а при последующем считывании требуется произвести коррекцию информации из неисправной ЗЯ, то этого можно достичь следующим путем: а) обнаружить ошибку в ЗЯ; б) записать искаженное выходное слово $\hat{Y}_i$ неисправной ячейки с адресом A_i в заведомо исправную ЗЯ с адресом A_j; в) записать и считать из ЗЯ A_i единичный вектор $1=\{11...1\}$, а затем результат инвертировать и сложить его по mod 2 с содержимым ЗЯ A_j; г) записать и считать из ЗЯ A_i нулевой вектор $0=\{00...0\}$, а затем результат считывания сложить по mod 2 с ранее скорректированным значением искаженного слова $\hat{Y}_i$: $Y_i=\hat{Y}_i\vee\vee e_{0i}\vee\vee e_{1i}$. При равновероятных событиях ($e_{0i}=e_{1i}=0{,}5$) достоверность счи-

тываемой информации по данному способу повышается вдвое по сравнению с известными методами.

Если же при записи информации не соблюдается условие отсутствия составляющих $\tilde{e}_i$ и требуется осуществить коррекцию искаженного выходного слова $\tilde{Y}_i$ при его считывании из неисправной ЗЯ A_i, то для этого необходимо: а) записать входное слово

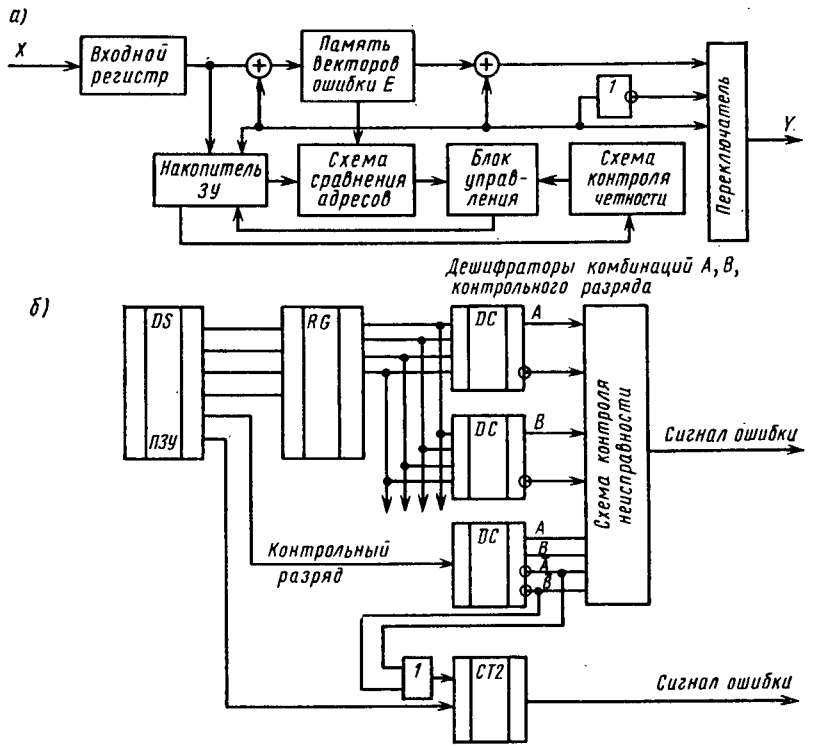


Рис. 6.13. Структурные схемы ЗУ с использованием кодов, обнаруживающих ошибки при временной избыточности:
а — схема ЗУ с контролем четности; *б* — схема надежного ПЗУ

X_i в ЗЯ по адресу A_i, а затем считать слово Y_i из этой же ячейки; б) вычислить вектор ошибки $e_i = X_i \bigvee \bigvee Y_i$; если $e_i = 0$, то сформировать признак четности в дополнительном разряде x_{iN}; если же $e_i \neq 0$ и содержит хотя бы одну «1», то входное слово X_i инвертировать ($X_i \to \bar{X}_i$), а вектор e_i вместе с адресом A_i записать в ЗУ по адресу A_j; в) произвести считывание Y_i и вычислить вектор $\tilde{e}_i = X_i \bigvee \bigvee \bar{Y}_i$; если $\tilde{e}_i = 0$, то данные в ячейке ЗУ по адресу A_j стереть, а в дополнительный разряд x_{iN} записать признак нечетности; г) при считывании информации из ЗУ вычислить синдром $S_i =$

$=Y_iH^T$; если $S_i=0$, то выходное слово Y_i выдать на выход ЗУ; если $S_i\neq 0$, то осуществить сравнение адреса A_i считываемого слова $\hat{Y}_i$ с содержимым ячеек с адресами A_{j+n} хранения векторов ошибки e_{i+n}. Если адрес A_i есть в области A_{j+n}, то $Y_i=\hat{Y}_i$, если же адреса A_i в области A_{j+n} не существует, то $Y_i=\bar{\hat{Y}}_i$. Типовая структура ЗУ, реализующего данный алгоритм, приведена на рис. 6.13, *а*.

Можно обойтись и без операции инвертирования $X_i\to\bar{X}_i$ при $e_i\neq 0$, $i=\overline{1,\ (N-1)}$. При этом после выполнения п. а), б) предыдущего алгоритма достаточно осуществить следующее: если $e_i=0$ ($e_i\neq 0$), то в дополнительном разряде x_{iN} ЗУ сформировать признак четности (нечетности) и произвести запись e_i и A_i в ЗУ по адресу A_j. При считывании информации из ЗУ вычислить синдром $S_i=Y_iH^T$: при $S_i=0$ выходное слово выдается на выход ЗУ; при $S_i\neq 0$ $Y_i=\hat{Y}_i\bigvee\bigvee e_i$, где e_i выбирается из ЗУ по адресу A_{j+n}.

В целом второй способ обладает меньшей временно́й избыточностью, но требует большего объема аппаратуры и большего числа ЗЯ для хранения e_i и A_i из-за неиспользования неисправной ячейки ЗУ для хранения X_i.

Для сохранения однородности структуры ЗУ, состоящей из триггеров, операцию сравнения по mod 2 в ЗУ рекомендуется осуществлять не на элементах сравнения, а с помощью триггеров с информационными и счетными входами, когда на информационный вход подается входное слово X_i, а на счетные — выходное Y_i.

При синтезе надежных ПЗУ с использованием кодовой и временно́й избыточности необходимо учитывать следующие особенности: а) ПЗУ работает лишь в режиме считывания информации, его математическая модель имеет вид $M=\{Q,\ Y\}$ и для минимизации аппаратных затрат целесообразнее вводить не дополнительные проверочные разряды, а кодировать информационные разряды; б) множество состояний выходных слов

$$Y=\{y_0,\ y_1,\ldots,\ y_i\ldots,\ y_{N-1}\}$$

на отдельных участках программы

$$(Y_i,\ Y_{i+1},\ldots,\ Y_{i+j}),\quad i=\overline{0,\ N-1},\ j=\overline{i,\ N-1},$$

принимает не произвольные, а строго определенные значения.

Если закодировать все выходные слова ПЗУ

$$y_k=(y_0,\ y_1,\ldots,\ y_{N-1}),\quad k=\overline{i,\ i+j},$$

и сравнить их по mod 2, то при их считывании на выходах схем сравнения наиболее часто будут встречаться две кодовые комбинации: $A=\{a_0,\ a_1,\ldots,a_{N-1}\}$ и $B=\{b_0,\ b_1,\ldots,b_{N-1}\}$. Если в дополнительный разряд N выходных слов Y_i и Y_{i+1} записать комбинацию $C=\{11\}$ при появлении кодовой комбинации A или комбинацию $D=\{00\}$ при появлении кодовой комбинации B, а также $E=\{01\}$

или $F=\{10\}$ при отсутствии комбинаций A или B, то при возникновении неисправности в ПЗУ будут выполняться условия

$$AD \vee AE \vee AF \vee \bar{A}C \vee BE \vee BF \vee BC \vee \bar{B}D = 1.$$

ПЗУ с реализацией данного способа контроля (рис. 6.13, *б*) обладает по сравнению с известными методами более чем вдвое лучшими характеристиками по быстродействию и достоверности работы и содержит: а) регистр, счетные входы которого подключены к входным шинам блока постоянной памяти, хранящего микропрограммы; б) два дешифратора кодов комбинаций A и B микропрограмм; в) дешифратор кода входной шины контрольного разряда; г) элемент ИЛИ; д) счетчик, выход которого соединен с выходной шиной устройства. Кроме того, здесь имеется также схема контроля условий неисправности, на выходе которой фиксируется сигнал ошибки. Введение в ПЗУ некоторой аппаратной избыточности позволяет обнаруживать сигналы ошибки, используя их в качестве сигналов прерывания для повторения фрагмента микропрограммы, приостановления процесса вычислений или перехода на выполнение других фрагментов микропрограммы.

Синтез ЗУ МПСУ на основе метода двойного кодирования информации. При синтезе надежных ЗУ, обеспечивающих коррекцию сбоев и дефектов ЗЯ с вычислением синдрома ошибок при считывании информации, предполагается, что для ЗУ время считывания ($t_{сч}$) может значительно превышать время записи ($t_{зп}$), т. е. $t_{зп} \ll t_{сч}$. Однако если наложить ограничение, что $t_{зп} \gg t_{сч}$, то скорректировать сбои и дефекты ЗЯ можно и в режиме записи путем применения *двойного кодирования информации,* обеспечивающего раздельную коррекцию дефектов и сбоев. При этом ошибки из-за дефектов устраняются, как правило, путем применения аддитивного кода, а ошибки из-за сбоев — путем использования линейных кодов (в частности, кода Хэмминга).

Типовая структура отказоустойчивого ЗУ с двойным кодированием информации (рис. 6.14) содержит накопитель, адресный блок, входной и выходной регистры, кодер и декодер, ПЗУ для хранения семи 16-разрядных согласующих кодовых слов матрицы, устройство управления ПЗУ, блок сумматоров по mod 2 и ИЛИ.

В *режиме записи* входные слова $X=\{x_1, ..., x_8\}$ подаются на вход кодера. В ЗУ используется систематический модифицированный код Хэмминга (16, 11) с 11 информационными и 5 проверочными разрядами, причем первые 3 информационных символа кода (a_1, a_2, a_3) — служебные (для идентификации слов, согласующихся с дефектами ЗУ).

Кодер формирует проверочные символы (y_1, y_2, y_3, y_4, y_5), составляющие укороченный модифицированный код Хэмминга при условии, что $a_1=a_2=a_3=0$. Сформированное полное слово $X_0=$ $=(a_1a_2a_3x_1 \ldots x_8y_1 \ldots y_5)$ из входного регистра записывается в на-

копитель по заданному адресу A_i, а затем из этой ячейки считывается слово X_0, которое попадает в выходной регистр и далее сравнивается со словом X_0 посредством сумматоров по mod 2 и 16-входовой схемы ИЛИ ($e_i = X_0 \vee \vee \bar{X}_0$). При совпадении этих слов $e_i = 0$ и цикл записи заканчивается. Если же $e_i \neq 0$, то устройство управления (УУ) ПЗУ формирует команду повторной записи. При этом в ПЗУ считывается первое согласующее кодовое

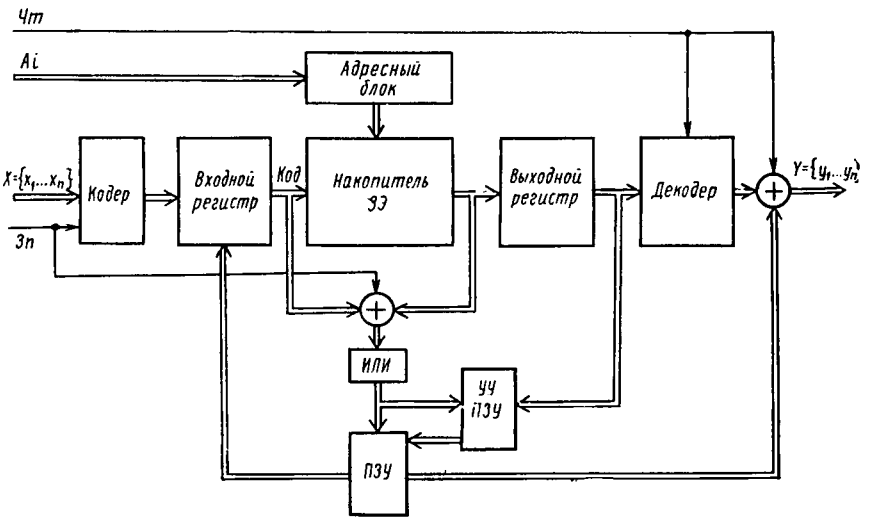


Рис. 6.14. Реализация ЗУ по методу двойного кодирования информации

слово, которое подается на счетные входы входного регистра, при этом формируется новое кодовое слово $X_1 = X_0 \vee \vee C_1$ с записью его в накопитель по старому адресу. Затем операция считывания повторяется аналогично вышеописанной, отличаясь тем, что при $e_{i+1} \neq 0$ из ПЗУ считывается следующее кодовое слово, а слово X_0 во входном регистре восстанавливается.

Процесс перезаписи и считывания продолжается до полного согласования записываемой в накопитель по адресу A_i информации с дефектными ЗУ, при этом вектор ошибок $e_{i+j} = 0$. Согласовать информацию не удается только при наличии в ячейке с заданным адресом трех (и более) дефектных ЗЭ, однако об этом формируется сигнал наличия дефектов высокой кратности.

Для исключения операций восстановления во входном регистре слова X_0 перед каждой следующей попыткой согласования записываемой информации с дефектами в ЗЯ накопителя можно в ПЗУ вместо второй и последующих строк матрицы **C** согласования кодовых слов записать матрицу **C′** со строками C_1, $C_1 \vee \vee C_2$, ..., $C_6 \vee \vee C_7$. Тогда, если во входном регистре содержится слово

$X_0 \vee\vee C_1$ и требуется записать новое слово $C_2 = C_1 \vee\vee C_2$ из ПЗУ, то получится слово $X_0 \vee\vee C_1 \vee\vee (C_1 \vee\vee C_2) = X_0 \vee\vee C_2$, что эквивалентно восстановлению в регистре слова X_0:

$$\mathbf{C}=\left\|\begin{array}{ccc|c} a_1 \ldots a_3 & x_1 \ldots x_8 & y_1 \ldots y_5 & \\ 001 & 01011011 & 11101 & C_1 \\ 010 & 00110111 & 11110 & C_2 \\ 011 & 11111111 & 00011 & C_3 \\ 100 & 00000000 & 11111 & C_4 \\ 101 & 10101101 & 00100 & C_5 \\ 110 & 11001110 & 01000 & C_6 \\ 111 & 11110000 & 10000 & C_7 \end{array}\right\|;$$

$$\mathbf{C}'=\left\|\begin{array}{ccc|c} 001 & 01011011 & 11101 & C_1 \\ 011 & 01101100 & 00010 & C_1 \vee \vee C_2 \\ 001 & 11001000 & 11101 & C_2 \vee \vee C_3 \\ 111 & 11111111 & 11100 & C_3 \vee \vee C_4 \\ 001 & 10101101 & 11011 & C_4 \vee \vee C_5 \\ 011 & 01100011 & 01100 & C_5 \vee \vee C_6 \\ 001 & 00111110 & 11000 & C_6 \vee \vee C_7 \end{array}\right\|$$

В *режиме считывания* слово из накопителя поступает в выходной регистр. Декодер модифицированного кода Хэмминга осуществляет исправление одиночных сбоев и обнаружение двойной ошибки. В случае одинарной случайной ошибки код $a_1 a_2 a_3$ согласующего слова C_i, использованного при втором кодировании для записи в накопитель слова $X_0 \vee\vee C_i$, поступает в УУ ПЗУ и по нему из ПЗУ считывается слово C_i, которое поступает на вход выходных сумматоров по mod 2 и формируется сумма $(X_0 \vee\vee C_i) \vee\vee C_i = X_0$. При возникновении случайных ошибок кратности 2 и выше декодер формирует сигнал о их наличии.

Применение на практике метода двойного кодирования с раздельной коррекцией случайных ошибок и дефектов позволяет создавать отказоустойчивые ЗУ с минимальной аппаратурной избыточностью (менее 1% общих затрат оборудования). Данный метод может быть программно реализован в условиях естественной временной избыточности.

Сравнительный анализ эффективности применения рассмотренных выше методов избыточности в подсистемах памяти МПСУ показал следующее [19, 37]:

— методы двойного кодирования информации, а также вычисления и хранения синдромов повторных ошибок гораздо эффективнее методов резервирования ЗУ в виде мажоритарных структур с однократной связью и традиционного применения корректирующих кодов;

— по величине средней наработки на отказ 8-разрядные ЗУ, построенные на базе методов двойного кодирования или вычисления и хранения синдромов повторных дефектов, в 5—7 раз превышают наработку на отказ аналогичных неизбыточных ЗУ (при равной абсолютной надежности схем электронного обрамления);

— для тех же условий коэффициент повышения наработки на отказ изменяется в пределах от 1 до 10 (при использовании БИС ЗУ емкостью до 1 К) и от 100 до 1000 (при использовании БИС ЗУ емкостью 4-64 К);

§ 6.4. СИНТЕЗ НАДЕЖНЫХ ЗУ МПСУ С ПРИМЕНЕНИЕМ АППАРАТУРНОЙ И ВРЕМЕННОЙ ИЗБЫТОЧНОСТИ

Более эффективных решений при проектировании надежных МПСУ можно достичь путем введения смешанной (аппаратурной и временно́й) избыточности, синтеза средств, адаптирующихся к возникающим неисправностям и сбоям. Рассмотрим некоторые из этих методов.

Синтез ЗУ, адаптирующихся к возникающим неисправностям. Применение чисто аппаратурной избыточности для коррекции ошибок в ЗУ с введением для каждой ЗЯ или их группы определенного числа резервных ячеек при объемно-полном или объемно-неполном резервировании требует значительного увеличения аппаратурных затрат.

Более рациональными способами являются:

1. Применение для каждых двух рабочих ЗУ одного резервного накопителя с записью в него информации, получаемой от сравнения по mod 2 данных, содержащихся в ячейках основных ЗУ с одинаковыми адресами. При отказе одного из ЗУ информация получается путем обращения к исправному ЗУ и резервному. Исправление информации производится путем сложения выбранных слов по mod 2.

2. Использование резервного ассоциативного накопителя (АсН), ячейки которого резервируют отказавшие ячейки рабочего ЗУ. Для этого в признаковой части АсН записываются адреса отказавших ячеек ЗУ, а в информационной части — их содержимое. При обращении отказавшей ячейки ЗУ по признаковой части АсН происходит блокирование ЗУ и обращение к информационной части АсН. Возможно также замещение не всей, а лишь части ячейки, содержащей отказавший ЗЭ. При таком подходе в условиях малого числа отказов в рабочем ЗУ резервное ЗУ практически не используется, а при числе отказавших ячеек, превышающих объем резервного ЗУ, происходит отказ рабочих ЗУ.

Значительного повышения надежности добиваются созданием ЗУ, адаптирующихся к возникающим неисправностям, с изменением режима их работы на основе аппаратурной и временно́й избыточности. Рассмотрим некоторые из типовых решений.

Типовая структура одного из таких ЗУ (рис. 6.15, *а*) функционирует по следующему принципу: в ЗЯ по адресу A_i производится запись слова $X_i=\{x_0, ..., x_{N-1}\}$, а затем из этой же ячейки считывается выходное слово $Y_i=\{y_0, ..., y_{N-1}\}$ с последующим вычислением схемой по mod 2 вектора ошибки $e_i=X_i\vee\vee Y_i$. Если ошибки

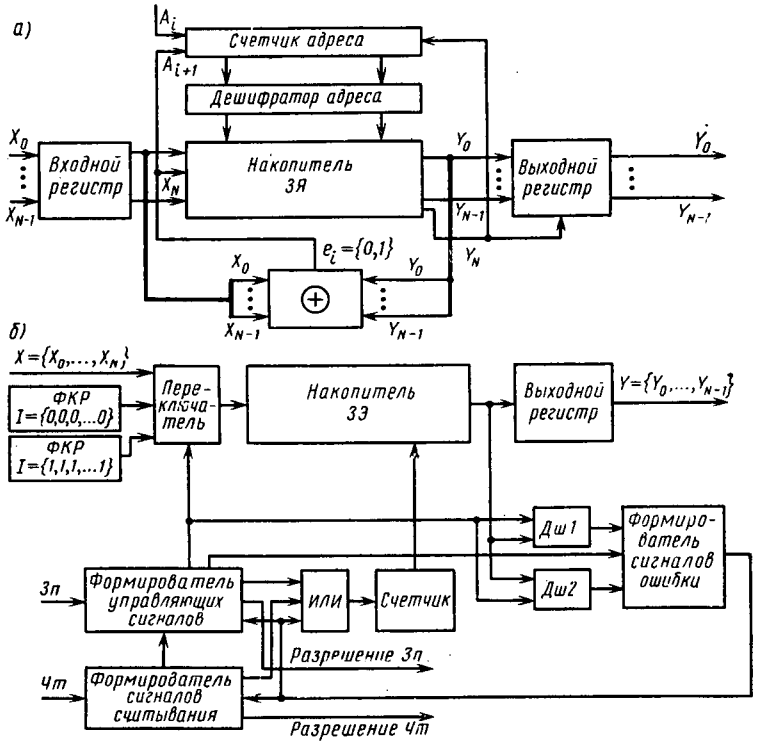


Рис. 6.15. Реализация ЗУ с использованием аппаратурной и временной избыточности:
а — ЗУ с дополнительным разрядом; *б* — ЗУ с формирователями кодов

нет ($e_i=0$), то в дополнительный разряд ЗУ записывается $x_{iN}=0$ и происходит переход к ЗЯ с адресом A_{i+1}. Если же ошибка есть ($e_i=1$), то в дополнительный разряд ЗУ записывается $x_{iN}=1$, происходит переход к ЗЯ с адресом A_{i+1}, в нее записывается то же слово X_i и производится вычисление $e_{i+1}=X_i\vee\vee Y_i$ с повторением вышеописанных процедур.

Информация считывается из ЗУ с учетом значения дополнительного разряда y_N. Если $y_{iN}=0$, то считываемое слово выдается на выход ЗУ, а если $y_{iN}=1$ (означающее, что считываемое слово Y_i находится по адресу A_{i+j} и его необходимо искать в следующих ячейках), то выходной регистр получает сигнал запрета

на выдачу информации. Достоинством такого ЗУ является малая аппаратурная избыточность при эффективном использовании емкости накопителя и исправных ЗЯ.

Известна структура ЗУ (рис. 6.15, *б*), обеспечивающая согласование разрядов кодового слова $X_i=\{x_1, ..., x_{N-1}\}$ с дефектом ЗЯ $d(S_0, S_1)$ по адресу A_i без использования дополнительного разряда, при этом используют условие

$$X_i=\begin{cases} x_i, & \text{если } i=\overline{S_0 \vee S_1}, \\ 0, & \text{если } i \in S_0, \\ 1, & \text{если } i \in S_1. \end{cases}$$

Принцип работы такого ЗУ состоит в выполнении последовательности операций записи в накопитель, последующего считывания и вычисления вектора ошибок ($e_{0i}=0_i \vee \vee \hat{0}_i$) сначала для кодов $0=\{00...0\}$ и $\hat{0}=\{\hat{0}\hat{0}...\hat{0}\}$ и затем для $e_{1i}=1_i \vee \vee \hat{1}_i$ — для кодов $1=\{11...1\}$ и $\hat{1}=\{\hat{1}\hat{1}...\hat{1}\}$, формируемых с помощью ФК. Если $e_{0i}=e_{1i}=0$, то слово $X=(x_1, ..., x_n)$ записывается в ЗЯ по адресу A_i. Если же $e_{0i}=1$, $e_{1i}=1$, то адрес ЗЯ увеличивается на 1 (A_{i+1}) до выполнения условия $e_{0i}=e_{1i}=0$. При считывании информации из ЗУ аналогично режиму записи производится проверка ЗЯ на условие $e_{0i}=e_{1i}=0$, обеспечивая на выходе ЗЯ выдачу истинной информации.

Сложность реализации структур вышеописанных ЗУ примерно равноценна.

Кроме аппаратно-временно́й избыточности, в ЗУ может использоваться и чисто временна́я избыточность, производимая по всем компонентам — входам $\{X\}$, выходам $\{Y\}$, состояниям $\{Q\}$, функциям переходов и выходов. *Временная избыточность* в ЗУ может осуществлять коррекцию сбоев путем:

n-кратной записи и считывания кодовых слов X_i, Y_i; вычисления векторов ошибок $E=\{e_i, e_{i+1}, ..., e_{i+n}\}$; проверки условия устойчивой работы ЗУ $E(t_i)=E(t_{i+1})= ... =E(t_{i+n})$.

При этом используются различные алгоритмы записи и считывания данных, обеспечивающие коррекцию сбоев в ЗУ МПСУ.

Синтез ЗУ отказоустойчивых МПСУ с применением специализированных БИС. Обнаружения и исправления ошибок в ЗУ МПСУ можно добиться с помощью специализированных БИС — устройств обнаружения и исправления ошибок (УОИО), нашедших уже достаточно широкое применение на практике [19].

В большинстве БИС УОИО используется удлиненный код Хэмминга, обеспечивающий работу с 8, 16 и 32-разрядными данными, причем в большинстве из них предусматривается наращивание секций УОИО для увеличения разрядности обрабатываемого слова. Почти все УОИО работают в четырех основных режимах: вычисление проверочных разрядов в режиме записи, обнару-

жение и исправление ошибок, обнаружение ошибок и прямое пропускание в режиме считывания.

В первом режиме (вычисление проверочных разрядов в режиме записи) происходит вычисление указанных разрядов по данным на входе устройства (обычно хранящихся во входном регистре данных УОИО) и выдача данных и проверочных разрядов для записи в память (непосредственно или с выходных регистров).

Во втором режиме (обнаружение и исправление ошибок) УОИО используется для определения ошибок в считываемом слове ЗУ и коррекции их при каждом обращении к памяти.

В третьем режиме (обнаружение ошибок) УОИО осуществляет чисто контрольные функции. Ошибки обнаруживаются и индицируются (исправимые и неисправимые) путем выставления флагов ошибок, но данные проходят от входа до выхода устройства без коррекции. Это позволяет контролировать данные без существенного замедления работы УОИО путем включения состояния ожидания в каждый цикл чтения с целью определения наличия ошибки. Если ошибка не обнаруживается, на что требуется примерно 30 нс после поступления слова, то сигнал ожидания снимается и продолжается цикл считывания. Если в слове выявляется ошибка, то ее коррекция осуществляется или УОИО (при блокировке данных, считанных из памяти) или программным путем с помощью центрального процессора МПСУ. Для обеспечения этого режима в УОИО предусмотрены выводы флагов ошибок.

В четвертом режиме (прямое пропускание в режиме считывания) УОИО обеспечивает прохождение разрядов данных и проверочных разрядов от входа к выходу.

В дополнение к перечисленным режимам некоторые УОИО имеют режим диагностики, обеспечивающий выбор соответствующих диагностических операций или контроля памяти и УОИО программными средствами.

Рассмотрим особенности применения УОИО в МПСУ с целью коррекции возникающих ошибок. Известно, что в блоках памяти на БИС ОЗУ с организацией ($N\times1$) возникновение независимых ошибок в словах является наиболее вероятным событием по отношению к сбоям и к отказам ЗЭ, поэтому применение УОИО, рассчитанных на исправление одиночных и двойных ошибок, в блоках памяти на одноразрядных ОЗУ является эффективным средством надежного хранения информации.

При проектировании систем памяти на основе многоразрядных БИС ОЗУ или ПЗУ (16×4; 64×4; 8×8 К бит и др.) применять УОИО, использующие коды Хэмминга, становится неэффективно, так как этот код устраняет одиночные сбои, но не исправляет ошибок хранения из-за постоянных отказов целых БИС или значительной их части, а данные ошибки в БИС ЗУ наиболее вероятны среди постоянных ошибок (однобитовых, строки или столбца,

всей БИС). При выходе из строя хотя бы одного БИС ЗУ в системе памяти ошибочными могут оказаться сразу несколько разрядов слова (4, 8, 16). Подобные ошибки называют *модульными,* а обнаруживаются и корректируются они с помощью модульных кодов. Данные коды позволяют не только обнаруживать и определять местоположение отказавшего модуля, но и исправлять ошибки, возникающие при искажениях подблоков (модулей) кодовых слов и отдельных их разрядов. Для повышения быстродействия ЗУ рекомендуется применять модульные коды с минимальной плотностью проверок и минимально возможным числом проверочных разрядов.

В зависимости от архитектуры МПСУ, ее назначения производительности, надежностных и стоимостных характеристик, используемой емкости памяти рекомендуются к применению следующие типовые решения:

— если в МПСУ в любой момент времени производится обращение лишь к одному из модулей ЗУ, то может использоваться структура (рис. 6.16, *а*) с одним УОИО на систему. При этом размещение УОИО не в блоке памяти, а рядом с МП позволяет корректировать как ошибки хранения, так и ошибки, возникающие в шинах;

— если в МПСУ допускается одновременное обращение к различным модулям ЗУ, то может использоваться структура (рис. 6.16, *б*) с размещением УОИО в каждом отдельном модуле и стандартной шиной данных, что упрощает разработку ММПСУ с распределенным управлением;

— предотвращения ошибок в памяти МПСУ с использованием УОИО добиваются аппаратно-программным способом с реализацией двухступенчатой системы контроля или чисто аппаратным способом.

В структуре (рис. 6.16, *в*) для первого случая одна из ступеней контроля обнаруживает заданное число ошибок с помощью применяемого кода, а другая — исправляет ошибки с помощью программ обработки прерываний от схем первой ступени (схем обнаружения ошибок). При появлении обнаруживаемых кодом ошибок схема прерывания запоминает адреса текущего слова, данные процессора и выходные данные контроля по коду (синдром) в соответствующих регистрах и выдает МП принудительный сигнал о включении программы восстановления информации. Данная структура, обеспечивающая повышенное качество хранения информации, рекомендуется для быстродействующих МПСУ, так как время обращения к памяти здесь практически не снижается из-за наличия схем первой ступени. Однако при продолжительной эксплуатации МПСУ частота появления ошибок хранения увеличивается, начинает расти частота прерываний и понижается быстродействие системы из-за затрат времени на восстановление информации. Кроме того, программа восстановления в этом слу-

чае должна храниться в дополнительной памяти, более надежной, чем основная контролируемая память.

В структуре (рис. 6.16, *г*) для второго способа контроля ЗУ МПСУ исправление ошибок происходит при каждом обращении к памяти только с помощью УОИО, а при возникновении неисправимых ошибок — под управлением блока контроля.

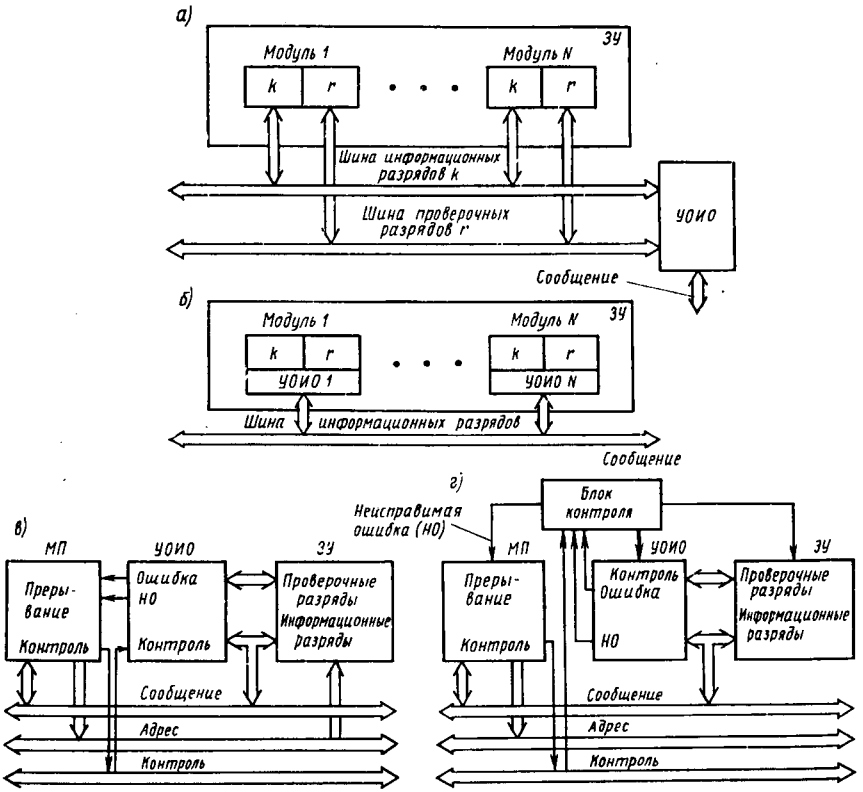


Рис. 6.16. Типовые решения применения УОИО для контроля систем памяти: *а* — структура с одним УОИО на систему; *б* — структура с индивидуальным УОИО на каждый модуль памяти; *в, г* — аппаратно-программный и аппаратный методы контроля систем памяти в МПС

Возможен и комбинированный способ использования УОИО, когда ошибки малой кратности исправляются аппаратным методом, а большой кратности — аппаратно-программным с применением резервных слов и разрядов для замены отказавших по мере накопления отказов. Это позволяет нейтрализовать ошибки большой кратности с помощью несложного избыточного оборудования, повысить живучесть и быстродействие системы. Подобное использование УОИО и программных средств в сочетании с ре-

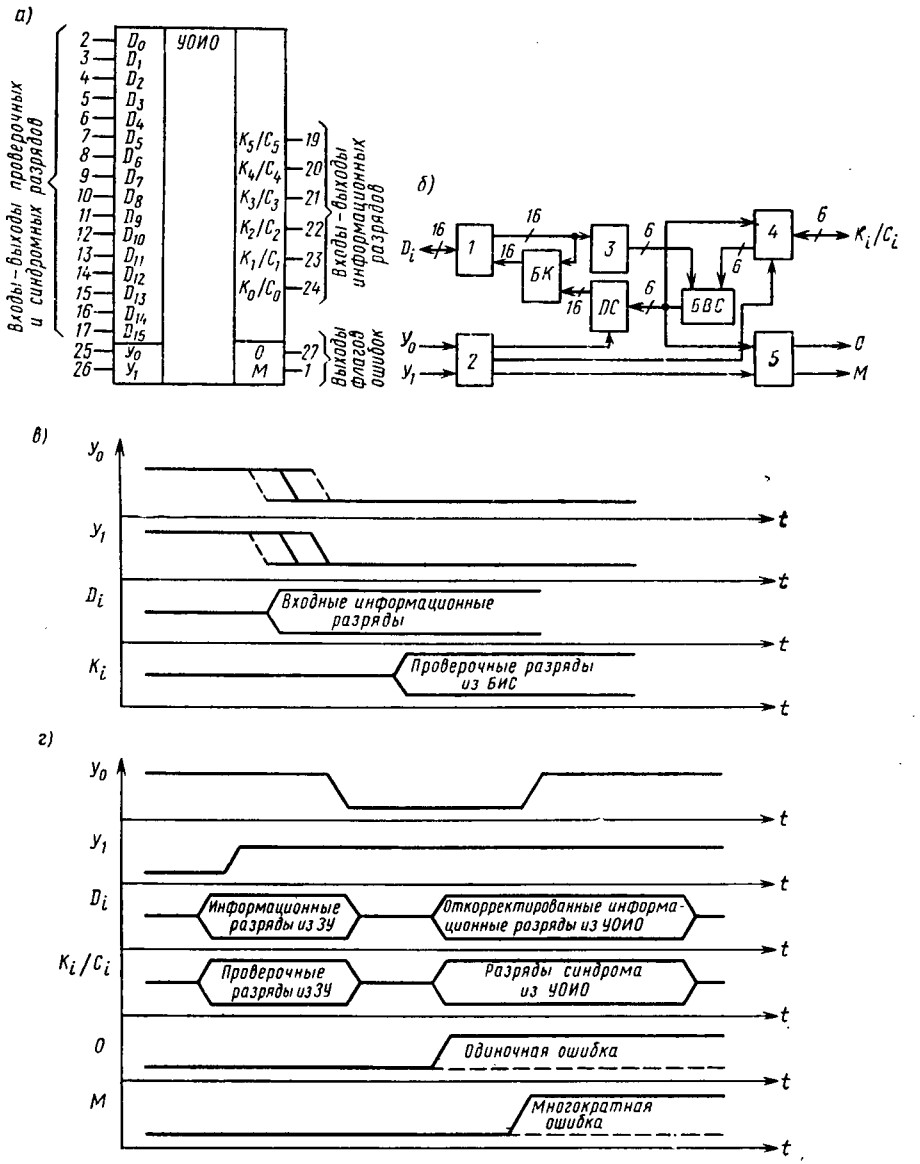


Рис. 6.17. БИС УОИО типа К555ВЖ1:

а, б — условное обозначение и структура; *в, г* — временные циклограммы работы УОИО в режимах записи и считывания; *1* — блок задания направления обмена данными; *2* — блок управления; *3* — блок вычисления проверочных разрядов; *4* — блок задания направления обмена проверочными и синдромными разрядами; *5* — блок формирования признаков ошибок

зервными элементами рекомендуется для построения управляющих систем повышенной надежности с большим временем эксплуатации.

Рассмотрим некоторые типовые структуры МПСУ с применением УОИО на базе отечественной БИС К555ВЖ1, предназначенной для исправления одиночных и обнаружения двойных и тройных ошибок, а также всех «0» и всех «1». Структура, режимы работы и временные диаграммы БИС К555ВЖ1 приведены на рис. 6.17 и в табл. 6.2.

Таблица 6.2. **Режимы работы БИС УОИО К555ВЖ1**

Входы управления		Цикл памяти	Режим работы	Характер информации на шине D_i	Характер информации K_i/C_i	Флаги ошибок О и М
y_0	y_1					
0	0	Запись	Вычисление проверочных разрядов	Входные данные в ЗУ	Выход. проверочные разряды r с БИС	Запрещены
1	0	Чтение	Запись информации K_i и r из ЗУ в БИС	Входные данные из ЗУ	Вход. проверочные разряды r из ЗУ	»
1	1	Чтение	Блокировка информации и размещение флагов ошибки	Отключенное состояние	Отключенное состояние	Разрешены
0	1	»	Выдача исправленных данных и синдрома ошибки	Выходные данные с БИС	Выходные разряды синдрома ошибки с БИС	»

Структура МПСУ с использованием одной БИС К555ВЖ1 (рис. 6.18, *а*) позволяет осуществлять непосредственное сопряжение УОИО с интерфейсом «Общая шина» (ОШ) и реализовать режим ожидания в каждом цикле считывания для обнаружения возможных ошибок в ЗУ. Если ошибок в ЗУ нет, то по окончании периода ожидания продолжается цикл считывания непосредственно из ЗУ. При обнаружении одиночной ошибки осуществляется блокировка выдачи данных в ЗУ, а их считывание производится через БИС УОИО. При наличии на управляющем входе $У_0$ перепада 0→1 информационные и проверочные разряды, считываемые из ЗУ, блокируются во входных регистрах БИС. В результате их

обработки формируются флаги ошибок О и М. Если ошибка отсутствует, то МП может принять 16-разрядное слово непосредственно из ЗУ. При обнаружении одиночной ошибки блок управления УОИО изменяет сигнал на входе $У_0$ с 1→0 и обеспечивает выдачу скорректированных данных и синдрома ошибки из БИС. Обнаружение многократной ошибки, не корректируемой БИС УОИО, является условием прерывания передачи данных из ЗУ.

Уменьшения периода ожидания и упрощения синхронизации БИС УОИО и интерфейсных схем достигают путем разделения управления режимами работы и применения структуры МПСУ с двумя БИС, функционирующими как кодер и декодер соответственно (рис. 6.18, *б*). Данное решение особенно эффективно при байтовой обработке в режиме «считывание — модификация — запись».

Еще более высокого быстродействия МПСУ добиваются введением в ее структуру входного и выходного регистров и мультиплексора и организации внутренней информационной шины (рис. 6.18, *в*). Здесь в цикле записи данные с внешней ШД поступают через входной регистр и мультиплексор, предназначенный для генерации проверочных разрядов. При этом данные фиксируются в регистрах УОИО, обеспечивая освобождение внешней ШД для организации передачи данных следующего цикла, что повышает быстродействие в режиме записи.

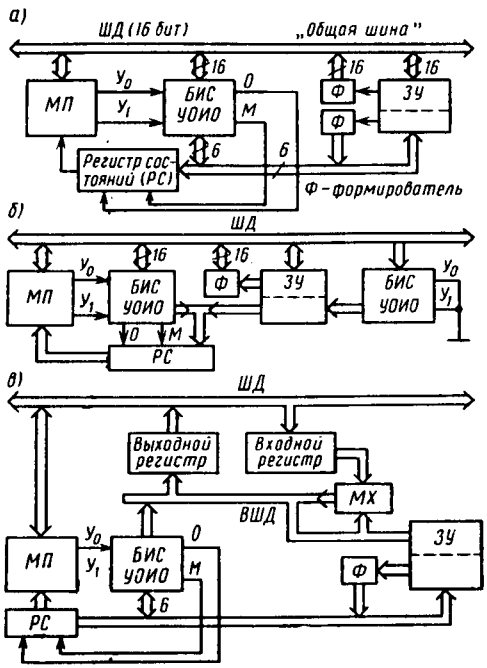


Рис. 6.18. Типовые реализации МПС с использованием УОИО К555ВЖ1:

а — для МПС одним УОИО и интерфейсом ОШ; *б* — для МПС двумя УОИО; *в* — для МПС с одним УОИО и дополнительной логикой; Ф — формирователь; РС — регистр состояний; МХ — мультиплексор

В цикле считывания для ускорения обработки данных вводится режим «Прямая передача», состоящий в фиксации считываемых из ЗУ данных в выходном регистре с последующей передачей их на внешнюю шину. Если ошибки отсутствуют, то такое решение практически не снижает быстродействия ЗУ. При обнаружении корректируемой ошибки сигнал признака ошибки «0» разрешает запись в выходной регистр исправленной информации из БИС УОИО с последующей выдачей на внешнюю шину.

§ 6.5. СИНТЕЗ САМОВОССТАНАВЛИВАЮЩИХСЯ МПСУ

В последнее время создания надежных МПСУ добиваются путем применения сочетания маскирующей избыточности и динамического резервирования с замещением отказавших элементов. К числу систем, построенных на этих принципах, относятся, например, известные вычислительные системы типа JPL-STAR, TANDEM-16, C^*_{mmp}, C^*_m, MICRONET, AXE, ARPA-PLURIBUS и др. [5, 12, 13, 17].

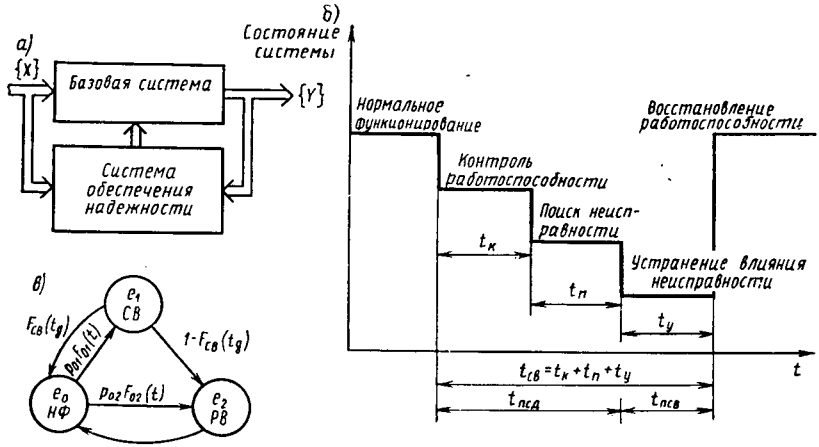


Рис. 6.19. Самовосстанавливающиеся МПСУ:
а — упрощенная структура системы; *б* — временна́я диаграмма процесса самовосстановления; *в* — упрощенный граф поведения системы

Системы с введением в их структуру динамического замещения отказавших элементов, средств оперативного контроля и диагностики, обеспечивающих устранение влияния неисправностей, сбоев и отказов за время, не приводящее к нарушению работоспособности системы, называют *самовосстанавливающимися.* Укрупненно структуру самовосстанавливающейся МПСУ (СВМПСУ) (рис. 6.19, *а*) можно представить совокупностью *базовой системы* (БС), выполняющей предписанные алгоритмы управления, и дополнительной *системы обеспечения надежности* (СОН), обеспечивающей с помощью аппаратных и программных средств автоматическое полное или частичное восстановление работоспособности системы при появлении различного рода неисправностей.

В процессе самовосстановления СВМПСУ можно выделить (рис. 6.19, *б*) три основных подпроцесса: контроля работоспособного состояния, поиска места неисправности с точностью до сменного замещаемого блока, устранения влияния неисправности. В соответствии с этим в структуре СОН выделяются подсистема самодиагностирования (контроля и поиска) и подсистема само-

восстановления (устранения влияния неисправностей), реализуемые аппаратным или программным путем.

Подсистемой самодиагностирования (ПСД) производится функциональное и тестовое диагностирование. Особенность *функционального диагностирования* состоит в его оперативности, достигаемой введением в структуру контролируемого блока избыточных аппаратных средств. Для *тестового диагностирования* характерна задержка времени обнаружения отказа, присущая программным методам.

Подсистема самовосстановления (ПСВ) полностью или частично восстанавливает работоспособность системы путем перестройки структуры и программного рестарта с предыдущей контрольной точки. При полном самовосстановлении вместо отказавшего функционального модуля (МЭВМ, МП и др.) включается идентичный ему резервный. При частичном самовосстановлении, основанном на функциональном резервировании, отказавший функциональный модуль исключается из рабочей конфигурации системы путем аппаратной и (или) программной перестройки, производимой с целью сохранения наиболее важных функций. В СВМПСУ для сохранения работоспособности необходим резерв времени, отводимый на реализацию процедур контроля, поиска и устранения отказов. Его длительность может быть обусловлена цикличностью обработки поступающей информации, инерционностью управляемого объекта или наблюдаемого физического процесса, когда потеря одной-двух выходных точек существенно не сказывается на качестве управления. Другим источником резерва времени может служить запас производительности системы или алгоритм ее функционирования при неполной загрузке, когда время выполнения задания меньше периода поступления заданий.

Проектирование системы обеспечения надежности СВМПСУ включает этапы системного и структурного синтеза. На этапе *системного синтеза* производится выбор показателя надежности системы как основного критерия качества и выбор наиболее приемлемого варианта из множества вариантов реализации подсистем самодиагностирования и самовосстановления СОН за счет учета имеющихся уже ограничений, оценок затрат оборудования, времени на реализацию каждого способа построения подсистем и др.

Например, если выбрана структура интерфейса типа ОШ, а в качестве элементов применяются серийные МЭВМ без встроенного контроля, то это ограничивает построение СОН вариантами самовосстановления на основе изменения адреса на ОШ и тестовыми методами диагностирования. Результатом системного этапа проектирования СОН является перечень допустимых вариантов построения ПСД и ПСВ. Дальнейший поиск рационального варианта структуры СВМПСУ производится на структурном этапе синтеза.

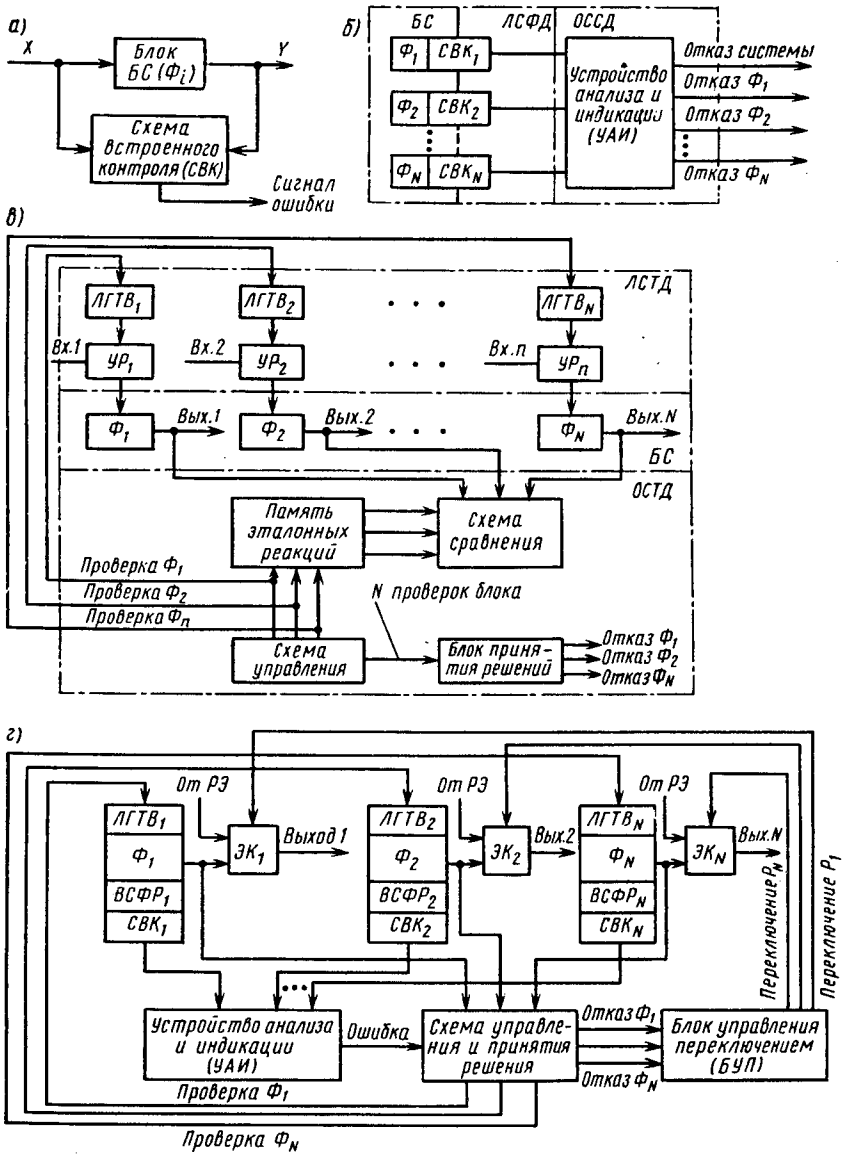


Рис. 6.20. Типовые структуры организации СВМПСУ:

а, б — структура организации оперативного контроля в отдельном блоке и базовой системе; *в* — структура организации тестового контроля базовой системы; *г* — обобщенная структура СВМПСУ

Рассмотрим некоторые типовые подходы к организации СОН СВМПСУ и построению ПСД и ПСВ.

Диагностирование СВМПСУ производится с целью контроля правильного функционирования и поиска места неисправности с точностью до сменного замещаемого блока. Контроль за обнаружением факта ошибки может осуществляться *оперативно* (параллельно с протекаемым вычислительным процессом), *с некоторой задержкой* по отношению к каждой выполняемой программе или группе программ и *периодически* в заданные или случайные моменты времени.

Функциональное диагностирование, или оперативный контроль обычно реализуется схемами встроенного контроля (СВК). СВК используют принцип избыточности информации с разбиением всех входных, промежуточных и выходных сигналов на допустимые и запрещенные подмножества, принадлежность к которым и фиксируется этими СВК.

Каждый узел базовой системы (Φ_1—Φ_N) содержит свои СВК (рис. 6.20, *а, б*), сигналы от которых распознаются устройством анализа и индикации (УАИ) с указанием номера отказавшего блока, инициализацией процесса перестройки структуры СОН и программного рестарта системы.

В системах *тестового* диагностирования применяют программные и аппаратные средства (рис. 6.20, *в*). Объектом диагностирования являются основные элементы БС — рабочие (резервные) функциональные блоки ($\Phi_1 - \Phi_N$). Диагностирование производится локальными генераторами тактовых воздействий (ЛГТВ) с использованием управляемых регистров (УР).

Общие средства тестового диагностирования (ОСТД) осуществляют общее управление $ЛГТВ_1 - ЛГТВ_N$ и переключением резерва, сравнение выходных сигналов каждого из рабочих функциональных блоков Φ_i с их эталонными реакциями, хранящимися в памяти, и принятие наилучшего решения. При этом активно применяются схемы организации ПСД с сосредоточенным встроенным или распределенным ядром средств контроля.

Программная часть ПСД включает в себя: *резидентные диагностические программы,* осуществляющие элементарные проверки контроля исправности аппаратуры и поиск места неисправности; *внешние диагностические программы,* загружаемые с внешних ЗУ, для глубокого контроля исправности элементов системы во время пауз в вычислительном процессе и поиска места неисправностей, обнаруженных резидентными программами; *управляющую программу* — диагностический супервизор. В них широко применяются разнообразные программные (алгоритмические, логические, программно-временные) и аппаратно-программные методы (контроль затрат времени обращений в запрещенные участки памяти, микропрограммный контроль).

По завершении работы подсистемы диагностики СОН и указания места неисправности с точностью до сменного заменяемого блока начинает работать подсистема самовосстановления СОН. Она устраняет влияние неисправности путем физической перестройки системы (замещением отказавшего элемента резервным при полном восстановлении или перераспределением функций между оставшимися исправными элементами при частичном) с последующим программным рестартом с последней контрольной точки.

Сложность физической перестройки структуры СВМПСУ определяется сложностью замещаемого блока и типом его связей с остальными блоками системы. Наиболее просто перестройка реализуется в многомашинных и мультипроцессорных МПСУ при стандартных интерфейсных системах, где развитые связи между МЭВМ позволяют создавать либо обходные пути (в системах с непосредственной связью по полному графу), либо исключать МЭВМ из обращения (при связи устройств через канал, общее внешнее ЗУ, общую шину и матричный коммутатор), обеспечивать частичное восстановление с постепенной деградацией вычислительных возможностей.

При частичном восстановлении работоспособности СВМПСУ для сохранения наиболее важных выполняемых функций производится перераспределение функций отказавшего элемента между соседними (функциональное резервирование), что требует дополнительных аппаратурных затрат. В качестве средств частичного самовосстановления в СВМПСУ, в частности, используются средства реконфигурации байтовых слоев, при отказе которых МП продолжает вычисления оставшимися разрядами с потерей точности или производительности [36].

Ядро СВМПСУ строится на более надежных элементах с использованием мажоритарного принципа. Самопроверяемые СВК обычно строятся при двухпроводном кодировании выходных сигналов, при этом их сложность обычно не превышает сложности схем контроля, реализованных по методу дублирования. В общем случае в СВМПСУ могут использоваться различные средства функционального, тестового комплексного самодиагностирования (или их сочетания) и самовосстановления. Обобщенная структура аппаратных средств СВМПСУ (рис. 6.20, *г*) включает в себя: основные элементы ($\Phi_1 - \Phi_N$) БС; локальные средства функционального диагностирования (ЛСФД), включающие в себя $СВК_1 - СВК_N$; локальные средства тестового диагностирования (ЛСТД), включающие в себя ЛГТВ и УР; локальные средства самовосстановления (ЛССВ), включающие в себя вспомогательные средства функционального резервирования (ВСФР) и элементы коммутации (ЭК); общие средства самодиагностирования и самовосстановления (ядро СОН), включающие в себя устройство анализа и индикации (УАИ), схемы управления и принятия решения, блок

управления переключением резерва (БУП). Обнаруженный отказ встроенных в каждый блок ЛСФД и ЛСТД инициирует процедуру самовосстановления, при успешном завершении которой в допустимые интервалы времени отказ ЛСФД и ЛСТД не нарушает работоспособности системы. Неисправность в ЛСТД не приводит к отказу системы до момента отказа в базовой системе. Отказ ядра СОН вызывает отказ системы. Упрощенно модель поведения СВМПСУ (рис. 6.19, *в*) можно представить графом, содержащим три обобщенных состояния — нормального функционирования (НФ), ручного восстановления (РВ) и самовосстановления (СВ) — и отражающим действия по поиску места неисправности и устранению ее влияния. В случае самовосстановления за допустимое время система возвращается в состояние НФ и не теряет при этом работоспособности; в противном случае наступает отказ системы и требуется ее ручное восстановление.

Контрольные вопросы

1. Перечислите основные пути и принципы повышения надежности МПСУ. 2. Что такое резервирование: а) замещением? б) постоянное? в) комбинированное? Приведите примеры построения избыточных структур с использованием этих методов резервирования. 3. Чем привлекательна для МПСУ кодовая избыточность по сравнению со структурной? Отметьте достоинства и недостатки кодовой избыточности. Какие технические решения используются в реализации избыточных корректирующих автоматов? 4. В чем заключаются особенности методов синтеза и технических решений МПСУ: а) при мажоритарном резервировании? б) при комбинированном резервировании? в) при использовании адаптивных ВО? г) при использовании кодовой и временной избыточности? д) при использовании аппаратурной и временной избыточности? 5. Что такое УОИО? Поясните принцип действия УОИО на примере работы микросхемы К555ВЖ1. 6. В чем заключается сложность синтеза самовосстанавливающихся МПСУ? Какие подсистемы СВМПСУ участвуют в процессе обеспечения надежности и каким образом? Поясните особенности процесса функционирования СВМПСУ при возникновении отказов и сбоев.

ГЛАВА 7

ПОВЫШЕНИЕ НАДЕЖНОСТИ И КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПРИ ПРОЕКТИРОВАНИИ МПСУ

Программное обеспечение МПСУ по сравнению с их аппаратными средствами является более сложной, крупной и ответственной компонентной, гибко реализующей большой круг функций, возлагаемых на МПСУ. В связи с этим особенно важно овладеть навыками и приемами проектирования программных средств, уметь применять на практике методы повышения качества и надежности разрабатываемых программных комплексов, модели оценки сложности и надежности ПО.

§ 7.1. ОБЩИЕ СВЕДЕНИЯ

Высокая надежность, быстродействие и универсальность применения МП и МЭВМ для управления различными технологическими процессами в значительной степени определяются качеством их ПО. Весь комплекс ПО МПСУ подчинен созданию рабочих прикладных программ, реализующих заданные функции управления и составляющих *специальное ПО.*

Процесс разработки комплекса рабочих программ для МПСУ решается с учетом ограниченности имеющихся ресурсов, эффективности функционирования их в системе управления, качества и сроков создания программ, трудоемкости обнаружения и исправления ошибок в программах. Успешное решение этой задачи невозможно без автоматизации большинства этапов программирования, создания комплекса универсальных программных средств, называемого в отличие от специального ПО *общим* или *системным ПО.* Системное ПО в зависимости от средств реализации подразделяется на внешнее и внутреннее. Под *внешним ПО* обычно понимают комплекс программных средств, реализованных на ЭВМ, мини-ЭВМ или даже на какой-либо МЭВМ, не участвующей непосредственно в управлении конкретным технологическим процессом. Доступность таких межмашинных средств позволяет осуществлять одновременную разработку аппаратурной части МПСУ и ее ПО.

Под *внутренним ПО* понимают комплекс программ, реализованных на языке МЭВМ, непосредственно участвующей в процессе управления. Так как подробный анализ компонентов внутреннего и внешнего ПО, языков программирования, применяемых для МПСУ, проведен в ряде монографий и учебных пособий [1—3, 20, 31, 34 и др.], то далее эти вопросы не рассматриваются. Наряду с обеспечением надежности аппаратных средств при проектировании МПСУ необходимо обеспечивать надежность разрабатываемых программных средств. Важность повышения надежности ПО обусловливается еще и тем, что оно реализует основные функции управления и обработки данных в условиях действия разнообразных помех, вызывающих отказы и сбои технических средств, существенно влияет на длительность перерывов, качество функционирования МПСУ.

Под *надежностью ПО* понимают вероятность его безотказной работы в заданных условиях в течение определенного времени с учетом «веса» (степени серьезности) отказов. *Отказ* — отклонение характеристик процесса функционирования за допустимые пределы; *ошибка* — изъян программы, приводящий к отказу. В большинстве случаев надежность ПО базируется на понятиях количества ошибок, корректности и устойчивости программ.

В зависимости от важности функций, выполняемых программами, при выявлении в них конкретных ошибок применяют раз-

нообразные средства и методы предупреждения, обнаружения и исправления ошибок, а также обеспечения качества и надежности разрабатываемого ПО. Рассмотрим эти вопросы более подробно.

§ 7.2. КРИТЕРИИ, СРЕДСТВА И МЕТОДЫ ОБЕСПЕЧЕНИЯ НАДЕЖНОСТИ И КАЧЕСТВА ПРОГРАММ

При разработке, эксплуатации или выборе наиболее целесообразных вариантов программных комплексов разработчик должен знать и уметь применять на практике разнообразные методы повышения надежности и качества программ, влияющих на эффективность функционирования микропроцессорных систем управления в целом.

Основные критерии качества ПО. Принципиальной особенностью комплексов программ (КП) является невозможность выделения для них единого критерия качества. В связи с этим оценку качества КП или сравнение ряда программ осуществляют с помощью функциональных и конструктивных критериев.

Функциональные критерии отражают основную специфику применения и степень соответствия программ их целевому назначению и включают в себя показатели точности выходных данных, диапазоны изменения параметров, время реакции, адаптивность к внешним воздействиям (для программ управления), объем исходных данных, достоверность результатов, разнообразие функций редактирования (для программ обработки информации). Иногда функциональные критерии сводят к некоторым обобщенным показателям — экономической эффективности применения КП, их стоимости и др.

Конструктивные критерии качества КП более инвариантны к их целевому назначению и основным функциям. К ним относятся, например, сложность программ, надежность функционирования, используемые ресурсы ЭВМ, корректность и др.

Основными критериями качества КП на этапе проектирования являются сложность и корректность программ, при этом различают сложность программных модулей, сложность КП и межмодульных связей, сложность структуры данных.

Структурная сложность программ определяется числом взаимодействующих ПМ, числом связей между ними и сложностью их взаимодействия. Установлено, что сложность ПМ связана не столько с размером программы по числу команд, сколько с числом отдельных путей ее исполнения, набором путей — маршрутов для возможной обработки данных. Это позволяет оценивать трудоемкость тестирования и обслуживания модуля, а также потенциальную надежность его функционирования.

Сложность КП оценивают с учетом внутренней сложности ПМ и сложности межмодульных связей. Интуитивное решение этой задачи привело к формализации правил структурного построения

КП, организации межмодульного интерфейса с ограничением размеров ПМ (30—100 операторов на языке высокого уровня, 100—600 операторов автокода, 2—3 страницы текста в распечатках).

Структурную сложность построения КП оценивают при учете глубины информационных и управляющих связей между модулями. В простейшем случае комплексы и группы программ оценивают числом входящих в них модулей, суммарным количеством команд или операторов, числом уровней иерархии, обращений, а также информационных и управляющих внутри- и межмодульных связей через глобальные и обменные переменные и другими непосредственно измеряемыми показателями.

При статистическом подходе к оценке сложности используются такие показатели, как длина программы, число ветвлений, число операндов и др. Установлено, например, что количество ошибок в программе сильнее коррелировано с числом условных переходов $K=0{,}92$, чем с числом команд $K=0{,}83$. Это позволило использовать в качестве меры трудоемкости и сложности разработки ПМ число условных переходов в программе. Наиболее полно статистическая теория сложности ПО разработана М. Х. Холстедом [43].

В целом при сравнительной оценке сложности разрабатываемых ПМ используются следующие показатели (см. табл. П.III.1) [42]:

а) логическая сложность, измеряемая числом логических операторов (разветвлений, циклов, условных переходов);

б) сложность взаимосвязей, измеряемая числом прикладных и системных интерфейсов рассматриваемого модуля, т. е. числом вызываемых из него прикладных и системных программ;

в) сложность вычислений, измеряемая числом операторов присваивания, содержащих арифметические операции;

г) сложность ввода — вывода, измеряемая числом операций ввода — вывода;

д) удобочитаемость, измеряемая числом комментариев.

Под *корректностью,* или *правильностью,* всегда подразумевается соответствие проверяемого объекта некоторому эталонному объекту или совокупности эталонных характеристик и правил. Корректность программы при проектировании наиболее полно определяется степенью ее соответствия предъявляемым к ней формализованным требованиям — программной спецификации.

Корректность сложных КП определяется корректностью: а) текстов программ на языках спецификаций и программирования; б) программных модулей; в) данных, правилами их структурирования и упорядочения; г) групп и комплексов программ, правилами модульного построения КП и организации межмодульных связей. Эти виды корректности программ устанавливаются применением различных средств, характеризуемых наличием эталонов, а также методов проверки соответствия программ эталонам.

Установление корректности программ и выявление ошибок в них осуществляется на разных этапах проектирования КП. Считается, что в программе есть ошибка, если программа не выполняет того, что пользователь от нее ожидает.

Ошибки в программах подразделяются:

а) на первичные ошибки, включающие в себя технологические ошибки документации и фиксирования программ в памяти ЭВМ; программные ошибки (ошибки типов операций, ошибки управления, циклов и др.); алгоритмические ошибки, обусловленные некорректной постановкой задач, сопряжением модулей и групп программ; системные ошибки, обусловленные неполной информацией о протекающих процессах;

б) на вторичные ошибки, вызывающие искажение выходных результатов исполнения программ и требующие выполнения ряда операций по локализации и устранению первичной ошибки.

Затраты на исправление первичных ошибок в программах неравноценные: требуется, например, в среднем 6 команд для исправления каждой программной ошибки, 14 команд для каждой алгоритмической ошибки и около 25 команд для каждой системной ошибки. В то же время проявление ошибок обусловливает их глубокую связь с методами структурного построения программ, типом языка программирования, степенью автоматизации технологии программирования и другими факторами.

Относительные затраты времени, отводимые на создание ПО, обычно распределяются следующим образом: проектирование — 1/3; написание программ — 1/6; отладка компонентов ПО (подсистем, программ и модулей) — 1/4; комплексная отладка всех компонентов ПО — 1/4.

Методы обеспечения надежности ПО. Известно множество технологических и организационных методов и средств обеспечения надежности и качества программ, применяемых на всех этапах жизненного цикла ПО (см. табл. П.III.2).

На этапе задания требований и спецификаций используются специальные языки, представляющие собой, по существу, подмножества языков программирования или моделирования. На этапе проектирования ПО осуществляется преобразование постановки задачи в плане алгоритмического или вычислительного решения с учетом реальных возможностей ЭВМ и методов программирования (проектирование сверху вниз, проектирование структур данных и др.). На этапе отладки выполняется предварительная или окончательная проверка проекта с анализом соответствующих характеристик и объемов информации (описания выполненных алгоритмов и функций, правил взаимодействия, структур данных, инструкций пользователя) и отображением проектов ПО в графический и неграфический вид (блок-схемы, решающие таблицы, языки проектирования программ и др.).

Сущность *методов технологического обеспечения надежности*

программ заключается в широком применении передовых способов разработки ПО, механизмов обратной связи, осуществляемых как через проверку проектов вручную, так и через их автоматический контроль на базе ЭВМ. Важное место при этом занимают вопросы выбора средств программирования, реализации ПО одним из методов: сверху вниз, снизу вверх или модульным программированием.

Программирование сверху вниз начинается с самого высокого уровня с последовательным переходом к разработке более низких уровней вплоть до уровня программных модулей (ПМ). Каждый незапрограммированный модуль заменяется при сборке заглушкой, удовлетворяющей требованиям интерфейса. Далее заглушки дорабатываются до законченных ПМ в соответствии с планом программирования. На каждой стадии процесса реализации созданная программа тестируется.

Программирование снизу вверх начинается с реализации ПМ самого низкого уровня. Отладку и тестирование ведут обычно с помощью специальных отладочных средств на основе отработанных способов тестирования.

При *модульном программировании* написание программ производится небольшими частями — модулями, оформляемыми, как правило, в виде подпрограмм, функций или пакетов подпрограмм, выполняющих одну или несколько функций, входящих в задачу. Информация между ПМ передается в виде списка параметров, глобальных данных, общих блоков, пакетов и др.

Большинство из методов повышения надежности программ основывается на принципе исключения ошибок. Другая часть методов (дуальное или *N*-версное программирование) базируется на принципе допущения ошибок в создаваемом ПО, ограничивающем при этом последствия ошибок путем их выявления и исправления.

Известны методы программирования, влияющие на скорость написания, простоту доработки программ, что косвенно отражается на надежности ПО. К их числу относятся методы применения языков высокого уровня, структурного кодирования и программных стандартов. Все они, вместе взятые, обеспечивают надежность программ уже на ранних стадиях разработки ПО.

Организационные средства и методы повышения надежности ПО заключаются в грамотном планировании, организации и документировании работ по созданию программ. На качество и надежность разрабатываемых программ особенно влияют величина проекта (малый, средний или большой проект ПО) и степень его критичности. При этом малым считается проект, в создании которого участвует не более 5 человек — специалистов, средним — от 6 до 29 человек, большим — свыше 30 человек.

Основные практические рекомендации по выбору методов повышения надежности программ различной степени сложности на

этапах жизненного цикла ПО сведены в табл. П.III.2 с указанием ранга «ценности» того или иного метода и приоритета его использования в проектах ПО определенного назначения.

Факторы, влияющие на **надежность ПО**, тесно связаны: с особенностями внешних абонентов и пользователей; с искажениями исходных данных, типами ошибок в ПО, ведущих к отказам и сбоям; с методами структурного проектирования, контроля и исполнения программ и данных; с методами введения избыточности, предпускового и оперативного контроля, программного восстановления.

Отказовые ситуации особенно часто возникают при искажениях исходных данных, их отклонениях от значений, использовавшихся при тестировании и отладке программ. На надежность ПО влияют искажения данных в процессе их накопления и хранения в ЭВМ, а также ошибки в программах, которые могут приводить к различным последствиям и даже к полной или частичной потере работоспособности систем.

В зависимости от характера проявления обнаруженных искажений возможны следующие оперативные меры по ликвидации их последствий, восстановлению данных и сохранению процесса обработки информации: а) игнорирование обнаруженного искажения из-за незначительного его влияния на процесс обработки данных и выходные результаты; б) повторное решение функциональной задачи или исполнение программ при тех же исходных данных; в) исключение сообщения из обработки из-за его сильной искаженности или трудности продолжения вычислительного процесса; г) кратковременное прекращение решения текущей задачи до обновления исходных данных или устранения источника искажений; д) перестройка режима работы КП с целью уменьшения влияния перегрузки или в связи с потерей значительного объема информации о ходе процесса управления и обработки информации; е) переход на резервную ЭВМ с накопленной информацией о состоянии внешних процессов или восстановлений информации путем ее дублирования; ж) восстановление процесса управления и обработки информации с режима начального пуска всего ПО. Необходимо заметить, что при всех перечисленных типах оперативных реакций, за исключением последнего, существуют более или менее длительные отключения от нормального хода процесса обработки информации, а при последнем методе не обеспечивается непрерывность и устойчивость во взаимодействии ЭВМ с внешними абонентами.

Основные показатели надежности ПО и их измерение. Практически для оценки уровня надежности ПО, существенным образом влияющей на надежность МПСУ, рекомендуется использовать такие составные ее свойства, как устойчивость, корректность, восстанавливаемость и исправляемость ПО [38].

Под *устойчивостью ПО* понимают способность его функциони-

рования в условиях возмущений внешней среды, отклонение от нормы внешних воздействий (выход параметров решаемых задач за пределы допустимых областей, отказы и сбои используемых для реализации ПО вычислительных средств, ошибки в работе оперативного персонала, собственные ошибки ПО).

Различают устойчивость ПО в широком смысле, или абсолютную устойчивость, и в узком смысле, или относительную устойчивость. Под *абсолютной* устойчивостью понимают оптимальную (желательную, хотя и трудно осуществимую) способность ПО продолжать нормальное функционирование, обеспечивая решение задач при перечне определенных (допустимых) внешних возмущений. Под *относительной устойчивостью* понимается способность ПО при перечне возмущений, не позволяющих решать задачу, осуществлять перевод МПСУ в некоторое заранее предусмотренное состояние «защитного отказа». При этом обеспечивается либо прекращение работы МЭВМ с сигнализацией о выходе системы в защитный отказ, индикацией состояния аппаратно-программных средств системы и выдачей рекомендаций по возвращению системы к нормальному функционированию, либо прекращение решения данной задачи с сигнализацией о невозможности ее решения и указанием причины, а также с автоматическим переходом к решению других задач.

Количественно устойчивость ПО оценивается списками допустимых внешних возмущений, в пределах которых ПО либо сохраняет способность правильно решать задачи из заданного множества (список длиной l_a для абсолютной устойчивости), либо обеспечивает выход системы в «защитный отказ» (список длиной l_o для относительной устойчивости). При прочих равных условиях устойчивость ПО тем выше, чем шире соответствующий список допустимых возмущений. Оценка абсолютной ε_a и относительной ε_o устойчивости ПО вычисляется следующим образом:

$$\varepsilon_a = l_a/l_\Sigma; \quad \varepsilon_o = l_o/l_\Sigma; \quad \Delta\varepsilon = |\varepsilon_a - \varepsilon_o|. \tag{7.1}$$

Основная трудность в использовании показателей ε_a, ε_o, $\Delta\varepsilon$ заключается в сложности определения списка общего числа возмущений l_Σ, а также в неравноценной значимости различных учитываемых возмущений. Последний недостаток можно устранить введением приоритетов возмущений друг перед другом, оценки их матрицей бинарных предпочтений с расчетом «веса» и «цены» каждого из допустимых возмущений (см. § 5.1). Это позволяет более объективно определить количественные показатели устойчивости ПО.

Под *корректностью ПО* (правильностью, безошибочностью) понимают его соответствие целям и специфике решаемых задач. Аналогично устойчивости ПО, корректность ПО рассматривается в двух аспектах: а) корректность в узком смысле, или относитель-

ная корректность, отвечающая перечню выполняемых функций и условий функционирования, отраженных в спецификациях; б) корректность в широком смысле, или абсолютная корректность, отвечающая перечню реализуемых функций, возложенных на ПО и определяемых из фактического назначения и реальных условий применения и эксплуатации ПО. Однако практически для ПО МПСУ целесообразно использовать лишь относительную корректность из-за невозможности проведения полного тестирования. В этом случае спецификации на разрабатываемое ПО (с целью корректности его функционирования) должны содержать: а) область I невозмущенных исходных данных (множество возможных наборов исходных данных на входах ПО), обычно задаваемую при отсутствии требований по устойчивости ПО; б) области II и III для «слабых» и «сильных» допустимых возмущений, задаваемые при наличии требований к устойчивости ПО. В пределах «слабых» допустимых возмущений ПО должно обеспечиваться полное и правильное решение задач, а в пределах «сильных» возмущений — выход системы в «защитный отказ» с последующим восстановлением их реального функционирования.

В качестве количественных показателей корректности ПО используются оценки меры близости рассматриваемого ПО к идеальному (абсолютно корректному, не содержащему ошибок ПО). Такими показателями корректности ПО служат: $\bar{r}$ — среднее число ошибок в ПО; $K_{o.o} = \text{Вер}\{\bar{r}=0\}$ — вероятность отсутствия ошибок в ПО.

Показатели $\bar{r}$ и $K_{o.o}$ для конкретного ПО являются величинами дискретными и вероятностными и зависят, кроме его собственных характеристик (сложности, степени разветвленности программ, объема используемых данных и др.), от уровня квалификации программистов-разработчиков и могут рассматриваться только по отношению к этой квалификации. При пуассоновском законе распределения величин $\bar{r}$ и $K_{o.o}$ имеет место равенство $K_{o.o} = e^{-\bar{r}}$.

Недостаток показателей $\bar{r}$ и $K_{o.o}$ заключается в неучете местоположения ошибок в программе и статистики свойств реального потока решаемых задач, считая их равновероятностными. Так как учет этой статистики очень важен, то может использоваться еще один показатель — *вероятность успешного решения L данным ПО произвольной задачи,* случайным образом выбранной из реального потока задач: $L = \text{Вер}\{W\}$, где W — событие, состоящее в успешном решении одной задачи.

Показатель L отражает как число ошибок, имеющихся в ПО, и их размещение в ветвях программы, так и статистические свойства реального потока задач, что позволяет переходить от заданного уровня корректности ПО к значениям надежности МПСУ в целом.

Под *восстанавливаемостью ПО* понимают его приспособленность к быстрому возвращению к нормальному функционированию из состояния «защитного отказа».

Под *исправляемостью ПО* понимают его приспособленность к внесению исправлений для устранения замеченных ошибок (программистом или группой программистов определенного уровня квалификации), а также к внесению доработок и усовершенствований, не изменяющих существенным образом функции, назначение и структуру ПО.

Свойства восстанавливаемости и исправляемости ПО являются адекватными свойствами ремонтопригодности и модернизируемости технических средств.

Если принять, что время, необходимое для возвращения ПО к нормальному функционированию после «защитного отказа», равно $T_в$, а время, необходимое для внесения необходимых изменений с целью исправления обнаруженной ошибки при некотором среднем уровне квалификации программистов, равно $T_и$ и эти величины являются случайными, то исчерпывающей характеристикой свойства восстанавливаемости и исправляемости ПО является *распределение случайных величин* $T_в$ и $T_и$. В качестве числовых показателей восстанавливаемости и исправляемости могут использоваться математическое ожидание $\overline{T}_в$ и $\overline{T}_и$; вероятность восстановления или исправления за фиксированное время τ $T_в(\tau)$ или $T_и(\tau)$, квантили случайной величины $T_{вк}$ или $T_{ик}$ $(к=\overline{1, N})$.

§ 7.3. ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ И ОБЕСПЕЧЕНИЯ НАДЕЖНОСТИ ПО

При проектировании микропроцессорных систем управления повышенные требования предъявляются к надежности программного обеспечения, так как отказ или ошибки в программах могут привести к серьезным последствиям в работе реальных устройств и объектов управления.

Основные этапы проектирования надежного ПО. Проектирование ПО включает в себя ряд этапов (рис. 7.1), начиная с постановки задачи, определения требований и целей ПО и кончая написанием текста программы (в простейшем случае) или совокупности текстов программных модулей (в крупном программном комплексе).

На первом этапе проектирования ПО с участием группы разработчиков и пользователей ставится задача — определение требований к ПО с анализированием существующих систем, получением полной информации от пользователя, оценкой достоинств от возможной реализации ПО и др. После определения требований к ПО пользователь отвечает за их полноту и точность,

а разработчик — за реализуемость, достижимость и понимание сформулированных требований, указание их приоритета с целью принятия компромиссов на последующих этапах проектирования ПО.

На втором этапе проектирования требования пользователя переводятся в наборы конкретных целей ПО с учетом компромиссов между требованиями с анализом и формулировкой следующих факторов:

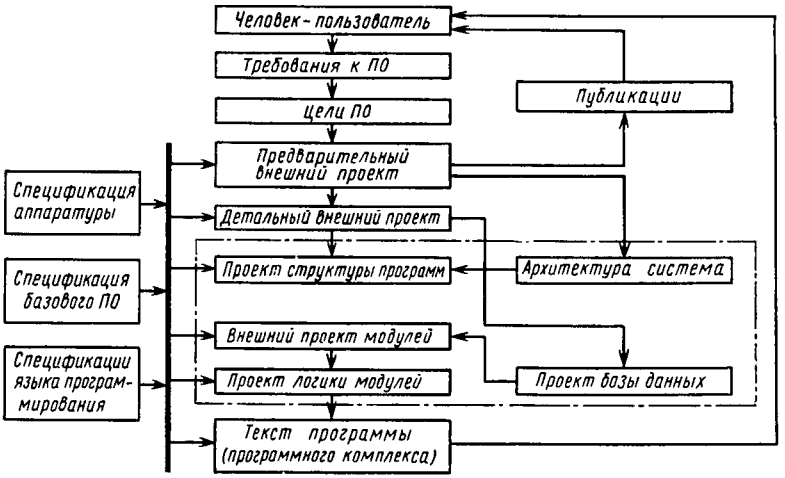


Рис. 7.1. Основные этапы проектирования программного обеспечения

а) показателей надежности ПО — среднего времени между отказами с учетом степени их серьезности (отказ системы, ошибка пользователя, отказы конкретных функций); среднего времени восстановления системы после отказов; последствий влияния отказов системы или отказа наиболее важных функций; допустимого объема данных, утрачиваемых в случае отказа; ориентировочного числа ошибок в ПО, их типа и допустимого времени обнаружения; перечня дополнительных функций, необходимых для обнаружения, исправления, обеспечения устойчивости к ошибкам, восстановления работоспособности;

б) перечня предоставляемых пользователю сервисных функций ПО;

в) степени адаптируемости (возможности расширения ПО при добавлении ряда функций);

г) удобства сопровождения (затраты времени и средств на исправление ошибки в работающей системе ПО);

д) безопасности (защита и изоляция данных и программ разных пользователей друг от друга и от операционной системы при попытках вмешательства или разрушения ПО);

е) стоимости (затраты на первоначальную разработку ПО и его сопровождение в процессе эксплуатации);

ж) календарного плана (получение результата к конечному сроку) и документации (качество, количество и тип публикаций для каждого вида пользователей);

з) показателей эффективности или производительности системы ПО (временны́е характеристики, пропускная способность, использование ресурсов, требуемые средства измерения производительности).

На третьем этапе осуществляется внешнее проектирование ПО, т. е. конструирование его внешних взаимодействий без конкретизации внутренней структуры, но с подготовкой полных и правильных спецификаций аппаратуры и базового ПО.

На четвертом этапе проектирования ПО осуществляется проект каждой функции пользователя с подробным описанием:

— входных данных с описанием их синтаксиса (формат, допустимые значения, области изменения) и семантики (смыслового значения);

— выходных данных с описанием результатов всех функций и связей между выходными и входными данными при правильном и ошибочном их задании;

— всех состояний и преобразований информации;

— характеристик надежности системы (влияния разнообразных отказов функций на систему, файлы и пользователя);

— характеристик эффективности системы (затраты времени, ресурсов памяти) с учетом стандартных конфигураций аппаратуры и ПО, интерфейса, числа пользователей и др.;

Составление полных и точных внешних спецификаций с отображением всевозможных входных данных (допустимых и недопустимых) и соответствующих реакций системы, а также внешних сопряжений имеет большое значение для дальнейшего проектирования ПО. Исходными данными на третьем и последующих этапах являются: публикации по использованию ПО; спецификации аппаратуры, описывающие особенности функционирования применяемых аппаратных средств (ЦП, памяти, УВВ, интерфейса) с указанием технических и временны́х характеристик, уровней сигналов управления, сопряжения и синхронизации; спецификации используемого языка программирования и базового ПО, с которым взаимодействуют прикладные программы, по запросам на ввод — вывод и динамическое распределение ресурсов. Неправильное понимание документации по базовому ПО и спецификациям аппаратуры, а также синтаксиса и семантики языка программирования является причиной ошибок в ПО.

На пятом этапе разрабатываются архитектура программного комплекса с декомпозицией его на более мелкие компоненты (подсистемы, программы и модули) и соответствующие им спецификации.

Результатом разработки архитектуры ПО является документация по описанию основных частей ПО (компонентов, подсистем, программ, модулей), их функций и сопряжений с указанием иерархии передач управления возвратом; структуры потоков данных в системе; иерархии задач (параллельных процессов); структуры памяти в системе.

На шестом этапе проектируется структура ПО. Исходными данными здесь являются: детальные внешние спецификации и архитектура ПО (для крупного проекта); детальные внешние спецификации и соответственно единая программа или несколько модулей этой программы (для мелкого проекта). Конечным результатом этапа является структура всех компонентов ПО с определением всех его модулей, их иерархии и сопряжений между ними. Структура ПО отражает отношения между всеми модулями (кто кого вызывает), функции каждого модуля и сопряжения между ними.

На седьмом этапе проектирования определяются внешние спецификации каждого программного модуля с указанием их имени, выполняемых функций, списка входных и выходных параметров, внешних ситуаций. Внешние спецификации модулей целесообразно помещать в виде комментариев в начале текста исходной программы каждого из модулей, а при наличии модулей с несколькими входами — у каждого из них.

На восьмом этапе проектируется и собственно программируется (кодируется) внутренняя логика каждого программного модуля на одном из языков программирования.

В целом проектирование ПО является сложной задачей, тесно связанной с использованием принципов обеспечения корректности функционирования программ, предупреждения, обнаружения и исправления в них ошибок, создания устойчивого к ошибкам ПО.

Рассмотрим особенности методов и средств обеспечения надежности и качества ПО, применяемых на этапах проектирования.

Обеспечение надежности ПО на этапе определения требований и спецификаций. Большое влияние на эффективность, качество и надежность функционирования будущего ПО оказывают решения, применяемые на этапе определения требований и спецификаций ПО.

Прежде чем составить машинную программу, необходимо понять и каким-либо образом описать решаемую задачу, т. е. ее специфицировать (слово «спецификация» означает «описание», «получение описания»). Однако так как сама программа является некоторым описанием задачи, то необходимо знать, чем спецификация отличается от программ, зачем она нужна, какую роль играет и какое место занимает в разработке и жизненном цикле программ, какая информация о задаче и в какой форме должна содержаться в спецификации.

В целом спецификация в отличие от программы должна говорить, что надо сделать, а не как это делать, она не должна содержать деталей реализации и навязывать программисту свое решение. Спецификация должна быть более понятным, полным, точным, легким в написании и чтении описанием задачи, чем программа. В то же время спецификация должна отличаться от менее полного и точного, более эскизного предварительного описания, называемого *требованиями к программе* и включенного в ЕСПД в качестве раздела технического задания.

Таким образом, спецификация — это достаточно точное и полное описание задачи, которое человеку, участвующему в решении, легче написать, понять и прочесть, чем программу решения этой задачи на доступном ему языке программирования. Средства спецификации — это любые средства получения или построения таких описаний, а язык спецификаций — рационально организованный, синтаксически оформленный набор таких средств, как правило, выразительный язык более высокого уровня, чем традиционный язык программирования. В перспективе языки спецификаций — это будущее языков программирования [41].

Разрабатываемая документация по определению полного перечня требований к ПО (см. табл. П.III.3) должна: а) задавать только внешнее поведение системы без уточнения какой-либо конкретной реализации; б) задавать ограничения на реализацию, особенно в части интерфейса с аппаратурой; в) служить справочным материалом для опытных программистов, давать ответы на конкретные вопросы без общих объяснений действий программы; г) предвидеть дальнейшие изменения, возможность исключения ряда функций; д) описывать допустимые реакции на нежелательные события, сбои аппаратуры и ошибки пользователя.

В процессе проектирования и разработки программ спецификации занимают промежуточное положение между эскизными требованиями и готовой программой. Различают «внешние» спецификации, обращенные к потребителю программы, внешнему пользователю, и «внутренние» спецификации, обращенные к разработчику программы. Первые соответствуют исходным спецификациям, а вторые — это одна из форм промежуточных спецификаций.

В целом подготовка полных и правильных спецификаций считается самой ответственной задачей в разработке ПО, так как они: а) служат заданием на разработку программ, являются частью соглашения между заказчиком и разработчиком ПО, описанием задачи для заказчика, не знающего программирования; б) используются при модульном построении программ, при проверке готовых программ с целью выяснения полноты решения поставленной задачи; в) облегчают сопровождение программ, их понимание и внесение необходимых изменений при модификации.

В спецификациях программ выделяют: а) *функциональную спе-*

цификацию, описывающую набор функций программы, объектов, участвующих в задаче, разбиение задачи на иерархию подзадач, входные и выходные данные, связи между ними (для задач преобразования данных) и вычисление функций, взаимодействия с внешней средой, реакции на исключительные ситуации (для задач управления); б) *эксплуатационную спецификацию,* включающую в себя вопросы производительности программы, используемые ею ресурсы, характеристики применяемой аппаратуры, специальные требования к надежности и безопасности. В целом считается, что точности спецификаций или языка спецификаций не обязательно добиваться путем полной формализации, а достаточно применения некоторых средств, позволяющих анализировать *семантическую* сторону спецификаций [41].

Известны три класса спецификаций и соответственно средств их построения: алгебраические, аксиоматические и модельные. Наиболее распространены на практике модельные средства, включающие в себя: таблицы; равенства и системы подстановок; логические средства; графы, сети и диаграммы; математические структуры и модули; типы, схемы и фреймы; операции, процедурные средства; средства именования; средства описания исключительных и ошибочных ситуаций.

В зависимости от особенностей решаемых задач при создании спецификаций ПО могут использоваться: а) языки описания подсистем обработки данных (например, AXES, SA), б) языки для «детального» описания (например, алгебраические спецификации абстрактных типов данных); в) языки описания «архитектуры» с выделением языков, ориентированных на данные (например, HIPO, PSL), языков, ориентированных на управление (LOGOS, R-язык, PAD, язык конечно-автоматных диаграмм), и языков, ориентированных на описание асинхронных параллельных систем и систем реального времени (семафоры Дейкстры, мониторы Хоэра и Бринк-Хансена, языки АДА, Паскаль и др.). В то же время разработчиком спецификаций лучше применять более простые понятийные средства, позволяющие непосредственно и естественно описывать решаемые задачи в терминах, соответствующих природе задач.

Специалисту (математику-спецификатору) нужно уметь расспрашивать заказчика с целью правильного понимания всех содержательных аспектов задачи и уметь объяснять заказчику и программисту их интерпретацию в виде принимаемой математической модели. Так как таким умением обладает далеко не каждый специалист, то следует изучать особенности языков спецификаций и применять наиболее подходящие из них для свободного понимания написанных спецификаций каждым из участников проекта с целью быстрого контроля их правильности и повышения качества и надежности ПО в конечном итоге. В помощь разработчикам спецификаций ПО разработан ряд справочников, содержащих примеры,

конкретные свойства и рекомендации по использованию спецификаций, их составлению и обнаружению в них ошибок.

Обеспечение надежности ПО на этапах проектирования и программирования. Начинающие разработчики ПО МПСУ основное внимание уделяют этапу записи алгоритма функционирования проектируемых устройств на одном из языков программирования, хотя он является конечным этапом в разработке ПО, обладает малой долей трудозатрат (до 10%) и степенью влияния на надежность получаемого ПО. В связи с этим приведем общие рекомендации по синтезу крупного проекта ПО, которые для конкретных условий могут быть избыточными.

На этапе постановки и описания задачи рекомендуется:

— иметь прочную связь разработчика с пользователем при принятии решений по определению требований, целей и внешних характеристик систем ПО;

— разработчику глубоко изучить особенности по эксплуатации и применению будущего ПО, а пользователю учитывать целесообразность последующих изменений сформулированных требований и целей ПО;

— группа разработчиков ПО должна быть небольшой: по два человека от организации-пользователя (один — с полномочиями принятия решений, другой — будущий пользователь) и от организации-разработчика (один — главный специалист по внешнему проектированию, второй — по внутреннему проектированию ПО);

— разработчикам проявлять максимум аккуратности и точности в работе, особенно при определении требований пользователя к ПО с целью внесения в проект минимума ошибок.

На этапе перевода требований пользователя в наборы конкретных целей рекомендуется:

— добиться четкого, ясного задания целей ПО, достаточно подробно сформулированных, с указанием приоритета для каждой цели, а также указания нежелательных ситуаций и допустимых последствий при отказе наиболее важных функций;

— учитывать вопросы совместимости разрабатываемого ПО с другими системами ПО; конфигурацию аппаратных и программных средств, в которых ПО может функционировать; методы и средства тестирования и настройки системы на конкретные условия эксплуатации.

На этапе внешнего проектирования ПО рекомендуется:

— использовать принцип концептуальной целостности ПО, стремиться иметь небольшой набор хорошо согласованных функций;

— иметь одного-двух специалистов по внешнему проекту и составлению спецификации ПО, знакомых со всеми фазами проек-

тирования и тестирования ПО, работающих на ЭВМ и хорошо разбирающихся в обстановке пользователя;

— во внешних спецификациях (аппаратуры, базового ПО, языка программирования) описывать все возможные входные данные системы (допустимые и запрещенные) и соответствующие реакции системы, а также все внешние сопряжения.

Минимизации ошибок пользователя можно добиваться путем:

— соответствия способа взаимодействия в системе ПО с уровнем подготовки и квалификацией пользователя; ввода пользователем в систему максимально коротких сообщений с соблюдением единообразия их форматов и типов;

— выдачи на каждое входное сообщение пользователя подтверждения о его принятии;

— применения набора специальных сервисных функций, помогающего пользователю в затруднительных ситуациях; выдачи корректных ответов из системы на языке пользователя при ошибочных и неожиданных сообщениях пользователя.

Обнаружения ошибок пользователя можно добиваться за счет приема системой любых данных, но при обнаружении среди них недопустимых, неполных или противоречивых информировать об этом пользователя; при вводе сложных закодированных с многократными обращениями к системе сообщений предоставлять пользователю возможность проверки правильности и коррекции этих сообщений перед их обработкой в системе; точности и аккуратности представления входных данных в системе с введением при необходимости требуемой доли избыточности.

На этапе проектирования архитектуры и структуры ПО рекомендуется:

— для рационального распределения функций и связей между компонентами ПО (подсистемами, программами, модулями), оптимизации системных ресурсов и функциональной связности программных модулей использовать метод композиционного проектирования с оценкой качества программных модулей по шкалам прочности и сцепления;

— добиваться, чтобы максимум модулей ПО обладал функциональной прочностью (выполнял одну определенную функцию) и сцеплением по данным (минимальным сцеплением с другими ПМ);

— стремиться к небольшому размеру модулей (от 10 до 100 операторов языка высокого уровня), их предсказуемости (независимости от предыстории пользования), минимизации объема данных, использованию в модулях внутренних процедур (программ) с оформлением их в виде отдельных ПМ.

Построение структуры ПО можно вести с использованием метода «сверху вниз», для чего:

— представить структуру задачи тремя — десятью процессами;

— определить главные входные и выходные потоки данных;

— проследить по структуре задачи и выделить точку изменения формы входного потока;

— аналогично проанализировать выходной поток данных с конца структуры задачи до точки превращения выходного потока в абстрактную форму, отметить ее;

— представить три части структуры, образованные найденными точками, как три функции и определить модули, выполняющие каждую из этих функций;

— определить сопряжение этих модулей и вид данных на входе и выходе;

— разбиение задачи повторить для более низких уровней иерархической структуры, пока не определится логика каждого из модулей;

— построить структурную схему ПО с отражением упорядочения между всеми модулями (кто кого вызывает), функций каждого из модулей и сопряжения между ними.

На этапе проектирования и программирования (кодирования) каждого программного модуля рекомендуются изложенные ниже мероприятия.

1. Тщательно планировать процесс разработки логики модуля, для чего:

— выбор языка проводить на начальном этапе разработки ПО с учетом требований технического задания, принятых стандартов, необходимости обучения программистов, подготовки компиляторов, средств тестирования и др.;

— проверку правильности внешних спецификаций ПМ проводить на основе сравнения с информацией, полученной при разработке структуры ПО, с анализом их корректности всеми программистами, разрабатывающими вызывающие модули;

— для алгоритмов и структур данных осуществлять поиск готовых решений с выбором из них наиболее рациональных и простых (особенно для систем с многоуровневой памятью).

2. При разработке программных модулей главное внимание уделять высокому качеству программ, (а не искусству программирования), для чего следует:

— стремиться к простой, ясной, а не к сложной программе, сначала записывая ее на легко воспринимаемом псевдоязыке, а затем переводя на требуемый язык программирования;

— не останавливаться на первом варианте программы, переписывая заново, а не исправляя неудачные участки программы;

— выбирать способ представления данных, упрощающий программу; инициализацию переменных проводить до их использования;

— обеспечивать возможность нормального завершения программы при неверном задании исходных данных, их распознавание и (при возможности) исправление;

— для сокращения времени работы программы искать лучший алгоритм, а не тратить времени на изменение ее хода;

— обеспечивать соответствие комментария тексту программ; он должен пояснять смысл выполняемых условий, а не повторять текст программы;

— идентификаторы переменных и метки в программе должны нести смысловую нагрузку.

3. Быстродействия и эффективности организации программ добиваться оптимальным использованием регистров, проверкой особых случаев, вынесением безусловных переходов и операций с результатами, не зависящими от работы цикла, за его пределы, а также объединением проверок многих условий в одну проверку и др.

4. При составлении программ на языках высокого уровня рекомендуется:

— использовать содержательные имена, а не их коды, что облегчает чтение программы при минимуме комментариев;

— не использовать ключевые слова языка в качестве идентификаторов, при необходимости использования в идентификаторах цифр помещать их только в конце; во избежание неоднозначности употреблять скобки;

— внимательно следить за использованием констант как аргументов;

— каждое предложение с объявлением некоторой переменной комментировать с объяснением ее смысла;

— активно использовать рекурсию, внутренние процедуры, библиотечные и встроенные функции;

— внимательно изучать в руководствах по используемому языку главу о распространенных ошибках;

— все переменные, а также все атрибуты каждой переменной объявлять явно; не применять переменную более чем для одной цели, а также значения переменных с особым смыслом;

— не использовать несколько имен для одной области памяти, а также операций со значениями смешанных типов; не беспокоиться о повышении эффективности программы до тех пор, пока программа не будет правильной; не жертвовать легкостью чтения ради эффективности; никогда не оптимизировать программу без необходимости;

— добиваться эффективности за счет макроэффективности (правильной организации ввода — вывода, выбора оптимальных алгоритмов и структур памяти, размещения модулей и размеров рабочего множества программ и др.), а не микроэффективности (исключения индексации, замены возведения в степень умножением, поиска быстрого обнуления регистров, программирования на машинном языке и т. п.); добиваться эффективности на основе измерений, а не догадок.

Заботясь о надежности ПО, следует помнить о позиции (или психологии) программиста; обычно программист подсознательно считает недостатки программы проявлением собственных недостатков. Поэтому рекомендуется применять принцип безличного программирования, когда считается, что общий успех в создании ПО зависит от минимума числа ошибок, поэтому каждый программист стремится показать свою программу коллегам для конструктивной критики и нахождения ошибок в ней с целью исключения их повторения в других разработках.

§ 7.4. ПРОГНОЗИРУЮЩИЕ И ЭКСПЕРИМЕНТАЛЬНЫЕ МОДЕЛИ ОЦЕНКИ НАДЕЖНОСТИ ПО

Проектные, или прогнозирующие, модели оценки надежности программных средств применяются для оценки качества разработанного ПО на различных этапах его создания при наличии определенного состава исходных данных. К данным о ПО, доступным на этапах проектирования и программирования, относятся: уровень квалификации программистов, используемый язык программирования, общее число операторов и операндов (длина ПО), разнообразие операторов и операндов (словарь ПО), число логических операторов, число ветвей в ПО и др. К данным о ПО, доступным на этапах отладки и тестирования, относятся: длительность тестирования, количество ошибок, обнаруженных и устраненных при тестировании, уровень отлаженности ПО и др.

Вопросам оценки надежности ПО с подробным изложением существующих методов посвящены, например, отдельные работы [5, 31, 38, 40, 43]. Ниже рассмотрим ряд моделей оценки надежности программ на ранних этапах их создания.

Модели оценки надежности ПО, применяемые на этапе проектирования. В зависимости от сложности ПО может рассматриваться двояко: а) как единое целое, не расчленяемое на отдельные программы, подпрограммы и модули, с оценкой его надежности на основе общих характеристик самого ПО и условий его разработки; б) как программная система, состоящая из более мелких модулей, для которых известны их надежностные (и некоторые другие) свойства и взаимосвязи, определяющие результирующие надежностные свойства исследуемого ПО в целом.

Проектные модели оценки надежности проектируемого ПО как единого целого. Определение надежности ПО как единого целого ведется на основе некоторой статистической или детерминированной математической модели надежности, связывающей тот или иной показатель надежности ПО с общими характеристиками ПО и условиями разработки. Для получения интегральной оценки надежности программной системы следует выбрать состав показателей надежности для каждого из используемых элементов, применить один из способов формального

описания надежностной структуры ПО с отображением взаимосвязи его элементов в обеспечении надежности, произвести расчет показателей надежности.

Так как показатель устойчивости ПО задается лишь списками «слабых» и «сильных» допустимых возмущений, на которые разрабатываемое ПО должно реагировать определенным образом, то он не может прогнозироваться. Показатели корректности ПО — среднее число ошибок в ПО, вероятность отсутствия ошибок определяются как по данным, получаемым на этапах проектирования и программирования, так и по данным, получаемым на этапах тестирования и отладки ПО.

Наиболее простая приближенная модель зависимости числа ошибок в ПО от числа операторов в программе взята из практического опыта многих поколений программистов: число ошибок в сложных программах составляет 1—2% от общего числа объектных команд в программе. При тщательном системном проектировании и программировании на ЯВУ число ошибок в ПО уменьшается в несколько раз. При программировании на ассемблере на каждую тысячу команд первого варианта исходной программы приходится от 3,36 до 7,98 ошибок.

В [43] предложены следующие две модели оценки надежности ПО, связывающие ряд характеристик ПО с количеством ошибок в ПО $\hat{r}$:

$$\hat{r}_1 = \frac{1}{3000}(N_1 + N_2)\log_2(n_1 + n_2);$$

$$\hat{r}_2 = \frac{1}{3000}\left[(N_1 + N_2)\log_2(n_1 + n_2)\frac{n_1 N_2}{2n_2}\right]^{2/3}, \qquad (7.2)$$

где N_1, N_2 — общее число операторов и операндов в ПО; n_1, n_2 — общее число различающихся операторов и операндов в ПО.

Эти модели (более простая и более сложная) позволяют получить оценку сверху ($\hat{r}_1$ и $\hat{r}_2$ соответственно) и характеризуют корректность ПО до начала его тестирования. Чем ниже уровень языка, тем в бо́льшей степени оценки $\hat{r}_1$ и $\hat{r}_2$ уклоняются от истинного значения в сторону увеличения.

В [42] на основании исследований сложных программных комплексов по 12 основным характеристикам выделяются два показателя с наиболее сильной корреляцией относительно числа ошибок: а) число ветвей программы — $z_в$; б) число интерфейсов программы — $z_с$. При этом число ошибок $\bar{r}$ оценивается следующей зависимостью:

$$\bar{r} = 0{,}0454 z_в + 0{,}254 z_с. \qquad (7.3)$$

Если программу представить в виде графовой модели, то в ней можно определить [40]: число различающихся операторов в программе — Z_W; число операторов в кратчайшем маршруте (ветви)

программы — Z_L; число логических операторов — Z_T; максимальную степень захода для вершин графовой модели — Z_D; среднеквадратическое отклонение степеней вершин графа — Z_G; максимальное значение степени вершин графа — Z_S; общее количество операторов (команда) в программе W. Тогда число ошибок в ПО

$$\bar{r}=\bar{\theta}_{\text{ош}}W=e^{0,429}Z_W^{0,273}Z_L^{0,48}Z_T^{0,129}Z_D^{-0,125}Z_G^{-0,164}Z_S^{0,036}W, \qquad (7.4)$$

где $\bar{\theta}_{\text{ош}}$ — удельное число ошибок в программе на тысячу операторов в программе.

Проектные модели оценки надежности ПО, основанные на данных о надежности его элементов и структур. Исходными данными при такой оценке ПО являются надежностные свойства элементов (модулей) программной системы и характер их взаимосвязей. Если всю программу разделить на процедурные и логические операторы и по каждому процедурному оператору задать показатели $\bar{r}$, $K_{\text{о.о}}$ и L_i, то тогда для всей программы можно записать, что

$$\bar{r}=\sum_{i=1}^{N_n}\bar{r}_i;\quad K_{\text{о.о}}=\prod_{i=1}^{N_n}K_{\text{о.о}i}, \qquad (7.5)$$

где N_n — общее число процедурных операторов.

При заданной статистике задач для каждой ветви программы соответствует свое значение вероятности ее использования $p_{\text{в}j}$ и вероятности успешной реализации $L_{\text{в}j}$:

$$p_{\text{в}\,j}=\prod_{л=1}^{N_{\text{в}\,j}}p_л;\quad L_{\text{в}\,j}=\prod_{j=1}^{Z_{\text{в}j}}L_{sj}, \qquad (7.6)$$

где $N_{\text{в}j}(Z_{\text{в}j})$ — общее число логических (процедурных) операторов, входящих в j-ю ветвь программ; $p_л$ — вероятность i-го логического оператора в j-й ветви, определяющая выход в данную ветвь: L_s — показатель корректности s-го процедурного оператора в j-й ветви, $j=\overline{1,\ m_{\text{в}}}$.

Зная вероятность выхода $p_{\text{в}i}$ на i-ю ветвь программы и вероятность успешного решения задачи $L_{\text{в}i}$, вышедшей на эту i-ю ветвь, находим общую вероятность программной системы путем усреднения вероятностей по всем ветвям программы:

$$L=\sum_{i=1}^{m_{\text{в}}}p_{\text{в}i}L_{\text{в}i}. \qquad (7.7)$$

Чтобы определить необходимые исходные данные, например $p_{\text{в}i}$, требуется наблюдение и анализ реального потока задач. Однако из-за сложности реальных задач, насчитывающих большое число (сотни и тысячи) ветвей в программах, требуется переход

к решению на ЭВМ с использованием аналитико-машинных методов или методов вероятностного моделирования.

Алгоритм аналитико-машинного метода должен включать в себя: определение числа ветвей $m_в$ в программе и состава процедурных операторов L_s в каждой ветви; вычисление вероятностей выхода на каждую ветвь программы $p_{вi}$ по формуле (7.6); вычисление показателей корректности для всех ветвей программы $L_{вj}$ по формуле (7.6); вычисление заданного значения L по формуле (7.7).

Данный подход позволяет учитывать не только собственные ошибки в программе, но и сбои технических средств, на основе которых реализуются те или иные элементы структуры ПО. Это происходит, если в формуле (7.6) величину L_s заменить на $L_s e_s$, где ε_s — вероятность того, что при реализации программного элемента не возникнут сбои соответствующих технических средств.

Экспериментальные методы оценки надежности ПО. Экспериментальные методы оценки реально достигнутого уровня надежности ПО дополняют проектные оценки надежности ПО, получаемые аналитическими методами или методами моделирования. Они предполагают проведение специальных испытаний (активный эксперимент) или сбор и обработку данных о надежности в нормальных условиях эксплуатации ПО (пассивный эксперимент). Наиболее эффективным экспериментальным методом оценки надежности ПО является метод организации и проведения специальных испытаний.

Для экспериментальной оценки показателей устойчивости ПО, количественно определяемой объемом и содержанием двух списков допустимых внешних возмущений (списка «слабых» допустимых возмущений, при которых ПО должно обеспечивать успешное решение задачи; списка «сильных» допустимых возмущений, которые должны распознаваться ПО и выводить его в состояние «защитного» отказа), рекомендуется [38]:

1. В каждом из этих списков выделять две части: а) возмущения, задаваемые на некоторых исходных данных и определяемые на входе ПО; б) возмущения, вызываемые сбоями в работе технических средств и воздействующие на ПО в процессе решения задачи.

2. Осуществлять проверку соответствия фактической реакции ПО на допустимые возмущения плановой реакции, предусмотренной спецификацией на ПО, для чего:

— первую часть списков (возмущений, задаваемых на векторах входных данных) проверять следующим путем:

— подбирать задачи с векторами исходных данных, соответствующими допустимым возмущениям («слабым» или «сильным» в зависимости от требований);

— по спецификации определять правильный результат решения каждой задачи;

— отобранные задачи последовательно подавать на вход ПО;

— результаты, получаемые на выходе ПО, фиксировать и сопоставлять с правильными результатами решения;

— при совпадении пары результатов решения считать, что требования к устойчивости по данной строке списков выполняются, в противном случае — не выполняются.

Из-за непредсказуемого влияния на устойчивость ПО допустимых возмущений, обусловленных действием технических средств, реакцию ПО на такого рода возмущения следует проверять многократно по каждой строке списков, вводя эти возмущения в случайные моменты времени.

Для экспериментальной оценки показателей корректности ПО (показателей $\bar{r}$ и $K_{\text{о.о}}$) рекомендуется использовать метод Миллса. Он заключается в том, что в исследумое ПО вносится и случайным образом размещается определенное число s искусственно создаваемых ошибок в дополнение к r собственным ошибкам ПО, после чего по обычной методологии проводится тестирование.

Считается, что собственные и внесенные ошибки в ПО распределены одинаково при равной вероятности их обнаружения. Тогда на основании методов максимального правдоподобия делается вывод об общем числе собственных ошибок $\hat{\bar{r}}$ в ПО:

$$\hat{\bar{r}} = sx/v, \tag{7.8}$$

где s — число дополнительно внесенных ошибок; x — число собственных, v — число внесенных ошибок, обнаруженных в процессе тестирования.

Точность оценки $\hat{\bar{r}}$ будет тем выше, чем больше будет внесено дополнительных ошибок s и обнаружено собственных x ошибок ПО при тестировании. При этом тестирование проводится до тех пор, пока не будут обнаружены все ошибки.

Мера доверия к такой модели C_k имеет под собой прочное статистическое обоснование, выражаемое зависимостью

$$C_k = \begin{cases} 1 \text{ при } n > k, \\ s/(s+k+1) \text{ при } n \leqslant k, \end{cases} \tag{7.9}$$

где n — число обнаруженных собственных ошибок в процессе тестирования; k — число предполагаемых в ПО собственных ошибок перед началом тестирования.

Если же тестирование не доводить до полного обнаружения всех внесенных в ПО ошибок ($j<s$), то

$$C_k = \begin{cases} 1 \text{ при } n > k, \\ \binom{s}{j-1} \Big/ \binom{s+k+1}{k+j+1} \text{ при } n \leqslant k. \end{cases} \tag{7.10}$$

При этом получается, что

$$K_{о.о}=C_0, \text{ если } C_0=0,$$

$$K_{о.о}>C_0, \text{ если } C_0>0.$$

Недостатком метода Миллса является предположение о равнораспределенности внесенных и собственных ошибок, что требует анализа свойств и статистики данных и ошибок в ПО. Кроме того, поток задач, подаваемых на ПО с целью тестирования, должен подбираться из условия равномерного охвата всех допустимых наборов исходных данных.

Экспериментально оценку вероятности L правильного решения по данным ПО произвольной задачи, взятой из потока реальных задач, при специальных испытаниях получить сложно, так как, во-первых, достигнуть внешних условий реального функционирования ПО невозможно, и, во-вторых, действия ПО при корректной обработке невозмущенных и сильно возмущенных данных существенно различаются, что обусловливает определение показателя L для каждой из трех областей исходных данных при разрешенных, слабых и сильных возмущениях соответственно; в-третьих, требуется создание специализированного имитатора (генератора) потока задач со статистическими свойствами, отвечающими свойствам реального потока задач, поступающих на вход ПО при его эксплуатации.

При отсутствии такого имитатора можно в нормальных условиях провести запись достаточно длинной последовательности задач, поступающих на вход рассматриваемого ПО. Затем в условиях специальных испытаний прогнать эту последовательность задач через ПО и провести оценку показателя L или определить статистические свойства записанной последовательности задач. И наконец, построить программный имитатор потока задач и провести испытания L со значительно бóльшим объемом статистических данных.

Относительно экспериментальной оценки времени восстанавливаемости $T_в$ и исправляемости $T_и$ ПО отметим следующее. Так как $T_в$ и $T_и$ — случайные величины, зависящие от уровня квалификации персонала ($T_в$ — это затраты времени оперативного персонала на перезапуск ПО после защитного отказа, а $T_и$ — это затраты времени программистов на внесение необходимых изменений в ПО и устранение обнаруженной ошибки), то на практике при нормальных условиях эксплуатации ПО такие ситуации происходят редко и затрудняют набор статистики. Поэтому прибегают к специальным испытаниям, когда защитные отказы в ПО вызываются искусственно. Более подробно эти вопросы рассмотрены в [38].

§ 7.5. СИНТЕЗ МПСУ С ПРИМЕНЕНИЕМ ПРОГРАММНОЙ ИЗБЫТОЧНОСТИ

Как отмечалось выше, надежные МПСУ можно проектировать на основе программной избыточности, вводимой в любую уже работающую систему, и создания программного обеспечения, позволяющего предупреждать, обнаруживать и исправлять ошибки.

Организация надежного функционирования таких МПСУ осуществляется поэтапно следующим путем:

а) на первом этапе для обнаружения ошибок в работе МПСУ применяется один или несколько программных методов контроля за правильностью протекания вычислительного процесса, подразделяющихся на алгоритмические и логические;

б) на втором этапе после обнаружения ошибки программным путем восстанавливается нормальное функционирование МПСУ. При этом восстановление осуществляется на основе применения многоуровневых итерационных структур. Уровни образуют такую последовательность программных процедур, что любой i-й уровень структуры позволяет осуществлять итерацию или повторение некоторых последовательностей, находящихся на $(i-1)$-м уровне, а его индивидуальный выход является входом для следующего уровня. Восстановление с помощью итерационных структур наиболее просто реализуется при глобальном подходе, когда при появлении ошибок осуществляется перепрогон программы.

Алгоритмические методы контроля заключаются в программной проверке правильности решения задач и предназначены для обнаружения и исправления случайных ошибок в процессе реализации алгоритмов. Один из них — метод дуального (N-версного) программирования, требующий создания двух (N) версий программы, выполняющей одну и ту же функцию. Версии должны по возможности отличаться алгоритмами, результаты выполнения которых сравниваются между собой. При этом происходит маскирование сбоев модулей за счет самокоррекции результата.

Логические методы программного контроля в силу своей простоты и относительной эффективности применяются в более широком спектре структур, чем алгоритмические методы. В практике проектирования МПСУ наиболее распространены следующие логические методы.

Метод обратного счета, заключающийся в нахождении исходных данных по вычисленному результату и сравнении их с начальными исходными данными. При совпадении с заданной точностью выполнение программы считается верным. С помощью данного метода в МПСУ можно обнаружить сбои и отказы. Ограничения метода состоят в том, что не все математические действия имеют обратные операции, и он приемлем только для небольших сегмен-

тов программ, так как для его реализации требуется время, вдвое превышающее время выполнения исходной программы.

Метод контрольного суммирования, обеспечивающий проверку сохранности различного рода констант, больших массивов информации, используемых в процессе выполнения программ. Полученная сумма сравнивается с заранее подсчитанным эталоном.

Метод проверки на допустимость в отличие от контрольного суммирования организует проверку того, укладывается ли значение контролируемой величины в заранее отведенный диапазон значений. Данный метод применим в тех случаях, когда переменные легко физически интегрируются и отвечают некоторым разумным или физически допустимым областям существования значений. Метод достаточно удобен и применим в МПСУ, работающих в режиме реального времени.

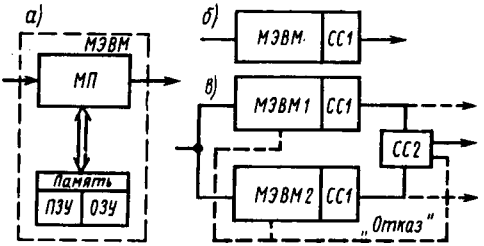


Рис. 7.2. Варианты структур МПСУ

Метод повторного счета позволяет обнаруживать и маскировать ошибки, возникающие при сбоях в работе МПСУ. При этом важен вопрос выбора уровня повторения с учетом требуемой достоверности результатов решения задачи. Необходимым условием правильного повторения любого фрагмента программы является сохранение исходной информации с помощью как повторяемых, так и неповторяемых команд. Данный метод наиболее прост в реализации, но требует удвоенного времени выполнения программ.

Рассмотренные методы могут применяться порознь или совместно в зависимости от конкретных требований и специфики эксплуатации программ. Например, для МПСУ, работающих в реальном масштабе времени, наиболее целесообразно использовать многопроцессорные структуры в совокупности с методами алгоритмического и логического контроля, в частности N-версного программирования, проверки на допустимость и повторного счета. Для просчета вычислений и фиксации времени выполнения программ могут применяться контрольные таймеры.

Ниже на конкретном примере рассмотрим влияние введения программно-аппаратной избыточности на надежность функционирования МПСУ.

Пример 7.1. Оценить влияние на быстродействие и надежность функционирования программно-аппаратных методов введения избыточности в структуру МПСУ при сравнительном анализе следующих вариантов алгоритмов и типов структур:

1) неизбыточной структуры МПСУ с отсутствием средств защиты от отказов и сбоев (рис. 7.2, *а*);

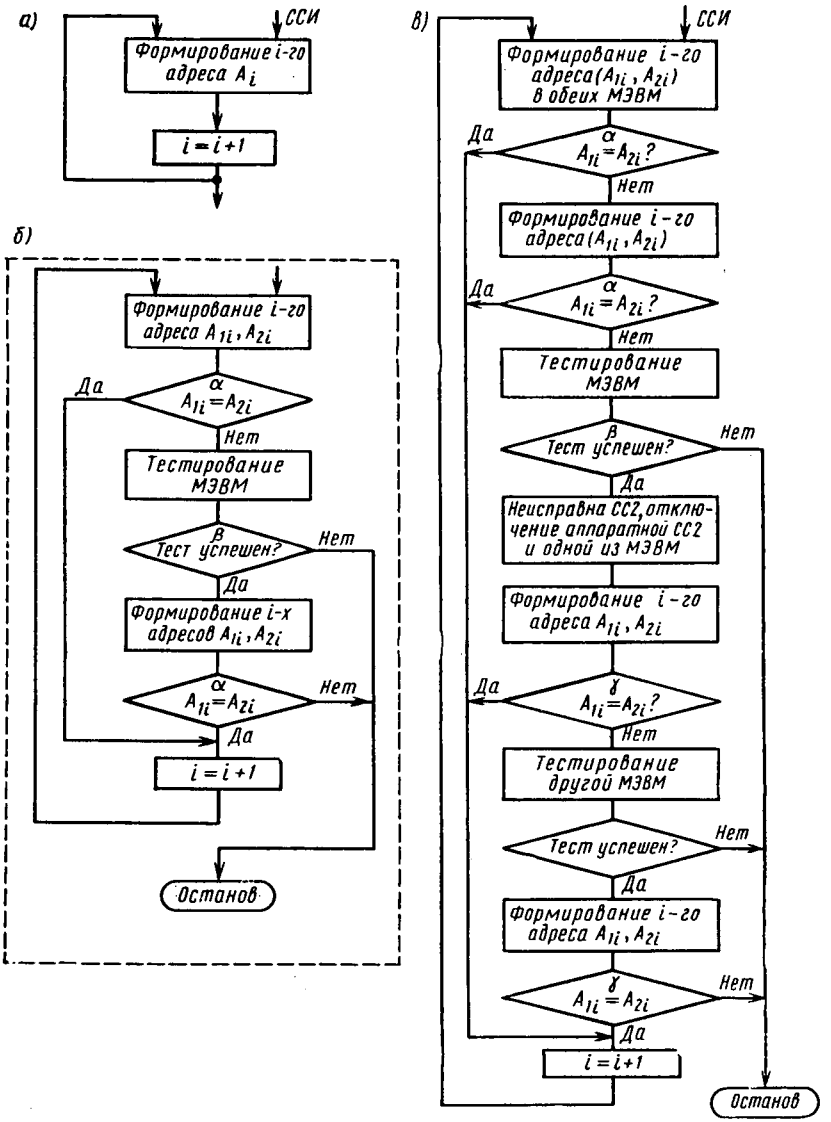


Рис. 7.3. Варианты алгоритмов функционирования МПСУ для структур рис. 7.2,

2) структуры МПСУ с использованием логического метода повторного счета для каждого адреса A_i (обращения к датчикам объекта управления) и программно реализованного сравнения результатов формирования адресов (рис. 7.2, *б*);

3) структуры МПСУ из двух дублированных МП с общей памятью и программно реализованным сравнением (ПС) результатов в каждом из МП, выполняющих формирование адресов по идентичным программам. Результаты вычислений каждого из МП сравниваются аппаратно реализованной схемой (АС) при нормальном функционировании МП (рис. 7.2, *в*). Предусматривается деградация структуры МПСУ путем отключения одного из МП и схемы АС при их отказе с переходом функционирования МПСУ по второму варианту.

Исходные данные для расчета надежностных показателей МПСУ при программной реализации алгоритмов формирования адресов A_i сведены в табл. 7.1. Произвести расчет: 1) вероятности безотказной работы $P(t)$ за период 50 000 ч; 2) вероятности правильного формирования одного адреса A_i $P_A(T)$; 3) среднего времени, отводимого на формирование одного i-го адреса.

Заметим, что время t_A одного прогона программы определяется как сумма времени выполнения каждой команды из программы формирования адресов A_i, т. е.

$$t_A = \sum_{l=1}^{n} t_l,$$

где n — общее число команд в программе.

Если в качестве МП использовать, например, К580ИК80, то тогда время прогона программы t_A соответственно составляет: для первого варианта МПСУ — 340 мкс; для второго — 350 мкс; для третьего — 360 мкс.

Приведем результаты расчета показателей надежности МПСУ. Первый вариант неизбыточной структуры МЭВМ без защиты от отказов и сбоев показан на рис. 7.3, *а*.

Вероятность правильного формирования одного адреса определяется как

$$P_A(T) = P(T)\,P_{\text{АЛГ}}(t_A),$$

где $P(T)$ — вероятность безотказной работы МПСУ за период времени $t \in [t_0, T]$; $P_{\text{АЛГ}}(t_A)$ — вероятность правильного формирования адреса за время t_A одного прогона программы.

Отсюда имеем:

$$P_A(T) = e^{-\left(\lambda^{o}_{\text{МП}} + \lambda^{o}_{\text{П}}\right)T}\, e^{-\left(\lambda^{o}_{\text{МП}} + \lambda^{c}_{\text{МП}}\right)t_A} =$$

$$= e^{-(1\cdot 10^{-7} + 1\cdot 10^{-7})\,50000}\, e^{-(1\cdot 10^{-7} + 1\cdot 10^{-5})\,340\cdot 10^{-6}} = 0{,}9900489923.$$

Второй вариант неизбыточной структуры МЭВМ с использованием логического метода повторного счета и программного сравнения результатов (рис. 7.2, *б*).

Согласно алгоритму (рис. 7.3, *б*), в такой структуре МЭВМ возможны следующие пути успешного формирования адреса:

1) цикл формирования каждого адреса дважды с программным сравнением результатов осуществляется без сбоев:

$$P_{\text{АЛГ1}}(t_{A1}) = e^{-\left(\lambda^{o}_{\text{МП}} + \lambda^{c}_{\text{МП}}\right)2t_A}\, e^{-\left(\lambda^{o}_{\text{МП}} + \lambda^{c}_{\text{МП}}\right)t_{\text{ср}}} = 0{,}9999999920715;$$

2) процесс начального цикла формирования адреса оказывается неудачным из-за сбоя МП, поэтому после получения неверного результата первый раз тес-

тируется МЭВМ с проведением повторного цикла формирования адреса и получения одинаковых результатов:

$$P_{\text{АЛГ2}}(t_{A2}) = e^{-\lambda^{\text{о}}_{\text{МП}} 2t_A}\left(1 - e^{-2\lambda^{\text{с}}_{\text{МП}} t_A}\right) e^{-\left(\lambda^{\text{о}}_{\text{МП}}+\lambda^{\text{с}}_{\text{МП}}\right) t_{\text{ср}}} \times$$

$$\times e^{-\left(\lambda^{\text{о}}_{\text{МП}}+\lambda^{\text{с}}_{\text{МП}}\right) t_T} e^{-\left(\lambda^{\text{о}}_{\text{МП}}+\lambda^{\text{с}}_{\text{МП}}\right) 2t_A} e^{-\left(\lambda^{\text{о}}_{\text{МП}}+\lambda^{\text{с}}_{\text{МП}}\right) t_{\text{ср}}} = 0{,}7 \cdot 10^{-8};$$

3) после проведения начального цикла формирования адреса во время операции программного сравнения результатов происходит искажение из-за сбоя МП, прежде чем получен окончательно правильный результат:

$$P_{\text{АЛГ3}}(t_{A3}) = e^{-\left(\lambda^{\text{о}}_{\text{МП}}+\lambda^{\text{с}}_{\text{МП}}\right) 2t_A}\left(1 - e^{-\lambda^{\text{с}}_{\text{МП}} t_{\text{ср}}}\right) e^{-\left(\lambda^{\text{о}}_{\text{МП}}+\lambda^{\text{с}}_{\text{МП}}\right) t_{\text{т}}} \times$$

$$\times e^{-\left(\lambda^{\text{о}}_{\text{МП}}+\lambda^{\text{с}}_{\text{МП}}\right)\left(4t_A+t_T+t_{\text{ср}}\right)}\left(1 - e^{-\lambda^{\text{с}}_{\text{МП}} t_{\text{ср}}}\right) = 0{,}85 \cdot 10^{-9}.$$

Отсюда вероятность успешного формирования адреса за один прогон программы равна

$$P_{\text{АЛГ}}(t_{\text{АЛГ}}) = P_{\text{АЛГ1}}(t_{A1}) + P_{\text{АЛГ2}}(t_{A2}) + P_{\text{АЛГ3}}(t_{A3}) =$$

$$= 0{,}9999999920715 + 0{,}7 \cdot 10^{-8} + 0{,}85 \cdot 10^{-9} = 0{,}9999999999215.$$

Среднее время формирования одного адреса для данной реализации МПСУ равно

$$t_{\text{ср}} = P_{\text{АЛГ1}} t_{A1} + P_{\text{АЛГ2}} t_{A2} + P_{\text{АЛГ3}} t_{A3} = 386 \text{ мкс}.$$

Итак, вероятность успешного формирования адреса на протяжении периода времени $t \in [t_0, T]$ равна

$$P_A(T) = P(T) P_{\text{АЛГ}}(t_{\text{АЛГ}}) = e^{-\left(\lambda^{\text{о}}_{\text{МП}}+\lambda^{\text{с}}_{\text{МП}}\right) T} P_{\text{АЛГ}}(t_{\text{АЛГ}}) =$$

$$= 0{,}9900489999995.$$

Третий вариант структуры МЭВМ с дублированными МП и общей памятью обеспечивает более гибкое формирование адресов A_i (рис. 7.3, *в*) по сравнению с другими. На первом этапе функционирования работают дублированные МП с выполнением идентичных программ формирования адресов и сравнением результатов с помощью аппаратной схемы совпадения АС. При совпадении адресов считается, что как в МП, так и в АС не было отказов или сбоев. В противном случае повторно выполняются программы формирования адреса и выполняется проверка на индентичность сформированных адресов. При их совпадении считается, что при формировании адреса первый раз имел место сбой и дальнейшее правильное функционирование системы возможно. При несовпадении повторно вычисленных адресов можно предположить, что в системе имели место: а) отказ какого-либо одного или двух МП; б) отказ АС; в) длительность сбоя превышает время двухкратного формирования и сравнения адресов (вероятность появления такого события очень мала).

Для идентификации отказа в МПСУ предусмотрено тестирование МП, в результате которого однозначно определяются отказавшие МП (при допущении, что тестовая программа позволяет определить отказ МП с вероятностью, равной 1). В том случае, если результат тестирования МП положительный, считается, что отказала АС. В дальнейшем структура переходит к функционированию на одном МП, причем алгоритм функционирования будет аналогичен описанному выше подпроцессу с использованием программных методов повторного счета и тестирования МП,

Таким образом, в избыточной структуре с дублированными МП и возможностью реконфигурации структуры вероятность формирования одного адреса равна

$$P_A(T) = P_1 P_{\text{АЛГ1}} + P_2 P_{\text{АЛГ2}},$$

где P_1 — вероятность того, что МПСУ успешно проработает на интервале $t \in [t_0, T]$ в основном (дублированном) варианте; P_2 — вероятность того, что МПСУ успешно проработает на интервале $t \in [t_0, T]$ в деградированном состоянии; $P_{\text{АЛГ1}}$, $P_{\text{АЛГ2}}$ — вероятности правильного формирования адреса при работе в первом и во втором состояниях соответственно за один прогон программы.

Таблица 7.1. **Исходные и результирующие расчетные показатели надежности структур МПСУ по критерию выполнения команд МП**

Показатели надежности МПСУ	Сравнительные структуры МПСУ рис. 7.2		
	вариант 1	вариант 2	вариант 3
Исходные показатели			
Интенсивность:			
отказа МП $\lambda^{o}_{\text{МП}}$, 1/ч	$1 \cdot 10^{-7}$	$1 \cdot 10^{-7}$	$1 \cdot 10^{-7}$
модуля памяти $\lambda^{o}_{\text{П}}$, 1/ч	$1 \cdot 10^{-7}$	$1 \cdot 10^{-7}$	$1 \cdot 10^{-7}$
сбоя МП $\lambda^{c}_{\text{МП}}$, 1/ч	$1 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$1 \cdot 10^{-5}$
отказа в схеме АС $\lambda^{o}_{\text{АС}}$, 1/ч	—	—	$5 \cdot 10^{-8}$
сбоя в схеме АС $\lambda^{c}_{\text{АС}}$, 1/ч	—	—	$5 \cdot 10^{-6}$
Время:			
функционирования МПСУ T, ч	50000	50000	50000
выполнения программы при формировании A_i-го адреса t_A, нс	340	350	360
сравнения адресов программным методом $t_{\text{ср}}$, нс	—	85	90
тестирования МП, $t_{\text{т}}$, нс	—	30	60
сравнения в схеме АС $t_{\text{АС}}$, нс	—	—	80
Объем требуемой памяти для реализации алгоритма контроля, байт	40	128	208
Расчетные показатели надежности МПСУ			
Вероятность безотказной работы структуры МПСУ $P(T)$ в интервале $[t, T_0]$	0,990049	0,990049	0,982652
Вероятность правильного формирования одного адреса $PA_i(T)$	0,990048992	0,990048999	0,9949504157
Среднее время формирования A_i-го адреса, мкс	340	386,0	365,9

Отсюда $P_1 = e^{-\left(\lambda^{o}_{\text{АС}} + 2\lambda^{o}_{\text{МП}} + \lambda^{o}_{\text{П}}\right) T} = 0{,}982652$.

Для определения P_2 необходимо рассмотреть следующие возможные ситуации, приводящие к деградации дублированной системы МПСУ:

1) отказал один МП в момент времени t, схема совпадения АС проработала правильно до момента времени t, оставшийся МП успешно проработал время $\tau \in [t, T]$, модуль памяти не отказал за время $\tau \in [t, T]$;

2) два МП успешно проработали до момента времени t, произошел отказ схемы совпадения АС, тестирование не отказало до момента времени t, в результате переходим к одному из двух работоспособных МП, который успешно доработал период времени $\tau \in [t, T]$, модуль памяти не отказал за время $\tau \in [t, T]$.

Опуская промежуточные расчеты, имеем, что $P_2 = 0{,}01229858$.

В алгоритме работы МПСУ при определении $P_{\text{АЛГ1}}$ возможны шесть путей успешного формирования и при определении $P_{\text{АЛГ2}}$ — два пути успешного формирования адресов A_i, вытекающие из вышеописанных особенностей функционирования.

В результате подстановки числовых данных получаем

$$P_{\text{АЛГ1}} = 0{,}999999999983; \quad P_{\text{АЛГ2}} = 0{,}999999999915.$$

Следовательно, вероятность правильного формирования одного адреса

$$P_A(T) = P_1 P_{\text{АЛГ1}} + P_2 P_{\text{АЛГ2}} = 0{,}982652 \cdot 0{,}9999999999983 +$$

$$+ 0{,}01229858 \cdot 0{,}999999999915 = 0{,}994950415730.$$

Среднее время формирования i-го адреса $t_{\text{ср}} = P_1 t_{\text{АЛГ1}} + P_2 t_{\text{АЛГ2}}$, где $t_{\text{АЛГ1}}$ — время формирования адреса в дублированной структуре, $t_{\text{АЛГ2}}$ — время формирования адреса в деградированной структуре:

$$t_{\text{АЛГ1}} = \sum_{i=1}^{6} P_{\text{АЛГ1}}\, t_{A1i} = 362{,}5 \cdot 10^{-6} \text{ с},$$

$$t_{\text{АЛГ2}} = \sum_{i=1}^{2} P_{\text{АЛГ2}}\, t_{A2i} = 786{,}989 \cdot 10^{-6} \text{ с}.$$

Отсюда получаем

$$t_{\text{ср}} = (0{,}982652 \cdot 362{,}5 + 0{,}01229858 \cdot 786{,}989)\, 10^{-6} = 365{,}889 \cdot 10^{-6} \text{ с} \approx 365{,}9 \text{ мкс}.$$

Таким образом, из расчета показателей надежности сравниваемых структур МПСУ следует, что с точки зрения вероятности правильного формирования адресов лучше всего выбирать структуру МПСУ по третьему варианту, хотя по значению среднего времени формирования адреса она незначительно уступает безызбыточному варианту.

Контрольные вопросы

1. Что подразумевается под качеством и надежностью ПО? В чем их отличие? Перечислите основные критерии и факторы, влияющие на надежность и качество ПО. **2.** Какие факторы учитываются при оценке сложности разрабатываемого ПО? **3.** Перечислите основные технологические и организационные методы и средства обеспечения надежности и качества программ. **4.** Какие основные показатели учитываются при оценке уровня надежности ПО МПСУ? Поясните их сущность. **5.** Перечислите основные этапы проектирования ПО. Какими способами обеспечивается надежность ПО на этапах: а) определения требований и спецификаций ПО; б) проектирования ПО; в) программирования ПО; г) тестирования и отладки программ. **6.** Какие прогнозирующие и экспериментальные модели оценки надежности ПО применяются на этапах: а) проектирования программ; б) тестирования программ; в) отладки и сопровождения? Дайте математическую интерпретацию этих моделей. **7.** Какими способами обеспечивается отказоустойчивость в МПСУ, синтезируемых на основе программной избыточности? Поясните на примерах.

ГЛАВА 8

ПРОЕКТИРОВАНИЕ СИСТЕМ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ ДЛЯ СЛОЖНЫХ ТЕХНИЧЕСКИХ ОБЪЕКТОВ И ПРОИЗВОДСТВ

Среди множества дискретных систем важное место занимают системы логического управления (СЛУ) и их более простые компоненты — цифровые управляющие автоматы (ЦУА), описание поведения которых базируется на алгебраических или графовых моделях и законах математической логики. Построение надежных, эффективно функционирующих СЛУ и ЦУА очень важно при автоматизации ответственных производственных объектов и технологических процессов. Особенно это относится к таким отраслям, как машиностроительный комплекс, космические и летательные аппараты, транспорт, энергетика, научное приборостроение, обеспечивающие создание новой техники, эффективных систем машин и приборов на базе вычислительных и микропроцессорных средств и современного технологического оборудования. Подобные объекты характеризуются большим числом единиц оборудования, дискретностью процессов, значительной размерностью (порядка сотен и тысяч входов и выходов) и сложностью систем управления, проектирование которых должно базироваться на теории дискретных процессов, учитывающей особенности современного производства.

Ниже применительно к специфике эксплуатации сложных технических объектов из области гибкого производства и бортовой автоматики рассмотрим особенности проектирования и обеспечения надежности СЛУ и ЦУА.

§ 8.1. ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ СЛОЖНЫХ СИСТЕМ УПРАВЛЕНИЯ

При создании сложных технических систем временна́я регламентация работ обусловливает выделение этапов проектирования, а детализация представлений о системах и подсистемах с отражением их свойств с той или иной степенью подробности — выделение уровней проектирования [50—53].

В качестве типовых этапов проектирования обычно выделяются:

— *этап научно-исследовательских работ (технического предложения)*, оканчивающийся формулированием принципиальных возможностей построения системы, определением ее основных физических и технических свойств, элементов структуры и ПО, анализом влияния разнообразных видов отказов на основные характеристики системы и ее компонентов;

— *этап эскизного проектирования*, включающий в себя детальную проработку возможности построения системы, укрупненную функционально-логическую и конструкторско-технологическую реализацию ее подсистем, синтез разнообразных устройств или применение типовых решений, обеспечивающих выполнение заданного закона функционирования и защиту от отказов и сбоев в условиях влия-

ния динамических воздействий и помех, эффективное преобразование информации и сопряжение подсистем друг с другом и с УВВ и магистралями, разработку структуры ПО;

— *этапы технического*, а затем и *рабочего проектирования*, заканчивающиеся тщательной проработкой принятых программно-аппаратных решений, схемотехники и конструкций отдельных устройств, блоков, подсистем и системы в целом с учетом заданных технико-экономических критериев оптимизации, технических и эксплуатационных ограничений;

— *этап рабочих испытаний*, оканчивающийся экспериментальной проверкой выбранных решений по системе, оценкой точностных, временных и надежностных характеристик на соответствие требованиям технического задания.

Уровни проектирования технических систем подразделяются на алгоритмический, структурный, функционально-логический, схемотехнический и конструкторско-технологический, причем на каждом из уровней решаются следующие вопросы:

— *на алгоритмическом уровне* — построения системы в целом, анализа алгоритмов управления и обмена информацией, определения необходимых (а при возможности оптимальных) временных соотношений;

— *на структурном уровне* — укрупненного анализа и выбора принципов организации и архитектуры системы, распределения и рационального соотношения аппаратных и программных средств, определения состава структурных блоков и программных модулей, способов их взаимодействия и реконфигурации, реализации требований, накладываемых на параметры основных подсистем, формирования частных ТЗ на их разработку (при необходимости);

— *на функционально-логическом уровне* — детализации по реализации функций отдельных подсистем и структурных блоков и их интерфейса, технической, алгоритмической и программной реализации каждого из блоков и всей системы с удовлетворением требованиям ТЗ;

— *на схемотехническом и конструкторско-технологическом уровнях* проектирования производится завершение разработки блоков и подсистем в целом с оформлением принципиальных электрических и другого рода схем, конструкторско-технологических чертежей и прочей необходимой документации.

Таким образом, на каждом из этапов проектирования систем решаются задачи всех уровней, но весомость этих решений различна: на этапе технического предложения это в основном задачи структурного уровня проектирования; на этапе эскизного проекта — задачи функционально-логического уровня; на этапе технического проекта — задачи схемотехнического и конструкторско-технологического уровней; на этапе испытаний — комплексная проверка принятых решений и качества реализации задач проектирования по всем уровням.

При разработке ответственных СЛУ промышленного и специализированного назначения (в дальнейшем просто СЛУ) особое внимание должно уделяться структурному и функционально-логическому уровням, так как на них формируются решения, определяющие качество и надежность системы, содержание и направление работ для остальных уровней. Заметим, что при проектировании СЛУ на основе программируемых БИС, МПК БИС и серийных микропроцессорных КТС — микроЭВМ, систем числового программного управления (СЧПУ), программируемых контроллеров (ПК) и других задачи структурного, функционально-логического и конструкторско-технологического уровней взаимосвязаны из-за использования крупных функционально законченных модулей и типовых детерминированных решений по организации их архитектуры.

Процесс проектирования сложных СЛУ базируется на трех основных компонентах: на методе проектирования; на способе принятия решений; на совокупности показателей качества, используемых при оценке окончательных решений.

Методы проектирования СЛУ подразделяются по ряду признаков: по приемам процесса проектирования — автоматизированные и неавтоматизированные; по алгоритму организации процесса создания системы — анализ и синтез; по направлению композиции системы — «сверху вниз» (декомпозиция) и «снизу вверх» (агрегатирование). Наиболее приемлем при проектировании СЛУ декомпозиционный подход: в начальный период разработки из-за сложности проектируемой системы необходимо сформировать представление о классе и типе систем в выделенном классе; технические характеристики элементов системы на нижних уровнях декомпозиции не определены; большинство разработчиков ориентируется на композицию «от общего к частному». При данном способе проектирования СЛУ осуществляется последовательный переход от общего представления о классе создаваемой системы к формированию представления о типе данной системы в выделенном классе систем и ее функциях в общем виде; к укрупненному представлению о структурном решении и принципах построения ПО СЛУ; к выделению типов структурных блоков и связей между ними с последующей детальной проработкой их структурной организации; к представлению основных узлов и устройств структурных блоков (счетчиков, таймеров, регистров, ЗУ, интерфейсных БИС и др.) в виде функционально-логических схем с конкретизацией связей между ними; к окончательной детализации узлов и устройств системы на уровне микросхем, МП, МЭВМ, СЧПУ, ПК и др.

Сложные СЛУ проектируют на основе следующих принципов и признаков их структурной организации: иерархичности, модульности, многофункциональности, перестраиваемости, универсальности связей и интегрированности. При этом в качестве главного принципа построения перспективных систем выступает интегрированность, в качестве основных принципов — иерархичность и универсальность связей, в качестве рабочих принципов — модульность, многофункциональность и перестраиваемость, а в качестве признаков организации структур магистральность, мультиплексность, процессорность, функциональная законченность и программируемость.

Иерархическое построение систем с последовательным вертикальным расположением и соподчинением отдельных подсистем, когда проявляется приоритет действий подсистем более высокого уровня и зависимость их действий от фактического исполнения функций подсистемами нижнего уровня, повышает надежность и производительность системы. Наличие универсальных связей при стандартном сопряжении подсистем и блоков позволяет реализовать многофункциональные иерархические структуры из конструктивно и функционально законченных модулей, сокращать аппаратурные затраты и увеличивать надежность. Использование модульности ведет к функционально-блочному исполнению систем, высокой их работоспособности, сокращению времени наладки и восстановления, увеличению надежности. Многофункциональность и перестраиваемость позволяют осуществлять решение разнородных задач управления, сокращение объема аппаратуры, управлять

ресурсами с различных уровней иерархии системы с целью повышения ее надежности, живучести или пропускной способности. В целом построение СЛУ на базе указанных принципов и признаков организации структур позволяет получать надежные эффективные решения с высокими тактико-техническими и эксплуатационными характеристиками.

Автоматизированное проектирование СЛУ на базе современных САПР («Рапира», ПРАМ, ЕСАП ЭВМ, УАССМА, «Академсинтез» и др.), призванных объединить процессы «проектирование — производство», показывает, что пока в них отсутствует сквозной процесс проектирования, проектные решения принимаются неавтоматизированным способом при недоиспользовании возможностей наличных программно-аппаратных средств, структурному уровню и специфике работы СЛУ и условиям эксплуатации уделяется недостаточное внимание.

В большинстве САПР используется метод сравнительного анализа с последовательным выполнением: выбора алгоритма функционирования проектируемой СЛУ; формирования множества структур и схем, реализующих заданный алгоритм функционирования; сравнения вариантов решений по каждой из заданных характеристик с исключением не отвечающих требованиям ТЗ вариантов и выбора наилучшего решения по результатам оценки. Выбор допустимого множества вариантов реализации проектируемого СЛУ здесь производится неавтоматизированным путем и не по совокупности показателей качества, в комплексе учитывающих множество технических характеристик проектируемой системы и аналогичные параметры класса таких систем, а по отдельным конкретным характеристикам. Такая стратегия выбора по определенному параметру эвристически заданного подмножества вариантов, рассматриваемого в дальнейшем, не отрицает возможности потери наилучшего для всех последующих этапов варианта реализации, создания новых технических решений.

§ 8.2. РАЗНОВИДНОСТИ МОДЕЛЕЙ СЛУ, ПРИМЕНЯЕМЫХ ПРИ СИНТЕЗЕ СЛОЖНЫХ ДИСКРЕТНЫХ СИСТЕМ

В практике проектирования и моделирования применяется множество моделей и языков, описывающих поведение дискретных динамических систем; классические конечные автоматы [22, 53], параллельные граф-схемы (ПГСА) и логические схемы (ПЛСА) алгоритмов [45, 46], операторные схемы параллельных алгоритмов с памятью (ОСПАП) [53], билогические [54] и структурные графы, системы взаимосвязанных графов (СВГ), сети Петри и их модификации [55—57], мографы [58, 59], расплывчатые и детерминированные гиперграфы [60, 61] семантические [62] и нагруженные сети [57] и др.

Применение модели конечного автомата (КА), являющейся ранее научной основой логического управления, крайне затруднительно при описании и оптимизации функционирования СЛУ и ЦУА, требующих комбинаторных вычислений, превышающих возможности ЭВМ.

Позднее были предложены более общие по сравнению с КА модели — сети Петри (К. Петри, Дж. Денис, А. Хольт, М. Хэк, С. Патил, О. Л. Бандман, В. Е. Котов, С. А. Юдицкий и др.), мографы (В. А. Горбатов), гиперграфы (К. Берж, А. А. Зыков, А. Д. Закревский, А. Н. Мелихов, Л. С. Берштейн), ПГСА и ПЛСА (В. И. Варшавский, О. Л. Бандман, В. Г. Лазарев), ОСПАП (В. В. Девятков, А. Б. Чичковский), СВГ (В. В. Руднев), билогические графы (Г. Эстрин и др.).

Данные модели позволили строго и конструктивно определить дискретный процесс, сформулировать его основные свойства, создать методы их распознавания, выделить классы дискретных процессов и разработать соответствующие этим классам упрощенные методы анализа, эквивалентных преобразований и оптимизации описания процессов, а также решить ряд других задач.

Дадим формальное описание и содержательную трактовку вышеперечисленных моделей описания дискретных систем и процессов.

Сети Петри. Сеть Петри формально определяется как пятерка:

$$N=\langle P, T, I, O, M_0\rangle, \tag{8.1}$$

где $P=\{p_i|i=\overline{1, n}\}$ — конечное множество позиций; $T=\{t_j|j=\overline{1, m}\}$ — конечное множество переходов; $P\cap T=\varnothing$ $I\leqslant P\times T$ — бинарные отношения инцидентности; $M_0: P\to E$ — начальная маркировка сети, которая каждой позиции ставит в однозначное соответствие элемент из множества неотрицательных целых чисел $E=\{0, 1, 2, \ldots\}$.

Множества входных и выходных позиций по отношению к переходу t_j обозначаются соответственно через $I(t_j)$ и $O(t_j)$, а множества входных и выходных переходов по отношению к позиции p_i — соответственно через $I(p_i)$ и $O(p_i)$. Маркировка изображается вектором $M=M(p_1), \ldots, M(p_n)$, где $M(p_i)$ — целое неотрицательное число, сопоставленное p_i.

В аналитическом виде динамика состояний сети Петри воспроизводится рекуррентным алгебраическим уравнением (8.2) и логическим уравнением (8.3)

$$M_k=M_{k-1}+A^*U_k, \quad k=1, 2, \ldots, \tag{8.2}$$

где M_k — состояние, которое следует после состояния M_{k-1} в результате k-го воздействия U_k; M_0 — начальное маркирование; A^* — матрица, полученная транспонированием матрицы инцидентности позиций и переходов, причем ее элементы a_{ji}, равны n,—n или 0,

если переход t_j имеет соответственно n исходящих дуг к позиции p_i, n входящих дуг в позицию p_i или не имеет связи с позицией p_i; U_k — управляющий вектор, компоненты которого $u_{jk}=\{0, 1\}$, причем если $u_{jk}=1$, то в k-й момент времени происходит срабатывание перехода t_j, если же $u_{jk}=0$, то срабатывания t_{jk} не происходит. Элементы a_{ji} матрицы A определяют воздействия, оказываемые процессом на состояние системы, указывая число меток, добавляемое в позиции p_i при срабатывании переходов t_i. Матрицу A можно вычислить на основе операций над матрицами I и O, задающими число дуг, следующих соответственно за переходами и позициями $A=I-O^*$. Элементы матрицы O задают число меток, которые должны быть изъяты из позиции p_i при срабатывании перехода t_j. Элементы строки t_j матрицы I определяют число меток, направляемых в позиции сети при срабатывании перехода t_j.

Для вычисления управляющего вектора U_k необходимо определить те переходы t_j, которые срабатывают при маркировании M_{k-1}. Если t_j срабатывает, то $u_j=1$ и произведение A^*U_k дает число меток, добавляемых в позицию p_i в k-м такте. Из анализа матрицы O можно установить те переходы t_j, которые возбуждаются при заданном маркировании M. Условие срабатывания перехода t_j и выработки $u_j=1$ представляется логическим уравнением, управляющим действиями t_j:

$$\forall i\,[m(p_i) \geqslant I(t_j,\ p_i)] \Rightarrow (U_j=1), \tag{8.3}$$

где $m(p_i)$ — число меток в позиции p_i.

Таким образом, для генерирования диаграммы состояний сети Петри достаточно использовать уравнения (8.2) и (8.3).

Графическим изображением сети Петри является двудольный ориентированный граф, в котором p_i обозначаются кружками, переходы t_j — черточками, а дуги направлены только от кружков к черточкам либо от черточек к кружкам. В начальной маркировке M_0 в кружок, соответствующий позиции p_i, помещается $M_0(p_i)$ фишек, изображаемых жирными точками.

Маркировка определяет состояние сети Петри, при этом динамика изменения состояний моделируется движением точек по позициям сети, осуществляемым в результате срабатывания (выполнения) переходов. Сработать в маркировке M_i может только такой переход t_j, для которого в каждой его входной позиции содержится по меньшей мере одна точка. Результатом срабатывания перехода является изъятие из каждой его входной позиции и добавление в каждую выходную позицию по одной точке. Срабатывание перехода считается мгновенным процессом. Маркировка M_l достижима из маркировки M, если существуют последовательности маркировок M_0, M_1, ..., M_l и переходов $\tau=t_1, t_2, ..., t_k$ таких, что $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \ldots \xrightarrow{t_k} M_k$ (обозначают кратко $M \xrightarrow{\tau} M_l$ или $M \to M_l$)

Основными особенностями сети Петри как модели дискретной

динамической системы являются: отсутствие явного времени с заменой тактирования последовательностей изменения состояний причинно-следственными связями между событиями; недетерминизм сети, проявляющийся в неизвестности реализации процесса в системе, для устранения которого вводятся дополнительные формальные средства (пометки позиций и переходов элементами некоторых множеств); возможность моделирования параллельной работы и синхронизация асинхронных процессов.

Сети Петри обладают рядом типовых свойств, отображающих ситуации и явления в моделируемых системах, к числу которых относятся свойства ограниченности, живости, консервативности, устойчивости, достижимости. В зависимости от ограничений, налагаемых на графовую структуру (топологию) сети, либо от процедур построения сети из более простых сетей выделяется ряд подклассов базовых сетей Петри, подробно рассмотренных в специальной литературе, например [55—57].

Помимо базовых сетей известны более сложные сети Петри (временные сети, простые и временные сети с приоритетами переходов, простые и приоритетные раскрашенные сети и др.) [57], относящиеся к классу сетевых моделей. В целом сети Петри обладают высокой мощностью моделирования и разрешения, что позволяет использовать их для оценки уровня выразительности других классических моделей СЛУ.

Конечный автомат. Математическое описание КА приведено в § 1.2. Граф переходов конечного автомата изоморфен помеченной автоматной сети Петри: вершины графа взаимно однозначно соответствует позициям сети, дуги графа — ее переходам, переходы помечаются состояниями входа, а позиции — состояниями выхода. Для любого конечного автомата существует адекватная ему сеть Петри, а обратное — не всегда. В целом класс сетей Петри строго мощнее класса конечных автоматов.

ПГСА и ПЛСА. Параллельной граф-схемой алгоритма (ПГСА) называется ориентированный граф

$$G = \langle V,\ E \rangle, \tag{8.4}$$

где $V = \{v_i \mid i = 1,\ 7\}$ — конечное множество вершин семи типов $V = \{F,\ W,\ U,\ \Omega,\ \Delta,\ H,\ K\}$: F — операторные; W — перехода к выполнению параллельных ветвей алгоритма; U — слияния параллельных ветвей; Ω — проверки логического условия с двумя выходами (да, нет); Δ — слияния взаимно исключающих ветвей алгоритма; $H(K)$ — вершина начала (конца) выполнения алгоритма; $E = \{e_{ij}\}$ — конечное множество дуг. Существование дуги e_{ij} означает, что действие, соответствующее v_j, может быть выполнено только после действия, соответствующего v_i.

Преобразование ПГСА в сеть Петри производится заменой ее элементов элементами сети: вершины типа F, Ω, H, K ПГСА сопоставляются с позициями сети Петри, а вершины W и U — с пе-

реходами с несколькими входящими или выходящими дугами соответственно. Вершины типа Δ с h входящими дугами в сети Петри сопоставляются с переходами $t_{\Delta 1}$, ..., $t_{\Delta h}$, причем если из h идет дуга в вершину типа F, Ω, K, то из каждого перехода t_{ij} $(1 \leqslant j \leqslant h)$ проводится дуга в соответствующую этой вершине позиции сети Петри. Если же на ПГСА из вершины Δ идет дуга в вершину типа W или U, то на сети Петри из каждого перехода $t_{\Delta j}$ проводится дуга в одну и ту же дополнительную позицию, а из нее уже проводится дуга в тот переход, который сопоставлен W или U. Если после замены два перехода на сети Петри окажутся связаны дугой, то между этими переходами вводится дополнительная позиция.

Так как ПЛСА является аналитическим представлением ПГСА, то они также легко интерпретируются сетями Петри.

Операторные схемы алгоритмов с памятью. ОСПАП представляют собой иерархию ориентированных графов, в которых различаются вершины пяти типов (F, W, Ω, H, K), имеющие смысл, аналогичный ПГСА. Вершины называются соответственно функторами, разветвителями, предикатами, стартерами и остановами. Каждый граф в ОСПАП имеет один стартер и один или несколько остановов. Каждая вершина типа F помечается конечно-автоматным оператором, задающим отображение наборов входных и внутренних переменных в наборы внутренних и выходных переменных. Каждая вершина типа Ω помечается логической функцией (предикатом) над входными, внутренними и выходными переменными. Некоторые функторы и предикаты графов i-го уровня $(i \geqslant 2)$ дополнительно помечаются символами стартеров графов $(i-1)$-го уровня, что соответствует операциям запуска нижестоящих графов графами более высокого уровня. В ОСПАП предусмотрено преобразование иерархической системы помеченных графов (собственно ОСПАП).

Билогические графы. Билогическим (дилогическим) графом (БГ) [54] называется ориентированный граф с управляющей логикой, установленной на началах дуг, исходящих из одной вершины, и на концах дуг, входящих в одну вершину. Если из вершины графа исходит несколько дуг, то между их начальными точками устанавливается логическое отношение И ($*$) или ИЛИ ($+$). Аналогичные отношения устанавливаются между концами дуг, входящими в одну вершину. Если в вершину входит или из вершины исходит только одна дуга, то обозначение входной или выходной логики опускается.

При преобразовании БГ в сеть Петри каждой вершине и каждой дуге графа сопоставляется позиция сети Петри, а переходы сети вводятся согласно определенному правилу. Если p и t на сети связаны дугой, то аналогичная дуга показывается и на билогическом графе. При нескольких дугах на входе или на выходе

вершины графа, соответствующей позиции p, устанавливается логика ИЛИ.

Система взаимосвязанных графов. СВГ представляет множество определенных (ингибиторных) сетей Петри $\{N_1, \ldots, N_m\}$, где на каждой сети N_i, имеющей множество позиций p_i и переходов T_i, задана функция τ_i: $T_i \to K_i$ (K_i — множество элементарных конъюнкций от логических переменных, соответствующих позициям $p \in \bigvee_{i=1}^{n} p_i \setminus p_i$ и от их отрицаний). Переменная p определяется как

$$p = \begin{cases} 0, \text{ если } M(p) = 0, \\ 1, \text{ если иначе.} \end{cases}$$

Таким образом, переходы в ингибиторной сети N_i помечаются элементарными конъюнкциями от логических переменных — позиций остальных сетей — и от отрицаний этих переменных. Переход t в сети N_i возбужден, если помимо наличия точек в его входных позициях и отсутствия точек в тех позициях, из которых в t ведут ингибиторные дуги, имеет место $\tau_i(t) = 1$. Совокупность переходов $t \leqslant T = \bigvee_{i=1}^{n} T_i$ реализуема в маркировке M, если каждый переход $t \subset \bar{t}$ возбужден в своей сети N_i и все переходы совокупности $\bar{t}$ принадлежат различным сетям. В зависимости от свойств СВГ различают: *детерминированные* (если в любой маркировке $M \in R(M_0)$ реализуемо не более одной совокупности переходов $\bar{t}$); *обыкновенные* (если в функциях τ_i $(1 \leqslant i \leqslant n)$ ни одна конъюнкция не содержит переменных с отрицанием); *автоматные* (если все N_i сети — автоматные); *конечноавтоматные* (если все N_i сети — автоматные и содержат только одну точку).

В целом, если упорядочить рассмотренные выше модели СЛУ по их выразительным возможностям, получим следующее:

$$|\text{КА}|_M < |\text{ПГСА}|_M = |\text{ОСПАП}|_M < |\text{БГ}|_M < |\text{СВГ}|_M < |\text{СП}|_M,$$

где $|\text{A}|_M$ — мощность моделирования; $A = \{\text{КА, ПГСА, ОСПАП, БГ, СВГ, СП}\}$ — модели конечного автомата, ПГСА, ОСПАП, бинарного графа, системы взаимосвязанных графов и сетей Петри соответственно.

По мощности моделирования (выразительным возможностям) и мощности разрешения из группы рассмотренных эффективнее других являются сети Петри, поэтому они могут использоваться для разработки методов описания и анализа дискретных технологических процессов.

Рост сложности и многообразия видов современных дискретных управляющих систем и объектов управления гибких производств вызвал необходимость создания новых сетевых моделей с целью эффективного проектирования распределенных структур, процессов, ресурсов, алгоритмов и программ.

Системы управления, ориентированные на информационно-алгоритмические преобразования, производственные объекты, ориентированные на материально-энергетические преобразования, могут быть описаны общими системными моделями, базирующимися на единых принципах, функционально идентичных элементах, логико-время-количественных взаимоотношениях [57]. Например, в составе управляющих вычислительных систем (УВС) ГАП можно выделить универсальные и проблемно-ориентированные процессоры (арифметико-логические, интерфейсные, коммуникационные, ввода — вывода, управления базой данных), а также типы функциональных элементов, как память, преобразователи, средства коммутации и обмена, средства управления процессами и ресурсами и др.

Аналогично в объектах управления ГАП выделяются элементы, функционально подобные элементам УВС: технологическое оборудование (обрабатывающие средства); промежуточные накопители и склады изделий, заготовок и инструментов (средства хранения); средства распределения и транспортировки; локальные (встроенные) системы управления. В процессах технологии производства также существуют особенности, идентичные процессам в УВС, например, очереди, блокировки, синхронизация, активные и пассивные (первичные и вторичные) ресурсы, совместное использование ресурса и др. Все это свидетельствует о необходимости использования сетевых моделей.

Эффективное представление алгоритмов управления реальных ГАП также может базироваться на сетевых моделях, позволяющих объединять и оптимизировать в единой форме действия УВС и конкретных объектов, детализировать на различном уровне последовательности операций, действий отдельных компонентов, выделять и анализировать циклограммы работы, узкие места, оценивать производительность, варианты расписаний, взаимодействия прикладных программ нижнего уровня, управлять динамикой перемещения изделий, материалов и инструментов.

Переход от централизованных к распределенным структурам УВС обусловил высокий параллелизм и асинхронность информационно-алгоритмических процессов, необходимость проверки качественной корректности системных действий с анализом таких свойств дискретных процессов, как живость, отсутствие тупиков, ограниченность, безопасность [56, 63]. Распараллеливаемость алгоритмов обусловливается разновидностью архитектур УВС, языков представления, форм их отображения, обеспечивающих быстрое и эффективное использование информации о взаимозависимости операторов, а также способов преобразования алгоритмов из исходной формы в заданную. В целом, при анализе или синтезе архитектуры УВС, структуры вычислительных процессов необходима модель их функционирования, отображающая процессы с различным уровнем

детализации в виде сетевых, автоматных имитационных или аналитических моделей.

К числу сетевых моделей можно отнести: многоклассовые детерминированные и стохастические сети очередей [63], временны́е сети Петри, простые и временны́е сети с приоритетами переходов, раскрашенные сети, приоритетные раскрашенные сети, нагруженные сети и др.

Наиболее распространенными и упорядоченными по степени возрастания их моделирующей мощности являются:

— *простые,* или *базовые, сети Петри;*

— *временны́е сети Петри,* использующие кроме уравнений (8.2), (8.3) упорядоченные списки переходов, определяющие порядок наступления событий;

— *стохастические сети* — временны́е сети Петри, в которых дугам $(p_i t_j)$ сопоставлены вероятности p_{ij} блуждания меток (процессов) в сетевой модели;

— *марковские сети,* использующие для описания структуры систем сети Петри и предусматривающие определение пространства состояний и начального маркирования с последующим расчетом вероятностей состояний и всех необходимых характеристик марковских процессов;

— *нагруженные сети,* являющиеся расширением сетей Петри, эффективно представляющие распределенные структуры, процессы, ресурсы, алгоритмы и программы при проектировании программно-технических средств УВС;

— *логические сети,* представляющие разновидность нагруженных сетей и позволяющие описывать структуры данных, функции и алгоритмы логико-временно́го управления (разметкой дуг сети величинами $a_i \in \{0, 1\}$ и действиями на переходах можно отображать любую из логических операций, снимая тем самым ограничения в представлении логических функций, свойственные базовым сетям Петри);

— *структурированные сети,* использующие модульное иерархическое описание процесса, модели или структуры системы в виде совокупности взаимосвязанных элементов, хранящихся в базе данных и используемых в качестве отдельных системных единиц (извлечение модулей из нее и их связывание по определенным правилам порождает структурированную модель);

— *составные сети,* содержащие модули из сетей различного типа (простых или нагруженных), что создает условия для преемственности разработок, естественности отображения объектов, повышения эффективности управления интерпретацией на различных уровнях описания.

Рассмотрим более подробно особенности нагруженных сетей.

Нагруженная сеть (НС) формально задается пятеркой

$$HC = \{N, D, F, \tau, M_0\}, \qquad (8.5)$$

где N — ориентированный биграф; $D=\{D_s\}$ — множество описателей меток; F — нагружающее отображение, в котором $F_1 : A_1 \to \varphi_1$; $F_2 : A_2 \to \varphi_2$; $F=F_1 \vee F_2$; $\varphi_1=\{r\}$ — множество управляющих функций; $\varphi_2=\{\varphi_{ij}\}$ — множество операционных функций; $T \times d_t \to R_0$ — функция, определяющая время срабатывания переходов; d_t — атрибут метки m_s, управляющий временем; M_0 — начальное маркирование сети; R_0 — множество неотрицательных чисел $\{0, 1, 2, \ldots\}$. При отсутствии операций управления φ_1 или обработки φ_2 на дугах $a \in A$ используется символ λ.

Ориентированный биграф N задается как

$$N=\{T, P, I, O\}, \tag{8.6}$$

где $T=\{t_j\}$ — переходы; $P=\{p_i\}$ — позиции; $I : T \times P \to \{0, 1\}$ — функция следования; $O : P \times T \to \{0, 1\}$ — функция предшествования; $a_{ij} \in A_1$; $a_{ji} \in A_2$ — дуги биграфа; $A=A_1 \vee A_2$, причем $a_{ij}=(p_i t_j)$ и $a_{ji}=(t_j p_i)$ — дуги, следующие соответственно от позиции к переходу и наоборот.

Маркирование сети сопоставляет каждой из позиций множество меток из M, а каждой из меток — описатель со сформированными значениями атрибутов. В каждом из состояний сети маркирование задается набором отображений $M : (p \to M,\ M \to d)$. Начальное маркирование сети M_0 задает начальное распределение меток по позициям и начальные значения атрибутов $d_k \in D_s$.

Для адекватного представления свойств систем на множествах значений атрибутов D_k определяются бинарные отношения r_k (например, отношения порядка, задаваемые решеткой, матрицей, орграфом). Управление процессами и ресурсами на сети выполняется путем сопоставления дугам биграфа N управляющих параметров. Каждой дуге $a_{ij}=p_i t_j$ сопоставляется значение $C_{ij}^k \in X_k$, определяющее условие возбуждения перехода t_j по входящей в него дуге a_{ij}, где X_k — множество допустимых значений k-го атрибута метки. Если в p_i разрешается метка m_s, k-й атрибут которой $d_k(m_s)$ в составе пары $(d_k(m_s), C_{ij}^k)$ удовлетворяет отношению r_k, то m_s может участвовать в возбуждении перехода t_j на дуге a_{ij}, т. е. если $(d_k(m_s), C_{ij}^k) \in r_k$, то метка m_s, находясь в p_i, является кандидатом на возбуждение t_j. Если не исключено появление нескольких переходов — кандидатов на возбуждение, в которых участвует одна и та же метка, то задаются вторичное отношение или ограничения на правила построения сети, обеспечивающие определенность выбора.

Время между моментами возбуждения переходов t_j и их срабатывания задается отображением $\tau : T \times d_t \to R_0$, где $d_t \in D_s$ — атрибуты метки, управляющие временными параметрами сети.

В интервале между возбуждением и срабатыванием перехода t_j метки закрепляются за ним; при срабатывании переходов осу-

ществляется пересылка меток в позиции, следующие за t_j. Дугам $a_{ji}=(t_jp_i)$ присваиваются функции φ_{ji}, определяющие операции над параметрами описателей D_s тех меток, что участвуют в возбуждении t_j. Если срабатывание перехода t_j связано с выполнением действий над некоторой структурой данных, то операционная функция φ_{ji} задает вид обработки атрибутов меток-данных, участвующих в возбуждении.

Класс НС-моделей обладает высокой изобразительной способностью и может порождать более простые типы сетей. Например, при $D=F=\tau=\varnothing$ — это сети Петри, а при $D=F=\varnothing$ — временные сети Петри. Подклассами НС являются многоклассовые, многопотоковые, раскрашенные и другие разновидности сетей с различными метками, в которых соответствующим образом определены D, F и τ.

Построение НС-модели для распределенных программно-аппаратных систем производится следующим путем: на основе анализа зависимостей по данным и управлению, отраженным в алгоритме распределенной обработки, находится структура сетевой модели алгоритма; осуществляется оценка временных свойств операторов; определяется порядок выделения ресурсов; строится бинграф, описывающий взаимодействие процессов и ресурсов; формируются функции, управляющие динамикой сети, и определяется ее начальное маркирование.

В целом НС являются новым классом моделей для проектирования программно-технических средств распределенных микропроцессорных систем управления.

Интерпретация алгоритма с помощью НС-сети естественна, метки и их нагружающие атрибуты могут представлять команды со всеми их полями, данные произвольной структуры с их описаниями, управляющие переменные; операции над данными выступают как управляющие функции НС-сети; условия ветвления и циклы алгоритмов могут представляться с помощью раскрашенных меток и управляющих функций, приписываемых дугам, а параллельные ветви и точки синхронизации — с помощью переходов сети. НС-модель алгоритма позволяет найти требуемые объемы статической и динамической распределенной памяти, выбрать наилучший вариант его структуры или программной реализации.

При структурной реализации алгоритма переходам бинграфа сопоставляют процессорные устройства, позициям — элементы памяти (регистры, триггеры, ЗУ). Наличие метки в позиции-триггере используется для представления потоков управления; метки данных в позиции-регистре — для представления элементов потока данных. Операциями над сетями совмещаются функции в каком-либо устройстве, используется один общий ресурс и др. Реализация в НС единой модели «алгоритм — средства управления взаимодействием — ресурсы» может быть осуществлена по-разному: в виде автономных фрагментов, выполняемых независимо на различ-

ных процессорах, функционально связанных фрагментов, выполняемых на одном процессоре, или макрофрагментов различного типа, выполняемых на определенном количестве проблемно-ориентированных и специализированных процессоров. Выбор конкретного способа реализации определяется требуемыми критериями оптимизации, составом УВС, а также техническими ограничениями и условиями эксплуатации.

§ 8.3. ОСОБЕННОСТИ ЗАДАЧ, СРЕДСТВ И МЕТОДОВ ПРОЕКТИРОВАНИЯ СИСТЕМ УПРАВЛЕНИЯ ГАП

Ускорение научно-технического процесса в машиностроении обусловливается созданием новых прогрессивных технологий и гибких автоматизированных производств (ГАП), использующих вычислительную и микропроцессорную технику, робототехнику, высокопроизводительное технологическое оборудование (станки с числовым программным управлением, обрабатывающие центры, автоматизированные складские, транспортно-накопительные, контрольно-измерительные системы и др.). Анализ специфики производственных и технологических процессов и операций показывает возможность их полной или частичной автоматизации (табл. 8.1) с созданием разнообразных гибких производственных систем (ГПС) и модулей (ГПМ). Стратегическая линия автоматизации ГПС состоит в разработке многоуровневых интегрированных систем управления, обеспечивающих эффективное взаимодействие автоматизированных систем различного назначения (САПР, АСТПП, АСУП, АСНИКИ, АСУ ГПС) со складскими, станочными, транспортными ГПМ, роторно-конвейерными линиями и другим технологическим оборудованием.

Как известно, любая гибкая производственная система (ГПС) механообработки включает в себя три основные подсистемы —

Таблица 8.1. **Особенности применения ряда технологических процессов и операций в ГПС**

Технологические процессы или операции	Примерный объем партии, шт.	Ориентировочное время цикла	Степень применимости в ГПС
Механическая обработка режущим инструментом	1—1000	1—20 ч	Полная
	>1	20—100 ч	Полная с реализацией систем загрузки и транспортировки (ЗРТ)
	>1000	1—10 мин	Достаточная при хорошей системе ЗРТ
		0,3—1 мин	Недостаточная (целесообразна жесткая автоматизация с некоторой гибкостью)

Продолжение табл. 8.1

Технологические процессы или операции	Примерный объем партии, шт.	Ориентировочное время цикла	Степень применимости в ГПС
Шлифование:			
круглое	1—1000	—	Полная
плоское	1—1000	—	Достаточная
Прессование	>2000	Любое	Достаточная с необходимостью учета экономической эффективности
	⩽2000	—	Недостаточная
Вырубка и фальцовка	<1000	Любое	Полная
Сварка	Любой	2 мин	Требуется дорогая инструментальная оснастка
	⩽1000	—	Полная при хорошей системе ЗРТ
Литье металлов:			
под давлением	⩽1000	—	Достаточная
в песчаные формы	⩽1000	—	Незначительная
Ковка и объемная штамповка	⩽1000		Недостаточная
Холодная штамповка	—	—	Достаточная
Сборка	1—1000	>3 мин	Полная
	>1000	20—180 с	Полная при наличии спецоборудования
		3—20 с	Недостаточная
		⩽3 с	Неприменима
Контрольно-измерительные операции	—	—	Полная (возможны короткие циклы с помощью простых операций)

станочную, транспортно-погрузочную и управляюще-вычислительную, последняя обеспечивает эффективную реализацию функций координации и взаимодействия подсистем и модулей, оптимальное управление протекающими процессами, высокую гибкость и надежность их функционирования при минимуме затрат времени, трудовых и материальных ресурсов. Состав реализуемых функций управ-

ления, характеристики систем управления ГПМ, содержание объемно-временны́х параметров информационных потоков между подсистемами ГАП, требований каналов связи со смежными АСУ ГПС обусловливает выбор структуры управляющих вычислительных комплексов ГАП, реализуемых на базе современных микропроцессорных средств, мини- и микроЭВМ, систем числового программного управления (СЧПУ) и программируемых контроллеров (ПК), соединенных каналами связи друг с другом и с ЭВМ более высокого уровня.

В производстве эксплуатируются сотни типов СЧПУ и ПК, построенных на различных принципах и элементной базе, что затрудняет их обслуживание, диагностику неисправностей и сбоев, сопряжение в единые управляющие сети. Реализация СЧПУ и ПК производится по пяти классам (NC, CNC, NC — CNC, DNC, MPST), отличающимся друг от друга спецификой построения структуры, элементной базой, функциональными возможностями, разрядностью центрального процессора, степенью гибкости структуры, ее избыточности и универсальности, способностью перепрограммируемости алгоритмов управления и др. Отношение к развитию СЧПУ у специалистов неадекватное: одни видят перспективы развития СЧПУ на пути совершенствования универсальных средств CNC- и MPST-типа, другие — на пути создания СЧПУ класса NC на базе специализированных БИС с повышенной надежностью, малой стоимостью, с использованием их в специальном или универсальном (работающем в режиме специального) оборудовании.

В целом развитие СЧПУ и ПК связывается с ростом их вычислительной мощности и производительности, с расширением функциональных возможностей, с введением развитых периферийных устройств, с применением помехоустойчивых БИС и СБИС специального и универсального назначения, с развитием многопроцессорных структур СЧПУ и ориентацией процессора на решение крупных фрагментов задач; с повышением уровня автоматизации с функциями автоматического измерения, адаптации, коррекции и др., с переходом на общепринятые языки программирования высокого уровня, стандартные интерфейсы и протоколы системного обмена данными типа MAP — TOP и др.

Номенклатуру выпускаемых промышленностью СЧПУ можно подразделить на три группы:

1) сложные СЧПУ для управления станочными ГПМ, характеризующиеся многопроцессорной структурой, повышенной вычислительной мощностью, большим объемом памяти системного программного обеспечения (до 1,0 Мбайт), расширенным составом функций управления, включая автоматические измерения обрабатываемой детали, коррекцию и контроль выработки ресурса и степени износа инструмента, самодиагностику системы «СЧПУ — станок», автоматической подготовкой управляющих программ (УП),

включением в состав структуры системы ПК с числом входов — выходов до 1024 и др.;

2) СЧПУ пониженной стоимости и составом функций и возможностей, ориентирующихся на конкретные модели станков;

3) СЧПУ для специальных видов оборудования. К числу развитых ПК можно отнести гаммы ПК, разработанных фирмами «Мицубиси» и «Фанук» (Япония), «Аллен — Бредли» и «Дженерал Электрик» (США), «Сименс» (ФРГ) и др.

В целом СЧПУ и ПК являются сегодня подотраслью общей компьютеризации машино- и станкостроительных производств с широким использованием микропроцессорной и дисплейной техники, периферийных устройств, памяти, базовых системных соглашений по организации и комплексированию средств в системах, общих решений по языкам программирования, операционным системам, программному обеспечению.

Развитие новых СЧПУ и технологического оборудования обусловили наряду с типичными продукционными, сервисными и специальными функциями появление новых функций управления. Из числа продукционных — это введение новых видов многокоординатной интерполяции, увеличение числа независимых движений до 5—12 с числом координат в каждом из них до 2—3, одновременное позиционирование нескольких десятков и даже сотен координат, расширение диапазонов задаваемых значений до 10^6—10^8 дискрет при цене дискреты до 0,00001 мм. Развитие сервисных функций связывается с сокращением объемов управляющих программ (УП) с широким применением многоуровневых циклов, циклов с произвольными параметрами, введением программного формирования инструкций оператору, разновидностей коррекций, влияющих на динамические характеристики процессов обработки всех видов контроля, повышающих надежность функционирования и упрощающих эксплуатацию средств управления, и др.

Преобразование информации в УВК, СЧПУ и ПК (от чертежа детали до отработки конкретных приводов станка) связывается с последовательностью пяти этапов, начиная от составления УП, их редактирования и запоминания, последующего преобразования в вид, удобный для манипуляции с геометрической информацией, вычисления вспомогательных величин, требуемых для реализации сложных функций, интерполяции определенных участков траектории с обеспечением постоянства скорости резания, контурной скорости, стабилизации ускорений, определения корректирующих поправок и других величин; представления информации о координатах перемещений в виде унитарных кодов или многоразрядных слов с преобразованием их в управляющие команды на перемещение рабочих органов и исполнительных двигателей и заканчивая отработкой этих команд в соответствии с заданными требованиями по точности, повторяемости и т. п.

Возможность возникновения множества ошибок на каждом из перечисленных этапов требует быстрого их исправления и корректирования, что обусловливает применение развитых СЧПУ, обеспечивающих реализацию задач управления и преобразования информации следующего вида:

ввода — вывода и редактирования входных данных, организации библиотеки УП;

формообразования, задающего в реальном масштабе времени движения рабочих органов объектов управления по заданной траектории с заданной скоростью;

программируемого интерфейса, преобразующего команды и сигналы, выданные с СЧПУ, пульта оператора, датчиков и др., в управляющие исполнительными устройствами и приводами команды заданной амплитуды, формы и длительности; диагностирования объекта и СЧПУ.

Перечисленные классы задач реализуются в четырех основных подсистемах СЧПУ: ввода — вывода и редактирования данных, формообразования, программируемого интерфейса, диагностирования.

К задачам подсистем ввода — вывода и редактирования данных перспективных СЧПУ относятся: увеличение максимального объема УП до 8—16 раз, создание библиотеки УП и подпрограмм развитого пульта оператора с многофункциональной клавиатурой, позволяющего осуществлять ручной ввод и редактирование УП в режиме диалога; использование графического дисплея с выдачей для оператора типовых решений по условиям обработки деталей; использование широкого набора периферийных устройств (КНМЛ, НГМД, телетайп и др.), энергонезависимой памяти на ЦМД- или ПЗС-структурах, существенно повышающих надежность функционирования СЧПУ.

К задачам подсистемы формообразования перспективных СЧПУ относятся: увеличение управляемых координат до 12—15 при возможности их переназначения; роста значений максимума перемещений до 100; круговая интерполяция в произвольно наклонной плоскости, линейная и круговая интерполяция в полярной и цилиндрической системах координат; повышение скорости рабочих подач до 15 мм/мин, а скорости быстрого хода — до 20 м/мин; увеличение числа и видов коррекции УП с возможностью поворота координат, автоматическим изменением содержимого корректоров; использование всех видов ручного управления исполнительными органами объекта управления; восстановление геометрической и технологической информации при выходе в исходную точку в ручном или автоматическом режимах; учет коррекции ходового винта, прогиба, неидеальности кинематики звеньев станков с компенсацией механической деформации деталей; управление приводами подач по более сложным алгоритмам с замыканием обратной связи

через процессор; адаптивное управление по моменту резания и величине вибрации и др.

Задачи программируемого интерфейса в СЧПУ связаны с выполнением технологических М-, S-, Т-функций, обслуживанием пульта станка, анализом аварийно-блокировочных ситуаций и подразделяются на реализованные и перспективные задачи. К числу реализованных задач относятся задачи управления шпинделем и другими компонентами ГПМ, инструментальным магазином согласно функциям М, S, Т, а также ряд системных функций по включению, сбросу функций М, S, Т, анализу аварийно-блокировочных ситуаций, зажиму — отжиму заготовок, салазок, бабки, поворотного стола и др. К перспективным задачам СЧПУ можно отнести такие, как расширение функций управления шпинделем (контроль и индикация скорости шпинделя; коррекция величины кода S-функции при отработке алгоритмов адаптации; переключение двигателя главного движения в следящий режим); расширение функций управления инструментом (задание номера инструмента при первоначальной их раскладке по гнездам с дальнейшим поиском гнезда с требуемым инструментом по таблице соответствия, оптимизацией раскладки и автоматической коррекцией при каждой смене инструмента; введение дублирующих инструментов с учетом их стойкости и модификацией алгоритма поиска при переходе на дублер; вычисление путевых перемещений при смене инструмента; учет номера инструмента при реализации алгоритмов контроля стойкости и др.); функции измерения обработанной детали на ГПМ и за его пределами с выполнением измерения параметров отверстий по точкам, обработкой допусков на фактический размер и соотношений между геометрическими элементами; функции измерения, контроля и прогнозирования износа инструмента, его поломки; функции адаптивного управления, связанные со стабилизацией мощности резания, подачи и температуры нагрева шпинделя и др.

К задачам подсистемы диагностирования перспективных СЧПУ относятся функции функционального и тестового диагностирования, обеспечивающие: периодическую проверку работоспособности источников питания, состояния объекта управления с достаточной глубиной диагностирования (микроЭВМ, модель СЧПУ, периферийный блок, отдельный элемент); проверку правильности числовых значений УП и ее сохранности в процессе эксплуатации, обрыва линий питания датчиков перемещения; диагностирования модулей программного обеспечения по входным и выходным данным; границ рабочих зон детали и приемного стола, состояния станка в процессе обработки детали; расширения систем измерения детали с замером фактических размеров и сравнением с требуемыми размерами; учет времени работы компонентов СЧПУ (выработки ресурса режущего инструмента, простоев, выполнения планового задания; наличие модуля обработки сбойных ситуаций и библиотеки

слов аварийных предупреждений и сбоев с выдачей их на экран дисплея).

Анализ специфики основных требований пользователя и алгоритмов функционирования станков с ЧПУ и ГПМ показал, что, во-первых, выделять классы задач СЧПУ по признаку группы станков нецелесообразно и, во-вторых, все многообразие задач СЧПУ можно подразделить на следующие категории задач:

а) задачи, алгоритмы решения которых не зависят от оборудования и форм производимых деталей, имеют типовую схему реализации в соответствии с общими функциональными законами (задачи интерполяции, управления приводом, компенсации погрешностей кинематических цепей, управления скоростью подач и др.); в зависимости от вида оборудования здесь изменяются лишь их количественные характеристики;

б) задачи, алгоритмы решения которых незначительно зависят от вида оборудования, сохраняя общие черты для всех групп станков различной сложности, однако их количественные характеристики меняются даже в рамках одного класса станков (задачи управления электроавтоматикой станков, подготовки кадров);

в) задачи, алгоритмы решения которых и количественные характеристики полностью зависят от вида оборудования, характера технологического процесса и формы детали (задачи адаптации по производительности и точности, согласование основных движений станка и др.).

На повышение надежности функционирования систем управления ГПМ влияют задачи, реализуемые подсистемой программируемого интерфейса с организацией одного из возможных типов связей между станком и СЧПУ, пультом оператора, датчиками контроля состояния, перемещений и т. п., исполнительными приводами и рабочими агрегатами и органами ГПМ. В перспективных СЧПУ этот круг задач характеризуется существенным ростом удельного веса и сложности вычислительных операций над числами и таблично организованными массивами чисел, номенклатуры разнообразных датчиков, вариантностью обработки алгоритмов в зависимости от вида подготовительных функций или параметров, задаваемых в УП, с частым обращением к УВВ (дисплею, печати), возможностью изменения порядка обработки кадров УП и коррекцией ее числовых значений по результатам обработки алгоритмов при смене коррекций на инструмент, результатов измерений деталей в процессе обработки и др. Этот класс задач можно классифицировать (табл. 8.2) по виду обрабатываемых задач, характеру процедур обращения, способу взаимодействия при реализации технологических функций. При этом по виду обрабатываемых задач до 80 % составляют задачи логической обработки дискретных сигналов (т. е. функции СЛУ); по характеру процедур обращения до 60 % составляют однократно выполняемые технологические M-, S-, T-функции, при реализации которых на 84 % используется способ

выдачи декодированного сигнала по определенному периферийному разряду с анализом ответа обработки.

Описание алгоритмов и программную реализацию задач логического управления, присущих подсистеме программируемого интерфейса систем управления ГПМ на базе перспективных СЧПУ и ПК, можно производить с помощью большого числа языков. К числу наиболее известных относятся такие языки, как PLC

Таблица 8.2. **Характер задач, реализуемых в подсистеме программируемого интерфейса перспективных СЧПУ**

Классификационные признаки и типы реализуемых задач	Мощность, задач, %
1. По виду обрабатываемой информации:	
логическая обработка дискретных сигналов с контролем исполнения по времени	80
простые расчетные операции с числами	10
сложные вычисления с обработкой информации от датчиков	10
2. По характеру процедур обращения СЧПУ к задачам ППИ:	
однократно выполняемые системные функции (при запуске системы)	3
однократно выполняемые технологические функции (в пределах кадра)	60
циклически выполняемые системные функции (аварийно-блокировочные ситуации, опрос пульта)	37
3. По способу реализации технологических функций:	
а) задание технологической функции по определенному периферийному разряду слова памяти:	
без выдачи числовой информации и сигнала сбоя	84
с выдачей числовой информации, декодированием сигнала и сигнала о сбое	8
б) задание технологической функции по усложненным алгоритмам	8

(США); PC100, PC200, PC400, Step-5 (ФРГ); Siprom (Италия); Программа-700 (Болгария); Фанук (Япония); PLC700 (Швеция); Festo (Австрия); ЭЛА, ЯФП, ЯФПТ. Элавт, Ярус-2 (СССР) и др. Наибольшее практическое применение получили: макроязыки высокого уровня типа ЯФП, ЯФПТ, содержащие в своем составе макрокоманды, повышающие производительность разработки прикладных программ привязки СЧПУ к конкретному технологическому оборудованию; проблемно-ориентированные языки типа ПЛ/М, Паскаль, Ада, применяемые в ряде ПК; предметно-ориентированные языки типа Festo, Ярус-2, ЭЛА; машинно-ориентированные языки типа Фанук, Siprom, Step-5. Команды языков могут быть представлены в самой различной форме, начиная со словесного описания (Festo, Ярус-2), уравнений булевой алгебры (PC,

Siprom, Программа-700), мнемокодов (PC, Step-5, Фанук, PLC700), макрокоманд (ЯФП, ЯФПТ, ЭЛА) и кончая релейно-контактными схемами (PLC, Программа-700, Фанук). В целом чем выше уровень языка, тем эффективнее создавать простые по структуре, ясные и мобильные программы, устранять за один шаг несколько уровней ошибок, выражать любую функцию небольшим числом операторов, сокращать стоимость разработки прикладных программ.

Анализ эффективности использования языков описания и программирования задач и функций программируемого интерфейса систем управления ГПМ механообработки можно вести на базе ряда качественных K_i и количественных C_j критериев, например, соответствия команд языка требуемым элементарным операторам (K_1), а подпрограмм обработки дискретных сигналов (ПОДС) — заданным функциям (K_2); простоты конструкций языка (K_3); ясности структуры ПОДС (K_4); отсутствия разночтений команд (K_5); возможности расширения языка и ПОДС (K_6); легкости обучения и программирования (K_7); легкости программной реализации (K_8); длины и объема программ (C_1), быстродействия их выполнения (C_2); сложности структуры программ (C_3), требуемого объема памяти (C_4) для хранения программы и др. Окончательно выбрать наиболее эффективный вариант описания и реализации алгоритмов управления ГПМ механообработки можно на базе методики, изложенной в § 5.1, с использованием аддитивной скалярной оценки, получаемой в процессе вычисления ее значений по частным критериям качества с учетом весовых оценок важности или отношений предпочтительности критериев друг перед другом.

Выбор языков описания и способов реализации алгоритмов управления ГПМ на практике ограничивается использованием конкретных типов СЧПУ или ПК, наличием в них одного языка (максимум двух), спецификой организации системного, базового ПО и др., что проиллюстрировано в § 8.6 на одном из примеров.

В целом создание эффективного прикладного ПО для управления ГПС предусматривает следующее:

— оптимизацию программ по критериям быстродействия исполнения и минимума объема памяти, декомпозицию ПО на постоянную и варьируемую части в условиях ограниченности ресурсов применяемых МЭВМ, СЧПУ и ПК, высокой вычислительной сложности решаемых задач, многочисленности требований и ограничений по использованию и эксплуатации;

— поиск рационального соотношения централизации и распределения функций управления;

— выбор рациональных способов организации работы с памятью, вариантов доступа к данным (распределение памяти, оптимизация функций адресации и т. п.) и организации обмена данными между программными модулями с учетом специфики работы наличных программно-аппаратных средств.

§ 8.4. ПРОЕКТИРОВАНИЕ НАДЕЖНЫХ ПОМЕХОУСТОЙЧИВЫХ СТРУКТУР МИКРОПРОЦЕССОРНЫХ СИСТЕМ УПРАВЛЕНИЯ ГИБКИМИ ПРОИЗВОДСТВЕННЫМИ КОМПЛЕКСАМИ

Экономическая эффективность МПСУ ГПК определяется не только снижением стоимости, но и уменьшением потерь от ненадежности функционирования систем в процессе эксплуатации, основную долю которых составляют отказы и сбои аппаратуры,

Таблица 8.3. **Значения параметров помех в электрических сетях напряжением 380/220 В**

Параметры	Производственные условия			
	механический цех	штамповочный цех	сборочный цех	вычислительный центр
Время наблюдения, ч . . .	186/1617	34	22	—/108
Количество помех	15432	1256	214	
Число провалов	—/209			—/24
Число отключений	—/125			—/2
Максимальная амплитуда, В	295	65	153	
Средняя амплитуда, В . .	15	20	54	
Средняя глубина провалов, В	—/68	—	—	—/73
Средний период слежения, ч:				
провалов	—/7,7	—	—	—/4,5
отключений	—/12,9	—	—	—/5,4

Примечание. В числителе указаны параметры импульсных помех, в знаменателе — параметры длительных помех.

возникающие в результате действия различного рода помех. Поэтому повышение надежности функционирования систем управления за счет обеспечения помехоустойчивости их собственных программно-аппаратных средств и реализуемых вычислительных процессов является важной задачей при разработке микропроцессорных систем управления гибкими производственными комплексами (МПСУ ГПК).

Исследования реальной помеховой обстановки в подразделениях ГАП (табл. 8.3) показывают, что в любой момент времени в МПСУ может проявляться два вида сбоев аппаратуры, соответственно приводящих и не приводящих к нарушению вычислительного процесса. Первые называются *функциональными* сбоями, поражающими информационные связи в МПСУ и вызывающими искажение данных или нарушение в последовательности вычислений, вторые — *структурными* сбоями.

Процесс функционирования МПСУ с точки зрения изменения ее работоспособности под воздействием помех можно интерпретировать графовой моделью (рис. 8.1), где вершины a_{0i}, a_{1i} обозначают состояния нормальной работы и функционального сбоя вычислительного процесса, а вершина a_{2i} — функциональный отказ системы. Веса на дугах графа имеют следующий смысл: $e_{\phi i}$ — вероятность функционального сбоя системы, обеспечивающая переход из состояния a_{0i} в состояние a_{1i} под воздействием потока помех интенсивностью $\Lambda_п$; $e_{вi}$, e_{0i} — вероятности перехода системы из состояния a_{1i} в a_{0i} или в a_{2i} соответственно. Величины вероятностей e_{0i} и $e_{вi}$ характеризуют способность системы к самовосстановлению $(e_{0i}+e_{вi}=1)$.

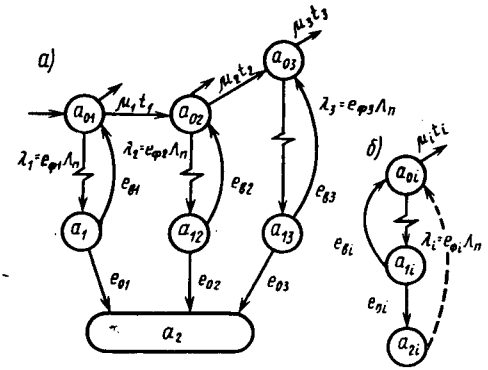


Рис. 8.1. Интерпретация процесса функционирования МПСУ в условиях действия помех:

а — фрагмент функционирования МПСУ; *б* — укрупненный фрагмент; a_{0i} — состояние нормального функционирования; a_{1i} — состояние функционального сбоя системы; a_{2i} — состояние функционального отказа системы; $\lambda_i = e_{\phi i}\lambda_п$ интенсивность потока помех; $\mu_i t_i$ — вероятности переходов при нормальной работе под воздействием программы управления; $e_{вi}$ — вероятность восстановления системы после сбоя; e_{0i} — вероятность функционального отказа системы

Реальное время существования сбоя (время активизации состояния a_{1i}) очень мало, поэтому сбои и функциональные отказы могут быть представлены разреженными потоками помех вида $\Lambda_с = e_ф\Lambda_п$; $\Lambda_{фо} = e_о e_ф \Lambda_п$, причем $e_ф = e_а e_п$, где $e_а$ $(e_п)$ —вероятность того, что структурный сбой не приводит (приводит) к нарушению вычислительного процесса. Отсюда следует, что для построения надежной МПСУ достаточно определить величины $e_ф$ и $e_о$ в виде функций, зависящих от параметров помеховой обстановки $\Lambda_п$ и сбоев $\Lambda_с$.

Так как МПСУ состоит из множества A функциональных элементов аппаратуры и имеет M рабочих состояний $a_{oi} | i = \overline{1, M}$ вычислительного процесса, то переходы состояний $a_{o1} \to a_{o2} \to , \ldots \ldots, \to a_{oM}$ осуществляются под воздействием программы управления, а переходы $a_{oi} \to a_{1i}$ — в результате сбоя подмножества элементов аппаратуры $A_i \in A$ с интенсивностью $\lambda_i = e_{\phi i}\Lambda_п$. Из состояния сбоя a_{1i} процесс либо с вероятностью $e_{вi}$ восстанавливается $(a_{1i} \to a_{oi})$, либо с вероятностью e_{oi} переходит в состояние функционального отказа $a_{1i} \to a_{2i}$. В этом случае имеем

$$E_ф = \sum_{i \in M} B_i e_{\phi i} \quad \text{или} \quad E_ф E_0 = \sum_{i \in M} B_i e_{\phi i} e_{0i},$$

где B_i — вероятность того, что при воздействии одиночной помехи процесс находится в состоянии a_{oi}.

Таким образом, для построения помехоустойчивой МПСУ необходимо определить параметры $e_{\phi i}$, e_{oi}, $B_i | i \in M$, которые могут быть получены: $e_{\phi i}$ — из анализа помехоустойчивости фиксированного множества аппаратуры A_i; B_i и e_{oi} — из анализа свойств вычислительного процесса, протекающего в МПСУ, а также из характера функционального сбоя (вида сбоев в МПСУ — одиночных или групповых). В целом величина $e_{\phi i}$ является функцией от помехоустойчивости всех элементов, входящих в A_i. Однако наиболее вероятной причиной сбоя в A_i являются сбои элементов памяти (ЭП) V_i, фиксирующих все изменения, происходящие в МПСУ (логических элементах, линиях связи и др.). Помехоустойчивость ЭП определяется совокупностью таких параметров помех, как их длительность τ и амплитуда U, подразделяющих помехи на сильные (h-помехи) и слабые (s-помехи).

В целом повышения надежности функционирования МПСУ, работающих в реальных производственных условиях, можно добиться на основе: а) концепции недопуска отказов, связанной с устранением причин ненадежности; б) концепции допуска отказов, связанной с созданием отказоустойчивых, или толерантных, систем, до некоторой степени не ощущающих ненадежность своих функциональных компонент. По отношению к сбоям вычислительных процессов, протекающих в МПСУ, эти подходы могут интерпретироваться соответственно как предотвращение сбоев (Fault Avidance — FA-процессы) и как построение процессов, нечувствительных к сбоям (Fault Tolerance — FT-процессы). В своей основе методы построения МПСУ на FA-принципе сводятся к минимизации вероятности функционального сбоя E_ϕ, а МПСУ на FT-принципе — к минимизации вероятности функционального отказа E_o, причем их реализация может быть выполнена на аппаратном, программно-аппаратном или чисто программном уровнях.

Анализ известных методов повышения надежности и помехоустойчивости МПСУ, характеризуемых избыточной и неизбыточной структурой используемых программно-аппаратных средств, позволил подразделить их на следующие классы:

1. Методы аппаратной реализации неизбыточных структур МПСУ, обеспечивающих достижение минимума вероятности структурного сбоя E_a в системе. К их числу относятся методы синтеза МПСУ с минимальным количеством и объемом высококачественной аппаратуры, применением помехоустойчивых БИС, а также качественных источников питания, обеспечивающих надежную фильтрацию импульсных помех, возникаемых в шинах питания, и защиту от провалов напряжения питания, вызываемых помехами. Сфера применения данных методов довольно обширна, но она ограничивается стоимостью создаваемых при этом систем.

2. Методы аппаратной реализации избыточных структур МПСУ, обеспечивающих устранение последствий сбоев системы с минимизацией (максимизацией) параметра $E_o(E_в)$ за счет введения параллельных основному вычислительному процессу процедур контроля, диагностики и восстановления. К их числу относятся методы структурного резервирования с восстановлением нормальной работоспособности системы путем переключения на резерв либо изменения функции голосования, а также методы информационной избыточности с применением специальных и самокорректирующихся кодов с аппаратной реализацией механизма восстановления. Применение данных методов ограничивается стоимостными, весогабаритными и энергомощностными факторами, а также возрастанием опасности структурного сбоя и увеличением параметра $E_ф$ в условиях избыточности аппаратуры, используемой для реализации процедур контроля и восстановления.

3. Методы программно-аппаратной реализации помехоустойчивых структур МПСУ при введении схемной избыточности для выполнения непрерывных либо периодических процедур оперативного контроля и программной — для восстановления вычислительного процесса, производимого по сигналу прерывания, поступающему от подсистемы аппаратного контроля, с выполнением рестарта программы или ее участков либо с выполнением процедуры устранения сбоя (например, восстановления информации в ОЗУ с помощью корректирующих кодов). Применение данных методов может быть достаточно эффективным при небольшой аппаратной избыточности схем контроля и резерве времени, достаточном для выполнения процедуры восстановления.

4. Методы программной реализации неизбыточных структур МПСУ, обеспечивающих повышенную помехоустойчивость за счет минимизации вероятности структурного сбоя $E_п$, приводящей к нарушению вычислительного процесса. Данные методы базируются в основном на идеологии конструирования качественных (в частности, структурированных) программ.

5. Методы чисто программной реализации помехоустойчивых МПСУ при периодическом выполнении процедур контроля и восстановления, встраиваемых в ход основного вычислительного процесса. Данные методы наиболее применимы при инерционных объектах управления, так как не требуют введения дополнительных аппаратных средств. Процедуры контроля здесь выполняются программным путем на базе методов алгоритмического и логического контроля. Вместе с тем реализация процедур контроля, встроенных в основной процесс, требует дополнительных ресурсов памяти, увеличения объемов запоминаемых данных, длительности хранения, задержки обработки полученной ранее информации. И если низкая оперативность контроля для функций передачи, хранения и преобразования данных еще может быть допустимой, то задержки в обнаружении неправильного хода основной программы могут приве-

сти к значительным искажениям информации и к невозможности восстановления вычислительного процесса.

6. Программные методы без процедур контроля и с автовосстановлением.

Ускорить выполнение процедур восстановления можно путем исключения программной реализации процедур контроля, что используется в МПСУ с автовосстановлением вычислительного процесса при достаточно инерционных объектах управления. В этом случае на исполнительные устройства объектов допускается подача ложных управляющих сигналов длительностью, не превышающей некоторой величины, за которую очередной цикл основного вычислительного процесса уже заканчивается, что обеспечивает ликвидацию последствий сбоя и самовосстановление нормального процесса. К ограниченному числу подобных методов построения вычислительных процессов относятся методы, способствующие уменьшению цикла обращения к исполнительным механизмам инерционных объектов, расширению циклов программ, что позволяет достичь эффекта, сопоставимого с применением утроенного мажоритарного контроля.

Таким образом, высокой помехоустойчивости МПСУ ГПК можно достичь программным путем без введения или при незначительной избыточности аппаратуры контроля, что очень привлекательно при использовании серийных законченных КТС (МЭВМ, СЧПУ, ПК). Вместе с тем здесь необходима рациональная организация вычислений и тщательная проработка структуры ПО, обеспечивающих в итоге надежную реализацию функций управления ГПК. Однако наибольший эффект программные методы дают при наличии надежного оперативного механизма защиты от нарушений последовательности операций, реализованного на основе аппаратного контроля хода программы. В условиях значительной временной избыточности, характерной для МПСУ ГПС, аппаратный контроль хода программы особенно важен, так как все остальные процедуры могут быть проконтролированы программным путем. В связи с важностью и ответственностью реализации в МПСУ контролирующих и восстановительных функций проведем анализ наиболее известных методов контроля и восстановления вычислительных процессов.

Методы контроля и восстановления процессов в МПСУ. К основным классам методов аппаратного контроля за ходом вычислительных процессов в МПСУ можно отнести контроль посредством: а) меток времени; б) контрольных участков; в) меток операторов; г) контрольного алгоритма.

Контроль с помощью меток времени основывается на знании длительностей исполнения отдельных участков программы, разделенных метками начальных и конечных точек контролируемых участков и измерении временных интервалов между последователь-

ностью меток, по значениям которых определяется корректность реализуемых процессов.

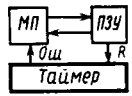
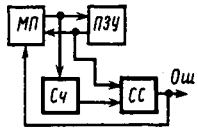
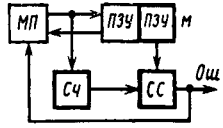
Аналогичным образом реализуются *процедуры с использованием контрольных участков,* однако при этом между отдельными метками подсчитываются не временные интервалы, а некоторые контрольные соотношения с последующим сравнением их с эталонными значениями. По совпадению измеренных и эталонных значений контрольных соотношений (обычно это количество исполненных команд либо различные свертки кодов реализованных операторов) судят о корректности протекающего процесса.

При сокращении участков программ до минимальной длины (отдельной команды или оператора) описанный метод контрольных участков преобразуется в *метод меток операторов.* Коды меток операторов заносятся в дополнительные разряды основного ПЗУ или в установленное параллельно основному дополнительное ПЗУ. В процессе реализации вычислительного процесса система контроля вычисляет код ожидаемой метки и сравнивает с кодом очередной эталонной метки, считываемой из ПЗУ. При их совпадении процесс продолжается, при несовпадении — выдается сигнал о сбое процесса. Механизм вычисления кода ожидаемой метки основывается на подсчете числа операторов, свертки их кодов либо комбинированном методе.

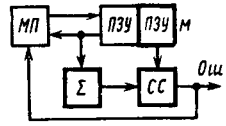
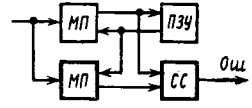
При использовании наиболее развитого метода контроля вычислительного процесса, отслеживаемого на основе *контрольных алгоритмов,* правильность хода процесса проверяется путем реализации алгоритма и анализа состояния шин МПСУ на дополнительном (сторожевом) МП или на специализированной аппаратуре с выработкой сигнала ошибки при сбое МПСУ, а также путем дублирования МП с постоянным сравнением результатов вычислений при реализации контрольного алгоритма дубль-процессом. В системах хранения и передачи данных в качестве контрольных алгоритмов часто используются алгоритмы помехоустойчивого кодирования.

Анализируя эффективность перечисленных методов контроля с точки зрения применимости их для МПСУ инерционными объектами ГПК, отметим, что основная доля сбоев вычислительного процесса в МПСУ приходится на режим отработки исполнительными устройствами и механизмами соответствующих управляющих команд. Влияние сбоев при этом проявляется в некорректном вычислении текущих значений вектора управляющих воздействий, искажении информации в ОЗУ и входных данных, в ложном переходе в другой режим и нарушении последовательности работы программного счетчика, причем если в первом случае нарушение может быть исправлено благодаря самовосстановлению значений вектора управляющих воздействий посредством цикличности процедур вычислений, во втором случае сбой может быть ликвидирован встроенными программными средствами, то сбой программного счетчика приводит к изменению выполняемой операции с заменой

Таблица 8.4. **Сравнительные характеристики аппаратно реализуемых методов контроля для повышения помехоустойчивости МПСУ ГПК**

Метод контроля	Вероятность отказа системы E_0 при воздействии s- и h-сбоев	Время обнаружения	Вероятность обнаружения	Сфера контроля	Типовая структура МПСУ
Метки времени	$E_0 = \begin{cases} 1-(0{,}5\ldots1)(1-\tau_к/(2\tau_ц)) & \text{при } \tau_к < \tau_ц \\ 1-(0{,}5\ldots1)\tau_ц/(2\tau_к) & \text{при } \tau_к \geqslant \tau_ц \end{cases}$	Б	Н	У	
Контрольные участки	$E_0^s = \begin{cases} \tau_к/(2\tau_ц) & \text{при } \tau_к < \tau_ц, \\ 1-\tau_ц/(2\tau_к) & \text{при } \tau_к \geqslant \tau_ц, \end{cases}$ $E_0^h = \begin{cases} (1-g^{-1})\tau_к/(2\tau_ц) & \text{при } \tau_к < \tau_ц, \\ (1-g^{-1})(1-\tau_ц/(2\tau_к)) & \text{при } \tau_к \geqslant \tau_ц, \end{cases}$	Б	В	У	
Метки операторов: подсчет операторов	$E_0^h = \begin{cases} 0{,}5 & \text{при } m_к=1,\ g=2,\ k=1 \\ 0{,}33 & \text{при } m_к=2,\ g=3,\ k=2 \\ 0{,}14 & \text{при } m_к=3,\ g=7,\ k=3 \\ 0{,}04 & \text{при } m_к=8,\ k=1 \end{cases}$ $E_0^s = \begin{cases} 1-1/l & \text{при } m_к=1\ g=2\ k=1 \\ (l-1)/\varkappa_0 & \text{при } m_к=2\ g=3\ k=2 \\ (l-1)^2/\varkappa_0^2 & \text{при } m_к=3\ g=7\ k=3 \\ 2^{-m_0} & \text{при } m_к=8\ k=1 \end{cases}$	М	B^s H^h	У	

Продолжение табл. 8.4

Метод контроля	Вероятность отказа системы E_0 при воздействии s- и h-сбоев	Время обнаружения	Вероятность обнаружения	Сфера контроля	Типовая структура МПСУ
контрольное суммирование	$E_0^s = E_0^h = 2^{-m_0}$	М	В	С	
Контрольный алгоритм	$E_0^s = 0{,}5\varkappa_0(1-\varkappa_0)^{2p-2}$ $E_0^h = 2^{-p}$.	М	В	С	

Примечания: 1. На схемах утолщенными линиями обозначены избыточные элементы структуры МПСУ, к которым относятся: Сч — счетчик; СС — схема сравнения, ПЗУм — ПЗУ меток; Σ — сумматор; МП — сторожевой дубль-микропроцессор.

2. Б — большое; М — малое; В — высокая; Н — низкая; У — узкая; С — средняя, B^s — высокая для слабых сбоев; H^h — низкая для сильных сбоев.

3. Обозначения параметров: $\tau_к$ — период контроля; $\tau_ц$ — время цикла; g — величина модуля подсчета; l — разрядность программного счетчика; $\varkappa_0$ — вероятность искажения одной линии из p в выходных шинах МП; m_0 — разрядность МП; $m_к$ — разрядность кода метки; k — кратность невыявленных ошибок.

истинной управляющей информации на ложную. Однако для системы контроля МПСУ такая информация является корректной, что весьма затрудняет обнаружение и исправление подобного рода ошибок. В табл. 8.4 приведены оценки эффективности применения в МПСУ рассмотренных выше методов контроля к указанному типу сбоев в программном счетчике без учета влияния других типов сбоев.

Эффективность соответствующего метода контроля можно оценивать по параметру E_o: чем он меньше, тем больше вероятность восстановления вычислительного процесса $E_в$ в МПСУ ($E_o + E_в = 1$). В общем виде величину E_o можно определить как

$$E_o^{(i)} = \int_0^\infty \left[1 - P_{ср}^{(i)}(\tau) P_{об}^{(i)}\right] \varphi(\tau)\, d\tau,$$

где $P_{ср}^{(i)}(\tau)$ — вероятность выработки системой контроля диагноза состояния МПСУ за время τ (степень оперативности системы контроля); $P_{об}^{(i)}$ — вероятность обнаружения сбоя системой контроля; $\varphi(\tau)$ — плотность вероятности наступления отказа МПСУ за время τ после сбоя; i — порядковый номер анализируемого метода контроля ($i = \overline{1, 4}$).

Учитывая, что на протяжении цикла $\tau_ц$ между двумя последовательными обращениями к ПЗУ сбой программного счетчика МП может произойти в любой момент с равной вероятностью, можно записать, что

$$\varphi(\tau) = \begin{cases} 1/\tau_ц, & \text{если } \tau \leqslant \tau_ц, \\ 0, & \text{если } \tau > \tau_ц. \end{cases}$$

Тогда имеем

$$E_o^{(i)} = \frac{1}{\tau_ц} \int_0^{\tau_ц} \left[1 - P_{ср}^{(i)}(\tau) P_{об}^{(i)}\right] d\tau = 1 - \frac{P_{об}^{i}}{\tau_ц} \int_0^{\tau_ц} P_{ср}^{i}(\tau)\, d\tau.$$

Из табл. 8.4 следует, что наиболее эффективными являются методы контроля посредством меток операторов с контрольным суммированием или контрольных алгоритмов, характеризующиеся высокой вероятностью обнаружения слабых и сильных сбоев, достаточной оперативностью и широтой контроля, но вместе с тем и наибольшей аппаратной избыточностью.

Восстановление вычислительного процесса после сбоя в МПСУ производится на базе рестарта программы или помехоустойчивого кодирования.

Рестарт программы обеспечивается введением в тело программы последовательности контрольных точек с записью в них информации о текущем состоянии конкретного процесса, что позволяет повторно запускать неверно выполненный в результате сбоя фраг-

мент программы с зафиксированной последней по ходу вычислительного процесса точки. При потере информации об этой точке происходит рестарт программы к зафиксированной предпоследней контрольной точке и т. д., пока не обнаружится точка с корректной информацией. Может применяться и другая дисциплина рестарта, но в любом случае рестарт программы должен происходить только в точки с надежным хранением информации и учитывать специфику функционирования конкретного объекта. Например, стратегия начального запуска, часто используемая в ЭВМ общего пользования, не может быть применена в МПСУ ГПК из-за возможности аварийных ситуаций при незапланированном движении исполнительных органов к исходной точке.

Надежность хранения информации при сбое элементов памяти в БИС ОЗУ достигается введением помехоустойчивого кодирования и соответствующего механизма восстановления с реализацией их на программном либо аппаратном (внешнем или встроенном в ЗУ) уровнях. Например, способность БИС ЗУ к восстановлению информации в памяти с помехоустойчивым кодированием и механизмом коррекции всех одиночных ошибок при наличии слабых и сильных сбоев можно оценить по вероятности искажения $P_{\text{иск}}$ данных:

$$P^s_{\text{иск}} \approx V P^s_{\text{о иск}} \approx V C_m^2 \varkappa_0^2 (1-\varkappa_0)^{m-2} \approx V C_m^2 \varkappa_0^2,$$

где $P_{\text{о иск}}$ — вероятность искажения одного слова памяти (вероятность появления ошибки двойной кратности); V — число ячеек памяти; C_m^2 — число сочетаний из m по 2; $m = m_{\text{о}} + m_{\text{к}}$ — разрядность ЗУ с $m_{\text{о}}$ информационными и $m_{\text{к}}$ контрольными разрядами $(2^{m_{\text{к}}-1} < 1 + m_0 + m_{\text{к}} \leqslant 2^{m_{\text{к}}})$.

$$P^h_{\text{иск}} = \left(2^{V(m_{\text{о}}+m_{\text{к}})} - \frac{2^{V(m_{\text{о}}+m_{\text{к}})}}{2^{Vm_{\text{о}}}}\right) \Big/ 2^{V(m_{\text{о}}+m_{\text{к}})} = 1 - 2^{-Vm_{\text{о}}},$$

где $2^{V(m_{\text{о}}+m_{\text{к}})}(2^{Vm_{\text{о}}})$ — число общих состояний (информационных состояний) ЗУ; $2^{V(m_{\text{о}}+m_{\text{к}})}/2^{Vm_{\text{о}}}$ — число состояний памяти на одно информационное слово.

Отсюда следует, что восстановление вычислительного процесса в БИС ЗУ известными методами может быть эффективным только при наличии слабых сбоев в МПСУ, так как в условиях сильных сбоев память с помехоустойчивым кодированием адекватна обычному ЗУ. При этом имеет место значительная избыточность ОЗУ (50—75 % при $m_{\text{о}} = 4, 8$), что ведет к росту нагрузки на источник питания, снижению быстродействия и производительности системы памяти, увеличению $\varkappa_{\text{о}}$, ухудшению помеховой обстановки и условий для введения программной избыточности.

Реализация в МПСУ вычислительных процессов, уклоняющихся от сбоев (FA-процессов). Как указывалось ранее, реализовать вычислительные процессы, уклоняющиеся от сбоев, можно путем ор-

ганизации вычислений, минимизирующей вероятность перерастания структурного сбоя в функциональный $E_п$, фактически характеризующей степень восприимчивости вычислительного процесса к сбоям аппаратуры в МПСУ. Применительно к МПСУ ГПК задача реализации FA-процессов связывается с минимизацией вероятности сбоев системы в режимах задания и отработки функций управления, что может быть обеспечено повышением помехоустойчивости разрабатываемых программных модулей (ПМ) и предотвращением сбоев памяти МПСУ в указанных режимах функционирования. Рассмотрим эти пути повышения помехоустойчивости МПСУ более детально.

Обеспечение помехоустойчивости ПМ. Пусть имеется некоторый ПМ, характеризующийся одним входом XI и одним выходом XO, выполняемой функцией преобразования $XI \to XO$ и объемом памяти, составляющим V ячеек. Любой функциональный сбой может привести к неверному исполнению ПМ с вероятностью $r_с$, равной

$$r_с = 1 - \exp(-e_ф \bar{t} \bar{\Lambda}_п) = 1 - \exp[-(e^s \varkappa_в \bar{V} + e^h) \bar{t} \bar{\Lambda}_п] =$$

$$= 1 - \exp[-(\varkappa_в \bar{V} \Lambda_п^s + \Lambda_п^h) \bar{t}],$$

где $\bar{V}$, $\bar{t}$ — средние значения объема используемой памяти и времени реализации модуля, $\varkappa_в$ — вероятность искажения информации в ячейке ЗУ под влиянием слабой помехи (для ЗУ без механизма коррекции сбоев $\varkappa_в = m_о \varkappa_о$).

Для ПМ с линейной структурой, характеризующегося объемом ЗУ $V = V(t)$ и временем t исполнения программы за интервал $[0, t]$, имеем

$$\bar{V} = \frac{1}{t} \int_0^t V(t)\, dt; \quad \bar{t} = t.$$

Для ПМ более сложной структуры имеют место соотношения

$$\bar{V} = \sum_{j \in T} P_j \frac{1}{t_j} \int_0^{t_j} V_j(t)\, dt = \sum_{\rho \in ПМ} V_i \frac{\rho_i}{\rho_{ПМ}}; \quad \bar{t} = \sum_{j \in T} P_j t_j,$$

где T — множество вариантов реализации ПМ; $P_j(t_j)$ — вероятность (время) реализации ПМ по j-му варианту; V_j — функция изменения объема ЗУ в процессе реализации ПМ по j-му варианту; V_i — объем ЗУ; $\rho_i/\rho_{ПМ}$ — доля времени, приходящаяся на реализацию в ПМ некоторой i-й элементарной процедуры.

В целом $\bar{V}$ и $\bar{t}$ являются конструктивными параметрами, отражающими структурные свойства ПМ, поэтому можно отыскать условия оптимального построения структуры ПМ, при которых величина $r_с \to \min$. Так как при сильных помехах в МПСУ всегда может произойти нарушение хода вычислительного процесса на лю-

бой его стадии, то оптимизация может быть корректной лишь применительно к слабым помехам. Критерием оптимальности структуры ПМ в смысле способности уклонения от воздействия слабых помех может служить показатель динамического объема ЗУ $D=\overline{Vt}$, минимизации которого можно добиться распараллеливанием вычислительных процессов и последовательной их реализации в виде отдельных ПМ, что характерно для однопроцессорных МПСУ.

Пусть распараллеливаемый модуль C (рис. 8.2) состоит из ряда $c_i|i=\overline{1,\ m}$ ПМ, стоящих на определенных позициях и выполняемых последовательно друг за другом. Каждый из ПМ имеет множество разнообразных (пересекающихся или непересекающихся) $XI_{эj}$ входных и XO_i выходных данных $\left(\bigcup_{j=1}^{k} XI_{эj}=\bigcup_{l=1}^{m} XI_l^{(l)}=XI;\right.$ $\left.\bigcup_{i=1}^{m} XO_i^{(l)}=XO;\ l=\{\text{мд, пз}\}\right.$ — множество входных данных i-го ПМ либо некоторого ПМ, стоящего на i-й позиции), занимающих соответственно $\sum_{j=1}^{k} VI_{эj}^{(l)}=VI$ и $\sum_{l=1}^{m} VO_i=VO$ объем ЗУ, необходимый для хранения данных из множеств $XI_{эj}$, $XI_l^{(l)}$, $XO_i^{(l)}$. Тогда можно записать

$$D=\overline{Vt}=\sum_{l=1}^{m}\overline{V}_l^{(\text{пз})}\overline{t}^{(\text{пз})}=\sum_{l=1}^{m}D_l^{(\text{пз})}+\sum_{l=1}^{m}t_l^{(\text{пз})}VR_l^{(\text{пз})},$$

где $V_l^{(\text{пз})}\left(\bar{t}_l^{(\text{пз})}\right)$ — средний объем памяти (среднее время), затраченный на реализацию i-го ПМ; $D_{jl}^{(\text{пз})}$ — динамический объем памяти i-го ПМ; $VR_l^{(\text{пз})}$ — объем ЗУ, используемый для хранения данных (входные данные для последующих ПМ, стоящих на j-х позициях $(i<j)$, и выходные данные предыдущих ПМ, стоящих на k-позициях $(k<i)$).

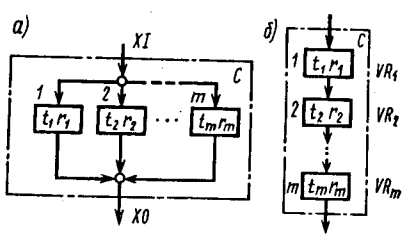


Рис. 8.2. Преобразование распараллеливаемого программного модуля C в последовательность модулей при реализации в однопроцессорной системе

Минимизировать выражение $D=\overline{Vt}$ можно за счет второго слагаемого, имеющего в развернутой форме следующий вид:

$$\sum_{l=1}^{m}t_l^{(\text{пз})}\cdot VR_l^{(\text{пз})}=\sum_{l=1}^{m}t_l\left[\sum_{l=1}^{l-1}VO_l^{(\text{пз})}+\sum_{l=l+1}^{m}VI_l^{*(\text{пз})}\right]$$

$$=\sum_{l=1}^{m}\bar{t}_{al}^{(\text{пз})}\left[\sum_{l=1}^{l-1}VO_{al}^{(\text{мд})}+\sum_{l=l+1}^{m}\sum_{j=1}^{k}V_{эj}\bigcup_{g=l+1}^{m}\eta_{ag}^{(\text{мд})}\right]\to\min;$$

$$\bar{\alpha}=\{\alpha_i\}|\, i=\overline{1, m};\ \alpha_i \neq \alpha_j, \text{ если } i \neq j,$$

где VI_l^* — объем памяти, отводимый под входные данные $XI_l^{(\text{пз})}$ для α_l-го ПМ, не используемые последующими k-ми ПМ ($k>l$); η — индикатор принадлежности входных данных $XI_{эj} \subset XI_i^{(l)}$.

Минимизации динамических объемов памяти под промежуточные данные, генерируемые программой в процессе реализации вычислений, можно добиться за счет второго слагаемого путем полного перебора и выбора соответствующего варианта, определяющего последовательность реализации конкретных ПМ, значительно влияющего на величину $\bar{V} \cdot \bar{t}$. Однако при m распараллеливаемых ПМ и полном переборе необходим громоздкий анализ $m!$ вариантов. Упростить процедуру выбора оптимальной последовательности реализуемых ПМ можно путем выполнения следующего алгоритма, сокращающего число анализируемых вариантов с величины $m!$ до m.

1. Для каждого i-го ПМ, входящего в общий распараллеливаемый модуль, определяется критерий K:

$$K_i=\Delta V_i^{(\text{мд})}/\bar{t}^{(\text{мд})}=(VO_i^{(\text{мд})}-VI_i^{(\text{мд})})/\bar{t}_i^{(\text{мд})}.$$

2. Произвести расположение ПМ в цепочке реализации в соответствии с ростом величины критерия K_i по условию

$$\frac{\Delta V_1^{(\text{пз})}}{\bar{t}_1^{(\text{пз})}} \leqslant \frac{\Delta V_2^{(\text{пз})}}{\bar{t}_2^{(\text{пз})}} \leqslant \ldots \leqslant \frac{\Delta V_m^{(\text{пз})}}{\bar{t}_m^{\text{пз}}}.$$

При таком подходе к размещению ПМ на определенном месте в последовательной цепочке модулей согласно указанной процедуре можно получить эффективную организацию распараллеливаемых вычислительных процессов и структуры ПМ, требующую минимального динамического объема памяти для хранения промежуточных данных, генерируемых программой в процессе реализации вычислений.

Другим способом сокращения объема памяти является выбор рационального способа хранения информации, управляющей ходом вычислительного процесса. Опыт эксплуатации МПСУ, например, показывает, что применение стековой организации памяти в условиях влияния помех не эффективно из-за возможности тяжелых сбоев, что серьезно ограничивает или полностью исключает использование стековых операций и требует всестороннего учета специфики применяемых способов программирования.

Одним из известных неструктурированных способов программирования является, например, метод BS-программ, широко использующий в структуре ПМ операторы типа GO TO и требующий большой квалификации программистов. С помощью данного метода для несложных задач создан ряд эффективных программ при полном отсутствии стековых операций (механизма подпрограмм), но с низ-

кой читаемостью и сопровождаемостью программ, ограничивающих широкое применение метода. Другим распространенным способом программирования является структурный подход, характеризующийся жесткими правилами по построению структуры, составу операторов типа следования, ветвления, цикла, разработке встроенных подпрограмм с обращением к ним с помощью команд вызова CALL и возврата RETURN. Данный способ существенно упрощает программирование, но требует применения стековых операций, что ведет к увеличению динамического объема стековой памяти, являющегося в силу указанных причин нежелательным явлением. Вместе с тем большое число функций управления ГПК можно реализовать посредством циклических операций над векторами входных и выходных данных или сведением к ним. Например, обработку n-компонентных векторов можно вести с помощью некоторых стандартных процедур на базе m-компонентных вектор-функций ($m<n$), вид которых зависит от характера обрабатываемых данных. Рассмотрим этот подход.

Пусть сложный ПМ построен по правилам структурного программирования и состоит из последовательности: а) простых ПМ, обозначенных $B_j|j=\overline{1, m}$ и включающих в себя ряд подпрограмм, реализующих определенные повторяющиеся процедуры $b_k|k=\overline{1, l}$; б) операторов инкрементирования указателя ($i=i+1$) обрабатываемой информации, размещенных в различных местах программы. В этом случае можно использовать параметр i для обращения к тому или иному простому модулю B_j в соответствии с ходом вычислительного процесса и преобразовать линейную структуру сложного ПМ в циклическую путем введения оператора многозначного условного перехода (МУП). В целом МУП имеет один вход и $n+1$ выходов, каждый из которых возбуждается в определенный момент времени и обеспечивает настройку процесса на исполнение той или иной функции в зависимости от состояния указателя. Первые n выходов МУП в зависимости от структуры исходного сложного ПМ объединяются в m групп, каждая из которых обеспечивает выборку определенного ПМ в конкретный момент времени, а оператор $i=i+1$ помещается в основание структуры с образованием цикла. При такой организации структуры МУП: а) не требуется реализации процедур b_k в виде подпрограмм второго уровня, что ведет к уменьшению динамического объема стековой памяти; б) подпрограммы, построенные на базе МУП-цикла, мало критичны к искажениям параметра i; в) несмотря на отсутствие стековых операций, сохраняются важнейшие достоинства структурного подхода (любой блок программы может быть вызван оператором МУП несколько раз и допустима любая перестановка блоков b_j); г) при распараллеливаемой структуре сложного ПМ оптимизировать вычислительный процесс можно за счет минимизации динамического объема памяти промежуточных результатов, описанной выше, что

выполнимо даже на последнем этапе разработки программ (после отладки) путем изменения лишь конфигурации выходных связей оператора МУП. В качестве примера на рис. 8.3 приведен вариант реализации оператора МУП.

В целом построение эффективных вычислительных FA-процессов МПСУ на базе минимизации динамического объема памяти способствует повышению помехоустойчивости подсистемы памяти МПСУ в условиях действия слабых помех и не требует введения

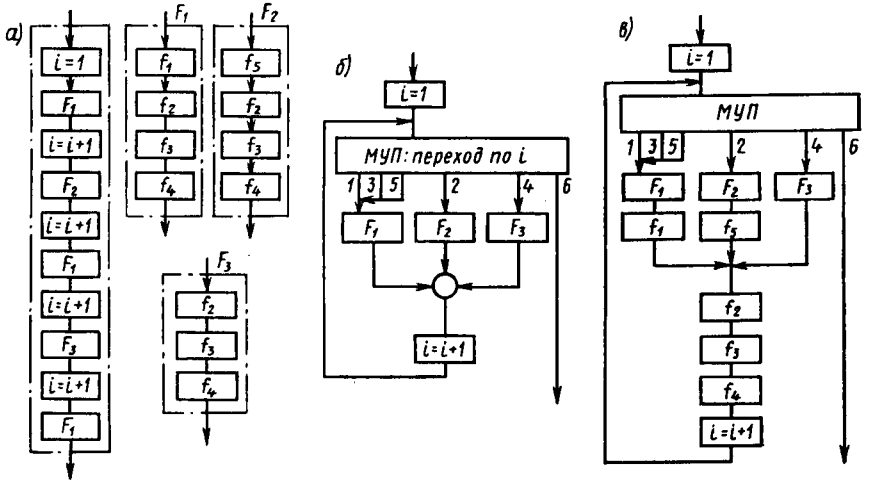


Рис. 8.3. Вариант организации оператора многозначного перехода (МУП) при наличии повторяющихся процедур и оператора инкрементирования указателя:
а — исходный вариант линейной структуры ПМ; *б* — промежуточный укрупненный вариант; *в* — окончательный вариант МУП с циклической структурой

дополнительных программно-аппаратных затрат. Если провести сравнение данного метода с широко распространенным методом помехоустойчивого кодирования памяти по критерию вероятности неверного исполнения функции хранения информации $r_c = 1 - \exp(-\varkappa_в V t \Lambda_п^s)$, то в качестве сопоставимого параметра может выступать величина $e = \varkappa_в V$, представляющая собой вероятность искажения информации при воздействии одиночной слабой помехи. Тогда при заданном интервале времени помехоустойчивое кодирование обеспечивает снижение величины $\varkappa_в$, а минимизация динамического объема — уменьшение объема памяти. При m_o-разрядной МПСУ будем иметь:

$$e^{(1)} \approx \varkappa_в V^{(1)} = m_0 \varkappa_0 V^{(1)};$$

$$e^{(2)} \approx \varkappa_b^{(2)} V = (m_0 + m_к)(m_0 + m_k - 1)\varkappa_0^2 V l/2;$$

$$\beta = e^{(1)}/e^{(2)},$$

где (1), (2) — индексы идентификации методов минимизации динамического объема и помехоустойчивого кодирования соответственно; $\varkappa_0$ — вероятность сбоя одного разряда памяти; m_0 (m_k) — число информационных (контрольных) разрядов МПСУ; β — коэффициент эффективности.

Сравнительный анализ эффективности использования этих методов для повышения надежности МПСУ показывает, что существует область ($\beta > 1$), которую трудно получить из-за малости величины $\varkappa_0$, но в которой метод минимизации динамического объема памяти эффективнее метода помехоустойчивого кодирования. Более эффективно совместное применение обоих методов.

Реализация вычислительных процессов, нечувствительных к сбоям (FT-процессов). Построение таких процессов базируется на введении дополнительных программно-аппаратных средств контроля и восстановления, обеспечивающих нормальное функционирование МПСУ и протекающих в них вычислительных процессов. Среди методов контроля особый интерес с точки зрения повышения помехоустойчивости МПСУ вызывают методы контроля динамических контрольных сумм (ДКС) и программной синхронизации. Рассмотрим их более подробно.

Методы динамических контрольных сумм. Для однопроцессорных МПСУ характерны процедуры обмена информацией между МП и ПЗУ команд, связанные с последовательным прохождением соответствующих сигналов по тракту «программный счетчик → шина адреса → ПЗУ → шина данных» с отработкой следующих операций. Программным счетчиком (ПС) МП задается адрес очередной команды, который по ША поступает на адресные входы ПЗУ. Далее по сигналу «Чтение» (Чт) код операции выдается на ШД, и после поступления его на МП он декодируется. Затем МП формирует новый адрес с повторением последовательности действий, причем в зависимости от типа МП возбуждение указанного тракта происходит от одного до трех раз, а искажение информации в любом из узлов тракта приводит к нарушению нормального хода программы. Для исключения этого нежелательного явления часто применяют оперативный контроль функционирования тракта обмена с использованием метода подсчета контрольных сумм (КС), широко применяемого в микропрограммных автоматах и заключающегося в сопоставлении каждого кода команды C_i с некоторой КС Σ_i:

$$\Sigma_i = F(C_i,\ C_{i-1},\ C_{i-2},\ldots) = (C_i + \Sigma_{i-1}) \bmod g,$$

где g — модуль суммирования.

При сложной конфигурации программ часто невозможно использовать КС в классе методов меток операторов без применения специальных процедур. Одна из подобных процедур называется уравновешиванием сумм программ (УСП) и обеспечивает соответствие любой произвольной точки программы динамической конт-

рольной сумме (ДКС), остающейся неизменной на протяжении всего вычислительного процесса, что позволяет использовать метод КС в классе меток операторов, в каждой точке программы производить подсчет ее ДКС и сравнение текущего значения ДКС с эталонным. При этом для базовых операторов (следования, ветвления, цикла), входящих в структуру любых сложных программ, постоянство ДКС определяется следующим образом.

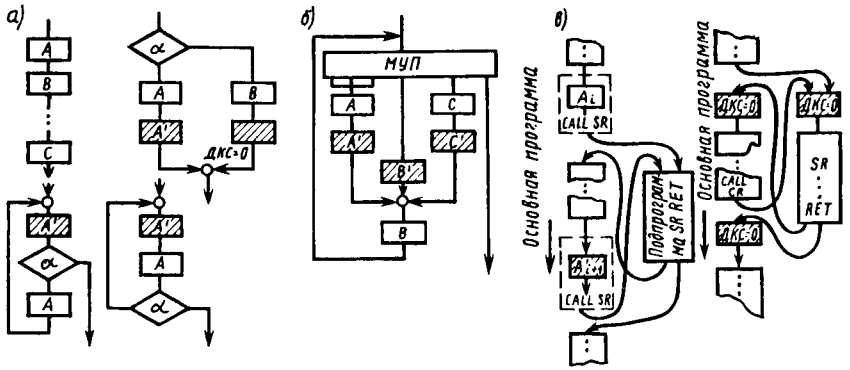


Рис. 8.4. Преобразование программных модулей с уравновешиванием контрольных сумм:
а — для базовых операторов; *б* — для оператора МУП; *в* — для иерархического уравновешивания

В общем виде ДКС в любой точке программы определяется по формуле

$$\Sigma O = (\Sigma I + \Delta) \bmod g,$$

где $\Sigma I\,(\Sigma O)$ — ДКС программы в момент вызова (окончания) базового оператора; Δ — результирующая ДКС базового оператора ($\Delta = \text{const}$).

В соответствии со сказанным имеем:

— для оператора следования $\Delta = (A + B + \ldots + C) \bmod g$;

— для оператора ветвления $\Delta = \begin{cases} (\alpha + A) \bmod g, \\ (\alpha + B) \bmod g, \end{cases}$

— для оператора цикла $\Delta = [n(\alpha + A) + \alpha] \bmod g$,

где α, A, B, C — результирующие ДКС элементарных блоков; n — число вхождений в цикл.

Перечисленные операторы обладают свойством УСП при следующих условиях: оператор следования — всегда; оператор ветвления, если $(\alpha + A) \bmod g = (\alpha + B) \bmod g$; оператор цикла, если $(\alpha + A) \bmod g = 0$.

Постоянство ДКС программ наиболее просто обеспечивается путем введения дополнительных блоков (A', B') (рис. 8.4, *а*, *б*), результирующие ДКС которых уравновешивают суммы отдельных

ветвей операторов до необходимой величины. Окончательный вид их следующий:

— для оператора ветвления $A'=(B-A) \bmod g$;

— для оператора цикла $A'=(-A-\alpha) \bmod g$.

Заметим, что отрицательное значение, получаемое при вычислении результирующих ДКС, означает их дополнение до g.

Наряду с одноуровневыми УСП могут быть реализованы иерархические УСП при организации вычислений с использованием подпрограмм. В этом случае в программу высшего уровня вводятся уравновешивающие блоки A_1', A_2', ... (рис. 8.4, *в*) с размещением их непосредственно перед операцией CALL в каждой i-й точке вызова подпрограммы. Величина ДКС для каждого из блоков A_i' определяется по формуле

$$A_i'=(\Sigma I-\Sigma I_i) \bmod g, \quad i \in \overline{1, n},$$

где ΣI_i — ДКС программы высшего уровня в i-й точке вызова подпрограммы без уравновешивающего блока; n — количество точек вызова подпрограммы; $\Sigma I=\text{const}$ — ДКС подпрограммы.

Реализация уравновешивающих блоков в УСП производится как естественным способом (использованием специальных чисто программных процедур), так и директивным способом (путем аппаратного обнуления ДКС). Например, естественное формирование ДКС для МП К580 реализуется универсальной процедурой BALANCE в расширенном или упрощенном варианте.

В обоих случаях желательно, чтобы модуль суммирования g не превышал величину 2^{m_0} (m_0-разрядного слова команды), так как иначе неизбежны значительные программные затраты на реализацию ДКС.

Если МПСУ построена на базе микропрограммируемого МП или $g>2^{m_0}$, то применяется директивное уравновешивание ДКС с принудительным обнулением контрольной суммы с помощью специальных команд (микрокоманд) типа ДКС=0, которые и играют роль уравновешивающих блоков.

При выборе того или иного способа уравновешивания ДКС, производимого в процессе реализации ПО, необходимо учитывать специфику методов контроля и эффективность, получаемую от их применения. Например, эффективность контроля хода УСП зависит от модуля подсчета и разрядности ДКС: чем выше разрядность ДКС, тем эффективнее контроль, но вместе с тем растет разрядность метки, а следовательно, объем памяти ПЗУ команд, ограничивая применение метода КС в МПСУ. Устранить данный недостаток можно процедурой свертывания кодов ДКС, реализуемой с помощью схемотехнических решений.

В целом введение средств контроля, осуществляющих оперативный контроль тракта передачи информации между МП и ПЗУ команд, обеспечивает обнаружение и искажение информации и сбоя

в любом из узлов МПСУ и аварийное прерывание хода вычислительного процесса системы.

С помощью метода УСП с равной вероятностью можно обнаружить как s-, так и h-сбои программного счетчика МП. В соответствии с ранее выведенными соотношениями имеем

$$F_0^{(s)}=E_0^h=\frac{1}{\tau_ц}\int_0^{\tau_ц}[1-P_{ср}(\tau)P_{об}]\,d\tau=\frac{1}{\tau_ц}\int_0^{\tau_ц}(2^{-m}g+2^{-lm_k})\,d\tau=$$

$$=2^{-m}g+\frac{\theta}{\tau_ц}\sum_{l=1}^{\tau_ц/\theta}2^{-lm_k}\approx 2^{-m}g+\frac{\theta}{\tau_ц}\cdot\frac{2^{-m_k}}{1-2^{-m_k}}\approx 2^{-m}g,$$

где $\theta\left(\frac{\tau_ц}{0}\right)$ — среднее время (среднее число обращений) между соседними обращениями к ПЗУ команд; $\frac{\theta}{\tau_ц}2^{-m_k}\Big/\left(1-2^{-m_k}\right)$ — вероятность того, что функциональный отказ системы наступит из-за малой оперативности обнаружения искажений ДКС.

Используя сведения табл. 8.4, можно отметить следующее:

а) при больших значениях $\tau_ц/\theta$ метод УСП по параметру E_0 адекватен методу КС при сокращении требуемого объема памяти ПЗУ меток в m_0/m_g раз (для МП К580, например, до 8 раз);

б) по сравнению с методом подсчета операторов метод УСП обладает высокой эффективностью, особенно при малой избыточности аппаратуры (чем избыточность меньше, тем относительная эффективность выше), причем повышение эффективности может осуществляться путем изменения разрядности ДКС (модуля суммирования g) при постоянной разрядности ПЗУ меток.

Метод программной синхронизации. Контроль за нормальным функционированием МПСУ может вестись на базе метода программной синхронизации. Метод обеспечивает разделение множества вырабатываемых МП адресных и управляющих слов на разрешенные и запрещенные. Так как закон генерирования слов определяется спецификой функционирования системы, ее управляющей программой, то искажения вырабатываемых МП как отдельных слов, так и их последовательностей (предложений) могут быть легко обнаружены и каким-либо образом исправлены.

В общем виде отдельное слово S_i, вырабатываемое МП, может быть представлено в виде

$$S_i=(A_n,\ldots, A_1, U_m,\ldots, U_1),$$

где $A_i(U_j)\in\{0, 1\}$, $i=\overline{1, n}$; $j=\overline{1, m}$,— значение i-го (j-го) разряда ША (ШУ) конкретного МП; n, m — разрядность ША и ШУ.

Тогда множество всех слов S, вырабатываемых МП, составляет $S=2^{n+m}$, из него множество рабочих (разрешенных) слов $S_р\in S$ определяется спецификой работы МПСУ, а остальные слова могут

быть отнесены в множество запрещенных слов $S_з \in S$, причем $S_р \cup S_з = S$, $|S_р| \ll |S_з|$. Поэтому если каждое слово, генерируемое МП в процессе исполнения управляющей программы, контролировать на предмет принадлежности к множеству $S_р$ или $S_з$, то можно сделать вывод об искажении реализуемой программы на определенном шаге ее исполнения. Эффективность такого метода контроля однозначно определяется величиной $K_э$ ($K_э = |S_з|/|S_р|$ (чем она больше, тем контроль эффективнее). Для контроля правильности функционирования МПСУ на базе МП К580 предложен ряд типовых структур блоков синтаксического контроля слов и отдельных предложений, позволяющих осуществлять прерывание или продолжать работу МП в зависимости от складывающейся ситуации при контроле. Эффективность метода программной синхронизации на базе анализа синтаксического контроля слов можно оценить следующим образом:

$$E_0^s = 1 - P_{об}^s = 1 - \sum_l \rho_l P_{об\,l}^s = 1 - \sum_l \rho_l \frac{\varkappa_l}{n+m};$$

$$E_0^h = 1 - P_{об}^h = 1 - \frac{S_з}{S_р} = 1 - K_э,$$

где ρ_l — вероятность того, что сбой приходится на l-е слово; $P^s_{об\,l}$ — вероятность обнаружения s-сбоев в l-м слове; $\varkappa_l$ — количество разрядов l-го слова, в которых БСК обнаружен сбой.

Так как МП имеют значительную функциональную избыточность, то коэффициент $K_э$ велик и может быть достигнута высокая помехоустойчивость системы. Кроме того, можно разгрузить механизм контроля хода вычислительных процессов на основе УСП, охватывать проверкой всю систему памяти, а не только ПЗУ. В целом по своим функциям программная синхронизация адекватна методу контрольных алгоритмов при гораздо меньшей избыточности по аппаратуре (примерно в 2,5 раза по сравнению с дубль-процессором).

Процедуры восстановления вычислительных процессов в МПСУ ГПК. Основой вычислительных процессов, характерных для многих МПСУ ГПК, является преобразование управляющей программы в отработку пространственного положения совокупности исполнительных устройств и приводов ОУ, причем в каждый момент времени по их текущему положению, а также по состояниям датчиков текущего контроля можно с высокой степенью достоверности судить о ходе вычислительного процесса в МПСУ ГПК. Зная, например, управляющую программу и текущее состояние ОУ, можно определить, какое пространственное положение он займет в следующий момент времени, какой кадр программы должен воспроизводиться.

В этой ситуации, если в памяти МПСУ происходит тяжелый сбой, искажающий полностью всю информацию, по показаниям

счетчика кадров, реализованного на базе энергонезависимого ППЗУ, по сигналу прерывания можно восстановить вычислительный процесс следующим путем: а) МП подготавливается к запуску процедуры восстановления (установка регистров, указателя стека, системы флагов в соответствующее положение); б) считывается текущее состояние ОУ с формированием вектора уставок; в) по счетчику кадров из ППЗУ выбирается текущий кадр K_i, подвергшийся сбою; г) реализуется новое значение уставки $У_i$ с учетом нового и предыдущего состояний.

$$СК_i = f(СК_{i-1},\ Л_{i-1});\ K_i = \langle СК_i \rangle;\ У_i = \varphi(K_i,\ У_{i-1}),$$

где $СК_{i-1}$, $СК_i$ — прежнее и новое состояния счетчика кадров; $Л_{i-1}$ — логические функции, получаемые в процессе исполнения предыдущего кадра, по которым производится ветвление процесса; K_i — очередной кадр управляющей программы; $У_i(У_{i-1})$ — уставка в новом (предыдущем) кадре.

Таким образом для осуществления процедуры восстановления в этом случае достаточно обеспечить надежное сохранение информации только об исполнительной части объекта управления (содержимого счетчика кадров), а вся остальная информация может быть полностью восстановлена даже в случае ее полного искажения. При этом не требуются дополнительные программные средства на введение контрольных точек, позволяющих защищаться от слабых и сильных сбоев.

В целом применение методов УСП для организации основных функций контроля хода вычислений и программной синхронизации для обнаружения сбоев, пропускаемых методом УСП, а также счетчика кадров УП для восстановления процесса позволяет получить высокую эффективность функционирования МПСУ ГПК с реализацией помехоустойчивых вычислительных процессов.

§ 8.5. ОСОБЕННОСТИ И ОЦЕНКА КАЧЕСТВА ПРОЕКТИРОВАНИЯ БОРТОВЫХ СИСТЕМ АВТОМАТИКИ

Проектирование бортовых систем автоматики (БСА) обусловливается рядом специфических факторов, связанных с общими и специальными требованиями, предъявляемыми к бортовым системам управления (БСУ) с целью улучшения тактико-технических характеристик объектов управления, необходимостью учета жестких условий эксплуатации — больших перепадов температур (10—150 °С), относительной влажности (до 100 %), действия механических нагрузок в диапазоне частот от единиц герц до единиц килогерц при ускорениях от единиц до десятков g [50, 51].

Особенности алгоритмов функционирования БСА зависят от широты диапазона и характера управляющих воздействий, номенклатуры абонентов (датчиков, исполнительных механизмов, отдельных

приборов, подсистем и УВВ), разнообразия режимов при работе в РМВ, возрастания «цены» ошибок из-за неверного функционирования программно-аппаратных средств. Воздействия в типовой структуре БСУ определяются управляющими входными сигналами со стороны БЦВМ, изменениями внутренних состояний системы, происходящих под влиянием внешних сигналов, сигналов, поступающих от БСУ и инициирующих работу БСА и БЦВМ. Решение конкретных задач в РМВ при штатном использовании объектов управления в сложных условиях эксплуатации требует надежного функционирования БСА, обеспечиваемого соответствующими программно перестраиваемыми средствами и режимами работы.

Большое разнообразие существующих классов и типов БСА определяется множеством структур БСУ, типов управляющих БЦВМ, элементной базой и др. БСА характеризуются многообразием реализуемых функций, форм и способов преобразования, представления и передачи информации; предварительной обработки данных; широтой диапазонов скоростей обмена данными; множеством (50—1000 и более) разнотипных абонентов, их функциональных возможностей и режимов работы. Все это в комплексе влияет на структуру проектируемого БСА, выбор конкретной элементной базы, состав разрабатываемого программного обеспечения, конструктивных решений, способов поддержания заданного уровня надежности аппаратных и программных средств.

Обычно сложные БСА оценивают следующим комплексом технических характеристик:

$$\mathbf{X}=\{\mathbf{X}_{об},\ \mathbf{X}_{с},\ \mathbf{X}_{по},\ \mathbf{X}_{э},\ \mathbf{X}_{к}\},$$

где $\mathbf{X}_{об}$ — характеристики обслуживания (среднее время $\bar{T}_{об}$ и вероятность $P_{об}$ обслуживания; требования на обмен; среднее время $T_{пр}$ и вероятность простоя $P_{пр}$ канала обмена; вероятность нахождения системы управления в одном из рабочих состояний $P_{сост}$); $\mathbf{X}_{с}$ — характеристики структур (тип $M_{т}$, вид $M_{в}$ и режим $M_{р}$ обмена; тип управления; число каналов управления и обмена; способы передачи кодов и их типа; параметры контроля; уровень программируемости, адаптивности, перестраиваемости, надежности функционирования $P_{нф}$; пропускная способность $C_{пр}$; тип интерфейса; электрические и временные характеристики входных и выходных сигналов; тип используемой элементной базы); $\mathbf{X}_{по}$ — характеристики ПО (тип операционной системы, язык программирования $Я_{п}$, количество сервисных и стандартных программ, требуемый объем памяти для хранения ПО); $\mathbf{X}_{э}$ — характеристики условий эксплуатации (параметры среды и первичной сети электропитания, время непрерывной работы, технический ресурс, параметры микроклимата $X_{ср}$, время готовности к работе); $\mathbf{X}_{к}$ — многочисленные характеристики конструкций (основными из которых являются весогабаритные характеристики, масса $G_{а}$, тип конструкции, энергопотреб-

ление W, коэффициенты технологичности, унификации и стандартизации, ремонтоспособность, стоимость $C_а$ и др.).

На практике же в зависимости от назначения БСА с достаточной достоверностью используют минимальную совокупность технических характеристик $\mathbf{X}_{min}$. Например, для бортовых систем обработки информации (БСОИ) в [50] используют набор из 13 параметров:

$$X_{min}=\{\overline{T}_{об},\ P_{сост},\ M_т,\ M_в,\ M_р,\ C_{пр},\ P_{нф},\ m_а,\ Я_п,\ X_{ср},\ G_а,\ W,\ C_а\},$$

где $m_а$ — число абонентов.

При ориентировочной оценке нескольких вариантов реализации БСОИ применяется более усеченный набор из 4 параметров:

$$X_{ор}=\{M_р,\ C_{пр},\ m_а,\ G_а\},$$

где $C_{пр}=(1\div5)\cdot10^7$ бод; $m_а=50\div1000$; $G_а=5\div36$ кг [50].

Оценить качество аппаратной реализации БСА на структурном и функционально-логическом уровнях проектирования можно с помощью следующих комплексных показателей (табл. 8.5):

$$X=\{X_{тп},\ X_э,\ X_с\},$$

где $\mathbf{X}_{тп}$ — подмножество технических параметров; $\mathbf{X}_э$ — подмножество параметров эксплуатации; $\mathbf{X}_с$ — подмножество параметров структуры управления.

Эти показатели можно классифицировать также по таким признакам, как источники получения, характер стабильности и детерминированности, влияния на структуру и тип связей (табл. 8.5) с выделением подмножества комплексных параметров.

Расширение функций БСА, ужесточение требований по массе, габаритам и надежности функционирования обусловливают применение микропроцессорных средств, а также специализированных и программируемых БИС и другой надежной элементной базы.

В целом МПСУ БСА должны обеспечивать высокую надежность функционирования (0,97—0,98 за десятки часов работы); практически неограниченный ресурс ($T\leqslant100\,000$ ч); автономность всех видов, типов и режимов обмена сигналами, высокую степень интеграции архитектуры, ее модульность, адаптивность, перестраиваемость и наращиваемость; автоматизированный поиск и локализацию неисправностей; развитое ПО на языках высокого уровня; тестируемость аппаратуры и ПО; высокую степень стандартизации по структуре, связям, сигналам, конструкции, процедурам контроля и ремонта.

Для оценки структурных решений БСА можно использовать следующие комплексные показатели:

Таблица 8.5. **Перечень основных технических параметров оценки качества БСА**

№ поз.	Наименование параметра	Разновидности критериев качества											
		I			II			III		IV		V	
		$X_{тп}$	$X_э$	$X_с$	$X_{тз}$	$X_{нтд}$	$X_{кд}$	$X_п$	$X_и$	$X_з$	$X_у$	$X_г$	$X_в$
1	Число каналов управления или обмена данными	+			+			+		+		+	
2	Пропускная способность каналов	+			+			+		+		+	
3	Вероятность отказов	+			+				+	+			+
4	Вероятность сбоев	+			+				+	+			+
5	Вероятность нормального функционирования	+			+	+		+		+		+	
6	Коэффициент готовности к эксплуатации	+			+	+			+		+		+
7	Характеристики входных сигналов . .	+			+	+		+		+		+	
8	Характеристики выходных сигналов . . .	+			+				+	+		+	
9	Точность передачи информации	+			+			+		+		+	
10	Характеристики системы питания	+			+			+		+			+
11	Число обслуживаемых абонентов	+			+			+		+			+
12	Время готовности к работе после включения сети питания . .	+			+				+	+			+
13	Коэффициенты конструирования и технического качества (технологичности, повторяемости, унификации, стандартизации и др.)	+			+		+		+	+			+

Продолжение табл. 8.5

№ поз.	Наименование параметра	Разновидности критериев качества											
		I			II			III		IV		V	
		$X_{тп}$	$X_э$	$X_с$	$X_{тз}$	$X_{нтд}$	$X_{кд}$	$X_п$	$X_и$	$X_з$	$X_у$	$X_г$	$X_в$
14	Содержание драгоценных металлов . . .	+			+		+		+		+		+
15	Тип элементной базы	+				+			+		+		+
16	Весогабаритные характеристики	+			+		+		+	+			+
17	Тип конструкций . .	+					+		+		+		+
18	Условия эксплуатации (механические, климатические, специальные воздействия)		+		+		+	+		+			+
19	Характеристики технического ресурса . .		+		+			+		+			+
20	Время непрерывной работы		+		+				+	+			+
21	Характеристики микроклимата, термостатирования . . .		+			+	+		+		+		+
22	Ремонтопригодность		+			+	+		+		+		+
23	Стоимость		+		+	+		+		+		+	
24	Экономическая эффективность		+			+			+		+	+	
25	Тип каналов (алгоритм функционирования)			+	+			+		+		+	
26	Тип интерфейсных связей			+	+			+		+		+	
27	Характеристики памяти канала			+		+			+		+		+

№ поз.	Наименование параметра	Разновидности критериев качества											
		I			II			III		IV		V	
		$X_{тп}$	$X_э$	$X_с$	$X_{тз}$	$X_{нтд}$	$X_{кд}$	$X_п$	$X_и$	$X_з$	$X_у$	$X_г$	$X_в$
28	Характеристики контроля			+	+	+		+		+		+	
29	Способ передачи кодов и их тип			+	+	+		+		+		+	
30	Типы и виды обмена			+	+			+		+		+	
31	Типы и характеристики задач			+	+			+		+		+	

Комплексные показатели качества проектирования:

на структурном уровне:

— пропускная способность канала $C_{пр}$, кбод {1, 2, 7, 25, 30}

— частота внешних воздействий $\Lambda_п$, Гц {31}

— множество вероятностей рабочих состояний $P_{сост}$ . . . . . {5, 7, 9, 25, 30}

— сложность структурной реализации $H_{ст}$, бит . . . . . {1, 7, 25, 26, 29, 30}

стоимость разработки структуры $C_{ст}$, тыс. руб. . . . . {23}

на функционально-логическом уровне:

— среднее время обслуживания требований на обмен $T_{об}$ . . . . . {12, 15}

— вероятность отказов нормального функционирования $Q_{нф}$ . . . . . {3, 4, 5, 6, 16, 17, 18, 19, 20, 21, 22, 28}

— сложность логической организации $H_{фл}$ {8, 10, 11, 16, 26, 27, 28, 29}

— стоимость изготовления функциональной логической схемы $C_{фл}$ . . . . . {13, 14, 23, 24}

Примечания: 1. В таблице знаком «+» обозначено вхождение параметра в определенное множество критериев, классифицируемых следующим образом.

I — по виду параметров при проектировании на структурном и функционально-логическом уровнях: $X_{тп}$ — технические параметры; $X_э$ — параметры эксплуатации; $X_с$ — параметры структуры управления;

II — по источникам получения: $X_{тз}$ — техническое задание; $X_{нтд}$ — научно-техническая документация; $X_{кд}$ — конструкторская документация;

III — по степени изменяемости в процессе проектирвания; $X_п$ — постоянные, неизменяемые; $X_и$ — изменяемые;

IV — по степени определяемости при проектировании: $X_з$ — задано до проектирования; $X_у$ — уточняемые в процессе проектирования;

V — по весу значимости на всех этапах и уровнях проектирования: $X_г$ — определяющие, главные параметры; $X_в$ — обеспечивающие (вторичные) требования технического задания.

2. В комплексных показателях качества указаны номера позиций частных параметров.

— *на структурном уровне проектирования систем*

$$Ц_1 = \{C_{пр},\ \Lambda_п,\ P_{сост},\ H_{ст},\ C_{ст}\},$$

где $C_{пр}$ — пропускная способность канала данного типа, кБод; $\Lambda_п$ — частота внешних воздействий, Гц; $P_{сост}$ — множество вероятностей рабочих состояний системы; $H_{ст}$ — сложность структурной реализации, бит; $C_{ст}$ — стоимость разработки БСА по принятым структурным решениям, тыс. руб.;

— *на функционально-логическом уровне проектирования систем*

$$Ц_2 = \{\overline{T}_{об},\ Q_{нф},\ H_{фл},\ C_{фл}\},$$

где $T_{об}$ — среднее время обслуживания требования на обмен; $Q_{нф}$ — вероятность отказов нормального функционирования; $H_{фл}$ — сложность логической организации; $C_{фл}$ — стоимость разработки и изготовления БСА по данной функционально-логической схеме;

— *на структурном уровне проектирования блоков*

$$Ц_{1б}^{(1)} = \{H_{ст.б},\ C_{ст.б}\},$$

где $H_{ст.б}$ — сложность структурной реализации блока; $C_{ст.б}$ — стоимость разработки блока;

— *на функционально-логическом уровне проектирования блоков*

$$Ц_{2б}^{(2)} = \{\theta_б,\ Q_{нф.б},\ H_{фл.б},\ C_{фл.б}\},$$

где $0_б$ — временны́е характеристики конкретного блока; $Q_{нф.б}$ — вероятность отказов нормального функционирования блока; $H_{фл.б}$ — сложность функционально-логической реализации блока; $C_{фл.б}$ — стоимость разработки и изготовления блока в данной функционально-логической реализации.

Заметим, что сложность и стоимость структурной реализации блоков существенно зависят от конфигурации системы, которая, в свою очередь, определяется требуемыми или ожидаемыми временными, надежностными, эксплуатационными и конструктивно-технологическими характеристиками блоков.

На уровне математического (МО) и программного (ПО) обеспечения систем формирование совокупности частных показателей качества проводят с учетом соответствующих требований. Для МО основные требования заключаются в реализации максимальной точности (степени совпадения истинных и расчетных результатов), простоты (количества информации, содержащейся в моделях и алгоритмах проектных операций), алгоритмической надежности (вероятности выдачи МО правильных результатов) и универсальности (возможности применения), для ПО — в минимуме объемов памяти для хранения моделей и вычислительных затрат.

Модели устройства или системы можно оценить по показателям погрешности (адекватности) модели, ее сложности, количеству ин-

формации, вносимой разработчиком в процессе создания модели (в битах). Считается нецелесообразным использовать показатель погрешности в качестве оценки моделей проектируемых систем из-за его неопределенности.

Сложность модели системы (количество информации в битах) характеризуется энтропией получения модели системы M_c по ее качественному описанию K, определяемой как [50]:

$$H(M_c/K)=\sum [H(Я(j)/Я_0,\ K)+H(\beta(j)/Я(j)]=$$

$$=\sum_{j=1}^{n}\left[\log_2\left(\frac{m_{яз}+n(j)-1}{n(j)}\right)+0{,}5\alpha(j)[\alpha(j)-1]-\right.$$

$$\left.-\log_2\left[\prod_{kk=1}^{p}(j,\ kk)!\right]\right],$$

где n — общее число компонент в модели системы; $Я(j)$ — совокупность элементов языка, используемого при построении модели компонент j-го типа; $\beta(j)$ — совокупность связей между элементами языка в множестве $Я(j)$ в модели компонент j-го типа; $Я_0$ — общая совокупность элементов языка, используемого при построении моделей компонентов всех типов; $m_{яз}$ — число типов элементов языка в $Я_0$; $n(j)$ — число элементов языка в $Я(j)$; $\alpha(j)$ — общее число «выводов» элементов языка в $Я(j)$ и $B_я(j)$; $B_я(j)$ — совокупность типов компонентов, из которых состоит разрабатываемое устройство или система на данном уровне проектирования; kk — номер подпоследовательности идентичных элементов языка в $Я(j)$; $n(j,\ kk)$ — число элементов языка в kk-й подпоследовательности в $Я(j)$; p — число подпоследовательностей идентичных элементов языка в $Я(j)$.

Объем вычислительных затрат T_c определяется затратами машинного времени работы процессора на один прогон модели системы на ЭВМ. Они зависят от принципа моделирования (событийного или синхронного), сложности используемых алгоритмов и др. Например, при событийном моделировании и однопроцессорной ЭВМ

$$T_{с.ож\,j}=\sum_{i=1}^{n}T_{маш\,i},$$

где n — ожидаемое число активизируемых событий; $T_{с.\,ож\,j}$ — ожидаемые затраты машинного времени в j-й цикл просчета модели; $T_{маш\,i}$ — затраты времени на счет модели i-го компонента.

Объем памяти V_c модели системы определяется объемом памяти, занимаемой основной программой $V_{оп}$, реализующей заданный алгоритм функционирования системы; объемом памяти, отводимым

под модели компонентов V_i^k, принципом распределения памяти в процессе вычислений:

$$V_c = \begin{cases} V_{оп} + \max\{V_i^k\} & \text{при многопроцессорной структуре;} \\ V_{оп} + \sum_{i=1}^{m} V_i^k & \text{при однопроцессорной структуре,} \end{cases}$$

где m — число компонентов в модели системы.

§ 8.6. РАЗРАБОТКА СИСТЕМЫ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ ДЛЯ ТОКАРНОГО ГПМ

Как указывалось ранее, при разработке системы управления конкретным объектом ГАП (станком с ЧПУ, промышленным роботом, обрабатывающим центром, гибким модулем и т. д.) в зависимости от используемой элементной базы проектировщику приходится выполнять различный объем работ, оптимизировать проектные решения по разнообразным критериям качества. При аппаратной реализации алгоритмов управления в качестве критериев качества обычно выступают объем схемных затрат (число микросхем, СИС, БИС и др.), стоимость, надежность, помехоустойчивость; при программно-аппаратной реализации — надежность комплекса «технические средства — программы», стоимость, объем оборудования; при чисто программной реализации — минимум модификации аппаратных средств используемых комплексов, объема требуемой памяти, длины программ, максимум их надежности и отказоустойчивости и др.

Ниже на примере разработки СЛУ для токарно-винторезного станка с ЧПУ и токарного ГПМ, создаваемого на его базе, проиллюстрируем особенности реализации алгоритмов управления, характерных для компонентов станочного комплекса, при использовании указанных в гл. 2 элементов (мультиплексоров, ПЗУ, ПЛМ, БИС УП) и серийных КТС (микроЭВМ, СЧПУ, МПК).

Управляющий комплекс токарно-винторезного станка с ЧПУ модели ФТ-23 включает в себя серийную СЧПУ «Луч-2Т», систему логического управления электроавтоматикой, а также пульты (рабочий, вспомогательный и приводов подач). Структура системы управления станка приведена на рис. 8.5.

Данный комплекс осуществляет: одновременное продольное перемещение каретки и поперечное перемещение суппорта с дискретностью 0,005 мм при рабочих перемещениях и 0,02 мм при ускоренных перемещениях; согласование скорости перемещения суппорта и каретки при обработке конических поверхностей с криволинейной образующей; автоматическое задание скорости вращения шпинделя; автоматическую смену инструмента; автоматическое включение и отключение вспомогательных механизмов (гидростанции, пиноли) и др.

Основой СЧПУ является СЛУ электроавтоматикой. Общими функциями СЛУ являются: прием команд от СЧПУ, пультов управления, сигналов от датчиков станка; выработка на основе полученной информации соответствующих сигналов управления силовыми элементами, обеспечивающими работу исполнительных механизмов станка: двигателя главного движения (ДГД), приводов подач, переборной группы шпинделя, двигателей шпинделя, резцедержателя, системы охлаждающей жидкости (СОЖ), гидростанции и пиноли; электромагнитных муфт автоматической коробки скоростей (АКС), электромагнитов гидростанции и пиноли; формирование ответных сигналов в СЧПУ об исполнении управляющих и технологических команд; контроль состояния электрооборудования, сигнализация о неисправностях и автоматическое отключение приводов станка при аварийных и некоторых других ситуациях.

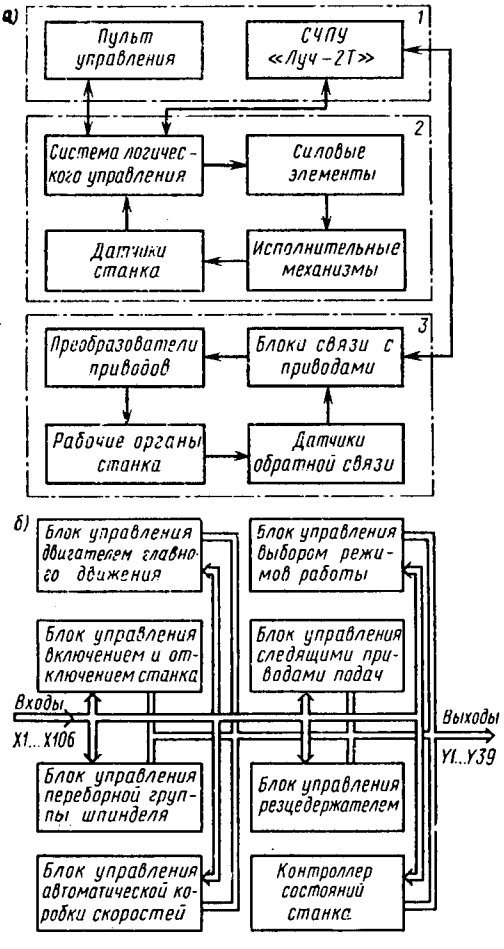


Рис. 8.5. Структура системы управления для станка с ЧПУ:

а — обобщенная структура системы управления; *б* — структура системы логического управления

Задачи, решаемые СЛУ, разделяются на задачи ручного и автоматического режима. При реализации ручного режима обеспечивается возможность выполнения операций в любой требуемой оператору последовательности. При этом бóльшая часть блокировок, необходимых для безаварийной работы, исключается, а безаварийность обеспечивается оператором. В автоматическом режиме используется максимум блокировок, исключающих возникновение аварийных ситуаций и обеспечивающих получение годных деталей.

В ручном режиме реализуется две группы команд: толчковые, выполняемые оператором в течение требуемого интервала времени;

пусковые (стóповые), выдаваемые в виде единичных импульсов. Прекращение операции, включенной пусковой командой, осуществляется или выдачей другой команды, отменяющей первую, или приходом соответствующего сигнала от датчиков станка, или по истечении заданного интервала времени. Операции, выполняемые под влиянием толчковой или пусковой команды, прерываются автоматически при появлении сигналов от действующих блокировочных или ограничивающих датчиков.

В общем виде СЛУ токарно-винторезным станком может быть представлена асинхронным автоматом размерностью порядка 100 входов и 40 выходов при параллельно-последовательном алгоритме функционирования.

Входные сигналы СЛУ станка подразделяются на командные входы однократного действия (опрашиваемые входы состояний станка, инициативные входы состояний станка, инициативные командные входы), которые формируются органами пультов управления, СЧПУ и датчиков станка, определяющих положение рабочих и исполнительных механизмов в ходе выполнения команд. Эти входы могут опрашиваться выборочно. Последние два типа сигналов, формируемых блокировочными и аварийными датчиками, имеют приоритет перед остальными и исполняются сразу же после появления. В ответ на входные сигналы временнóго, потенциального и импульсного типа СЛУ вырабатывает распределенную во времени последовательность управляющих сигналов (команд), согласует работу всех механизмов станка, обеспечивает обмен информацией с СЧПУ, контроль и сигнализацию состояний рабочих органов и исполнительных механизмов и защиту их в аварийных ситуациях.

Алгоритм функционирования СЛУ может быть описан двенадцатью микропрограммами (рис. 8.6), каждая из которых характеризуется относительно малым числом внутренних состояний (2—8) с длинными и разветвленными выражениями логических условий переходов из одного состояния в другое внутреннее состояние. К ним относятся микропрограммы включения и отключения станка (ВОС), управления двигателем главного движения (ДГД), выбором режима работы (ВРР), разрешением работы и контурным управлением следящими приводами подач (СПП), управления резцедержателем (РД), переборной группой шпинделя (ПГШ) и диапазоном перебора, а также АКС, пинолью и др., отображенные граф-схемами алгоритма ГСА1 — ГСА13.

Управление станком начинается с включения вводного автомата F_{11}, подающего напряжение питания на СЛУ станком с установкой начального состояния системы. Если микропрограмма ВОС (ГСА1) находится в рабочем состоянии ($a_{13}=1$), то происходит переход к микропрограмме ДГД (ГСА2), иначе осуществляется переход в аварийное состояние, обнуление памяти и прекращение работы станка. При нормальной работе ДГД станка выполняются

микропрограммы ВРР (ГСА4), контурного управления СПП (ГСА5) и управления РД (ГСА9). Если заданная позиция резцедержателя совпадает с установившейся, выполняются микропро-

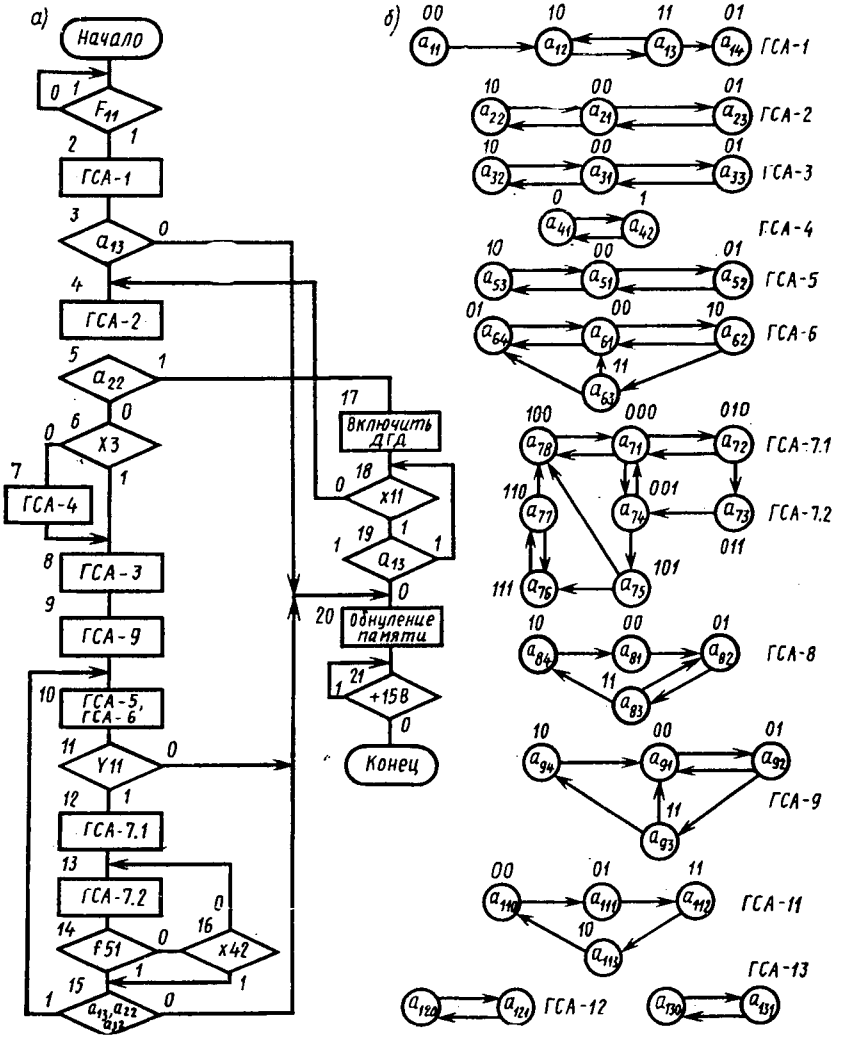
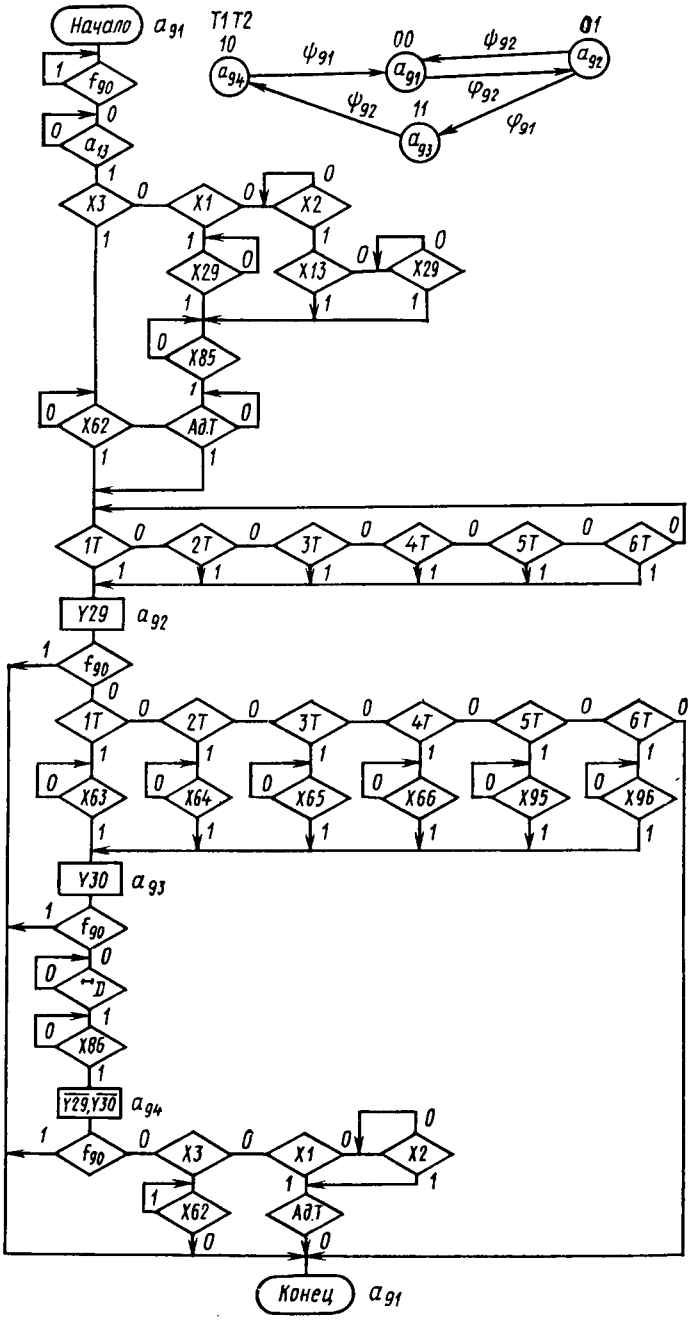


Рис. 8.6. Алгоритм функционирования станка с ЧПУ:
а — общий укрупненный алгоритм; *б* — структурные графы переходов для граф-схем алгоритмов ГСА (1—9; 11—13)

граммы управления ПГШ и диапазона перебора (ГСА5,6). При совпадении истинного и заданного диапазонов выдается команда о прямом или обратном вращении гидродвигателей перебора соот-



Рис, 8.7. Микропрограмма ГСА-9 управления резцедержателем станка

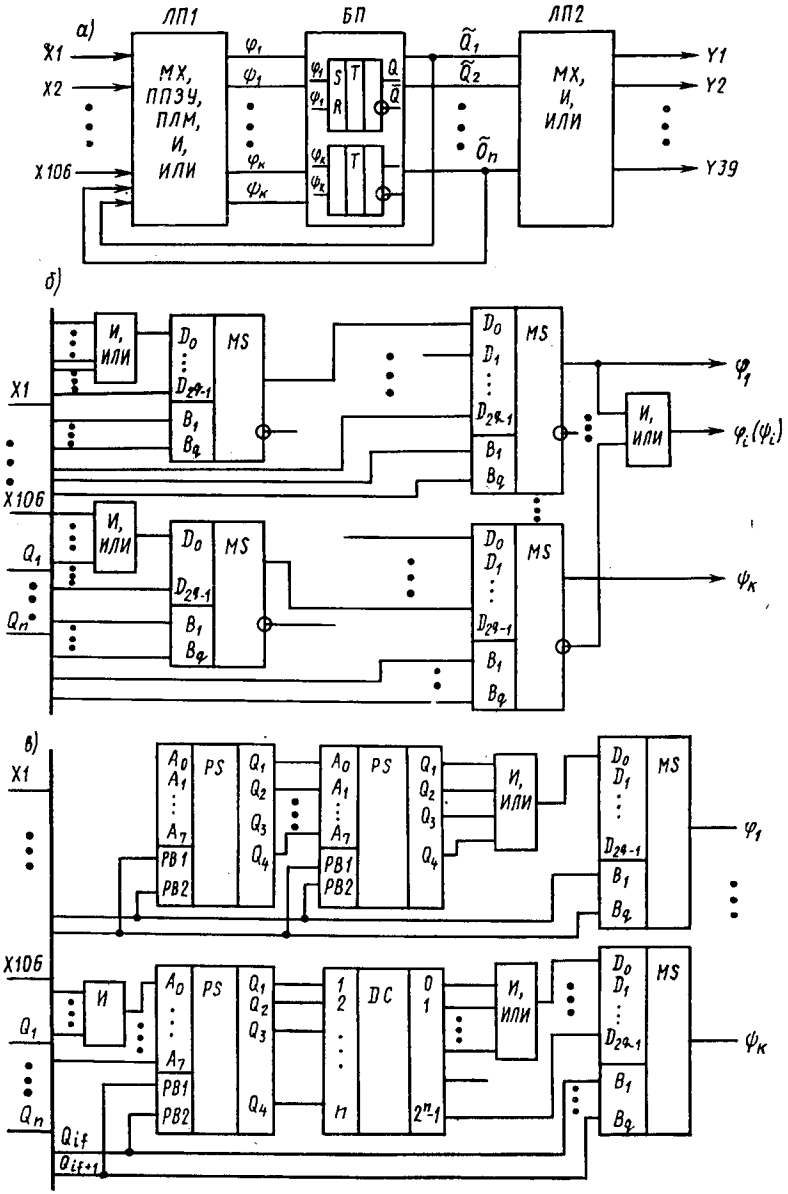


Рис. 8.8. Структурные реализации СЛУ станка с ЧПУ на основе микросхем средней и высокой степени интеграции:

а — модель конечного автомата Мура; *б* — реализация логического преобразователя ЛП1 на мультиплексорах; *в* — реализация ЛП1 на ППЗУ, дешифраторах и мультиплексорах

ветственно. Далее формируется команда «Толчок», после чего выполнение предшествующих микропрограмм прерывается и исполняется микропрограмма управления АКС в автоматическом (ГСА7.1) или толчковом (ГСА7.2) режимах работы. При нормальном сцеплении электромагнитных муфт в АКС производится контроль рабочих ситуаций станка, ДГД и СПП, и при их нарушении происходит

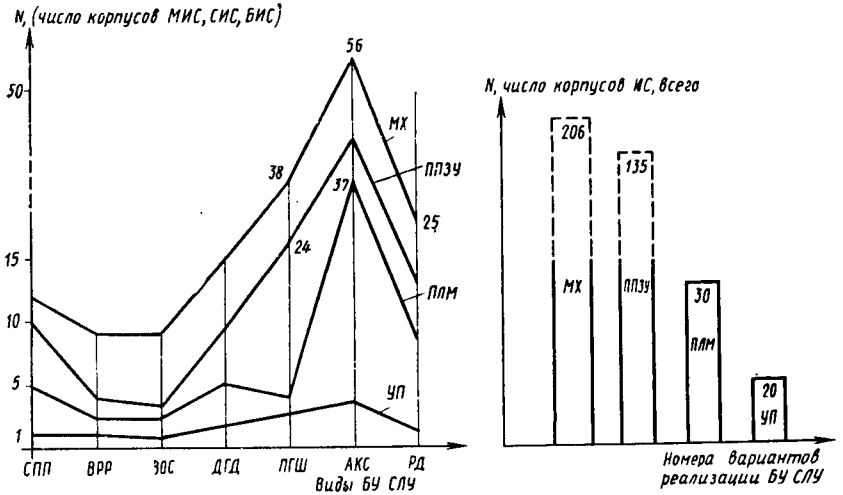


Рис. 8.9. Сравнительные оценки аппаратных затрат (в корпусах микросхем) на реализацию СЛУ токарно-винторезного станка с ЧПУ:

а — по отдельным блокам СЛУ; *б* — суммарные затраты при использовании однотипных элементов (МХ — мультиплексоров; ППЗУ — БИС ППЗУ; ПЛМ — БИС ПЛМ; УП — БИС УП)

отключение напряжения питания. Последующее возобновление работы станка возможно только после повторного включения вводного автомата F_{11}. В качестве примера на рис. 8.7 приведена подробная граф-схема алгоритма управления РД.

Реализация СЛУ базируется на модели асинхронного конечного автомата Мура (рис. 8.8, *а*), включающего блоки логического преобразователя (по входной ЛП1 и выходной ЛП2 логике) и памяти, реализованной на RS-триггерах. Основную долю затрат автомата составляет реализация ЛП1 и ЛП2 из-за достаточно сложных условий переходов при различной связности входных, промежуточных и выходных переменных временно́го, потенциального и импульсного типа, что затрудняет применение изложенных в гл. 3 методов синтеза комбинационных схем на основе одного какого-либо типа СИС и БИС и обусловливает подбор композиции элементов нескольких типов для наиболее эффективной реализации. Получен ряд вариантов аппаратной реализации логического преобразователя ЛП1 (рис. 8.8, *б*, *в*) на основе разнотипных мультиплексоров и дешифраторов серий 155, 564 и 511, БИС ППЗУ серии

566, БИС ПЛМ серии 840, БИС УП серии 587, а также их композиции. Сравнительные оценки сложности реализации блоков управления СЛУ станком (рис. 8.9, *а*, *б*) показывают, что наиболее эффективной по надежности, быстродействию и аппаратным затратам является структура с использованием БИС УП К587РП1 и триггеров К564ТМ2.

При микропроцессорной реализации СЛУ токарно-винторезного станка основную сложность представляют вопросы синтеза и оптимизации программ реализации систем логических функций. Исследования показали, что использование МП К589 по сравнению с МП К580 позволяет создать более быстродействующую и экономичную реализацию СЛУ. МП К580 целесообразно использовать при программной реализации алгоритмов, характеризующихся большим числом вычислительных операций, малым удельным весом логических функций и невысоким быстродействием.

Использование микропроцессорных контроллеров (МПК) и СЧПУ различных типов для программно-аппаратной реализации алгоритмов функционирования станков с ЧПУ требует детального учета специфических особенностей построения структуры технических средств, организации системного и базового программного обеспечения, принятых языков проектирования и программирования, что позволяет эффективно реализовать совокупность отдельных управляющих и вычислительных процедур и разнообразных алгоритмов управления.

Для реализации СЛУ используются отечественная и зарубежная СЧПУ типа 2Р32 и CNC 600-3, имеющие свои отличительные особенности в структуре, организации ПО, языках программирования.

В состав СЧПУ 2Р32 входят МЭВМ «Электроника 60М» и совокупность блоков, обеспечивающих связь со станком и устройствами ввода — вывода (УВВ). Прием и выдача дискретной информации от подсистемы ЭА станка, пульта станочника-оператора (ПСО) и УВВ, а также связь с ЭВМ верхнего ранга производятся через специализированное периферийное устройство связи — блок технологических команд (БТК).

МЭВМ «Электроника 60М» производит алгоритмическую обработку входной информации, поступающей через входные модули и модули интерфейса. Связь МЭВМ с БТК осуществляется путем адресного обращения из базовой системной технологической программы (БСТП) к модулям связи по абсолютным адресам в системе команд «Электроника 60М». Обращение МЭВМ к заданному модулю связи производится по конкретному адресу, которому соответствует своя группа входов и выходов, имитируемых числом. Сигналы, поступающие от станка, внутри БТК не запоминаются, и обращение к ним производится только в режиме считывания информации. Для сигналов «СЧПУ — станок» в модулях БТК предусмотрены регистры, обеспечивающие хранение информации до следующего цикла записи, что при обращении по тому же адресу позволяет производить многократное считывание информации.

Функциональное ПО СЧПУ 2Р32 состоит из БСТП, кодируемой в абсолютной двоичной системе в соответствии с системой команд МЭВМ «Электроника 60М», и программ привязки (ПрПр), обеспечивающих привязку БСТП с конкретной модели станков. ПрПр включают в себя:

список станочных (СК) и технологических констант (ТК), задающих числовые значения параметров станка (разгона-торможения, люфтов и др.), и установку признаков обработки технологических *M*-, *S*-, *T*-функций;

программы обработки дискретных сигналов станка, представляющие собой последовательности подпрограмм (ПП), реализующих одну из программных *M*-, *S*-, *T*-функций; взаимодействие с ПСО; обработку блокировочных и аварийных ситуаций.

Запись алгоритмов управления станками в СЧПУ 2Р32 осуществляется на проблемно-ориентированном языке функциональных программ (ЯФП), интерпретируемом БСТП с помощью ПрПр в соответствующую последовательность операций МЭВМ. С помощью ЯФП производится запись операций над данными, поступающими от станочного оборудования, содержимым зоны ячеек памяти ОЗУ, доступных ПрПр и БСТП при выполнении различных функционально-ориентированных алгоритмов управления станком ОЦ. ЯФП обеспечивает правильность записи и выполнение арифметических операций, сдвигов, пересылок, установок, условных и безусловных переходов, перевода двоичных кодов в десятичные числа и обратно, получение прямого кода отрицательного цикла, обнуление массивов, задание временных циклов обращения к таймеру, выполнение пересылочных операций по косвенной адресации и др.

Таблица 8.6. **Сравнительные оценки эффективности прикладных программ СЛУ токарно-винторезного станка**

Наименование подпрограммы	Критерии реализации СЛУ на СЧПУ		
	расчетное быстродействие, мс	объем памяти, бит	число шагов программ
Подключение станка	13,3/44	16/20	19/134
Ручные перемещения	45/37	17/6	25/68
Отключение станка	6,5/14,5	50/6	9/29
Управление шпинделем	221/139	203/119	30/78
Управление резцедержателем	706/340	52/45	133/198

Примечание. Количественные данные в числителе относятся к СЧПУ CNC 600-3, в знаменателе — к СЧПУ 2Р32.

СЧПУ CNC 600-3 является мультимикропроцессорной системой из девяти микропроцессоров и двух микроЭВМ, проблемно-ориентированных на решение специализированных задач. Для описания алгоритмов управления используется язык булевых функций (ЯБФ).

Программные реализации СЛУ токарно-винторезного станка, выполненные на СЧПУ 2Р32 и CNC 600-3 с языками ЯФП и ЯБФ соответственно, сравнивались по критериям расчетного быстродействия, требуемого объема памяти и объема прикладного ПО (табл. 8.6).

Расширение функциональных возможностей токарно-винторезного станка при встраивании его в ГПС требует оснащения его ПР загрузки-выгрузки грузов. Эта задача рассматривается на примере создания токарного ГПМ на базе ФТ-23 для гибкого автоматизированного участка (ГАУ) механообработки одного из опытных производств, компоновка которого приведена на рис. 8.10.

В состав ГАУ входят: ОЦ ИР500МФ4, оснащенный СЧПУ «Фанук-6В», три токарно-винторезных станка модели ФТ, оснащенных СЧПУ типа «Луч-2Т», автоматизированный склад типа СТАС. Компоновка ГАУ представляет собой однорядное линейное расположение оборудования (рис. 8.10. *а*), в котором кран-штабелер автоматизированного склада (АС) выполняет функции транспортной системы в ГАУ, доставляя заготовки, тару и полуфабрикаты в требуемые места хранения, переустановки или запросов со стороны станков. Заготовки деталей типа тел вращения хранятся в ячейках АС в кассетах, в каждой из которых

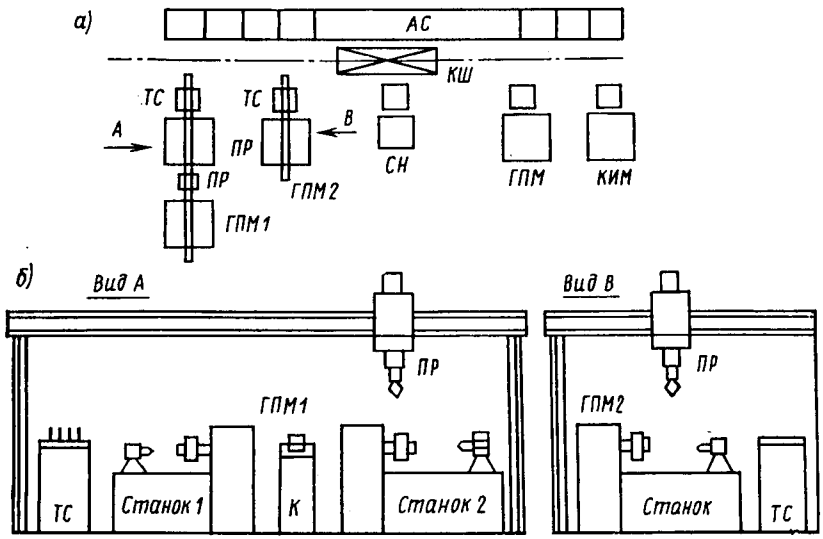


Рис. 8.10. Компоновка ГАУ опытного производства:

а — компоновка технологического оборудования; *б* — компоновка токарных ГПМ1, НПМ2; АС — автоматизированный склад; ТС — тактовый стол; КШ — кран-штабелер; ГПМ — гибкий производственный модуль; ПР — промышленный робот; К — кантователь; КИМ — контрольно-измерительная машина; СН — стол наладчика

число заготовок равно размеру отдельной партии. Инструмент также хранится в ячейках АС в кассетах, в каждой из которых содержится полный набор инструментов, необходимых для обработки конкретной партии. Заготовки корпусных деталей хранятся в АС, откуда они поступают на стол наладчика с целью установки их на палеты для последующей обработки на ОЦ. Установка инструмента в инструментальные магазины станков производится вручную. Предусматривается контроль обработанных деталей с помощью контрольно-измерительной машины, после чего детали возвращаются на склад. Обслуживание токарных станков предусматривается с помощью промышленных роботов (ПР), доставляющих заготовки с тактовых столов (ТС), устанавливаемых рядом с оборудованием, к станку и обратно. Для обработки деталей типа «втулка» и «фланец» дополнительно устанавливается кантователь с целью возможности обработки деталей с обеих сторон. Примерная компоновка токарных ГПМ приведена на рис. 8.10, *б*, где два станка обслуживаются одним портальным ПР и выполняют обработку втулок и фланцев, а третий станок обрабатывает валы.

Исходя из специфики используемого оборудования и конструктивно-технологических особенностей обрабатываемых деталей, для обслуживания ГПМ, обрабатывающего втулки и фланцы, выбран

ПР модели М20Ц05.11, а для ГПМ, обрабатывающего валы,— ПР модели СМ80Ц25.01А. Для АС исходным положением крана-штабелера (КШ) считается его размещение у пункта ввода-вывода грузов (стола наладчика), при этом каретка КШ находится на уровне стола, захват КШ — в среднем положении и без груза.

Алгоритм функционирования токарного ГПМ механообработки деталей вращения типа «втулка» начинается с проверки исправности датчиков контроля оборудования, входящего в состав ГПМ. При выполнении данного условия СУ ГПМ выдает заявку в СУ АС на загрузку ГПМ партией заготовок. После подтверждения приема заявки со стороны АС происходит ожидание выполнения операции транспортировки кассеты с заготовками на ТС ГПМ. Далее кассета с заготовками попадает с помощью ТС в зону захвата руки ПР, из нее берется первая заготовка и осуществляется транспортировка заготовки и установка ее в патрон станка. После этого ПР выводится из зоны резания станка и он перемещается вновь к кассете для извлечения второй заготовки для второго станка и загрузки ее в патрон для механической обработки. В это время на первом станке зона резания закрывается шторкой, начинается процесс механической обработки по заданной программе. То же самое происходит со вторым станком, а ПР перемещается в позицию ожидания окончания обработки заготовки после первого установа. Как только на одном из станков обработка заканчивается, ПР берет обработанную заготовку из патрона, транспортирует ее к кантователю, устанавливает в него, после чего кантователь поворачивается определенным образом с целью возможности обработки детали с другой стороны. Далее процесс транспортировки к станку повторяется для обработки оставшейся части заготовки, после чего готовая деталь захватывается ПР и транспортируется к кассете, устанавливаясь в ту же ячейку, откуда она была извлечена ранее. Затем из кассеты извлекается очередная заготовка и процессы транспортировки, установки, манипуляций и механической обработки повторяются по циклу вплоть до окончания обработки всей партии заготовок, после чего СУ ГПМ посылает запрос в СУ АС на освобождение ТС от кассеты с готовыми деталями и доставки ее далее на хранение в АС или на контроль в КИМ.

Контроль перемещений и манипуляций захватывающим устройством (ЗУ) ПР производится с помощью дискретных датчиков, выдающих в момент достижения заданной точки позиционирования цифровой сигнал. Надежного функционирования ПР и ГПМ в комплексе при этом можно достигнуть благодаря применению более надежных элементов контроля, их резервирования или создания системы автоматического контроля и устранения отказов (САКУО). Однако для уменьшения сложности реализации САКУО необходимо нахождение оптимального числа точек контроля, так как малое их количество не гарантирует высокой работоспособности ГПМ, а большое их число усложняет реализацию САКУО.

Анализ показал, что для токарного ГПМ характерны следующие виды отказов: S_1 — перенос заготовки в кассете; S_2 — несовпадение положения ЗУ ПР с положением заготовки; S_3 — захват заготовки при загрузке, S_4 — неточное положение заготовки в патроне станка; S_5 — отказ патрона при зажиме; S_6 — неточное позиционирование инструмента; S_7 — размерный износ инструмента; S_8 —

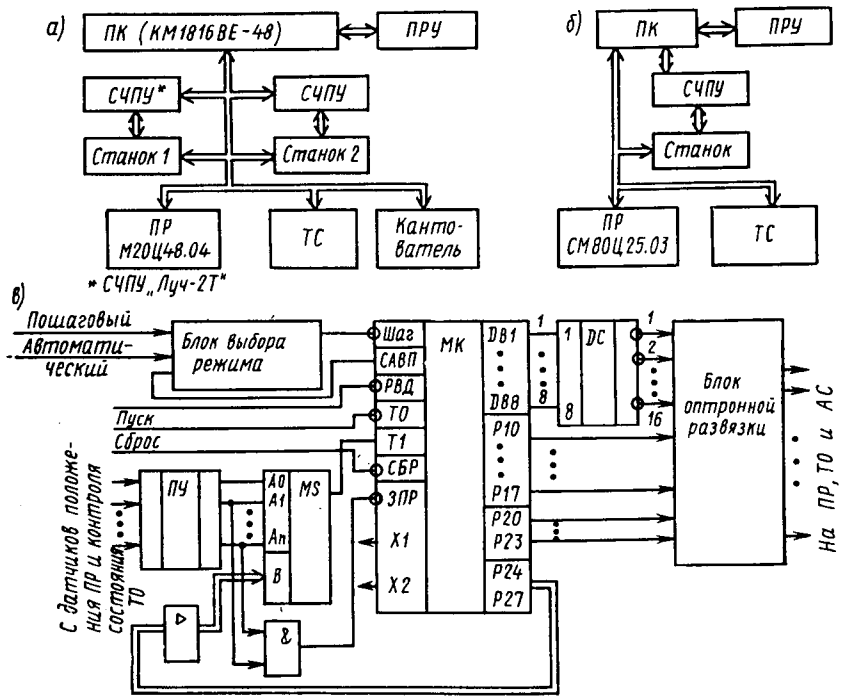


Рис. 8.11. Структура системы управления ГПМ:
а — для ГПМ из двух станков и одного промышленного робота; *б* — для одностаночного ГПМ; *в* — функциональная схема программируемого контроллера ГПМ

поломка инструмента; S_9 — отказ патрона при съеме детали; S_{10} — отказ ЗУ ПР при съеме. Данные виды отказов можно устранить в процессе активного контроля следующих признаков; f_1 — проверка положения заготовки в кассете; f_2 — наличие детали в захвате ПР; f_3 — точность осевого положения заготовки в патроне; f_4 — точность позиционирования инструмента; f_5 — мощность резания; f_6 — обработанный размер; f_7 — наличие детали при съеме; f_8 — наличие детали в кассете.

Расчет вероятности работоспособного состояния ГПМ с использованием метода определения оптимального числа контролируемых признаков показывает, что наибольшего значения надежности

САКУО и СУ ГПМ в целом можно достигнуть при реализации признаков контроля осевого положения заготовки в патроне, обрабатываемого размера и наличия детали в ЗУ ПР при съеме.

Программная реализация алгоритма функционирования токарного ГПМ выполнена на языке Ассемблер однокристального микроконтроллера КМ1816, являющегося ПК для имеющихся СЧПУ «Луч-2Т», обеспечивающих управление станками токарной группы.

Таблица 8.7. **Сравнительные оценки сложности программной реализации алгоритмов функционирования токарного ГПМ на языке Ассемблер КМ1816ВЕ48**

Наименование типовых процессов	Разновидности и число команд программы					
	суммарное	число меток	операции ввода — вывода	передача управления	таймерные операции	установка режимов
Общее функционирование ГПМ «втулка»	62	13	21	39	2	—
Управление загрузкой ГПМ	36	4	21	14	1	—
Взятие заготовки из кассеты	42	5	22	17	1	—
Загрузка станка № 1	39	6	19	13	2	4
Разгрузка станка № 1	30	4	10	15	1	4
Кантование заготовки	52	5	29	18	—	5
Установка детали в свободную ячейку кассеты	20	1	9	9	—	1
Определение координаты, откуда была взята заготовка	24	4	5	15	—	
ГПМ «вал»	122	9	45	66	—	11

В алгоритме ГПМ выделены следующие типовые процессы, реализованные в общей программе в виде отдельных подпрограмм: загрузка ГПМ; взятие заготовок деталей из кассеты; разгрузка станка; загрузка станка; перемещение ЗУ ПР в точку с координатой X; определение координаты ячейки кассеты, откуда была взята заготовка; определение ячейки со следующей заготовкой; кантование заготовки; установка обработанной детали в свободную ячейку кассеты; проверка наличия заготовки в ЗУ ПР и в патроне станка.

Структура комплекса технических средств СУ ГПМ, входящих в ее состав, приведена на рис. 8.11. В целом СУ ПР для ГПМ можно описать конечным автоматом размерностью 62 входа и 26 выходов. В табл. 8.7 приведены сравнительные оценки сложности программной реализации типовых процессов ГПМ.

ГПМ механической обработки деталей вращения типа вала имеют более простой алгоритм функционирования из-за отсутствия в их составе кантователя, так как эту функцию в ГПМ выполняет сам ПР. Остальные типовые процессы здесь аналогичны вышеописанным.

Дальнейший рост автоматизации ГАУ опытного производства может быть связан с решением задач автоматической смены инструментов в ГПМ, оперативно-календарного планирования и учета хода производства, САПР технологических процессов, АСТПП и др., что потребует объединения имеющихся и дополнительных управляющих и вычислительных средств в единую интерсеть предприятия на базе локальной сети, проработки вопросов сопряжения интерфейсов и протоколов данных используемых СЧПУ и других средств вычислительной и микропроцессорной техники.

§ 8.7. РАЗРАБОТКА СИСТЕМЫ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ ВЫСОКОТОЧНЫМ ЭЛЕКТРИЧЕСКИМ СКАНИРУЮЩИМ УСТРОЙСТВОМ

При исследовании природных ресурсов с использованием новейших научно-технических средств важное значение приобретают космические методы исследования поверхности Земли, отличающиеся глобальностью обзора земной поверхности, возможностью его проведения в короткие промежутки времени, оперативностью доставки информации о любых, в том числе и труднодоступных, районах земного шара. Сегодня высока эффективность использования космической информации в геологии при поисках нефти, газа и других полезных ископаемых, при решении широкого круга сельскохозяйственных, гидромелиоративных, лесоустроительных и других научных и практических задач сохранения и рационального использования природных ресурсов Земли.

Исследования Земли из космоса строятся на основе регистрации радиационных потоков, излучаемых наземными образованиями. Наиболее перспективен метод многоспектральной съемки, производимой с помощью специальных сканирующих устройств, устанавливаемых на спутниках, которые производят последовательную фотосъемку узких участков Земли на широком поле обзора. Это позволяет выделить маломерный объект на фоне внешних излучений, улучшить качество изображения, уменьшить размеры приемника излучений.

Сканирующее устройство (сканер) располагается перед объективом оптической системы. В зависимости от траектории сканирования оно может быть выполнено самым различным образом.

Из всего многообразия сканирующих устройств на подвижных объектах наиболее удобно применять качающееся в одной плоскости зеркало, производящее построчную траекторию сканирования и управляемое с помощью магнитоэлектрического привода.

Таким образом, при съемке поверхности Земли из космоса возникает задача управления магнитоэлектрическим приводом, обеспечивающего с высокой точностью движение сканирующего зеркала.

Технические требования к СЛУ сканирующим устройством. СЛУ сканирующим устройством входит в состав бортовой сканирующей системы и обеспечивает формирование сигналов управления изменяющейся длительности. В СЛУ поступают парные сигналы «Начало» (Н) и «Конец» (К). СЛУ должна работать по заднему фронту второго сигнала Н и по переднему фронту первого сигнала К. Уровень лог. 1 этих сигналов — $+8 \div 9$ В, уровень лог. 0 — $0 \div +3$ В, длительность сигналов — $0{,}001 \div 0{,}003$ с, ток лог. 1 — не более 0,02 мА. По переднему фронту сигнала К формируется первый временной интервал длительностью от 0,008 с до 0,014 с и стабильностью не хуже $\pm 0{,}04$ %. После окончания этого интервала формируется второй временной интервал длительностью 0,06 с $\pm 0{,}04$ %, а по его окончании — третий временной интервал длительностью от 0,003 до 0,006 мс $\pm 0{,}04$ %. По сигналу К формируется эталонный временной интервал длительностью 0,025 с и стабильностью не хуже $\pm 0{,}05$ %, по сигналу Н — формирование эталонного временного интервала длительностью 0,0512 с и стабильностью не хуже $\pm 0{,}01$ %. Если интервал «Конец» — «Начало» меньше 0,0256 с, то длительность первого временного интервала должна увеличиваться, если больше — уменьшаться, при равенстве длительность первого временного интервала не должна изменяться. Если интервал «Начало» — «Конец» больше 0,0512 с, то длительность третьего временного интервала должна увеличиваться, если меньше — уменьшаться, при равенстве длительность третьего временного интервала не должна изменяться. Зона нечувствительности составляет 10—20 мкс. Энергопотребление СЛУ $\leqslant 3$ Вт, среднее время наработки на отказ 5000 ч, вероятность безотказной работы $P \geqslant 0{,}95$.

Описание сканирующего устройства и процесса сканирования. Сканирующее устройство (СУ) включает в себя зеркало и его привод и служит для отклонения светового потока, отраженного от исследуемой поверхности, по определенному закону, осуществляя механическую развертку светового луча (рис. 8.12).

Зеркало, укрепленное на оси вращения и совершающее возвратно-вращательное движение в пределах угла $\alpha = 5°$, приводится в движение с помощью двух магнитных систем и двух укрепленных на зеркале катушек, перемещающихся в рабочих зазорах магнитных систем.

При включении питания электрический ток, протекающий в катушках, взаимодействует с магнитным полем магнитных систем, заставляя катушки, а вместе с ними и зеркало, отклоняться в ту или иную сторону вокруг оси, проходящей через опоры. При этом скорость вращения и угол отклонения зеркала зависят от подво-

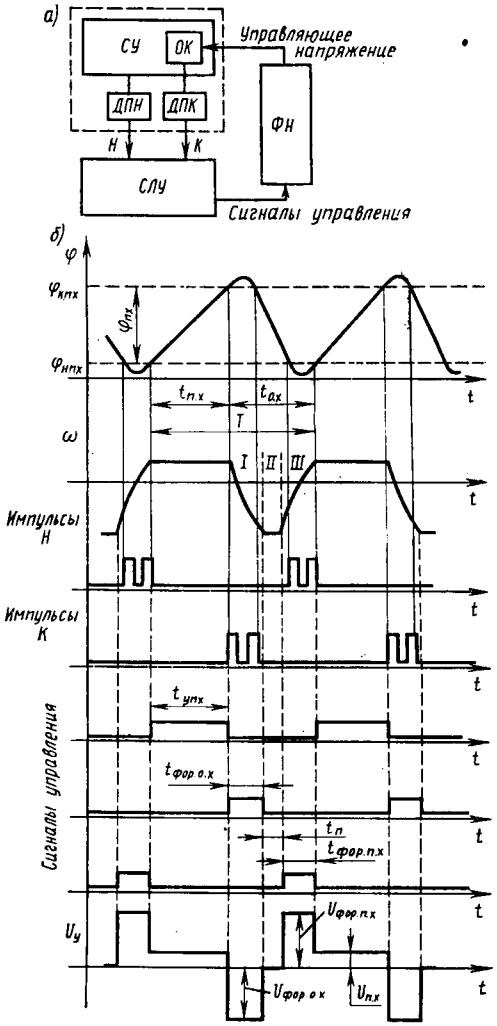


Рис. 8.12. Система управления сканирующим устройством:

а — структура системы управления; *б* — графики движения зеркала сканера и характера сигналов управления: СУ — сканирующее устройство; ОК — обмотки катушек магнитоэлектрического привода; ФН — формирователь напряжения; *ДПН*, *ДПК* — датчики положения начала прямого хода и конца прямого хода; СЛУ — система логического управления; Н, К — сигналы начала и конца прямого хода; $t_{п.х}$ — длительность рабочего участка; $t_{о.х}$ — длительность обратного хода; T — период сканирования; φ — угол поворота зеркала вокруг своей оси; $\varphi_{н.п.х}$, $\varphi_{к.п.х}$ — углы, под которыми расположены датчики ДПН и ДПК; ω — угловая скорость движения зеркала

димого к катушкам напряжения и временно́го действия. Изменяя полярность напряжения, подаваемого в обмотки катушек, можно произвести изменение направления движения зеркала.

Скорость движения зеркала изменяется по пилообразному закону, при этом соотношение времени прямого и обратного хода составляет 2 : 1, частота качания зеркала равна 12—15 Гц. К закону движения зеркала предъявляются чрезвычайно высокие требования по равномерности прямого хода и стабильности временного интервала «начало прямого хода (НПХ) — конец прямого хода (КПХ)». Так, например, последний параметр должен быть не хуже 0,01%, что при указанной частоте и угле сканирования представляет собой погрешности в движении зеркала по времени ± 2 мкс, а по углу $\pm 1{,}5'$. Геометрические положения начала и конца прямого хода зеркала устанавливаются с помощью датчиков положения начала прямого хода и конца прямого хода (ДПН и ДПК), представляющих собой оптронные пары.

На рабочем участке прямого хода зеркала фотографируется поверхность Земли, на участке обратного хода оптическая система закрывается. Для обеспечения качественной фотосъемки необходимо, чтобы зеркало

на рабочем участке прямого хода двигалось равномерно, т. е. угловая скорость была постоянной ($\omega_{п.х}$=const), что и является целью управления. Управление движением СУ ведется с помощью СЛУ и формирователя напряжения (ФН) согласно рис. 8.12, *а*.

График движения зеркала сигналов управления и управляющего приводом напряжения представлен на рис. 8.12, *б*.

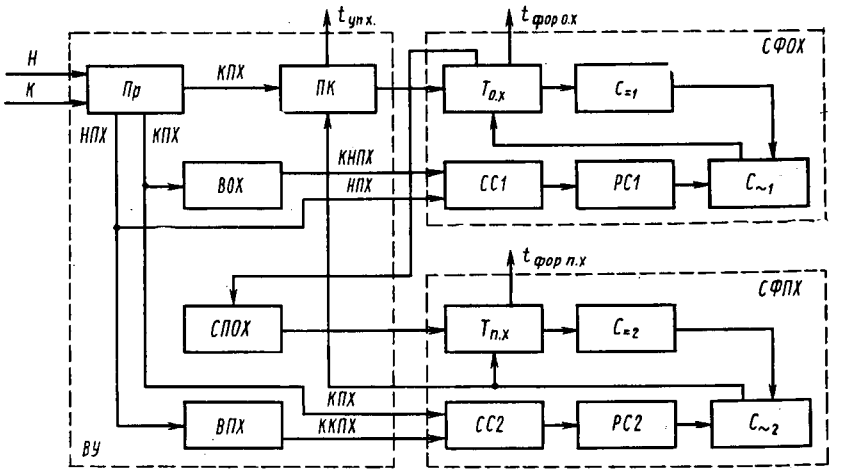


Рис. 8.13. Структурная схема СЛУ сканирующим устройством:

Пр — преобразователь; ПК — переключатель каналов; *ВУ* — времязадающее устройство; $T_{пх}(T_{о.х})$ — триггер прямого (обратного) хода; ВПХ (ВОХ) — времязадающее устройство прямого (обратного) хода; СС1, СС2 — схемы сравнения; СПОХ — счетчик паузы обратного хода; РС1 (РС2) — реверсивные счетчики; СФПХ (СФОХ) — счетчики формирования прямого (обратного) хода; $C_{=1}$, $C_{=2}$ — счетчики постоянных составляющих значений времени длительностью $t_{фор.\,п.\,х}$, $t_{фор.\,о.\,х}$; $C_{\sim 1}$, $C_{\sim 2}$ — счетчики переменных составляющих значений времени $t_{фор.\,п.\,х}$ ($t_{фор.\,о.\,х}$); НПХ (КПХ) — сигналы начала (конца) прямого хода

На участке прямого хода скорость ω зеркала постоянна, на участке обратного хода ω изменяется и делится на три подучастка: I — торможение прямого хода, начало обратного; II — движение зеркала по инерции; III — торможение обратного хода, начало прямого.

Указанное движение зеркала осуществляется подачей в обмотку катушки (ОК) сканирующего устройства импульсного управляющего напряжения со следующими уровнями и их длительностями:

$U_{фор.\,о.\,х}(t_{фор.\,о.\,х})$ — форсировки обратного хода;

$U_{фор.\,п.\,х}(t_{фор.\,п.\,х})$ — форсировки прямого хода;

$U_{п.\,х}(t_{у.\,п.\,х})$ — напряжения прямого хода, $U_{п}(t_{п})$ — напряжения паузы.

Сканирование ведется следующим образом. На вход СЛУ поступают сигналы «Начала» (Н) и «Конца» (К) прямого хода с датчиков ДПН и ДПК. Сигналы идут парные, так как зеркало проходит перед каждым датчиком дважды. Частота следования сигналов зависит от скорости вращения зеркала.

СЛУ производит обработку сигналов Н и К и формирует сигналы управления $U_{\text{п. х}}$, $U_{\text{фор. о. х}}$, $U_{\text{п}}$ необходимой длительности $t_{\text{у. п. х}}$, $t_{\text{фор. о. х}}$ и после некоторой паузы $t_{\text{п}}=\text{const} - U_{\text{фор. п. х}}$ длительностью $t_{\text{фор. п. х}}$. Сигналы управления поступают на ФН, где формируются импульсы напряжения определенной амплитуды $U_{\text{у}}$ с длительностью, равной длительности сигналов управления.

Описание структуры проектируемой СЛУ. Из анализа особенностей функционирования и расчета временны́х параметров СЛУ следует, что в состав структуры СЛУ (рис. 8.13) должны входить следующие блоки:

преобразователя (Пр), предназначенного для преобразования входных сигналов Н и К в сигналы НПХ и КПХ;

времязадающего устройства обратного хода (ВОХ), предназначенного для формирования эталонного времени обратного хода $t_{\text{э. о. х}}$; отсчет $t_{\text{э. о. х}}$ начинается с момента появления сигнала КПХ и заканчивается формированием импульса КНПХ (контрольный сигнал начала прямого хода);

времязадающего устройства прямого хода (ВПХ), предназначенного для формирования эталонного времени прямого хода $t_{\text{э. п. х}}$; отсчет $t_{\text{э. п. х}}$ начинается с момента поступления сигнала НПХ и заканчивается формированием импульса ККПХ (контрольный сигнал конца прямого хода);

переключателя каналов (ПК), предназначенного для переключения каналов прямого и обратного хода;

триггера обратного хода ($T_{\text{о. х}}$), предназначенного для формирования управляющего импульса формирования прямого хода $U_{\text{фор. о. х}}$ длительностью $t_{\text{фор. о. х}}$;

триггера прямого хода ($T_{\text{п. х}}$), предназначенного для формирования управляющего импульса формирования прямого хода $U_{\text{фор. п. х}}$ длительностью $t_{\text{фор. п. х}}$;

схем сравнения (СС1 и СС2), необходимых для сравнения длительностей $t_{\text{о. х}}$ и $t_{\text{э. о. х}}$ по временно́му рассогласованию между сигналами НПХ и КНПХ (СС1), а также для сравнения длительностей $t_{\text{п. х}}$ и $t_{\text{э. п. х}}$ по временно́му рассогласованию между сигналами КПХ и ККПХ (СС2);

счетчиков $C_{=1}$ и $C_{=2}$ — постоянных составляющих значений времени $t_{\text{фор. о. х}}$ и $t_{\text{фор. п. х}}$ соответственно. Они формируют длительности $t_{\text{пост}}^{\text{фор. о. х}}$ и $t_{\text{пост}}^{\text{фор. п. х}}$;

счетчиков $C_{\sim 1}$ и $C_{\sim 2}$ — переменных составляющих значений времени $t_{\text{фор. о. х}}$ и $t_{\text{фор. п. х}}$ — соответственно. Они формируют длительности $t_{\text{перем}}^{\text{фор. о. х}}$ и $t_{\text{перем}}^{\text{фор. п. х}}$;

реверсивных счетчиков РС1 и РС2, предназначенных для управления счетчиками $С_{\sim 1}$ и $С_{\sim 2}$ в зависимости от состояния схем сравнения СС1 и СС2.

В структуре СЛУ можно выделить три укрупненных блока: счетчики форсировки обратного (СФОХ) и прямого (СФПХ) хода, обеспечивающие формирование и регулировку длительности $t_{\text{фор. о. х}}$ и $t_{\text{фор. п. х}}$ соответственно; времязадающее устройство (ВУ), выполняющее преобразование сигналов Н и К в сигналы НПХ и КПХ; формирование эталонных интервалов длительностью $t_{\text{э. о. х}}$, $t_{\text{э. п. х}}$, $t_{\text{п}}$; формирование управляющего сигнала длительностью $t_{\text{у. п. х}}$.

Детальный алгоритм функционирования СЛУ сканирующим устройством заключается в следующем. По окончании прямого хода зеркала с датчика ДПК на вход системы поступает сигнал «Конец» (К), который с помощью преобразователя преобразуется в сигнал КПХ и инверсный ему сигнал $\overline{\text{КПХ}}$. Сигнал $\overline{\text{КПХ}}$ устанавливает ПК в положение, разрешающее работу канала обратного хода. ПК представляет собой RS-триггер ($Т_{\text{с}}$). Если триггер находится в состоянии «1», то разрешается работа канала прямого хода с формированием управляющего импульса $t_{\text{у. п. х}}$, если в состоянии «0», то разрешается работа канала обратного хода.

Сигнал разрешения обратного хода с триггера $Т_{\text{с}}$ поступает на триггер $Т_{\text{о. х}}$, который перебрасывается в состояние «1». С этого момента начинается формирование управляющего сигнала форсировки обратного хода $t_{\text{фор. о. х}}$.

Длительность $t_{\text{фор. о. х}}$ формируют узлы $Т_{\text{о. х}}$, $С_{=1}$, СС1, $С_{\sim 1}$, РС1, которые входят в СФОХ. С появлением лог. 1 на прямом выходе $Т_{\text{о. х}}$ включается счетчик $С_{=1}$, который отсчитывает постоянную составляющую $t_{\text{фор. о. х}}$ (по заданию 8 мс). Затем включается счетчик $С_{\sim 1}$, который отсчитывает переменную, составляющую $t_{\text{фор. о. х}}$; длительность $t_{\text{перем}}^{\text{фор. о. х}}$ зависит от величины кода, записанного в реверсивном счетчике РС1, и изменяется согласно заданию от 0 до 14 мс. По истечении времени $t_{\text{перем}}^{\text{фор. о. х}}$ со счетчика $С_{\sim 1}$ поступает сигнал, который перебрасывает $Т_{\text{э. х}}$ в состояние «0», на этом формирование $t_{\text{фор. о. х}}$ заканчивается. В этот же момент происходит включение счетчика паузы СПОХ, который отсчитывает длительность $t_{\text{п}}$, после чего формируется сигнал $Т_{\text{з}}$, который перебрасывает $Т_{\text{п. х}}$ в состояние «1», что соответствует началу формирования управляющего сигнала форсировки прямого хода $t_{\text{фор. п. х}}$. Длительность $t_{\text{фор. п. х}}$ формируется устройством СФПХ, в которое входят узлы $Т_{\text{п. х}}$, $С_{=2}$, $С_{\sim 2}$, РС2, СС2. СФПХ работает аналогично СФОХ.

По истечении $t_{\text{фор. п. х}}$ со счетчика $С_{\sim 2}$ поступает сигнал, который перебрасывает триггер $Т_{\text{п. х}}$ в состояние «0», а также воздействует на $Т_{\text{с}}$ (ПК), перебрасывая его в состояние «1», т. е. заканчивается формирование $t_{\text{фор. п. х}}$ и начинается формирование $t_{\text{у. п. х}}$, ко-

торое заканчивается с появлением сигнала $\overline{КПХ}$. Так завершается первый цикл работы СЛУ.

Регулировка скорости вращения зеркала, а также длительности прямого и обратного хода зеркала происходит за счет изменения длительностей $t_{фор.\,о.\,х}$ и $t_{фор.\,п.\,х}$.

Изменение $t_{фор.\,о.\,х}$ ведется следующим образом. Сигнал $\overline{КПХ}$ включает двоичный счетчик ВОХ, выходы которого собираются на элемент И — НЕ. С приходом сигнала $\overline{КПХ}$ счетчик сбрасывается в «0». На выходе элемента И — НЕ возникает лог. 1, которая разрешает поступление частоты на счетный вход счетчика. С этого момента начинается отсчет эталонного времени обратного хода $t_{э.\,о.\,х}$. Отсчет времени заканчивается появлением лог. 1 на всех выходах счетчика. Тогда на выходе элемента И — НЕ образуется сигнал лог. 0, который запрещает поступление частоты на вход счетчика. В этот же момент времени формирователем F формируется короткий сигнал $\overline{КНПХ}$, который информирует о конце эталонного времени обратного хода.

По окончании обратного хода зеркала с датчика ДПК на вход системы поступает сигнал Н, который с помощью Пр преобразуется в сигнал $\overline{НПХ}$ и инверсный ему сигнал $\overline{НПХ}$. Сигналы $\overline{НПХ}$ и $\overline{КНПХ}$ поступают на схему сравнения СС1, где определяется временно́е рассогласование между ними, которое записывается в реверсивный счетчик РС1, в виде двоичного кода поступает на счетчик $С_{\sim 1}$, который увеличивает, уменьшает либо оставляет без изменения длительность $t_{фор.\,о.\,х}$ в зависимости от величины кода РС1.

Возможны следующие временны́е состояния сигналов $\overline{НПХ}$ и $\overline{КНПХ}$:

1. Сигнал $\overline{НПХ}$ приходит на СС1 раньше сигнала $\overline{КНПХ}$, т. е. $t_{о.\,х} < t_{э.\,о.\,х}$. Оба сигнала не входят в зону нечувствительности. В этом случае триггер знака перебрасывается в состояние «1» и разрешается сложение РС1. Код в РС1 увеличивается, $t_{фор.\,о.\,х}$ увеличивается.

2. Сигнал $\overline{НПХ}$ приходит после сигнала $\overline{КНПХ}$, т. е. $t_{о.\,х} > t_{э.\,о.\,х}$. Оба сигнала не входят в зону нечувствительности. В этом случае триггер знака перебрасывается в состояние «0». РС1 производит вычитание, $t_{фор.\,о.\,х}$ уменьшается.

3. Сигналы $\overline{НПХ}$ и $\overline{КНПХ}$ входят в зону нечувствительности, т. е. с требуемой точностью выполняется равенство $t_{о.\,х} = t_{э.\,о.\,х}$. На выходе узла, задающего зону нечувствительности, образуется сигнал, запрещающий работу РС1. $t_{о.\,х}$ сохраняет значение, имевшее место до сравнения НПХ и КНПХ.

Изменение длительности $t_{фор.\,п.\,х}$ ведется следующим образом. Сигнал $\overline{НПХ}$ включает счетчик ВПХ, формирующий эталонное вре-

мя прямого хода $t_{э.п.х}$. Счетчик ВПХ работает аналогично ВОХ. По истечении $t_{э.п.х}$ ВПХ формирует контрольный сигнал конца прямого хода ККПХ. Сигналы КПХ и ККПХ поступают на схему сравнения СС2, где определяется временно́е рассогласование между ними, которое записывается в РС2 и в виде двоичного кода поступает на счетчик $С_{\sim 2}$. Счетчик $С_{\sim 2}$ увеличивает, уменьшает или оставляет без изменения длительность $t_{фор.п.х}$ в зависимости от кода РС2. Здесь также возможны три временны́х состояния $\overline{КПХ}$ и $\overline{ККПХ}$:

Таблица 8.8. **Расчетные данные счетчиков типа 134ИЕ**

Обозначение счетчика	Число разрядов	Опорная частота, кГц	Максимальное время отработки, мс
СПОХ	12	500	8,19
ВОХ	12	125	32,76
ВПХ	14	250	65,53
$С_{=1}$	12	500	8,19
$С_{=2}$	12	500	8,19
$С_{\backsim 1}$	11	500	4,09
$С_{\backsim 2}$	11	500	4,09

1. Сигнал $\overline{КПХ}$ приходит раньше сигнала $\overline{ККПХ}$, т. е. $t_{п.х} < t_{э.п.х}$, $t_{фор.п.х}$ уменьшается.

2. Сигнал $\overline{КПХ}$ приходит позже сигнала $\overline{ККПХ}$, т. е. $t_{п.х} > t_{э.п.х}$, $t_{фор.п.х}$ увеличивается.

3. Сигналы $\overline{КПХ}$ и $\overline{ККПХ}$ входят в зону нечувствительности, т. е. с необходимой точностью выполняется равенство $t_{п.х} = t_{э.п.х}$, $t_{фор.п.х}$ остается без изменения.

Так как разрабатываемая СЛУ входит в состав бортовой сканирующей системы, то с учетом весогабаритных ограничений, требований по надежности, помехозащищенности, энергопотреблению в качестве элементной базы выбраны микромощные микросхемы серии 134.

Данные расчета частот и числа разрядов счетчиков СЛУ сведены в табл. 8.8. Потребляемая мощность СЛУ составила 2,17 Вт, а вероятность безотказной работы при времени активной работы в 5000 часов — 0,957 $ч^{-1}$.

ЗАКЛЮЧЕНИЕ

За пределами данной книги осталось много вопросов, связанных с описанием особенностей функционирования конкретных микропроцессорных средств, их проверкой, программированием, тестированием, контролем, отладкой, моделированием, а также применением в конкретных областях науки и техники. Не рассматривались проблемы комплексирования разнообразных микропроцессорных средств в единые локальные управляющие и вычислительные сети, способы их организации, обеспечения помехоустойчивости приема — передачи информации по каналам связи и др. Ряд из этих вопросов подробно рассмотрен, например, в [3, 6, 8—10, 14 и др.] и при желании может быть изучен по этим источникам.

В настоящее время программно-настраиваемые БИС и СБИС, микропроцессорные комплексы технических средств, а также их программное обеспечение бурно развиваются. Ряд моделей МП и МЭВМ, упомянутых в книге, заменяется более развитыми и совершенными моделями, содержащими наборы команд микропроцессорных средств предыдущих поколений, что позволяет быстро переходить на использование новых 16-, 32-разрядных МП и МЭВМ при сохранении имеющихся систем поддержки программного обеспечения. К числу подобных микропроцессорных средств относится большинство зарубежных и отечественных моделей, в особенности таких фирм как «Интел», «Моторола» и др.

Рассмотренный в книге круг вопросов проектирования является лишь малой частью методов синтеза и обеспечения надежности и отказоустойчивости МПСУ, применяемых на практике. Придание этих свойств ранее созданным системам, разработанным на основе перспективных комплексов технических средств, не оснащенных встроенным контролем, осуществляется преимущественно с помощью методов резервирования (дублирования, маскирования со сравнением результатов и др.) и программного восстановления работоспособности. При разработке новых систем предпочтение отдается встроенным схемам контроля работоспособности, а ее восстановление, как правило, производится аппаратными средствами.

Авторы надеются, что приведенные в настоящей книге методы проектирования СЛУ на основе МП, МЭВМ и других программируемых средств, пути и способы повышения их надежности и отказоустойчивости будут способствовать ускорению внедрения микропроцессорной техники в разнообразные системы управления.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ 1

Основные конструктивно-технологические и функциональные особенности СИС мультиплексоров и дешифраторов

В табл. П.1.1 приведены основные конструктивно-технологические параметры мультиплексоров, дешифраторов и их разновидностей из числа наиболее распространенных серий микросхем.

На рис. П.1.1 — П.1.4 приведены условные обозначения, струк-

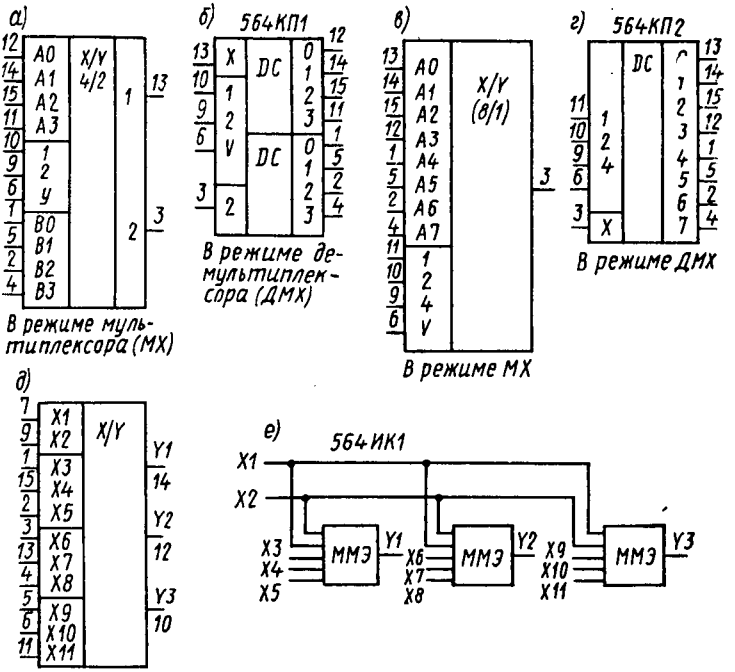


Рис. П.1.1. СИС цифровых КМДП — мультиплексоров:

а — г — мультиплексор-демультиплексор серии К564КП1, К564КП2; *д — е* — строенный мажоритарно-мультиплексорный элемент серии К564ИК1

турные схемы и таблицы состояний отдельных СИС, приведенных в табл. П.1.1, а также в табл. П.1.2 — П.1.4.

Основные характеристики серийных БИС ЗУ.

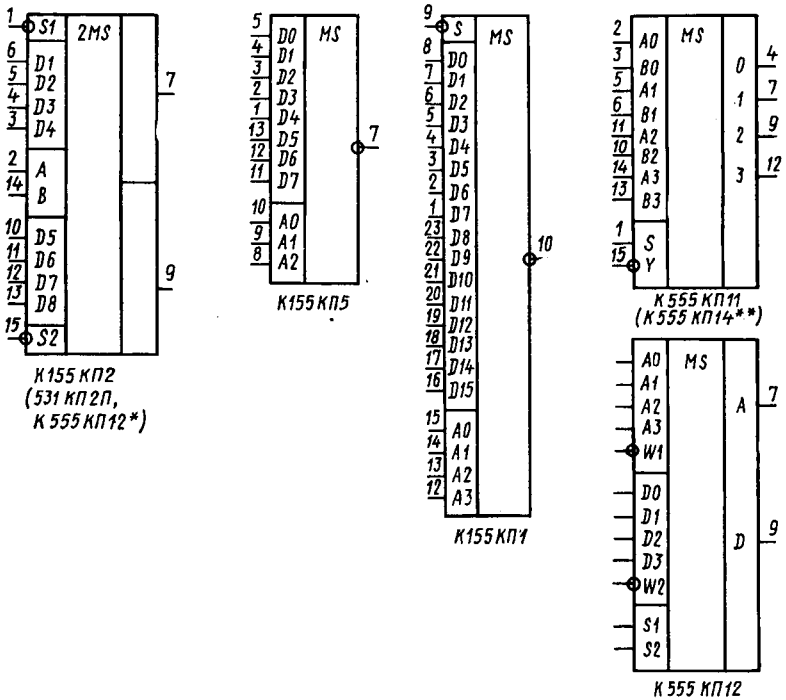


Рис. П.1.2. Мультиплексоры серий К155, К555 (условные обозначения)

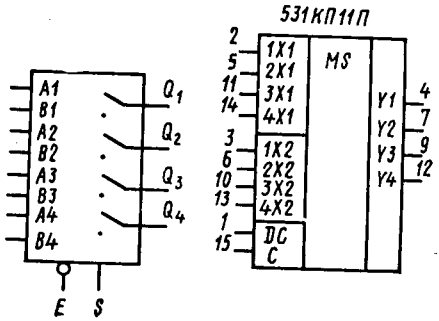


Рис. П.1.3.
СИС К531КП11П:

а — пояснение принципа действия; *б* — условное обозначение

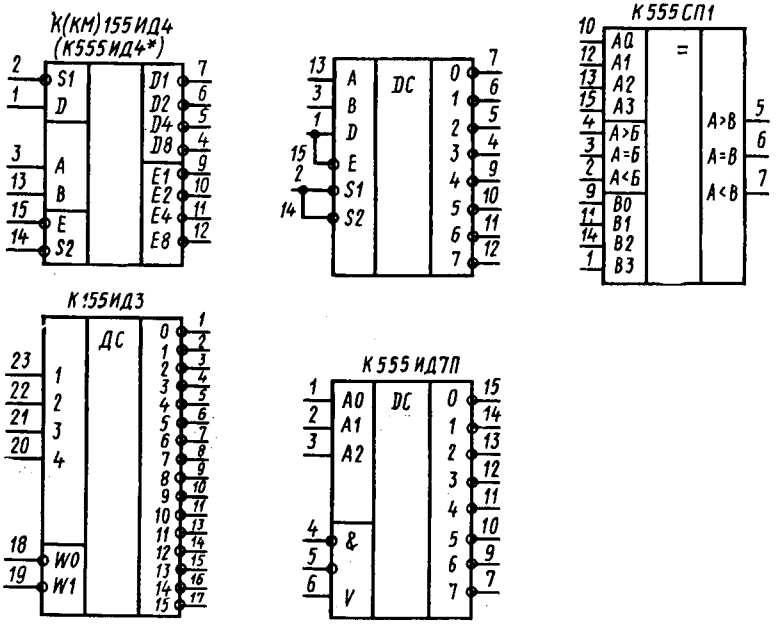


Рис. П.I.4. Дешифраторы серий К155, К555 (условные обозначения)

В табл. П.II.1 — П.II.3 и на рис. П.II.1 приведены основные характеристики серийных БИС ПЗУ и ОЗУ, их электрические параметры и временны́е диаграммы режимов работы.

В табл. П.II.4 — П.II.6 приведены основные характеристики некоторых типов отечественных и зарубежных программируемых БИС.

Ниже рассмотрены некоторые модели, методы повышения надежности и оценки сложности программных средств и рекомендации по их применению.

В табл. П.III.1 приводятся основные показатели оценки сложности программных средств, в табл. П.III.2 — методы повышения надежности программ и рекомендации по их применению в проектах ПО, в табл. П.III.3 — состав требований, предъявляемых к будущему программному обеспечению.

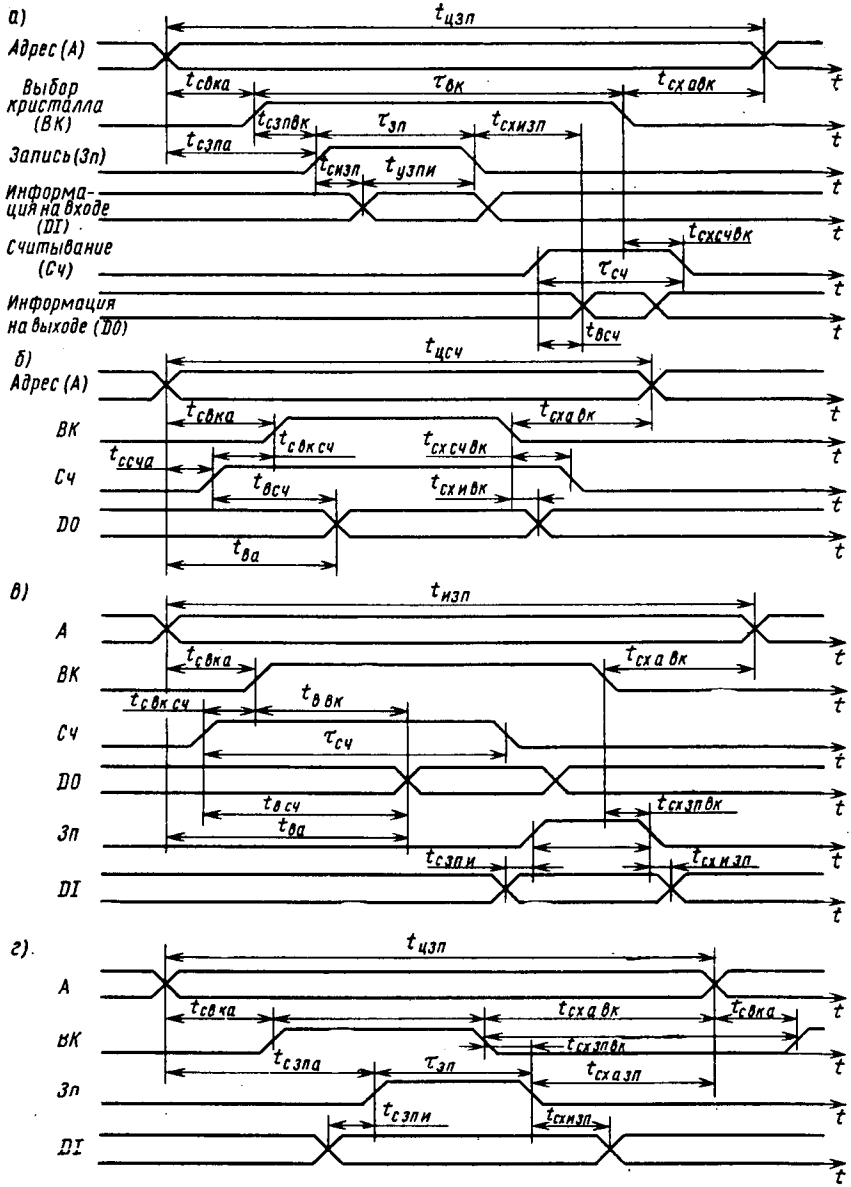


Рис. П.II.1. Временны́е диаграммы режимов работы БИС ЗУ:
а — записи-считывания; *б* — считывания; *в* — считывания-записи; *г* — записи

Таблица П.1.1. **Основные конструктивно-технологические параметры комбинационных СИС**

Наименование микросхемы	Тип, обозначение	Всего входов				Выходов	
		управляющих	информационных	стробирующих		в том числе	
				прямых	инверсных	прямых	инверсных
Селектор-мультиплексор данных на 16 каналов со стробированием	К155КП1	4	16	—	1	—	1
То же, но на 8 каналов со стробированием	К(КМ)155КП7	3	8	—	1	1	1
То же, но без стробирования	К(КМ)155КП5	3	8	—	—	—	1
Сдвоенный селектор-мультиплексор 4 на 1 со стробированием	К(КМ)155КП2 К531КП2П	2	2×4	—	2×1	2×1	—
Сдвоенный дешифратор-мультиплексор 2 на 4	К(КМ)155ИД4	2 2	2×1 3	— —	2×1 2	— —	2×4 8
Дешифратор-демультиплексор 4 линии на 16 (преобразование двоично-десятичного кода в десятичный)	К155ИД3	—	4	—	2	—	16 8
Двоичный дешифратор на 8 направлений	К555ИД7П	3	3	1	2	—	
Четыре двухвходовых коммутатора с тремя состояниями, прямым выходом	К555КП11	1	2	1	1	4×1	—
То же, но с инверсными выходами	К555КП14	1	2	—	1	—	4×1
Четырехканальный МОП-ключ со схемой управления	К590КН2	4	4	—	—	4	—
Двоично - десятичный дешифратор	564ИД1	—	4	—	—	10	—
Строенный мажоритарный мультиплексорный элемент	564ИК1	2	3×3	—	—	3×1	—

Продолжение табл. П.1.1

Наименование микросхемы	Тип, обозначение	Всего входов				Выходов	
		управляющих	информационных	стробирующих		в том числе	
				прямых	инверсных	прямых	инверсных
Двойной четырехканальный мультиплексор-демультиплексор	564ИК1	2	4/2	—	1	2/4	—
Мультиплексор-демультиплексор	564КП2	3	8/3	—	1	1/8	—
Четырехразрядный селектор 2 канала на 1 с тремя состояниями	К531КП1П	1	4×2	1	—	4×1	—
Двухразрядный четырехканальный коммутатор с тремя состояниями	К555КП12	2	2×1	—	2×1	2×1	—
Сдвоенный дешифратор 2 входа — 4 выхода	К555ИД4	2	2×1	—	—	2×4	2×4

Таблица П.1.2. **Таблица состояний для К564КП1** (см. рис. П.1.1, *а*, *б*)

V	2	1	Вход — выход
Н	Н	Н	13—12 3—1
Н	Н	В	13—14 3—5
Н	В	Н	13—15 3—2
Н	В	В	13—11 3—4
В	Х	Х	—

Таблица П.I.3. Таблица состояний для К564КП2
(см. рис. П.I.1, *в*, *г*)

V	4	2	1	Выход
Н	Н	Н	Н	3—13
Н	Н	Н	В	3—14
Н	Н	В	Н	3—15
Н	Н	В	В	3—12
Н	В	Н	Н	3—1
Н	В	Н	В	3—5
Н	В	В	Н	3—2
Н	В	В	В	3—4
В	Х	Х	Х	—

Таблица П.I.4. Таблица состояний для К564ИК1
(см. рис. П.I.1, *д*, *е*)

x_1	x_2	y_1	y_2	y_3
Н	Н	$x_3x_4 \vee x_3x_5 \vee x_4x_5$	$x_6x_7 \vee x_6x_8 \vee x_7x_8$	$x_9x_{10} \vee x_9x_{11} \vee x_{10}x_{11}$
В	Н	x_3	x_6	x_9
Н	В	x_5	x_8	x_{11}
В	В	x_4	x_7	x_{10}

Таблица П.I.5. **Таблица состояний для К155КП2** (см. рис. П.I.2)

Входы						Выход
управляющие	информационные				стробирующие	Y1, Y2
B A	D1, D5	D2, D6	D3, D7	D4, D8	S1, S2	
X X	X	X	X	X	B	H
H H	H	X	X	X	H	H
H H	B	X	X	X	H	B
H B	X	H	X	X	H	H
H B	X	B	X	X	H	B
B H	X	X	H	X	H	H
B H	X	X	B	X	H	B
B B	X	X	X	H	H	H
B B	X	X	X	B	H	B

Таблица П.I.6. **Таблица состояний для К531П11П** (см. рис. П.I.3)

E	*S*	Вход — выход
H	*H*	A_i—Q_i
H	*B*	B_i—Q_i
B	*X*	$Q_i=0$ или трехстабильное состояние

Таблица П.I.7. **Таблица состояний для К155ИД4** (см. рис. П.I.4)

Входы				Выходы			
B	A	S1 / S2	D / E	D1 / E1	D2 / E2	D3 / E3	D4 / E4
X	X	1	X	1	1	1	1
0	0	0	1/0	0	1	1	1
0	1	0	1/0	1	0	1	1
1	0	0	1/0	1	1	0	1
1	1	0	1/0	1	1	1	0
X	X	X	0/1	1	1	1	1

Таблица П.II.1. Основные характеристики серийных БИС ПЗУ

Наименование ЗУ	Тип ПЗУ	Технология	Организация	Быстродействие t, нс, не более	Энергопотребление, мВт, не более	Напряжение питания, В	Диапазон рабочих температур, °С
К505РЕ3	ПЗУМ	p-МОП	512×8	1300	1000	+5, —12	—60 ... +85
1601РР2А	ЭППЗУ	p-МНОП	512×2	1200	610	+5, —12	—60 ... +85
1601РР2Б	ЭППЗУ	p-ЛИЗ/МОП	512×2	800	610	+5, —12	—60 ... +85
К1601РР1	ЭППЗУ	p-ЛИЗ/МОП	1К×4	1500	570	+5, —12	—45 ... +70
558РР1	ЭППЗУ	p-ЛИЗ/МОП	256×8	2000	200	+5, —12	—60 ... +55
К558РР2	ЭППЗУ	p-ЛИЗ/МОП	4К×4	800	800	+5, —12	—10 ... +70
573РФ1 (11; 13)	УФПЗУ	n-ЛИЗ/МОП	1К×8	450	1200	±5, +12	—60 ... +85
К573РФ2	УФПЗУ	n-МОП	2К×8	900	1200	+5, —12	—10 ... +70
Электроника-8071	УФПЗУ	n-МОП	4К×16	400	450	+5	—10 ... +70
К568РЕ1	ПЗУМ	n-МОП	2К×8	800	350	±5, +12	—60 ... +85
КР565РТ1	ППЗУ	Сложные n-МОП с обрамлением	1К×4	300	13,5	+5, —12	—10 ... +70
556РТ4	ППЗУ	ТТЛШ	256×4	70	680	+5	—60 ... +125
556РТ5	ППЗУ	ТТЛШ	512×8	90	1000	+5	—60 ... +125
К556РЕ4	ПЗУМ	ТТЛШ	2К×8	110	850	+5	—10 ... +70
541РТ1	ППЗУ	$И^2Л$	256×4	80	450	+5	—60 ... +125
500РЕ149	ППЗУ	ЭСЛ	256×4	35	700	—5,2	—10 ... +70
К596РЕ1	ПЗУМ	ТТЛ	8К×8	350	655	+4	—10 ... +70
К155РЕ21 ... К155РЕ24	ПЗУМ	ТТЛ	256×4	60	650		
К505РР4	ЭППЗУ	p-МДП	512×2	1200	650		
К505РР1	ЭППЗУ	p-ЛИЗ/МДП	256×8	850	700		
К541РТ5	ППЗУ	ТТЛШ	256×4	80	400		
К555РЕ4	ПЗУМ	ТТЛШ	2К×8	110	850		

Таблица П.II.2. **Основные характеристики БИС ОЗУ**

Наименование БИС ЗУ	Тип ЗУ	Технология	Организация	Быстродействие t, нс, не более	Энергопотребление, мВт, не более	Напряжение питания, В	Диапазон рабочих температур, °С
505РУ4(2)	»	КМОП	4К×1	390 (430)	50	+5	—10 ... +75
132РУ2А(Б)	»	КМОП	256×1	600	50		
КР565РУ4	Статическое	p-МОП	256×1 (1К×1)	850 (600)	460 (750)	+5, —12	—60 ... +85
К565РУ2А(Б)	»	n-МОП	1К×1	400 (550)	350	+5	—60 ... +85
132РУ1	»	n-МОП	4К×1	160	700	+5	—10 ... +70
505РУ6	»	n-МОП	1К×1	450 (850)	350	+5	—10 ... +70
565РУ3А(Б)	»	n-МОП	1024×1	400	250	+5	—60 ... +125
565РУ1А(Б, В)	n-МОП	n-МОП	1024×1	450	200	+5	—60 ... +85
К565РУ5	Динамическое	n-МОП	16К×1	200 (280)	655 (490)	±5, +12	—10 ... +70
КР581РУ4	»	n-МОП	4К×1	200 (300, 340)	717	±5, +12	—10 ... +70
К552РУ1	Динамическое	n-МОП	64К×1	200 ... 300	400	+5 +5, +12, —3	—10 ... +70
537РУ1(2)	»	n-МОП	16К×1	150	500		—10 ... +70
1603РУ1	Квазистатическое	n-МОП	16К×1	450	325	±5, —12	—10 ... +70
564РУ2А(Б)	Статическое	КМОП	1К×1 (4К×1)	800 (320)	30 (50)	+5	—60 ... +85
КР188РУ3А(Б)	»	КМОП	256×4	450	160	+5	—60 ... +85
КР188РУ2А	»	КМОП	256×1	650 (1180)	2	+3 ... +15	—60 ... +125
К561РУ2А	»	КМОП	4К×1	350 (500)	100	+5	—10 ... +70
К176РУ2	»	КМОП	256×1	500	0,05	+5	—10 ... +70
Б537РУ1	»	КМОП	256×1	600	0,1	+6 ... +12	—45 ... +85
737РУ1	»	КМОП	256×1	550	4,5	+9	—45 ... +70
КР537РУ2А(Б)	»	КМОП	1024×1	800	21	+5	—60 ... +85
К188РУ1	»	КМОП	1024×1	800	21	+5	—60 ... +85

Продолжение табл. П.II.2

Наименование БИС ЗУ	Тип	Технология	Организация	Быстродействие, нс, не более	Энергопотребление, мВт, не более	Напряжение питания, В	Диапазон рабочих температур, °С
133РУ5	»	ТТЛ	256×1	60	700	+5	—60 ... +125
185РУ4(5)	»	ТТЛ	256×1 (1К×1)	160 (250)	350 (450)	+5	—60 ... +85
185РУ2(3)	»	ТТЛ	64×1	70 (160)	200 (240)	+5	—60 ... +85
155РУ2(5)	Статическое	ТТЛ	16×4 (256×1)	60	525 (700)	+5	—10 ... +75
530РУ2	»	ТТЛШ	16×4	60	525	+5	—60 ...+125
(КР)541РУ1 (А, Б)	»	$И^2Л$	4К×1 (2К×1)	120	450	+5	—60 ... +85
(КР)541РУ2(3)	»	$И^2Л$	4К×1 (16К×1)	90 (120)	500 (750)	+5	—10 ... +70
К134РУ6	»	ТТЛ	1К×1	500	600	+5	—10 ... +70
100РУ415(401)	»	ЭСЛ	1К×1 (16×1)	30	700	—5,2	—10 ... +75
К500РУ410(470)	»	ЭСЛ	256×1 (4К×1)	40 (50)	760 (820)	—5,2	—10 ... +70
К500РУ411(412)	»	ЭСЛ	128×1	40 (50)		—5,2	—10 ... +70
К507РУ1	Динамическое	*p*-МДП	1К×1	600	750		
К500РУ415	Статическое	ЭСЛ	1К×1	30	700		

Примечание. В скобках указаны разновидности БИС ОЗУ и их технические параметры. Пропуски в параметрах свидетельствуют об отсутствии данных в технической литературе.

Таблица П.II.3. **Основные электрические параметры БИС ЗУ**

Название параметра	Обозначение
Статические параметры	
1. Параметры сопряжения БИС ЗУ с входными и выходными устройствами:	
напряжение источника питания	$U_{ип}$
напряжение лог. 0 (лог. 1) сигнала выходной информации	$U^0_{вых.и}$ ($U^1_{вых.и}$)
ток лог. 0 (лог. 1) сигнала выходной информации	$I^0_{вых.и}$ ($I^1_{вых.и}$)
пороговое напряжение лог. 0 (лог. 1)	$U^0_{пор}$ ($U^1_{пор}$)
пороговый входной ток	$I_{вх.пор}$
входной ток лог. 0 (лог. 1)	$I^0_{вх}$ ($I^1_{вх}$)
ток потребления	$I_{пот}$
2. Параметры устойчивости БИС ЗУ при предельных режимах функционирования:	
максимальное напряжение источника питания	$U_{ип\ max}$
максимальное (минимальное) выходное напряжение	$U_{вых\ max\ (min)}$
максимальное (минимальное) входное напряжение (ток)	$U_{вх\ max\ (min)}$ ($I_{вх\ max\ (min)}$)
максимальный выходной ток	$I_{вых\ max}$
максимальная емкость нагрузки	$C_{н\ max}$
3. Технологические параметры:	
ток короткого замыкания	$I_{к.з}$
ток утечки на входе (выходе)	$I_{ут.вх\ (вых)}$
входная (выходная) емкость	$C_{вх(вых)}$
Динамические параметры	
1. Время выборки — задержка получения информации на выходе относительно сигнала:	
считывания	$t_{в.сч}$
адреса	$t_{в.а}$
выбора кристалла	$t_{в.вк}$
2. Время сдвига — интервал времени между началами двух заданных входных сигналов на разных входах БИС ЗУ:	
адреса и выбора кристалла	$t_{с.в.к.а}$
адреса и записи	$t_{с.зл.а}$
записи и информации	$t_{с.и\ зп}$
адреса и считывания	$t_{с.сч.а}$
адреса и выбора кристалла	$t_{с.вк.сч}$
информации и записи	$t_{с.зп.и}$
3. Время удержания — интервал времени между началом сигнала входной информации и окончанием сигнала записи	$t_{у\ зп.и}$
4. Время восстановления — интервал времени между окончанием сигнала выполнения данного режима и началом сигнала, определяющего новый цикл работы БИС ЗУ:	
выбора кристалла и адреса	$t_{вос.в.к.а}$
считывания и адреса	$t_{вос.с.ч.а}$
записи и адреса	$t_{вос.зп.а}$

Продолжение табл. П.II.3

Название параметра	Обозначение
5. Время сохранения — интервал времени между окончанием двух заданных входных сигналов на разных входах:	
выбора кристалла и считывания	$t_{\text{сх. сч. в. к}}$
выбора кристалла и информации	$t_{\text{сх.и вк}}$
выбора кристалла и записи	$t_{\text{сх.зп.в.к}}$
записи и информации	$t_{\text{сх и зп}}$
выбора кристалла и адреса	$t_{\text{сх.а.в.к}}$
6. Время цикла — интервал времени между началами (окончаниями) сигналов на одном из управляющих входов, в пределах которого микросхема выполняет одну из функций;	
записи	$t_{\text{ц.зп}}$
считывания	$t_{\text{ц.сч}}$
записи — считывания	$t_{\text{ц.зп.сч}}$
считывания — записи	$t_{\text{ц.сч.зп}}$
7. Длительность сигнала — минимальный интервал времени между началом и окончанием данного сигнала, необходимый для однозначного выполнения заданного режима БИС ЗУ:	
выбора кристалла	$\tau_{\text{в. к}}$
записи	$\tau_{\text{зп}}$
считывания	$\tau_{\text{сч}}$

Таблица П.II.4. **Некоторые типы зарубежных программируемых матриц логики (ПМЛ)**

Тип ПМЛ	Число входов	Число выходов	Число секций с выходами типа I*	Число секций с выходами типа II*	Функция схемы на выходе ПМЛ
PAL10H8	10	8	—	—	И—ИЛИ
PAL12H6	12	6	—	—	И—ИЛИ
PAL14H4	14	4	—	—	И—ИЛИ
PAL16H2	16	2	—	—	И—ИЛИ
PAL10L8	10	8	—	—	И—ИЛИ—НЕ
PAL12L6	12	6	—	—	И—ИЛИ—НЕ
PAL14LA	14	4	—	—	И—ИЛИ—НЕ
PAL16L2	16	2	—	—	И—ИЛИ—НЕ
PAL16C1	16	2	—	—	И—ИЛИ, И—ИЛИ—НЕ
PAL16L8	10	8	6	—	И—ИЛИ—НЕ
PAL16R8	8	8	—	8	И—ИЛИ—НЕ
PAL16R6	8	8	2	6	И—ИЛИ—НЕ
PAL16R4	8	8	4	4	И—ИЛИ—НЕ
PAL16X4	8	8	4	4	Сложная функция
PAL16A4	8	8	4	4	То же

* На выходе секции типа I устанавливается управляемый буфер (рис. 2.5, *в*), на выходе типа II — синхронный D-триггер (рис. 2.5, *б*).

Таблица П.II.5. **Перечень некоторых типов зарубежных программируемых логических устройств (ПЛУ)**

Тип ПЛУ	Вид ПЛМ	Число				Емкость, бит	Время выборки, нс	Энергопотребление, мВт	Число выводов в БИС
		входов	выходов	термов	элементов памяти				
IM 5200	ППЛМ	14	8	48	—	1728	100	675	24
93458/59	ППЛМ	16	8	48	—		25		28
MMI 5775A	МПЛМ	14	8	96	—	3456	80	895	
MMI 6870/71	ППМЛ	14	8	48	—	1728			
DM8575/75A	МПЛМ	14	8	96	—	3456	100	895	
DM7576	МПЛМ	14	8	96	—				
82 S100/101	ППЛМ	16	8	48	—	1720	50	600	28
μPB450 (450A)	ППЛМД	24	16	72	16		250 (100)		48
SN 74 330/331	ППЛМ	12	6	50	—	1500	30	550	20
μPB450A-I	ППЛМД	24	16	72	16		70		48
82S104/105	ППЛМ	16	8	48	6		90	600	28
6275/6276	МПЗУ	11	8	—	—		110		
82S102/103	ППМВ	16	9	—	—		35	600	28
29693	ППМ	10	4	3*					20

* — число управляющих входов; — отсутствие элементов памяти; цифры в скобках приведены для модификации типа ПЛУ, свободные места в перечне параметров таблицы — отсутствие технических данных.

Таблица П.II.6. **Перечень серийных БИС ПЛМ, выпускаемых отечественной промышленностью**

Тип ПЛМ	Наименование и вид ПЛМ	Технология, диапазон температур, °С	Число				Время цикла, нс	Энергопотребление, мВт	Напряжение питания, В	Число выводов
			входов	выходов	термов	элементов памяти				
К587РП1,	БИС УП*	КМДП	18+6**	14	128	4	4,0	50	+9	42
КР587РП1	ППЛМ	—25 ... +70								
К588ВУ1	БИС УП* ППЛМ	КМДП —60 ... +85	16+4**	13	100	4	1,5	1,0	+5	42
U831—К1883РТ1	БИС УП* ППЛМ	*n*-МДП 0 ... +70	20+8	18	140 (256)	8	1,0	900	+5	48
К840ДИП24	ППЛМ		14	8	96	—	100	700—900	+5	24
КР556, 556РТ1,	ППЛМ	ТТЛШ								
556РТ2, М556РТ1	»	—10 ... +70	16	8	48	—	50	1000	+5	28
Р556РТ1, Р556РТ2	»									
582ИК1 ... 582ИК47 582ИК1А ... 582ИК47А	ППЛМ	И²Л	10	24		—		280	+5	48

* Управляющая память.
** Выходы элементов памяти.

Таблица П.III.1. **Основные показатели оценки сложности программных средств**

Показатели сложности ПО	Расчетные формулы	Составные компоненты
Общая логическая сложность программного модуля (L_{TOT})	$L_{TOT}=LS/EX+L_{LOOP}+L_{IF}+L_{BR}$	LS — общее число логических операторов; EX — число исполняемых операторов; L_{LOOP} — показатель сложности циклов, определяемый ниже; L_{IF} — показатель сложности условных операторов *IF*; L_{BR} — число ветвлений (всех операторов ветвления) в условных операторах BR×0,001 (0,001 — коэффициент значимости числа ветвлений в общей логич. сложности)
Показатель сложности взаимосвязей (C_{INF})	$C_{INF}=AP+0,5(SYS)$	AP — число связей с прикладными программами; SYS — число связей с системными программами; 0,5 — коэффициент значимости
Показатель сложности вычислений (CC)	$CC=(CS/EX)\times\times(L_{SYS}/\Sigma CS)CS$	CS — число операторов вычислительного характера; $L_{SYS}=\Sigma L_{TOT}$ — суммарная логическая сложность ПО (суммирование осуществляется по всем L_{TOT} — модулям)
Показатель сложности ввода — вывода ($C_{I/O}$)	$C_{I/O}=(S_{I/O}/EX)\times\times(L_{SYS}/\Sigma S_{I/O})S_{I/O}$	$S_{I/O}$ — число операций ввода — вывода, причем суммирование осуществляется по всем модулям системы
Удобочитаемость, или показатель простоты программы	$U_{READ}=COM/(TS+COM)$	TS — общее число операторов (исполняемых и неисполняемых, исключая комментарии), COM — число комментариев
Показатель общей сложности программного модуля	$C_{TOT}=L_{TOT}+0,1C_{INF}++0,2CC+0,4C_{I/O}++(-0,1)U_{READ}$	Коэффициенты задают относительно веса соответствующих показателей в аддитивной модели общей сложности ПО
Сложность циклов модуля	$L_{LOOP}=\Sigma m_i w_i/1000,$ $w_i=4^{i-1}\frac{3}{4Q-1}$ $\sum_{i=1}^{Q} w_i=1.$	m_i — число циклов в модуле, имеющих i-й уровень вложения; w_i — весовой множитель, Q — наивысший уровень вложенности циклов; 4 — нормирующий множитель
Сложность условных операторов IF	$L_{IF}=\Sigma n_i w_i/1000$	n_i — число условных операторов i-го уровня; w_i — весовой множитель (аналогичен L_{LOOP})

Таблица П.III.2. **Классификация методов повышения надежности программ и рекомендации по их применению в проектах ПО**

Методы обеспечения надежности ПО на этапах жизненного цикла	Ранг «ценностей»	Тип проекта			
		ТП	ПМС	ПЗТН	ПКН
Требования и спецификации					
Язык задания требований и спецификаций	1	1	1	4	4
Моделирование и имитация систем	*	2	1	5	5
Проектирование					
Проектирование сверху вниз	7	2	6	6	6
Проектирование структур данных	7	1	6	6	6
Проверка проекта	9	6	6	7	7
Блок-схемы	4	7	2	2	2
Решающие таблицы	*	6	6	6	6
Язык проектирования программ	7	1	6	6	6
HIPO (вход-процесс-выход)	5	1	1	6	6
Автоматический контролер проекта	3	1	3	3	6
Реализация					
Программирование сверху вниз	5	2	4	4	4
Программирование снизу вверх	5	7	6	6	6
Модульное программирование	10	7	7	7	7
Защита от ошибок	*	2	1	6	7
Отладка					
Проверка за столом	9	6	6	7	7
Коллективная проверка	9	2	1	5	7
Структурный анализ	4	1	1	3	5
Доказательство корректности	1	1	1	1 (7)	1 (7)
Поиск ошибок в исходном тексте	9	3	3	3	3 (7)
Анализатор полноты текста	6	1	1	3	3 (7)
Контролер утверждений	6	1	1	3	3
Диспетчер тестирования	*	7	6	6	6
Имитатор внешней среды	*	6	6	6	6
Генератор входных данных	*	2	3	3	3 (6)
Стандартизированное тестирование	*	3	3	3	3
Интерактивный метод поиска ошибок	5	3	3	3	3

Продолжение табл. П.III.2

Методы обеспечения надежности ПО на этапах жизненного цикла	Ранг «ценностей»	Тип проекта			
		ТП	ПМС	ПЗТН	ПКН
Преднамеренное внесение ошибок	2	1	1	1	1
Поиск ошибок посторонними лицами	2	1	1	1	1
Символическое выполнение	1	1	1	1	1
Математический контролер	*	1	1	3	3
Приемосдаточные испытания	8	6	6	7	7
Сопровождение					
Профилактическое сопровождение	9	6	7	7	7
Контроль изменений	8	6	6	7	7
Повторное тестирование	9	3	7	7	7
Регистрация ошибок	9	2	6	7	7
Руководство					
Группа контроля качества		(2)	(2)	(7)	7
Группа тестирования ПО		2	1	7	7
Группа контроля изменений		7	7	7	7
Бригада главного программиста		2	4	7	7
Планы, процедуры и отчеты тестирования		(7)	(7)	7	7
Моделирование надежности ПО		1	1	1	6
Определение момента окончания тестирования		1	1	4	4

Примечания: 1. Цифры 1—7 характеризуют степень применения приведенных методов: 1 — очень редко; 2 — иногда; 3 — при наличии, но не приобретайте; 4 — при наличии опыта эксплуатации; 5 — в критических областях; приобретайте, если нужно; 6 — всегда, если метод подходит; приобретайте, если нужно; 7 — всегда; приобретайте, если нужно.

2. Цифры в скобках приведены для больших и средних проектов ПО, * — ранжирование затруднено.

3. Типы проектов ПО обозначены: ТП — традиционный проект; ПМС — проект с малой стоимостью; ПЗТН — проект с заданными требованиями по надежности; ПКН — проект с критической надежностью.

Таблица П.III.3. **Структура и содержание документации по определению требований, предъявляемых к будущему ПО**

Глава	Содержание
Введение	Принципы организации документа; краткое содержание остальных глав; обозначение
1. Характеристика ЭВМ	Если ЭВМ задана, то ее общее описание с учетом специфических черт; если нет — основные требования к ЭВМ
2. Интерфейсы с аппаратурой	Краткое описание информации, получаемой и передаваемой ЭВМ
3. Функции программы	Что должна делать программа, чтобы удовлетворить требованиям в разных ситуациях; какой должна быть ее позиция на различные события
4. Временные ограничения	Как быстро и как часто должна выполняться каждая из функций
5. Требования к точности	Каковы допустимые отклонения выходных параметров от точных значений
6. Реакция на нежелательные ситуации	Что программа должна делать при выходе из строя датчиков, получения некорректных данных от оператора и т. п.
7. Подмножества	Какие части надо сделать легко удаляемыми из программы
8. Функциональные предположения	Характеристики программы, которые сохраняются при любых модификациях
9. Изменения	Типы внесенных или ожидаемых изменений
10. Словарь сокращений	Сначала составить словарь терминов, сокращений для себя, а затем для новичков
11. Источники	Аннотированная документация и список сотрудников с указанием, кто и на какие вопросы может ответить

СПИСОК ЛИТЕРАТУРЫ

Основная

1. *Балашов Е. П.* и др. Микро- и мини-ЭВМ. Л., 1984. 376 с.
2. *Каган Б. М., Сташин В. В.* Основы проектирования микропроцессорных систем автоматики. М., 1987.
3. Микропроцессоры/Под ред. Л. Н. Преснухина. В 3-х кн. М., 1986. 496 с.
4. *Соломатин Н. М.* Микропроцессоры. Выбор микроЭВМ. М., 1987.
5. *Арсеньев Ю. Н., Журавлев В. М.* Методы проектирования и обеспечения надежности микропроцессорных систем управления. Фрунзе, Фрунзенский политехнический ин-т, 1984. 120 с.
6. *Советов Б. Я.* и др. Применение микропроцессорных средств в системах передачи информации. М., 1987. 256 с.
7. *Трофимова И. П.* Системы обработки и хранения информации. М., 1989. 191 с.
8. *Казаринов Ю. М.* и др. Применение микропроцессоров и микроЭВМ в радиотехнических системах. М., 1988. 207 с.
9. *Балашов Е. П., Пузанков Д. В.* Микропроцессоры и микропроцессорные системы. М., 1981.
10. Микропроцессоры. В 9-ти кн. / Под ред. Л. Н. Преснухина. М., 1984.

Дополнительная

11. *Фурунжиев Р. И.* и др. МикроЭВМ в динамических системах. Минск, 1982, 207 с.
12. *Каляев А. В.* Многопроцессорные системы с программируемой архитектурой. М., 1984. 240 с.
13. *Каляев А. В.* Однородные коммутационные регистровые структуры. М., 1978.
14. *Алексенко А. Г.* и др. Проектирование радиоэлектронной аппаратуры на микропроцессорах. М., 1984. 272 с.
15. *Алексенко А. Г.* и др. МикроЭВМ и особенности их организации. М., 1982. 68 с.
16. *Пупырев Е. И.* Перестраиваемые автоматы и микропроцессорные системы. М., 1984. 192 с.
17. *Прангишвили И. В.* Микропроцессоры и локальные сети микроЭВМ в распределенных системах управления. М., 1985. 272 с.
18. *Баранов С. И., Скляров В. А.* Цифровые устройства на программируемых БИС с матричной структурой. М., 1986. 272 с.
19. *Конопелько В. К., Лосев В. В.* Надежное хранение информации в полупроводниковых запоминающих устройствах. М., 1986. 240 с.
20. *Левенталь Л.* Введение в микропроцессоры: Программное обеспечение, аппаратные средства, программирование. М., 1983. 464 с.
21. *Евреинов Э. В., Прангишвили И. В.* Цифровые автоматы с настраиваемой структурой. М., 1974.
22. *Глушков В. М.* Синтез конечных автоматов. М., 1962. 476 с.
23. *Макаров И. М.* Робототехника и гибкие автоматизированные производства. В 9-ти кн. М., 1986.
24. Полупроводниковые запоминающие устройства и их применение / Под ред. А. Ю. Гордонова. М., 1981. 344 с.
25. *Балашов Е. П., Смолов Б. В.* и др. Многофункциональные регулярные вычислительные структуры. М., 1978. 288 с.
26. *Артюхов В. Л.* и др. Настраиваемые модули для управляющих логических устройств. Л., 1981. 168 с.
27. *Горбатов В. А.* и др. Логическое управление технологическими процессами. М., 1978. 272 с.
28. *Горбатов В. А.* и др. Регулярные структуры автоматного управления. М., 1980. 216 с.
29. *Закревский А. Д.* Логический синтез каскадных схем. М., 1981. 416 с.
30. *Макаров И. М.* и др. Теория выбора и принятия решений. М., 1982.

31. *Майерс Т.* Надежность программного обеспечения. М., 1980. 360 с.
32. *Фет Я. И.* Параллельные процессоры для управляющих систем. — М., 1981. 160 с.
33. *Чернышев Ю. А., Аббакумов И. С.* Расчет и проектирование устройств ЭВМ с пассивным резервированием. М., 1979.
34. *Хоровиц П., Хилл У.* Искусство схемотехники. М., 1983.
35. *Шевкопляс Б. В.* Микропроцессорные структуры. Инженерные решения. М., 1986. 264 с.
36. *Яковлев Ю. С.* О распределении ресурсов памяти микроЭВМ. Киев, 1982. 47 с.
37. *Букреев И. Н.* Микроэлектронные схемы цифровых устройств. М., 1975. 368 с.
38. *Заренин Ю. Г., Стоянова А. И.* Определительные испытания на надежность. М., 1978. 168 с.
39. *Кошевой А. А.* Телеметрические комплексы летательных аппаратов. М., 1975. 312 с.
40. *Карповский Е. Я.* Надежность специального математического обеспечения управления. Киев, 1982. 150 с.
41. Требования и спецификации в разработке программ. М., 1984. 344 с.
42. *Боэм Б.* и др. Характеристики качества программного обеспечения. М., 1981. 208 с.
43. *Холстед М. Х.* Начала науки о программах. М., 1981. 128 с.
44. Микропроцессорные комплекты интегральных схем (состав и структура) / Под ред. А. А. Васенкова, В. А. Шахнова. М., 1982. 192 с.
45. *Лазарев В. Г.* и др. Построение программируемых управляющих устройств. М., 1984. 192 с.
46. *Лазарев В. Г., Пийль Е. И.* Синтез управляющих автоматов. М., 1978. 408 с.
47. *Якубайтис Э. А.* Логические автоматы и микромодули. Рига, 1975. 259 с.
48. *Мячев А. А., Иванов В. В.* Интерфейсы вычислительных систем на базе мини- и микроЭВМ. М., 1986. 248 с.
49. *Батищев Д. И.* Методы оптимального проектирования. М., 1984. 248 с.
50. *Пронин Е. Г., Могуева О. В.* Проектирование бортовых систем обмена информацией. М., 1989. 240 с.
51. *Пронин Е. Г., Шохет В. С.* Проектирование технических средств ЭВА. М., 1986. 192 с.
52. *Норенков И. П., Маничев В. Б.* Системы автоматизированного проектирования электронной вычислительной аппаратуры. М., 1983. 272 с.
53. *Гаврилов М. А., Девятков В. В., Пупырев Е. И.* Логическое проектирование дискретных автоматов. М., 1977. 352 с.
54. *Юдицкий С. А., Тагаевская А. А., Ефремова Т. К.* Проектирование дискретных систем автоматики. М., 1980. 232 с.
55. *Питерсон Дж.* Теория сетей Петри и моделирование систем. М., 1984. 264 с.
56. *Котов В. Е.* Сети Петри. М., 1984. 160 с.
57. *Слепцов А. И., Юрасов А. А.* Автоматизация проектирования управляющих систем гибких автоматизированных производств. Киев, 1986. 160 с.
58. *Горбатов В. А.* Теория частично упорядоченных систем. М., 1976. 336 с.
59. *Горбатов В. А.* Семантическая теория проектирования автоматов. М., 1979.
60. *Кристофидес Н.* Теория графов. Алгоритмический подход. М., 1978. 432 с.
61. *Мелихов А. Н.* Ориентированные графы и конечные автоматы. М., 1974.
62. *Нильсон Н.* Принципы искусственного интеллекта. М., 1985. 376 с.
63. *Клейнрок Л.* Вычислительные системы с очередями. М., 1979. 600 с.
64. *Виттих В. А., Цыбаков В. А.* Оптимизация бортовых систем сбора и обработки данных. М., 1985. 176 с.
65. *Капур К., Ламберсон Л.* Надежность и проектирование систем. М., 1980. 264 с.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

ОГЛАВЛЕНИЕ

Учебное издание

Арсеньев Юрий Николаевич,
Журавлев Владимир Михайлович

ПРОЕКТИРОВАНИЕ СИСТЕМ
ЛОГИЧЕСКОГО УПРАВЛЕНИЯ
НА МИКРОПРОЦЕССОРНЫХ СРЕДСТВАХ

Зав. ред. Н. И. Хрусталева. Редактор В. И. Милешин. Мл. редакторы Г. Г. Бучина, Л. А. Гусакова, Н. В. Панюшкина. Художник В. В. Гарбузов. Художественный редактор Г. М. Скворцова. Технические редакторы Н. А. Битюкова, С. В. Светикова. Корректор Г. И. Кострикова

ИБ № 8810

Изд. № СТД—692. Сдано в набор 05.03.91. Подп. в печать 25.07.91. Формат 60×88$^{1}/_{16}$. Бум. офс. № 2. Гарнитура литературная. Печать офсетная. Объем 19,60 усл. печ. л. 19,60 усл. кр.-отт. 21,55 уч.-изд. л. Тираж 14 000 экз. Заказ № 970. Цена 1 р. 80 к.

Издательство «Высшая школа», 101430, Москва, ГСП-4, Неглинная ул., д. 29/14.

Московская типография № 8 Государственного комитета СССР по печати 101898, Москва, Хохловский пер., 7.