# ПРИМЕНЕНИЕ МИКРОПРОЦЕССОРНЫХ СРЕДСТВ В СИСТЕМАХ ПЕРЕДАЧИ ИНФОРМАЦИИ

Допущено
Министерством высшего и среднего
специального образования СССР
в качестве учебного пособия
для студентов вузов,
обучающихся по специальности
«Автоматизированные системы управления»

МОСКВА «ВЫСШАЯ ШКОЛА» 1987

ББК 32.973.2
П76
УДК 681.32

Рецензенты:

кафедра автоматизированных систем управления Московского высшего технического училища им. Н. Э. Баумана (зав. кафедрой — д-р техн. наук, проф. В. Н. Четвериков); Б. М. Каган — д-р техн. наук, проф. (Московский институт инженеров железнодорожного транспорта)

П76 **Применение** микропроцессорных средств в системах передачи информации: Учеб. пособие для вузов по спец. АСУ/Б. Я. Советов, О. И. Кутузов, Ю. А. Головин, Ю. В. Аветов. — М.: Высш. шк., 1987. — 256 с.: ил.

В пособии рассмотрены особенности современных систем передачи информации, обусловливающие широкое применение микропроцессорных средств для реализации устройств передачи информации. Значительное внимание уделено специфическим вопросам системного и технического проектирования устройств систем передачи информации на базе микроЭВМ и микропроцессоров. Изложены вопросы программной реализации на базе микроЭВМ функций помехоустойчивого кодирования. Анализ проектных решений проведен с использованием аппарата сетей Петри и имитационного машинного моделирования.

П $\frac{2405000000-(4309000000)-211}{001(01)-87}$ 198—87

ББК 32.973.2
6Ф7.3

## СПИСОК СОКРАЩЕНИЙ

АЗО — алгоритм защиты от ошибок
АКД — аппаратура окончания канала данных
АМ — абонентская машина
АП — абонентский пункт
АПД — аппаратура передачи данных
БИС — большая интегральная схема
БВУВ — блок выработки управляющих воздействий
ВЗУ — внешнее запоминающее устройство
ВМ — вычислительный модуль
ВС — вычислительная система
ВЦ — вычислительный центр
ВЦКП — вычислительный центр коллективного пользования
ГВМ — главная ЭВМ
ГММ — гипотетическая микромашина
ДК — дискретный канал
ЗУ — запоминающее устройство
ИВС — информационно-вычислительная сеть
ИК — информационный кадр
КК — коммутация каналов
КМ — коммутационная машина
КНМЛ — кассетный накопитель на магнитной ленте
КП — коммутация пакетов
КС — канал связи
КЦ — концентратор
ЛК — логический канал
МККТТ — Международный консультативный комитет по телефонии и телеграфии
МОС — Международная организация по стандартизации
МП — микропроцессор
МПД — мультиплексор передачи данных
МПК — микропроцессорный комплект
МПС — микропроцессорное средство
МПУ — микропроцессорное устройство
МСА — матричная схема алгоритма
НГМД — накопитель на гибких магнитных дисках
ОЗО — оператор защиты от ошибок
ОЗУ — оперативное запоминающее устройство
ОКП — оконечный пункт
ООД — оконечное оборудование данных
ОПП — общее поле оперативной памяти
ОС — операционная система
ОСЦК — оборудование сопряжения с цифровыми каналами
ОУ — оконечная установка
ПА — периферийный адаптер
ПД — передача данных
ПДП — прямой доступ к памяти
ПЗУ — постоянное запоминающее устройство

ПК — программируемый кодек
ПО — программное обеспечение
ППЗУ — перепрограммируемое постоянное запоминающее устройство
ПРП — промежуточный пункт
РОК — редакционно-отладочный комплекс
РОН — регистр общего назначения
РМВ — реальный масштаб времени
РМО — рабочее место оператора
СА — связной адаптер
СМО — система массового обслуживания
СОИ — сеть обмена информацией
СОП — секция оперативной памяти
СП — связной процессор
СПД — сеть передачи данных
СПИ — система передачи информации
ТС — техническое средство
УВВ — устройство ввода — вывода
УЗО — устройство защиты от ошибок
УКН — устройство концентрации нагрузки
УУ — устройство управления
ФМ — функциональный модуль
ЦК — центр коммутации
ЦКК — центр коммутации каналов
ЦКП — центр коммутации пакетов
ЦКС — центр коммутации сообщений
ЧМСА — частичная матричная схема алгоритма

## ПРЕДИСЛОВИЕ

В настоящее время одним из важнейших направлений научно-технического прогресса является развитие микропроцессорной техники (микропроцессоров и микропроцессорных средств) и широкое ее применение в различных областях народного хозяйства, в частности в системах передачи информации (СПИ) автоматизированных систем управления (АСУ). Последнее объясняется тем, что сложность процессов передачи информации в АСУ, высокие и разнообразные требования, предъявляемые к параметрам аппаратуры передачи, делают особенно актуальным использование микропроцессоров и микроЭВМ, которые благодаря малым размерам, высокой надежности, развитым логическим возможностям, программной настройке на конкретное применение позволяют создавать высокоэффективные, функционально развитые устройства СПИ.

Использование микропроцессорных средств в СПИ требует от разработчиков коренного пересмотра традиционных методов проектирования и применения электронной аппаратуры, заменяя во многих случаях проектирование схем разработкой программ настройки микропроцессорной аппаратуры на выполнение определенных функций. Поэтому основное внимание в данном пособии уделено описанию и раскрытию этапов методики проектирования микропроцессорных устройств, программно реализующих функции СПИ в АСУ.

Цель пособия — углубить знания студентов, обучающихся по специальности 0646 — «Автоматизированные системы управления», в области методов и средств проектирования микропроцессорных устройств СПИ, используемых в АСУ.

Содержание пособия определено прежде всего стремлением авторов изложить те аспекты применения микропроцессорных средств в СПИ, которые слабо отражены в монографической литературе. В пособии представлен в основном оригинальный материал, являющийся следствием многолетней совместной работы авторов в данной

области. В пособии кратко изложены вопросы, посвященные особенностям современных СПИ и микропроцессорных средств.

Несмотря на то что авторы специально не ставили своей целью систематическое изложение вопросов автоматизированного проектирования микропроцессорных устройств СПИ, рассматриваемые в пособии решения задач, составляющих содержание основных этапов проектирования ориентированы на программную реализацию.

Авторы считают своим приятным долгом поблагодарить аспирантов и сотрудников кафедры «Автоматизированные системы обработки информации и управления» П. Ц. Антонова, А. В. Матвееву, Н. В. Луговую за помощь при написании некоторых параграфов пособия. Авторы глубоко признательны рецензентам — проф. Б. М. Кагану и коллективу кафедры АСУ Московского высшего технического училища им. Н. Э. Баумана (заведующий кафедрой проф. В. Н. Четвериков) за ряд ценных замечаний, сделанных при рецензировании рукописи.

*Авторы*

# ВВЕДЕНИЕ

Одной из характерных особенностей нашего времени является широкое применение ЭВМ в самых различных областях человеческой деятельности. В 60-х годах началось активное использование ЭВМ при решении задач народного хозяйства, и прежде всего в автоматизированных системах управления (АСУ) различного назначения.

Сегодня задачи широкой электронизации и комплексной автоматизации производства, разработка и массовое освоение современной компьютерной техники, постановка программного обеспечения ЭВМ и автоматизированных систем управления на индустриальную основу рассматриваются в партийных документах как одно из важнейших направлений ускорения научно-технического прогресса, его активной роли в деле эффективности и интенсификации всего общественного производства.

Необходимым условием функционирования любой АСУ является осуществление надежного и оперативного обмена информацией между ее звеньями. Причем если в первых АСУ обмен производился чаще всего буквально вручную, когда информация к ЭВМ доставлялась непосредственно самими пользователями (из цехов, складов, отделов и т. п.), то сегодня обмен в АСУ реализуется системами передачи информации (СПИ), представляющими собой сложные многоэлементные и многофункциональные инженерно-технические комплексы с высокой степенью автоматизации функций и пространственным рассредоточением своих компонент. Это объясняется не только созданием АСУ городского, регионального, республиканского, союзного и других уровней, в которых громадные массивы информации транспортируются на большие расстояния по каналам разного вида и назначения (телефонным, телеграфным, специальным, проводным, радиорелейным, стекловолоконным, КВ- и УКВ-радио, спутниковым и т. п.), но и постоянным и значительным повышением многообразия, сложности и оперативности решаемых АСУ задач.

Подавляющая часть всей информации АСУ представляется в дискретной форме. Это объясняется следующими основными причинами: во-первых, практически вся информация, циркулирующая в АСУ, вырабатывается и потребляется цифровыми вычислительными машинами, а следовательно, должна быть дискретной; во-вторых, значительно проще обрабатывать и защищать информацию, представленную в дискретной форме, чем в непрерывной, и, наконец, только дискретный вид представления информации в принципе позволяет унифицировать все многообразие видов связи, сведя их к единым методам передачи дискретной информации по дискретным каналам. О последнем свидетельствуют, в частности, активные разработки и внедрение систем дискретного (цифрового) радиовещания, дискретной звуко- и видеозаписи, дискретной телефонии, дискретного телевидения, интегральных систем передачи информации.

Таким образом, СПИ в современных АСУ представляют собой многоуровневые, сетевые, интегральные системы передачи информации, разработка которых является нелегкой задачей, требующей овладения как современными методами и средствами проектирования, так и последними достижениями микроэлектроники и вычислительной техники, к которым относятся, в частности, микропроцессоры (МП) и микропроцессорные средства (МПС). Благодаря своим замечательным свойствам МП и МПС открывают широкие возможности по использованию их как качественно новой элементной базы построения СПИ и АСУ, позволяющей создавать в первую очередь многофункциональные и высокоэффективные оконечные и промежуточные устройства СПИ, использующие кодонезависимые процедуры передачи данных, принципы пакетной передачи информации, стандарты и рекомендации Международной организации по стандартизации (МОС), Международного консультативного комитета по телефонии и телеграфии (МККТТ).

Вместе с тем специфика элементной базы в виде МП и МПС определяет программную реализацию устройств СПИ, которая в отличие от аппаратной вносит существенные особенности в процесс проектирования устройств СПИ как микропроцессорных систем, где необходимо учитывать, например, конкуренцию процессов за ограниченную вычислительную мощность, наличие нескольких асинхронных параллельных процессов, ограничения

на внутреннюю оперативную память, приоритетность обслуживания информационных потоков и пр.

В пособии пять глав, первая из которых посвящена изложению особенностей современных СПИ, связанных с сетевой идеологией их построения, и, как следствие этого, усложнением и расширением функций устройств передачи данных, обусловивших широкое применение микропроцессорных средств для их реализации. Здесь проанализированы также тенденции внедрения микропроцессорных средств как новой технологической базы при построении СПИ в АСУ.

Во второй главе представлены основные сведения по современным микропроцессорным средствам и их программному обеспечению. На основе общих понятий функциональной и физической структуры микропроцессорных устройств СПИ рассматриваются связные процессоры, программируемые абонентские пункты, концентраторы и мультиплексоры, центры коммутации.

В третьей главе раскрываются вопросы общей методики проектирования микропроцессорных устройств СПИ, характеризуются моделирующие возможности аппарата СМО, сетей Петри, машинного имитационного моделирования и их использование при анализе и синтезе устройств СПИ.

В четвертой главе рассматривается методика системного проектирования микропроцессорных устройств СПИ, позволяющая на основе использования аппарата сетей Петри и имитационного моделирования производить строго обоснованный выбор структуры устройства и системных параметров протокола обмена информацией между звеньями сетевой СПИ как основной функции устройства СПИ, а также методика технического проектирования как процесс получения работоспособного комплекса программ, обеспечивающих реализацию функций проектируемого микропроцессорного устройства СПИ.

В пятой главе в качестве примера рассмотрены вопросы программной реализации на базе микроЭВМ функции помехоустойчивого кодирования и декодирования сообщений как относительно наиболее простой, автономной и в то же время присущей каждому узлу СПИ.

# ГЛАВА 1
# ПРИЧИНЫ И ТЕНДЕНЦИИ ВНЕДРЕНИЯ МИКРОПРОЦЕССОРНЫХ СРЕДСТВ В СИСТЕМЫ ПЕРЕДАЧИ ИНФОРМАЦИИ

## § 1.1. ОБМЕН ИНФОРМАЦИЕЙ В АСУ

**Информация в системе управления.** В самом общем виде система, в которой реализуется автоматизированное управление, включает: внешнюю среду, объекты управления, управляющие объекты, устройство обработки информации и канал взаимодействия (рис. 1.1). Окружающая среда воздействует как на управляющие объекты (например, через директивные органы), задавая им нормативную модель (план), так и на объекты управления (через изменение условий функционирования), изменяя их состояние. Состояние объектов управления отображается контрольной информацией, которая, пройдя устройство обработки информации, поступает в управляющие объекты и в виде информационной модели объектов управления отображает их текущее состояние. В результате сопоставления нормативной и информационной моделей объектов управления вырабатывается рассогласование (сигнал ошибки) между текущим действительным состоянием объектов управления и тем, каким оно должно быть в соответствии с планом. На основе анализа сложившейся ситуации управляющие объекты принимают решение и вырабатывают воздействие, которое в виде управляющей (командной) информации поступает на объекты управления, переводя их в новое состояние.

Процесс управления является непрерывным циклическим. Один замкнутый цикл включает в себя следующие этапы: сбор от объектов управления информации состояния; обработку и преобразование информации состояния в информацию управления (формирование решений) и передачу информации управления. В результате выполнения распоряжений объекты изменяют свое состояние, что вызывает новый цикл процесса управления.

Таким образом, процесс управления состоит из сбора, обработки, преобразования и передачи информации, в результате чего происходит изменение состояний объектов управления. В соответствии с этим важнейшим условием осуществления управления является *обеспечение обмена информацией* между объектами системы управления.

Обмен информацией осуществляется через канал взаимодействия, который в общем случае реализуется в виде системы обмена информацией. Основу любой систе-

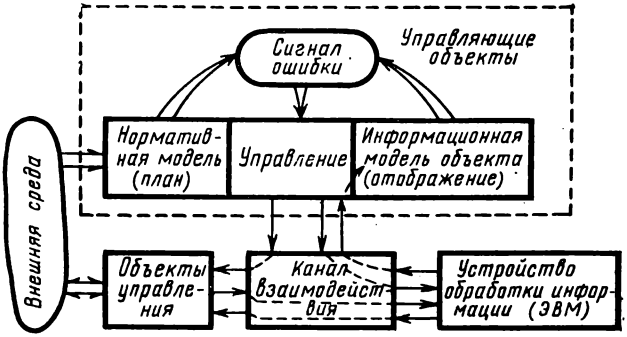



Рис. 1.1. Структура АСУ

мы обмена информацией составляет *система передачи информации* (СПИ).

**Требования системы управления к СПИ.** Информация, передаваемая в системе управления в виде сообщений, неоднородна по своему содержанию и может быть разбита по группам важности и срочности информации, каждая из которых характеризуется определенным уровнем требований к процессу передачи. В первом случае (важность) к процессу передачи информации предъявляют требования по надежности доставки, во втором (срочность) — требования по допустимой задержке.

По названным группам в АСУ различают сообщения оповещения и телеметрическую информацию, диалоговую и справочную информацию, общие донесения и распоряжения.

На рис. 1.2 приведены ориентировочные графики допустимой задержки ($T_{доп}$) информации, содержащей различные сведения, в зависимости от объемов этих сооб-

щений в битах. Соответствующие группы сообщений образуют категории срочности.

Со стороны требований к надежности передачи в общем случае допускают различные значения вероятностей ошибки и потери сообщения. Так, например, в системах оповещения о возможной «катастрофе» (под «катастрофой» обычно понимается событие практически недопустимое) системы доставки равнозначно должны обеспечивать практическую невозможность ошибочного приема или пропадания информации оповещения. В большинстве случаев значения допустимой ошибки и потери сообщений оповещения выбираются в пределах $10^{-9}$... ...$10^{-10}$.

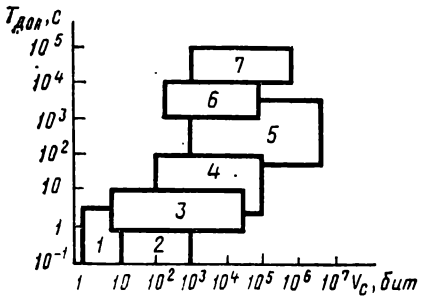



Рис. 1.2. Зависимость допустимого времени задержки $T_{доп}$ от объема сообщений $V_c$ разной категории:

*1* — оповещение; *2* — телеметрическая информация; *3* — диалог «человек — машина»; *4* — получение справочной информации; *5* — обмен программами между ЭВМ; *6* — общие донесения и распоряжения; *7* — обычная телеграфная информация

В то же время для телеметрических систем допустимые значения вероятностей ошибки и потери сообщения могут быть различными. Обычно более жесткими являются требования к достоверности передачи. Это объясняется тем, что при пропадании какого-либо из отсчетов измеряемого процесса последствия могут быть сглажены за счет информации соседних отсчетов. При появлении ошибки возможны резкий выброс отсчета и значительные изменения в действиях управляющего элемента. Опыт эксплуатации ряда телеметрических систем в АСУ технологическими процессами показывает, что граничные значения вероятностей потери и ошибки следует выбирать равными $10^{-3}$ и $10^{-5}$ соответственно.

Большинство систем широкого использования, обеспечивающих передачу диалоговой и справочной информации, вполне успешно функционирует при вероятности ошибки или потери сообщения порядка $10^{-6}$.

Под общими донесениями и распоряжениями обычно понимают смысловую информацию, которая передается между людьми при решении в системе управления задач,

не подлежащих автоматизации. В настоящее время имеется достаточно большой опыт в работе систем передачи такого рода информации, обеспечивающих вполне приемлемое качество функционирования систем управления при вероятностях потери или ошибки $10^{-5}$.

Таким образом, множество сообщений, передаваемых в АСУ, делится на подмножества — категории, причем к различным сообщениям, но относящимся к одной категории, со стороны системы управления предъявляются идентичные требования. Для того чтобы эти требования выполнялись, применяется регулирование сообщений по категориям. При этом сообщениям каждой из категорий присваивается определенный ранг преимущества (приоритет), в соответствии с которым происходит обслуживание сообщения, в частности в СПИ.

С другой стороны, на принципы построения СПИ существенное влияние оказывает структура системы управления, определяющая в процессе управления взаимосвязь множества управляющих объектов и объектов управления.

*Централизованная система управления* предполагает реализацию всех процессов управления в едином центральном управляющем органе, который осуществляет обработку информации, поступающей от всех объектов, об их состоянии. При выработке управляющей информации для каждого из элементов управления в централизованной структуре учитывается информация состояния всех объектов. По такому принципу, в частности, строятся системы управления предприятиями.

В системах управления *с децентрализованной структурой* для каждого объекта управления предусмотрен свой управляющий орган, с которым он обменивается информацией. Если при этом имеется единая цель управления, то управляющие органы в процессе выработки решений также могут использовать информацию о состоянии объектов управления в совокупности. По децентрализованному принципу построены, например, системы управления технологическими процессами.

Системы управления, имеющие *комбинированную структуру,* сочетают в себе черты централизованной и децентрализованной структур (например, системы управления промышленными объединениями).

В системах с *иерархической структурой функции управления* распределены между несколькими соподчиненными органами с одновременным соблюдением принци-

па централизации. Обмен информацией состояния в таких системах производится «снизу — вверх», а управляющей информацией — «сверху — вниз». Не исключается возможность передачи информации состояния и между элементами одного уровня. Характерными примерами указанных систем служат системы управления отраслью.

Система передачи информации, создаваемая в интересах системы управления, строится либо с учетом системы управления, либо независимо от нее. Следует учитывать, что в первом случае система передачи информации раскрывает структуру системы управления, определяя тем самым структуру системы обмена информацией АСУ.

**Разновидности систем передачи информации АСУ.** Системы передачи информации, применяемые в разнообразных АСУ, информационно-вычислительных и других системах, с точки зрения используемой техники и функционирования можно разделить на: локальные СПИ для автоматизированного управления технологическими процессами или предприятием, расположенным на небольшой территории (до 10...15 км$^2$); СПИ замкнутых АСУ с объектами, рассредоточенными на большой территории; СПИ вычислительных центров коллективного пользования (ВЦКП); глобальные СПИ; СПИ сетей ЭВМ; СПИ массового обслуживания (по продаже и бронированию мест на транспорте, в гостиницах, больницах и т. п.).

*Система передачи информации АСУ технологическими процессами* (ТП) состоит из автоматических датчиков, объекта управления, информационного канала (канала передачи сигналов сообщений) и устройства сопряжения с ЭВМ для обмена импульсными или аналоговыми сигналами, поступающими от датчиков объекта управления в управляющую ЭВМ и по цепи обратной связи к приводному исполнительному механизму, регулирующему режим работы технологического оборудования (объекта, станка, доменной печи и т. п.). СПИ АСУ ТП должны работать в реальном масштабе времени. В СПИ АСУП (цеха, отдела, склада, материально-технического снабжения, отдела подготовки производства и т. п.) движение информации начинается от составителя документа, который с помощью регистратора производства собирает и обобщает информацию, поступающую с рабочих мест, и переносит ее на технический носитель (перфокарту, перфоленту и т. п.). Носитель транспортируется на

вычислительный центр ВЦ или вводится в аппаратуру передачи данных АПД для дистанционной передачи информации на ВЦ по каналу связи. В СПИ АСУП всегда должны присутствовать средства двустороннего обмена информацией между человеком, находящимся у регистратора производства, и ЭВМ с целью обеспечения переспросов человека со стороны ЭВМ, если ЭВМ обнаружит ошибку или отсутствие каких-либо данных во вводимой в ЭВМ информации, а также переспроса человеком ЭВМ при обнаружении ошибки в полученных результатах вычислений. Необходимость в последнем бывает вследствие введения в ЭВМ ошибочных исходных данных или программ. Работа СПИ АСУП допускает переменное время задержки в передаче информации. Это позволяет использовать средства обмена информацией, работающие в полудуплексном режиме с ожиданием предоставления канала передачи данных ПД.

Как правило, СПИ АСУ ТП и АСУП работают по физическим парам кабелей внутрипроизводственной телефонной сети.

*СПИ замкнутых АСУ с объектами, рассредоточенными по большой территории,* отличается от предыдущих тем, что в ней используются не только физические пары кабелей, но и каналы многоканальных систем связи, а также часто применяются ВЦ двух или трех уровней (региональные, кустовые, главные). По выполняемым функциям и требованиям к ним эти системы разделяются на технологические и управленческие.

*СПИ коллективного пользования* создаются в тех случаях, когда различные отрасли производства зависят друг от друга и информация, циркулирующая в одной области, необходима для использования в другой или когда в одном районе сосредоточено много предприятий, для которых невыгодно иметь отдельные СПИ. Применение таких систем позволяет резко сократить затраты на вычислительные комплексы и СПИ за счет более полного использования ЭВМ и каналов ПД, а также время обработки данных путем применения более быстродействующих средств обмена информацией и ЭВМ. Кроме того, появляется возможность многократного использования информации, хранимой в обобщенной системе банков данных, в интересах всех объединяемых АСУ.

Абонентские пункты АП СПИ коллективного пользования могут работать в различных режимах обмена информацией с ВЦ (пакетном, диалоговом и справочном).

Наиболее крупными системами коллективного пользования являются общегосударственные СПИ, доступные для использования любыми ВЦ или АСУ.

*Глобальные (всемирные) системы* технически создаются по тем же принципам, что СПИ коллективного пользования, но только их центры коммутации распределены по всему земному шару. Примером такой сети является сеть, предназначенная для сбора и обмена метеорологической информацией. У этой сети главные ВЦ размещены в Москве, Вашингтоне, Сиднее, на каждый из которых замыкаются десятки кустовых ВЦ. Для межцентровых связей используются спутники связи и межконтинентальные кабели. Глобальные сети состоят из нескольких абонентских сетей и межцентровой сети.

*СПИ сетей ЭВМ* используются при поэтапном решении сложных задач и состоят из нескольких территориально распределенных ЭВМ и ВЦ и высокоскоростных каналов ПД. Между ЭВМ может предусматриваться обмен данными, программами и промежуточными результатами вычислений. Структура СПИ сетей ЭВМ зависит от ее назначения, географического расположения ЭВМ, требуемой надежности и «живучести». Размеры сети определяются возлагаемыми на нее задачами: она может включать сотни и даже тысячи ЭВМ, расположенных на большой территории, охватывающей несколько стран, или быть локальной и иметь всего несколько небольших ЭВМ, объединяемых одним моноканалом длиной до 1...2 км. Отличительной особенностью СПИ сетей ЭВМ является применение для обмена между ЭВМ высокоскоростных каналов ПД. Отметим, что локальные сети широко применяются в АСУ для научных исследований и экспериментов.

*Системы передачи информации массового обслуживания* (СПИ СМО) должны обеспечивать полуавтоматический обмен информацией между разнесенными по территории страны пультами операторов и ВЦ о наличии свободных мест, их бронировании, печатании билетов и т. п. Отличительной чертой СПИ СМО являются коллективное использование одного общего канала ПД для обмена информацией ВЦ с несколькими пультами операторов или справочными автоматами, расположенными в одном месте, работа в диалоговом режиме, а также обеспечение взаимодействия всех ВЦ, входящих в данную СМО. СПИ СМО работают в квазиреальном времени с задержкой ответа не более 0,1...0,5 мин.

**Формы обмена информацией в АСУ.** Обмен информацией в АСУ происходит непосредственно между людьми (в виде телефонных и телеграфных сообщений), между техническими устройствами, а также между техническими устройствами и человеком (в виде сообщений данных). Сообщение данных — это формализованная информация, закодированная по определенным правилам с целью обеспечения возможности ее обработки техническими средствами (ЭВМ).

Данные не предназначены непосредственно для человека как получателя информации. Осмысливание их человеком происходит только после соответствующей об-

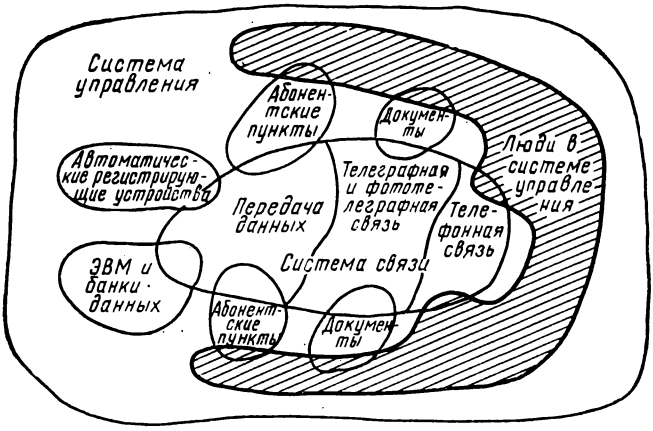



Рис. 1.3. Виды связи в АСУ

работки и представления в удобной для дальнейшего использования форме. Важной особенностью данных является то, что сообщения данных не имеют внутренней избыточности в отличие, например, от телефонных и телеграфных сообщений. Эта особенность, в частности, предъявляет повышенные требования к верности передачи данных.

На рис. 1.3 изображен принцип взаимодействия людей и устройств автоматизации в процессе управления на основе использования различных видов связи. При телефонной связи происходит процесс обмена информацией между людьми, приближенный к личному общению. Телеграфная связь также обеспечивает обмен информаци-

ей между людьми, но в этом случае информация предварительно оформляется в виде документов (телеграмм). При передаче данных операторы передают (получают) информацию через абонентские пункты, в которых происходит преобразование ее в данные и обратное преобразование.

Технические средства, являющиеся потребителями и источниками сообщений данных, могут быть разбиты на следующие группы:

1. *Автоматические регистрирующие датчики,* которые измеряют некоторую физическую величину и преобразуют результаты измерения в сообщения данных. Сюда же относятся устройства, обеспечивающие обратное преобразование данных в некоторую физическую величину.

2. *Абонентские пункты* (иногда их называют терминалами) предназначены для преобразования сформированной человеком информации в данные.

Существует много различных типов абонентских пунктов, отличающихся по сложности и своим возможностям. Простейшие абонентские пункты (АП) состоят из телеграфного аппарата и электрической пишущей машинки или специального устройства для считывания информации с промежуточных носителей, на которые ее заносит предварительно человек — оператор. Более сложные АП позволяют осуществлять ввод и вывод информации с помощью устройств отображения информации (дисплеев), что облегчает работу оператора по подготовке данных к передаче.

Существуют АП, обеспечивающие некоторые функции по предварительной обработке информации и сообщений (так называемые «интеллектуальные терминалы»). Роль и число таких АП постоянно возрастает в связи с широким применением в них микропроцессоров и микроЭВМ.

3. *ЭВМ и банки данных* осуществляют прием информации, ее обработку (решение задач), хранение, поиск и выдачу для передачи на любой АП по требованию оператора этого пункта или в результате автоматического запроса.

Все перечисленные технические средства автоматизации в АСУ разнесены на значительные расстояния, причем, как правило, необходимо обеспечить передачу данных между любыми двумя техническими средствами. Выполнение этой функции возлагается на систему передачи информации (систему связи), в которой создается специальная подсистема — сеть передачи данных.

Из рассмотрения различных автоматизированных систем с точки зрения обмена информацией можно сделать следующие выводы:

При создании автоматизированных систем управления решающее значение имеет электросвязь — область техники, занимающаяся вопросами передачи информации на расстояние с помощью электрического тока. Вычислительные машины и средства связи, являющиеся технической основой АСУ, взаимозависимы и все в большей степени влияют друг на друга.

Преобладающей и достаточно устойчивой тенденцией в области автоматической обработки информации стала телеобработка, которая представляет собой интеграцию «ЭВМ — связь» и в качестве составной части предусматривает передачу данных. С помощью сетей связи для передачи данных (сетей передачи данных) оконечные установки соединяются между собой так, что между ними можно производить обмен данными в форме дискретных (цифровых) сигналов.

Сети передачи данных в настоящее время создаются в виде вторичных сетей. Первичные сети представляют собой совокупность узлов и типовых каналов связи (каналов с нормированными характеристиками). Поэтому каналы связи как совокупность линейных и коммутационных устройств выступают в качестве заданных компонентов при проектировании сети данных.

Обычно сети передачи данных строятся и функционируют аналогично сетям телефонной связи, однако им свойствен целый ряд особенностей, которые в основном определяются высоким уровнем автоматизации процессов, обеспечивающих передачу информации.

## § 1.2. КАНАЛЫ И СЕТИ ПЕРЕДАЧИ ДАННЫХ В АСУ

Задачу доставки данных по заданному адресу с обеспечением соответствующих качественных показателей по скорости, верности и надежности выполняет сеть передачи данных (СПД), передающая электрические сигналы, несущие информацию. При создании СПД, как и любой другой сети связи, возникают две проблемы: 1) передача информации на расстояние, 2) распределение и доставка информации по заданным адресам. Первая проблема связана с созданием каналов для передачи данных, а вторая — с коммутацией этих каналов и сообщений.

Структурная схема канала передачи данных (дис-

кретных сообщений) приведена на рис. 1.4. Дискретная информация (сообщения) вырабатывается *источником сообщений* (ИС), в качестве которого могут выступать любые автоматические или ручные датчики (телеграфный аппарат, ЭВМ, трансмиттер). *Устройство сопряжения* (УС) согласует ИС с остальными частями системы по электрическим параметрам колебаний, скорости и коду. Устройство сопряжения является составной частью *оконечного оборудования данных* (ООД). *Устройство*

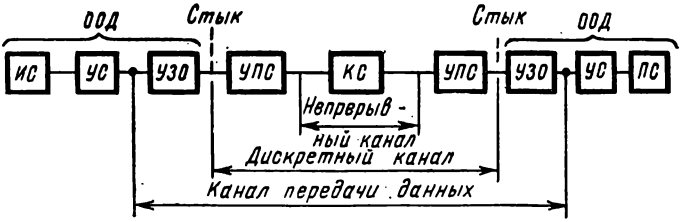



**Рис. 1.4. Структурная схема канала передачи данных**

*защиты от ошибок* (УЗО) обеспечивает защиту передаваемой информации с помощью помехоустойчивого кодирования. В *устройстве преобразования сигналов* (УПС) кодовым комбинациям сопоставляются сигналы, пригодные для распространения по заданным *каналам связи* (КС). На приемной стороне названные устройства следуют в обратном порядке, осуществляя обратные преобразования сигналов к элементам алфавита сообщения, выдаваемого получателю сообщения (ПС).

В качестве КС используют как физические цепи, так и каналы, образованные путем частотного и временно́го уплотнения физических цепей. Многие КС являются типовыми со строго нормированными параметрами. Поэтому при проектировании СПИ для АСУ, как правило, КС является заданной компонентой будущей системы. В общей абстрактной схеме СПИ КС выступает в роли непрерывного канала.

Большинство типовых КС (например, телефонный канал тональных частот, широкополосные групповые каналы) для непосредственной передачи по ним сигналов данных оказываются непригодными. Требуется дополнительное преобразование сигналов данных перед вводом в типовой КС. Эта операция и осуществляется с помощью УПС, получивших в системах передачи данных по теле-

фонным каналам название *модемов*. Соответствующее обратное преобразование осуществляется на выходе КС. За последние годы разработан унифицированный ряд модемов, и задачей проектировщика СПИ в АСУ является выбор подходящего модема из имеющихся стандартных конструкций.

Выбранный модем совместно с заданным КС определяет так называемый *дискретный канал* (ДК), свойства которого четко определены и изменению в процессе проектирования не подлежат.

Действия помех в КС приводят к искажениям передаваемых сигналов. Уровень искажений на выходе типового КС и соответственно на выходе ДК часто оказывается неудовлетворительным для передачи данных. В этом случае линия передачи сообщений данных как раз и дополняется УЗО.

Наряду с функцией помехоустойчивого кодирования — декодирования УЗО обеспечивает формирование кадра и режим работы (с обратной связью, без обратной связи). Это логические функции, допускающие как аппаратную, так и программную реализацию. Точки входа — выхода в логические модули, реализуемые УЗО, образуют канал, получивший название *канала передачи данных*.

Первоначально УЗО объединялось с УПС в *аппаратуру передачи данных* (АПД), устанавливаемую у абонента (пользователя). В современных системах эта задача решается в тесной связи с управлением передачей данных и возлагается на *оконечное оборудование данных* (ООД). АПД выполняет функции сопряжения ООД с КС. Помимо преобразования сигналов, поступающих от ООД в одну из форм, пригодных для передачи по КС, и обратного преобразования, АПД преобразует команды управления ООД о начале и об окончании связи в соответствующие сигналы, передаваемые через КС к другому ООД.

Отметим, что в терминах МККТТ эту аппаратуру, выполняющую отмеченные функции сопряжения ООД с КС, называют *аппаратурой окончания канала данных* (АКД).

Назначение и электрические свойства сигналов, а также физические свойства соединений между ООД и АПД унифицированы в рамках международных и национальных норм и *рекомендаций по стыкам*. Стыки ООД и АПД различных видов регламентируются стан-

дартами на АПД. Эти регламентации имеют целью, с одной стороны, сделать ООД возможно более независимым от типа АПД и вида КС, а с другой стороны — сохранить и независимость АПД от типа подключенного ООД. На рис. 1.5 показан пример стыка между ООД и АПД. В зависимости от типа АПД количество цепей соединений и их управление, связанное с режимом ПД, могут быть различными.

Таким образом, стык обеспечивает физический интерфейс между ООД и АПД. В специальной литературе совокупность технических средств, находящихся между точками физического интерфейса, часто называют *физическим каналом.*

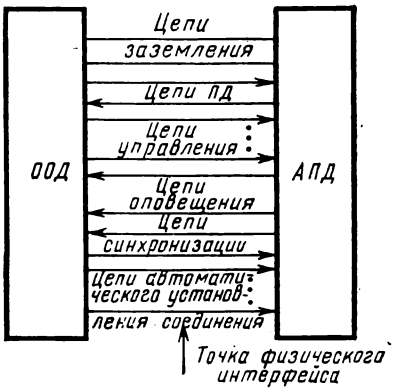



Рис. 1.5. Пример стыка между ООД и АПД

Несмотря на многообразие возможных СПИ самого различного назначения, все они имеют в принципе сходную структуру. В простейшем случае система связывает две оконечные установки (ОУ), в более сложном случае это сеть, связывающая множество ОУ. Каждый из видов соединения может существовать постоянно или временно. Поток информации через эти соединения направлен в одну сторону, попеременно или является двусторонним; при этом ОУ ведут обмен сообщениями соответственно в симплексном, полудуплексном или дуплексном режиме.

Совокупность территориально разнесенных ОУ, соединенных между собой каналами передачи, вместе с алгоритмами и программами обмена информацией образует *сеть обмена информацией* (СОИ).

В общем случае физическую структуру СОИ можно представить как совокупность ОУ, концентраторов (КЦ), центров коммутации (ЦК) и соединяющих их КС. Выделяют две структурные части СОИ: *базовую сеть* ПД (магистральную, коммуникационную), которая охватывает ЦК и соединяющие их магистральные КС, и *абонентскую (терминальную) сеть,* которая обеспечивает

подключение ОУ абонентов через КЦ или непосредственно к ресурсам базовой сети (рис. 1.6).

В качестве ОУ в СОИ выступают и абонентские пункты, устанавливаемые у пользователей, как отдельные, так и объединенные в терминальные комплексы, и рабочие ЭВМ, и целые вычислительные системы (ВС). Рабочие ЭВМ и ВС для освобождения их от связных функций часто снабжаются устройствами сопряжения с КС. С точки зрения связи все объекты, подключенные к ба-

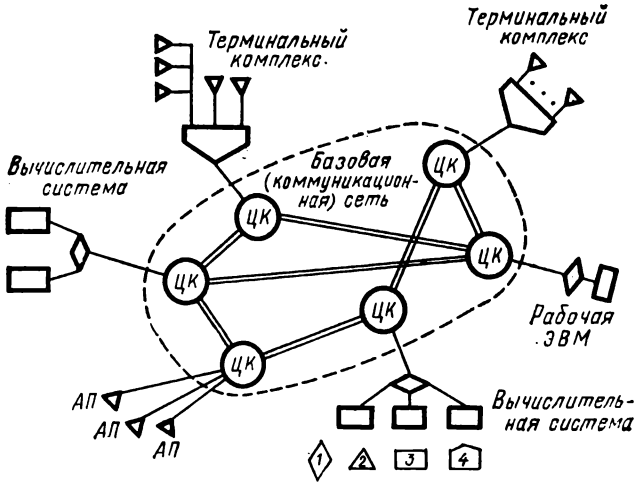



Рис. 1.6. Физическая структура СОИ:

*1* — устройство сопряжения; *2* — абонентский пункт; *3* — рабочая ЭВМ; *4* — концентратор; = — магистральный КС; — — абонентский КС

зовой сети ПД, являются «терминалами». Используя «связной» термин, такие объекты называют *абонентами.*

Рабочие ЭВМ, подсоединенные к базовой сети, часто называют *главными ЭВМ* (ГВМ) или *хост-машинами* (термин заимствован из сети ARPA и получил широкое распространение в переводной литературе). Устройство сопряжения может реализовывать и функции мультиплексора, чтобы позволить ГВМ выполнять одновременно несколько диалогов. Терминалы, которые просто передают и принимают потоки байтов и не имеют стандартного интерфейса с базовой сетью ПД, подключаются к базовой сети через «терминальный» концентратор, обеспечи-

вающий взаимодействие таких терминалов с базовой сетью ПД. Кроме того, основным назначением как концентратора, так и мультиплексора, является повышение эффективности использования соединительных линий как между ОУ и ЦК, так и между ЦК.

При организации связи через базовую сеть различают два вида соединений: *оперативные* соединения (коммутация каналов, сообщений, пакетов) и *долговременные* (кроссовые соединения каналов и линий).

По некоммутируемым и коммутируемым сетям может передаваться информация, не допускающая задержки в процессе ее передачи, например при телефонной связи, и допускающая задержку, например при передаче телеграфом. В первом случае для передачи информации между абонентами сети должен быть предварительно скроссирован или скоммутирован прямой (сквозной) канал, как это делается, например, на телефонной автоматической станции. Такие сети называют *сетями с коммутацией каналов,* а узлы сети — *центрами коммутации каналов* (ЦКК). Во втором случае такой канал может и не составляться. Подлежащее передаче сообщение с приписанным ему адресом передается в запоминающее устройство (ЗУ) узла. После анализа адреса сообщения и выбора направления передачи это сообщение передается в ЗУ соседнего узла и т. д., до тех пор, пока сообщение не поступит в ЗУ узла, к которому подключен адресат. Сети с таким способом распределения информации называются *сетями с коммутацией сообщений,* а узлы такой сети — *центрами коммутации сообщений* (ЦКС). Коммутация сообщений обеспечивает более эффективное использование соединительных линий, но стоимость коммутационного оборудования выше, чем в сетях с коммутацией каналов.

В сетях с коммутацией сообщений стремятся к весьма большой загрузке линий ($\leqslant$0,8 Эрл.). В таких условиях, поскольку длина сообщений не ограничивается, время ожидания передачи в ЦКС может быть весьма значительно (до десятка секунд). В диалоговых системах и системах типа «запрос—ответ» это воспринимается как нежелательное и мешающее нормальной работе явление. Поэтому в сетях с коммутацией сообщений применяют специальный режим — режим пакетной коммутации, при котором время ожидания короче.

В *сети с пакетной коммутацией* сообщения распределяются на части, называемые пакетами. Каждый пакет

содержит собственный заголовок с управляющим и адресным полями, а также собственную проверочную последовательность знаков. Разложение сообщения на пакеты и восстановление его после передачи в случае использования пакетного ООД осуществляется самим этим оборудованием, а при стартстопном и синхронном ООД — в сети с помощью специальных пакетирующих и депакетирующих аппаратных и программных средств. В сети пакеты в соответствии с правилом пакетной коммутации передаются от одного *центра коммутации пакетов* (ЦКП) к другому, причем для ЦКП нагрузку от каждого пакета можно примерно приравнять нагрузке от одного вызова в сети с коммутацией каналов. Пакеты могут вводиться или выводиться из сети со скоростью, отвечающей требованиям терминалов; таким образом, сеть действует как преобразователь скоростей.

Передача информации по сети с коммутацией пакетов (КП) возможна в режимах *датаграммном, виртуального вызова* или *постоянного виртуального канала.*

При виртуальном вызове различают три фазы сеанса связи: установления соединения, передачи данных и разъединения. В фазе *установления соединения* на основе посланного вызывающим абонентом адресного пакета («пакета набора») между вызывающим и вызываемым абонентами организуется «виртуальное соединение». Коммутационные узлы, к которым непосредственно подключены линии вызывающего и вызываемого абонентов, резервируют ячейки памяти для числа пакетов, указанного в адресном пакете.

В фазе *передачи данных* пакеты, поступающие от передающего ООД или устройства пакетирования, заносятся в резервированные ячейки памяти. Для передачи отдельных пакетов каждого сообщения по сети могут выбираться различные пути, что обеспечивает более гибкое и оперативное приспособление к состояниям занятости тех или иных КС и ЦК, а могут быть и фиксированные маршруты. В том случае, если пакеты из-за разных путей прохождения в сети поступили на ЦК пункта назначения в неверной последовательности, перед выдачей их на принимающее ООД или соответственно устройство депакетирования они располагаются (сортируются) в надлежащем порядке.

В фазе *разъединения* соединения резервирование ячеек памяти, произведенное в ЦК в фазе установления соединения, снимается.

В режиме постоянного виртуального канала между партнерами по связи организуется постоянный канал (долговременное виртуальное соединение). Характеристики передачи по нему совпадают с характеристиками фазы ПД при виртуальном вызове; фазы установления и разъединения отсутствуют.

При датаграммном режиме, который возможен при обмене информации только между пакетными ООД, фазы установления и разъединения отсутствуют. В сети с КП каждый из поступающих от ООД пакетов направляется на принимающее ООД непосредственно на основе адреса, содержащегося в заголовке пакета. При этом по сети передаются только одиночные пакеты даже тогда, когда выдаваемое ООД сообщение превышает максимальную длину пакета. После прохождения по сети разными путями с различной задержкой во времени пакеты сортируются на приемке перед их обработкой самим ООД.

Таким образом, точки стыка ЦКП с ОУ и КЦ образуют точки входов в базовую сеть ПД, в которых принимаются или выдаются пакеты, без разборки (сборки) передаваемые через всю сеть. Вся сеть ПД должна быть «прозрачной», т. е. передавать пакеты, данные в которых могут быть закодированы любым способом.

При характерной для пакетов малой длине коротким становится время ожидания на ЦК. Поэтому число битов, которые накапливаются перед линией связи, невелико, и в ЦК пакетов запоминающее устройство может иметь меньшую емкость, чем в ЦКС. Кроме того, поскольку ЦК может передать каждый пакет далее сразу же после его получения, не ожидая приема всего сообщения целиком, то время прохождения сообщений в сети с КП оказывается меньшим, чем в сети с коммутацией сообщений.

В современных СОИ наибольшее распространение получил метод коммутации пакетов. На их основе создаются цифровые сети интегрального обслуживания, обеспечивающие переход от раздельных сетей передачи речи и данных в режиме коммутации каналов и коммутации пакетов к совмещению этих режимов в единой сети. Особенностью таких сетей является разнородность трафика: диалог пользователя с машиной, речевая информация, обмен информационными массивами (файлами).

## § 1.3. УПРАВЛЕНИЕ СИСТЕМАМИ ПЕРЕДАЧИ ИНФОРМАЦИИ И ПРОТОКОЛЫ

С позиций современных задач автоматизации связи и в ходе сеанса связи могут быть выделены пять важных этапов: 1) установление соединения; 2) установление тракта передачи информации; 3) передача сообщения или разговор; 4) завершение связи; 5) разъединение. При автоматизации телефонной связи были переведены на автоматическую работу и регламентированы лишь этапы 1 и 5. Этапы же 2, 3 и 4 полностью выполняются абонентами.

В ходе дальнейшего развития электросвязи, когда в качестве отправителей и получателей сообщений все чаще используются ЭВМ и другие автоматические устройства, приходится автоматизировать и регламентировать также этапы 2, 3, 4.

Организация системы обмена информацией предполагает сопряжение всех входящих в систему элементов. Большие трудности вызывает сопряжение ОУ, связанное с автоматизацией всех пяти указанных выше этапов осуществления связи.

Говоря о сопряжении устройств связи, необходимо различать *уровни* этого *сопряжения. На низшем уровне* естественным и необходимым является *физическое сопряжение.* Оно включает *механическое* (установление числа проводов, формы и размеров разъемов) и *электрическое* (установление уровней напряжений в проводах и других характеристик сигналов). На следующем уровне необходимо *функциональное сопряжение,* определяющее сигналы, которые могут появляться в проводах, и их значения. Многочисленные примеры названных сопряжений можно найти в сетях ПД, где важнейшим объектом стандартизации является сопряжение ООД с оконечным оборудованием дискретного КС (см. рис. 1.4 и 1.5).

Дальнейшее развитие автоматизации системы связи потребовало применения логического и процедурного сопряжений. *Логическое сопряжение* связано с установлением способов представления информации в виде конкретных форматов, используемых для управления передачей сообщений. *Процедурное сопряжение* устанавливает значение различных форматов и порядок их использования для управления процессом связи. Логическое и процедурное сопряжения образуют протокол обмена информации.

Если физическое и функциональное сопряжения определяются выбором технических средств, то протоколы носят весьма сложный характер и являются предметом теоретических исследований. Протокол для конкретной системы распределения (обмена) информации зависит от того, для чего служит эта система (внешний фактор) и каковы ее характеристики (внутренний фактор). Например, в технике цифровой связи ЭВМ протокол представляет собой набор правил установления связи между ОУ, которыми могут быть системы файлов, терминалы, абонентские ЭВМ, узлы КП и др. Если имеется два каких-либо оконечных устройства $A_1$ и $B_2$ (первого уровня), связывающиеся между собой с помощью системы связи $C_1$, то можно говорить о протоколе первого уровня, устанавливающем правила связи двух указанных оконечных устройств (рис. 1.7). Полученная таким образом система первого уровня может рассматриваться как система передачи $C_2$ (второго уровня) для некоторых новых оконечных устройств $A_2$ и $B_2$. Система уровня $C_3$ ($A_2$—$C_2$—$B_2$) может служить для связи новых оконечных устройств $A_3$ и $B_3$ с использованием низших уровней по новым правилам и т. д. Так возникает понятие иерархии протоколов (см. рис. 1.4: непрерывный канал $C_1$, дискретный канал $C_2$, канал передачи данных $C_3$).

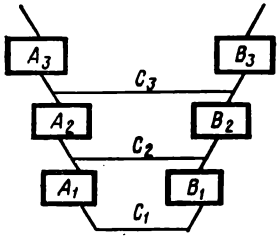


Рис. 1.7. Уровни системы связи

Как уже отмечалось, отличительной особенностью современных СПИ является их объединение с ЭВМ, что привело к созданию сетей различного назначения. Поэтому для разработчика средств транспортировки сообщений, т. е. СПИ, особый интерес представляет *иерархический подход к модели вычислительной сети.* Этот подход в настоящее время принят практически повсеместно, хотя предлагаемые архитектуры часто различаются как по числу уровней, так и по перечню функций, выполняемых в пределах уровня. Под архитектурой понимают совокупность логической, физической и программной структур сети.

Основным логическим понятием модели вычислительной сети является с и с т е м а — иерархическая группа функций, реализуемых в одной либо нескольких ЭВМ,

предназначенная для решения определенных сетевых задач. Системы связываются друг с другом условными линиями, именуемыми *физическими соединениями,* образуя логическую структуру вычислительной сети.

Все системы делятся на ряд слоев (рис. 1.8), называемых *уровнями,* каждый из которых выполняет в вычислительной сети определенную функциональную задачу. Правила взаимодействия смежных уровней одной и той же системы (например, $A$, или $B$, или $W$) именуются *межуровневым интерфейсом*, а взаимодействие объектов одноименных (одинаковых) уровней различных систем (например, уровня $n$ системы $A$ и уровня $n$ системы $B$)

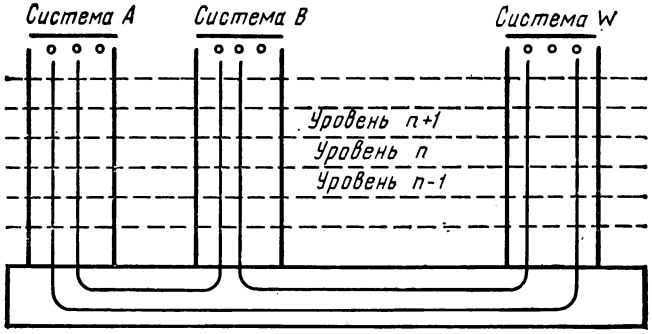



Рис. 1.8. Схема взаимодействия прикладных программ вычислительной сети:

О — прикладная программа; — — линия взаимодействия прикладных программ

задается соответствующим протоколом. *Протоколом* называют набор соглашений о форматах и порядке следования во времени обмениваемых сообщений.

Решение разнообразных сетевых задач обеспечивается специализацией систем, реализуемых абонентскими и коммуникационными машинами.

Термин «*абонентская машина*» (АМ) используют для обозначения любого программируемого устройства, способного подключаться к базовой сети. При этом не делают различия между АМ, представляющей вычислительные ресурсы (ГВМ), и АМ, работающей в качестве концентратора терминалов. Базовая сеть ПД состоит из *коммуникационных машин* (КМ), связанных друг с другом физическими соединениями. КМ обеспечивает маршрутизацию потоков информации, передаваемой между

АМ. Базовая сеть ПД предоставляет всем подключающимся к ней АМ средства доступа к ресурсам друг друга.

Каждая АМ фактически осуществляет ряд отдельных процессов, являющихся источниками и получателями информации, передаваемой по сети. Эти процессы не различаются базовой сетью ПД, но система, включающая АМ и сеть ПД, может рассматриваться как средство связи между процессами и называется *программой управления сетью*. АМ может вести несколько процессов, и задача программы управления сетью состоит в том, чтобы объединить сообщения от нескольких процессов в одной линии и затем распределить их по процессам на приемном конце. Программа управления сетью решает и другие задачи, направленные на повышение качества передачи между процессами.

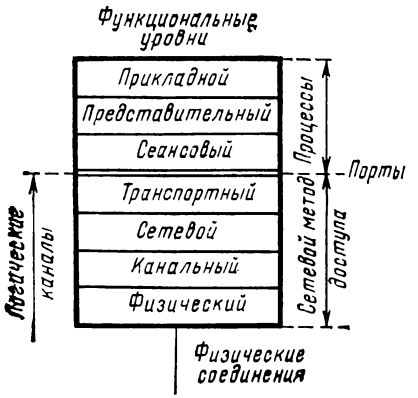



Рис. 1.9. Функциональные уровни модели вычислительной сети МОС

В рассматриваемой Международной организацией по стандартизации модели вычислительной сети имеется семь функциональных уровней (рис. 1.9), в соответствии с которыми строится программная структура, реализуемая абонентскими и коммуникационными станциями сети. Функции прикладного уровня связаны с выполнением вычислительных, информационно-поисковых либо справочных работ с логическим преобразованием информации пользователей. Представительный уровень обеспечивает генерацию и интерпретацию команд взаимодействия процессов, представление данных программе пользователя. Сеансовый уровень осуществляет интерфейс с транспортным уровнем, организацию, поддержание и окончание сеансов связи. Эти три высших уровня образуют область обработки данных. Объекты этой области (процессы) используют сервис по транспортировке данных, обеспечиваемой областью транспортировки данных, которую образуют четыре нижних (первых) уровня.

С точки зрения описания СПИ как транспортного средства можно достаточно наглядно представить СОИ пятиуровневой системой протоколов.

Совокупность уровней, не относящихся непосредственно к процессам передачи сообщений на сети и образующих область обработки данных, условно объединяют и называют верхним (прикладным) уровнем. Протокол «процесс—процесс» верхнего уровня устанавливает связь между процессами пользователя и обеспечивает управление доставкой сообщений по видам связи (диалог пользователя с ГВМ, обмен информационными массивами, речевая информация и др.).

Протокол «процесс — процесс» базируется на понятии логического канала ЛК, что позволяет не учитывать конкретную организацию физического канала и программ, обеспечивающих непосредственно передачу информации между процессами в АМ. Вход в процесс осуществляется через логические программно организованные порты, расположенные между процессами и уровнем управления передачей (транспортной областью). Условную линию, проведенную между портами различных АМ, принято называть *логическим каналом* (ЛМ). Он связывает пару удаленных портов и идентифицирует поток сообщений между процессами.

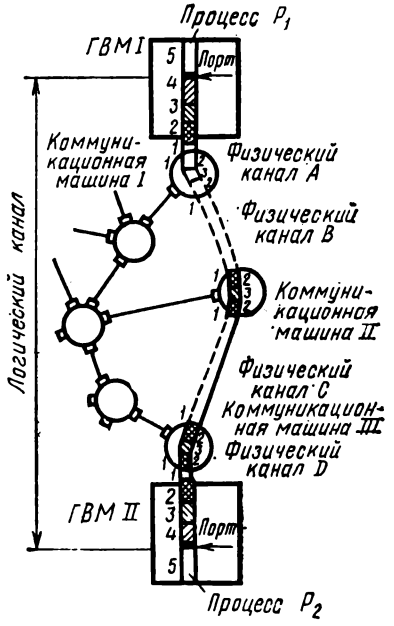



Рис. 1.10. Логический канал между процессами $P_1$ и $P_2$

На рис. 1.10 показан один из ЛК, связывающий процесс $P_1$ в ГВМI с процессом $P_2$ в ГВМII. Этот канал проходит от порта в ГВМI через коммуникационные машины и физические каналы к порту ГВМII. Следует иметь в виду, что во время передачи информации в зависимости от загрузки и состояния коммуникационной сети ЛК мо-

жет проходить не только по маршруту, показанному на рис. 1.10, но и по другим возможным в этой сети маршрутам. Одновременно в вычислительной сети может существовать (как постоянно, так и временно) множество различных логических каналов. При временном существовании логический канал создается между парой процессов на время обмена информацией между ними. После окончания сеанса связи ЛК ликвидируется.

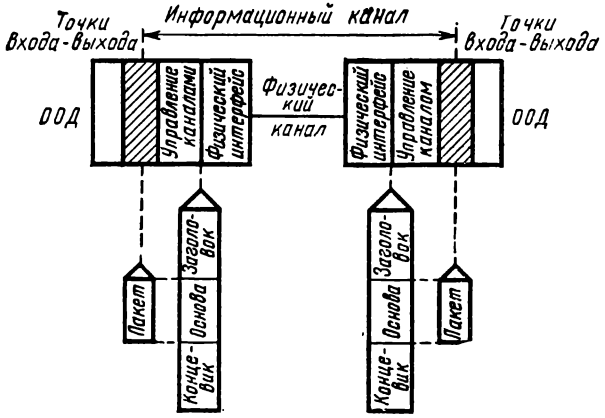



Рис. 1.11. Структура информационного канала

К основным функциональным уровням сети ПД в соответствии с последними рекомендациями относят уровни 1, 2, 3 и 4.

Протоколы 1-го уровня описывают физические, электрические и процедурные характеристики установления, поддержания и разъединения физического интерфейса в точке между ООД и АКД. Протоколы физического интерфейса представлены в рекомендациях МККТТ *X*.20, *X*.20бис, *X*.21, *X*.21бис.

Протоколы 2-го уровня регламентируют управление информационным каналом (каналом передачи данных). *Информационный канал* — это логическая система, предназначенная для передачи информации между двумя смежными сетевыми объектами и состоящая из двух связанных физическим каналом логических модулей, каждый из которых включает программы управления каналом и физический интерфейс (рис. 1.11). Функциями управления информационным каналом являются защита от оши-

бок, контроль дискретных каналов и мультиплексирование передаваемых блоков информации. Обмен данными по дискретному каналу осуществляется посредством организованных последовательностей битов — кадров. Кадры имеют три части: заголовок, основу (текст) и концевик.

Существует два вида протоколов управления информационным каналом: байт-ориентированный и бит-ориентированный. В *байт-ориентированных* протоколах управления информационным каналом осуществляется побайтовая (посимвольная) передача информации, а в *бит-ориентированных* протоколах передается неделимый на байты поток битов.

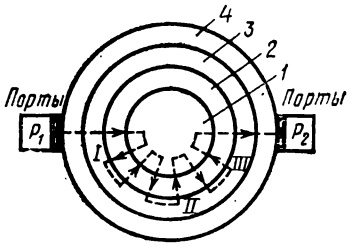



Рис. 1.12. Схема движения информации через слои программной структуры при передаче по логическому каналу:

*1* — физический уровень; *2* — канальный уровень; *3* — сетевой уровень; *4* — транспортный уровень

Наиболее распространенным байт-ориентированным протоколом управления информационным каналом является протокол BSC (Binary Sunchronous Communication). Этот протокол применяется во многих вычислительных сетях и является международным стандартом. На его основе разработан двоично-синхронный метод управления информационным каналом в ЕС ЭВМ. Протокол BSC определяет синхронную полудуплексную передачу информации по физическому каналу, соединяющему две станции ПД.

Наиболее популярным бит-ориентированным протоколом является протокол HDLC, который более подробно будет рассмотрен в гл. 4.

Протоколы 3-го уровня (сетевой уровень) регламентируют управление передачей в сети и включают функции, обеспечивающие: приоритетное обслуживание сообщений, выбор маршрутов передачи, межузловой обмен, локальное управление потоками, доступ АМ в базовую сеть ПД и различные сервисные функции.

Протоколом 4-го (транспортного) уровня — протокол управления передачей — определяется доставка сообщений по ЛК. Транспортный протокол выполняет следующие функции: адресование сообщений (поиск места подключения требуемого абонента к сети), контроль ошибок, потерь сообщений и их задержек, восстановления

сообщений на приеме и мультиплексирование сообщений для передачи. Протоколы так же, как и функциональные уровни, должны быть независимы друг от друга, что позволяет изменять любой из них, не затрагивая остальных.

Таким образом, передача информации от одного порта (процесса) к другому связана с многократным ее прохождением через различные слои программной структуры сети. Например, при передаче информации по ЛК от процесса $P_1$ к процессу $P_2$ (см. рис. 1.10) проход через


Рис. 1.13. Слоевая структура сообщений

слои программной структуры осуществляется так, как показано на рис. 1.12.

Многослойный характер структуры сети приводит к *слоевой структуре сообщений* (рис. 1.13). На каждом уровне к сообщению присоединяется заголовок, связанный с выполнением протокола на этом уровне и включающий, например, порядковую нумерацию, информацию для защиты от ошибок, идентификатор источника. По мере прохождения сообщения в направлении нижних уровней к нему присоединяется все больше информации. Когда сообщение на удаленном конце начинает подниматься от уровня к уровню, информация в заголовке и окончании отбрасывается и вычерчивается, а данные передаются следующему уровню. При этом протоколом каждого уровня фиксируется формат передаваемых на данном уровне управляющих и информационных сообщений.

В настоящее время во многих странах создаются и эксплуатируются сети ПД общего пользования с КП, имеющие многоуровневое сопряжение с ООД в соответствии с рекомендацией МККТТ *X*.25, состоящей из трех частей. Первая часть *X*.25/1 включает рекомендации *X*.21 и *X*.21бис, содержащие описание процедур и цепей стыка 1-го уровня и протоколов 2-го и 3-го уровней. Вторая часть *X*.25/2 определяет процедуры и форматы протоко-

ла информационного канала (HDLC). Третья часть X.25/3 содержит описание процедур и форматов, регламентирующих пакетный доступ в сеть ПД с КП.

Таким образом, в общем случае СОИ АСУ включает ряд технических комплексов:

1. Комплекс средств, обеспечивающих образование каналов ПД на основе каналов первичной сети. Он реализуется в виде совокупности отдельных образцов аппаратуры передачи данных, допускающих как чисто аппаратурную, так и программно-аппаратурную реализацию;

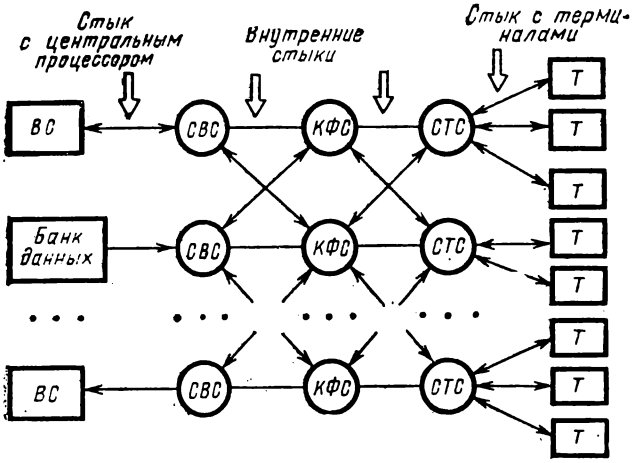



Рис. 1.14. Абстрактная схема СОИ

2. Комплекс технических средств, обеспечивающий целенаправленную передачу сообщений между абонентами сети при выполнении требований АСУ к вероятностно-временным характеристикам задержки. Этот комплекс реализуется как совокупность центров коммутации (каналов, сообщений, пакетов) вместе с их программным обеспечением;

3. Комплекс средств сопряжения, представляющий собой совокупность устройств, алгоритмов и программ, обеспечивающих физическое, логическое и процедурное согласования различных элементов сети ПД, а также элементов сети с техническими средствами источников и потребителей информации;

4. Комплекс средств контроля состояния технических средств и управления сетью ПД, представляющий собой

совокупность организационных и технических служб, а также технических и программных средств, обеспечивающих функционирование сети ПД в изменяющихся условиях.

Элементы перечисленных комплексов рассредоточены в сети и условно могут быть объединены в *проблемно-ориентированные модули* (рис. 1.14), каждый из которых выполняет строго определенные задачи по передаче данных и взаимодействию с другими модулями, вычислительной системой, банком данных и терминалами.

Модуль связи вычислительной системы (или банка данных) с сетью (СВС) осуществляет взаимодействие между разнородными ЭВМ и сетью ПД. Модуль связи терминала с сетью (СТС) обеспечивает взаимодействие между различными группами терминалов и другими элементами сети. Модуль коммуникационных функций (КФС) распределяет потоки сообщений по каналам первичной сети. Технические и программные средства модуля вместе с их взаимосвязями образуют *архитектуру модуля*, определяющим для которой является реализованный в сети способ коммутации. Для реализации части функций модуля, выполняемых программными методами, широко используются специализированные или универсальные мини- и микроЭВМ.

## § 1.4. МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА КАК НОВАЯ ТЕХНОЛОГИЧЕСКАЯ БАЗА ПОСТРОЕНИЯ СПИ

**Переход от аппаратной к программной реализации связных функций.** Развитие средств связи в первой половине нашего века шло практически раздельно от эволюции вычислительных средств. Для коммутации в телефонных сетях начиная с 1892 г. и по 30-е годы XX в. использовались электромеханические искатели. Затем начали применять искатели на электронных лампах. Толчком к использованию ЭВМ в связи явилось запатентованное в 1955 г. предложение на применение программного управления узлом коммутации телефонных каналов. Внедрение вычислительной техники в телефонных и телеграфных сетях связи позволило существенно расширить услуги, предоставляемые абонентам, повысить эффективность этих сетей.

Развитие ЭВМ, создание распределенных вычислительных систем и автоматизированных систем управления явились стимулом к разработке новых средств связи, внедрению новых методов передачи информации. В существующих к настоящему времени *информационно-вычислительных сетях* ИВС, построенных на основе аналоговых линий связи, широко используются программные методы обработки, специализированные связные ЭВМ. С увеличением объема дискретной информации стало экономичным не подключать ЭВМ к аналоговым сетям, а строить специальные высокоскоростные дискретные сети связи, внедрение которых началось в 1970 г. Передача по таким сетям ведется со скоростью порядка 100 Мбит/с. Применение волоконно-оптических линий позволяет достичь скорости 800 Мбит/с и более. В последние годы происходит быстрое слияние средств вычислительной техники и связи в единые информационно-вычислительные сети, что стало возможным благодаря успехам микроэлектронной техники.

Рассмотрим тенденцию перехода к программной реализации на примере такого устройства ИВС, как *мультиплексор передачи данных* (МПД).

Включение ЭВМ в состав ИВС означает превращение ЭВМ в ГВМ сети, на которую, как было показано выше, накладываются требования по реализации протоколов 2...4-го уровней. Становится необходимым определенный набор модулей и средств, выполняющих процедуры 2-, 3- и 4-го уровней транспортной сети ИВС. Существует два основных подхода к решению этой задачи:

1. Все основные функции ГВМ, относящиеся к обмену по транспортной сети, выполняются программным модулем ЭВМ, определяемым как *«телекоммуникационный метод доступа»* (рис. 1.15, *а*). К каналу ввода—вывода ЭВМ подсоединяется специализированный аппаратный блок — МПД, на который возлагается ряд функций 2-го уровня (управления информационным каналом). К ним относятся: распознавание управляющих символов «конец блока», «конец текста» и др.; сборка (разборка) символов из двоичных разрядов; управление работой аппаратуры передачи данных.

Такой подход не требует дополнительного оборудования (за исключением достаточно простого МПД), однако при этом значительные ресурсы ГВМ тратятся на обеспечение обмена информацией, что ведет к увеличению времени решения задач пользователей.

2. Значительная часть процедур протоколов 2...4-го уровней выполняется в интерфейсном блоке. Сложность большинства протокольных процедур обычно приводит к программной реализации МПД в виде *буферного процессора*. При таком подключении (рис. 1.15, *б*) сложный комплекс программ «телекоммуникационный метод доступа» заменяется в ГВМ относительно простой программой управления работой буферного процессора.

Остановимся несколько подробнее на эволюции технической реализации МПД. Первые отечественные се-

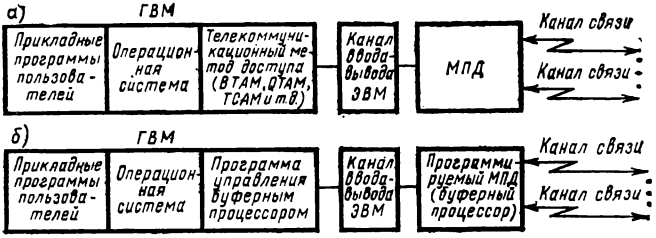



Рис. 1.15. Структура аппаратного (*а*) и программного (*б*) подключения ГВМ к ИВС

рийно выпускаемые мультиплексоры появились в составе технических средств ЕС ЭВМ: МПД-1А (ЕС-8400), МПД-1 (ЕС-8401), МПД-3 (ЕС-8403). Аппаратная реализация этих МПД, ограниченный набор функций не позволяли добиться требуемой гибкости и универсальности сопряжения ЭВМ с различными АП и ИВС, требовали значительного объема ПО и ресурсов ЭВМ на проведение обмена информацией.

В дальнейшем все большее распространение стали получать программируемые МПД. На рис. 1.16 показано несколько способов сопряжения такого МПД с ГВМ: *а*— МПД не имеет собственной буферной памяти, что выгодно с точки зрения стоимости, но при этом возможности МПД ограничены; *б* — ГВМ и МПД имеют общую внешнюю память; *в* — МПД комплектуется достаточным объемом ЗУ для организации буферизации и хранения местных массивов информации. В этом случае МПД приобретает возможность выполнения протоколов 2...4-го уровней, необходимых для доступа в транспортную сеть. Протоколы реализуются программным путем на ЭВМ (буферном процессоре) с достаточной производительностью. Использование буферной ЭВМ освобождает до 25 % процессорного времени ГВМ, увеличивает скорость

передачи на 70...150 % и сокращает задержку на 130... 275 %.

Программная реализация МПД возможна на базе как мини-, так и микроЭВМ. Такой же программный подход к реализации связных функций характерен в настоящее время и для всех остальных элементов ИВС.

**МиниЭВМ в системах связи.** В начале 60-х годов на смену большим дорогостоящим ЭВМ пришли миниЭВМ, которые оказались более удобны для использования в системах связи. Эти машины позволяли реализовать устройства, работающие в реальном масштабе времени. Удобной для обработки символов и байтов была и длина слова миниЭВМ—8...16 бит. Наличие универсальных блоков сопряжения, шин и памяти с прямым доступом обеспечивают большую гибкость при расширении системы. Эти обстоятельства, а также относительно низкая их стоимость явились стимулом к появлению в начале 70-х годов связных процессоров на базе миниЭВМ.

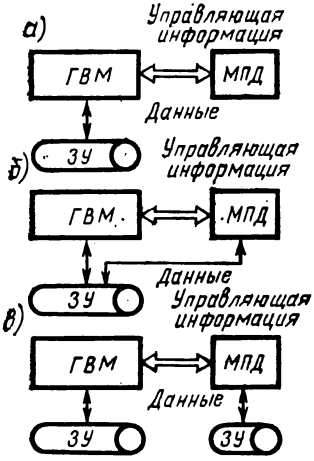



Рис. 1.16. Варианты сопряжения программного МПД с ГВМ

При выделении класса связных минимашин выявились следующие их особенности:

1. Ориентация на достаточно простую обработку массивов информации, побайтовую и побитовую обработку данных. В связи с этим разрядность большинства машин составляла 16 бит. Ряд специфических команд (ввода—вывода, чтения—записи, управления буферами памяти, проверки на четность, обработки прерываний) реализуется для повышения быстродействия микропрограммно. Эти ЭВМ имеют, как правило, большое количество универсальных и индексных регистров. Индексные регистры обеспечивают несколько способов адресации к памяти, что необходимо вследствие малой длины машинного слова ЭВМ. Универсальные регистры позволяют сократить время на обработку прерываний и упростить процесс перекодирования.

2. Работа связной ЭВМ в реальном масштабе времени требует высокого быстродействия процессора (сотни тысяч операций в секунду) и достаточного объема памяти (16...128 Кслов). Необходима также система прямого доступа к памяти, когда устройства ввода—вывода обмениваются информацией с внешней памятью, минуя процессор. Кроме того, от структуры ЭВМ требуется развитая система прерываний.

3. Универсальность использования связных процессоров привела к магистральной (шинной) структуре ЭВМ. Такой способ организации, при котором процессор, ОЗУ и внешние устройства подключаются к специальной многопроводной шине, позволяет обеспечить модульный принцип построения системы, быстрое расширение памяти и числа подсоединяемых устройств, удобство организации многопроцессорных комплексов.

4. Специфика средств связи связной ЭВМ требует обеспечения высокой надежности функционирования и простоты обслуживания. С этой целью связные системы часто делают резервированными и дублированными, оснащают их сложными диагностическими программными средствами.

Обобщенная блок-схема *связного процессора* приведена на рис. 1.17, где выделены три типичные структуры связного процессора, соответствующие его использованию в качестве различных устройств ИВС. Блок БСК выполняет функции сопряжения с каналом ввода — вывода ГВМ.

Как видно из рисунка, связные минимашины характеризуются универсальностью применения, которое зависит от комплектации ЭВМ (БСК, линейные адаптеры, модули сопряжения с устройствами ввода — вывода) и соответствующего ПО. Таким универсальным элементом является, например, отечественная миниЭВМ Р-10 (ЕС-1010), на базе которой создаются АП (АП-50), буферные процессоры и т. д. Другая миниЭВМ KRS — 4201 (ГДР) служит базой построения концентраторов данных, буферных процессоров (ЕС-8404), программируемых АП (АП-4210).

МиниЭВМ являются в настоящее время основным элементом построения устройств ИВС. На их базе реализуются программируемые АП (АП-4, АП-50 и т. д.), мультиплексоры передачи данных (МПД-4 в ЕС ЭВМ, IBM 3705 и др.), концентраторы и удаленные мультиплексоры. Для создания высокопроизводительных центров комму-

тации часто используются мультиминипроцессорные системы (например, 5-процессорный ЦКС СС1-7000).

**МикроЭВМ в системах связи.** Прогресс технологии изготовления больших интегральных схем (БИС) привел к созданию в 70-х годах универсальных БИС с программируемым поведением — микропроцессоров. Объединение же МП с постоянной и оперативной памятью и устройствами управления вводом—выводом позволило соз-

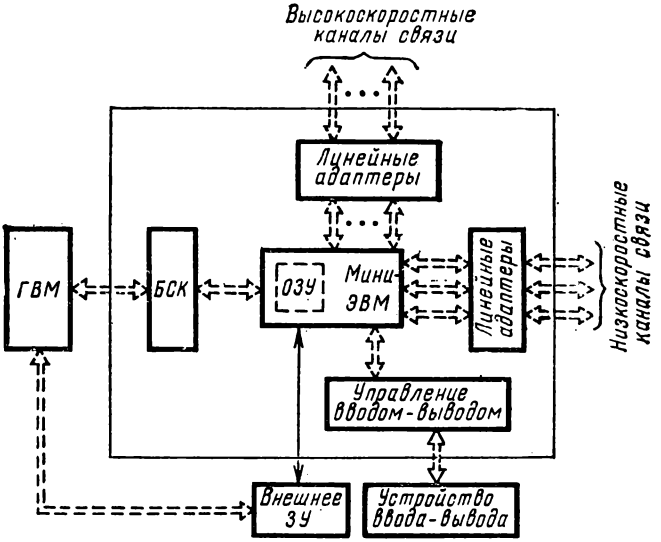



Рис. 1.17. Структурная схема связного процессора

дать микроЭВМ — вычислительную машину, выполненную на одном или нескольких кристаллах.

Выделяют два основных случая использования МП средств: в системах, гдо они выполняют функции, ранее реализуемые на минимашинах; взамен аппаратных устройств на интегральных схемах (ИС). Такая замена аппаратной логики микропроцессорами повышает гибкость устройств управления в системах автоматики и связи.

Основной областью применения МП стали *контроллеры* — относительно небольшие устройства управления некоторыми объектами или технологическими процессами. Их используют в бытовой аппаратуре, устройствах связи, АСУ ТП, робототехнике, при создании гибких автоматизированных производств и других сферах. МП БИС на-

ходят все большее применение как составные элементы мини- и больших ЭВМ, а также как средства для создания программируемых АП, различные устройства телеобработки информации и сетей связи.

Применение миниЭВМ в качестве контроллера характеризовалось в большинстве случаев функциональной и аппаратной избыточностью и высокой их стоимостью. Низкая стоимость микропроцессорных средств сделали экономически целесообразной программную реализацию любого аппаратного контроллера, содержащего более 50 ИС малой степени интеграции. Такой МП контроллер (рис. 1.18, *б*) может быть разработан за меньший срок, отличается простотой модернизации и расширения множества функций, большей надежностью.

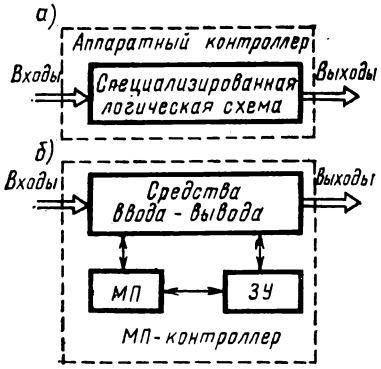



Рис. 1.18. Аппаратный (*а*) и микропроцессорный (*б*) варианты контроллера

Средства связи явились одним из самых крупных направлений использования микромашин. Еще в 1976 г. 16 % (по стоимости) всех выпускаемых в мире МП потреблялись для нужд связи. Одной из первых сфер применения МП оказалась телефония. В 1974 г. была создана автоматическая телефонная станция, BCS-50, использующая МП модели Intel 8008, в которой 8-разрядный МП обеспечивает мультиплексный режим работы, управляет соединением, отключением и другими состояниями абонентских точек. Станция большей производительности была разработана на базе пяти микропроцессоров: два из них являются препроцессорами, взаимодействующими с абонентами, еще два — выполняют функции обработки вызовов, а пятый управляет работой станции.

Опыт последних лет показывает, что практически все устройства систем связи и телеобработки перспективно строить на базе МП средств. В тех случаях, когда производительности единичного МП оказывается недостаточно, используют мультипроцессорные и многомашинные системы. Наибольшее число МП потребляют оконечные уст-

ройства ИВС — терминалы, программируемые АП, буферные процессоры ГВМ. Вычислительные мощности позволяют наделять эти устройства некоторыми интеллектуальными возможностями — производить частичную обработку информации, простые расчеты, иметь местные массивы, реализовывать протоколы 2...4-го уровней ИВС.

Применение МП в связных устройствах вместо схем с «жесткой» логикой обеспечивает такие положительные моменты, как универсальность, гибкость и надежность. К числу таких устройств относят МПД, аппаратуру передачи данных, модемы. В тех же случаях, когда микромашины заменяют связные миниЭВМ, выигрыш проявляется в стоимости, массе, габаритах, энергопотреблении, надежности. В последние годы в результате развития ряда машин миникласса появляются высокопроизводительные микроЭВМ, например модели СМ-50/40-1 и СМ-50/50-2, которые могут сопрягаться с миниЭВМ СМ-3, СМ-4. Применение данных микроЭВМ позволяет модернизировать ранее разработанные системы, строить иерархические программные комплексы.

Использование микропроцессорных средств в устройствах телеобработки обладает теми же специфическими особенностями, что были отмечены выше для минимашин. Главными среди них являются: наличие системы прямого доступа к памяти и достаточный набор контроллеров управления вводом — выводом информации. Перспективна также функциональная ориентация МП в сетевых устройствах с разделением процессоров на выполнение функций 2-, 3- и 4-го уровней протоколов ИВС.

**Тенденции внедрения МП в устройства связи.** МикроЭВМ в связных устройствах не только заменяют аппаратные блоки с «жесткой» логикой, но и завоевывают все традиционные области использования связных минимашин. Последнее объясняется наличием у микроЭВМ таких признаков современных миниЭВМ, как: достаточно мощная система команд с развитой системой адресации, например у МП К580 число команд около 80, а с учетом модификаций — более 200; большое число регистров общего назначения, которое часто больше, чем в миниЭВМ; многоуровневая система прерываний и малое время реакции на запросы; наличие канала прямого доступа к памяти; периферийный интерфейс в виде одной или нескольких БИС ввода—вывода.

Происходит постепенное стирание различий между мини- и микроЭВМ. Так, например, в программе I очере-

ди СМ-ЭВМ в нашей стране (1976—1980 гг.) разработано четыре модели этих минимашин: С-1, СМ-2, СМ-3, СМ-4. Первым классом II очереди является микроЭВМ СМ-50 с характеристиками, близкими к миниЭВМ. У микроЭВМ СМ-50/50-2 длина слова составляет 16 разрядов, объем адресуемой памяти — 256 Кслов. Эта машина обладает высокой производительностью и может дополнительно оснащаться арифметическим МП АРР-30, позволяющим обрабатывать 32...64-разрядные слова. Таким образом, в ближайшие годы деление малых ЭВМ на мини- и микромашины станет, по-видимому, весьма условным — образуется единый класс малых ЭВМ на БИС.

# ГЛАВА 2
# СОВРЕМЕННЫЕ МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА И ИХ ПРИМЕНЕНИЕ ДЛЯ РЕАЛИЗАЦИИ УСТРОЙСТВ СИСТЕМ ПЕРЕДАЧИ ИНФОРМАЦИИ

## § 2.1. МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА И СИСТЕМЫ

**Основные определения.** Понятие микропроцессорных средств и систем включает в себя множество элементов и модулей, используемых при применении МП в конкретных системах.

*Микропроцессорные средства и системы* — совокупность изделий вычислительной и управляющей техники и их функционально и конструктивно законченных составных частей, построенных на основе микропроцессорных интегральных микросхем. Синонимами являются термины «микропроцессорная техника» и «микропроцессорные средства вычислительной техники».

*Микропроцессорные средства* — конструктивно и функционально законченные изделия вычислительной и управляющей техники, построенные в виде или на основе микропроцессорных интегральных микросхем, которые с точки зрения требований к испытаниям, приемке и поставке рассматриваются как единое целое и применяются при построении более сложных микропроцессорных средств или микропроцессорных систем.

Конструктивно микропроцессорные средства выполняются в виде микросхемы, одноплатного изделия, моноблока или типового комплекса, причем изделия младше-

го уровня конструктивной иерархии могут использоваться в изделиях старшего.

*Микропроцессорные системы* — вычислительные или управляющие системы, построенные на основе микропроцессорных средств, применяются автономно или встраиваются в управляемый объект.

Конструктивно микропроцессорные системы выполняются в виде микросхемы, одноплатного изделия, моноблока, комплекса или нескольких изделий указанных ти-

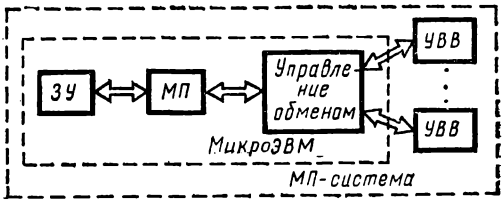



Рис. 2.1. Структура микроЭВМ и микропроцессорной системы

пов, встроенных в аппаратуру управляемого объекта или выполненных автономно.

*Микропроцессор* — программно управляемое средство, осуществляющее процесс обработки цифровой информации и управления им, построенные на одной или нескольких интегральных микросхемах.

*Микропроцессорная электронная вычислительная машина* (микроЭВМ) — цифровая ЭВМ с интерфейсом ввода—вывода, состоящая из микропроцессора, полупроводниковой памяти и при необходимости пульта управления и источников электропитания, объединенных общей несущей конструкцией.

Соотношение между МП, микроЭВМ и микропроцессорной системой показано на рис. 2.1.

*Семейство микроЭВМ* — совокупность архитектурно совместимых микроЭВМ. Под архитектурной совместимостью понимается совместимость программного обеспечения и интерфейсов.

*Микропроцессорный контроллер* — функционально законченное и конструктивно оформленное изделие, содержащее микропроцессорный модуль, модули памяти и интерфейса, предназначенное для встраивания в конкретное технологическое и прочее оборудование, и ориентированное на выполнение конкретных задач путем программирования постоянной памяти, конфигурирова-

ния оперативной памяти и внешних устройств, а также разработки необходимых для данного применения специальных интерфейсов.

*Большая интегральная микросхема* (БИС) — полупроводниковая интегральная схема, содержащая 500 и более элементов, изготовленных по биполярной технологии, 1000 и более элементов, изготовленных по МОП-технологии.

*Микропроцессорная интегральная микросхема* — интегральная микросхема, выполняющая функцию МП (микроконтроллера) или его части.

*Микропроцессорный модуль* — конструктивно и функционально законченное изделие, имеющее унифицированные присоединительные характеристики (интерфейс, конструкция, программное обеспечение и т. п.).

Микропроцессорный модуль может выполнять функции МП, ОЗУ или ПЗУ, контроллера, внешнего запоминающего или периферийного устройства, микроЭВМ и т. п. Конструктивное исполнение его: микросхема, частичный и комплектный блок.

*Микропроцессорный комплект БИС МПК* — совокупность микропроцессорных и других интегральных микросхем, совместимых по архитектире, конструктивному исполнению и электрическим параметрам и обеспечивающих возможность совместного применения.

*Микропроцессорный набор* — совокупность микропроцессорных и других интегральных микросхем микропроцессорного комплекта БИС, номенклатура и количество которых необходимы и достаточны для построения конкретного изделия вычислительной или управляющей техники.

*Микропроцессорная секция* — микропроцессорная интегральная микросхема, реализующая часть МП (микроконтроллера) и обладающая средствами простого функционального объединения с однотипными и другими микропроцессорными секциями для построения законченных МП, микроконтроллеров или микроЭВМ.

*Однокристальный МП* — МП, выполненный в виде одной БИС.

*Однокристальная микроЭВМ* — микроЭВМ, выполненная в виде одной БИС.

*Специализированный микропроцессор* — МП, структура которого оптимизирована для решения задач определенного класса.

*Разрядность микропроцессора* — число разрядов регистров арифметико-логического устройства (АЛУ) микропроцессора.

Разрядность МП не всегда соответствует разрядности обрабатываемых данных. Например, 8-разрядное АЛУ микропроцессорного комплекта БИС К581 осуществляет (за два такта) обработку 16-разрядных данных.

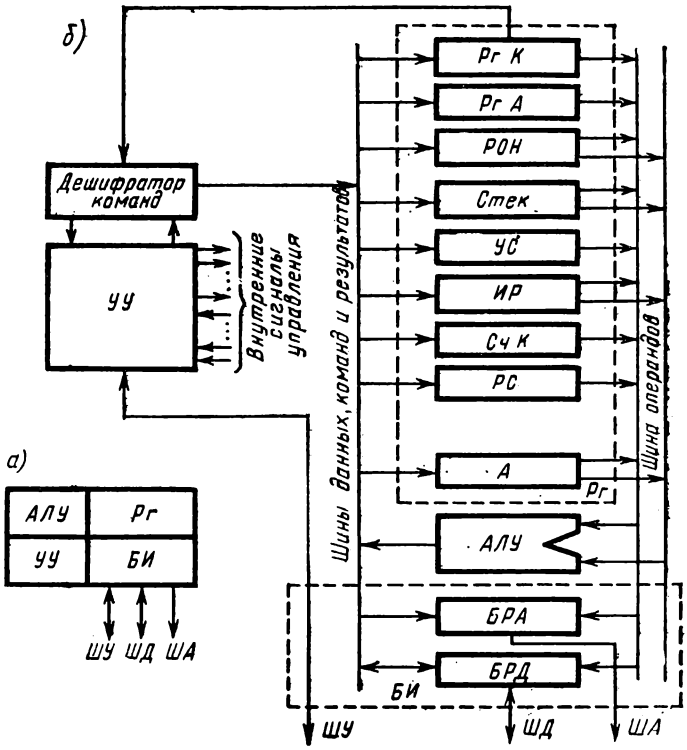



Рис. 2.2. Обобщенная структура МП

**Структура микропроцессоров.** Рассмотрим структурную организацию МП, используя упрощенную схему. Конкретные МП могут в деталях существенно отличаться друг от друга, однако в них выделяют следующие основные узлы (блоки) (рис. 2.2, *а*): арифметико-логическое устройство (АЛУ); блок внутренних регистров (Рг); устройство управления (УУ); блок интерфейса (БИ).

Рассмотрим состав этих блоков по упрощенной структуре (рис. 2.2, *б*).

*Арифметико-логическое устройство* выполняет несколько простейших операций: сложение, вычитание, пересылку, логическое И, логическое ИЛИ, сложение по модулю 2 и сдвиг. Признаки операций АЛУ, а также состояние МП запоминаются на специальных триггерах, образующих регистр состояния РС.

*Блок внутренних регистров* образует внутреннюю память МП и содержит регистры специальные и общего назначения РОН. Регистры блока связаны с другими внутренними блоками МП общими шинами. Отметим, что каждый конкретный МП не содержит обычно всех типов регистров, показанных на рисунке. Функции отсутствующих элементов Рг выполняют либо РОН, либо ячейки внешней памяти. Блок РОН содержит от 4 до 64 регистров, доступных программисту. Использование РОН как сверхоперативной памяти для хранения операндов и других данных позволяет в 1,5—3 раза уменьшить число обращений к ОЗУ при выполнении программы.

*Счетчик команд* СчК содержит адрес выбираемой из ЗУ следующей по порядку команды в программе.

*Регистр адреса* РгА служит для хранения адреса операнда, находящегося во внешней памяти или другом регистре, или же адреса ячейки памяти, куда необходимо передать результат из аккумулятора.

*Аккумулятор* А является накопительным регистром, в котором производится хранение одного из операндов и промежуточных результатов операций.

*Регистр команды* РгК принимает и хранит код очередной команды, адрес которой перед этим был определен СчК. По сигналу РгК выдает информацию в УУ.

*Регистр состояния* РС фиксирует состояние МП в каждый момент выполнения программы. Его содержимое (флаги) используются для перехода внутри программы по заданным признакам и условиям.

*Стек* представляет собой ЗУ, куда информация заносится последовательно и извлекается в порядке, обратном порядку занесения. *Указатель стека* (УС) содержит адрес первой свободной ячейки в стеке. В ряде случаев в блоке Рг реализуется только УС, а сам стек формируется в ОЗУ.

*Индексные регистры* (ИР) служат для формирования адресов ЗУ.

*Устройство управления* (УУ) использует код опера-

ции команды для формирования внутренних сигналов управления работой блоков МП (в частности, АЛУ).

*Интерфейсный блок* БИ — это аппаратура, обеспечивающая сопряжение между МП, ЗУ и периферийными устройствами, а также между блоками МП. В состав БИ входят буферный регистр адреса (БРА) и буферный регистр данных (БРД). *Шина данных* (ШД) служит для передачи операндов и команд, с которыми оперирует МП. *Шина адреса* (ША) МП чаще всего бывает 16-разрядная, что достаточно для прямой адресации внешней памяти емкостью 64 Кслов. Двунаправленная *шина управления* (ШУ) обычно содержит 6...10 разрядов. По ней передаются управляющие сигналы: от внешних устройств к МП (об их состоянии); запроса на прерывание от внешних устройств к МП и разрешения прерывания от МП к внешним устройствам; записи слова во внешнюю память и чтения слова из памяти.

Для многокристальных секционированных МП характерно то, что операционная часть МП, содержащая АЛУ, блок Рг, дешифратор микрокоманд и схемы пересылки адресов и информации внутри МП, разделена (расщеплена) на равные части, представляющие собой 2-, 4- или 8-разрядные слои со своими адресными и информационными шинами. Управление выполнением операций осуществляется от отдельного кристалла микропрограммного управления, причем микропрограммное слово поступает параллельно на все секции. В таком секционированном МП количество секций зависит от требуемой разрядности МП и легко наращивается до 16, 32 и более разрядов.

**Основные характеристики и классификация микропроцессоров.** Основные характеристики МП, определяющие наиболее эффективную область их использования и особенности разработки систем на их основе, включают в себя параметры, присущие как вычислительным средствам, так и интегральным схемам. Важнейшими схемотехническими параметрами МП являются: разрядность; емкость адресуемой памяти; универсальность (сцециализация МП); число внутренних регистров; наличие микропрограммного управления; число уровней прерывания; тип стековой памяти и число основных регистров; состав резидентного и кросс-программного обеспечения.

Основными характеристиками МП, присущими интегральным схемам и определяемыми интегральной технологией, являются: быстродействие; мощность потребле-

ния; габариты и масса; количество уровней питания; надежность; эксплуатационная стойкость; стоимость.

Перечисленные характеристики позволяют классифицировать МП по наиболее существенным из них с целью выбора эффективной сферы применения. Один из возможных вариантов такой классификации приведен на рис. 2.3.

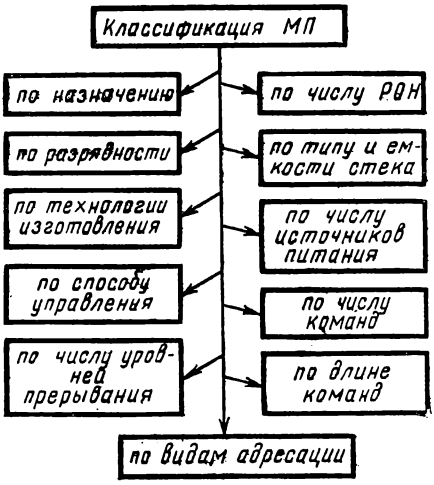



Рис. 2.3. Признаки классификации МП

По назначению МП делятся на универсальные и специализированные. Универсальным МП или МП общего назначения называют прибор, имеющий широкое применение в различных областях народного хозяйства. Специализированными называют МП, оптимизированные по некоторым параметрам на конкретные применения. Так, например, подключение специализированного арифметического МП Am 9511 к универсальному прибору Intel 8080 позволяет уменьшить время выполнения умножения в 40 раз, извлечения квадратного корня — в 200 раз, вычисления синуса — в 60 раз.

По разрядности МП подразделяются на МП с фиксированной разрядностью и с изменяемой разрядностью слова (модульные). При фиксированной разрядности наиболее распространены модели с длиной слова 8 и 16 бит (хотя имеются и 4- и 12-разрядные МП). При модульном принципе возможно построение 8, 16, 24, 32-разрядных МП и т. д. из секций разрядностью 2, 4 или 8.

Технология изготовления является одним из определяющих факторов в МП-системе. От нее зависят параметры МП по быстродействию, температурному диапазону, потребляемой мощности, стоимости и др. Так, при наиболее широко используемых технологиях

*p*-МОП, *n*-МОП, КМОП, КМОП/КНС, И²Л, ТТЛШ, ЭСЛ скорость выполнения операций типа «регистр — регистр» возрастает от 100 тыс. операций/с (для *p*-МОП) до 10 и более млн. операций/с при ЭСЛ технологии.

По способу управления МП делятся на микро- и макропрограммируемые. Микропрограммное управление характерно для микропроцессорных секций с наращиваемой разрядностью (таких, как отечественные серии К584, К587, К589) и позволяет пользователю установить свой собственный набор инструкций (команд), оптимальных для реализации некоторых конкретных задач. Макропрограммное (жесткое аппаратное) управление принципиально не допускает такой возможности.

МП могут не иметь возможности прерываний выполняемой программы или иметь одно- или многоуровневую систему прерывания. При многоуровневом прерывании разрешается прерывание прерывания. Наличие такого многоуровневого прерывания является неотъемлемым свойством систем, работающих в реальном масштабе времени.

Число регистров общего назначения характеризует объем сверхоперативной памяти МП с малым временем обращения. Количество РОН в МП колеблется от 2 до 64 и служит одним из показателей вычислительных возможностей прибора.

Большинство МП имеют магазинную память — стек, при обращении к которой не требуется указания адреса. В МП используется два типа стека: встроенный и автономный. Встроенный стек размещается полностью на кристалле МП. Емкость (глубина) стека при этом, как правило, не превышает 16...32 слов. Автономный стек реализуется в ОЗУ, поэтому глубина его может быть равна адресуемой емкости памяти, однако при этом увеличивается время обращения к стеку.

Число источников питания определяет сложность монтажа МП системы, ее габариты, надежность, стоимость. Количество необходимых источников питания колеблется от 1 до 3.

Существует и более детальная классификация МП, например по таким признакам, как число шин МП, организация системы ввода — вывода, число программных счетчиков, аккумуляторов и др.

Нашей промышленностью освоены все современные технологические процессы производства БИС. Выпускаемые МПК БИС могут быть разделены на две группы.

Универсальные МПК БИС предназначены для применения в самых различных средствах вычислительной техники. К этой группе принадлежат такие комплекты, как К580, К584, К587, К588, К589, U 83-К1883. Примерами зарубежных МПК являются MCS 86 (фирма «Intel»), M6800 и M10800 (фирма Motorola) F8 (фирма «Fairchild») и др.

Специализированные МПК БИС предназначены для построения только одного типа вычислительных машин. В данную группу входят комплекты К536, К581, К586, СР-1600 (фирма «General Instruments»), LSI 12/16 (фирма «General Automation») и др.

Отечественные МПК БИС базируются на различных микроэлектронных технологиях. Комплект БИС серии К536 изготовляют по *p*-МОП-технологии. Технология *n*-МОП используется в МПК К580, К581, К586, U 83-К1883, К1801. На базе КМОП-технологии реализованы МПК К587, К588. Комплекты КР582, К583, КР584 используют технологию И²Л. МПК К589, КР1802, КР1804 реализованы по технологии ТТЛШ. Серия К1800 построена по ЭСЛ-технологии.

Комплекты БИС К536 и К586 используются для построения микроЭВМ семейства «Электроника С5». Серия К581 послужила основой для микроЭВМ «Электроника 60». МПК К587 и К588 предназначены для построения ряда совместимых микроЭВМ «Электроника НЦ». Быстродействующий МПК К589 широко используется при разработке технических средств серии СМ ЭВМ.

На базе МПК К1801, включающего в свой состав однокристальные МП К1801ВМ1 и К1801ВМ2, созданы одноплатные микроЭВМ ряда «Электроника МС 1201», предназначенные для применения в различных промышленных и научных системах обработки цифровой информации.

Данные микроЭВМ полностью совместимы с микроЭВМ ряда «Электроника 60» по системе команд, системной магистрали и конструктивному исполнению. Разрядность микроЭВМ — 16 бит, емкость ОЗУ — 64 Кбайт. Наиболее быстродействующей в ряду является микроЭВМ «Электроника МС 1201.02», реализованная на базе МП КМ1801ВМ2. Число команд микроЭВМ равно 72, быстродействие выполнения команд типа «сложение» — около 800 тыс. операций/с.

Современные микроЭВМ успешно заменяют миниЭВМ в различных областях применения. Например, микроЭВМ

«Электроника 60М» выполняет те же функции в вычислительных системах, которые раньше могли быть реализованы только с помощью миниЭВМ типа СМ-3, «Электроника 100/16И». Дальнейшая миниатюризация средств вычислительной техники позволяет применять вновь разработанные микроЭВМ вместо миниЭВМ типа СМ-4, «Электроника 100/25». Так, например, микроЭВМ «Электроника ИС 1211» и «Электроника МС 1212» имеют по сравнению с микроЭВМ типа «Электроника 60М» более высокое быстродействие (в 2...3 раза), расширенную систему команд, увеличенный объем памяти, возможность мультипрограммного режима работы.

Сохраняя программную совместимость с микроЭВМ «Электроника 60М» и «Электроника 100/25», новые микроЭВМ дополнительно могут выполнять 46 команд над числами в форме с плавающей запятой. МикроЭВМ «Электроника МС 1212» характеризуется объемом адресуемой памяти — 4Мбайт, разрядностью ЭВМ — 16 бит, числом команд — 138, быстродействием при выполнении команд типа «сложение» — 580 тыс. операций/с.

Ведется также выпуск массовых однокристальных микроЭВМ серий К1814, К1820, К1816, предназначенных для применения в самых различных областях, в том числе и в разнообразных высокопроизводительных системах управления и обработки данных. МикроЭВМ К1814 реализована по *p*-МОП технологии, имеет разрядность 4 бит, число команд 43 и длительность командного цикла 20 мкс. Серии К1820 и К1816 используют более быстродействующую *n*-МОП технологию. МикроЭВМ К1820 имеет разрядность 4 бит, число команд — 49, а длительность командного цикла составляет 4 мкс. Разрядность микроЭВМ К1816 равна 8 бит, число команд — 96, длительность командного цикла — 2,5 мкс.

**Программное обеспечение микропроцессорных средств.** *Программное обеспечение* (ПО) МП-систем является совокупностью всех средств, необходимых для разработки и решения на данной системе определенных задач, а также для обеспечения эффективного функционирования системы. ПО можно разделить на три основные группы средств: языки программирования МП-систем, ПО пользователя и системное ПО (рис. 2.4).

*Программное обеспечение пользователя* включает в себя исходные, объектные и рабочие программы. Под исходной понимается программа, написанная на языке ассемблера (макроассемблера) или языке высокого уров-

ня. Программа, полученная после трансляции исходной, называется объектной. Объектное представление соответствует уровню машинных кодов микроЭВМ. Скорректированная и протестированная объектная программа представляет собой готовую рабочую программу МП-системы.

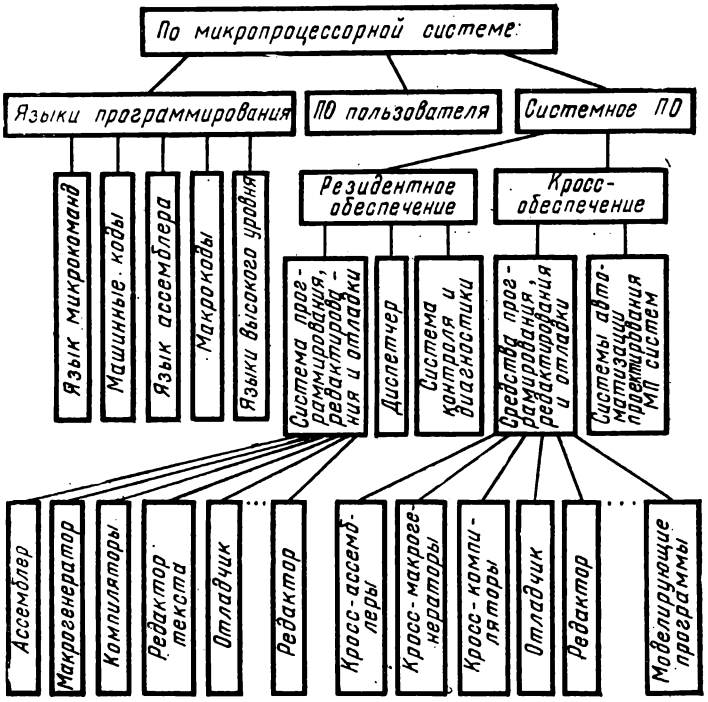



Рис. 2.4. Структура ПО микропроцессорных систем

*Системное* ПО — это вспомогательные (служебные) программы, не зависящие от конкретного применения МП-системы. В зависимости от того, реализуются ли программы системного ПО на той же микроЭВМ, на которой будут работать прикладные программы, или же на ЭВМ другого типа, это ПО называют соответственно резидентным или кросс-обеспечением.

*Резидентное системное* ПО обеспечивает:

организацию совместного функционирования всех элементов структуры МП-системы. Эту задачу выполня-

ет программа-диспетчер, определяемая в зависимости от сложности функций как супервизор, монитор или операционная система;

контроль правильности функционирования МП-системы и диагностику отказов в ее работе (система контроля и диагностики);

создание пользователем рабочих программ (система программирования, редактирования и отладки).

*Кросс-программное* обеспечение включает в себя средства разработки рабочих программ, реализованные не на микроЭВМ, для которой разрабатывается ПО а на какой-либо другой машине. Применение кросс-средств делает возможной одновременную разработку МП-системы и ее ПО. Эта группа системного ПО содержит программы кросс-ассемблеров, кросс-компиляторов, моделирующие программы и т. д. Широкое применение МП-систем привело к появлению специализированнных вычислительных систем, предназначенных для повышения эффективности процесса проектирования ПО (см. гл. 4).

Остановимся кратко на составе программ — систем программирования, редактирования и отладки резидентного и кросс-обеспечения.

*Программа редактор текста* используется для упрощения процесса создания исходных программ. С ее помощью производятся операции по корректировке и редактированию (добавление, исключение, замена частей и т. д.) программы пользователя, записанной во внешнюю память машины.

Преобразование исходных программ в объектные производят *транслирующие программы.* Программа ассемблер осуществляет трансляцию исходных модулей с языка ассемблера, а программы компиляторов — с языков высокого уровня (БЕЙСИК, ПЛ/М и др.). При трансляции производится выявление синтаксических ошибок в тексте исходной программы.

*Макрогенератор* представляет собой специальную программу обработки макрокодов, каждый из которых объединяет группу последовательных команд на языке ассемблера. Макрогенератор заменяет каждую команду макрокода соответствующей ей группой команд.

Отладка и проверка объектного модуля выполняется с помощью *программы отладчик.* Используя отладчик, можно осуществить выполнение программы по частям, вводить точки остановка, выводить на дисплей (печать) содержимое регистров и ячеек памяти, производить оста-

новку программы по условию и другие операции, позволяющие выявить семантические ошибки.

Отдельно составленные и отлаженные части программы пользователя объединяются в единое целое с помощью *программы редактора,* налаживающей связи между модулями в соответствии с общей структурной схемой.

*Моделирующие программы* позволяют имитировать на кросс-системе работу будущей микроЭВМ и тем самым отлаживать объектный модуль. При моделировании выдается разнообразная диагностическая информация, однако после моделирования неизбежна окончательная отладка программ в самой МП-системе.

## § 2.2. ФУНКЦИОНАЛЬНАЯ И ФИЗИЧЕСКАЯ СТРУКТУРЫ МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВ СПИ

Технические средства СПИ, при построении которых в первую очередь находят применение микропроцессоры и микроЭВМ, можно разделить на абонентские пункты; устройства концентрации нагрузки; центры коммутации; устройства сопряжения ЭВМ с КС.

В общем случае программируемые устройства СПИ могут быть заданы в форме функциональной (логической) структуры, отображаемой в физическую структуру аппаратных и программных средств микропроцессорного устройства. Такое выделение целесообразно как с точки зрения анализа устройства, так и при решении ряда вопросов их проектирования.

Под *функциональной,* или *логической,* структурой понимают конечное множество $F$ взаимосвязанных функций, выделяемых в том или ином устройстве. Множество $W$ взаимосвязанных аппаратных и программных модулей, составляющих устройство и реализующих заданную логическую структуру, определяют как *физическую структуру.* Между логической и физической структурами устройства того или иного класса нет взаимно однозначного соответствия, так как заданная логическая структура может быть в общем случае реализована различными физическими структурами. Выбор $W$ и нахождение наилучшего отображения $F$ на $W$ является одной из важных задач при проектировании программируемых устройств СПИ.

Анализ множества задач, выполняемых устройствами СПИ, показывает, что множество $F$ выполняемых функ-

ций целесообразно разбить на ряд подмножеств $\bigcup_{i \in I} F_i = F$. Так могут быть выделены подмножества *сетевых* функций $F_N$, связанных с передачей сообщений и управлением потоками и сетью в целом, *терминально-пользовательских* функций $F_T$, функций *обмена* с устройствами ввода—вывода (УВВ) $F_O$, *управляющих* функций $F_C$, *сервисных* функций $F_S$, функций *контроля и обеспечения надежности* $F_K$.

Подмножества $F_O$, $F_C$, $F_S$ и $F_K$ носят довольно общий характер для названных выше устройств. Функции подмножества $F_O$ целесообразно рассматривать по группам в зависимости от типа УВВ. Внутри каждой из таких групп присутствуют элементы, отображающие управление работой УВВ, контроль его состояния, преобразование кодов, диагностику неисправностей и т. п.

Работой программируемого устройства управляет специализированная операционная система (диспетчерская программа). Реализуемое ею подмножество $F_C$ содержит элементы, связанные с общей диспетчеризацией работ устройства, распределением памяти, ведением очередей и начальным пуском системы. Сервисные функции $F_S$ связаны со сбором статистических данных об обмене, нагрузке, состоянии КС, с ведением архива форматов, хранением сообщений, отображением состояния системы оператору, реализацией связи с оператором и рядом других вспомогательных операций. Подмножество $F_K$ содержит элементы, связанные с задачами контроля состояния КС и перекоммутацией на резервные каналы, средствами самодиагностики, резервированием аппаратуры и переходом на резервные модули, реконфигурации системы, восстановлением операционной системы и рядом других.

Подмножество $F_T$ функций, связанных с взаимодействием с оператором, наиболее характерно для АП и включает в себя такие элементы, как редактирование вводимых данных и их контроль, управление изображением, защита от несанкционированного доступа и др.

Специфическим устройством СПИ, обеспечивающих передачу в СОИ, является подмножество функций $F_N$. Состав его как количественно, так и качественно, определяется классом устройств СПИ и будет рассмотрен ниже.

Независимо от класса, для всех МПУ СПИ характерным является реализация подмножества функций

установления, ведения и отбоя связи между абонентом—источником сообщений и абонентом — приемником. В общем случае информация по КС поступает в МПУ СПИ непрерывными кадрами, имеющими постоянный формат и состоящими из поля фазового пуска, поля служебных признаков, содержащего адреса абонентов—приемников сообщений, приоритеты, тип связи и информационного поля, распределенного, например, по временным каналам.

Один из вариантов алгоритма обработки кадра сообщения в приемном канале представлен на рис. 2.5, а в передающем канале — на рис. 2.6, где группа блоков *1* соответствует передаче информационного сообщения, а группа блоков *2* — передаче запросного сообщения.

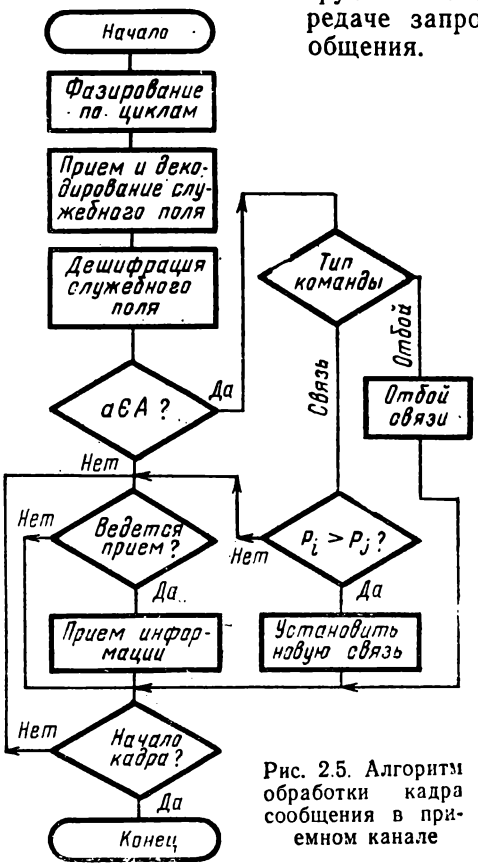



Рис. 2.5. Алгоритм обработки кадра сообщения в приемном канале

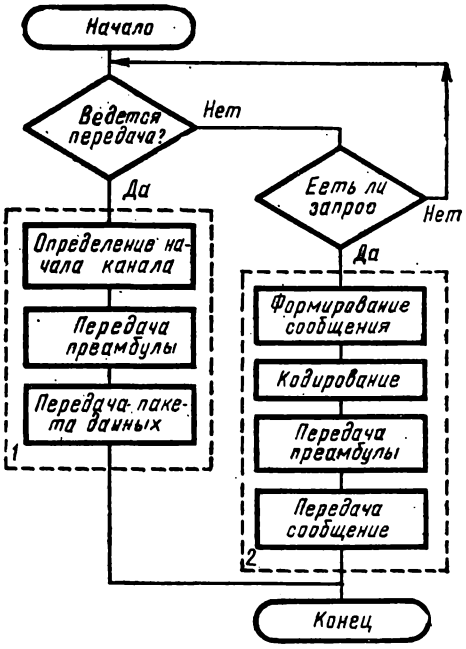



Рис. 2.6. Алгоритм обработки кадра сообщения в передающем канале

Если адрес $a$ принадлежит множеству адресов $A$ абонента—приемника сообщения, то необходимо продолжить дешифрацию признаков данного поля. Новая связь, указанная в служебном поле, устанавливается при условии, что ее приоритет $P_i$ выше, чем приоритет текущей связи $P_j$.

Прием информации состоит в поиске выделенного временного канала и приеме определенного числа символов в память МП-системы из данного канала для дальнейшего вывода на печатающее устройство либо для последующей трансляции. Для передачи абонентом сообщения необходимо сформировать запрос, содержащий следующие сведения: адрес абонента, передающего запрос, адрес абонента — приемника сообщения, тип канала предлагаемой связи, приоритет сообщения. При этом запросное сообщение передается с использованием помехоустойчивого кода. По запросу абонента выделяется временной канал (интервал) длительностью $\tau_j$ с периодом $T$, в течение которого должны передаваться только символы от данного абонента,

Получив разрешение и номер канала (данная информация поступает по приемному каналу и содержится в служебном поле кадра), необходимо синхронизировать работу на передачу таким образом, чтобы передаваемые сигналы попадали точно в отведенный интервал времени. Для этого по приемному каналу передается специальная комбинация, являющаяся цикловой комбинацией для всех абонентов — источников сообщений.

Информация, подлежащая передаче, хранится в ЗУ или накапливается в течение интервала времени $T$, а затем передается пакетом за время $\tau_j$. При этом сначала выдается преамбула (вводная часть), состоящая из последовательности сигналов для восстановления несущей и тактовой синхронизации символов, кодового слова для фазирования по циклам.

Реализация фазирования по циклам сводится к выделению синхросигнала из двоичных последовательностей, принимаемых из КС, и определению момента перехода от режима поиска к режиму контроля и от режима контроля синхронизма к режиму поиска. В режиме поиска синхронизма решение о переходе к режиму контроля принимается при обнаружении фазового пуска на одних и тех же позициях в кадре $l_1$ раз подряд. После выделения первого предполагаемого синхросигнала позиция следующего синхросигнала определяется с помощью таймера. В режиме контроля синхронизма решение о переходе к режиму поиска принимается при отсутствии синхросигнала на анализируемых позициях в кадре $l_2$ раз подряд.

Операция выделения синхросигнала сводится к выполнению сравнения $m$-разрядного входного кода с кодом эталона, хранящимся в памяти МПУ СПИ, подсчету числа единиц $z$ в результате суммирования, сравнению $z$ с допустимым числом ошибок $z_0$. При $z \leqslant z_0$ входной код является синхросигналом, в противном случае при поиске первого предполагаемого синхросигнала происходит прием следующего бита и операция выполняется сначала, т. е. время реализации операции ограничено периодом поступления битов. Базовыми операциями при определении момента перехода являются подсчет единичных приращений, установка флагов, формирование сигналов прерывания. Анализ программной реализации функций фазирования по циклам показал, что 75 % времени затрачивается на выделение синхросигнала.

Информация служебного поля для повышения достоверности, как правило, передается в помехоустойчивом коде. Процедура декодирования служебного поля, например для линейных кодов, заключается в формировании синдрома ошибки.

Базовыми операциями при декодировании являются операции неравнозначности, сдвига, обмена, подсчета единичных приращений. Если синдром равен нулю, то информация принята без ошибок и выполняется дешифрация служебного поля, целью которой, как уже говорилось, является определение адреса абонента, типа команды, приоритета сообщения и других признаков.

Анализ специфики реализации основных функций подмножества $F_N$ позволяет выделить ряд особенностей, которые необходимо учитывать при построении МПУ СПИ:

необходимость обработки МП-системой непрерывного потока кадров, имеющих постоянную структуру, в режиме реального времени;

необходимость мультипрограммного режима работы МП-системы при обслуживании приемного и передающего КС;

широкий диапазон требований к скорости передачи информации, типу используемых помехоустойчивых кодов, числу и типам КС.

Таким образом, программируемые устройства СПИ в общем случае являются устройствами с достаточно сложной логической структурой.

Обобщенная физическая структура МПУ СПИ (центры коммутации, абонентские пункты, концентраторы и мультиплексоры), не имеющих внешних запоминающих устройств большой емкости, может быть представлена в едином виде (рис. 2.7).

Комплекс технических средств (ТС) МПУ СПИ имеет в своем составе аппаратуру передачи данных (АПД) для работы по аналоговым каналам связи либо оборудование сопряжения с цифровыми каналами (ОСЦК) для работы по цифровым каналам с ИКМ или дельта-модуляцией, вычислительный комплекс — связной процессор (СП), выполняющий функции накопления и обработки информационных блоков — кадров и пакетов, а также рабочее место оператора (РМО), обеспечивающее взаимодействие с оператором в режимах загрузки программного обеспечения, управления работой и отображения состояния технических средств и программно-

го обеспечения МПУ СПИ, формирования служебных сообщений для посылки их в другие центры коммутации или АП. В случае АП то же или несколько другое РМО используется в качестве терминала пользователя. АПД (ОСЦК) обеспечивает побитовый прием и передачу информации и представляет собой специальную аппаратуру. СП — это программируемый микропроцессорный комплекс, осуществляющий обработку информации, находящейся в оперативной памяти комплекса, за-

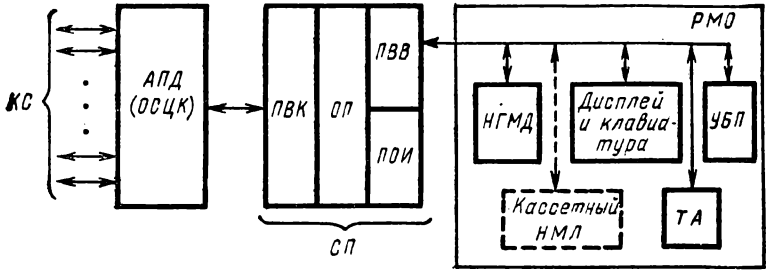



Рис. 2.7. Обобщенная структура МПУ СПИ

писанной в нее из каналов связи или РМО, и управление вводом — выводом. В составе СП обычно выделяется подсистема взаимодействия с каналами (ПВК), подсистема обработки информации (ПОИ) и подсистема ввода—вывода (ПВВ).

ПВК осуществляет управление АПД, прием из АПД и накопление в оперативной памяти (ОП) информационных кадров и пакетов, передачу их из ОП в АПД. ПОИ осуществляет логическую обработку информационных кадров и пакетов, находящихся в ОП.

ПВВ управляет вводом — выводом информации на устройства ввода — вывода РМО.

Каждая из подсистем представляет собой технические средства с соответствующей программной поддержкой. Возможные структуры СП более подробно будут рассмотрены ниже.

РМО состоит из комплекса устройств ввода—вывода, обеспечивающих оперативный ввод директив оператора или информации пользователя и отображения состояния технических средств и программного обеспечения МПУ СПИ, загрузки исходящих направлений связи, информационных сообщений и т. п., а также при необходимо-

сти ее документирование с использованием устройств быстрой печати (УБП). В состав РМО могут входить запоминающие устройства типа накопителей на гибких магнитных дисках (НГМД) и кассетных накопителей на магнитной ленте (КНМЛ) для хранения библиотек программных модулей и информационных файлов различного назначения.

Таким образом, базовым элементом для создания МПУ систем и сетей телеобработки информации являются связные процессоры. Выделение этой группы вычислительных устройств обусловлено своеобразием выполняемых ими функций, рассмотренных выше. Функциональная структура СП находит отражение в физической (программно-аппаратной) структуре. Для таких систем известно достаточно большое многообразие возможных аппаратно-программных реализаций. В связи с этим целесообразно выделить ряд типичных микропроцессорных структур, на основе которых могут быть построены СП.

## § 2.3. СВЯЗНЫЕ ПРОЦЕССОРЫ

Для связных процессоров характерны: обязательное наличие механизма прерываний; введение операций для эффективной обработкой полей данных (циклического сдвига, маскирования и т. д.); операции обработки единичных битов и групп битов информации (нахождение самой правой единицы или нуля и установка и сброс для любого разряда в памяти); наличие развитых средств адресации для упрощения обработки информационных массивов, ведения списков оборудования; введение специальных команд для диагностики неисправностей; включение в состав комплекса аппаратуры контроля, модулей коммутации рабочих и резервных блоков.

Совершенствование физической структуры связных процессоров идет по следующим направлениям:

1. Увеличение числа универсальных регистров;
2. Использование ЭВМ с микропрограммным управлением, позволяющих построить специальную связную систему команд и микрокоманд;
3. Ускорение обработки информации за счет быстродействующей стековой памяти;
4. Введение в состав процессора развитой (многоуровневой) системы прерывания;
5. Обеспечение прямого доступа к памяти;

6. Использование мультипроцессорных структур связных процессоров.

Выбор физической структуры связного процессора обусловливается в основном требованиями производительности (число и пропускная способность КС, объем функций по дополнительной обработке) надежности, гибкости и возможности расширения системы.

Элементной базой связных процессоров первого поколения были ИС малой и средней степени интеграции. Такие связные миниЭВМ имели постоянную вычислительную мощность, которая для некоторых областей применений является избыточной, а следовательно, и увеличивает стоимость системы. Другим недостатком связных процессоров на базе минимашин является отсутствие возможности расширения системы — при росте нагрузки на процессор приходится менять тип минимашины, а следовательно, и все ПО. Для ряда связных процессоров при росте нагрузки оказывается невозможным расширение необходимого объема памяти.

Совершенствование структуры и повышение производительности МП стали основой появления связных процессоров второго поколения — на базе МП и широкого применения периферийных БИС. Рассмотрим те варианты физической структуры связных процессоров, которые являются типичными для современных систем и сетей связи. Известны три типа СП: однопроцессорный с реализацией многопрограммного режима, многомашинный и мультипроцессорный.

**Однопроцессорная структура.** Для построения связных процессоров невысокой производительности при отсутствии требований повышенной надежности используется структура с одним МП (рис. 2.8). В состав СП входят модули ЗУ (ОЗУ и ПЗУ), внешняя память (накопители на гибких магнитных дисках или кассетный накопитель на магнитной ленте), сопрягаемая с магистралью через периферийные адаптеры. Линии связи, работающие с различными скоростями и протоколами, сопрягаются с СП посредством блоков *связных адаптеров* (СА). Производительность СП во многом определяется сложностью функционального состава задач, выполняемых СА. В простейшем случае —это контроллеры, осуществляющие сопряжение с линиями, однако в настоящее время разрабатываются все более сложные БИС программируемых СА, обеспечивающие реализацию функций 2...4-го уровней протоколов. По такой однопро-

цессорной схеме выполнены, например, СП Datamax 6 и Compunet (с МП типа Z-80). Потребитель производит окончательное программирование для СП в зависимости от его назначения с записью программы в перепрограммируемое ПЗУ (ППЗУ). Производительность такой структуры СП ограничивается из-за конечной пропускной способности и конфликтов при обращении к магистрали.

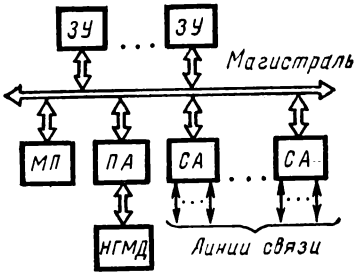



Рис. 2.8. Пример однопроцессорной структуры СП

Более распространенной является двухпроцессорная структура СП с функциональной специализацией МП (рис. 2.9). В такой системе $МП_1$ осуществляет общее управление, сложную логическую обработку информации, управление УВВ (НГМД, КНМЛ и т. д.). Связные функции локализованы в МП-комплексе, включающем $МП_2$, местное ЗУ (ОЗУ и ПЗУ) и СА. При таком построении СП уменьшается число обращений к главной магистрали, значительно увеличивается производительность. Например, СП Level 6 может обслуживать до 96 линий связи со скоростями от 50 бит/с до 72 Кбит/с при использовании различных протоколов (BSC, SDLC, HDLC и др.). Возможен вариант двухпроцессорной структуры с другим распределением функций между МП. Так, например, в СП Comdata 600 на $МП_2$ возлагаются задачи управления низкоскоростными линиями, а $МП_1$ кроме общего управления СП руководит также обменом по высокоскоростному каналу.

К недостаткам такой архитектуры относятся ограниченные надежность и возможности по наращиванию системы. Рассмотренная двухпроцессорная структура СП является промежуточным вариантом между однопроцессорной и многомашинными мультипроцессорными структурами.

**Многомашинные структуры.** К многомашинным системам (однородным и неоднородным) относят системы, состоящие из набора вычислительных модулей (ВМ), не имеющих общего поля оперативной памяти (ОПП) и обменивающихся между собой способом коммутации

сообщений. Типовые конфигурации многомашинных систем включают кольцевую, полносвязную, с общей шиной (рис. 2.10). Такая распределенная обработка и управление позволяют многомашинному СП достигнуть достаточно высокой пропускной способности. Отдельные машины выступают в роли автономных функциональных ВМ, реализуя разнообразные функции из множества $F$.

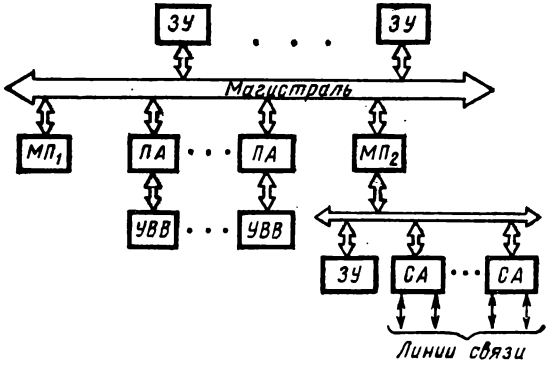



Рис. 2.9. Пример структуры СП на базе двух МП

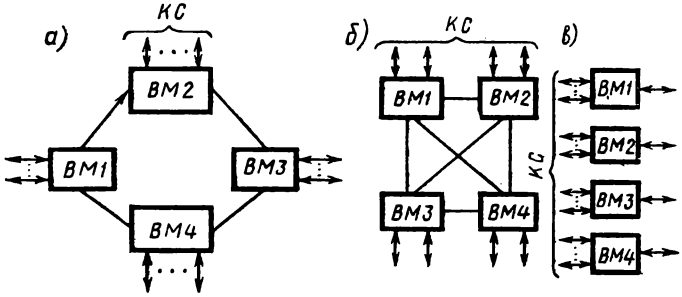



Рис. 2.10. Типовые конфигурации многомашинных систем:
*а* — кольцевая; *б* — полносвязная; *в* — с общей шиной

В таких системах время задержки, например пакета, зависит от числа переприемов, т. е. чем большей производительностью должен обладать СП, тем из большего числа ВМ он должен состоять и тем большее число переприемов будет осуществляться. Этот недостаток ограничивает число автономных микроЭВМ в структуре СП.

**Мультипроцессорные структуры.** Отличительным признаком мультипроцессорных систем является наличие общего ОПП для информационного взаимодействия

между ВМ в СП. При этом во всех случаях производится лишь однократная запись информации в ОПП.

На рис. 2.11 представлены типовые конфигурации мультимикропроцессорных структур СП.

*Одношинную структуру* (рис. 2.11, *а*) можно считать частным случаем системы с перекрестной коммутацией (рис. 2.11, *в*). при числе шин памяти, равном единице. Недостатком одношинной структуры является возможность возникновения «узкого места» при увеличении за-

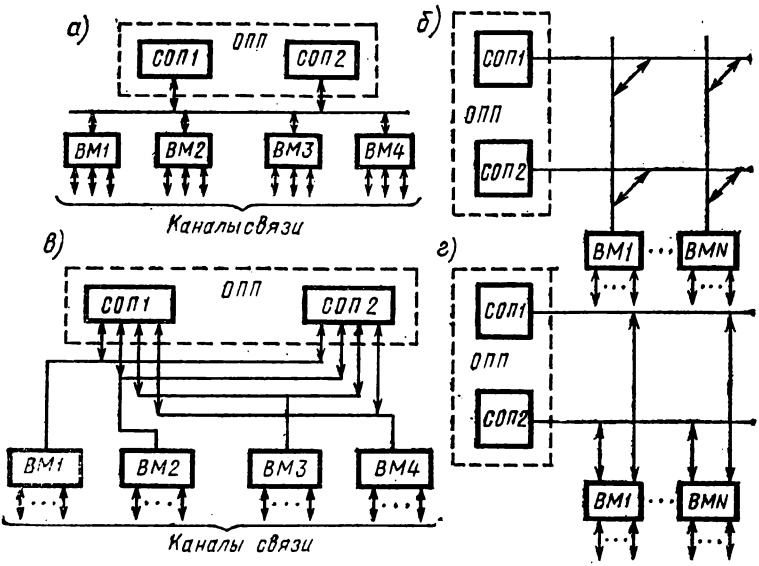



Рис. 2.11. Типовые конфигурации мультимикропроцессорных структур:

*а* — одношинная структура; *б* — структура с многовходовой памятью; *в*, *г* — структуры с перекрестной коммутацией

грузки шины, связанной, например, с ростом числа ВМ в системе, а также ограниченную надежность.

Недостатком систем *с многовходовой памятью* (рис. 2.11, *б*) является структурное ограничение — обычно небольшое число входов в секции оперативной памяти (СОП). Очевидно, что максимальное число ВМ, объединяемых в мультипроцессорную систему, в данном случае, равно числу входов в СОП.

Система *с перекрестной коммутацией* (рис. 2.11, *в*) структурно неограничена и имеет возможность (как и в

случае с многовходовой памятью) производить одновременный обмен несколькими ВМ с различными СОП.

Система с перекрестной коммутацией (рис. 2.11, *г*) отличается от системы (рис. 2.11, *в*) исполнением коммутаторов шин.

В мультимикропроцессорных СП используются структуры с прямым и функциональным типом организации. В *прямой* мультипроцессорной системе каждый микропроцессорный модуль может решать любые задачи, тогда как в *функциональной* мультипроцессорной системе микропроцессорный модуль способен выполнять лишь некоторый ограниченный набор функций.

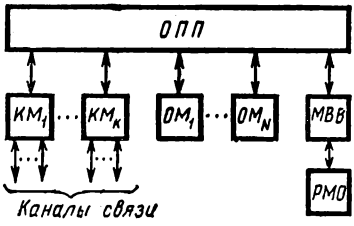



Рис. 2.12. Структура СП на основе функциональных модулей

Выше было отмечено, что СП состоит из трех подсистем: ПВК, ПОИ, ПВВ. Этим подсистемам в мультипроцессорных структурах соответствуют три группы аппаратно-ориентированных функциональных модулей (ФМ), реализующих функции соответствующих подсистем: *канальные* ФМ (КМ), часто называемые линейными процессорами (ЛП), *обрабатывающие* ФМ (ОМ), ФМ *ввода-вывода* (МВВ).

На рис. 2.12 приведена структура СП на основе функциональных модулей. Все ФМ взаимодействуют с ОПП на уровне аппаратных средств (физического протокола взаимодействия ФМ с ОПП), операционной системы (логического протокола) и функциональных программ (информационного протокола). Сами ФМ могут быть построены на основе микроЭВМ, содержащей процессор, локальную память (оперативную и постоянную), аппаратные средства сопряжения с ОПП, а также дополнительные устройства для сопряжения с дискретными каналами (в случае КМ) либо для сопряжения с УВВ (в случае МВВ).

Для обеспечения работы с высокоскоростными каналами связи КМ могут строиться как каналы прямого доступа, реализующие пересылку блоков информации между КМ и ОПП в режиме прямого доступа.

Таким образом, каждый ФМ осуществляет аппаратную поддержку функций, реализуемых соответствующей

подсистемой СП. При этом МВВ выполняет функции подмножества $F_T \cup F_O$, ОМ — функции подмножества $F_C \cup F_S \cup F_K$, КМ выполняет функции подмножества $F_N$.

Рассмотрим несколько примеров мультимикропроцессорных СП.

**СП Codex 6000.** Структура СП Codex 6000 является развитием двухпроцессорной магистральной системы, рассматривавшейся выше (см. рис. 2.9). СП может содержать до 8 линейных процессоров (типа М6800), обес-

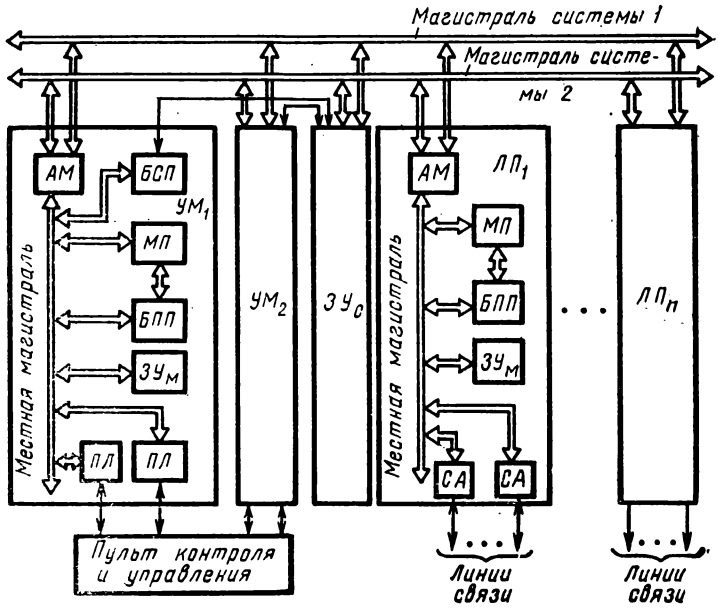



Рис. 2.13. Структура СП «Тезис-5»

печивающих обслуживание 252 линий со скоростями 50—9600 бит/с. Кроме того, в системе выделен центральный МП (типа Intel 3000), выполняющий задачи общей диспетчеризации, управления внешними устройствами, реконфигурации. По своей производительности данный СП эквивалентен возможностям двух миниЭВМ PDP — 11/40.

**СП «Тезис-5».** Мультимикропроцессорная система «Тезис-5» разработана фирмой «CTNE» для оснащения сети RETD (Испания). Данный СП выполнен по модульному принципу и отличается повышенной надежностью.

Структура СП магистральная, с двумя системными магистралями, позволяющими производить обмен со скоростью 2,5 М 16-разрядных слов в секунду (рис. 2.13). В целях повышения надежности в СП используются два управляющих модуля (УМ), выполненных на базе МП Intel 8086. Каждый УМ содержит местную магистраль, к которой подключаются МП, БПП, местное ЗУ ($ЗУ_м$) емкостью до 128 Кбайт, блок связи с общей памятью системы (БСП) и периферийные адаптеры (ПА). Управляющий модуль сопрягается через ПА с пультом конт-

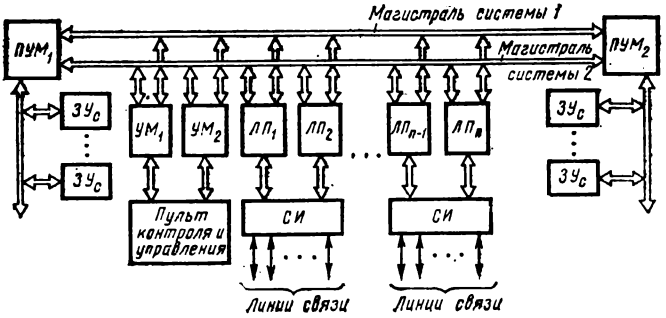



Рис. 2.14. Структура СП СР 9000

роля и управления СП, а через адаптеры магистрали (АМ) — с магистралями системы.

Общая память СП ($ЗУ_с$) может наращиваться блоками по 64 Кбайт до емкости в 1 Мбайт. Линейные процессоры (ЛП) по своей структуре аналогичны УМ. Каждый ЛП может осуществлять управление и обработку информации по 32 асинхронным линиям (50—1200 бит/с), либо по 16 синхронным линиям (600—19 200 бит/с), или же по двум линиям (64 Кбит/с, протокол HDLC).

Число ЛП в СП может наращиваться до 32, позволяя тем самым подключать к СП до 1024 линий связи. Таким образом, максимальное число МП в данном СП равно 34.

**СП типа СР 9000.** Как отмечалось выше, одним из важнейших требований к связным устройствам является повышенная надежность. Для обеспечения высокой надежности в СП магистральной структуры СР 9000 применяется не только дублирование магистралей и УМ, но и линейных процессоров. Каждая группа линий связи подключается через блок связного интерфейса (СИ) к двум ЛП (рис. 2.14). Управление магистралями систе-

мы осуществляется процессорами управления магистралью (ПУМ), выполняющими функции интеллектуального арбитра и обеспечивающими доступ к общим ЗУ системы. Как и в СП «Тезис-5», в данном случае каждый УМ и ЛП обладает собственной местной памятью. Максимальный объем местной и общей памяти может превышать 1 Мбайт. Система является модульно наращиваемой, и в максимальном варианте число МП составляет 64. Линейные процессоры могут обслуживать как асинхронные, так и синхронные линии связи.

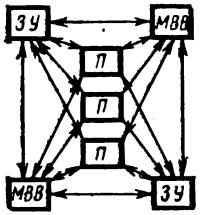



Рис. 2.15. Упрощенная схема взаимосвязи функциональных модулей в СП Pluribus

**СП Pluribus.** Примером мультимикропроцессорной системы с прямым типом организации обработки заданий может служить СП Pluribus.

Связной процессор, построенный по модульному принципу состоит из однотипных процессорных модулей (П), имеющих доступ ко всем ресурсам системы и способных выполнять любые части системных и прикладных программ. Процессорные модули объединяются с несколькими модулями памяти и ввода-вывода. Это позволяет обеспечить необходимую живучесть за счет введения избыточных блоков и гибкость, т. е. возможность компоновки системы с учетом специфики применения. В СП применяется единая система адресации: для всей памяти и устройств ввода-вывода используется единое адресное поле.

СП Pluribus содержит до 13 процессорных блоков, 2 модуля ЗУ и 2 модуля ввода—вывода (МВВ). Каждый модуль П связан со всеми модулями ЗУ и МВВ и каждый модуль ЗУ с МВВ (рис. 2.15). Система обладает высокой производительностью — до 1,5 Мбит/с, модульностью, возможностью расширения при возрастании нагрузки.

## § 2.4. ПРОГРАММИРУЕМЫЕ АБОНЕНТСКИЕ ПУНКТЫ

Абонентские пункты являются наиболее многочисленным по объему классом устройств системы и сетей передачи информации АСУ. Современные информационно-вычислительные сети рассчитаны на подключение к ним тысяч и десятков тысяч АП. Существующие АП различаются по множеству признаков — областям при-

менения, способам реализации функций управления, скорости и способу передачи данных, составу и количеству устройств ввода—вывода, способу взаимодействия с ЭВМ и т. д.

Преобладающую роль в современных системах и сетях передачи информации играет класс программируемых АП, к которому относятся устройства, работающие под управлением запоминаемых программ. Программное управление АП, наличие на АП некоторой вычислительной мощности позволяют не только улучшить его параметры, но и реализовать ряд новых операций.

В зависимости от возможностей программного управления выделяют АП *с жесткой программой, перепрограммируемые* АП и *интеллектуальные* АП. Первая группа характеризуется наличием неизменяемых в процессе эксплуатации программ управления АП, размещаемых в ПЗУ. Такие АП имеют жесткую аппаратную и программную структуру. Более гибкой модульной организацией отличаются перепрограммируемые АП, в которых возможна полная или частичная замена управляющих программ за счет смены модулей ПЗУ или же изменения содержимого перепрограммируемого ПЗУ. Наиболее же перспективен класс интеллектуальных АП, которые характеризуются наличием свободной вычислительной мощности и значительным объемом ОЗУ, что позволяет выполнять большой объем функций по передаче и обработке данных. Подобные АП допускают адаптацию к различным областям применения и требованиям пользователя. Возможно решение на АП некоторых программ пользователя.

Рассматривая физическую структурную организацию программируемых АП, можно выделить три основных аспекта. Во-первых, в зависимости от количества обрабатываемых модулей выделяются одно- и мультимикропроцессорные АП. Усложнение набора функций и снижение стоимости МП привели в настоящее время к тенденции использования мультимикропроцессорных АП. Вторым важным элементом структуры АП является набор внешних ЗУ (ВЗУ). Простейшие типы программируемых АП не имеют в своем составе ВЗУ. Более совершенные модели включают накопители на гибких магнитных дисках или кассетные накопители на магнитной ленте. Наличие ВЗУ значительно расширяет функциональные возможности АП. Третьей существенной структурной характеристикой АП служит набор используемых УВВ.

По этому признаку выделяют печатающие, дисплейные алфавитно-цифровые, дисплейные графические АП и АП с комбинированным составом УВВ. В свою очередь, например, печатающие АП подразделяются на устройства с матричной головкой, с цилиндрической и шаровой головками, струйной печати и т. д.

В зависимости от способа взаимодействия с другими терминалами и ГВМ в ИВС выделяют интерактивные (диалоговые) АП и устройства пакетной обработки. В современных ИВС преобладающим является диалоговый режим взаимодействия, поэтому практически все выпускаемые типы программируемых АП относятся к интерактивному типу или являются комбинированными с возможностью пакетной обработки.

Программное управление позволяет достаточно просто организовать одновременную работу нескольких операторов на АП. Такие групповые программируемые АП часто определяют как терминальные комплексы.

Можно также отметить, что в ИВС преобладают среднескоростные программируемые АП, использующие дуплексный режим обмена и буферированный ввода—вывода информации.

Появление МП и микроЭВМ явилось толчком к резкому росту выпуска и увеличению числа моделей АП. В программируемых АП находят применение практически все выпускаемые в настоящее время типы универсальных МП. Широкое использование МП и БИС в АП объясняется тем, что по сравнению с аппаратными вариантами исполнения внедрение микроэлектронных модулей позволяет сократить число компонентов АП на 70 % и более, число печатных плат — до 1...2, значительно уменьшается стоимость, габариты и потребляемая мощность и при этом примерно на порядок повышается надежность.

Характеристики МП являются определяющими для таких параметров АП как предельная скорость обмена по каналу связи, объем выполняемых функций, надежность, потребляемая мощность и т. д. Повышение быстродействия МП, расширение и специализация набора команд, увеличение разрядности, появление новых БИС в составе МПК непосредственно отражаются на параметрах выпускаемых АП. Несмотря на появление 16-разрядных МП и микроЭВМ все же преобладающую роль в АП продолжают играть 8-разрядные МП. На рис. 2.16 приведена типичная структура программируемого АП на

базе 8-разрядного МПК. В состав АП кроме БИС МП входят БИС ОЗУ и ПЗУ, БИС контроллера прямого доступа к памяти (ПДП), а также БИС периферийных и связных адаптеров. Разработанные в настоящее время для основных МПК ПА связи с клавиатурой управления дисплеем, управления НГМД, СА для функций управления информационным каналом и ряд других, позволяют значительно разгрузить МП и увеличить объем выполняемых АП-функций.

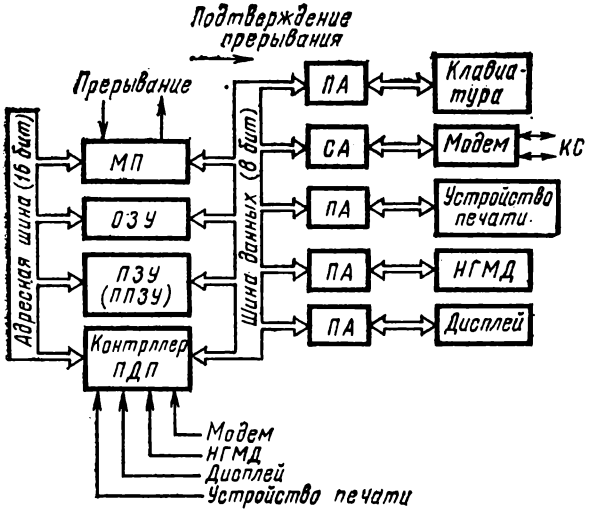



Рис. 2.16. Типичная структура программируемого АП на базе 8-разрядного МПК БИС

Конфигурация программируемого АП допускает простое модульное расширение. Так, например, АП модели DT-9621 включает в основном наборе алфавитно-цифровой дисплей на 1920 символов, ОЗУ емкостью 256 Кслов, устройство печати, 16-разрядный МП, 2 НГМД и обеспечивает обмен по КС со скоростью 9600 бит/с. Допускается расширение состава АП такими УВВ, как КНМЛ, устройства ввода с перфоленты и т. д.

Рассматривая программируемый АП как элемент ИВС с типичным набором функций (см. § 2.2), можно отметить, что специфика функциональной структуры АП заключается в более широком спектре терминально-пользовательских функций $F_T$. Важнейшими из этих функций являются:

1. Редактирование вводимых данных. Для этой цели

в АП предусматриваются: стирание информации в любом месте экрана дисплея, сдвиги информации на экране дисплея, запись информации на любое место экрана, выделение и работа с отдельными полями данных.

2. Контроль вводимых данных: проверяется использование при вводе разрешенных символов (слов выражений), контролируется полнота ввода данных в соответствующий формат.

3. Управление изображением: регулировка яркости, изменение цвета, использование «мигающих» символов, подчеркивание символов, протяжка кадра, растяжение и сжатие отдельных полей, работа со световым пером и т. д.

4. Уплотнение данных с целью уменьшения информационной избыточности источника информации. Для этого применяются методы: кодирования слов (выражений); преобразование данных с использованием малоизбыточного кода, замена последовательности пробелов специальным символом с указанием числа пробелов и др.

5. Служебный диалог с оператором, заключающийся в выдаче сообщений об ошибках при вводе данных, об отказах и сбоях аппаратуры, вывод отметок времени приема и выдачи сообщений, включение звуковой и световой сигнализации, вывод на экран конструкций по работе с системой.

6. Управление режимом работы АП: переключение УВВ, скоростей и режимов обмена с отображением результатов коммутации на пульте управления АП.

7. Защита от несанкционированного доступа, включающая такие средства, как применение паролей и кодов пользователей, ведение таблиц разрешения доступа, регистрация доступа к АП.

8. Смена наборов символов, которая позволяет использовать различные первичные коды и наборы функциональных символов.

Как уже отмечалось, относительно физической структуры программируемые АП относятся к общему классу связных процессоров. Вместе с тем анализ существующих моделей АП позволяет утверждать, что при построении микропроцессорных АП преобладающее распространение получила магистральная структура. При этом большинство АП строятся на основе одиночного МП, значительно реже встречаются двух- и трехпроцессорные магистральные структуры.

В мультимикропроцессорных АП используется функциональный тип организации системы, т. е. на каждый МП возлагается выполнение некоторого подмножества полного множества функций $F$. По признаку однородности используемых МП в АП преобладают структуры с однородным типом модулей. Это объясняется тем, что хотя применение неоднородных структур (с различными типами МП) и позволяет более полно загрузить процессоры, но при этом значительно увеличивается сложность и стоимость разработки программного обеспечения и аппаратного сопряжения.

## § 2.5. ПРОГРАММИРУЕМЫЕ КОНЦЕНТРАТОРЫ И МУЛЬТИПЛЕКСОРЫ

Чтобы решить задачу транспортировки данных, базовая сеть связи использует методы коммутации и передачи. С этим связаны два родственных понятия: концентрация и уплотнение. Они могут осуществляться различными способами и не всегда явно отличаются друг от друга.

Обычно при проектировании СОИ сталкиваются с тем, что абоненты не распределяются равномерно на большой территории, а сосредотачиваются группами в здании учреждения, городе или промышленном районе. Когда группа соединений идет к центральному коммутатору (ЦК), находящемуся на некотором расстоянии (рис. 2.17, *а*), то линии используются неэкономно, так как только небольшая часть линий (порядка 10 %) будет активна одновременно. Более рационально поместить вспомогательный коммутатор вблизи группы пользователей, что позволит уменьшить число линий, идущих к центральному коммутатору (рис. 2.17, *б*). Эти вспомогательные коммутаторы могут быть как концентраторами (КЦ), так и мультиплексорами (МПЛ). Концентрация определяется как динамическая процедура распределения нагрузки, которая часть входных (активных) каналов в соответствии с запросами распределяет между меньшим числом выходных каналов. Уплотнение обычно определяется как статическая процедура, при которой группа линий заменяется одним трактом передачи с более высокой пропускной способностью. Уплотненный канал обеспечивает ресурс — часть частотного спектра или времени занятости, предоставляя его в фиксированном, заранее заданном порядке. Таким образом,

мультиплексор (уплотнитель) имеет одинаковые скорости ввода и вывода нагрузки и не содержит риска перегрузки выходного канала. Концентрация же включает эффект сглаживания трафика, но влечет за собой риск перегрузки (число входных активных каналов может превышать число выходных).

В мультиплексорах для объединения отдельных потоков используется метод частотного или временного разделения каналов (ЧРК и ВРК). Метод ЧРК не является цифровым и не использует новые микропроцессоры,

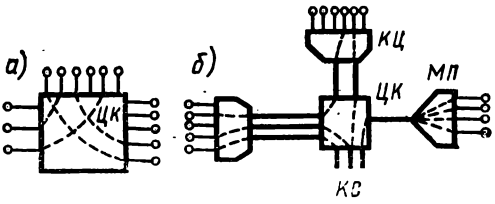



Рис. 2.17. Коммутация, концентрация и мультиплексирование каналов:

*а* — коммутация без концентрации и мультиплексирования; *б* — коммутация с концентрацией и мультиплексированием

ЗУ и средства математического обеспечения. В противоположность этому при временном разделении, которое может быть побитовым, позначным, метод является цифровым. Мультиплексоры этого типа обрабатывают несколько цифровых входных потоков и преобразуют их в суммарный выход.

Таким образом, *мультиплексором* в электросвязи является устройство уплотнения, которое производит только уплотнение данных без записи их в память или изменения содержимого или формата.

По определению [17], *концентратором* является электронно-вычислительная система, которая осуществляет комбинирование некоторого числа линий связи в меньшее число с более высоким быстродействием или в меньшее число линий связи с тем же быстродействием, но с более высоким уровнем использования по сравнению с уровнем использования отдельных линий связи. Концентратор по своих функциям сложнее, чем мультиплексор, так как концентратору приходится менять структуру сообщения, скорость передачи, иногда коды. Концентраторы, соединяющие КС с различным быстродействием, выполняют одновременно и функции буфера.

При применении временного мультиплексирования следует различать два случая: 1) мультиплексор передачи данных как многоканальное устройство сопряжения каналов связи с ЭВМ, задачей которого является согласование относительно низкой скорости передачи информации по абонентским каналам связи с высокой скоростью ввода—вывода информации ЭВМ; 2) мультиплексор-уплотнитель, основной задачей которого является временное уплотнение группового канала связи. В пер-

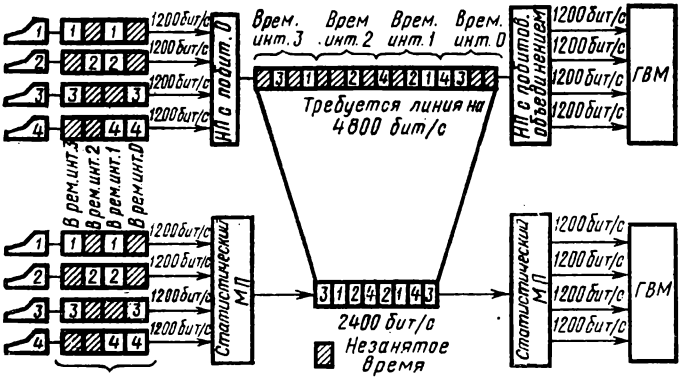



Рис. 2.18. Временное и статистическое мультиплексирование

вом случае мультиплексор уплотняет селекторный канал ЭВМ, во втором случае (рис. 2.17, *б*) мультиплексор уплотняет КС, который соединяет удаленные пользователи с ближайшим ЦК.

Преобладающая роль в современных СОИ программируемых устройств, применение МП позволили реализовать в мультиплексорах ряд новых операций, в том числе характерных прежде только для концентраторов. Различие между мультиплексированием и концентрацией становится менее явным при введении принципа «мультиплексирование по требованию», который реализуется так называемыми *статистическими мультиплексорами* (мультиплексорами—концентраторами). Статистическое объединение отличается от ВРК, поскольку источники информации не выдающие сигналов, полностью игнорируются (рис. 2.18).

Статистические мультиплексоры рассчитаны на использование самой распространенной ситуации в процес-

се передачи данных — наличия пауз между знаками или передачами. Паузы динамически учитываются микропроцессором и используются для предоставления пропускной способности выходного тракта активным каналам. При таком методе длина цикла не постоянна и определяется периодически в соответствии с активностью входов мультиплексора. В буферном ЗУ, которое хранит в себе поступившие данные до их передачи, также накапливается копия передаваемого цикла на случай его повторной передачи, если потребуется.

Статистическое объединение потоков данных — метод, позволяющий увеличивать число подключенных к временному мультиплексору терминалов, не повышая скорости передачи на тракте между потребителем и мультиплексором.

При использовании статистического объединения потоков передаваемая полоса частот предоставляется только активным терминалом. Когда пропускная способность тракта меньше предлагаемой нагрузки, входящая нагрузка накапливается в буфере и ставится в очередь под управлением МП мультиплексора. Она находится в памяти с произвольной выборкой до тех пор, пока не появится возможность передать эту информацию. В этом смысле статистические мультиплексоры мало чем отличаются от концентраторов с микропроцессорной обработкой и управлением.

Практически реализация того или иного набора функций определяется назначением и требованиями к характеристикам устройства концентрации нагрузки (УКН). На рис. 2.19 приведены типичные случаи включения УКН в сеть. Обозначив через $\bar{r}/L$ отношение средней длины линии от абонента до УКН (концентратора, мультиплексора), где $\bar{r}$ — средний радиус зоны, обслуживающей УКН, к расстоянию $L$ от УКН до ЦК, выделяют: абонентские (терминальные) концентраторы и мультиплексоры ($\bar{r}/L \ll 1$); удаленные (линейные) концентраторы и мультиплексоры ($\bar{r}/L \approx 0{,}1...0{,}3$); встроенные (стативы) концентраторы и мультиплексоры ($L \approx \approx 0$), которые входят в состав ЦК.

Характерным для мультиплексоров и концентраторов является переход от аппаратной логики к программной, применение в качестве мультиплексоров и концентраторов связных процессоров. На рис. 2.20 приведена типичная структура программируемого мультиплексора с МП

управлением. В состав связного процессора мультиплексора входят БИС МП, БИС ОЗУ и ПЗУ, БИС контроллера прямого доступа к памяти (ПДП), БИСы связных низкоскоростных адаптеров ($СА_{н1}$...$СА_{нk}$), обеспечивающих сопряжение магистрали и контроллера ПДП с низкоскоростными (абонентскими) КС, и БИС связного группового адаптера $СА_г$, выполняющего ту же функцию по отношению к групповому КС. На быстродействие такой МП-системы при выполнении отдельных функций

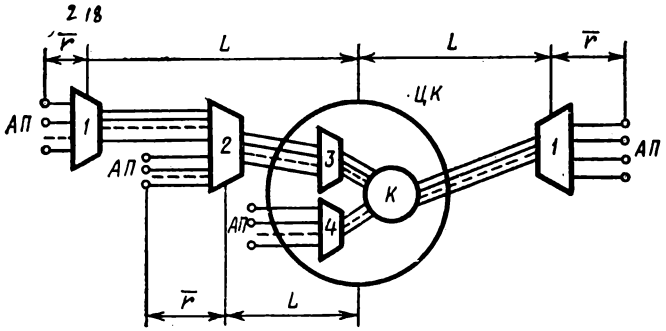



Рис. 2.19. Типовые случаи включения УКН в СОИ:

*1* — терминальное УКН; *2* — линейное УКН; *3* — линейный статив-концентратор; *4* — статив-концентратор; *К* — коммутатор

оказывает влияние несоответствие разрядности процессора и обрабатываемых кодовых слов. Например, поиск синхросигнала требует выполнения логических операций над кодами разрядности 20...30 бит, что приводит к последовательно-параллельной обработке на 8...16-разрядных МП. Увеличение разрядности МП при использовании, например, секционных процессорных элементов типов К584, К589 является нецелесообразным, поскольку некоторые другие функции используют операнды значительно меньшей разрядности (6...8 бит). Поэтому обычно программная реализация сочетается с аппаратной реализацией функций наиболее критичных по времени выполнения. В случае необходимости приведенная структура может быть дополнена соответствующими БИС для связи с пультом оператора или высокоскоростными НГМД.

Примером мультимикропроцессорной системы, применяемой в качестве концентратора, может служить связной процессор СР 9000, рассмотренный в § 2.3,

структурная схема которого представлена на рис. 2.14. Структура может включать в себя три типа линейных процессоров: для поддержки асинхронных терминалов, синхронных байт-и бит-ориентированных протоколов.

Асинхронные ЛП могут поддерживать до 8 линий со скоростями от 50 до 9600 бит/с; байт-синхронные ЛП — до 4 линий со скоростями от 2,4 до 56 Кбит/с; бит-синхронные ЛП — до 8 линий со скоростями до 64 Кбит/с. При наличии полного комплекта ЛП (62 шт.) к одному СП типа СР 9000 можно подключить 480 линий.

В настоящее время, например, на базе микроЭВМ «Электроника НЦ-80-01Д» создан электронный телеграфный концентратор, предназначенный для автоматической коммутации и обработки телеграмм в областных и крупных районных узлах телеграфной сети общего пользования. Схема связи электронного телеграфного концентратора обеспечивает телеграфный обмен оконечных пунктов своей зоны с ЦКС. Максимальная конфигурация электронного телеграфного концентратора включает 8 модулей на 128 каналов с пропускной способностью 0,7 телеграмм/с. Структура электронного телеграфного концентратора представлена на рис. 2.21.

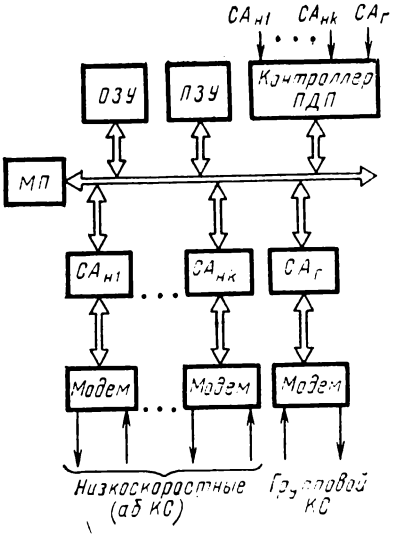



Рис. 2.20. Структура программируемого мультиплексора с МП-управлением

Специфика функциональной структуры программируемых мультиплексоров и концентраторов как УКН заключается в расширении спектра сетевых функций $F_N$. Отличительной особенностью связных процессоров, применяемых в качестве УКН, является частичная обработка сообщений, включающая следующие функции: реализацию различных алгоритмов управления доступом абонентов в базовую сеть в зависимости от типа подключенного АП; концентрацию сообщений, их анализ, контроль, обработку и передачу соответствующему абоненту;

управление работой отдельных АП; ПД с различными скоростями; коммутацию сообщений; контроль правильности принимаемой информации; накопление, буферизацию, хранение данных и различные функции управления сетью.

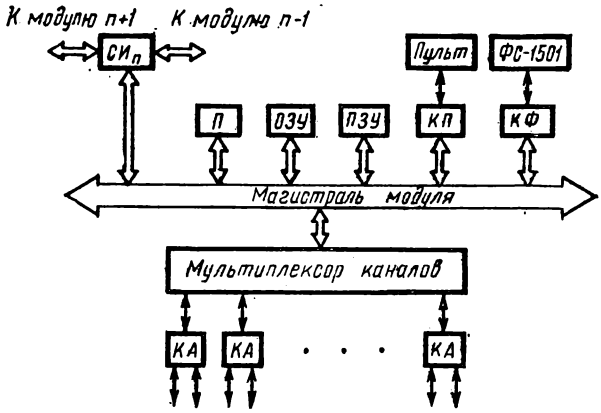



Рис. 2.21. Структура электронного телеграфного концентратора

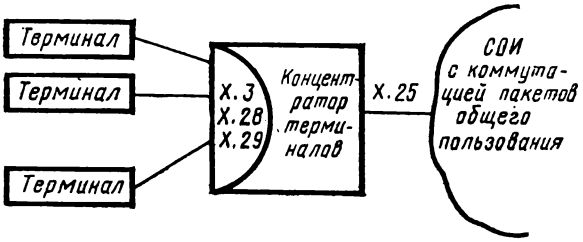



Рис. 2.22. Структура терминальной части СОИ с коммутацией пакетов

Современным подходом к решению проблемы выхода из сети с пакетной коммутацией является реализация мультиплексоров и концентраторов, совместимых со стандартом *X*.25. Так для СОИ с коммутацией пакетов мультиплексор для подключения источников и приемников данных различных типов определен МККТТ как схема объединения — разъединения пакетов (ОРП). Для задания интерфейсов ОРП МККТТ разработаны рекомендации *X*.3, *X*.28 и *X*.29. Устройство ОРП выполняет функции *концентратора терминалов* (рис. 2.22), через

который несколько асинхронных терминалов подключаются к одному КС, работающему в соответствии с рекомендациями *X*.25. Концентратор объединяет последовательности битов, поступающие от терминалов в пакеты, и передает их в ЦКП по виртуальному каналу. При выдаче информации непакетному терминалу концентратор расшифровывает пакеты и передает информацию позначно. Важной функцией КЦ является накопление информации перед ее выдачей в быстродействующий КС — *буферизация данных*. В связи с этим для концентраторов типична комплектация ЗУ большой емкости. На концентраторы и статистические мультиплексоры в СОИ возлагаются в общем случае реализации ряда уровней протоколов. Среди них большую роль играют протоколы уровня образования информационного канала (HDLC, SDLC, BSC и др.). В пределах реализуемых этими устройствами протоколов можно выделить и более мелкие функциональные блоки — защита от ошибок, генерирование заголовков, подтверждение приема пакетов и сообщений, реализация приоритетности сообщений, контроль и управление соединениями и др.

Анализ функциональной и физической структур устройств концентрации нагрузки позволяет сделать следующие выводы.

Наблюдается тенденция усложнения функциональной структуры концентраторов и мультиплексоров, включения в состав задач этих устройств функций по преобразованию протоколов, сборке — разборке пакетов, диагностике, контролю КС.

Стандартизация в области протоколов обмена информацией в значительной степени затронула концентраторы и мультиплексоры СОИ, при этом преобладающее число новых моделей рассчитано на реализацию протоколов *X*.25/3, *X*.25/2 по высокоскоростным КС, выполнение функций ОРП, преобразование данных, поступающих по низкоскоростным КС в стандартную форму, определяемую рекомендацией *X*.25 МККТТ.

Совершенствование МП и микроЭВМ отразилось как на структуре, так и на характеристиках концентраторов и мультиплексоров СОИ. Расширяется использование мультипроцессорных структур, широко внедряются специализированные БИС для реализации линейной части концентраторов и мультиплексоров.

Стандартизация процедур обмена информацией ведет к унификации аппаратной части устройств СОИ. Выпус-

каемые в последние годы связные процессоры имеют модульную структуру и наращиваемое программное обеспечение, что позволяет их использовать в качестве концентраторов, мультиплексоров и других устройств СОИ.

## § 2.6. ЦЕНТРЫ КОММУТАЦИИ

Основная функция коммутации состоит в том, чтобы дать доступ одному абоненту сети ко всем другим. Эта функция контролируется самим пользователем, когда он указывает адрес сообщения. Коммутация является средством, с помощью которого сообщение или запрос соединения направляются к адресату. Распределение потоков сообщений осуществляется в центрах коммутации различного уровня с помощью соответствующих систем кроссирования и коммутации. Центры коммутации представляют собой пункты сети, в которых сходятся три и более ветвей (магистралей, пучков) и имеются устройства распределения входящих потоков информации по направлениям (ветвям). Центры делятся на *транзитные* и *оконечные*; первые коммутируют магистральные КС, связывающие между собой различные ЦК, а вторые производят соединения магистральных каналов с абонентскими линиями. Различают ЦК *каналов* (ЦКК), *сообщений* (ЦКС) и *пакетов* (ЦКП). В последние годы интенсивно развиваются гибридные методы коммутации, сочетающие в центрах КК и КП.

Для центров коммутации состав подмножества $F_N$ во многом определяется типом ЦК. Так, для ЦК каналов основными элементами подмножества $F_N$ являются функции, связанные с обработкой информации о вызовах и соединениях, управлением работой коммутационного поля, организацией взаимодействия с другими ЦК каналов. Для ЦК сообщений и пакетов специфичными являются функции, связанные с распределением входящих потоков с минимальной задержкой и минимальной ошибкой в центре в направлении, определяемом адресом сообщения, его приоритетом и величиной действующей нагрузки сети, при выборе которого имеет место наибольшая пропускная способность сети при минимальной задержке в передаче сообщений по КС. Для ЦК пакетов элементы множества $F_N$ связаны с реализацией ряда уровней протоколов — обычно физического, канального и сетевого. В соответствии с этими протоколами ЦКП осуществляет прием сигналов из каналов связи, выделение и накопле-

ние в оперативной памяти информационных кадров (ИК), логическую обработку ИК и содержащихся в них пакетов различного назначения (данных, речи, служебных и т.д.), маршрутизацию пакетов по исходящим направлениям связи и передачу ИК из ОП в исходящие каналы связи. Один и тот же ЦКП может оказаться для одних пакетов оконечным, для других—транзитным. Поэтому в ЦКП обычно реализуются функции сопряжения сетевых протоколов межцентрового и абонентского участков.

Для сетей с КП, обеспечивающих для передачи режим виртуального канала, специфическим является процесс сборки-разборки пакетов. Процессы сборки-разборки осуществляются в ЦКП и выполняются с помощью протоколов сетевого (пакетного) уровня, например 3-го уровня протокола $X_t25$, рекомендованного МККТТ. Разборка сообщения на пакеты в узле-отправителе и его сборка узлом-получателем должны обеспечивать адекватный прием сообщения, т. е. упорядочить принятые пакеты таким образом, чтобы получить ту очередность, которая была в исходном (переданном) сообщении. Алгоритмы сборки-разборки существенно зависят от принятых процедур маршрутизации и управления потоками на сети.

Основной функцией ЦК является осуществление коммутации. Коммутация информационных единиц (сообщений, пакетов, кадров, битов) во многом определяет эффективность работы сети в целом. Практическое распространение получили два метода коммутации — пространственный и временной. Пространственное разделение (абонентов) является исходным. В процессе передачи оно может преобразовываться в частное, временное и пр. Для современного уровня техники наиболее распространенным является временной метод коммутации: синхронный и асинхронный. При *синхронном* методе элементы информации (биты, байты), соответствующие каждому каналу, занимают в групповом тракте фиксированный временной интервал. Процесс коммутации сводится к изменению временного положения канала. Коммутация осуществляется через ЗУ. Число каналов, которое может обслужить синхронный коммутатор, тем больше, чем короче длительность цикла памяти «запись — чтение». При *асинхронном* методе коммутации происходит переадресация значащего момента (ЗМ): вместо адреса входящего канала ЗМ в ЦК присваивается адрес исходящего канала. Для этого

в ЦК имеется ЗУ, в котором за каждым каналом закреплена своя область памяти. Под управлением сигнала набора в нее записывается код исходящего канала. При обнаружении в каком-либо канале ЗМ устройство управления дает сигнал на считывание информации по адресу исходящего канала, в котором специальным устройством формируется ЗМ. Значащий момент является характеристическим моментом модуляции либо восстановления, в котором знак посылки изменяется на противоположный.

Метод асинхронной временной коммутации является универсальным. При КК адрес вызываемого абонента передается по всему пути на сети, в результате чего до начала передачи сообщения образуется сквозной канал. В системе с КС и КП адрес передается в составе каждой накапливаемой части сообщения, и на основе его анализа в каждом ЦК последовательно выбирается направление дальнейшей передачи. Так как при коммутации ЗМ, битов или байтов нецелесообразно каждой короткой части сообщения присваивать адрес получателя, в таких системах адрес передается предварительно и именно такие системы следует относить к КК. В системах с запоминанием более длинных частей сообщения (пакетов, сообщений) адрес передается в составе каждой из этих частей. Отметим, что асинхронный метод коммутации наиболее естественным образом сочетается с концентрацией. Концентраторы в ЦК позволяют более экономично строить устройства, распределяющие информационные потоки по адресам.

В состав ЦК входят: каналообразующая аппаратура (КОА); система распределения (СР), состоящая из кросса (статическая СР) и пространственного для КК или пространственно-временного для КП, КС, или гибридного коммутатора (динамическая СР), системы контроля (СК), устройства управления (УУ), а в отдельных случаях обрабатывающий процессор (Пр), используемый для первичной обработки сообщений в СОИ, например сжатия.

Раскрывая по выполняемым функциям элементы структуры, можно установить совпадение КОА и СР для ЦК и УКН, но различие УУ. Различие УУ ЦК и УКН состоит, во-первых, в отсутствии в УУ УКН блока режима распределения, который предназначен для организации обходных путей передачи информации. Во-вторых, управление выбором линии передачи для УКН осуществ-

ляется только для абонентских линий по отношению ко всем линиям в ЦК. В-третьих, УУ терминального УКН может содержать устройство управления скоростью аналого-цифрового преобразования (при наличии общего для терминалов АЦП), несвойственного ЦК. Опуская последнее, можно говорить, что УКН является частным случаем ЦК по сходству блоков и функций.

В зависимости от производительности и объема функций структуру ЦК и УКН можно реализовать по одному из вариантов: с программной реализацией функций универсальной или специализированной ЭВМ; с программной реализацией функций неоднородной вычислительной системой на базе специализированных программных автоматов (контроллеров) и универсальной ЭВМ; с программной реализацией функций однородной вычислительной системой на базе микроЭВМ и микропроцессоров; с аппаратно-программной реализацией, когда в сочетании с аппаратными средствами реализуется одна из вышеперечисленных программных реализаций.

Выбор того или иного варианта реализации ЦК и УКН зависит от конкретных требований к передаче сообщений, от производительности микроЭВМ. Так, например, СП ЦК и УКН в сети с КП должен обладать ступенчатым изменением коммутационной производительности и канальной емкости, параллельностью выполнения нескольких процессов в реальном масштабе времени, возможностью реконфигураций ТС и ПО при отказе отдельных элементов системы, высокой верностью обработки информации, возможностью работы с широким классом аппаратуры передачи данных и вводно-выводных устройств.

С точки зрения физической структуры СП выделяют одно- и многомашинные и многопроцессорные УКН и ЦК, варианты которых были рассмотрены выше (см. § 2.2; 2.3).

Обычная широко распространенная структура коммуникационной системы (рис. 2.23) состоит из центральной ЭВМ и микропроцессорных адаптеров, соединенных друг с другом шиной. Адаптеры здесь выполняют функции канального и физического уровней иерархии протоколов, а центральная ЭВМ реализует задачи сетевого уровня.

Работа рассматриваемой коммуникационной системы происходит следующим образом. Адаптер, получая из канала ПД кадр, вынимает из него пакет и через шину

передает центральной ЭВМ, где на основе таблицы маршрутов определяется канал, по которому этот пакет необходимо направить дальше. После этого через шину пакет передается соответствующему адаптеру. Здесь пакет упаковывается в кадр и направляется по указанному таблицей каналу ПД.

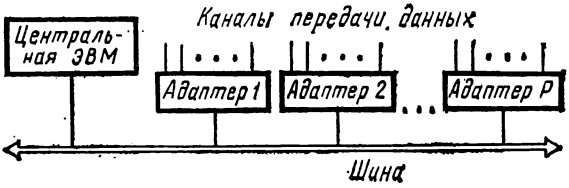



Рис. 2.23. Централизованная коммуникационная система

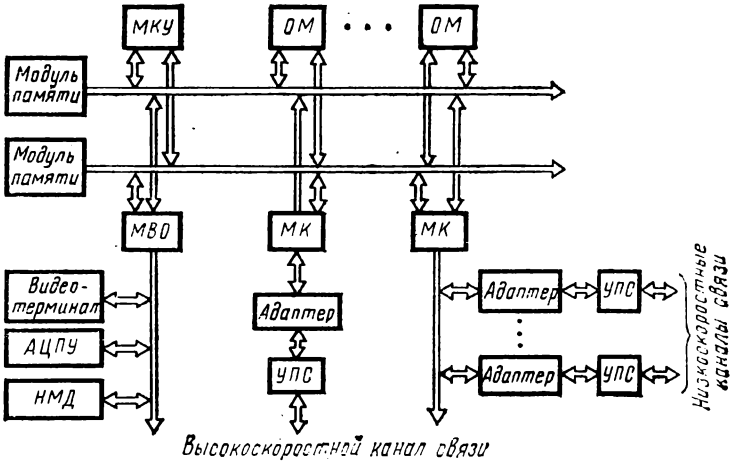



Рис. 2.24. Структура многопроцессорного ЦКП

Такая структура имеет ряд недостатков. К ним прежде всего относятся: использование центральной ЭВМ, от надежности которой зависит функционирование всей системы; ограничение быстродействия системы из-за скорости работы центральной ЭВМ; необходимость в центральном диспетчере шины, понижающем надежность работы; сложность шины, часто имеющей десятки различных цепей.

В условиях нестационарности нагрузки и высоких требований к надежности функционирования оборудова-

ния наибольшее распространение получают коммуникационные системы с модульной структурой как с прямым, так и с функциональным типом организации. Функциональная мультипроцессорная система проще с точки зрения организации памяти, так как отдельный МП-модуль располагает ограниченным набором программ, связанных с кругом возложенных на него задач. По сравнению с функциональными прямые мультипроцессорные системы имеют преимущества в возможности наращивания объема УКН и ЦК и расширения их функций, в более высокой функциональной надежности. Пример структуры, построенной на основе описанных в § 2.3 функциональных модулей, показан на рис. 2.24. Здесь МКУ — модуль контроля и управления, осуществляющий контроль за правильностью функционирования ПО ЦКП и управление реконфигурацией ЦКП при отказах ТС и сбоях ПО. ОМ, МВО и МК — соответственно обрабатывающие, взаимодействия с оператором и канальные ФМ. Таким образом, ЦКП состоит из набора функционально-ориентированных модулей (ФМ), объединенных для совместной работы общим полем памяти. Общая память ЦКП разбита на отдельные модули, доступ к которым осуществляется по независимым каналам. Организация взаимодействия процессов через общее поле памяти позволяет создать надежную и гибкую структуру ПО ЦКП.

# ГЛАВА 3

# ЭТАПЫ ПРОЕКТИРОВАНИЯ МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВ СИСТЕМ ПЕРЕДАЧИ ИНФОРМАЦИИ И ЕГО МАТЕМАТИЧЕСКИЙ АППАРАТ

## § 3.1. ХАРАКТЕРИСТИКИ ПРОГРАММИРУЕМЫХ УСТРОЙСТВ СПИ

Любое программируемое устройство СПИ характеризуется множеством внешних и внутренних параметров. Принято выделять подмножества внешних параметров устройства СПИ $Y=\{y_1, y_2, ..., y_i, ..., y_m\}$ и внутренних параметров

$$X = \{x_1, x_2, ..., x_j, ..., x_n\}.$$

Внешние параметры описывают устройство с точки зрения заказчика или потребителя. Они часто характеризуют данное устройство как подсистему в боль-

шой системе, каковыми являются АСУ или сеть связи. На внешние параметры накладываются ограничения, определяющие требуемые по техническому заданию значения параметров или области их допустимых значений. Математически эти ограничения задаются системой равенств или неравенств вида $y_i = a$; $y_k \leqslant b$; $y_l \geqslant c$; ...

В более сложных случаях ограничения на внешние параметры включают зависимости между ними и представляются в виде функциональных равенств или неравенств вида $\varphi_{yl}(Y) = 0$; $\varphi_{yj}(Y) \leqslant 0$, где левые части представляют собой функции многих переменных.

Совокупность ограничений на внешние параметры $\Phi_y$ определяет допустимое множество внешних параметров $Y_{\text{доп}}$, являющееся некоторым подмножеством $Y$.

Внутренние параметры описывают устройство с точки зрения разработчика (проектировщика). На них, как и на внешние параметры, накладываются ограничения $\varphi_{xl}(X) = 0$; $\varphi_{xj}(X) \leqslant 0$, где $\varphi(X)$ — некоторая функция внутренних параметров. Совокупность ограничений $\Phi_x$ определяет допустимое множество внутренних параметров $X_{\text{доп}}$.

Внутренние и внешние параметры устройств СПИ могут быть как непрерывными, так и дискретными величинами. Так, например, вероятность ошибки — это всегда непрерывная величина, а число каналов, подключаемых к устройству, принимает дискретное множество значений.

Множество внешних параметров устройств СПИ $Y$ включает три подмножества параметров: информационное, технико-экономическое и технико-эксплуатационное.

Важнейшими элементами подмножества *информационных* параметров являются:

количество каналов связи, обслуживаемых программируемым устройством $L$; скорости манипуляции в данных КС — $V_{\text{м}}$;

среднее и максимальное время задержки информации в устройстве для сообщений $i$-й $(i = \overline{1, m})$ категории срочности $\overline{T}_{\text{з}} = \{\overline{T}_{\text{з}i}\}$ и $T_{\text{зmax}} = \{T_{\text{зmax}i}\}$;

вероятность необнаруженных ошибок в сообщении $P_0$, которая обычно выражается через вероятность ошибки на знак сообщения;

вероятности потери сообщения по причине отказа в обслуживании $P_{\text{пот}i}$ — для каждой из $i$ категорий срочности;

производительность программируемого устройства СПИ, определяемую как количество обрабатываемых сообщений $N_\Sigma$, проходящих через устройство в единицу времени (например, за час) или же задаваемую как вектор $\bar{N}_\Sigma = \{N_1, N_2, ..., N_N\}$, компонентами которого являются количества сообщений $i$-й категории срочности, проходящих через устройство в единицу времени;

коэффициент загрузки $K_з$ обрабатывающих модулей для устройства СПИ.

*Технико-экономические* параметры устройства СПИ включают в себя стоимость, габариты (объем), массу.

Подмножество *технико-эксплуатационных* параметров содержит: среднее время безотказной работы, состав УВВ, а также ряд дополнительных требований (температурный диапазон, механические воздействия и т. д.).

Внутренние параметры $X$ программируемых устройств СПИ играют роль «независимых переменных» при проектировании этих устройств. К числу этих параметров относятся характеристики используемых элементов МП-системы и ее архитектуры. Так, например, для МП важнейшими параметрами являются: разрядность, система команд, система адресации, количество РОН, число уровней прерывания, технология изготовления, быстродействие, стоимость, число источников питания. Для ЗУ в качестве таковых выступают емкость и время выборки. Для архитектуры системы характерны тип структуры, количество МП, типы и функциональная сложность адаптеров, выбранный метод распределения функций по обрабатывающим элементам, наличие резервных блоков, вид системы диагностики и контроля и т. д.

Внешние и внутренние параметры устройства СПИ связаны определенными зависимостями. При этом в общем случае каждый из внешних параметров проектируемого устройства может быть представлен в виде функции его внутренних параметров

$$\left.\begin{aligned} y_1 &= f_1(x_1, x_2, ..., x_n); \\ y_2 &= f_2(x_1, x_2, ..., x_n); \\ &\cdots\cdots\cdots\cdots \\ y_m &= f_m(x_1, x_2, ..., x_n). \end{aligned}\right\}$$

Представленная выше система называется уравнениями связи $F(x, y)$ и задает математическую модель проектируемого устройства. Построение такой модели является одной из основных задач системного проектирования микропроцессорных устройств СПИ.

## § 3.2. ЗАДАЧИ И ЭТАПЫ ПРОЕКТИРОВАНИЯ МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВ СПИ

К настоящему времени при проектировании сложных систем установилось два основных подхода: классический и системный. Классический подход предполагает разработку системы на основе разработки отдельных ее подсистем и дальнейшее объединение этих подсистем как компонентов в единую систему. Классический подход основывается на движении от частного к общему и зачастую не раскрывает тех общих задач, которые необходимо решать в данной системе, учитывая, что она входит как подсистема в систему более высокого ранга.

Системный подход исходит прежде всего из тех целей и задач, которые должна решать данная система, как основная часть более сложной системы. В соответствии с этим системный подход возможен только при определении целей поведения системы, критериев эффективности, стратегии управления, построения модели системы и, наконец, разработки системы в целом с учетом различных подсистем, входящих в ее состав.

Системный подход является в настоящее время более общим подходом и предполагает, что разработка начинается прежде всего с определения цели, которую должна решать данная система. Очевидно, что основной целью микропроцессорных устройств СПИ является *передача заданных объемов информации различных уровней приоритетности по заданным каналам связи с требуемой верностью и быстродействием.* Такая постановка задачи допускает множество различных путей решения, дающих в результате различные варианты структуры устройств СПИ, удовлетворяющие заданным требованиям по количеству обслуживаемых КС, скорости передачи, верности, стоимости устройства, ее сложности, надежности и другим внешним характеристикам. Поэтому весьма существенной частью при системном подходе к проектированию является выбор критериев эффективности. Построение модели устройства СПИ включает в себя как разработку моделей подсистем микропроцессорного устройства, так и моделей внешней среды.

Внешней средой по отношению к устройствам СПИ являются различные входные воздействия, которые могут иметь либо детерминированный, либо случайный ха-

рактер. Для СПИ характерным оказывается *случайность всех входных воздействий,* а отсюда вытекает необходимость определения моделей входящих потоков сообщений, моделей ошибок, действующих в КС, моделей поведения различных обслуживающих подсистем, учитывающих разные способы организации очереди, метода обслуживания в системе и т. д.

При системном подходе к проектированию устройств СПИ, как и других сложных систем, выделяют два основных крупных этапа. Первый этап — это этап *системного проектирования* (определяют так же, как макропроектирование или внешнее проектирование). На этом этапе выбираются цели, критерии, модели и стратегии управления для устройства. Производится анализ исходных данных, моделей системы и синтезируется основная структура устройства СПИ. На втором этапе — *технического проектирования* (микропроектирования, внутреннего проектирования) — производится непосредственно разработка МП-устройства в соответствии с выбранной структурой и основными принципами функционирования.

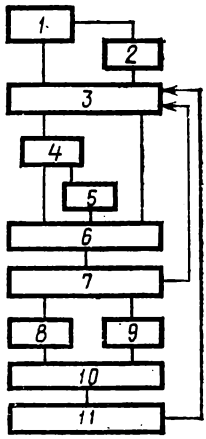



Рис. 3.1. Обобщенная последовательность этапов проектирования устройства СПИ при системном подходе

На рис. 3.1 представлена обобщенная последовательность этапов проектирования устройства СПИ при системном подходе.

Построение устройства начинается с определения перечня его задач, внешних параметров *1.* Одновременно с этим выбираются критерии качества функционирования системы *2.* На основе *1* и *2* формируются исходные данные для построения устройства *3.* Далее строятся модели внешней среды (источников сообщений, источников ошибок в КС) — *4* и модель самого устройства СПИ *5.* На основании этих моделей и исходных данных формируется общая математическая модель устройства СПИ — *6.* Исследование данной модели *7* позволяет синтезировать структуру разрабатываемого устройства.

Этап технического проектирования включает разработку технических средств устройства *8,* его программно-

го обеспечения *9*, а также изготовление макета *10*. В дальнейшем производится отладка задач системы на макете *11*.

Процесс проектирования носит итерационный характер (переходы от блоков *7* и *11* к блоку *3*). В том случае, если оценка параметров устройства на модели или макете приводит к отрицательному результату, производится корректировка исходных данных и далее выполняются рассмотренные выше шаги.

Разумеется, этап технического проектирования в большой степени зависит от используемой в программируемом устройстве элементной базы. Для процедуры проектирования вычислительных устройств на ИС малой и средней степени интеграции в макроэтап технического проектирования включается этап логического проектирования, на котором разрабатываются функциональные схемы каждого из блоков устройства. При использовании МП и БИС надобность в этом этапе либо совсем отпадает, или же разрабатываются лишь некоторые нестандартные функциональные узлы. Следует отметить большую роль, которая отводится на этапе технического проектирования таким вопросам, как разработка тестовых и диагностических средств, документации на устройство и его ПО, испытания и усовершенствование макета, выбор алгоритмов реализации важнейших функций микропроцессорного устройства СПИ.

Этап системного проектирования (блоки *1*...*7* на рис. 3.1) обычно подвергается дальнейшей детализации с выделением и использованием моделей разного уровня абстракции. Наличие большого числа шагов как при системном, так и при техническом проектировании устройств СПИ может привести к неоптимальному варианту построения, если на каждом шаге проектирования не учитывать принятых критериев эффективности. Обычно проектирование ведут таким образом, что на каждом шаге разрабатывают несколько вариантов устройства и далее принимают оптимальное решение. При этом исходят из принятых ранее критериев эффективности, т. е. на каждом шаге необходимо обеспечить требуемый уровень эффективности проектируемой системы. Любой из шагов (подэтапов) проектирования может быть структурно представлен в виде последовательности следующих стадий: выбора набора вариантов, принятия критериев определения оптимальности вариантов, проведения оценки выбранных вариантов, выбора лучшего варианта реше-

ния на данном этапе. Наиболее существенным здесь является выбор критериев оценки оптимальности решения. Этот вопрос применительно к микропроцессорным устройствам СПИ будет подробно рассмотрен в § 4.1.

## § 3.3. ВАЖНЕЙШИЕ ЭЛЕМЕНТЫ ПРОЕКТИРОВАНИЯ МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВ СПИ

Выше было показано, что проектирование программируемых микропроцессорных устройств СПИ содержит два макроэтапа — системного и технического проектирования. Детализируем последовательность и содержание этих этапов.

Первым шагом системного этапа проектирования является анализ и уточнение технического задания (ТЗ) на устройство (рис. 3.2). На этом шаге производится определение внешних параметров, которые описывают устройство с точки зрения заказчика или потребителя; уточняются ограничения, которым должны удовлетворять внешние параметры (см. § 3.1).

На первом шаге производится также выделение и уточнение множества $A=\{a_i\}$ задач, решаемых устройством, и определяются нагрузочные характеристики устройства. Наиболее важное (критичное) место занимают канальные нагрузочные характеристики, которые определяют структуру и объемы потоков информации от устройства в КС и из КС в устройство. Характеристики потоков сообщений в общем случае могут быть заданы в виде распределений интервалов между моментами поступления сообщений, условных вероятностей появления сообщений определенной категории срочности и распределений длин сообщений с детализацией по категориям срочности.

Связь между процессами в современных СОИ стандартизируется иерархической системой протоколов. Важным в этом случае является определение ряда системных параметров протокола применительно к заданному КС или классу КС.

Вторым шагом системного этапа является задание критерия эффективности, позволяющего оценивать оптимальность решения в процессе проектирования устройства. Задача выбора критерия $K_{\text{мпу}}$ занимает одно из важнейших мест при разработке (см. гл. IV).

При проектировании микропроцессорного устройства СПИ требуется определить множество его внутренних

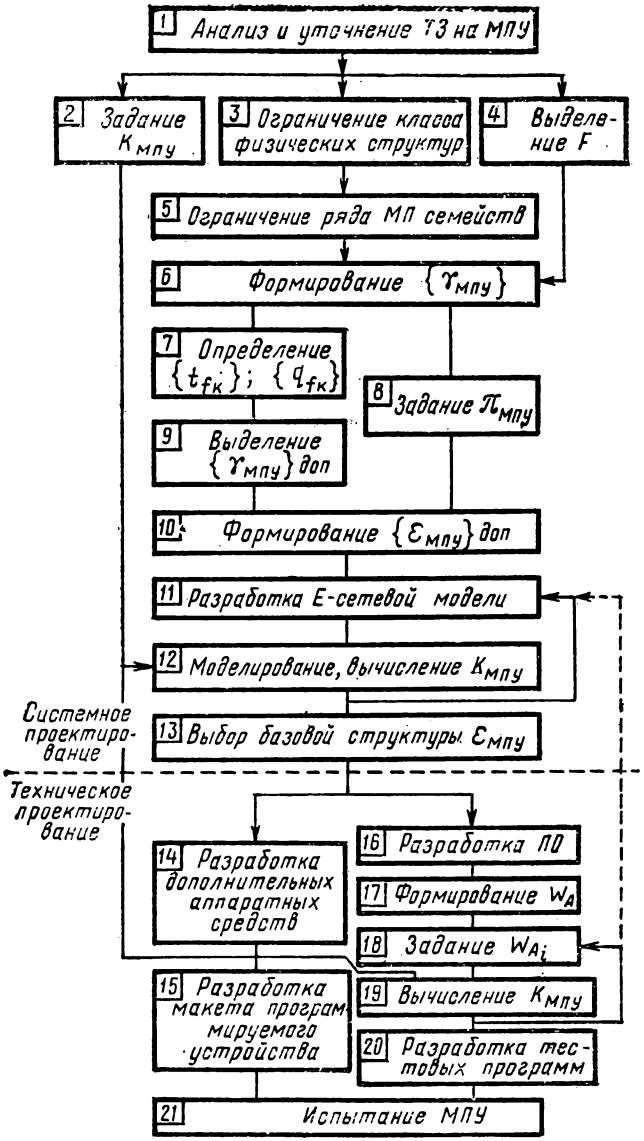



Рис. 3.2. Последовательность этапов проектирования программируемого МПУ СОИ

параметров (быстродействие МП, объем ЗУ, тип интерфейса и т. п.), описывающие устройство с точки зрения его разработчика. Они играют роль «независимых переменных» задачи проектирования и одновременно однозначно определяют значения внешних параметров. Множество допустимых проектов устройства включает в себя те варианты, для которых вектор внутренних параметров содержит компоненты, удовлетворяющие системе ограничений $\Phi_x$ на внутренние параметры, и в то же время доставляющие внешним параметрам значения, удовлетворяющие системе ограничений $\Phi_y$ на внешние параметры.

В общем случае задача проектирования оптимального микропроцессорного устройства может быть решена только путем полного перебора различных вариантов совокупностей внешних параметров устройства (удовлетворяющих $\Phi_x$) с последующим их сравнением. Таких вариантов может быть очень много. Поэтому в практике проектирования сложился подход, когда на основе опыта проектирования и ряда умозрительных соображений выбирается ограниченное число вариантов совокупностей основных технических характеристик, среди которых и ищется вариант, обеспечивающий максимальное значение критерия эффективности.

Ограниченный ряд МП семейств, использование типовых периферийных БИС, типизация структур и т. п. уменьшают количество вариантов структур МП устройств СПИ, однако и в этом случае необходимо ввести ограничения на класс рассматриваемых физических структур. Ранее (см. гл. 2) были рассмотрены наиболее типичные МП структуры с функциональным типом организации и широким использованием периферийных БИС, применяемые для построения СП устройств СПИ. На 3-м шаге системного этапа кроме ограничения такого вида могут также быть введены ограничения на число используемых МП в структуре, тип интерфейса с УВВ и т. д.

Путем анализа множества $A$ задач на 4-м шаге необходимо выделить полную логическую структуру $F$ устройства. Множество $F$ в дальнейшем используется для формирования вариантов программно-аппаратного распределения функций в устройстве с целью их реализации.

Основными элементами физической структуры МПУ являются БИС МП-семейств. В настоящее время выпускается несколько десятков таких наборов. Исходя из на-

личия, доступности, стоимости, массогабаритных характеристик, параметров надежности, допустимого температурного диапазона и т. п., на 5-м шаге системного этапа целесообразно ограничить ряд МП-семейств с целью уменьшения допустимого множества вариантов структур.

Для МПУ, характеризуемого заданной логической структурой и выбранной физической структурой, существует множество способов отображения логической структуры на физическую, задающих множество $\gamma_y = \{\gamma_{yi}\}$ программно-аппаратных разбиений функций. Формирование множества $\gamma_y$ производится на шаге 6 системного этапа. Число элементов $\{\gamma_{yi}\}$ существенно зависит от таких факторов, как: выбор программной реализации функций обмена с УВВ, использование для сопряжения с УВВ периферийных БИС или аппаратной логики; реализация сетевых функций на специализированном сетевом МП или программно-аппаратная реализация с использованием периферийных схем различной степени интеграции; выбор принципа обработки прерываний в МП (программный, аппаратный); наличие в МП-системе БИС прямого доступа к памяти; выбор принципа распределения функций по МП мультимикропроцессорной системы (равномерная средняя загрузка МП, минимизация обращений к общей памяти, концентрация функций подмножеств $F_i$ на одном МП и т. д.). Перечисленные факторы делают данный этап весьма трудоемким. Формирование $\{\gamma_{yi}\}$ часто основывается на анализе однотипных разработок, зависит от сложности алгоритмов обработки при реализации той или иной функции.

На основе анализа $\{a_i\}$ относительно их критичности к реальному масштабу времени и с использованием $\Phi_y$ определяется допустимое в устройстве подмножество принципов диспетчеризации, которое в простейшем случае может быть задано как $\pi_y : A \to N^D$, где $N^D = \overline{(1, K)}$ — число, задающее приоритетность задачи. Задание подмножества $\{\pi_y\}$ осуществляется на 8-м шаге системного этапа.

Элементы $\{\gamma_{yi}\}$ могут быть исследованы только в случае получения хотя бы ориентировочных реализационных характеристик для основных элементов логической структуры $F$, каковыми являются время $t_{f_k}$ выполнения функции $f_k$ и требуемая емкость $q_{f_k}$ ЗУ. Получе-

ние элементов $\{t_{fk}\}$ и $\{q_{fk}\}$ на 7-м шаге возможно различными путями: анализом однотипных проектов, методами эталонных задач и эталонных программ, построением модели базовой микроЭВМ.

Полученное множество $\{t_{fk}\}$ на 9-м шаге можно использовать для ограничения допустимого $\{\gamma_{yi}\}$ построением простой модели устройства и контрольной оценки производительности основных физических модулей на этой модели. Поскольку проверка носит приближенный характер, то при проведении ее цесесообразно учитывать только те $a_i$, которые выполняются в реальном масштабе времени и обеспечивают заданные скорости приема и передачи. Подходящими моделями для такого анализа являются системы массового обслуживания, синхронизированные сети Петри. Последние позволяют отразить асинхронные параллельные процессы при циклическом режиме функционирования.

Аналитические методы исследования не отражают ряд существенных особенностей программируемых МПУ. В связи с этим в качестве основного метода анализа структуры и характеристик программируемых МПУ СПИ на системном этапе широко используется имитационное машинное (программное) моделирование, в основе которого лежит метод статистических испытаний. При разработке и построении имитационных моделей устройств СПИ применяются различные математические схемы. Большие преимущества в этом смысле дает применение аппарата *E*-сетей. Во-первых, *E*-сеть служит средством задания структуры и параметров имитационной модели, дающим наглядное представление об информационных потоках в системе, во-вторых, при программной реализации *E*-сетевых модулей упрощаются процессы разработки и модификации моделирующей программы. Разработка *E*-сетевой модели и соответствующей программы моделирования осуществляется на 11-м шаге системного этапа.

Процедура моделирования программируемого МПУ СПИ (12-й шаг) позволяет получить основные характеристики множества внешних параметров устройства и оценить рассматриваемую структуру $(\gamma_y, \pi_y)$ по критерию эффективности $K_{мпу}$. Основным в этом шаге является синтез через анализ различных допустимых вариантов реализации $\{\varepsilon_y\}_{доп}$. Процедура управления моделированием, позволяющая исключить перебор всех возмож-

ных элементов $\{\varepsilon_y\}$, организуется с использованием измальное значение критерия $K_{\text{мпу}}$ (13-й шаг).

Системный этап проектирования завершается выбором некоторого элемента $\{\varepsilon_y\}$, обеспечивающего максимальное значение критерия $K_{\text{мпу}}$ (13-й шаг).

На этапе технического проектирования МПУ СПИ выполняются обычные для МП-систем процессы разработки дополнительных аппаратных средств и макета, ПО, тестовых и диагностических средств, испытания системы.

При техническом проектировании возможна дальнейшая оптимизация выбранного варианта $\varepsilon_y$ по критерию $K_{\text{мпу}}$. Наиболее широкий диапазон для этого представляет разработка ПО устройства. Ряд функций $f_k$ допускает различные способы их алгоритмической реализации, существенно отличающиеся по параметрам $t_{fk}$ и $q_{fk}$. В качестве примера это будет показано применительно к процедурам кодирования и декодирования (см. гл. 5). Оценка множества $W_A = \{w_{A_i}\}$ вариантов алгоритмической структуры проводится по единому критерию $K_{\text{мпу}}$.

Из изложенного следует, что изучение свойств МПУ СПИ и развитие методов их конструирования проводятся с помощью различных математических моделей и методов в зависимости от степени детализации структуры МПУ и их свойств, характера исследуемых проблем.

Специфика МПУ СПИ определяется наличием в их составе СП. Учитывая, что микропроцессорные СП устройств СПИ являются ЭВМ специального назначения, при исследовании характеристик СП используют опыт применения математических моделей для описания функционирования вычислительных систем (ВС).

Математические модели микропроцессорных СП могут быть как аналитическими, так и имитационными.

В последующих параграфах этой главы дается краткая характеристика некоторых особенностей применения наиболее используемых в настоящее время для исследования ВС математических моделей: моделей СМО, сетей Петри и имитационных.

Модели СМО могут быть использованы для приближенной оценки ряда показателей качества МПУ СПИ, по которым на 9-м шаге осуществляется ограничение $\{\gamma_{yi}\}$. Так, например, для систем с очередями, к которым относятся МПУ СПИ, необходимо прежде всего оценить загруженность системы, простейшей мерой которой яв-

ляется отношение средней длительности обслуживания заявки (обработки сообщения в СП) к среднему времени между моментами поступления заявок (сообщений на вход СП). Зная интенсивности потоков сообщений на входах устройств СПИ, можно предъявить определенные требования к средней скорости их обработки микропроцессорной системой. Другим показателем качества МПУ СПИ является их пропускная способность — среднее число заявок (сообщений), обслуженных за единицу времени. Это очень важный показатель, поскольку пропускная способность того или иного пути в СОИ определяется пропускной способностью элемента (узла, канала) пути с минимальной интенсивностью обслуживания. Среднее время пребывания заявки в СМО, которое обычно складывается из времени ожидания обслуживания и времени обслуживания, характеризует среднее время задержки сообщения в устройстве СПИ. Важной мерой загруженности является «длина очереди», изучение распределения которой требуется, например, при оценке объема буферной памяти, необходимой в устройствах СПИ для размещения поступающих из каналов связи и передаваемых в каналы связи сообщений.

Важным подмножеством сетевых функций является реализация протокольных процедур в СП. Поскольку описание протоколов в большинстве случаев имеет неформальный характер, то это приводит к неоднозначной трактовке процедур разными проектировщиками и усложняет процесс разработки СП. Применение формального аппарата сетей Петри на 4-м шаге позволяет существенно упростить и ускорить процесс создания СП, описать и исследовать алгоритмы функционирования протокольных станций и различные аспекты взаимодействия, является основой автоматизации процедур анализа протокольных структур.

Применение $E$-сетей на 11-м шаге в качестве математической схемы для имитационного моделирования (12-й шаг) наряду с отмеченными достоинствами обусловлено простотой и наглядностью $E$-сетевого описания; большой моделирующей мощностью, позволяющей отобразить динамику взаимодействия и асинхронный характер процедур; возможностью перенесения методов моделирования (например, на языке GPSS) МПУ на область протоколов, что определяет единообразие описания моделей протоколов и вычислительных устройств, их реализующих;

принадлежностью $E$-сетей и сетей Петри к одному классу моделей.

Имитационное моделирование используется как для проверки аналитических оценок, так и самостоятельно для получения оценок в тех случаях, когда аналитические методы неприменимы или не обеспечивают необходимую точность. При изложении метода имитационного моделирования, достаточно хорошо известного и широко применяемого на практике, основное внимание обращено на особенности моделирования редких событий, характерных для процесса функционирования устройств СПИ (например, вероятность блокировки и искажения информации для реальных СП порядка $10^{-5}...10^{-6}$).

## § 3.4. МОДЕЛИ СП НА БАЗЕ СМО

Исследование вероятностно-временных характеристик процесса функционирования СП аналитическим методом можно провести с использованием моделей систем массового обслуживания (СМО), если они являются марковскими или сводятся к марковским, например, методом вложенных цепей Маркова. В последнее время при создании моделей ВС все чаще стали применяться сетевые модели СМО, допускающие при определенных условиях декомпозицию сети на отдельные СМО. Методы теории СМО являются подходящими для анализа ТС и ПО СП по следующим причинам: 1) функционирование СП наиболее естественно описывается на языке очередей (очереди заданий, процессов к ресурсам, процессов к обрабатывающим модулям, очереди к секциям ОПП). Напомним, что задание представляет собой совокупность действий для выполнения вычислительной системой некоторой работы, предъявляемой пользователем; 2) показатели эффективности выражаются через основные показатели СМО — через время нахождения заявки в системе, через среднюю производительность устройства обработки, среднюю длину очередей, среднее время ожидания заявок в очереди. Основные элементы СМО показаны на рис. 3.3. *Заявки* поступают на обслуживающее устройство. Если в момент поступления заявки *обслуживающее устройство* занято, то заявка либо занимает очередь к нему, где ожидает начала обслуживания (СМО с ожиданием), либо теряется (СМО с потерями). Выбор заявки на обслуживание в какой-то момент времени производится в соответствии с некоторым пра-

вилом, которое называется *дисциплиной обслуживания.* Далее выполняется обслуживание заявки, после чего заявка покидает систему. Выходящий поток обслуженных заявок может оказаться весьма важным в тех случаях, когда он является входящим для другой СМО.

Модели структур ВС в виде СМО в простейшем случае рассматривают следующим образом: имеются два устройства, работающие, асинхронно, последовательно обслуживая заявки, и буферное запоминающее устройство конечной емкости, согласующее работу этих устройств. Первое из устройств работает в режиме генератора заявок, второе обслуживает заявки, поступающие с

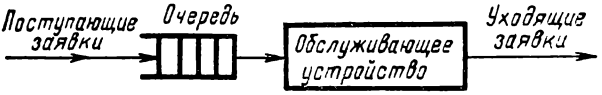



Рис. 3.3. Структура системы массового обслуживания

выхода первого устройства через накопитель (буферное ЗУ). Возможны две дисциплины работы системы, обусловленные появлением заявки на выходе первого устройства, когда накопитель полностью заполнен заявками: 1. Первое устройство (узел) прекращает генерировать заявки и хранит уже обслуженную им (сгенерированную) заявку до тех пор, пока не освободится место в накопителе, т. е. пока второй узел не примет из ЗУ к обслуживанию очередную заявку. Это *система с блокировкой,* она рассматривается как двухфазная СМО с накопителем конечной емкости и потоком заявок с интенсивностью их поступления, равной бесконечности. 2. Заявка, обслуженная первым узлом, теряется, а сам первый узел сразу же начинает обслуживать (генерировать) следующую заявку. Такая дисциплина приводит к СМО с одним обслуживающим прибором, одним потоком заявок и конечным накопителем, известной в теории СМО как *система с потерями* (рис. 3.3).

Одной из простейших моделей мультипрограммной ВС с постоянным уровнем мультипрограммирования является *замкнутая двухфазная* СМО (рис. 3.4). Первое обслуживающее устройство интерпретируется как центральный процессор (ЦП), второе — как устройство ввода — вывода (УВВ), объединяющее канал и внешнюю память (ВП), а заявками являются $m$ программ, одновременно находящихся в оперативной памяти. Пред-

полагается, что в процессе выполнения каждой программы возникают запросы к ВП, что соответствует перемещению заявки в рассматриваемой СМО из очереди 1 в очередь 2. После удовлетворения запроса программа готова к выполнению ЦП, чем соответствует перемещение заявки из очереди 2 в очередь 1. Предполагается, что длительности обслуживания заявок независимы между собой на каждой фазе в отдельности и на различных фазах и что обслуживание заявок происходит в порядке их поступления.

Если распределение длительности обслуживания на каждой фазе является экспоненциальным, то замкнутую двухфазную СМО можно свести к СМО типа $M/1/1/m$ и, имея характеристики среднего времени обслуживания на каждой фазе, можно для стационарного режима получить формулу вероятности распределения $m$ заявок по очередям 1 и 2. Это позволяет определить «узкое» место в ВС и соответственно расширить необходимый ресурс на фазе 1 либо 2. Расширение ресурсов за счет включения нескольких параллельно работающих устройств на каждой фазе приводит к различным модификациям рассмотренной двухфазной модели. Эти модификации позволяют оценить показатели, связанные с использованием одного или двух вычислительных ресурсов, выделяемых из общего множества ресурсов ВС. Для анализа более сложных моделей ВС, охватывающих как аппаратные, так и программные ресурсы, используются модели сетей массового обслуживания.

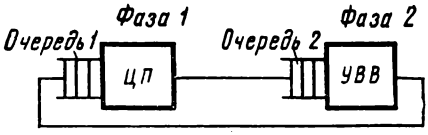


Рис. 3.4. Двухфазная модель вычислительной системы

Сеть массового обслуживания представляет собой совокупность СМО, в которой циркулируют заявки, переходящие из одной системы в другую. Различают замкнутые и открытые сети массового обслуживания. В *замкнутой* сети массового обслуживания заявки не поступают извне и не уходят из сети, т. е. в ней циркулирует постоянное число заявок. В *открытую* сеть массового обслуживания заявки поступают из бесконечного внешнего источника и могут покинуть сеть после завершения обслуживания.

Методы расчета средних показателей сети связаны с

громоздкими вычислениями, исключающими важное преимущество аналитических решений — скорость получения и наглядность результата. Приемлемым оказывается метод расчета средних показателей для бесприоритетных замкнутых сетей СМО с экспоненциальными обслуживающими приборами. Для замкнутой сети (рис. 3.5) предполагают, что она состоит из экспоненциальных устройств с постоянными интенсивностями обслуживания. Для $i$-й СМО средняя длительность обслуживания, среднее время пребывания (ответа) и средняя длина очереди зависят от общего числа заявок в сети $N$. От этого же зависит и пропускная способность рассматриваемой сети.

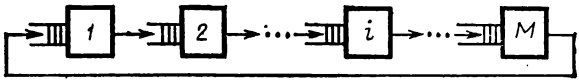



Рис. 3.5. Замкнутая сеть из последовательно соединенных обслуживающих устройств

При экспоненциальной длительности обслуживания среднее время ответа $i$-й СМО складывается из средней длительности обслуживания вновь поступившей заявки и средней длительности обслуживания всех заявок, находящихся в $i$-й СМО в момент поступления в нее новой заявки. При таких условиях удается получить замкнутую систему уравнений, связывающую между собой среднее время ответа, среднее время обслуживания, среднюю длину очереди для $i$-й СМО и пропускную способность сети для заданного значения $N$. Решая эти уравнения рекуррентно для $N=1, 2, \ldots$, находят интересующие средние показатели сети, рассматривая последнюю, как состоящую из совокупности независимых СМО. Замкнутыми экспоненциальными сетями СМО описывают модели мультипроцессорных систем с общей памятью и мультимашинных систем с блоками локальной памяти, структуры которых используются для построения МПУ СПИ. Замкнутость сети не противоречит тому, что сообщения (заявки) поступают в систему и покидают ее, так как в условиях высокой загруженности, представляющих наибольший практический интерес, в ВС циркулирует конечное число заданий.

Пример использования такой модели для расчета среднего времени задержки пакетов при обработке их в мультипроцессорном ЦК показан в [32].

Математическая модель построена для СП, представленного в виде мультипроцессорной системы (рис. 3.6), содержащей канальные модули $КМ_1$—$КМ_S$, обрабатывающие модули $ОМ_1$—$ОМ_N$, секции ОПП $СОП_1$—$СОП_M$, модуль комплексирования — полнодоступный матричный коммутатор (МК). Рассматривается вариант фиксированного размещения программных модулей в локальной памяти ОМ.

Каждый из КМ обслуживает один или несколько дискретных каналов. На приеме КМ производит накопле-

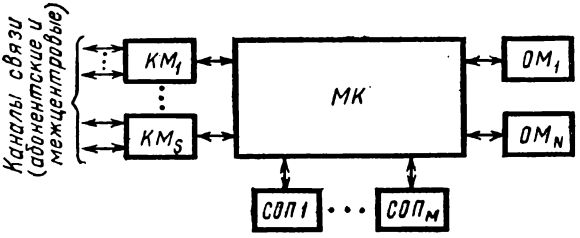



Рис. 3.6. Обобщенная структура технических средств СП ЦКП

ние блоков информации по каждому из каналов и их пересылку в режиме прямого доступа в отведенные буферы ОПП. На передаче КМ осуществляет поблочную пересылку информации в режиме прямого доступа из буферов ОПП в локальную память КМ с последующей побитовой передачей информации в исходящие КС. ОМ реализует логическую обработку информации, записанной в ОПП, согласно функциональным алгоритмам МПУ СПИ. ОПП состоит из секций оперативной памяти (СОП) с независимыми трактами доступа со стороны КМ и ОМ через МК.

Процесс обработки информационных кадров (ИК) в СП представлен в виде системы *технологических цепочек* (ТЦ). Каждый ИК, записанный КМ в ОПП, генерирует задание, которое обрабатывается на ОМ прикладными процессами, реализующими заданные функциональные алгоритмы, по некоторой ТЦ. Каждая ТЦ определяет последовательность обработки ИК и пакетов и задается управляющей информацией, содержащейся в их заголовках. Например, в $l$-ю ТЦ (рис. 3.7) входят процессы приема ИК, реализации процедуры HDLC на приеме, протокола *X*.25/3 на передаче, адаптивной мар-

шрутизации, реализации процедуры HDLC на передаче, передачи ИК. Считается, что в ТЦ отсутствуют петли.

Одновременно, если позволяет число ОМ, может обрабатываться несколько заданий как различными процессами, так и копиями одного процесса. В систему процесса $i$-го типа (рис. 3.7) входят одна или несколько копий процессов, способных параллельно обрабатывать

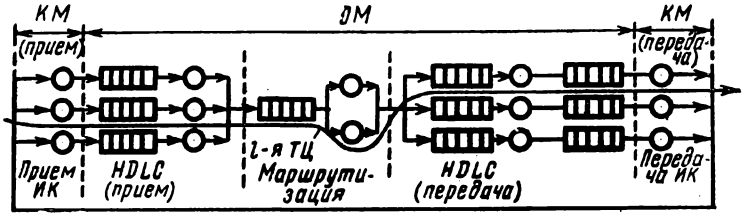



Рис. 3.7. Система ТЦ обработки информационных кадров в СП ЦКП

различные задания по одному и тому же алгоритму, и очередь заданий.

Все множество функциональных процессов разделено на группы в соответствии с распределением программ процессов по ОМ. Считается, что все копии процессов одного типа приписаны к одной группе ОМ, т. е. в различных группах ОМ не могут существовать процессы одного и того же типа. Каждый тип процессов (прием — передача, процессы операционной системы, функциональные процессы) взаимодействуют с определенными типами информационных ресурсов: буферами для хранения ИК, системными и прикладными.

Буферы ОПП для хранения ИК закреплены за каждым КМ и выделены в отдельный тип ресурсов. К системным ресурсам относятся различные таблицы операционной системы (ОС), заголовки очередей и т. п. Примером прикладных ресурсов являются таблицы маршрутизации и логических соединений.

Поскольку в ЦКП из-за ограниченного числа буферов циркулирует конечное число заданий, обрабатываемых прикладными процессами, для расчета среднего времени задержки пакетов в системе ТЦ используется модель замкнутой сети массового обслуживания (ЗСО) с экспоненциальными обслуживающими приборами. Система ТЦ представлена в виде ЗСО объединением всех выходов системы со входами (рис. 3.7). Каждый узел

рассматриваемой ЗСО является системой обслуживания соответствующего прикладного процесса. Расчет среднего времени выполнения прикладных процессов на каждом узле проведен с использованием замкнутой двухфазной СМО. Описанный алгоритм позволяет на основе единой рекурсивной процедуры определения средних характеристик замкнутой СМО рассчитать среднее время задержки пакетов в ЦКП с учетом потерь производительности, связанных с доступом ОМ и КМ в ОПП, а также с учетом временных задержек, связанных с реализацией макрокоманд ОС, с доступом к общим прикладным и системным ресурсам, с ожиданием предоставления ОМ процессам, готовым к выполнению. Результаты расчетов позволяют оценить влияние числа ОМ, СОП и входящей нагрузки на среднее время задержки пакетов в микропроцессорном ЦКП.

Применение моделей СМО для поиска аналитических решений при анализе таких сложных систем, какими являются микропроцессорные устройства СПИ, связано с большой степенью абстрагирования и упрощения реальности. Они используются обычно для первоначальной грубой оценки характеристик системы или отдельных ее подсистем. Схемы систем и сетей массового обслуживания находят широкое применение для формализованного описания ВС. Моделируемая система представляется как совокупность взаимосвязанных СМО, имитирующих функционирование аппаратурных и программных средств системы: процессоров, оперативной и внешней памяти, каналов и периферийного оборудования, управляющих и прикладных программ, а также средств, отображающих среду, в которой функционирует исследуемая система. Модели такого типа называются сетевыми имитационными моделями и строятся из набора типовых СМО. Сетевые имитационные модели с успехом используются при решении различных задач, связанных с проектированием микропроцессорных устройств СПИ.

## § 3.5. ПРИМЕНЕНИЕ АППАРАТА СЕТЕЙ ПЕТРИ

Проведенный выше анализ моделей устройств СПИ на базе СМО показал, что данные модели в очень малой степени пригодны для описания микропроцессорных устройств СПИ. Это объясняется рядом особенностей микропроцессорных устройств.

1. Прежде всего необходимо отметить, что микропроцессорные устройства СПИ работают в реальном масштабе времени (РМВ). Функционирование в РМВ объясняется ограничениями, накладываемыми на время выполнения ряда функций подмножества $F_N$, связанных с реализацией процедур протоколов (ввод информации в КС, вывод информации из КС, битовый анализ и т. д.). В масштабе времени, близком к реальному, в устройствах СПИ выполняются процедуры обработки кадров и пакетов информации, а также ряд функций подмножеств $F_T$ и $F_0$.

2. Программируемые устройства СПИ имеют несколько источников входных задач. Ввод кадра информации из каждого КС может рассматриваться в устройстве как начало задачи по обработке данного кадра (пакета) информации. Аналогичным образом можно выделить задачи вывода информации в КС, приема информации от устройств ввода — вывода, обработки команд оператора устройства. Таким образом, микропроцессорное устройство должно одновременно работать с рядом задач, эффективно распределяя при этом свои ресурсы, ограниченность которых (вычислительная мощность, емкость ОЗУ, пропускная способность КС, УВВ и магистралей) является причиной конкуренции задач за данные ресурсы.

При рассмотрении задач устройств СПИ удобно использовать понятие п р о ц е с с а как некоторую линейную последовательность функций по обработке информации, выполняемую на модулях устройства аппаратными и программными средствами. Каждый такой процесс, отражающий одну из задач (подзадач) устройства, можно представить как псевдопроцессор, обладающий в каждый момент времени определенным состоянием (пассивным, активным, ожидания).

Таким образом, даже в простейшем случае устройства СПИ являются устройствами с несколькими параллельными процессами (ввод кадров из КС, вывод кадров в КС, ввод с УВВ, вывод на УВВ, обмен с оператором, обработка пакетов и т. д.).

В однопроцессорных связных устройствах ряд программ выполняется псевдопараллельно, т. е. в инициированном состоянии находятся в общем случае несколько процессов, связанных с МП, хотя в каждый момент времени выполняется только одна программа.

Каждый процесс в устройствах СПИ строго последователен по порядку выполнения команд (функций), оп-

ределяемому программой и данными. Вместе с тем относительный порядок выполнения команд (функций), принадлежащих различным процессам, не детерминирован. Поэтому можно говорить о временной независимости одного процесса от другого и рассматривать множество этих процессов как полностью асинхронное.

Сказанное позволяет рассматривать микропроцессорные устройства СПИ как системы, функционирующие в РМВ с множеством асинхронных параллельных процессов и конкуренцией между процессами за ограниченные внутренние ресурсы.

Для описания и анализа систем с асинхронными параллельными процессами предложено к настоящему времени более 25 моделей, большинство из которых разработано в рамках теории параллельного программирования. Наиболее общими из данных моделей являются схемы параллельных программ Карпа — Миллера и *А*-программы Котова — Нариньяни. Такие же модели, как параллельные операторные схемы, счетчиковые схемы, бесповторные схемы, а также различные варианты потоковых схем (сети Дейвиса, схемы Родригеса, схемы Адамса, потоковые схемы Карпа — Миллера и др.), являются подклассами схем Карпа — Миллера.

Параллелизм проявляется в программах и системах на разных уровнях. Так, в программах это может быть микропараллелизм выражений, параллелизм независимых операторов, макропараллелизм сложных взаимодействующих процессов и т. п. Перечисленные выше модели относятся к уровню описания параллелизма независимых операторов. В то же время имеется и ряд абстрактных моделей параллельных систем и процессов, которые не содержат слишком детальной информации, требуемой для моделирования программ, а ориентированы на анализе структур управления параллельными процессами. Большинство таких абстрактных моделей строится на базе графиков специального вида с некоторой дополнительной разметкой.

Наибольшее распространение среди абстрактных моделей в начале 70-х годов получили билогические графы (UCLA-графы), представляющие собой ориентированный граф, вершины которого соответствуют операторам параллельной программы, а дуги — управляющим или информационным связям между операторами. С каждой вершиной связаны входное и выходное управления, которые могут быть двух типов — конъюнктивным или

дизъюнктивным. На рис. 3.8 показан фрагмент билогического графа, где операция * обозначает конъюнктивное, а + — дизъюнктивное управление. Вершинам и дугам графа в такой модели приписываются веса — времена выполнения операторов и передачи данных. Кроме того, каждой дуге, исходящей из оператора с дизъюнктивным выходным управлением могут быть приписаны вероятности перехода по этой дуге.

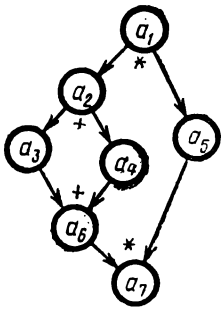



Рис. 3.8. Пример билогического графа

### Сети Петри и их особенности

К концу 70-х годов билогические графы и аналогичные им модели были практически вытеснены *сетями Петри* — самым распространенным в настоящее время формализмом, описывающим структуру и взаимодействие параллельных систем и процессов. Теория сетей Петри развивается в нескольких направлениях, среди которых — математическая теория сетей, структурная теория сетей, приложения к задачам параллельного программирования, применение сетей Петри как основы для создания моделей дискретных динамических систем (в том числе и вычислительных систем).

Среди достоинств аппарата сетей Петри можно указать следующие:

1. Сети Петри позволяют моделировать асинхронность и недетерминизм параллельных независимых событий, параллелизм конвейерного типа, конфликтные взаимодействия между процессами.

2. Как математическая модель сети Петри занимают промежуточное положение между конечными автоматами и машинами Тьюринга. При этом по выразительной мощности они значительно богаче автоматов и приближаются к машинам Тьюринга.

3. Сети Петри включают возможности ряда других моделей, предложенных для параллельных систем (семафоры Дийкстры, системы векторного сложения и векторного замещения, вычислительные схемы, модели повторно используемых ресурсов и др.), позволяя описывать как типовые ситуации в данных системах (распределение

ресурсов, взаимные блокировки), так и общую динамику работы сложной асинхронной системы.

4. Стремление расширить применимость аппарата сетей Петри привело в последние годы к появлению ряда классов сетей, ориентированных на моделирование сложных систем с учетом таких факторов, как приоритетность процессов (сети с проверкой на нуль, приоритетные сети), временные параметры событий (сети Мерлина, временные сети), совместного отображения структуры управления и потоков данных ($E$-сети).

5. В отличие от моделей параллельных программ (таких, как $A$-программы, схемы Карпа — Миллера и др.) сети Петри допускают произвольную интерпретацию элементов модели как в смысле типа выполняемого фрагмента (выражения, операторы, подпрограммы, аппаратные функциональные преобразования информации), так и по уровню абстракции. Таким образом, сети Петри позволяют производить иерархическую детализацию программных и аппаратных подсистем модели.

Перечисленные факторы дают возможность использовать данный аппарат как основу для различных этапов процедуры проектирования микропроцессорных устройств СПИ (см. § 3.3). При этом с учетом широкого диапазона имеющихся в настоящее время классов аппарата сетей Петри (языки сетей Петри, временные сети, сети Мерлина, $E$-сети и т. д.) с различными свойствами возможно использование указанного аппарата на разных уровнях проектирования — начиная от формализованного представления протоколов обмена информацией до разработки общей системной модели устройства СПИ.

Рассмотрим основные элементы аппарата сетей Петри. Формально сеть Петри задается как

$$N = (B, D, \Phi, H), \tag{3.1}$$

где $B$ — конечное множество символов, называемых *позициями*, $B \neq \varnothing$; $D$ — конечное множество символов, называемых *переходами*, $D \neq \varnothing$; $B \cap D = \varnothing$; $\Phi : B \times D \to \{0, 1\}$ — *прямая функция инцидентности*; $H : D \times B \to \{0, 1\}$ — *обратная функция инцидентности.*

Указанные четыре множества определяют структуру сети Петри. Такое задание удобно для формального рассмотрения, но не обладает наглядностью, поэтому сеть Петри обычно представляют двудольным графом со множеством вершин $B \cap D$ (рис. 3.9). Вершины-позиции изображаются кружками, а вершины-переходы — черточка-

ми. Из вершины-позиции $b_i$ в вершину-переход $d_j$ ведет дуга, если и только если $\Phi(b_i, d_j)=1$; из вершины-перехода $d_j$ в вершину-позицию $b_i$ ведет дуга, если и только если $H(d_j, b_i)=1$. Для каждого перехода $d_j \in D$ можно определить множества входных позиций перехода $\Phi(d_j)$ и выходных позиций перехода $\Phi(d_j)$ как:

$$\Phi(d_j) = \{b_i \in B \mid \Phi(b_i d_j) = 1\}; \qquad (3.2)$$
$$H(d_j) = \{b_i \in B \mid H(d_j b_i) = 1\},$$

где $i=\overline{1, n}$; $j=\overline{1, m}$; $n=|B|$; $m=|D|$.

Аналогичным образом вводятся определения множества входных переходов позиции $b_i$—$H(b_i)$ и множества выходных переходов данной позиции $\Phi(b_i)$:

$$H(b_i) = \{d_j \in D \mid H(d_j, b_i) = 1\};$$
$$\Phi(b_i) = \{d_j \in D \mid \Phi(b_i, d_j) = 1\}. \qquad (3.3)$$

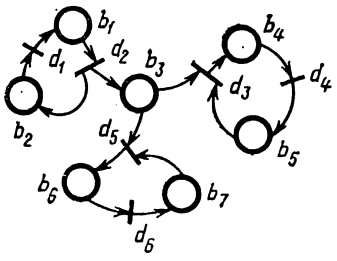



Рис. 3.9. Графовое представление сети Петри

Так, например, сеть Петри на рис. 3.9 может быть формально представлена как

$$N = (B, D, \Phi(d_j), H(d_j));$$
$$B = \{b_1, b_2, b_3, b_4, b_5, b_6, b_7\};$$
$$D = \{d_1, d_2, d_3, d_4, d_5, d_6\};$$
$$\Phi(d_1) = \{b_2\}; \qquad H(d_1) = \{b_1\};$$
$$\Phi(d_2) = \{b_1\}; \qquad H(d_2) = \{b_2, b_3\};$$
$$\Phi(d_3) = \{b_3, b_5\}; \qquad H(d_3) = \{b_4\};$$
$$\Phi(d_4) = \{b_4\}; \qquad H(d_4) = \{b_5\};$$
$$\Phi(d_5) = \{b_3, b_7\}; \qquad H(d_5) = \{b_6\};$$
$$\Phi(d_6) = \{b_6\}; \qquad H(d_6) = \{b_7\}.$$

Наибольшее применение сети Петри нашли для моделирования систем с дискретными событиями, которые могут происходить одновременно и параллельно. При моделировании отражаются два аспекта систем: *события* и *условия*. Позиции сети соответствуют условиям, а переходы — событиям. Функции инцидентности отражают связи между условиями и событиями в системе.

Приведенное выше определение сети Петри может использоваться только для отражения статики моделируемой системы (взаимосвязи событий и условий), но не позволяет моделировать динамику функционирования. Для представления динамических свойств вводится *функция разметки* (маркирования) сети

$$M: B \to \{0, 1, 2, \ldots\}.$$

С помощью функции $M$ позиции сети помечаются целыми неотрицательными числами. При графическом задании сети Петри разметка отображается помещением внутри вершин-позиций соответствующего числа точек, называемых *метками*.

Сеть Петри функционирует, переходя от разметки к разметке. *Начальная разметка* сети обозначается как $M_0$: $:B \to \{0, 1, 2, \ldots\}$. Смена разметок происходит в результате срабатывания одного из переходов $d_j \in D$ сети. Необходимым условием срабатывания перехода $d_j$ является $\forall\ b_i \in \Phi(d_j)\{M(b_i) \geqslant 1\}$, где $M(b_i)$ — разметка позиции $b_i$.

Переход $d_j$, для которого выполняется указанное условие, определяется как находящийся в состоянии готовности к срабатыванию или как возбужденный переход.

Срабатывание перехода $d_j$ изменяет разметку сети $M(b) = (M(b_1), M(b_2), \ldots, M(b_i), \ldots, M(b_n))$ на разметку $M'(b)$ по следующему правилу:

$$M'(b) = M(b) - \Phi(d_j) + H(d_j),$$

т. е. переход $d_j$ изымает по одной метке из каждой своей входной позиции и добавляет по одной метке в каждую из выходных позиций. Для изображения смены разметки $M$ на $M'$ применяют обозначение $M \xrightarrow{d_j} M'$.

Сеть Петри, в которой используется функция разметки, называют *размеченной сетью Петри* и задают набором

$$N_M = (B, D, \Phi, H, M_0), \qquad (3.4)$$

где $M_0$ — начальная разметка сети.

На рис. 3.10, *а* приведена размеченная сеть Петри с $M_0 = \{1, 0, 0, 0, 1, 0, 1\}$. При такой начальной разметке единственным готовым к срабатыванию переходом является переход $d_2$. Срабатывание его ведет к смене разметки $M_0 \xrightarrow{d_2} M_1$, где $M_1 = \{0, 1, 1, 0, 1, 0, 1\}$ (рис. 3.10, *б*). При разметке $M_1$ возможно срабатывание трех переходов — $d_1$, $d_3$ и $d_5$. В зависимости от того, какой из переходов сработал первым, получается одна из трех возмож-

ных новых маркировок (рис. 3.10, *в*, *г*, *д*). Функционирование сети Петри продолжается до тех пор, пока существует хотя бы один возможный переход.

Важной особенностью модели сети Петри является простота иерархического построения модели. Каждая сеть может рассматриваться как макропереход или макропозиция модели более высокого уровня. С другой сто-

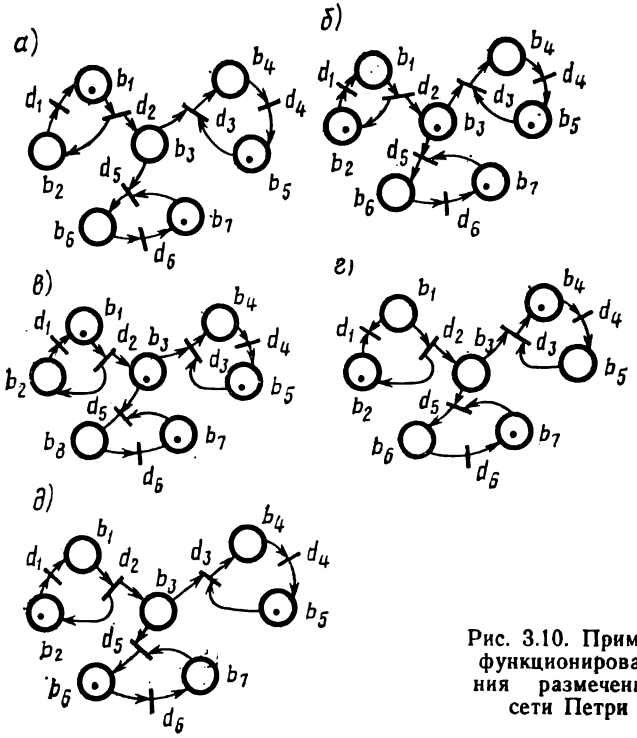



Рис. 3.10. Пример функционирования размеченной сети Петри

роны, переход или позиция могут детализироваться в форме отдельной подсети для более углубленного исследования аспектов моделируемой системы.

Обычные размеченные сети Петри, как они были определены выше, пригодны для описания событий произвольной длительности. Модель в этом случае отражает только порядок наступления событий в рассматриваемой системе. Функционирование сети Петри происходит недетерминированно в том случае, когда возникает разметка, соответствующая возбужденному состоянию двух или более переходов. Время срабатывания перехода в модели

на основе сети Петри принимается бесконечно малым и вводится предположение о том, что вероятность двух или более одновременных событий равна нулю. Тот факт, что обычные сети Петри не отражают временных параметров моделируемой системы, является существенным ограничением при их использовании в качестве моделей вычислительных систем. Ниже будет показано, что это ограничение можно обойти использованием ряда расширений аппарата сетей Петри.

Модель на основе сети Петри дает возможность анализа системы на наличие желательных или нежелательных свойств. Основные направления анализа для обычных сетей Петри следующие.

1. **Проблема достижимости.** В заданной сети $N_M$ с начальной разметкой $M_0$ требуется решить вопрос, достижима ли принципиально некоторая разметка $M'$ из $M_0$. В приложении к моделированию систем эта проблема интерпретируется как возможность достижения определенного состояния системы. При анализе используется понятие множества всех достижимых в $N_M$ разметок, которое обозначают $R(N_M)$.

2. **Свойство живости.** Под живостью перехода сети Петри понимают принципиальную возможность его срабатывания в некоторой $N_M$ при начальной разметке $M_0$. Сеть Петри определяется как живая, если все ее переходы живые. Анализ модели на свойство живости позволяет выявить неживые переходы, т. е. невозможные события в моделируемой системе.

3. **Ограниченность сети.** Позиция $b_i$ в $N_M$ называется $k$-ограниченной, если существует такое целое число $k$, что $M(b_i) \leqslant k$ для любой разметки $M$ в $R(N_M)$. Если в $N_M$ $\forall b_i \in B\{M(b_i) \leqslant k\}$ для $\forall M \in R(N_M)$, то есть $N_M$ называется $k$-ограниченной.

4. **Безопасность сети.** Сеть $N_M$ называется безопасной, если выполняется условие $\forall M(b) \in R(N_M)\{\forall b_i | M(b_i) \leqslant \leqslant 1\}$; $i=\overline{1, n}$; $n=|B|$. То есть безопасной является такая сеть Петри, у которой ни при каких условиях не может появиться более одной метки в каждой из позиций ($k$-ограниченность равна 1).

Для моделей реальных систем анализ на ограниченность (безопасность) позволяет проверить возможность функционирования системы в некотором стационарном режиме без перехода заданного предела по числу объектов в отдельных подсистемах. При анализе на это свой-

ство, в частности, могут быть выявлены требования к внутренним буферным накопителям в системе.

5. **Свойство сохранения.** Сеть $N_M$ называется сохраняющей, если $\sum_{b_i \in B} M(b_i) = \text{const}$ для $\forall M(b) \in R(N_M)$. Этим свойством $N_M$ может обладать только в том случае, когда $\forall d_j \{\Phi(d_j) = H(d_j)\}$. Другое определение свойства сохранения использует веса позиций $\omega_{b_i}$. В этом случае сеть $N_M$ является сохраняющей, если $\sum_{b_i \in B} M(b_i)\omega_{b_i} = \text{const}$ для $\forall M(b) \in R(N_M)$. Анализ модели на сохранение тогда важен, когда под динамическими объектами (метками) понимаются какие-либо неизменные ресурсы системы.

Сеть Петри может рассматриваться как форма задания некоторого формального языка. Для этого вводится *функция индексирования переходов* сети Петри $\varphi$: $D \to \Sigma$, где $\Sigma$ — алфавит формального языка. Функция $\varphi$ присваивает каждому переходу $d_j \in D$ некоторый символ из $\Sigma$. Последовательность срабатывания переходов $\tau$ определяется как слово $\Omega$ из множества $\Sigma^*$ всех слов в алфавите $\Sigma$. Множество всех возможных слов, связанных с функционированием размеченной индексированной сети Петри, определяет *язык сети Петри*

$$L(N_M) = \{\Omega \in \Sigma^* \mid M, M_0 \overset{\tau}{\vdash} M\}. \qquad (3.5)$$

Языки сетей Петри могут служить средством формального представления протоколов обмена информацией, что, в свою очередь, используется как для моделирования данных протоколов, так и для их реализации в микропроцессорных устройствах СПИ.

### Подклассы и расширения сетей Петри

Опыт использования обычных сетей Петри показал высокую трудоемкость анализа сетей большой размерности на наличие свойств достижимости, живости, ограниченности и т. д. Это явилось причиной разработки подклассов сетей Петри, в которых вводятся определенные ограничения на структуру сети, что позволяет использовать более простые алгоритмы для ее анализа. На рис. 3.11 показаны несколько таких подклассов, предложенных в последнее время, из которых аппараты автоматных графов и маркированных графов по своим моделирующим свойствам эквивалентны аппарату направленных графов.

Эти подклассы позволяют моделировать алгоритмы функционирования вычислительных систем, взаимосвязи между подсистемами и ряд других факторов без учета конкуренции между процессами.

К подклассу *автоматных графов* относят сети Петри с одной входной и одной выходной дугой у каждого перехода:

$$\forall d_j \in D \{\Phi(d_j) = H(d_j) = 1\}; \quad j = \overline{1,m}; \quad m = |D|. \qquad (3.6)$$

Ограничение на число входных и выходных переходов позиций сети Петри

$$\forall b_i \in B \{\Phi(b_i) = H(b_i) = 1\}; \quad i = \overline{1,n}; \quad n = |B| \qquad (3.7)$$

служит признаком выделения подкласса *маркированных графов.*

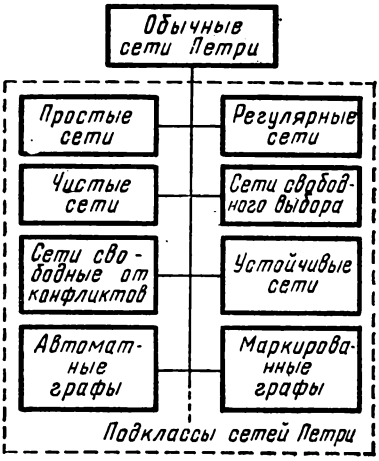



Рис. 3.11. Некоторые подклассы аппарата сетей Петри

Стремление уменьшить трудоемкость анализа сети при сохранении способности к моделированию конфликтных ситуаций между процессами явились причиной выделения подклассов простых сетей Петри, регулярных сетей Петри, чистых сетей Петри, сетей свободного выбора, сетей, свободных от конфликтов, а также устойчивых сетей Петри.

Условием выделения, например, подкласса *чистых сетей Петри* является

$$\forall d_j \in D \{\Phi(d_j) \cap H(d_j) = \varnothing\}, \qquad (3.8)$$

т. е. переход не может иметь позицию $b_i$ одновременно в качестве входной и выходной.

Если на сеть Петри накладывается ограничение, что каждая выходная дуга от позиции является либо ее единственным выходом, либо единственным входом перехода, т. е.

$$\forall b_i \in B \{(|\Phi(b_i)| = 1) \vee (\forall d_j \in \Phi(b_i)) \rightarrow (|\Phi(d_j)| = 1\};$$
$$j = \overline{1,m}; \quad i = \overline{1,n}; \quad m = |D|; \quad n = |B|, \qquad (3.9)$$

то такую сеть относят к подклассу *сетей свободного выбора.*

Признаком выделения подкласса *простых сетей Петри* является наличие у каждого перехода не более чем одной входной позиции, имеющей больше одного выхода, т. е.

$$\forall d_j \in D\{\forall b_i \in \Phi(d_j)/(|\Phi(b_i)| > 1) \leqslant 1\};$$
$$i = \overline{1,n}; \quad j = \overline{1,m}; \quad n = |B|; \quad m = |D|. \quad (3.10)$$

Для *устойчивых сетей Петри*

$$\forall M(b) \in R(N_M)\{\forall d_j, d_k | (\forall b_l \in (\Phi(d_j) \wedge \Phi(d_k))\{M(b_i) \geqslant$$
$$\geqslant 1\} \rightarrow (M \overset{d_j d_k}{|-} M') \vee (M \overset{d_k d_j}{|-} M')\}; \quad (3.11)$$
$$j, k = \overline{1,m}; \quad m = |D|,$$

т. е. если при любой маркировке $M(b)$, принадлежащей к множеству допустимых маркировок $R(N_M)$, два любых перехода $d_j$ и $d_k$ оказываются возбужденными, то срабатывание одного из них не исключает возможности срабатывания другого перехода.

Для *сетей Петри, свободных от конфликтов*

$$\forall b_i \in B\{(|\Phi(b_i)| = 1) \vee (b_i \in \Phi(d_j) \rightarrow b_i \in H(d_j))\}; \quad (3.12)$$
$$i = \overline{1,n}; \quad j = \overline{1,m}; \quad n = |B|; \quad m = |D|.$$

В этих сетях, если позиция $b_i$ принадлежит к подмножеству входных позиций перехода $d_j$, то она принадлежит и к подмножеству выходных позиций данного перехода, в противном случае $b_i$ должна иметь не более одной выходной дуги.

В [23] предложен подкласс *регулярных сетей Петри.* Вводится алгебра регулярных сетей, которая строится с помощью операций над сетями (наложение, разметка, слияние, итерация и т. д.) и класса элементарных сетей, что позволяет представить любую регулярную сеть в линейной формульной записи, однако при регуляризации сети вводится значительная избыточность и теряется наглядность модели.

В теории сетей Петри предложено также несколько расширений, ориентированных на увеличение моделирующих возможностей данного аппарата (рис. 3.12). Одним из расширений является *обобщенные сети Петри*

$$N_G = (B, D, \Phi, H, M_0),$$

где

$$\Phi: B \times D \rightarrow \omega_{\Phi} = \{0,1,2,...\};$$

$$H: D \times B \rightarrow \omega_{H} = \{0, 1, 2...\}. \qquad (3.13)$$

Граф обобщенной сети Петри имеет форму мультиграфа, в котором возможно несколько дуг между позициями и переходами, или же форму взвешенного графа с использованием весов дуг $\omega_{\Phi}$ и $\omega_{H}$. Такое расширение не увеличивает моделирующих свойств и обобщенные сети принципиально полностью эквивалентны обычным сетям Петри.

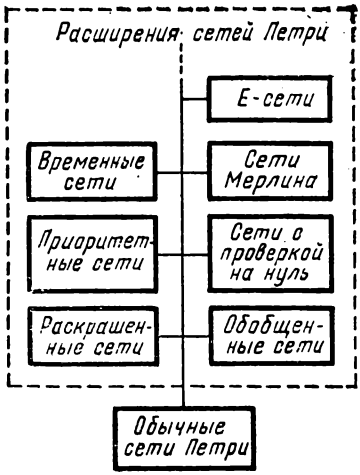



Рис. 3.12. Некоторые расширения аппарата сетей Петри

*Раскрашенные сети Петри* также по своим возможностям эквивалентны обычным сетям. Введение в данных сетях еще одного множества — раскрасок (цветов) меток позволяет уменьшить размерность графа при моделировании сложных систем.

*Приоритетные сети* и *сети с проверкой на нуль* позволяют учитывать в модели приоритетность событий. В сетях с проверкой на нуль для этой цели вводится дополнительное множество $\Phi_I$ прямых инцидентных дуг запрета, причем $\Phi \cap \Phi_I = \varnothing$ На графе сети Петри дуги, идущие от позиций к переходам, отображающие множество $\Phi_I$, называются дугами запрета. Для срабатывания некоторого перехода $d_j$ сети Петри с проверкой на нуль требуется кроме наличия меток во всех входных позициях $\Phi(d_j)$ также и отсутствие меток во входных позициях запрета $\Phi_I(d_j)$ данного перехода.

В приоритетных сетях Петри вводится специальная функция приоритетности, задающая соотношение приоритетов срабатывания для двух подмножеств переходов $D_\lambda$ и $D_0$, где $D_\lambda, D_0 \subset D$, $D \cup D_0 = D$; $D_\lambda \cap D_0 = \varnothing$.

При построении моделей очень важным является учет временных характеристик моделируемых событий. Два

расширения сетей Петри — временны́е сети и сети Мерлина — позволяют отразить в модели временны́е параметры системы.

Задание временно́й сети $N_S$ включает 7 множеств: $N_S=(B, D, \Phi, H, M_0, \upsilon, \nu)$, где $\upsilon=(t_1, ..., t_i, ...)$ — возрастающая последовательность действительных чисел, называемая временно́й базой. $\nu: B\times\upsilon\to\upsilon$ — функция временны́х задержек.

Фактор времени учитывается в $N_S$ путем введения пассивного состояния метки в позиции. При поступлении метки в позицию $b_i$ она остается в пассивном состоянии (не может участвовать в возбуждении переходов) на время $\nu(b_i, t_S)-t_S$ и только после этого переходит в активное состояние.

*Сеть Мерлина* задается как

$$N_J=(B, D, \Phi, H, M_0, \Lambda^*, \Lambda^{**}), \qquad (3.14)$$

где $\Lambda^*=\{t_i^*\}$ — множество времени минимальной задержки для переходов $d_i\in D$; $|\Lambda^*|=|D|=m$; $\Lambda^{**}=\{t_i^{**}\}$ — множество времен максимальной задержки для переходов $d_i\in D$; $|\Lambda^{**}|=|D|=m$;

$$t_i^*, t_i^{**}\geqslant 0, \quad i_i^*\leqslant t_i^{**}.$$

Срабатывание любого перехода $d_i$ сети Мерлина может наступить через время не менее $t_i^*$ после его возбуждения и не более $t_i^{**}$ после указанного события.

Временны́е сети и сети Мерлина не позволяют моделировать операции над данными, выполняемые в системе, не отражают зависимостей процессов обработки от типа или признаков задачи (сообщения). Учет указанных факторов возможен в наиболее мощном расширении сетей Петри — классе так называемых оценочных, или $E$-сетей.

*Е-сеть* задается совокупностью следующих множеств

$$N_E=(B, B_P, B_R, D, M_0), \qquad (3.15)$$

где $B$ — конечное множество позиций, $B\neq\varnothing$; $B_P$ — множество периферийных позиций, $B_P\subset B$; $B_R$ — множество решающих позиций, $B_R\subset B$; $D$ — конечное множество описаний переходов $d_i$, $D=\varnothing$, $d_i=(s, t(d_i), \rho)$ (здесь $s$ — тип перехода; $t(d_i)$ — время перехода; $\rho$ — процедура перехода).

Кроме учета фактора времени отличиями этого класса сетей Петри являются усложнение логики работы пе-

рехода, выделение нескольких базовых типов переходов, введение в модель различных операций над метками.

Таким образом, в аппарате сетей Петри можно выделить три класса, обладающих моделирующими свойствами, необходимыми для представления микропроцессорных устройств СПИ или их отдельных подсистем. Временны́е сети могут использоваться в тех случаях, когда возможно предположение о постоянном времени обработки для каждого события в модели. Сети Мерлина позволяют рассматривать случай равномерного распределения времени обслуживания в некотором заданном интервале. Наиболее же общий случай функционирования микропроцессорных устройств может моделироваться классом $E$-сетей.

### Временны́е сети Петри и сети Мерлина

Анализ временны́х сетей Петри $N_S$ производится при следующих ограничениях: 1. $N_S$ по своей структуре должна относиться к подклассу чистых сетей Петри, т. е. должно выполняться условие (3.8); 2. Временна́я задержка метки в позиции $b_i$ постоянна и не зависит от текущего времени, т. е. $\forall b_i \{\nu(b_i, t_k) - t_k = z_i = \text{const}\}$; $i = \overline{1, n}$; $n = |B|$; $k = \{0, 1, 2, \ldots\}$.

Для чистой сети Петри может быть определена *матрица инциденций* сети $C = \|c_{ij}\|$ размером $(n \times m)$, где $n = |B|$, $m = |D|$,

$$c_{ij} = \begin{cases} 1, \text{ если } H(d_j, b_i) = 1; \\ -1, \text{ если } \Phi(b_i, d_j) = 1; \\ 0 \text{ в противном случае.} \end{cases}$$

Матрицу C можно представить в виде

$$C = C^+ - C^-,$$

где $C^+ = \|c_{ij}^+\|$ размером $(n \times m)$,

$$c_{ij}^+ = \begin{cases} 1, \text{ если } H(d_j, b_i) = 1; \\ 0 \text{ в противном случае,} \end{cases}$$

$C^- = \|c_{ij}^-\|$ размером $(n \times m)$;

$$c_{ij}^- = \begin{cases} 1, \text{ если } \Phi(b_i, d_j) = 1; \\ 0 \text{ в противном случае.} \end{cases}$$

Разметка является во временно́й сети функцией от времени, поэтому разметку определяют, вводя понятие *переменной заряда* сети

$$\mathbf{Q}(t) = \begin{vmatrix} d_1(t) \\ \vdots \\ d_n(t) \end{vmatrix},$$

где $d_i(t)$ — разметка позиции $b_i$ в момент времени $t$.

Особенности динамики функционирования временно́й сети во времени отражаются введением *вектора тока* сети $I(t)$:

$$I(t) = \frac{\psi(t)}{\Delta t} = \begin{vmatrix} i_1 \\ \vdots \\ i_k \\ \vdots \\ i_m \end{vmatrix},$$

где $\psi(t)$ — вектор срабатываний переходов, $k$-й член которого $\psi_k(t)$ равен числу срабатываний перехода $d_k$ в общей последовательности срабатываний сети $\tau$; $i_k = \frac{\psi_k(t)}{\Delta t}$ — средняя частота срабатывания перехода $d_k$ за интервал времени $\Delta t$, равный $t - t_0$.

Исследование класса сетей Петри, проведенное в работах С. Рамчандани и Д. Сифакиса, позволило получить ряд аналитических выражений, характеризующих функционирование временно́й сети. Так, например, было показано, что при циклическом функционировании $N_S$ сеть работает с постоянными токами и суммарный заряд сети остается ограниченным. Токи через переходы сети могут быть найдены из уравнения

$$CI = 0 \text{ при } I > 0. \tag{3.16}$$

С начальной разметкой сети и задержками в позициях вектор $I$ связан следующим образом:

$$\{J_S' Q(t_0) \geqslant J_S' Z C^+ I\}_{S=1}^{k}, \tag{3.17}$$

где $Q(t_0)$ — переменная заряда в начальный момент функционирования сети;

$$\mathbf{Z} = \begin{vmatrix} z_1 & 0 & 0 & \dots & 0 \\ 0 & z_2 & 0 & \dots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \dots & z_n \end{vmatrix}$$ — квадратная матрица порядка $n$, задающая задержки в позициях сети;

$\{J'_S\}_{S=1}^{k}$ — генератор $G'$, представляющий собой набор таких векторов $J'_S$, что вектор $J'_0$, являющийся решением уравнения $J'_0 C = 0$, может быть найден как линейная комбинация $J'_S$ с неотрицательными коэффициентами пропорциональности, т. е. $J'_0 = \sum_{S=1}^{S=K} h_i J'_S$.

Режим работы временной сети, при котором сведены к минимуму взаимные блокировки меток, определяется как функционирование с естественной скоростью и описывается системой

$$CI = 0 \text{ при } I > 0; \tag{3.18}$$

$$\{J'_S Q(t_0) = J'_S ZC^+ I\}_{S=1}^{k}.$$

Для нахождения решения данной системы Д. Сифакисом предложены методы разбиения $N_S$ на элементарные состоятельные подсети и элементарные инвариантные подсети.

В сети $N_S$ можно выделить $d$-полную подсеть: $N_{S_1} = (B_1, D_1, \Phi_1, H_1, M_{0_1})$, задаваемую некоторым подмножеством $D_1 \subset D$. Тогда $B_1 = \forall b_i \in B\{\Phi(d_j) \cup H(d_j)\}$, где $d_j \in D_1$; $\Phi_1 : B_1 \times D_1$; $H_1 : D_1 \times B_1$.

Состоятельная подсеть $N_{S_1}$ сети $N_S$ определяется как $d$-полная подсеть, множество $D_1$ которой удовлетворяет условию $D_1 = \{d_j / i_{1j} \neq 0\}$, т. е. задается положительными членами вектора $I_1$, где $I_1 \in G$. Сеть $N_{S_1}$ называется в этом случае *поддержкой вектора* $I_1$, что записывается как $N_{S_1} = S(I_1)$. Элементарной состоятельной подсетью сети $N_S$ является подсеть $N_{S_1} = S(I_1)$, если не существует такого вектора $I_2$ ($I_2 \neq 0$, $I_2 \in G$), что $S(I_2) \subset S(I_1)$. Вектор $I_1$, определяющий элементарную подсеть, называется *элементарным вектором* $G$. За счет выделения элементарных состоятельных подсетей $N_{S_i}$ может быть определен генератор $G$ как минимальное по количеству членов множество элементарных векторов $I_S$.

Аналогичным образом в сети $N_S$ можно выделить $b$-полную подсеть $N_{S_1} = (B_1, D_1, \Phi_1, H_1, M_{0_1})$, которая задается указанием подмножества $B_1 \subset B$. Тогда $D_1 = \forall d_j \in D\{\Phi(b_i) \cup H(b_i)\}$, где $b_i \in B_1$; $\Phi_1 : B_1 \times D_1$; $H_1 : D_1 \times B_1$.

Инвариантная подсеть $N_{S_1}$ сети $N_S$ определяется как $b$-полная подсеть, множество $B_1$ которой удовлетворяет условию $B_1 = \{b_i / I_{1_i} \neq 0\}$, т. е. задается положи-

тельными членами $J_1' \in G$. Сеть $N_{S_1}$ называется в этом случае поддержкой вектора $J_1'$, что записывается $N_{S_1} S(J_1')$. Элементарным инвариантным компонентом сети $N_S$ называется подсеть $N_{S_1} = S(J_1')$, если не существует такого вектора $J_2'(J_2' \neq 0,\ J_2' \in G)$, что $S(J_2') \in S(J_1')$. Вектор $J_1'$, определяющий элементарную инвариантную подсеть, является элементарным вектором генератора $G'$. Выделение минимального множества векторов $J_S'$ позволяет найти $G$.

В качестве элементарных инвариантных подсетей удобны подсети класса автоматных графов. Для такой подсети решением уравнения $J'C = 0$ является вектор $J_0' = \{1, 1, ..., 1\}$.

Элементарными состоятельными подсетями целесообразно выбирать подсети класса маркированных графов. В этом случае решением уравнения $CI = 0$ является вектор $I_0' = \{1, 1, ..., 1\}$.

Таким образом, аппарат временны́х сетей Петри позволяет построить модель, отражающую динамику функционирования исследуемой системы с учетом времени выполнения различных функций. На такой модели могут быть получены аналитические зависимости для внешних и внутренних параметров микропроцессорных устройств СПИ. Однако использование временны́х сетей Петри при моделировании существенно ограничивается условием постоянства задержек $z_i$. Частично это ограничение снимается в классе сетей Мерлина.

Условие срабатывания перехода в сети Мерлина представлено в виде

$$\forall b_i \in B \{(M(b) - \Phi(b_i, d_j) \geqslant 0) \wedge (t_j^* \leqslant t_k \leqslant t_j^{**})\},$$

где $t_k$ — время, прошедшее с момента установления возбужденного состояния перехода $d_j$; $i = \overline{1, n}$; $j = \overline{1, m}$; $n = |B|$; $m = |D|$. Срабатывание перехода при сохранении возбужденного состояния происходит с равной вероятностью в интервале времени от $t_j^*$ до $t_j^{**}$.

Для анализа сетей Мерлина используется граф разметок, позволяющий отобразить множество $R(N_T)$ достижимых разметок. При анализе вводятся дополнительные переменные:

$T_{B_1}^*$, $T_{B_1}^{**}$ — минимальное и максимальное время в $N_T$, в течение которого элементы подмножества $B_1 \subset B$ содержат метки;

$T^{*}_{d_i}(M)$, $T^{**}_{d_i}(M)$ — минимальное и максимальное время в $N_T$, в течение которого сеть остается в состоянии $M$ до срабатывания перехода $d_j$.

Обычные сети Петри представляют собой частный случай сетей Мерлина при следующих значениях элементов множеств $\Lambda^*$ и $\Lambda^{**}$: $\forall d_j (t^{*}_j = 0, t^{**}_j = \infty)$.

Анализ сетей Мерлина позволяет получать как общие характеристики работы сети, так и частные условия достижимости отдельных переходов и разметок. Сети Мерлина нашли применение как аппарат исследования вопросов живучести и самовосстановления систем, а также для моделирования протоколов СПИ.

## Е-сети

Оценочные или $E$-сети были предложены Г. Натом как расширение сетей Петри и средство описания моделей функционирования вычислительных систем. Как уже отмечалось выше, класс сетей Петри обладает наибольшей моделирующей способностью и универсальностью и может быть использован для построения общих, отражающих наиболее существенные факторы, моделей микропроцессорных устройств СПИ.

Анализ этого класса сетей показывает, что как в структуре, так и в логике работы $E$-сети имеется ряд существенных отличий от остальных классов сетей Петри.

$E$-сети являются безопасными, т. е. $\forall b_i \in B\{M(b_i) \leqslant 1\}$, однако выполнение этого условия поддерживается в сети искусственно, за счет изменения логики работы перехода. В отличие от других классов сетей Петри в $E$-сети условие возбуждения перехода следующее:

$$\forall d_j \in D\,(\forall b_i \in \Phi_S(d_j) \to (M(b) - \Phi_S(b_i, d_j) = 0) \wedge \wedge \forall b_t \in H_S(d_j) \to (M(b) - H_S(d_j, b_t) = M(b))\}, \quad (3.19)$$

где $\Phi_S(d_j) \subset \Phi(d_j)$; $H_S(d_j) \subset H(d_j)$; $i = \overline{1, n}$; $j = \overline{1, m}$.

Состав подмножеств $\Phi_S(d_j)$ и $H_S(d_j)$ зависит от типа $E$-сетевого перехода $S$.

По логике работы $E$-сети являются сетями с приоритетами. Однако в отличие, например, от класса приоритетных сетей Петри приоритеты присваиваются в $E$-сети не группам переходов, а подмножествам общего множества $\Phi(d_j)$ для переходов определенных типов.

Структурно $E$-сети относятся к подклассу маркированных графов, так как $\forall\, b_i \in B/B_P\{|\Phi(b_i)| = |H(b_i)| = 1\}$.

Важной особенностью $E$-сети является детализация представления метки. С каждой меткой $k_l$ в $E$-сети связаны *n описателей,* что записывается как $k_l[n]$. Значение $i$-го описателя в позиции $b_k$ обозначается как $M(b_k(i))$. Каждый из описателей метки несет в себе определенную количественную информацию о моделируемом объекте.

Переход $E$-сети моделирует некоторое событие не только на уровне выполнения всех необходимых условий, но и отражает также ряд операций, связанных с данным событием, посредством модификации описателей меток. Набор операций и условия их выполнения задаются *процедурой перехода* $\rho$. В общем виде операция $\Xi$ над $i$-м описателем метки на переходе $b_k$ при истинности $j$-го предиката, задающего набор требуемых условий, может быть представлена как

$$l_{kij} = \{M(b_k(i)) := \Xi(M(b_k(i)))\}.$$

$E$-сеть задает конкретные детерминированные структуру модели и алгоритм ее функционирования, однако $E$-сетевая модель может включать в себя *множество внешних переменных* $\xi$, характер изменения которых не ограничивается. Поэтому $E$-сеть может служить как детерминированной, так и вероятностной моделью системы.

Для $E$-сетей определены пять основных типов переходов (рис. 3.13). Логика работы переходов задается указанием разрешенных смен разметок. Срабатывание перехода типа $T_E$ (рис. 3.13,*а*) происходит при наличии метки во входной позиции $b_1$ и отсутствии метки в выходной позиции $b_2$, т. е.

$$(1,0)\overset{T_E}{\vdash}(0,1).$$

Для перехода $F_E$ (рис. 3.13,*б*)

$$(1,0,0)\overset{F_E}{\vdash}(0,1,1).$$

Переход объединения типа $J_E$ (рис. 3.13*в*) описывается как

$$(1,1,0)\overset{J_E}{\vdash}(0,0,1).$$

Управляемый переход разветвления $X_E$ (рис. 3.13,*г*) с управляющей позицией $b_1 \in B_R$ задается соотношениями:

$$(0,1,0,0) \overset{X_E}{\vdash} (0,0,1,0);$$

$$(0,1,0,1) \overset{X_E}{\vdash} (0,0,1,1);$$

$$(1,1,0,0) \overset{X_E}{\vdash} (0,0,0,1);$$

$$(1,1,1,0) \overset{X_E}{\vdash} (0,0,1,1).$$

Приоритетный переход $Y_E$ (рис. 3.13, *д*), где $b_1 \in B_R$, задается следующим набором:

$$(0,1,1,0) \overset{Y_E}{\vdash} (0,0,1,1);$$

$$(0,1,0,0) \overset{Y_E}{\vdash} (0,0,0,1);$$

$$(0,0,1,0) \overset{Y_E}{\vdash} (0,0,0,1);$$

$$(1,1,1,0) \overset{Y_E}{\vdash} (0,1,0,1);$$

$$(1,1,0,0) \overset{Y_E}{\vdash} (0,0,0,1);$$

$$(1,0,1,0) \overset{Y_E}{\vdash} (0,0,0,1).$$

Приведенные выше пять основных типов переходов позволяют моделировать различные ситуации, встречающиеся в реальных системах. Переход $T_E$ моделирует событие, наступающее при выполнении одного условия. В случае необходимости двух условий используется переход $J_E$. Разветвление потока информации отображается в переходе $F_E$. При необходимости изменения направления потока информации по некоторому условию используется переход типа $X_E$. Переход типа $Y_E$ отражает приоритетность, устанавливаемую для одних потоков инфомации по отношению к другим. При этом в зависимости от внутренней логики позиции $b_i \in B_R$ возможно представление различных приоритетов для потоков меток: фиксированное, в виде функции от описателей меток или от внешних переменных системы.

Позиции $b_i \in B_R$ могут являться в сети как внутренними, так и периферийными. Если $b_i \in B_p$, то состояние позиции может быть нулевым $M(b_i)=0$, единичным $M(b_i)=1$ или же неопределенным $M(b_i)=\varnothing$. Для перевода позиции из неопределенного состояния в нулевое или единичное используется *решающая процедура* перехода

$$r:[\Pi_1 \rightarrow M(b_i):=\alpha;\quad \Pi_2 \rightarrow M(b_i):=1-\alpha],$$

где $\Pi_1$, $\Pi_2$ — предикаты, принимающие два значения (1 или 0) $\alpha=\{0, 1\}$.

Выполнение решающей процедуры начинается с вычисления $\Pi_1$. Если $\Pi_1$ истинно, то $M(b_i):=\alpha$ и далее

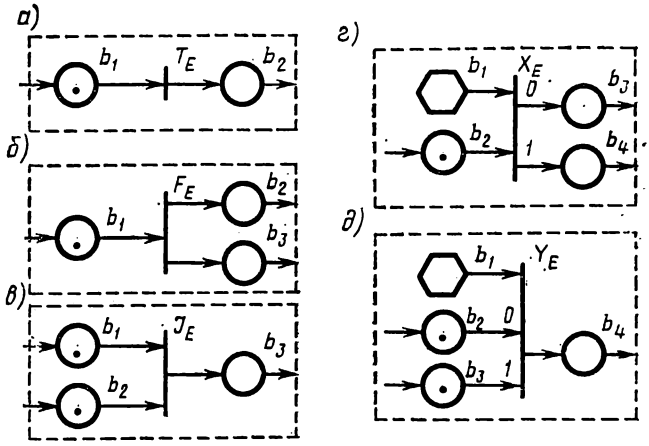



Рис. 3.13. Базовый выбор $E$-сетевых переходов

оценка не производится. Если же $\Pi_1$ ложно, то вычисляется $\Pi_2$. В том случае, если и $\Pi_2$ оказывается ложным, состояние $b_i$ остается неопределенным и повторное выполнение решающей процедуры производится только после изменения хотя бы одного из аргументов $\Pi_1$ или $\Pi_2$.

Если решающая позиция $b_i \in (B/B_p)$, то ее состояние устанавливается обычным для сетей Петри образом в результате срабатывания переходов сети.

Функционирование перехода $E$-сети определяется как последовательность четырех фаз.

Фаза *псевдоготовности* присутствует для тех переходов $d_j$, для которых $\exists b_i \in \Phi(d_j)/(b_i \in B_R \cap B_p)$. Она насту-

пает для $d_j$ после выполнения всех необходимых условий, связанных с состоянием позиций подмножеств $\Phi(d_j)$ и $H(d_j)$. В течение фазы псевдоготовности выполняется решающая процедура перехода. После установки состояния решающей позиции переход входит в фазу *готовности*, где производится вычисление $t(d_j)$ и затем устанавливается активная фаза.

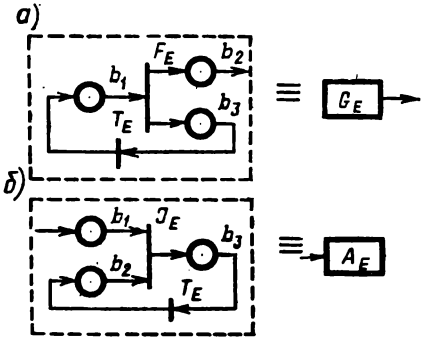



Рис. 3.14. Макропозиции генерации и поглощения меток

Длительность *активной* фазы перехода $d_j$ определяется вычисленным значением $t(d_j)$. На этой фазе производятся необходимые операции над описателями меток.

После истечения интервала $t(d_j)$ переход входит в *заключительную* фазу, на которой выполняется изменение разметки в соответствии с уравнениями перехода.

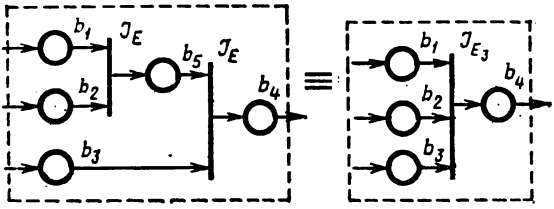


Рис. 3.15. Макропереход типа $J_{E3}$ с тремя входными позициями

Операции над описателями меток объединяются в $E$-сети в общее понятие процедуры перехода: $\rho:[\Pi_1 \rightarrow \rightarrow (l_{11}, l_{12}, \ldots, l_{1n}); \ldots; \Pi_s \rightarrow (l_{s1}, l_{s2}, \ldots l_{sm})]$, где $l_{11}, \ldots, l_{sm}$ — операции над описателями меток.

Составление $E$-сетевой модели системы значительно упрощается при использовании иерархического подхода к структуре модели, когда ряд наиболее часто встречающихся подсетей выделяется в качестве макроэлементов—

макропозиций и макропереходов. Необходимыми элементами $E$-сетевой модели являются обычно макропозиции генератора $G_E$ и поглощающая метки макропозиция $A_E$. На рис. 3,14, *а, б* показаны подсети, образующие данные $E$-сетевые макроэлементы.

На рис. 3.15 показана также процедура получения макроперехода типа $J_E$ с тремя входами $J_{E3}$. Более детально вопросы конструирования $E$-сетевых макроэлементов будут обсуждаться в гл. 4.

Таким образом, аппарат $E$-сетей, как показывает приведенный выше краткий обзор его возможностей, позволяет строить достаточно полные модели вычислительных систем, отражающие не только их структуру и логику работы, но и внутренние операции над данными.

## § 3.6. МЕТОД МАШИННОГО ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

Большое разнообразие возможных алгоритмов функционирования и сложность структур микропроцессорных устройств СПИ не всегда позволяют при проектировании свести устройство к математической модели, которая может быть исследована аналитическим методом даже с учетом возможных упрощающих предположений. Поэтому широкое распространение получили машинные методы исследования характеристик процесса функционирования микропроцессорных устройств СПИ на их имитационных моделях.

Имитационные модели позволяют рассматривать процессы, происходящие в устройствах СПИ, практически на любом уровне детализации. Для машинной реализации имитационной модели необходимо построить соответствующий моделирующий алгоритм, воспроизводящий в некотором масштабе времени процесс функционирования исследуемой системы (устройства) с той степенью детальности, которая необходима разработчику (исследователю), причем элементарные явления, составляющие процесс, имитируются с сохранением их логической структуры и последовательности протекания во времени.

При имитационном моделировании модель системы (устройства) задана на уровне связи реализаций входных и выходных процесов и внешних воздействий через характеристики проектируемой системы (характеристики блоков, составляющих систему и их связи); интересу-

ющие же разработчика характеристики выходных (в ряде случаев и промежуточных) процессов получают путем последующей обработки смоделированных реализаций этих процессов.

Термин «имитационное моделирование» не очень удачен, так как имитация сама по себе есть моделирование, а моделирование — имитация. Тем не менее этот термин устойчиво «прижился» в специальной технической литературе и часто отождествляется с терминами «машинная имитация», «машинный имитационный эксперимент», «имитационный эксперимент на ЭВМ», «моделирование на вычислительных машинах», «статистическое моделирование». Употребление последнего термина обусловлено тем, что статистическое моделирование как метод машинной реализации имитационных моделей систем, подверженных случайным воздействиям, в настоящее время наиболее употребим, хотя в детерминированных случаях имитационные модели реализуются и другими методами (например, метод «Динамо», метод ситуационного управления).

Статистическое моделирование представляет собой численный метод, дающий частное решение, отвечающее фиксированным значениям параметров системы, входной информации и начальным условиям. В основе его лежит процедура, применяемая для моделирования случайных величин и функций и носящая название *метода статистических испытаний* (метод Монте-Карло).

Общая схема метода Монте-Карло может быть записана в виде

$$\theta = \int y(x)\, p(x)\, dx \approx \hat{\theta} = \frac{1}{M} \sum_{i=1}^{M} y(x_i),\ x_i \sim p(x). \quad (3.20)$$

Результат ищется как математическое ожидание некоторой случайной величины $Y$, которая чаще всего является неслучайной функцией случайной величины $X$, имеющей распределение $p(x)$. Нестрогое выражение «случайная величина $X$ имеет распределение $p(x)$» и запись $X \sim p(x)$ означают для непрерывной случайной величины, что ее плотность вероятности равна $p(x)$; для дискретной случайной величины функцию $p(x)$ надо понимать как функцию вероятности. Для дискретной случайной величины интеграл (3.20) заменяется суммой $\Sigma y(x)\ p(x)$, в которой суммирование осуществляется по всем возможным значениям $X$. Функция $y(x)$ может

иметь несколько аргументов, т. е. зависеть от нескольких случайных величин. В таком случае запись (3.20) остается в силе, только интеграл надо считать многомерным, $X$ рассматривать как вектор, а $p(x)$ — как многомерную плотность (или функцию) вероятности. Приближенная оценка неизвестного математического ожидания, совпадающая с искомым результатом, находится как среднее арифметическое результатов независимых опытов. Это отражено в правой части (3.20). По закону больших чисел среднее арифметическое сводится к математическому ожиданию. В каждом опыте разыгрывается реализация $x$ случайной величины $X$ (в $i$-м опыте реализация $x_i$) в соответствии с распределением $p(x)$ и вычисляется значение функции в виде $y(x_i)$. Индекс $i$ подчеркивает, что для каждой ($i$-й) реализации процесса аргументы, составляющие вектор $X$, имеют свои случайные значения. Вычисленное очередное значение $y(x_i)$ добавляется к накапливаемой сумме $\Sigma y(x_i)$. На этом заканчивается очередной опыт. После того как проведено $M$ опытов, вычисляется итоговая оценка $\hat{\theta}=\frac{1}{M}\Sigma y(x_i)$. Опыты повторяются до тех пор, пока дисперсия оценки $\hat{\theta}$ не снизится до требуемой величины, зависящей от допустимой погрешности и коэффициента доверия.

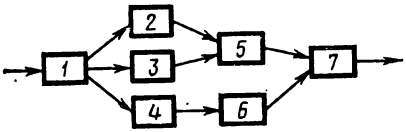


Рис. 3.16. Блочная структура системы

Проиллюстрируем суть метода Монте-Карло относительно простым примером. Пусть требуется оценить надежность системы (рис. 3.16). Система выполняет свою функцию, если работают цепочки блоков: 1, 2, 5, 7; 1, 3, 5, 7; 1; 4, 6, 7. Какие-то блоки могут отказать. Каждый блок характеризуется временем безотказной работы $\tau_i$, $i=\overline{1,7}$. Пусть заданы плотности распределения $p_i(\tau_i)$, $i=\overline{1,7}$. Какова надежность системы в целом? Рассмотрим случайную величину

$$\gamma = \min\{\tau_1, \max[\min(\tau_4, \tau_6), \min[\max(\tau_2, \tau_3), \tau_5]], \tau_7\} \quad (3.21)$$

$\gamma$ — время безотказной работы системы. В одном опыте разыгрывается значение $\tau_i$, $i=\overline{1,7}$ в соответствии с $p_i(\tau_i)$, $i=\overline{1,7}$. Через полученные реализации $\tau_i$, $i=\overline{1,7}$ по форму-

ле (3.21) вычисляем реализацию $\gamma$. Один опыт дает одну реализацию $\gamma$. Проводим $M$ опытов (испытаний), получаем «статистический» материал (выборку). Берем среднее арифметическое времени безотказной работы системы $\gamma_{ср}$ в качестве оценки надежности системы. При необходимости можно построить и закон распределения случайной величины $\gamma$.

Таким образом, испытания реальной системы заменены на испытания математической модели. Каждое испытание сопровождается расчетом. Поэтому имитационное моделирование и называют численным экспериментом

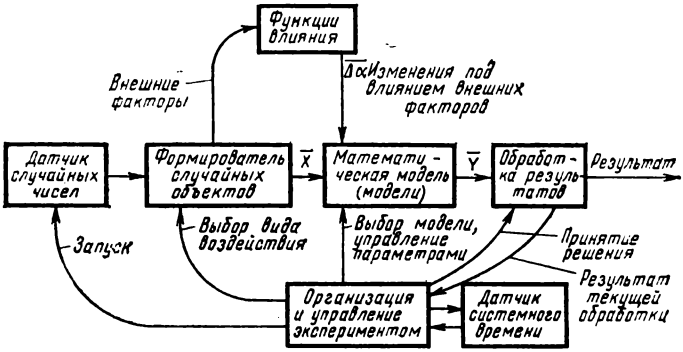



Рис. 3.17. Структурная схема эксперимента с имитационной моделью

на ЭВМ с математической моделью (модель выступает как объект испытания). При реализации испытания возможны и логические операции. И расчетные и логические операции реализуются на ЭВМ с помощью соответствующих алгоритмов, которые в совокупности составляют определенный выше моделирующий алгоритм.

Структурная схема эксперимента по имитационному моделированию показана на рис. 3.17.

Традиционно модель рассматривается как «черный ящик», имеющий входы $X$ и выходы $Y$. В качестве входов подразумевают параметры входных сигналов (например, параметры входных сообщений — тип, категория срочности, длина, время возникновения сообщения), в качестве выходов — различные характеристики, интересующие разработчика.

После выбора $X$ и $Y$ определяется план эксперимента, т. е. процедура выбора входных параметров $x_1, x_2, \ldots,$

на которые желательно знать реакцию модели. Само вычисление значений $y_1$, $y_2$, ... производится с помощью работы моделирующего алгоритма, позволяющего при заданных $x_1$, $x_2$, ... воспроизвести траекторию системы $Z(t)$, по которой вычисляются показатели $y_i$. В блоке обработки результатов происходит накопление сумм и статистическая обработка. Выбор плана эксперимента зависит от цели исследования. Если, например, интересует оптимизация некоторого критерия качества, то его значения являются выходом, а план заключается в выборе оптимизирующей последовательности $x_1$, $x_2$, ... в соответствии с определенным методом (например, градиентного поиска). Если необходимо выявить существенные параметры модели или выбрать структуру, то план заключается в изучении влияния того или иного параметра или вида структуры на выход модели.

Для формирования реализаций случайных объектов (событий, функций, процессов) используется датчик случайных чисел (ДСЧ), генерирующий в каждом опыте набор значений базовой случайной величины, на основе которых формируются реализации входных сигналов и влияющих внешних факторов. Функция влияния является характеристикой, отражающей действие внешних факторов на изменения $\overline{\Delta\alpha}$ параметров модели.

Величина цикла (число прогонов модели) может быть задана априори либо вычисляться по промежуточным результатам обработки; поэтому в ходе эксперимента может быть принято решение о прекращении исследования для заданного набора значений параметров модели.

Управление экспериментом сводится к установлению очередности системных событий, происходящих в процессе имитации, определению перехода к очередной реализации, к другому варианту исходных данных, изменению либо перестройке модели при смене задачи исследования.

При исследовании микропроцессорных устройств СПИ ставится задача определения большого числа показателей их функционирования. Эффективная оценка каждого показателя требует своей модели, своего моделирующего алгоритма. Здесь возможны два подхода. Первый связан с построением *универсальной автоматизированной имитационной модели* (УАИМ), модули которой могут настраиваться с разной степенью детализа-

ции, определяемой требованиями решаемой задачи. При этом каждый модуль и схема сопряжения должны быть реализованы с максимальной степенью детализации своих микрооператоров и связей. УАИМ в этом случае представляет сама по себе сложную программную систему, разработка и отладка которой требует значительных ресурсов, да и сам процесс настройки отдельных отлаженных модулей по ходу имитации сопряжен с большими затратами машинного времени. Второй подход связан с разработкой *библиотеки стандартных программных модулей,* реализованных с разной степенью детализации операторов. В этом случае каждому элементу соответствует множестно модулей, характеристики каждого из которых строго соответствуют уровню решаемой задачи. При этом настройка моделей сводится к компоновке модели из названных модулей, отвечающих условиям задачи исследования. Если в первом случае избыточность описания элементов внесена в модель системы и взаимодействие элементов строится с учетом этой избыточности, то во втором случае избыточность проявляется на уровне представления элементов и во взаимодействии не проявляется. В практике имитационного моделирования встречаются оба подхода.

При анализе вероятностно-временны́х характеристик устройств СПИ приходится сталкиваться с необходимостью оценки малых вероятностей, например вероятности отказа системы или вероятности ошибочного декодирования. Для получения оценок таких вероятностей имитационную модель рекомендуется строить с использованием методов ускоренного моделирования. Ускорить моделирование—значит добиться большей точности при том же числе опытов или получить требуемую точность при меньшем числе опытов. Точность результата моделирования измеряется его дисперсией. Чем меньше дисперсия, тем выше точность. Поэтому методы ускоренного моделирования чаще называют методами снижения дисперсии. Если при одном и том же числе опытов один метод дает дисперсию в $k$ раз меньшую, чем другой, значит для достижения заданной точности оценки первый метод требует в $k$ раз меньше опытов. Обычно ускоренные методы требуют на один опыт примерно столько же времени, сколько обычный метод, а иногда даже меньше. Следовательно, при сопоставлении одного метода моделирования с другим вместо затрат машинного времени можно сравнивать дисперсии

оценок при одном и том же числе опытов, если затраты времени на один опыт примерно одинаковы.

В практике ускоренного моделирования находят применение методы существенной выборки и расслоенной выборки. Идея *метода существенной выборки* заключается в замене функции $y(x)$ (3.20) другой функцией с тем же математическим ожиданием, но с меньшей дисперсией. Для сохранения математического ожидания необходимо соответственно изменить распределение аргумента. Схема метода существенной выборки выражается следующей записью:

$$\hat{\Theta} = \int y(x)\, p(x)\, dx = \int \frac{y(x)\, p(x)}{q(x)}\, q(x)\, dx \approx$$

$$\approx \hat{\Theta}_q = \frac{1}{M} \sum_{i=1}^{M} z(x_i), \qquad x_i \sim q(x), \qquad z(x) = \frac{y(x)\, p(x)}{q(x)}.$$

Математическое ожидание оценки $\hat{\Theta}_q$ совпадает с $\Theta$. Поскольку $y(x)$ и $p(x)$ считаются выбранными при первоначальной постановке задачи, необходимо выбрать $q(x)$ так, чтобы дисперсия оценки $\hat{\Theta}_q$ была как можно меньше. При этом должны быть выполнены требование $q(x) \neq 0$, если $y(x)p(x) \neq 0$, и условие нормировки

$$\int q(x)\, dx = 1. \tag{3.22}$$

Желательно $q(x)$ выбирать в виде

$$q(x) = q_0(x) = \frac{1}{c} |y(x)|\, p(x), \tag{3.23}$$

где постоянная $c$ находится из условия (3.22), т. е.

$$c = \int |y(x)|\, p(x)\, dx. \tag{3.24}$$

При таком выборе $q(x)$ оценка $\hat{\Theta}_q = \hat{\Theta}_0$ будет обладать наименьшей дисперсией. В действительности это возможно в принципе, но не реализуемо. Для нахождения $q(x)$ в виде (3.23) необходимо вычислить константу $c$ по формуле (3.24), а эта задача не менее трудна, чем исходная, так как интеграл в (3.24) очень похож на искомый интеграл (3.20). Тем не менее, хотя оптимальное распределение (3.23) невозможно реализовать, его вид указывает, к чему надо стремиться при выборе $q(x)$.

Чтобы уменьшить дисперсию оценки с помощью метода существенной выборки, надо использовать функцию $q(x)$, по возможности пропорциональную произведению $|y(x)|p(x)$ и при этом удовлетворяющую условиям нормировки. Если исходное распределение $p(x)$ предполагает примерно одинаковые вероятности выбора $x$ во всем диапазоне возможных значений, то новое распределение $q(x)$, пропорциональное $|y(x)|p(x)$, предписывает чаще выбирать существенные значения $x$ — те, при которых осредняемая функция $y(x)$ принимает большие значения; при этом учитывается и исходное распределение. Этим объясняется и название метода.

На практике, когда функция $y(x)$ задается сложным алгоритмом, а аргумент $x$ представляет собой вектор со случайным числом элементов, можно попытаться, не опираясь на приведенную выше рекомендацию (3.23), подобрать $q(x)$ методом проб и ошибок. Поиск $q(x)$ удобно вести в некотором классе распределений, зависящих от параметра. Важно, чтобы случайные числа с таким распределением легко генерировались на ЭВМ. В широко распространенном случае, когда аргументы являются равномерно распределенными в интервале $[0,1]$ случайными числами, удобно использовать семейство распределений с плотностью вероятности

$$q_a(x) = \begin{cases} \dfrac{\ln a}{a} a^x, & x \in [0,1]; \\ 0, & x \notin [0,1]. \end{cases}$$

Случайные числа с таким распределением получаются из равномерно распределенных чисел $u$ по формуле

$$x = \frac{1}{\ln a} \ln [(a-1)u + 1].$$

Значение параметра $a$, дающее наименьшую дисперсию оценки, ищется методом проб и ошибок на основе предварительных опытов.

Другим более распространенным методом ускоренного моделирования является *метод расслоенной (стратифицированной) выборки* (рис. 3.18).

Видно, что область значений аргумента $x$ может быть разбита на участки так, что изменение функции в пределах участка значительно меньше, чем в пределах всей области. В таком случае целесообразно сначала осреднить функцию на каждом участке отдельно, а затем из частных средних получить общее среднее с уче-

том неодинаковых вероятностей участков. Предполагается, что вероятности слоев $p_r$, $r=\overline{1, R}$ легко вычисляются через заданное распределение $p(x)$:

$$p_r = p\{x \in B_r\} = \int\limits_{B_r} p(x)\,dx, \quad r = \overline{1,R},$$

где $B_r$, $r=\overline{1, R}$ — подмножества множества $\{x\}$ значений аргумента $x$, называемые слоями (стратами).

Искомый результат $\theta$ представляется по формуле полной вероятности через условные математические ожидания

$$\theta = \sum_{r=1}^{R} p_r \theta_r = \sum_{r=1}^{R} p_r \int\limits_{B_r} y(x)\,\frac{p(x)}{p_r}\,dx.$$

Опыты проводятся, как в обычной схеме (3.20), но по слоям. Для каждого слоя вычисляются оценки условных математических ожиданий (частные оценки)

$$\hat{\theta}_r = \frac{1}{M_r} \sum_{i=1}^{M_r} y(x_i), \qquad x_i \sim \begin{cases} \dfrac{p(x)}{p_r}, & x \in B_r. \\ 0, & x \notin B_r, \end{cases}$$

где $M_r$ — количество опытов в $r$-м слое.

В конце вычисляется суммарная оценка $\hat{\theta}_\Sigma = \sum\limits_{r=1}^{R} p_r \hat{\theta}_r$.

Теория показывает: чтобы оценка $\hat{\theta}_\Sigma$ имела меньшую дисперсию, чем обычная оценка (без расслоения), достаточно, чтобы объем выборки распределялся по слоям пропорционально их вероятностям и чтобы условные средние по слоям не были одинаковы. При разбиении по слоям надо стремиться к тому, чтобы средние по слоям как можно больше отличались друг от друга.

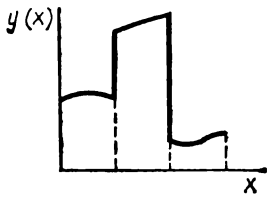



Рис. 3.18. Кусочно-непрерывная функция $y(x)$ случайного аргумента $x$

В реальных задачах аргумент $X$ является вектором, и разбиение на слои надо проводить в многомерном пространстве. Обычно это делают с помощью некоторой функции $h(x)$, называемой параметром расслоения. Разбивают множество значений $h(x)$ на подмножества $B_r^{(h)}$, $r=\overline{1, R}$ и считают $x$, относящимся к слою $B(r)$, если

$h(x) \in B_r^{(h)}$. Зависимость осредняемой функции $y(x)$ от параметра расслоения $h(x)$, как правило, носит вероятностный характер. Поэтому в качестве параметра расслоения следует выбирать функцию, сильно коррелированную с осредняемой функцией, и назначать слои путем разбиения множества значений параметра расслоения на последовательно расположенные отрезки.

Часто возникают трудности с определением вероятности слоев, что является основным препятствием метода расслоенной выборки. Здесь может помочь преобразование исходного распределения по аналогии с методом существенной выборки или расслоение после выборки. В последнем случае можно генерировать $x$ в соответствии с исходным распределением $p(x)$ и определять принадлежность $x$ к тому или иному слою после того, как значение получено. При этом объемы выборок по слоям окажутся случайными, но в среднем они будут пропорциональны вероятностям слоев, так что при удачном выборе параметра расслоения оценка $\hat{\theta}_\Sigma$ должна иметь меньшую дисперсию.

Оптимальный вариант метода существенной выборки не удается реализовать. Неоптимальный вариант подчас приводит к широкому диапазону значений усредняемой функции. Можно попытаться компенсировать большой разброс осредняемой функции путем расслоения выборки, для чего необходимо знать вероятности слоев. После применения метода существенной выборки (в неоптимальном варианте) новое распределение оказывается иногда более простым, чем исходное. Это дает возможность вычислить вероятности слоев аналитически. В этом смысле оба метода могут эффективно дополнять друг друга.

Независимо от конкретных способов моделирования и разных модификаций в области машинной реализации моделирующих алгоритмов существуют некоторые общие методологические принципы имитационного моделирования на ЭВМ. За основу такой методики обычно принимают следующие этапы имитационного моделирования: 1) изучение объекта моделирования; 2) формулировка задачи исследования; 3) построение математической модели; 4) составление программы для ЭВМ; 5) оценка достоверности и пригодности модели; 6) планирование и обработка эксперимента.

В процессе создания имитационной модели выделяют

три уровня описания; концептуальный, математический и программный, именуемые соответственно: концептуальная модель, математическая модель и программная модель.

На этапе построения *концептуальной модели* формулируется замысел модели. Основным содержанием этого этапа является переход от общего описания системы к ее математической модели, т. е. процесс формализации. Этот этап является эвристическим, т. е. разработчик в основном руководствуется лишь собственной интуицией, опирающейся на накопленные знания и понимание процесса функционирования системы. Достоверность полученной новой информации зависит от адекватности модели, т. е. от того, насколько правильно и полно модель отражает черты исследуемой системы, влияющие на искомый результат. Модель всегда дает упрощенное представление о системе. Говорить об адекватности модели можно лишь по отношению к определенной задаче. Поэтому при построении концептуальной модели очень важно четко конкретизировать: что моделируется и для чего (с какой целью).

Распространенными *математическими схемами формализации* функционирования ВС, к классу которых можно отнести микропроцессорные устройства СПИ, являются системы и сети массового облуживания, высказывательные формы, вероятностные автоматы, сети Петри, *Е*-сети, агрегаты. Наибольшее применение нашли СМО и сети СМО, сети Петри и их модификации, *Е*-сети.

Основным результатом этапа построения математической модели является моделирующий алгоритм, позволяющий имитировать функционирование системы во времени и осуществлять программную реализацию модели.

*Программные средства* построения имитационных моделей, применяемые в практике имитационного моделирования, развиваются в основном по двум направлениям: создание и использование языков моделирования и разработка специализированных систем моделирования с использованием алгоритмических языков общего назначения. Каждое из этих направлений имеет свои достоинства и недостатки.

В процессе применения моделирования выявились многочисленные проблемы его использования, вызванные не способом записи модели, а ее содержанием. Кардинальными являются вопросы адекватности, точности, чувствительности к вариациям различных параметров

модели. Для созданной модели возникают проблемы использования ее «внутренних» свойств, таких, как устойчивость, надежность, управляемость и др. Решение указанных проблем предполагает проведение целенаправленных экспериментов, использующих не только возможности ЭВМ по имитации, но и аналитические результаты исследования соответствующих математических моделей, проведенные качественными методами. Для первоначального ознакомления с методами решения названных проблем можно обратиться к работам В. В. Калашникова [20, 21]. Перспективными с точки зрения ускорения моделирования являются комбинированные методы определения вероятностных характеристик, базирующиеся на совместном использовании оценок, полученных аналитическим и имитационным методами [37].

# ГЛАВА 4
# СИСТЕМНОЕ И ТЕХНИЧЕСКОЕ ПРОЕКТИРОВАНИЕ МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВ СПИ

## § 4.1. ВЫБОР И ОБОСНОВАНИЕ КРИТЕРИЯ ЭФФЕКТИВНОСТИ ДЛЯ ПРОЕКТИРУЕМЫХ МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВ СПИ

Одной из важнейших процедур системного этапа проектирования МПУ СПИ является выбор критерия оптимизации $K$. Функционирование МПУ СПИ в общем случае оценивается некоторым множеством показателей эффективности $\mathbf{K}=\{k_i\}$, $i=\overline{1,m}$, причем обычно $|\mathbf{K}|=m\gg 1$. Под *эффективностью системы* понимают меру соответствия ее своему назначению, а *показатели эффективности* дают количественную оценку эффективности системы. В роли показателей эффективности системы могут выступать характеристики ее качества, т. е. некоторые числовые характеристики, являющиеся функционалом от процесса функционирования системы и определяющие одну из сторон качества этого процесса. Аргументами процесса функционирования системы являются ее параметры (в том числе и структурные). Таким образом, показатель эффективности $k_i$, $i=\overline{1,m}$ позволяет количественно оценить влияние параметров системы на то, как система справляется с задачами, для решения которых она предназначена. В свою очередь, количествен-

ные значения показателей эффективности позволяют сравнить между собой различные варианты реализации системы (которые могут отличаться как параметрами, так и структурой) и выбрать из них оптимальный либо наилучший из всевозможных.

Для успешного синтеза оптимального МПУ СПИ необходимо выбрать такое подмножество $\tilde{\mathbf{K}} \subset \mathbf{K}$, которое бы наиболее полно характеризовало различные свойства устройства с учетом его назначения и предъявлямых к нему требований. С другой стороны, для облегчения задачи синтеза микропроцессорного устройства СПИ желательно по возможности ограничить подмножество $\tilde{\mathbf{K}}$ показателями, позволяющими в наибольшей степени оценить способности устройства выполнять возложенные на него задачи с точки зрения его основного назначения. МПУ СПИ являются многоцелевыми (см. § 2.3). Однако каналы реализации терминально-пользовательских функций, функций обмена с УВВ, сервисных функций не являются специфическими для устройств СПИ. На этапе системного проектирования целесообразно для их реализации принять существующие стандартные решения. Поэтому возможные реализационные варианты МПУ СПИ уместно сравнивать по показателям эффективности, позволяющим в наибольшей степени оценить способность устройств выполнять возложенные на них задачи в рамках основного их назначения, каким является реализация сетевых функций, связанных с обработкой сообщений в процессе их транспортировки по сети с обеспечением заданных скоростей приема и передачи. Множество показателей эффективности МПУ СПИ целесообразно разбить на две группы: *целевого использования* $\mathbf{K}_{\text{ц}}$ (показатели эффективности) и *технико-экономической целесообразности* $\mathbf{K}_{\text{т}}$ (показатели качества). Показатели подмножества $\mathbf{K}_{\text{ц}}$ характеризуют степень выполнения основного назначения МПУ СПИ, группа $\mathbf{K}_{\text{т}}$ отражает технико-экономические качества устройства при выполнении им основного назначения. При этом $\tilde{\mathbf{K}} = \{k_i\} = \mathbf{K}_{\text{ц}} \cup \mathbf{K}_{\text{т}}$; $i = \overline{1, \tilde{m}}$, $\tilde{m} < m$. В общем случае показатели как первой, так и второй групп являются сложными функционалами вида $k_i = k_i(\Theta_{\text{п}}, \Theta_{\text{с}}, \Theta_{\text{д}}, \Theta_{\text{ср}})$, где $\Theta_{\text{п}}$, $\Theta_{\text{с}}$, $\Theta_{\text{д}}$, $\Theta_{\text{ср}}$ — соответственно множества параметров входящих потоков задач, структурных параметров, параметров режимов работы МПУ, внешней среды.

Рассмотрим возможный подход к формированию $\mathbf{K}$ для МПУ СПИ. По своему целевому назначению МПУ СПИ является элементом СОИ, и от его информационных параметров во многом зависит качество функционирования всей сети. К СОИ со стороны АСУ предъявляются три основных требования к доставке сообщений: по надежности, времени и верности. Эти характеристики могут быть обобщены введением понятия нормальной доставки внешнего сообщения в заданный конечный адрес за время, не превышающее нормы времени его доставки при соблюдении нормы степени соответствия для сообщений данной категории. Тогда в роли единого интегрального показателя качества СОИ выступает $\mathbf{K}_{\text{н}}$ — норма надежности нормальной доставки сообщения заданного объема $V_0^s$: $\mathbf{K}_{\text{н}}=p_q\{T_q\leqslant T_q^{\text{н}}\}$, где $P_q$ — вероятность нормальной доставки сообщения объемом $V_0^s$, $T_q$ — время доставки сообщения, $T_q^{\text{н}}$ — норма времени доставки сообщения.

Из внешних информационных показателей устройства СПИ наибольшее влияние на $\mathbf{K}_{\text{н}}$ оказывают два параметра — *время задержки* $T_{\text{з}}$ *сообщения* при его приеме — передаче в устройстве и *вероятность потери* $P_{\text{пот}}$ *сообщения* из-за ошибок в канале связи. Поскольку для СОИ в случае $|\alpha^s|>1$ ($\alpha^s$ — категория срочности сообщения) устанавливаются различные категории $\mathbf{K}_{\text{н}i}$, то можно ввести группы параметров $T_{\text{з}}=\{T_{\text{з},i}\}$ и $P_{\text{пот}}=\{P_{\text{пот}i}\}$; $i=\overline{1,\ |\alpha^S|}$. Таким образом, для МПУ СПИ $\mathbf{K}_{\text{ц}}=T_{\text{з}}\cup P_{\text{пот}}$.

При выборе элементов подмножества $\mathbf{K}_{\text{т}}$ необходимо также задать параметр, который характеризовал бы объемную сложность системы и качество ее аппаратурной и программной реализации. Параметр сложности $S_{\text{мпу}}$ программируемого устройства СПИ должен отражать как аппаратную сложность управляющего комплекса устройства, так и степень совершенства используемой элементной базы.

Характеристика аппаратурной сложности системы не имеет однозначного определения. Общий объем аппаратуры МПУ СПИ можно представить в виде $V^a=\sum_{i=1}^{3} V_i^a$, где $V_1^a$ — суммарный объем электронных компонентов, $V_2^a$ — объем соединений на всех конструктивных уровнях, $V_3^a$ — объем несущей конструкции,

обеспечивающей прочность и защиту элементов на всех уровнях. Использование БИС существенно уменьшает составляющие $V_1^a$ и $V_2^a$ (так, БИС МП позволяет исключить до 1500 внешних соединений по сравнению с реализацией на ИС). Уменьшается также составляющая $V_3^a$, так как сокращается число конструктивных уровней; при этом уменьшение $V^a$ оказывается пропорционально изменению $\sum_i n_i$, где $n_i$ — число выводов микроэлектронного $i$-го компонента (включая выводы заземления и питания). Аналогичная зависимость наблюдается и для весовых характеристик устройства. Уменьшение $\Sigma n_i$ для микроэлектронных систем является решающим фактором повышения их надежности, так как внешние «паяные» соединения оказываются наименее надежными компонентами этих систем. Таким образом, для ориентировочной оценки *аппаратурной сложности* МПУ СПИ целесообразно использовать параметр $S_{\text{мпу}}=\Sigma n_i$, поскольку он функционально связан с важнейшими аппаратурно-реализационными параметрами устройств СПИ. Уменьшение $S_{\text{мпу}}$ ведет к уменьшению веса и габаритов, повышению надежности МПУ. С точки зрения качества программной реализации функций процесса приема — передачи сообщений важным параметром является требуемый объем памяти $V^{\text{п}}$ при обработке блоков (информационных кадров) в МПУ СПИ.

Основным показателем группы $\mathbf{K}_{\text{т}}$ для МПУ СПИ является *стоимость* (оптовая цена) устройства, которая может быть представлена в виде $C_{\text{мпу}}=[C_{\text{ки}}+C_{\text{м}}+C_{\text{з}}(1+k_\alpha)](1+k_\beta)$, где $C_{\text{ки}}$ — стоимость комплектующих изделий, $C_{\text{м}}$ — стоимость материалов, $C_{\text{з}}$ — заработная плата рабочих; $k_\alpha$ — коэффициент накладных расходов; $k_\beta$ — коэффициент внепроизводственных расходов. В качестве стоимостного параметра может выступать критерий полных затрат $\mathbf{K}_{\text{пз}}=C_{\text{мпу}}+T_{\text{с}}W_{\text{э}}$, где $T_{\text{с}}$ — срок службы системы, $W_{\text{э}}$ — годовые эксплуатационные расходы, учитывающие заработную плату производственного персонала, расходы на материалы и запасные части, затраты на электроэнергию, амортизационные отчисления и прочие производственные расходы.

Таким образом, подмножество $\tilde{\mathbf{K}}$ для МПУ СПИ можно задать в виде следующего комплекса элементов $\tilde{\mathbf{K}}=\{T_{\text{з}1},\ldots, T_{\text{з}\tilde{m}}, P_{\text{пот}1}, \ldots, P_{\text{пот}\tilde{m}}, V^{\text{п}}, S_{\text{мпу}}, C_{\text{мпу}}\}$, где $\tilde{m}=|\alpha^s|$.

Синтез оптимального МПУ СПИ является задачей векторной оптимизации. Методика векторной оптимизации к настоящему времени даже в общем виде не разработана. Одним из предложенных приемов решения задач векторной оптимизации является построение интегрального критерия эффективности. Наибольшее распространение получили усредненные комплексные показатели средневзвешенного арифметического, геометрического или гармонического типов. Анализ показывает, что из этих оценок наилучшим является *средневзвешенный геометрический комплексный показатель* в виде $\prod_i \left(\frac{k_i^{min}}{k_i}\right)^{\nu_i}$, где $k_i^{\min}$ — минимальные значения соответствующих показателей качества по всем допустимым вариантам МПУ, СПИ, $\nu_i$ — нормированные весовые коэффициенты приоритетности соответствующих показателей, величина которых определяется экспертными методами; причем $\forall\, i \nu_i \geqslant 0$ и $\sum_i \nu_i = 1$.

В зависимости от назначения МПУ, наличия аппаратурных и программных ресурсов возможно относительное перераспределение значений весовых коэффициентов $\nu_i$. Наибольшие трудности вызывают назначение веса вероятности потери информации относительно весов других показателей. Поэтому целесообразно интегральный критерий эффективности МПУ СПУ представить в виде

$$\widetilde{K}_{\text{мпу}} = \left\{\left(\frac{T_{\text{з}_1}^{min}}{T_{\text{з}_1}}\right)^{\nu_1} \dots \left(\frac{T_{\text{з}\widetilde{m}}^{\min}}{T_{\text{з}\widetilde{m}}}\right)^{\nu_{\widetilde{m}}} \left(\frac{V_{\min}^{n}}{V^{n}}\right)^{\nu_{\widetilde{m}+1}} \times \right.$$

$$\left. \times \left(\frac{S^{min}}{S_{\text{мпу}}}\right)^{\nu_{\widetilde{m}+2}} \left(\frac{C^{min}}{C_{\text{мпу}}}\right)^{\nu_{\widetilde{m}+3}}\right\}$$

$$\text{при } P_{\text{пот}_i} \leqslant P_{\text{пот}_i}^{\text{доп}}, \quad i = \overline{1,\widetilde{m}},$$

где $P_{\text{пот}_i}^{\text{доп}}$ — допустимое значение вероятности потери сообщения $i$-й категории, $\sum_{i=1}^{m+3} \nu_i = 1$. В свою очередь, допустимым вариантом реализации МПУ СПИ считается такой, для которого $T_{\text{з}_i} \leqslant T_{\text{з}_i}^{\text{доп}}$, $1 = \overline{1,\widetilde{m}}$. Оптимальным из допустимых вариантов будет тот, который обеспечивает максимальное значение $\widetilde{K}_{\text{мпу}}$, причем $\widetilde{K}_{\text{мпу}} = 1$. Таким образом, $\widetilde{K}_{\text{мпу}}$ представляет собой нелинейную функцию

от нормированных безразмерных параметров $\{k_i\}$. В зависимости от весовых коэффициентов можно строить МПУ СПИ, оптимизируя по одному из следующих параметров: пропускной способности, затратам памяти, сложности аппаратурной реализации, стоимости реализации.

## § 4.2. ВЫБОР ТИПА МИКРОПРОЦЕССОРНОГО СРЕДСТВА

При разработке микропроцессорных устройств СПИ одним из первых этапов алгоритма проектирования (см. рис. 3.2) является выбор некоторого ограниченного ряда МП семейств БИС. Выполнение данного системного этапа производится после выбора определенного класса физических структур реализации устройства СПИ в микропроцессорном базисе.

Как уже отмечалось ранее, основные элементы МП семейств БИС — микропроцессоры — характеризуются множеством технико-экономических характеристик. При комплексной оценке качества МП должны учитываться следующие основные параметры:

1) время выполнения команд, определяющее быстродействие МП;

2) количество команд МП, определяющее гибкость его применения и универсальность;

3) число внутренних регистров, определяющее вычислительные способности МП;

4) емкость адресуемой памяти, определяющая максимальный объем обрабатываемой информации;

5) потребляемая (рассеиваемая) мощность, определяющая сложность источников питания и необходимость отвода теплоты;

6) количество различных уровней напряжения питания, определяющее сложность и стоимость микропроцессорной системы;

7) возможности прерывания, характеризующие мультипрограммный режим работы МП;

8) наличие канала прямого доступа к памяти;

9) наличие микропрограммного управления, обеспечивающего возможность изменения системы команд в зависимости от области применения;

10) наличие средств программирования и отладки, упрощающих и сокращающих время написания программ.

Наряду с этими параметрами в ряде случаев существенное значение имеют стоимость, наличие полной номенклатуры микропроцессорных БИС, редакционно-отладочных комплексов, пакетов прикладных программ, разрядность МП, наличие стековой памяти, ограничения по температурному диапазону, механическим и другим характеристикам.

Из перечисленных выше параметров для микропроцессорных устройств СПИ важнейшим является **быстродействие**. В этом случае все остальные параметры МП (стоимость, разрядность, емкость адресуемой памяти, количество источников питания и т. д.) выступают в процессе выбора как ограничения, т. е. не должны выходить за пределы, устанавливаемые разработчиком в результате анализа и уточнения ТЗ на устройство СПИ. Проведенный таким образом анализ характеристик доступных МП и МП-семейств позволяет выделить некоторое подмножество моделей (семейств), отвечающих всей совокупности ограничений на параметры, за исключением важнейшего параметра — быстродействия.

Задача оценки МП по быстродействию является достаточно сложной. В настоящее время не существует даже общепринятого параметра, характеризующего быстродействие МП. В каталогах и документации на МП-семейства БИС приводятся различные обобщенные параметры, которые не могут служить действительной основой сравнения МП. Например, такие как:

максимальная частота тактовых импульсов, которая в первую очередь характеризует технологию, но без учета особенностей архитектуры не может выступать как непосредственная оценка;

время цикла или время состояния МП, которое мало о чем говорит, так как для выполнения сложных команд небходимы многочисленные состояния МП и несколько циклов;

минимальное время выполнения команды (например, сложения «регистр—регистр»);

число команд, в очень небольшой степени характеризующее эффективность системы команд для некоторого конкретного применения.

Определение *реальной производительности* (быстродействия) является, как известно, актуальной задачей уже с момента появления ЭВМ. Одним из путей определения производительности служит применение спе-

циальных смесей Гибсона, учитывающих конкретную область применения ЭВМ. Например, наиболее распространенная смесь Гибсона № 3 учитывает только систему команд ЭВМ и обеспечивает оценку по показателю $p$—среднему времени выполнения эталонной операции:

$$p = \frac{1}{100} \sum_{r=1}^{k} h_r t_r,$$

где $h_r$ — весовые коэффициенты; $t_r$ — время выполнения команды типа $r$; $k$ — число анализирующих команд.

Смесь Гибсона № 3 учитывает 13 типов команд основных арифметических операций с фиксированной и плавающей запятой, операций выборки команд из ЗУ, сравнения и условного перехода.

Другими типами подобных оценок являются, например, смесь GAMM, оценка POWU и т. д. Основным недостатком такого рода оценок является необходимость введения множества произвольных исходных данных (типы команд, весовые коэффициенты). В связи с этим наилучшим способом для получения быстродействия МП является применение оценочных программ «benchmark».

Бенчмарковская программа — это программа решения на анализируемом МП такой задачи, которая по составу операций соответствует классу задач предполагаемого применения. Обычная длина бенчмарковской программы — 100...200 команд. В состав ее обязательно должны входить операции по вводу и выводу информации. Важным достоинством процедуры выбора МП на основе тестовых программ является то, что она не только определяет время решения задачи на конкретном МП, но и вскрывает достоинства и недостатки его системы команд для заданной области применения.

Выбранное в результате анализа ограничений допустимое подмножество МП семейств БИС упорядочивается по ориентировочному быстродействию МП. Исследование начинается с МП наименьшего быстродействия — бл. 2 (рис. 4.1).

Составив для заданного МП бенчмарковскую программу, следует проверить ее правильность на числовых примерах, а затем вычислить время ее выполнения для

грех случаев: простейшего, среднего и наихудшего. Определение времени выполнения программы производится следующим образом. Сначала нужно разделить бенчмарковскую программу на блоки. При этом в отдель-

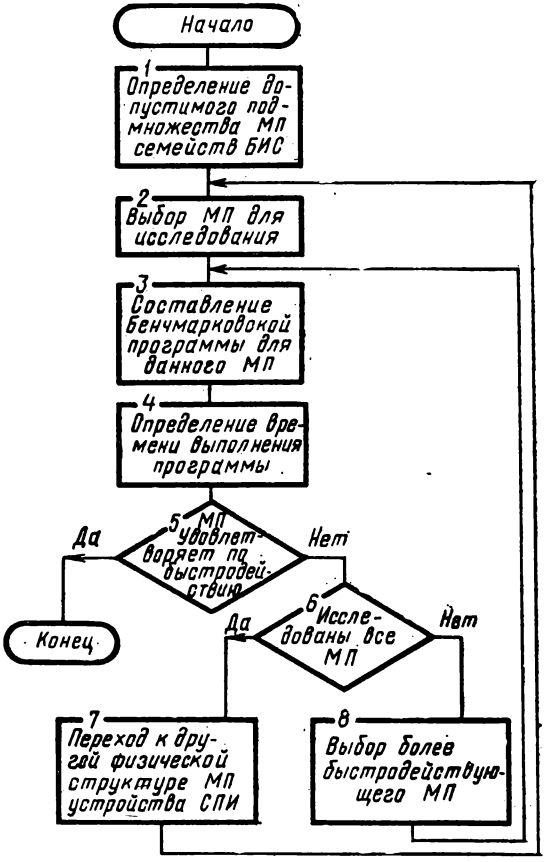



Рис. 4.1. Процедура выбора МП с использованием бенчмарковских программ

ные блоки должны быть выделены каждая линейная последовательность команд с одним входом и одним выходом, каждая команда условной передачи управления и каждая команда, следующая за командой условной передачи управления (если она пропускается при выполнении условия). Затем определяется время вы-

полнения каждого блока суммированием времен выполнения всех составляющих блок команд. После этого можно определить время выполнения бенчмарковской программы для простейшего, среднего и наихудшего случаев, суммируя произведения времен выполнения блоков на частоты их повторения. Поскольку в составленной программе могут быть не учтены все виды обработки, характерные для реального применения, целесообразно несколько увеличить полученные оценки по бенчмарковской программе (вплоть до их удвоения). По вычисленному таким образом значению можно сделать вывод о возможности использования данного МП для рассматриваемого применения.

Если проверка (бл. 5) покажет, что временные показатели неудовлетворительны, используется один из двух путей: 1) повторяется описанная выше процедура для МП с более высокой производительностью; 2) если все допустимое подмножество МП уже исследовано, то призводится выбор более совершенной физической структуры МП устройства СПИ и продолжается исследование, как было описано выше.

## § 4.3. ОГРАНИЧЕНИЕ МНОЖЕСТВА ДЕТАЛЬНО ИССЛЕДУЕМЫХ ПРОГРАММНО-АППАРАТНЫХ СТРУКТУР УСТРОЙСТВ СПИ

Одним из важных вопросов при разработке микропроцессорных устройств СПИ является ограничение числа детально исследуемых программно-аппаратных структур устройства, т. е. получение подмножества допустимых программно-аппаратных структур $\{\gamma\}_{\text{доп}}$. Такая предварительная проверка производится на 9-м шаге системного этапа проектирования (см. рис. 3.2). Воспользуемся для этой цели аппаратом *временны́х сетей Петри* (см. § 3.5) и рассмотрим методику построения и исследования модели на одном из простейших примеров, приведя затем общий алгоритм проведения данного этапа.

В качестве примера построим модель одного из двух микропроцессорных устройств СПИ — программируемого АП с магистральной организацией (рис. 4.2). АП содержит периферийные адаптеры входного и выходного каналов связи $ПА_{кс_1}$ и $ПА_{кс_2}$, а также одно устройство ввода — вывода — электрифицированную пишущую

машинку (ЭПМ), сопрягающуюся с магистралью АП через периферийный адаптер $ПА_{ЭПМ}$. Общее управление АП и обработка информации осуществляются МП.

Построим модель данного АП для двух основных задач $\{a_i\}$: $a_1$ — ввод сообщений из $КС_1$, их обработка и вывод на УВВ оператору АП; $a_2$ — прием информации от оператора через ЭПМ, ее обработка и вывод в $КС_2$.

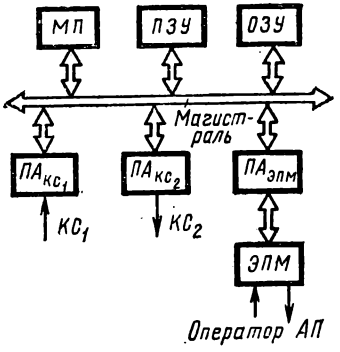



Рис. 4.2. Структура микропроцессорного АП

Модель строится в предположении о детерминированных по времени и объему потоках сообщений. В моделях присутствуют два параллельных потока задач $a_1$ и $a_2$, которые взаимодействуют друг с другом при обращении к общим ресурсам (МП, ЭПМ и буферный накопитель). Распространенным способом отображения таких конфликтных ситуаций являются *семафоры Дийкстры*. Семафором $S_D$ называют целочисленную переменную, над которой определяются две операции:

$$P(S_D) = S_D := S_D - 1, \text{ если } S_D > 0;$$

$$V(S_D) = S_D := S_D + 1.$$

Кроме того, вводится ограничение на недопустимость одновременного выполнения более чем одним процессом операций $P(S_D)$ или $V(S_D)$. Процесс, пытающийся выполнить операцию $P(S_D)$ над $S_D$, значение которого равно нулю, переводится в состояние ожидания до тех пор, пока некоторый другой процесс не выполнит над $S_D$ операцию $V(S_D)$.

Сеть Петри позволяет отражать механизм семафоров Дийкстры путем ввода специальных позиций $bs_D$, связанных с участками сети, моделирующими взаимодействующие процессы. Разметка позиции $b_{S_D}$ отражает текущее значение семафора $S_D$. Таким образом,

$$P(S_D) \leftrightarrow M\left(b_{S_D}\right) := M\left(b_{S_D}\right) - 1;$$

$$V(S_D) \leftrightarrow M\left(b_{S_D}\right) := M\left(b_{S_D}\right) + 1;$$

$$M_0\left(b_{S_D}\right) \geqslant 1.$$

Ограничения на $P(S_D)$ и $V(S_D)$ учитываются самими правилами функционирования сети Петри.

Рассмотрим случай динамического распределения памяти в микропроцессорном АП. Модель на основе временно́й сети Петри для данного случая приведена на рис. 4.3. Подсети ($b_1$, $b_2$, $d_1$, $d_2$) и ($b_{11}$, $b_{12}$, $d_8$, $d_9$) моделируют процессы поступления на АП потока задач $a_1$ и $a_2$. Подсеть ($b_3$, $b_4$, $d_2$, $d_3$) отражает работу $ПА_{кс_1}$, а подсеть ($b_{17}$, $b_{18}$, $d_{13}$, $d_{14}$) — $ПА_{кс_2}$. Работа МП по обработке двух потоков задач моделируется подсетью ($b_6$, $b_9$, $b_{15}$, $d_4$, $d_5$, $d_{11}$, $d_{12}$). Позиция $b_9$ играет роль семафора, предоставляющего доступ к МП. Функционирование ЭПМ и $ПА_{эпм}$ отражается подсетью ($b_8$, $b_{10}$, $b_{13}$, $d_6$, $d_7$, $d_9$, $d_{10}$), где семафором является позиция $b_{10}$. Общий буферный накопитель моделируется позицией $b_{19}$ с разметкой $M(b_{19}) = N_{в_0}$, где $N_{в_0}$ интерпретируется как число свободных блоков ОЗУ фиксированной емкости.

Построенная модель относится к классу чистых сетей Петри, и следовательно, может быть задана матрицей инциденций C:

$$C = \begin{vmatrix}
-1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\
0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1
\end{vmatrix}$$

Формальное задание временно́й сети Петри включает также матрицу задержек Z, элементы главной диагона-

ли которой образуют вектор

$$\overset{\vee}{\mathbf{Z}} = \{z_{\text{вх}_1}\, 0\, z_{RC_1}\, 00 z_{\text{мп}_1}\, 0 z_{\text{пм}_1}\, 000\, z_{\text{вх}_2}\, z_{\text{пм}_2}\, 0 z_{\text{мп}_2}\, 0 z_{\text{кс}_2}\, 00\}.$$

В матрице $Z$ величины $z_{\text{вх}_1}$ и $z_{\text{вх}_2}$ задают интервалы между поступлениями на микропроцессорный АП задач $a_1$ и $a_2$. Задержки на обработку в $\text{ПА}_{\text{кс}_1}$ и $\text{ПА}_{\text{кс}_2}$ отражаются временами $z_{\text{кс}_1}$ и $z_{\text{кс}_2}$. Средние времена занятия МП на обработку $a_1$ и $a_2$ задаются величинами $z_{\text{мп}_1}$ и $z_{\text{мп}_2}$. Элементы $z_{\text{пм}_1}$ и $z_{\text{пм}_2}$ отражают соответственно время работы ЭПМ для $a_1$ и $a_2$.

Полное задание модели включает также разметку $\mathbf{M}_0$;

$$\mathbf{M}_0 = \{0101000011100000001\, N_{B_0}\}.$$

Матрица **C**, **Z** и вектор $\mathbf{M}_0$ полностью задают модель исследуемой структуры АП в форме временно́й сети Пет-

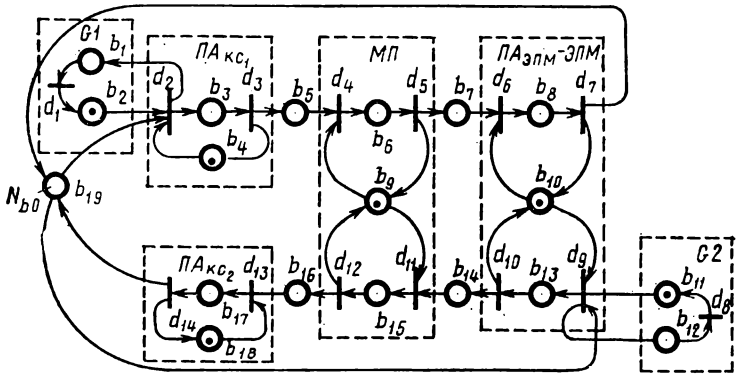



Рис. 4.3. Модель микропроцессорного АП на базе временной сети Петри

ри $N_S$. Для анализа $N_S$ удобно воспользоваться методом разбиения сети на инвариантные подсети (см. § 3.5). На рис. 4.4 приведено множество $\{N_{S_1}, \ldots, N_{S_r}\}$ элементарных инвариантных компонент, на которые разлагается данная $N_S$. Все $N_{S_i}$ относятся к классу автоматных графов, поэтому, как показано Д. Сифакисом, множество элементарных векторов генератора G′ может быть легко получено, исходя из того, что для каждой $N_{S_i}$ решением уравнения $\mathbf{J}^1\mathbf{C}=0$ является вектор $\mathbf{J}^1_0=\{111\ldots1\}$. Таким образом, G′ для данной $N_S$ задается матрицей

$$\mathbf{D}^e = \begin{vmatrix} J_{10} \\ J_{20} \\ J_{30} \\ J_{40} \\ J_{50} \\ J_{60} \\ J_{70} \end{vmatrix} = \begin{vmatrix} 1100000000000000000 \\ 0011000000000000000 \\ 0010111100001111101 \\ 0000000000110000000 \\ 0000000000000000110 \\ 0000010010000010000 \\ 0000000101001000000 \end{vmatrix}.$$

Функционирование временно́й сети $N_S$ в циклическом режиме при максимальной загрузке находится как решение (3.17):

$$i_1 = i_2 = i_3 = i_4 =$$
$$= i_5 = i_6 = i_7 = i_a;$$
$$i_8 = i_9 = i_{10} = i_{11} =$$
$$= i_{12} = i_{13} = i_{14} = i_b;$$
$$i_a z_{\text{вх}_1} = 1;$$
$$i_a z_{\text{КС}_1} = 1;$$
$$i_a z_{\text{КС}_1} + i_a z_{\text{МП}_1} + i_a z_{\text{ПМ}_1} +$$
$$+ i_b z_{\text{ПМ}_2} + i_b z_{\text{МП}_2} +$$
$$+ i_b z_{\text{КС}_2} = N_{b_0};$$
$$i_b z_{\text{вх}_2} = 1;$$
$$i_b z_{\text{КС}_2} = 1;$$
$$i_a z_{\text{МП}_1} + i_b z_{\text{МП}_2} = 1;$$
$$i_a z_{\text{МП}_1} + i_b z_{\text{ПМ}_2} = 1.$$

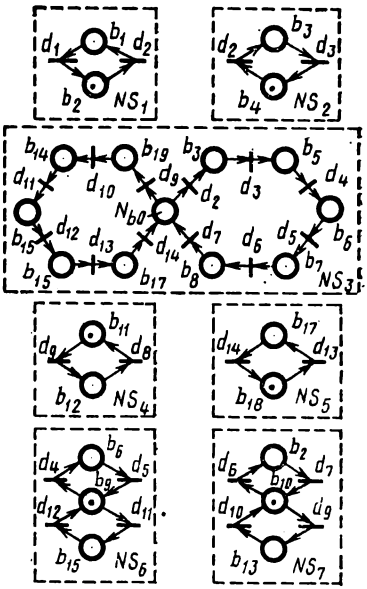



Рис. 4.4. Элементарные инвариантные подсети, выделенные в модели микропроцессорного АП

Введем обозначение: $\gamma_{\text{мп}} = z_{\text{мп}_1}/z_{\text{мп}_2}$. Тогда решение данной системы уравнений дает условия функционирования $N_S$ с полной загрузкой:

$$z_{\text{вх}_1} = z_{\text{КС}_1}; \quad z_{\text{вх}_2} = z_{\text{КС}_2};$$
$$N_{b0} = \left[ \frac{z_{\text{МП}_2} + z_{\text{ПМ}_2}}{z_{\text{вх}_2}} + \frac{z_{\text{МП}_1} + z_{\text{ПМ}_1}}{z_{\text{вх}_1}} + 2 \right];$$

$$z_{\text{МП}_1} = \frac{\gamma_{\text{МП}}\left(z_{\text{МП}_1} z_{\text{вх}_2} + z_{\text{ПМ}_2} z_{\text{вх}_2}\right)}{\gamma_{\text{МП}} z_{\text{вх}_2} + z_{\text{вх}_1}}.$$

Однако в реальном случае при использовании устройств из некоторого фиксированного ряда функционирование с полной загрузкой всех устройств недостижимо. Такой режим работы $N_S$ описывается соотношениями (3.16; 3.17). Введем обозначение $\gamma_i = i_a / i_b$, тогда максимально допустимые токи в модели определяются как

$$i_{a\max} = \min\left\{\frac{1}{z_{\text{вх}_1}};\quad \frac{1}{z_{\text{КС}_1}};\quad \frac{\gamma_i}{\gamma_i z_{\text{МП}_1} + z_{\text{МП}_2}};\right.$$

$$\frac{\gamma_i}{\gamma_i z_{\text{МП}_1} + z_{\text{ПМ}_2}};$$

$$\left.\frac{\gamma_i N_{b_0}}{z_{\text{КС}_2} + z_{\text{МП}_2} + z_{\text{ПМ}_2} + \gamma_i\left(z_{\text{КС}_1} + z_{\text{МП}_1} + z_{\text{ПМ}_1}\right)}\right\};$$

$$i_{b\max} = \min\left\{\frac{1}{z_{\text{вх}_2}};\quad \frac{1}{z_{\text{КС}_2}};\quad \frac{1}{\gamma_i z_{\text{МП}_1} + z_{\text{МП}_2}};\quad \frac{1}{\gamma_i z_{\text{ПМ}_1} + z_{\text{ПМ}_2}};\right.$$

$$\left.\frac{N_{b_0}}{z_{\text{КС}_2} + z_{\text{ПМ}_2} + z_{\text{МП}_2} + \gamma_i\left(z_{\text{КС}_1} + z_{\text{МП}_1} + z_{\text{ПМ}_1}\right)}\right\}.$$

Для нормального функционирования $N_S$ необходимо, чтобы минимальными членами приведенных выше выражений являлись $1/z_{\text{вх}_1}$ и $1/z_{\text{вх}_2}$, т. е. обеспечивалась обработка заданных потоков задач $a_1$ и $a_2$. При условии $a_{\max} = 1/z_{\text{вх}_1}$; $i_{b\min} = 1/z_{\text{вх}_2}$ могут быть определены коэффициенты загрузки блоков модели:

$$\Psi_{\text{КС}_1} = \frac{z_{\text{КС}_1}}{z_{\text{вх}_1}};\quad \Psi_{\text{КС}_2} = \frac{z_{\text{КС}_2}}{z_{\text{вх}_2}};$$

$$\Psi_{N_{b_0}} = \frac{z_{\text{вх}_2}\left(z_{\text{КС}_1} + z_{\text{МП}_1} + z_{\text{ПМ}_1}\right) + z_{\text{вх}_1}\left(z_{\text{КС}_2} + z_{\text{МП}_2} + z_{\text{вх}_2}\right)}{N_{b_0} z_{\text{вх}_1} z_{\text{вх}_2}};$$

$$\Psi_{\text{МП}} = \frac{z_{\text{МП}_1} z_{\text{вх}_2} + z_{\text{МП}_2} z_{\text{вх}_1}}{z_{\text{вх}_1} z_{\text{вх}_2}};$$

$$\Psi_{\text{ПМ}} = \frac{z_{\text{ПМ}_1} z_{\text{вх}_2} + z_{\text{ПМ}_2} z_{\text{вх}_1}}{z_{\text{вх}_1} z_{\text{вх}_2}}.$$

Для оценки структуры микропроцессорного АП по выбранному критерию необходимо получение показателей целевого использования $\text{К}_\text{ц} = \{T_{\text{з}_1}, T_{\text{з}_2}, P_{\text{пот}_1}, P_{\text{пот}_2}\}$.

Времена задержки информации в АП для потоков задач $a_1$ и $a_2$ могут быть представлены из решения системы уравнений:

$$T_{з_1} = \frac{z_{КС_2} + z_{МП_2} + z_{ПМ_2} + \gamma_i (z_{КС_1} + z_{МП_1} + z_{ПМ_1})}{\gamma_i N_{b_0}} +$$

$$+ z_{КС_1} + \frac{z_{МП_2}}{\gamma_i} + z_{МП_1} + \frac{z_{ПМ_2}}{\gamma_i} + z_{ПМ_1};$$

$$T_{з_2} = \frac{z_{КС_2} + z_{МП_2} + z_{ПМ_2} + \gamma_i (z_{КС_1} + z_{МП_1} + z_{ПМ_1})}{N_{b_0}} +$$

$$+ \gamma_i z_{ПМ_1} + z_{ПМ_2} + \gamma_i z_{МП_1} + z_{МП_2} + z_{КС_2}.$$

Если $i_{a_{max}} < 1/z_{вх_1}$ и (или) $i_{b_{max}} < 1/z_{вх_2}$, то происходит потеря части информации на входе АП. Коэффициенты потерь информации для потоков задач $a_1$ и $a_2$ равны $\Psi_{p_1} = 1 - i_{a_{max}} z_{вх_1}$; $\Psi_{p_2} = 1 - i_{b_{max}} z_{вх_2}$.

Используя полученные показатели, можно провести предварительную проверку соотношений, заданных параметрами $\Phi_у$:

$$T_{з_1} \leqslant [T_{з_1}]; \quad T_{з_2} \leqslant [T_{з_2}];$$

$$\Psi_{P_1} \leqslant [P_{пот_1}]; \quad \Psi_{P_2} \leqslant [P_{пот_2}].$$

Общий алгоритм предварительной оценки элементов множества программно-аппаратных структур $\{\gamma\}$ содержит следующие этапы:

1. Выбор подмножества $A_1 \subset A_2$ задач микропроцессорного устройства СПИ, которые в наибольшей степени влияют на загрузку устройства. При этом учитывается критичность задачи $a_i$ к реальному масштабу времени и элементы множества временных параметров функций $\{t_{fk}\}$, относящиеся к данным $a_i$.

2. Для выбранной структуры $\gamma_i$ проводится разработка графа модели микропроцессорного устройства на базе временно́й сети Петри $N_S$.

3. Сеть $N_S$ задается формально матрицами инциденций **C**, задержек **Z** и вектором начальной разметки $\mathbf{M}_0$.

4. Производится выделение в сети $N_S$ элементарных инвариантных или состоятельных подсетей. Строится матрица $D^e$ (или $B^e$) задающая элементарные составляющие генератора решения.

5. Решение системы уравнений (3.18) для условия полной загрузки модели позволяет получить выражения для верхней границы времени задержки в основных обслуживающих блоках модели и оценить минимальные емкости буферных накопителей.

6. Анализ модели при заданных $\{z_l\}$ на основе соотношений (3.16) и (3.17) позволяет получить коэффициенты загрузки $\Psi_l$ для основных блоков устройства и не рассматривать далее структуры с низкими коэффициентами загрузки.

7. Получение на модели параметров $T_{з_l}$ и $\Psi_{р_l}$ позволяет провести проверку соотношений $\Phi_\gamma$, отбросив структуры с недопустимыми по ТЗ параметрами.

8. Формирование подмножества $\{\gamma\}_{доп}$ выполняется с использованием критерия $К_{мпу}$, путем выбора элементов $\{\gamma\}$ с наибольшими значениями комплексного показателя.

Таким образом, использование аппарата временных сетей Петри позволяет получить аналитические соотношения между основными параметрами микропроцессорного устройства СПИ и произвести формирование допустимого подмножества программмно-аппаратных структур.

## § 4.4. ОПРЕДЕЛЕНИЕ БАЗОВОЙ СТРУКТУРЫ УСТРОЙСТВА МЕТОДОМ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

Важнейшим этапом разработки микропроцессорного устройства СПИ является синтез субоптимальной структуры, который ведется путем анализа множества допустимых структур (см. рис. 3.2, шаг 12). Как уже отмечалось, обычный процедурный подход к формализованному заданию имитационной модели сложной системы, при котором модель представляется в виде некоторого алгоритма, отражающего процесс функционирования системы, обладает рядом недостатков, среди которых большая трудоемкость процесса описания модели, отсутствие наглядного соответствия между структурой $\varepsilon_l$ и программой, имитирующей ее работу, сложность изменения модели при переходе к анализу альтернативной структуры $\varepsilon_j$.

Применение аппарата $E$-сетей позволяет осуществить *структурный подход* к построению имитационной модели микропроцессорного устройства, при котором обеспечиваются наглядность модели, модульный принцип ее разработки (сборки), возможность перехода к автоматизированной интерактивной процедуре проектирования.

## Отображение основных элементов и процедур микропроцессорного устройства в $E$-сетевой модели

В § 3.5 были кратко описаны основные элементы аппарата $E$-сети. Остановимся на вопросе использования $E$-сетевых элементов как средства отображения процедур микропроцессорных устройств СПИ.

Представим переходы $d_i = (S, t(d_i), \rho)$ $E$-сети как элементарные блоки модели устройства и покажем, какие процедуры могут отображаться $d_i$ в зависимости от значений элементов $S$, $t(d_i)$ и $\rho$.

1. Переход типа $T_E$ (см. рис. 3.13) позволяет отразить в модели занятость некоторого устройства или подсистемы на время, определяемое параметром $t(T_E)$. Про-

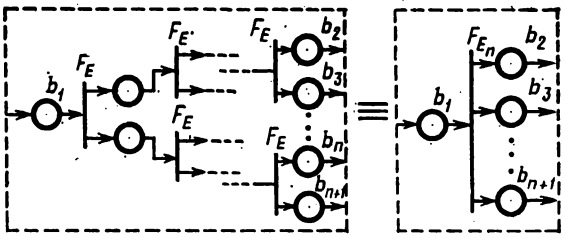



Рис. 4.5. Макропереход $F_{E_n}$

цедура перехода $\rho(d_i)$ задает набор операций для описателей меток, проходящих через переход, и условия их выполнения. К числу операций могут относиться изменение признаков длины сообщения, его приоритетности, порядкового номера и т. д.

2. Переход типа $F_E$ (см. рис. 3.13) кроме функций перехода $T_E$ отражает также разветвление одного потока информации на два направления. Это может, например, моделировать процедуру распределения обработки задачи на два МП, выработку управляющего сигнала одновременно с окончанием обработки и т. д.

Данный переход может быть обобщен в форме макроперехода $F_{E_n}$-распределителя на $n$ направлений (рис. 4.5). Логика работы $F_{E_n}$ описывается соотношением

$$(1,0,0,\ldots,0) \overset{F_{E_n}}{\vdash} (0,1,\ldots,1).$$

3. Переход типа $J_E$ (см. рис. 3.13) моделирует ситуацию объединения двух потоков информации. С его помощью может быть отражена процедура, выполняемая

только при наличии управляющей команды, необходимых исходных данных или ресурсов. Возможное макрорасширение этого перехода $J_{E_n}$ описывается соотношением

$$(1,1,\ldots,1,0) \vdash^{J_{E_n}} (0,0,\ldots,0,1).$$

На рис. 3.15 показан макропереход $J_{E_3}$.

4. В микропроцессорных устройствах СПИ ряд процедур требует наличия нескольких исходных массивов, команд на выполнение, сигналов о наличии необходимых

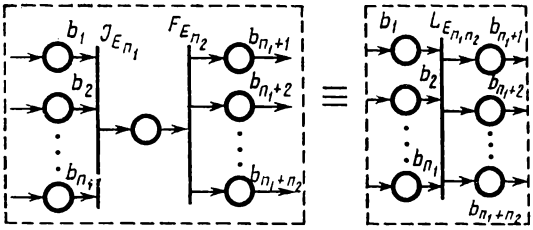



Рис. 4.6. Макропереход $L_{E_{n_1n_2}}$ как объединение макропереходов $J_{E_{n_1}}$ и $F_{E_{n_2}}$

ресурсов, а результатом их выполнения является также набор массивов, команд и сигналов. Такую ситуацию можно описать при объединении переходов $J_{E_n}$ и $F_{E_n}$ (рис. 4.6). Обозначим такой макропереход как $L_{E_{n_1n_2}}$. Функционирование $L_{E_{n_1n_2}}$ задается соотношением

$$(1,\ldots,1,0,\ldots,0) \vdash^{L_{E_{n_1n_2}}} (0,\ldots,0,1,\ldots 1).$$

5. Переход типа $X_E$ (см. рис. 3.13) позволяет отразить внешнее или внутреннее управление потоком информации, ее передачу после обработки на одно из двух направлений. Решающая позиция $b_i \in B_R$ может быть как периферийной ($b_i \in B_p$), так и внутренней ($b_i \in B/B_p$). Возможно задание двух элементов $t(X_E)$ — для каждого из направлений передачи. В модели микропроцессорного устройства переход $X_E$ позволяет моделировать процедуры обработки, зависящие от внутренних признаков задания (сообщения), изменение направления потока информации в зависимости от некоторой функции $\xi_i \in \xi$, которая может быть произвольной, в том числе и вероятностной.

Для передачи информации более чем на два направления удобно использовать макропереход $X_{E_n}$ (рис. 4.7), который определяется как

$$(1, 1, 0, f, \ldots, f) \overset{X_{E_n}}{\vdash} (e^0, 0, 1, f, \ldots, f);$$

$$(2, 1, f, 0, f, \ldots, f) \overset{X_{E_n}}{\vdash} (e^0, 0, f, 1, f, \ldots, f);$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$(n, 1, f, \ldots, f, 0) \overset{X_{E_n}}{\vdash} (e^0, 0, f, \ldots, f, 1),$$

где $f$ — служит обозначением разметки позиции $b_i$, равной $M(b_i) = 1 \vee 0$; $e^0$ обозначает разметку решающей по-

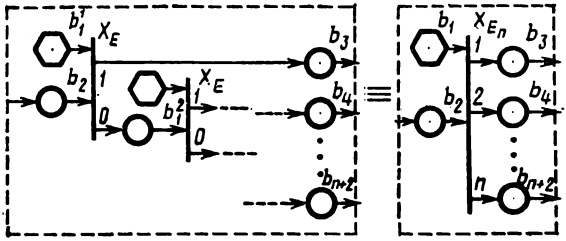



Рис. 4.7. Макропереход $X_{E_n}$

зиции $b_i \in B_R$, равной $M(b_i) = 0$, если $b_i \in B/B_p$ или $M(b_i) = \varnothing$, если $b_i \in B_p$.

Определение решающей процедуры является расширением (3.20):

$$r: [\Pi_1 \to M(b_i) := 1; \quad \Pi_2 \to M(b_i) := 2; \quad \ldots;$$
$$\Pi_n \to M(b_i) := n].$$

В модели микропроцессорного устройства макропереход $X_{E_n}$ удобно использовать, например, для отражения процедур распределения входного потока задач в зависимости от их типа по обрабатывающим модулям, направление задач на свободный в данных момент времени прибор и т. д.

**6.** Переход типа $Y_E$ (см. рис. 3.13) моделирует приоритетность обработки двух потоков заданий. В связи с тем, что в устройствах СПИ в общем случае $|a_i| > 2$, при моделировании часто требуется макропереход $Y_{E_n}$, име-

ющий произвольное число входов. На рис. 4.8 показано одно из возможных макрорасширений — переход $Y_{E_n}$, процедура смены разметок которого описывается соотношениями:

$$(m, f, \ldots, f, \{M(b_{m+1})=1\}, f, \ldots f, 0) \overset{Y_{E_n}}{\vdash}$$

$$(e^0, f, \ldots, f, \{M(b_{m+1})=0\}, f, \ldots, f, 1);$$

$$(m, 0, \ldots, 0, 1, f, \ldots, f, \{M(b_{m+1})=0\},$$

$$1, f, \ldots, f, 0) \overset{Y_{E_n}}{\vdash}$$

$$(e^0, 0, \ldots, 0, 0, f, \ldots, f, \{M(b_{m+1})=0\}, 1, f, \ldots, f, 1),$$

т. е. при $M(b_1)=m$, если имеется метка в позиции $b_{m+1}\in\Phi(Y_{E_n})$, то эта метка передается из $b_{m+1}$ в позицию $b_{n+2}\in H(Y_{E_n})$, в противном случае в выходную позицию $b_{n+2}$ передается метка из $b_i\in\Phi(Y_{E_n})\,|\,M(b_i)=1$ с наименьшим значением номера $i$.

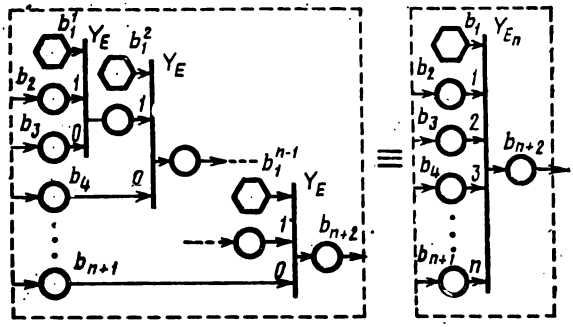



Рис. 4.8. Макропереход $Y_{E_n}$

Аналогичным образом могут быть составлены макропереходы с произвольной логикой смены приоритетов.

**7.** Переходы типа $Y_{E_n}$ позволяют моделировать приоритетность обслуживания заданий, но не отражают возможности прерывания обслуживания. Для этой цели может использоваться переход $T_E$ с прерыванием — $TI_E$ (рис. 4.9).

Позиции $b_1$ и $b_3$ перехода $TI_E$ идентичны входной и выходной позициям перехода $T_E$. Позиции $b_2$ выполняет

роль прерывающей обслуживание. Работа перехода описывается соотношениями

$$(1, 0, 0) \overset{TI_E}{\vdash} (0, 0, 1);$$

$$(1, 1, 0) \overset{TI_E}{\vdash} (0, 0, 1).$$

В переходе $TI_E$ производится прерывание активной фазы перехода, поэтому при его использовании в модели устройства необходимо выделение описателя метки и операции $l_i$ для записи оставшегося после прерывания времени обслуживания $t'(TI_E)$. Операция $l_i$ в процедуре $\rho(TI_E)$ должна выполнять присвоение

$$M(b(i)) := M(b(i)) + t'(TI_E).$$

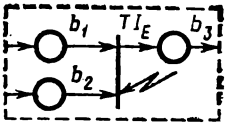



Рис. 4.9. Переход $TI_E$

8. К наиболее часто используемым объектам моделей микропроцессорных устройств СПИ относятся очереди типа *FIFO*, которые позволяет отразить макропозиция $Q_E$ (рис. 4.10), представляющая собой последовательность переходов $T_E$. В форме макропозиций могут быть заданы и очереди с другой дисциплиной обслуживания.

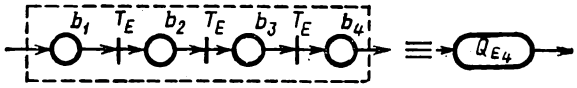



Рис. 4.10. Макропозиция $Q_E$

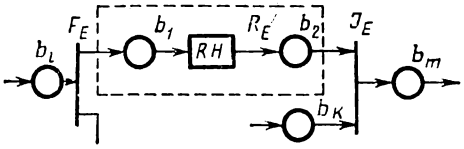



Рис. 4.11. Макропозиция $R_E$

9. В моделях микропроцессорных устройств часто используется динамическая система распределения памяти. Для моделирования такой системы применима макропозиция распределения ресурсов $R_E$ (рис. 4.11). В состав $R_E$ входит элемент *RH*, моделирующий состояние

некоторого ресурса, ограниченного объема. При $M(b_K)=1$ производится проверка соотношения

$$M(b_K(r)) \leqslant |R_E|,$$

где $r$ — номер описателя метки, в котором хранится число, интерпретируемое как требуемое количество ресурса;

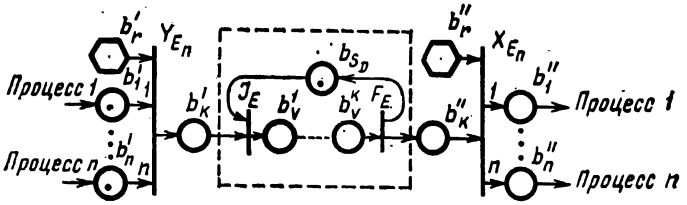



Рис. 4.12. Ограничение доступа к ресурсу в $E$-сети с помощью семафора Дийкстры

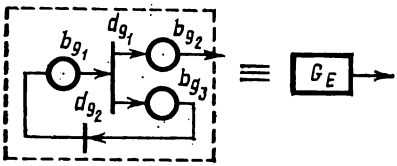


Рис. 4.13. Макропозиция $G_E$

$|R_E|$ — текущий объем ресурса в $R_E$. Если приведенное выше соотношение выполняется, то производится операция $M(b_r) := 1$; $|R_E| := |R_E| - M(b_K(r))$.

Возврат ресурса в макропозицию $R_E$ осуществляется при срабатывании перехода $F_E$ (рис. 4.12). Метка поступает в позицию $b_1$. Далее при условии $M(b_1(r)) > 0$ выполняется $|R_E| := |R_E| + M(b_1(r))$.

**10.** Взаимодействие нескольких процессов при обращении к ограниченному ресурсу удобно отражать в модели с помощью семафоров Дийкстры (§ 3.5). На рис. 4.12 и 4.13 показано использование семафора $S_D$ в $E$-сетевой модели.

### Пример $E$-сетевой модели

Рассмотрим в качестве примера построение $E$-сетевой модели программируемого АП (см. рис. 4.2). Модель построим в предположении $A=\{a_1, a_2\}$, где $a_1$ — ввод сообщения из КС, обработка и вывод на ЭПМ; $a_2$ — ввод сообщения от оператора, его обработка и вывод в КС. Используем для каждой метки в модели три описа-

теля: $M(b_i(1))$ — номер задачи, $M(b_i(2))$ — объем сообщения; $M(b_i(3))$ — временной параметр сообщения. При построении модели положим, что в микропроцессорном АП реализуется принцип фиксированного выделения буферных областей ЗУ для сообщений.

На рис. 4.14 приведен граф $E$-сетевой модели микропроцессорного АП. Поток входных сообщений из $КС_1$ задается генератором $G_{E_1}$, при этом интервалы между сообщениями задаются внешней функцией $\xi_1$, а описа-

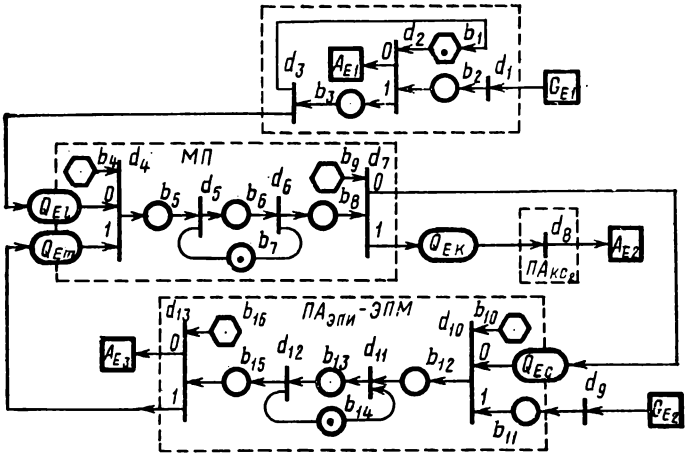



Рис. 4.14. $E$-сетевая модель микропроцессорного АП

тель $M(b_i(2)$ — функцией $\xi_2$. Переход $d_1$ имитирует процесс поступления сообщения на вход $ПА_{КС_i}$. В случае занятости буферного накопителя $(Q_{E_1})$ поступившее сообщение стирается фрагментом $(d_2—A_{E_1})$. Время обработки сообщения $(t(d_2))$ и изменение его объема в $ПА_{КС_i}$ задаются функциями $\xi_3$ и $\xi_4$. Сообщение, поступившее в очередь $Q_{E_1}$, передается на обработку МП через систему приоритетного доступа (переход $d_4$). В модели МП позиция $b_7$ выполняет роль семафора для ограничения доступа. Процедура обработки на МП моделируется переходом $d_6$, при этом $t(d_6)$ задается функцией $\xi_5$, а изменение объема сообщения — функцией $\xi_6$. Переход $d_7$ распределяет потоки задач в зависимости от $M(b_8(1))$ — в очереди $Q_{E_C}$ или $Q_{E_K}$. Сообщения из $Q_{E_C}$ через переход $d_{10}$, отражающий приоритетную выборку, поступают на ЭПМ. Работа ЭПМ моделируется переходом $d_{12}$ и зада-

ется внешними функциями $\xi_7$ и $\xi_8$. Переход $d_{12}$ определяет дальнейший путь сообщения. При $M(b_{15}(1))=1$ обработка метки завершается и она поступает в $A_{E_3}$; если же $M(b_{15}(1))=2$, то метки передаются в очередь к модели МП ($Q_{E_m}$). Поток сообщений от оператора АП моделируется генератором $G_{E_2}$, где интервалы между сообщениями и их длина задаются функциями $\xi_9$ и $\xi_{10}$. Работа $ПА_{КС_3}$ моделируется переходом $d_8$. Его параметр $t(d_8)$ определяется функцией $\xi_{11}$.

Дадим формальное описание рассматриваемой $E$-сетевой модели в форме определения элементов множеств $B$, $B_P$, $B_R$, $D$, $M_0$ и $\xi$.

$$M_0(b_1)=M_0(b_7)=M_0(b_{14})=1;$$

$$B=\{b_1,\ b_2[3],\ b_3[3],\ b_4,\ b_5[3],\ b_6[3],\ b_7,$$
$$b_8[3],\ b_9,\ b_{10},\ b_{11}[3],\ b_{12}[3],\ b_{13}[3],\ b_{14},\ b_{15}[3],\ b_{16},$$
$$Q_{E_l}[3],\ Q_{E_m}[3],\ Q_{E_C}[3],\ Q_{E_K}[3],\ G_{E_1}[3],\ G_{E_2}[3],$$
$$A_{E_1}[3],\ A_{E_2}[3],\ A_{E_3}[3]\};$$

$$B_P=\{\ b_4,\ b_9,\ b_{10},\ b_{16},\ G_{E_1},\ G_{E_2},\ A_{E_1};\ A_{E_2},\ A_{E_3}\};$$

$$B_R=\{b_1,\ b_4,\ b_9,\ b_{10},\ b_{16}\};$$

$$D=\{d_1,\ d_2,\ \ldots\ ,\ d_{13}\};$$

$$d_1=(T_E(G_{E_1},\ b_2);\ 0;\ -);$$

$$d_2=(X_E(b_1,\ b_2,\ A_{E_1},\ b_3);\ 0;-);$$

$$d_3=(F_E(b_3,\ b_1,\ Q_{E_l});\ \xi_3(M(b_3(2)));$$

$$M(Q_{E_l}(2)):=\xi_4(M(b_3(2));$$

$$d_4=(Y_E(b_4,\ Q_{E_l},\ Q_{E_m},\ b_5);\ 0;\ -);$$

$$d_5=(J_E(b_5,\ b_7,\ b_6);\ 0;\ -);$$

$$d_6=(F_E(b_6,\ b_8,\ b_7);\ M(b_6(1))=1\rightarrow\xi_5'(M(b_6(2))),$$
$$M(b_6(1))=2\rightarrow\xi_5''(M(b_6(2)));\ (M(b_6(2)));\ M(b_6(1))=1\rightarrow$$
$$M(b_8(2)):=\xi_6'(M(b_6(2)));\ M(b_6(1))=2\rightarrow$$
$$M(b_8(2)):=\xi_6''(M(b_6(2)));$$

$$d_7=(X_E(b_9,\ b_8,\ Q_{E_C},\ Q_{E_K});\ 0;\ -);$$

$$d_8=(T_E(Q_{E_K},\ A_{E_2}),\ \xi_{11}(M(Q_{E_K}(2)));$$

$$M(A_{E_2}(2)):=\xi_{12}(M(Q_{E_K}(2)));$$

$$d_9=(T_E(G_{E_2},\ b_{11});\ 0;\ -);$$

$$d_{10} = (Y_E(b_{10},\ Q_{E_C},\ b_{11},\ b_{12});\quad 0;\ -);$$

$$d_{11} = (J_E(b_{12},\ b_{14},\ b_{13});\quad 0;\ -);$$

$$d_{12} = (F_E(b_{13},\ b_{15},\ b_{14});\quad M(b_{13}(1)) = 1 \rightarrow$$

$$\xi_7'(M(b_{13}(2)));\quad M(b_{13}(1)) = 2 \rightarrow \xi_7''(M(b_{13}(2)));$$

$$M(b_{13}(1)) = 1 \rightarrow M(b_{15}(2)) := \xi_8(M(b_{13}(2)));$$

$$d_{13} = (X_E(b_{15},\ A_{E_3},\ Q_{E_m});\quad 0;\ -);$$

$$r(b_4) = [M(b_8) = 0 \rightarrow M(b_4) := 0];$$

$$r(b_9) = [M(b_8(1)) = 1 \rightarrow M(b_9) := 0;$$

$$M(b_8(1)) = 2 \rightarrow M(b_9) := 1];$$

$$r(b_{10}) = [M(b_{15}(1)) = 0 \rightarrow M(b_{10}) := 0];$$

$$r(b_{16}) = [M(b_{15}(1)) = 1 \rightarrow M(b_{16}) := 0;$$

$$M(b_{15}(1)) = 2 \rightarrow M(b_{16}) := 1];$$

$$\xi = \{\xi_1,\ \xi_2,\ \xi_3,\ \xi_4,\ \xi_5',\ \xi_5'',\ \xi_6',\ \xi_6'',\ \xi_7',\ \xi_7'',\ \xi_8,\ \xi_9,\ \xi_{10},\ \xi_{11},\ \xi_{12}\}.$$

Для полного задания модели необходимо детализировать макропозиции $G_{E_1}$ и $G_{E_2}$. Для $G_{E_1}$ (см. рис. 4.13) формальное задание следующее:

$$M_0(b_{g_1}) = 1;$$

$$B_P = \{b_{g_2}\};\quad B_R = \varnothing;$$

$$d_{g_1} = (F_E(b_{g_1},\ b_{g_2},\ b_{g_3});\quad 0;\quad -);$$

$$d_{g_2} = (T_E(b_{g_3},\ b_{g_1});\quad \xi_1,\ M(b_{g_1}(1)) := 1;$$

$$M(b_{g_1}(2)) := \xi_2;\quad M(b_{g_1}(3)) := t_k;$$

$$\xi = \{\xi_1,\ \xi_2,\ t_k\},$$

где $t_k$ — текущее время.

Задание $G_{E_2}$ аналогично, за исключением $M(b_{g_1}(1)) := 2$; а в качестве элементов $\{\xi_i\}$ выступают $\xi_{11}$ и $\xi_{12}$.

### Переход от $E$-сети к имитационной программе

Структурное задание модели микропроцессорного устройства СПИ в форме $E$-сети позволяет использовать модульный принцип разработки имитационной модели с применением библиотеки $E$-сетевых модулей и их параметрической настройки. В этом случае $E$-сетевая модель устройства является основой для сборки имитационной программы из модулей, реализованных на некотором языке программирования.

Базой для построения программных модулей служит обобщенный алгоритм функционирования $E$-сетевого

модуля. Поступление метки во входную позицию $b_i$-перехода $d_m$ (рис. 4.15) инициирует процедуру проверки готовности перехода к срабатыванию (рис. 4.16). Проверяемые при этом условия включают в себя анализ разметки решающей позиции $M\{b_k\}$, где $b_k \in B_R$, а также проверку разметки выходных позиций $M\{H(d_m)\}$ и сопряженных входных позиций $M\{\Phi(d_m)\}$. В том случае, когда $b_k \in B_P$, на данной фазе производится также вычисление решающей процедуры перехода $r(d_m)$.

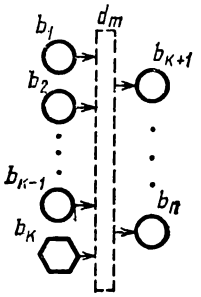



Рис. 4.15. Обобщенная структура $E$-сетевого перехода

При достижении требуемых условий переход вступает в активную фазу, содержание которой определяется элементами $t(d_m)$ и $\rho(d_m)$ описания перехода. Элемент $t(d_m)$ может быть задан непосредственно либо в виде функции. Если элемент $t(d_m)$ задан в виде функции, то требуется предварительное вычисление ее. Затем моделируется требуемое время $t(d_m)$. Процедура перехода $\rho(d_m)$ выполняется в два этапа. Сначала вычисляются предикаты $\Pi_j$, затем реализуется требуемое подмножество операций над описателями меток $\{l_{jp}\}$. Далее переход $d_m$ вступает в фазу завершения, содержа-

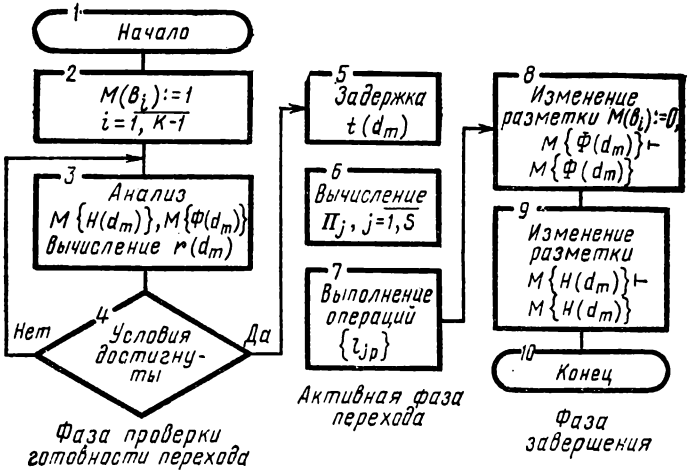



Рис. 4.16. Обобщенный алгоритм работы $E$-сетевого перехода

нием которой является смена разметки входных и выходных позиций перехода $M\{\Phi(d_m)\}$ и $M\{H(d_m)\}$. Разметка позиции $b_i$ меняется на нулевую $M(b_i) := 0$. При $b_i \in B_P$ разметка решающей позиции становится неопределенной $M(b_k) := e^0$. Затем выполняется перемещение меток в ряд выходных позиций $H(d_m)$ в соответствии с типом данного перехода. Эта операция завершает работу перехода по обработке поступившей метки.

Одним из важных вопросов, встающим перед разработчиком имитационной модели, является выбор языка программирования. Реализация $E$-сетевых модулей на машинно-ориентированном языке ассемблер или же языках высокого уровня АЛГОЛ, ФОРТРАН, $PL/1$ позволяет снизить затраты машинного времени и памяти при моделировании, однако характеризуется высокой трудоемкостью разработки библиотеки моделирующих программ. Трудоемкость разработки может быть значительно снижена при использовании специализированных языков, ориентированных на моделирование.

## Переход к модулям на специализированном языке моделирования

Рассмотрим этот вопрос на примере анализа соответствий между элементами $E$-сетей и объектами одного из наиболее распространенных языков моделирования GPSS. Анализ показывает следующее:

1. Метка $E$-сети может быть представлена динамическим объектом, называемым в GPSS *транзактом.*

2. Описатели меток аналогичны параметрам транзактов.

3. Позиция $E$-сети идентична аппаратно-ориентированному объекту типа накопителя единичной емкости.

4. Решающие позиции $b_k \in B_R$ $E$-сети в зависимости от принадлежности к множеству $B_P$ реализуются двумя способами: а) если $b_k \in B_P$, то $b_k$ эквивалентна набору вычисляемых объектов GPSS, называемых булевыми переменными; б) если $b_k \in B/B_P$, то $b_k$ может быть представлена накопителем единичной емкости;

5. Временные параметры переходов $t(d_m)$ реализуются операционными объектами GPSS-блоками ADVANCE.

6. Операции вычисления $\Pi_j$ $(j=\overline{1,s})$ для $\rho(d_m)$ соответствуют применению блоков TEST E в сочетании с булевыми переменными.

7. Операции $\{l_{ip}\}$ для $\rho(d_m)$ могут быть выполнены с помощью блоков ASSIGN в сочетании с арифметическими переменными.

8. Хранение значений описателей меток можно имитировать в GPSS записью значений параметров транзактов в ячейки хранимых значений (X, XH) с помощью блоков SAVEVALUE.

9. Процессы синхронизации движения меток через переход $d_m$ и удаления меток из решающей позиции $b_k \in B_k$ могут быть обеспечены с помощью логических переключателей GPSS в сочетании с блоками LOGIC R, LOGIC S.

10. Макропозиция генератора $G_E$ аналогична транзактно-ориентированному блоку GENERATE.

11. Макропозиция поглощения $A_E$ функционально идентична блоку TERMINATE.

12. Макропозиция очереди типа FIFO $(Q_E)$ может интерпретироваться в GPSS двумя способами: а) с помощью накопителя емкости $N$, где $N$ — длина очереди; б) записью транзакта в цепь пользователя (этот способ предпочтительнее с точки зрения затрат времени на моделирование).

Наиболее сложным способом, встающим при реализации на языке GPSS E-сетевых элементов, является разрешение временных узлов в модели, возникающих в случае, когда два или более события запланированы на одно и то же значение модельного времени. В языке GPSS процедура моделирования реализована так, что каждый транзакт продвигается по модели до тех пор, пока он либо не окажется задержанным из-за несоблюдения некоторого условия, либо не войдет в блок ADVANCE. Только после этого производится переход к обработке следующего транзакта из цепи текущих событий. $E$-сеть же предполагает параллельность обработки всех находящихся в данный момент в активном состоянии меток, поэтому для перехода с $|\Phi(d_m)| \geqslant 2$ при реализации на GPSS необходима дополнительная синхронизация продвижения транзактов. Требуется обработать все события, связанные с метками, находящимися в $\Phi(d_m)$ на фазе проверки готовности перехода, а уже после этого разрешать переход в активную фазу. Средством такой синхронизации в GPSS может служить блок PRIORITY 0, BUFFER.

Таким образом, структурированное $E$-сетевое представление моделей микропроцессорных устройств СПИ

позволяет упростить и автоматизировать этап определения базовой структуры устройства с использованием имитационного моделирования. $E$-сетевое задание модели допускает достаточно простую программную реализацию имитационной программы на одном из языков программирования высокого уровня или специализированных языков моделирования.

## § 4.5. ВЫДЕЛЕНИЕ ФУНКЦИОНАЛЬНОГО ПОДМНОЖЕСТВА, СВЯЗАННОГО С ПЕРЕДАЧЕЙ ИНФОРМАЦИИ ПО КАНАЛАМ СВЯЗИ

Одним из важных шагов системного этапа проектирования (см. рис. 3.2) является выделение функциональной структуры устройства. Функциональная структура микропроцессорного устройства СПИ (см. гл. 2) может быть представлена в виде объединения ряда подмножеств функций $F = F_N \cup F_T \cup F_O \cup F_C \cup F_S$, из которых наиболее многочисленным по составу и критичным к режиму реального времени является подмножество сетевых функций $F_N$, связанных с реализацией протоколов ИВС.

Современные ИВС с коммутацией пакетов характеризуются большим разнообразием микропроцессорных устройств. С функциональной точки зрения эти устройства выступают в качестве хостмашин сети, коммуникационных, интерфейсных, терминально-интерфейсных и терминально-коммуникационных машин, а также терминалов ИВС. В зависимости от назначения на данные устройства возлагаются задачи по реализации ряда уровней сетевых протоколов. В § 1.3 были уже приведены основные особенности организации взаимодействия прикладных процесов пользователей в ИВС в соответствии с семиуровневой архитектурой связи открытых систем. Микропроцессорные устройства реализуют некоторое множество функций, определяемых протоколами 2...4-го уровней, т. е. транспортного, сетевого уровня и уровня управления информационным каналом. Приведем основные функции указанных выше протоколов и остановимся более подробно на функциональном составе протокола управления информационным каналом, как элементе, реализация которого требуется практически во всех устройствах ИВС.

Основными в программной структуре ИВС являются программы пользователей и взаимосвязанные с ними программы управления представлением и сеансом. Совокуп-

ность этих трех элементов принято называть *процессом* (рис. 4.17). Правила, определяющие взаимодействие программ пользователей, управления представлением и сеансом задаются протоколами 7, 6 и 5-го уровней. Основной задачей ИВС является обеспечение взаимодействия удаленных процессов, которое осуществляется обменом между процессами массивами информации (сообщениями). Сообщение имеет произвольную длину. Поэтому его обычно перед передачей делят на части — *блоки* (рис. 4.17), максимальный размер каждого из которых огра-

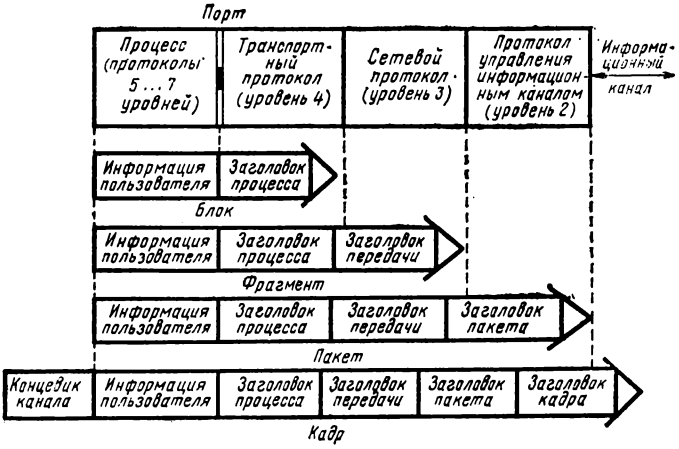



Рис. 4.17. Основные сообщения, обрабатываемые протоколами 2—4 уровней и структура этих сообщений

ничен определенной величиной. Каждый такой блок снабжается служебной информацией — *заголовком процесса.*

Ответственность за правильную и своевременную передачу блоков между хостмашинами ИВС несут протоколы транспортного, сетевого уровней и уровня управления информационным каналом. Каждый из этих уровней, как показано на рис. 4.17, обрамляет блок служебной информации. Добавление транспортным протоколом своего заголовка межконцевой передачи превращает блок во *фрагмент,* передаваемый сетевому уровню. Сетевой уровень в свою очередь снабжает фрагмент заголовком, образуя *пакет информации.* На 2-м уровне (управление информационным каналом) пакеты упаковываются в

*кадры,* содержащие информацию, необходимую для надежной передачи по информационному каналу. Именно в таком виде, с большим количеством служебных полей, кадры направляются в информационный канал.

Протоколы транспортного уровня обеспечивают передачу информации между портами транспортной сети (от одного порта до другого). Основными функциями транспортного протокола являются: установление и разъединение транспортных соединений; обеспечение адресации транспортных объектов; управление работой сетевого уровня через сетевой интерфейс; преобразование транспортных соединений в сетевые; управление потоками информации на всем пути от порта до порта; обнаружение и исправление ошибок передачи; восстановление передачи после появления неисправностей и ошибок; контроль последовательности передачи массивов информации на всем протяжении от одного порта до другого; сегментация и сборка массивов информации, передаваемых через транспортную сеть от порта до порта.

Протоколы сетевого уровня выполняют аналогичные транспортным протоколам функции, но по отношению к сетевому соединению объектов сети. Между двумя транспортными объектами возможно использование нескольких сетевых соединений. Это могут быть виртуальные цепи, выполненные в соответствии с рекомендацией *X*.25/3, дейтаграммные цепи, прямые каналы между портами.

Сетевое соединение между транспортными объектами организуется либо как постоянное, либо является временным. Оно может быть типа «точка — точка» или же «точка — многоточка». Во втором случае обеспечивается связь одного транспортного объекта с несколькими другими объектами.

Важнейшей функцией сетевого уровня является маршрутизация, обеспечивающая коммутацию потоков данных. На протоколы 3-го уровня возлагаются также задачи мультиплексирования нескольких сетевых соединений в одном информационном канале. Часто в число функций сетевого протокола включаются сегментация и сборка фрагментов сообщений, если размер пакета данных меньше, чем пакета сетевого уровня.

Задачи протокола управления информационным каналом рассмотрим на примере конкретного, наиболее широко распространенного протокола *X*.25/2 (HDLC), рекомендованного к применению МККТТ.

Протокол HDLC МККТТ определяет значительное число функций управления информационным каналом, к которым в первую очередь относятся: управление процедурами инициализации и деинициализации информационного канала; упаковка передаваемой информации в кадры перед передачей и ее разупаковка после передачи по физическому каналу; генерация и чтение управляющих кадров; обеспечение прозрачности информационного канала; передача и прием подтверждений о приеме кадров; кодирование и декодирование содержимого кадров с целью обнаружения ошибок при передаче; повторная передача тех кадров, которые оказались потерянными, либо в которых обнаружена ошибка; управление потоками кадров в обе стороны физического канала.

Рассмотрим кратко особенности протокола HDLC применительно к асинхронному несбалансированному режиму (ARM, LAP). Полное описание процедур протокола содержится в [39].

Передача информации, команд и ответов по каналу производится кадрами. Выделение их из потока битов, идущих по каналу, происходит с помощью введения специальной восьмибитовой комбинации — метки (код 01111110). Каждый кадр начинается и заканчивается комбинацией метки (*F*). Конечная метка одного кадра может являться одновременно начальной меткой следующего. Внутри кадра выделяются определенные поля по позиционному признаку. Это поля адреса (*A*), управления (*C*), информационное (*I*) и проверочной последовательности кадра (FCS). Применение специальной комбинации *F*, содержащей 6 подряд идущих единиц, потребовало обеспечения кодовой прозрачности внутри кадра, которое достигается за счет процедуры битстаффинга — введения при передаче и последующего удаления при приеме нулевых битов после любой комбинации внутри кадра, содержащей 5 единичных подряд идущих битов.

По функциональному назначению в протоколе HDLC различаются кадры трех типов: информационные (I-типа); супервизорные (S-типа); ненумерованные U-типа).

*Кадры I-типа* используются для передачи информации (пакетов). Длина информационного поля кадра по протоколу HDLC не ограничена (рис. 4.18).

*Кадры типа* S обеспечивают выполнение функций управления передачей информационных кадров. К ним относятся: RR — «готов к приему»; RNR — «не готов к при-

ему»; REJ — «отказ от кадров». Структура кадров S-типа показана на рис. 4.19.

*Ненумерованные кадры* (U-тип) используются для обеспечения дополнительных функций управления. В протоколе применяются два кадра-команды этого типа: SARM — «установить связь в асинхронном несбалансированном режиме»; DISC — «прекратить связь».

Кроме того, ненумерованными являются два кадра-ответа: UA — «ненумерованное подтверждение»; CMDR — «отказ от выполнения команды».

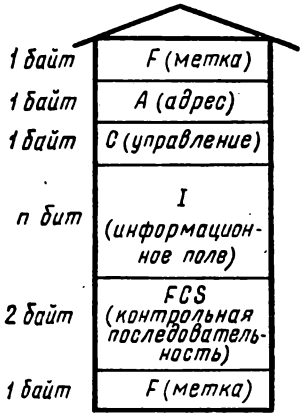



Рис. 4.18. Структура информационного кадра

Формат кадров SARM, DISC и UA не содержит информационного поля (рис. 4.19). Кадры CMDR включают 3-байтовую информацион-

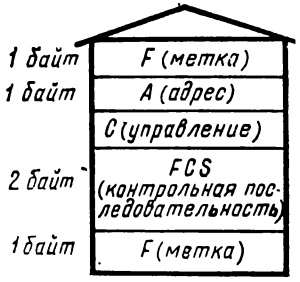



Рис. 4.19. Структура кадров RR, RNR, REJ, SARM, DISC, UA

ную область, дающую возможность приемной станции сообщить передающей о приеме кадра, который не может быть воспринят или противоречит правилам протокола (рис. 4.20).

Протокол используется в режиме работы «точка — точка». Для такого двухточечного соединения поле А кадра может содержать только две кодовые комбинации: [A] — 11000000; [B] — 10000000.

При кодировании поля управления С кадра (рис. 4.21), непосредственно перед отправкой информационного кадра в информационный канал в него записывается *номер передаваемого кадра* N(S). Кадры типов I и S содержат также *номер ожидаемого* (при приеме) *кадра* N(S). Разряд 5 в управляющем поле всех типов кадров используется для записи бита запроса — ответа P/F : P

записывается в кадре, содержащем команду, а $F$ — в кадре, являющемся ответом на команду. Бит P/F используется в протоколе для выполнения функций вызова ненумерованного или супервизорного ответа или же как указание об ошибке.

На рис. 4.22, *а* и *б* показано кодирование полей управления в кадрах S- и U-типов.

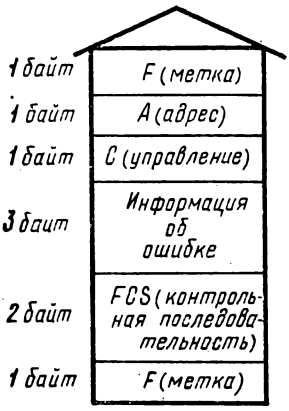



Рис. 4.20. Структура кадра CMDR

В соответствии с протоколом две станции, определяемые как DCE (аппаратура передачи данных) и DTE (оконечное оборудование данных) с адресами [A] и [B] для образования дуплексного соединения устанавливают два полудуплексных канала. Каждая из станций содержит в своем составе первичную и вторичную станции (рис. 4.23).

Функцией *первичной станции* является передача информационных кадров I, кадров SARM и DISC, и прием ответов от вторичной станции-адресата


Рис. 4.21. Кодирование поля управления кадра

в форме кадров-ответов RR, RNR, REJ, UA и CMDR. Первичная станция присваивает посылаемым информационным кадрам порядковые номера N(S). Нумерация ведется по модулю 8. Станция хранит порядковый номер следующего посылаемого кадра I в форме переменной состояния передачи V(S). На *вторичной станции* производится подсчет количества правильно принятых кадров

I, для чего используется переменная состояния приема V(R). Подсчет кадров ведется по модулю 8. В кадрах-ответах вторичная станция сообщает первичной с помощью поля N(R) порядковый номер следующего ожидаемого на приеме кадра I. В случае дуплексного обмена номер N(R) может передаваться не только в кадрах-ответах, но и в области управления информационных кадров I, передаваемых в обратном направлении. В поле адреса кадров согласно протоколу HDLC указывается всегда адрес вторичной станции (рис. 4.23), независимо от того, является она получателем или отправителем информации.

а)

| Кадр | S-биты | |
|---|---|---|
| | 3 | 4 |
| RR | 0 | 0 |
| RNR | 0 | 1 |
| REJ | 1 | 0 |

б)

| Кадр | M-биты | | | | |
|---|---|---|---|---|---|
| | 3 | 4 | 6 | 7 | 8 |
| SARM | 1 | 1 | 0 | 0 | 0 |
| DISC | 0 | 0 | 0 | 1 | 0 |
| UA | 0 | 0 | 1 | 1 | 0 |
| CMDR | 1 | 0 | 0 | 0 | 1 |

Рис. 4.22. Кодирование полей управления в кадрах *S*-типа (*а*) и *U*-типа (*б*)

Более подробное описание процедур данного протокола будет дано ниже при рассмотрении способов его формализации.

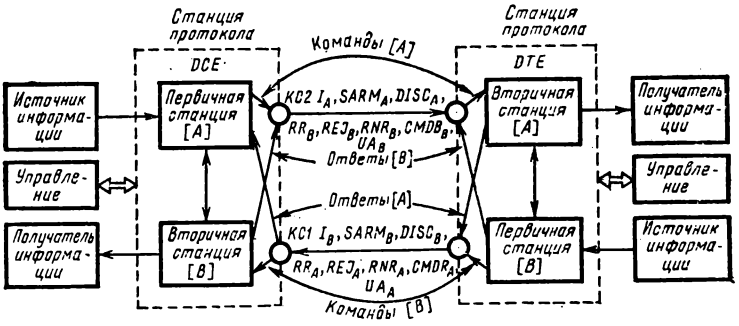



Рис. 4.23. Организация двухточечного дуплексного канала по протоколу *HDLC* МККТТ

## § 4.6. ФОРМАЛЬНЫЕ СПОСОБЫ ОПИСАНИЯ ПРОЦЕДУР ПРОТОКОЛА ИНФОРМАЦИОННОГО КАНАЛА

Словесное описание процедур протокола HDLC является неформализованным, что делает возможным неоднозначную трактовку процедур протокола, затрудняет процессы разработки ПО и моделирования протокола. При разнотипном характере процедур протокола невозможно

использовать единый математический аппарат формализации. В связи с этим попытаемся построить иерархическую модель протокола с несколькими уровнями детализации процедур.

### Модель на базе метода диаграмм состояний

На верхнем уровне описания протокола целесообразно выделить основные фазы передачи информации и их взаимосвязи. Для этой цели удобна формализация с использованием метода диаграмм состояний, наиболее распространенного и наглядного средства укрупненного моделирования протоколов ИВС.

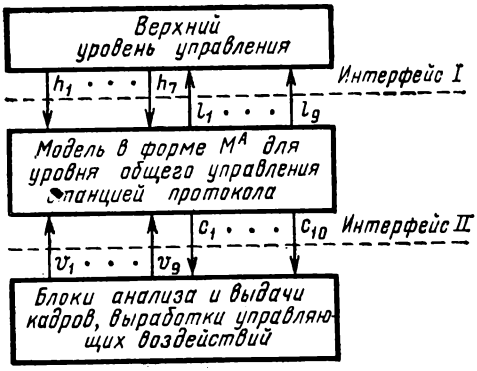



Рис. 4.24. Связь модели в форме $M^A$ с другими уровнями модели протокола

Рассмотрим уровень общего управления станцией протокола HDLC (рис. 4.24). Используем модель в форме последовательной машины (автомата) $M^A$, которая задается упорядоченной пятеркой $M^A = (Q^A, I^A, W^A, \tau^A, \omega^A)$, где $Q^A$ — конечное множество внутренних состояний; $I^A$ — конечное множество входов; $W^A$ — конечное множество выходов; $\tau^A$ — функция переходов, представляющая собой отображение подмножества $D_\tau^A \subseteq Q^A \times I^A$ на подмножество $Q^A$; $\omega^A$ — функция выходов, представляющая собой отображение подмножества $D_\omega^A \subseteq Q^A \times I^A$ на $W^A$.

Применение теории автоматов позволяет получить диаграмму состояний управления станцией протокола

HDLC (рис. 4.25), на которой выделены следующие состояния: $D_1$ — станция отключена от $КС_1$ и $КС_2$; $D_2$ — состояние готовности к соединению (ведется выдача флаговых комбинаций в $КС_2$ и анализируются сигналы, приходящие из $КС_1$); $D_3$ — состояние установления информационного канала; $D_4$ — состояние обмена информацией; $D_5$ — состояние разъединения информационного кана-

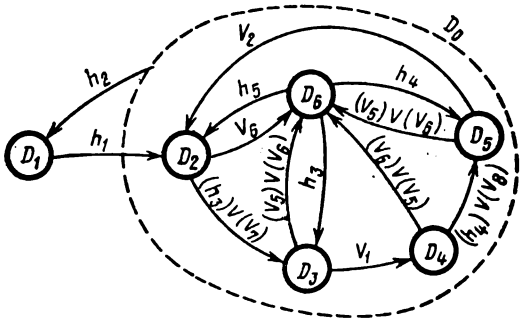



Рис. 4.25. Диаграмма состояний станции HDLC

ла; $D_6$ — состояние ожидания внешнего восстановления (от верхнего уровня управления).

Состояние $D_0$ объединяет те состояния, в которых станция подключена к каналам связи.

Сигналы интерфейса I и II (рис. 4.24 и 4.25) следующие:

$h_1$ — подключить станцию к $КС_1$ и $КС_2$, $h_2$ — отключить станцию от $КС_1$ и $КС_2$, $h_3$ — инициировать информационный канал (ИК), $h_4$ — деинициировать ИК, $h_5$ — перейти в состояние готовности к соединению, $h_6$ — установить состояние «занято», $h_7$ — снять состояние «занято»;

$l_1$ — ИК инициирован, $l_2$ — ИК деинициирован, $l_3$ — обнаружена ошибка в посланной команде, $l_4$ — обнаружена ошибка в принятом ответе, $l_5$ — требуется восстановление из-за $N2$ повторов, $l_6$ — требуется восстановление из-за отказа КС, $l_7$ — поступил запрос на инициализацию ИК, $l_8$ — поступил запрос на деинициализацию ИК, $l_9$ — получен кадр $I$;

$v_1$ — первичная станция (ПС) инициализировала ИК, $v_2$ — ПС деинициализировала ИК, $v_3$ — ПС обнаружила ошибку в посланной команде, $v_4$ — ПС обнаружила ошиб-

ку в принятом ответе, $v_5$ — на ПС не получено подтверждения после $N2$ попыток, $v_6$ — получен сигнал об отказе КС, $v_7$ — вторичной станцией (ВС) получен запрос на инициализацию ИК, $v_8$ — ВС получен запрос на деинициализацию ИК, $v_9$ — ВС получен информационный кадр;

$c_1$ — команда ПС на прекращение анализа кадров, $c_2$ — команда ПС на инициализацию ИК, $c_3$ — команда ПС на деинициализацию ИК, $c_4$ — команда ВС на прекращение анализа кадров; $c_5$ — команда ВС перейти к дежурному приему, $c_6$ — команда ВС разрешить прием

| Состояние | Входные сигналы | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $h_5$ | $h_6$ | $h_7$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ | $v_7$ | $v_8$ | $v_9$ |
| $D_4$ | $D_4/-$ | $D_1/C_4, C_{10}$ | $D_4/-$ | $D_5/C_3, C_4$ | $D_4/-$ | $D_4/C_7$ | $D_4/C_8$ | $D_4/-$ | $D_4/-$ | $D_4/l_3$ | $D_4/l_4$ | $D_6/C_4, l_5$ | $D_6/C_4, C_4, l_6$ | $D_4/-$ | $D_5/C_3, l_8$ | $D_4/l_9$ |

Рис. 4.26. Фрагмент таблицы переходов для станции протокола HDLC

данных, $c_7$ — команда ВС перейти в состояние «занято», $c_8$ — команда ВС снять состояние «занято»; $c_9$ — разрешение выдачи флаговых комбинаций в $КС_2$, $c_{10}$ — запрет выдачи флаговых комбинаций в $КС_2$.

Графическое задание в виде диаграммы состояний наглядно отражает изменение режимов функционирования станции при поступлении тех или иных входных сигналов через интерфейсы I и II. Вместе с тем недостаток диаграммы состоит в том, что она не отражает выходных сигналов через интерфейсы I и II. По этой причине удобно воспользоваться табличной формой задания автомата Мили — таблицей переходов. Из фрагмента таблицы переходов, отражающего реакцию станции, находящейся в состоянии $D_4$, на входные сигналы (рис. 4.26) следует, что при нахождении станции протокола в состоянии $D_4$ и поступлении входного сигнала $h_2$ происходит переход станции в состояние $D_1$ с одновременной выдачей сигналов $c_1$, $c_4$ и $c_{10}$. Поступление же сигналов $h_1$, $h_3$, $h_5$, $v_1$, $v_2$ или $v_7$ не изменяет состояния станции и не вызывает выдачу ответных сигналов.

Построенная диаграмма состояний станции протокола является слишком обобщенной, она не отражает в полной мере работу первичной и вторичной станций. Проведем ее детализацию в виде диаграмм ПС и ВС.

Будем рассматривать состояния ПС и ВС как подсостояния общей диаграммы состояний станции протокола, обозначая их соответственно как $\{D'_i\}$ и $\{D''_i\}$. Как показывает анализ, состояния $D'_1$, $D'_2$ и $D'_6$ характеризуются пассивным поведением ПС, поэтому объединим данные элементы в одно состояние, обозначив его как $D'_p$ (рис. 4.27). Состояние $D'_4$ требует на уровне ПС детали-

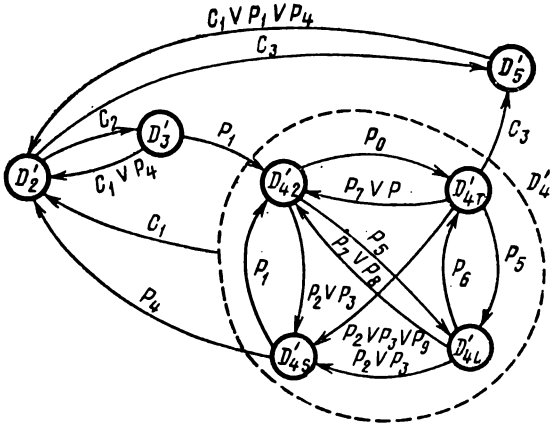



Рис. 4.27. Диаграмма состояний ПС протокола HDLC

зации на четыре подсостояния: передачи данных по КС ($D'_{4R}$), восстановления по истечении тайм-аута ожидания подтверждения ($D'_{4T}$), повторного установления соединения ($D'_{4S}$) и запрета передачи данных ($D'_{4L}$).

На рис. 4.27 используются следующие сигналы $\{p_i\}$: $p_1$ — поступил кадр $UA$ с битом $F=0$; $p_2$ — поступил кадр CMDR с битом $F=0$; $p_3$ — принят ошибочный кадр ответа; $p_4$ — соединение не установлено после $N2$ попыток; $p_5$ — поступил кадр RNR; $p_6$ — истек тайм-аут ожидания подтверждения; $p_7$ — поступил кадр REJ; $p_8$ — принят кадр RR; $p_9$ — не получено подтверждение после $N2$ повторов информационного кадра.

Аналогичное рассмотрение множества $\{D''_i\}$ показывает (см. рис. 4.25), что состояния $D''_1$, $D''_3$, $D''_5$ и $D''_6$ характеризуются отсутствием анализа кадров, приходящих с адресом [B], это позволяет объединить их в одно со-

стояние $D''_P$. Элемент же $D''_4$ для более детального представления работы ВС требуется разбить на четыре подсостояния — приема данных по каналу ($D''_{4K}$), восстановления посредством REJ($D''_{J4}$), неприема команды ($D''_{4E}$) и состояние «занято» ($D''_{4H}$). На рис. 4.28 используются следующие элементы $\{q_i\}$: $q_1$ — получен кадр I с номером $N(S) \neq V(R)$; $q_2$ — принят недействительный кадр; $q_3$ — получен кадр DISC; $q_4$ — получен кадр SARM; $q_5$ — получен кадр I с номером $N(S) = V(R)$.

Диаграммы состояний ПС и ВС могут быть представлены в форме *таблиц переходов*. Таким образом, верхний уровень модели протокола (рис. 4.29) представляет-

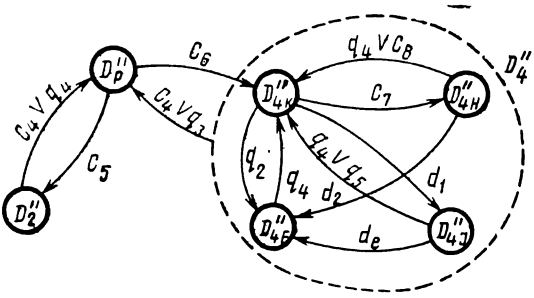



Рис. 4.28. Диаграмма состояний ВС протокола HDLC

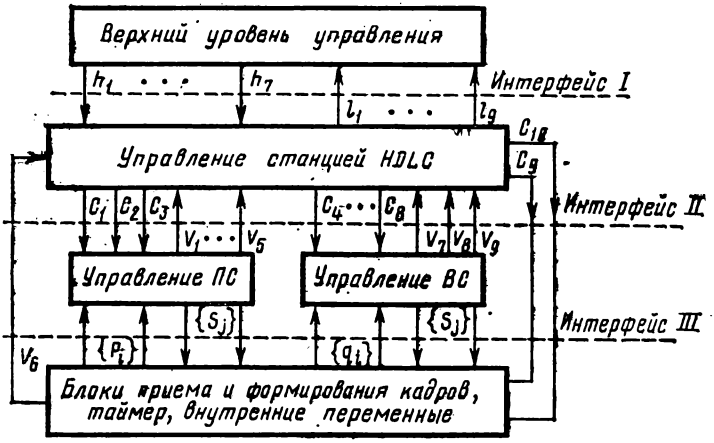



Рис. 4.29. Иерархическое представление процесса управления станцией протокола

ся в виде взаимосвязанных автоматов $M_i^A$. Задание этих автоматов в виде таблиц переходов позволяет упростить реализацию ПО протокола HDLC. Более детальное описание самих протокольных процедур и их последовательности требуют привлечения других методов.

### Описание последовательности процедур протокола сетями Мерлина

Диаграмма состояний не позволяет описать последовательность процедур протокола HDLC и их взаимосвязи. Удобным аппаратом для описания потоков асинхронных параллельных событий (что характерно для процедур обработки кадров в протоколе) являются сети

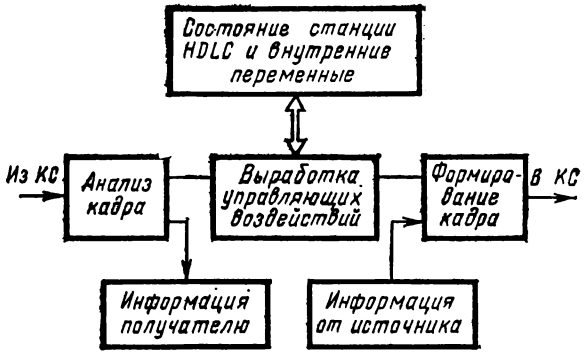



Рис. 4.30. Блоки обработки кадров станции протокола HDLC

Петри (см. гл. 3). Так как при рассмотрении процедур протокола необходим учет фактора времени, воспользуемся классом сетей Мерлина. Наличие в сетях Мерлина множеств $\Lambda^*$ (минимальных времен срабатывания переходов) и $\Lambda^{**}$ (максимальных времен срабатывания переходов) дает возможность оценивать временные характеристики для функциональных последовательностей.

На рис. 4.30 приведена обобщенная структура станции протокола с точки зрения последовательности выполняемых процедур.

Анализ показывает, что сети Мерлина хорошо отражают процедуры блоков анализа и формирования кадров. Так, например, на рис. 4.31 приведен граф сети Мерлина для блока анализа принятого из КС кадра. Процесс анализа делится на два этапа — предварительный и де-

тальный. На первом модуле производится проверка длины кадра и его декодирование; второй модуль осуществляет анализ адресного поля, типа кадра, его длины, бита P/F, номеров $N(S)$ и $N(R)$. Позиции $bs_1$ и $bs_2$ моделируют семафорные механизмы, ограничивающие доступ к внутренним переходам и позициям модели.

Данная модель построена с предположением двух обрабатывающих устройств (например, ПА и МП). Как видно из модели, конвейерный способ обработки может повысить общую пропускную способность блока. Для обработки информационных кадров (типа I) имеем:

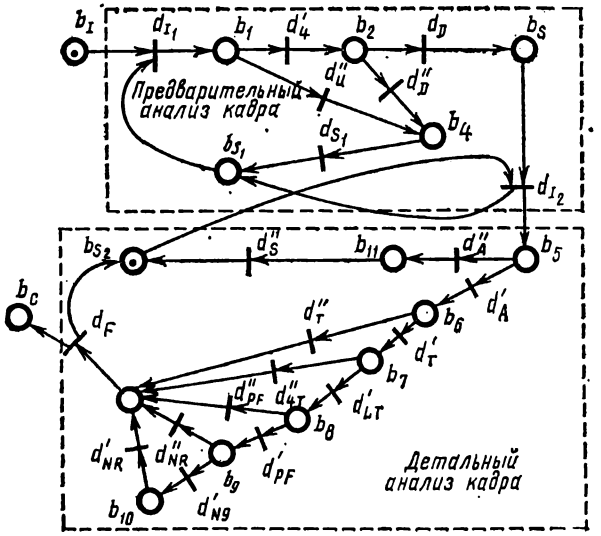



Рис. 4.31. Модель блока анализа кадров:
$d_{I_1}$ — ввод кадра в блок предварительного анализа; $d'_L$, $d''_L$ — предварительный анализ длины кадра ($L1 \geqslant 4$ байт); $d'_D$, $D''_D$ — декодирование; $d_{I_2}$ — передача кадра на детальный анализ; $d'_A$, $d''_A$ — анализ адресного поля; $d'_T$, $d''_T$ — анализ типа кадра; $d'_{LT}$, $d''_{LT}$ — анализ длины кадра; $d'_{PF}$, $d''_{PF}$ — выделение бита $P/F$; $d'_{NS}$ — проверка номера $N(S)$ для кадров типа I; $d'_{NR}$ — проверка номера $N(R)$ для кадров типа I; $d''_{NR}$ — анализ номера $N(R)$ для кадров RR, RNR, REJ; $d'_S$, $d''_S$ — стирание кадра; $d_F$ — передача информации об анализе в блок выработки управляющих воздействий, запись в буферную память кадра типа I

а) для блока предварительного анализа

$$t'_{I_{\max}} = t^{**}_{d'_L} + t^{**}_{d'_D};$$

б) для блока детального анализа

$$t''_{I_{\max}} = t^{**}_{d_{A'}} + t^{**}_{d''_T} + t^{**}_{d_{L'T}} + t^{**}_{d''_{PF}} + t^{**}_{d''_{NS}} + t^{**}_{d''_{NR}} + t^{**}_{d_F}.$$

Максимальная допустимая скорость поступления на вход кадров I

$$V_{I_{\max}} = \frac{1}{t_{I_{\max}}}, \text{ где } t_{I_{\max}} = \max\{t'_{I_{\max}}, t''_{I_{\max}}\}.$$

Полезным свойством модели такого типа является возможность отображения на ней параллельности выполнения ряда процедур анализа. Так, например, сеть на рис. 4.31 может быть модифицирована с целью параллельного анализа номеров N(S) и N(R) для кадров типа I.

На сетевых моделях Мерлина можно наглядно отразить распределение процедур анализа на ряд обрабатывающих блоков МП устройства СПИ и получить оценки для граничных значений времен обработки. Кроме того, сама сеть является укрупненным заданием алгоритма обработки кадра.

### Использование языков сетей Петри для формализации процедур выработки управляющих воздействий

В множестве процедур, определяемых протоколом HDLC, наиболее сложным по логике реализации является подмножество процедур управления процессом обмена, которое реализуется на станции HDLC в блоке выработки управляющих воздействий (БВУВ) (см. рис. 4.30). Формализация представления этого подмножества процедур и простота реализации данного блока являются важной задачей при построении МП устройства СПИ.

Как известно, понятие протокола в большой степени родственно определению формальных языков и языков программирования высокого уровня. Работа БВУВ представляется идентичной процессу трансляции фраз входного языка во фразы выходного в соответствии с некоторой формальной совокупностью правил. Широкий класс формальных языков составляет языки сетей Петри, поэтому рассмотрим возможность задания языков трансляции для БВУВ и на их основе.

Формальный язык определяется системой правил, называемых грамматикой языка, которая задается как кортеж из четырех элементов $G^L = (\Gamma, \Sigma, \Delta, \Gamma_0)$, где $\Gamma$ — конечное число нетерминальных символов (синтаксических переменных); $\Sigma$ — множество терминальных символов;

$$\Gamma \cap \Sigma \otimes,$$

$\Delta$ — конечное число правил подстановки вида $\alpha^L \to \beta^L$; $\Gamma_0$ — начальный символ; $\Gamma_0 \in \Gamma$.

Любая размеченная сеть Петри может являться формой задания формального языка $L(N_M)$ (см. гл. 3). При этом $L(N_M)$ зависит от структуры $N_M$, начальной разметки $M_0$, а также функции индексирования переходов $\varphi$. При синтезе грамматики языка, описывающего работу БВУВ, множество терминальных символов $\Sigma$ будет включать элементы, указывающие на тип кадра и результаты его обработки в блоке анализа кадров. Множество же $E$ выходных сигналов блока трансляции содержит команды на выдачу кадров определенного типа, установку элементов области управления кадров, изменение внутренних переменных станции протокола.

Воспользуемся языком сети Петри типа T, который определяется как

$$L_T(N_M) = \left\{\Omega \in \Sigma^* \,|\, M_F,\ M_0 \overset{\tau}{\vdash} M_F\right\},$$

где $\Omega$ — слово формального языка; $\Sigma^*$—множество всех слов в алфавите $\Sigma$; $M$ — элемент множества $R(N_M)$ всех допустимых в $N_M$ разметок; $\tau$ — последовательность срабатывания переходов сети.

Будем использовать язык с $\lambda$-свободной функцией индексирования переходов $\varphi$: $D \to \Sigma$, в котором допускается индексирование различных переходов $d_i$ одним и тем же символом из $\Sigma$.

Модель на базе языка сети Петри будем строить в четыре этапа:

Первый этап. Построим сети Петри, задающие грамматику соответствующего языка БВУВ для каждого из состояний ПС и ВС протокола. На рис. 4.32 приведен пример такой сети для состояния ВС протокола $D^*_{4H}$.

Второй этап. Грамматика, задаваемая сетью Петри (рис. 4.32), относится к грамматикам типа 3. Все правила $\Delta$ грамматики такого типа имеют вид $A \to \alpha^L B$, где $A, B \in \Gamma$, а $\alpha^L \in \Sigma^* - [\Lambda]$, где $\Lambda$ — пустое слово. Это

обстоятельство позволяет задать формально грамматику, например для анализа в состоянии, как

$$G^{L}_{D''_{4H}} = (\{b_0,\ b_1,\ b_2,\ b_3,\ b_4\},\ \{\text{sarm},\ \text{disc},,\ i,\ ce,\ le,\ rr;$$

$$rnr,\ rej,\ p0,\ p_1\},\ \{b_0 \to \text{sarm}\, b_1,\ b_1 \to p0;\ \ b_0 \to \text{disc}\, b_2,\ b_2 \to p0,$$

$$b_0 \to ib_3,\ b_3 \to p1;\ \ b_3 \to p0,\ b_0 \to l_e,\ b_0 \to ce,\ b_0 \to rrb_4;$$

$$b_0 \to rnrb_4,\ b_0 \to rejb_4,\ b_4 \to p1\},\ b_0).$$

Однако при такой форме записи не учитываются терминальные разметки сети Петри. Позиции (*fcmdr*),

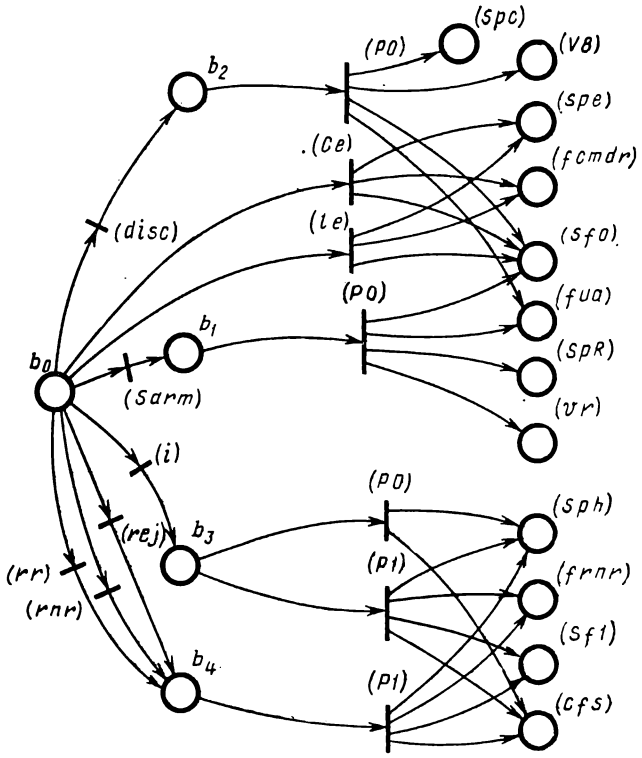



Рис. 4.32. Сеть Петри для состояния ВС $D''_{4H}$:

$b_0$ — начальная позиция сети Петри; $b_1$—$b_4$—внутренние позиции; (*i*), (*rr*), (*rnr*), (*rej*), (*disc*); (*sarm*) — тип принятого кадра; (*p0*), (*p1*) — значение бита $P$=0 или $P$=1; (*ce*), (*le*) — ошибочная область управления кадра команды или кадра ответа; (*fua*), (*frnr*), (*fcmdr*) — выдать кадр типа *UA*, *RNR* или *CMDR*; (*sf1*), (*sf0*) — установить бит $F$=1 или $F$=0; (*spe*), (*spk*), (*sph*), (*spc*) — установить состояние $BCD''_{4E}$, $D''_{4K}$, $D''_{4H}$ или $D''_{4P}$; (*vr*) — выполнить $V(R) := 0$; (*cfs*) — передать слово на транслятор ПС для анализа номера $N(R)$; (*v8*) — выдать сигнал *V8*

(*fua*), (*frnr*), (*sf*1), (*sf*0), (*spc*), (*spe*), (*spk*), (*sph*), (*vr*), (*cfs*), (*v*8) задают множество выходных сигналов блока трансляции для состояния $D''_{4H}$. Так как целью формального описания является не только распознавание допустимых слов $L_T(N_M)$, а также и определение выходного слова, дополним запись грамматики $G^L$ выражениями для тупиковых разметок $M_F$. Вместо формы $A \to \alpha^L$ будем использовать $A \to \alpha^L(M_i)$, где $M_i \in R(N_M)$. Такая модификация записи $G^L$ позволяет полностью формально задать соответствующую сеть Петри. Так, для состояния $D''_{4E}$ имеем:

$$G^L_{D''_{4E}} = (\{b_0, b_1, b_2, b_3, b_4\}, \{\text{sarm, disc, } i, rr, rnr;$$

$$rej, p0, p1\}, \{b_0 \to \text{sarm}\, b_1, b_1 \to p_0(M'_1);$$

$$b_0 \to \text{disce}\, b_2, b_2 \to p0(M'_2), b_0 \to ib_3, b_3 \to p1(M'_4);$$

$$b_3 \to p0(M'_3), b_0 \to rrb_4, b_0 \to rnrb_4;$$

$$b_0 \to rejb_4, b_4 \to p1(M'_4)\}, b_0),$$

где

$$M'_1 = \{(spk), (vr), (fua), (sf0)\};$$

$$M'_2 = \{(v8), (spc), (fua), (sf0)\};$$

$$M'_3 = \{(cfs),(spe)\};$$

$$M'_4 = \{(cfs), (spe), (fcmdr)\}.$$

Третий этап. На основе полученных аналогичным образом грамматик для каждого из состояний ПС и ВС протокола строятся объединенные грамматики ПС и ВС. При этом число внутренних позиций сети минимизируется путем объединения повторяющихся последовательностей переходов.

Четвертый этап. Грамматика типа 3 допускает табличную форму записи. Такая форма $G^L_{\text{ВС}}$ для *ВС* содержит 20×17 элементов. Удобно перейти к канонической форме таблицы. Для ПС построим две таблицы: $G^{L1}_{\text{ПС}}$ (объем 5×8 элементов) и $G^{L2}_{\text{ПС}}$ (объем 35×26 элементов). Табличная форма представления работы БВУВ позволяет непосредственно перейти к его реализации в ПОМП устройства СПИ. При этом обеспечивается простота модификации ПО при изменении версии протокола.

Рассмотренная выше методика формализации является, разумеется, только одной из возможных. На этапе

формализации могут использоваться обычные и раскрашенные сети Петри — для целей верификации (проверки логической правильности) процедур выбранной версии протокола; чисто автоматные модели в форме абстрактной протокольной машины; комбинированные модели с использованием языков высокого уровня и многие другие

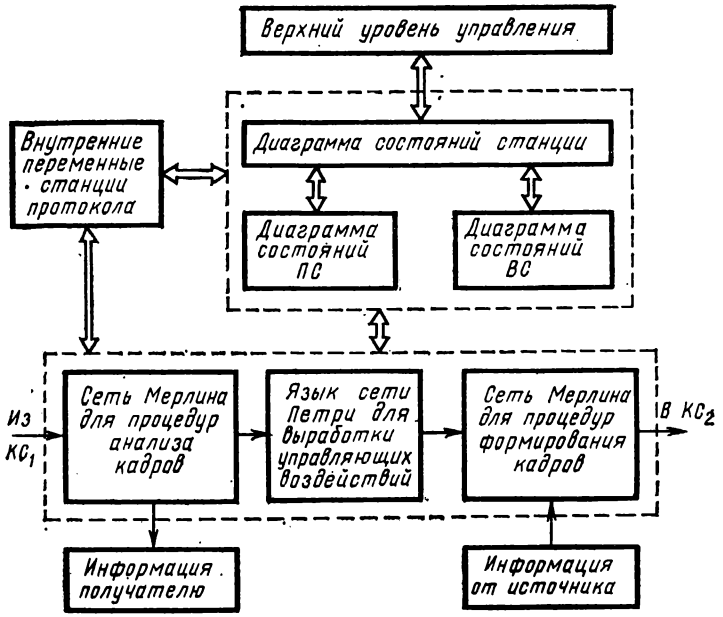



Рис. 4.33. Иерархическая модель процедур протокола HDLC

методы. Выбор определенных математических средств формализации целиком зависит от задачи дальнейшего использования получаемой спецификации.

Рассмотренная в данном параграфе иерархическая гибридная модель (рис. 4.33) ориентирована целиком на упрощение перехода к программной реализации протокола HDLC в микропроцессорном устройстве СПИ.

## § 4.7. МОДЕЛЬ ИНФОРМАЦИОННОГО КАНАЛА, РЕАЛИЗУЮЩЕГО БИТ-ОРИЕНТИРОВАННЫЙ ПРОТОКОЛ

Важное место в процессе разработки микропроцессорных устройств СПИ занимают вопросы согласования данного устройства с конкретным каналом связи. Решение

этих вопросов необходимо как на этапе анализа ТЗ на МП устройство, так и при разработке его ПО. В связи с этим часто возникает задача разработки моделей информационного, сетевого или же транспортного канала, которые бы позволяли как производить оценку пригодности заданных протоколов и их версий для создания канала требуемого качества, так и осуществлять выбор системных параметров протокола, обеспечивающих оптимальное по заданному критерию функционирование канала. Выбор системных параметров протокола входит и в комплекс вопросов определения оптимальной структуры МП устройства СПИ, так как он влияет на важный параметр — требуемый минимальный объем буферного ЗУ устройства. Рассмотрим задачу построения модели канала связи применительно к информационному каналу, использующему протокол HDLC.

В группу системных (устанавливаемых пользователем) параметров протокола HDLC входят: максимальная длина внутренней области кадра типа I (L1); максимальное число переданных, но не подтвержденных кадров — «окно передачи» (K1); тайм-аут ожидания подтверждения на переданный кадр (T1); максимальное число повторных передач кадра (N2).

Аналитические модели на основе аппарата марковских цепей не дают возможности учета ряда важных процедур протокола и их взаимосвязи. Единственным методом, позволяющим создать модель необходимой полноты и точности, является использование аппарата имитационного моделирования. Основными имитационными параметрами протокола HDLC являются средняя эффективная скорость передачи информации $\overline{V}_I$, среднее время доставки сообщения адресату $\overline{T}_д$ и вероятность необнаруженной ошибки при передаче $P_{н.о.}$ Очевидно, что на модели должно быть обеспечено получение данных параметров.

Рассмотрим вопрос выбора степени детализации разрабатываемой модели протокола. За основу целесообразно взять наиболее обобщенную модель протокола в форме диаграмм состояний (см. рис. 4.25, 4.26, 4.27). В процессе моделирования ИК нельзя сделать каких-либо обоснованных предположений о длительности сеансов связи между абонентами ИВС и о длине самих сообщений, передаваемых между абонентами. Поэтому ограничимся общим случаем симметричного трафика с постоянным наличием данных для передачи на обеих станциях прото-

кола и исключим из рассмотрения на модели процедуры инициализации и деинициализации ИК. Введем также допущение о невозможности отказа станции HDLC от приема информации (при перегрузках процессора, ЗУ), т. е. исключим из рассмотрения состояния $D'_{4L}$ («запрет передачи») и $D''_{4H}$ («занято»). Целесообразным представляется также исключение из модели режима нахождения ВС в состоянии $D''_{4E}$ («неприем команды»), так как такое состояние возникает достаточно редко — только в случае поражения необнаруженными ошибками поля управления (С) кадра и принятие решения при возникновении

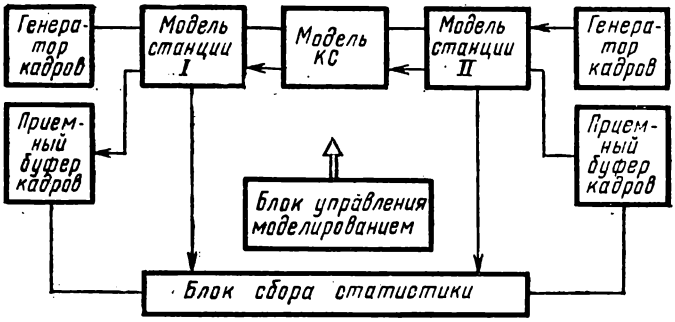



Рис. 4.34. Структурная схема модели ИК

указанного состояния возлагается на протокол более высокого уровня, а следовательно, оно выходит за рамки разрабатываемой модели. Блок-схема такой имитационной модели приведена на рис. 4.34.

В результате введенных ограничений определим задачу имитационной модели как отображение работы инициированного ИК при следующих состояниях ПС и ВС протокола HDLC (рис. 4.35, *а*): $D'_{4R}$ — передача данных; $D'_{4T}$ — восстановление по истечении тайм-аута; $D'_{4S}$ — повторное установление соединения; $D''_{4K}$ — прием данных; $D''_{4J}$ — восстановление посредством REJ.

Ограниченное число рассматриваемых состояний позволяет упростить модель при введении понятия общего состояния станции протокола HDLC как совокупности состояний ПС и ВС. Получаемая при таком объединении диаграмма состояний станции показана на рис. 4.35, *б*.

Для реализации модели воспользуемся специализированным языком моделирования GPSS. Потоки кадров будем моделировать динамическими объединениями языка GPSS — транзактами. Параметры транзактов используем для хранения характеристик кадров — адреса, типа кадра, номеров $N(S)$ и $N(R)$, бита P/F, отметок времени выдачи кадра и т. д. Реализацию модели проведем по модульному признаку в соответствии с рис. 4.34.

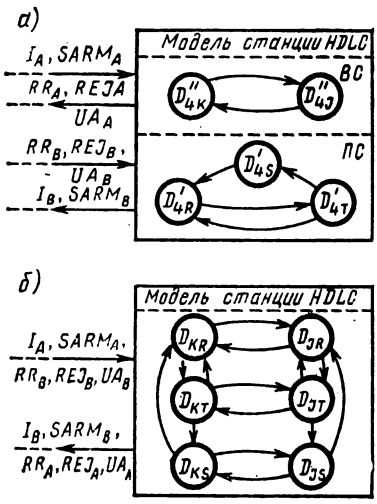



Рис. 4.35. Диаграмма состояний модели станции протокола HDLC при раздельном моделировании ПС и ВС *(а)* и для объединенной станции протокола *(б)*

При разработке модулей модели необходимо учесть такие особенности протокола, как более высокий приоритет на выдачу супервизорных и ненумерованных кадров (S- и U-типов), использование одной флаговой комбинации одновременно как для закрывающей для одного кадра и открывающей его следующего, временна́я буферизация выданных абоненту, но еще не подтвержденных кадров типа I. Вместе с тем откажемся от учета в модели процедуры бит-стаффинга. Функция распределения избыточности за счет бит-стаффинга зависит от совокупности различных факторов — типа внутреннего информационного кода, характера передаваемой информации, специализации системы. Статистические данные такого рода в настоящее время отсутствуют. При знании конкретного назначения системы может быть построена частная модель для определения распределения избыточности за счет бит-стаффинга и соответствующая функция легко включается в общую модель протокола. Ориентировочный же анализ показывает, что среднее значение данной избыточности не превышает 1...2 %. Таким образом, отказ от моделирования данной функции не повлияет существенно на конечные результаты.

На рис. 4.36 приведена внутренняя структура модуля,

имитирующего работу станции протокола HDLC. Модуль отражает работу станции в фазе передачи данных, когда канал обмена информацией установлен, т. е. модель отражает режим постоянного виртуального канала. Основным блоком этого модуля является блок анализа кадров, работа которого организуется в соответствии с выделенными на рис. 4.35 шестью совместными состояниями станции протокола. Поступивший из модели КС

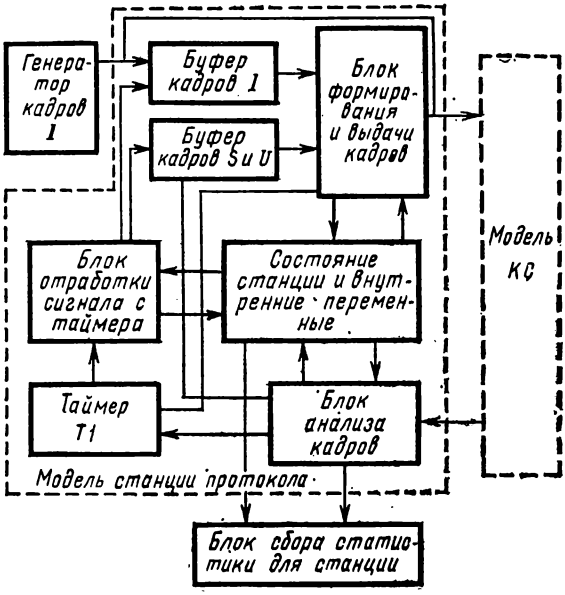



Рис. 4.36. Модель станции протокола HDLC

транзакт обрабатывается по одному из алгоритмов в соответствии с текущим состоянием станции, в результате чего происходит выдача сигналов на останов таймера T1, выдача ответных кадров типов S или U, на изменение состояния станции и ее внутренних переменных. При приеме кадра I без ошибок результаты посылаются также и в блок сбора статистики.

Входной поток транзактов, имитирующих кадры для передачи, подается в модель буфера кадров I. Буфер кадров I связан с блоком формирования и выдачи кадров в КС, где принимается решение о выдаче в КС кадров типов I, S или U, производится прерывание формирования

кадра типа I в случае необходимости посылки служебного кадра. При моделировании процедуры формирования кадра выполняется запись соответствующих признаков в параметры транзакта, имитируется время задержки на выдачу кадра в КС в зависимости от его длины и скорости модуляции в КС.

Модуль дуплексного КС между станциями протокола целесообразно разделить на ряд блоков (рис. 4.37). Наиболее сложным является блок формирования векторов ошибок $\{e_i\}$, где производится моделирование векторов $e_i=(e_1, e_2, e_3)$ для открывающей кадр флаговой комбинации ($e_1$), внутренней области кадра ($e_2$) и закрывающей кадр флаговой комбинации ($e_3$). Такое разделение необходимо в связи с тем, что флаговые комбинации кадра не охвачены циклическим кодом. В модель КС внесена процедура декодирования кадра в режиме обнаружения ошибок, моделирование которой выполняется делением вектора $e_2$ на образующий полином циклического кода $g(x)=x^{16}+x^{12}+x^5+1$. Для сокращения затрат времени на моделирование целесообразно выполнять декодирование только в том случае, когда число ошибок во внутренней области кадра превышает гарантируемую обнаруживающую способность применяемого циклического кода. После выполнения декодирования число необнаруженных ошибок записывается в один из параметров транзакта (для расчета вероятности необнаруженной ошибки $P_{н.о}$).

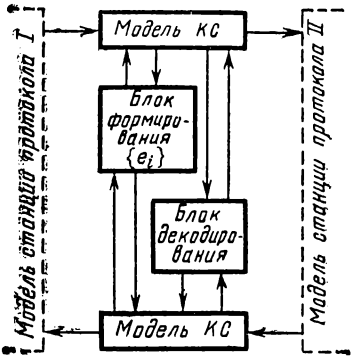


Рис. 4.37. Модель дуплексного КС

В моделях $КС_1$ и $КС_2$ имитируется время задержки кадра в канале в зависимости от типа и длины линии связи. Моделируется также работа приемной части станции HDLC по стиранию кадров при обнаружении ошибок во флаговых комбинациях или внутренней области кадра.

Для формирования $\{e_i\}$ удобно воспользоваться моделью двоичного симметричного КС с памятью, предложенной Э. Гильбертом. Эта модель отражает процессы пакетирования ошибок в КС и отличается относительной простотой получения по результатам обработки экспери-

ментальных данных об ошибках параметров $h$, $q_{01}$ и $q_{10}$, задающих модель Гильберта. Более подробно вопрос получения значений $h$, $q_{01}$ и $q_{10}$ будет рассмотрен в следующем разделе.

## § 4.8. ВЫБОР СИСТЕМНЫХ ПАРАМЕТРОВ ПРОТОКОЛА

Рассмотрим использование описанной выше модели для решения задачи выбора системных параметров протокола по заданному критерию, для чего воспользуемся сложившимся при исследовании протоколов подходом, который заключается в поиске максимума средней эффективности скорости передачи информации $\overline{V}_I$ при наложении ограничений на остальные параметры, и сформулируем *критерий выбора системных параметров протокола* HDLC ($K_P$) как $K_P = \max\limits_{\{\varepsilon_{P_t}\}} V_I$, при $T_д \leqslant [T_д]; P_{н.о} \leqslant [P_{н.о}]$, где $\{e_{P_t}\}$ — множество допустимых вариантов протокола, причем $\varepsilon_{P_t} = (L1_t, K1_t, N2_t, T1_t)$.

Определим скорость $V_I$ как $V_I = (L1 - m)N_I/T_C$, где $N_I$ — число правильно принятых кадров типа I на станции за время сеанса связи; $T_c$ — длительность сеанса связи; $m$ — число служебных битов в кадре типа I.

Время $T_д$ будем определять как длительность временно́го интервала между началом выдачи кадра I в КС на одной станции протокола и моментом конца приема доставленного без ошибок кадра на другой станции.

Основные информационные параметры $\overline{V}_I$, $\overline{T}_д$ и $P_{н.о}$ вычисляются на модели в блоках сбора статистики о результатах моделирования (см. рис. 4.36) и усредняются по двум станциям протокола.

Остановимся теперь на вопросе отражения в модели характеристик реального КС, для которого производится выбор системных параметров протокола. Как известно, путем обработки статистики ошибок в реальном КС может быть получена экспериментальная функция распределения длин интервалов между ошибками $F_{эксп}(u)$. Данную функцию удобно аппроксимировать как

$$F_{эксп}(u) = \sum_{i=1}^{K_b} A_{эксп_i} \exp\left(-\varphi_{эксп_i} u\right), \qquad (4.2)$$

где $i = \overline{1, k_B}$; $A_{эксп_i}$, $\varphi_{эксп_i}$ — параметры экспериментальной модели.

Учитывая, что в ИВС чаще всего используются КС, в которых ошибки обусловлены в основном кратковременными перерывами и импульсными помехами, можно представить $F_{\text{эксп}}(u)$ как

$$F_{\text{эксп}}(u) = A_1 \exp(-\varphi_1 u) + (1 - A_1) \exp(-\varphi_2 u). \quad (4.3)$$

Для такой формы экспериментальной модели требуется задание параметров $A_1$, $\varphi_1$ и $\varphi_2$. Получение параметров модели Гильберта основано на методике аппроксимации $F_{\text{эксп}}(u)$ суммой двух геометрических прогрессий

$$F_{\text{эксп}}(u) = A_G J_G^u + (1 - A_G) L_G^u. \quad (4.4)$$

Первая прогрессия в этом выражении отражает распределение длин интервалов между пакетами ошибок, а вторая — определяет вероятность появления «плохих» и «хороших» состояний внутри пакета ошибок. Параметры $A_G$, $J_G$ и $L_G$ находятся путем подбора кривой вида $\{A_G J_g^u + (1 - A_G) L_G^u\}$, аппроксимирующей $F_{\text{эксп}}(u)$. Сначала выбираются $A_G$ и $J_G$, чтобы правильно определить поведение $A_G J_G^u$ при больших $u$, затем определяется $L_G$, чтобы улучшить соответствие при малых $u$. Параметры модели Гильберта определяются из таких выражений:

$$h = \frac{L_G J_G}{J_G - A_G (J_G - L_G)};$$

$$q_{01} = \frac{(1 - L_G)(1 - J_G)}{1 - h};$$

$$q_{10} = A_G (J_G - L_G( + (1 - J_G) \left( \frac{L_G - h}{1 - h} \right).$$

Сравнение (4.3) и (4.4) показывает, что параметры модели Гильберта связаны с параметрами экспериментальной модели соотношениями: $A_1 = A_G$; $\varphi_1 = -\ln J_G$; $\varphi_2 = -\ln L_G$.

Управление имитационным экспериментом на модели протокола HDLC реализуется в блоке управления моделированием. В число входных параметров модели кроме рассмотренных выше коэффициентов модели Гильберта $h$, $q_{01}$ и $q_{10}$ входят следующие величины: $v_{\text{м}}$ — скорость модуляции в КС; $T_{\text{с}}$ — длительность сеанса связи; $l_{\text{КС}_1}$, $l_{\text{КС}_2}$ — длина физической линии связи для $\text{КС}_1$ и $\text{КС}_2$; $t_{\text{КС}_1}$, $t_{\text{КС}_2}$ — средняя задержка сигнала на единицу длины в физических линиях для $\text{КС}_1$ и $\text{КС}_2$.

Таким образом, полный вектор входных параметров модели, устанавливаемый в начале моделирования,

$$\mu^M = (L1, K1, T1, N2, V_M, T_C, l_{KC_1}, l_{KC_1}, t_{KC_2}, h, q_{01}, q_{10}).$$

Очевидно, что решение задачи оптимизации путем полного перебора на имитационной модели широкого диапазона системных параметров $L1$, $K1$, $T1$ и $N2$ весьма трудоемко. Задача, однако, упрощается, так как в реальных системах, использующих протокол HDLC, существуют весьма жесткие ограничения на эти параметры. Обычно величина $L1$ устанавливается дискретно со значениями, являющимися степенью числа 2 и лежащими в диапазоне 16...1024 байт. Число повторов кадра $N2$ редко составляет менее 3 и более 5. Параметр $K1$ является также дискретным со значениями от 1 до 7. Тайм-аут ожидания подтверждения принято настраивать на максимальную величину задержки прихода подтверждения на переданный кадр. При этом исследуется достаточно узкий интервал времен, близких к этому вычисленному значению. Таким образом, для конкретного применения МП устройства СПИ число исследуемых на модели сочетаний системных параметров протокола конечно (обычно несколько десятков) и вполне приемлемо по затратам времени на современных высокопроизводительных ЭВМ.

## § 4.9. ТЕХНИЧЕСКОЕ ПРОЕКТИРОВАНИЕ МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВ СПИ

### Общая технологическая схема технического проектирования

Как было показано выше, на этапе системного проектирования МП устройств СПИ выбирается базовая структура устройства. Таким образом, в распоряжении разработчика, переходящего к техническому проектированию МП-системы, имеются: функциональная структура МП-устройства СПИ; физическая структура МП-системы; формализованное описание важнейших функций устройства (подмножества сетевых функций); выбранная программно-аппаратная структура устройства, т. е. отображение элементов функциональной структуры $F$ на физическую структуру МП системы $W$.

Основной целью этапа технического проектирования МП-устройств СПИ является разработка и испытание всех необходимых аппаратных и программных средств на

макете устройства, составление технической документации на производство его опытного образца.

Общая последовательность этапов технического проектирования приведена на рис. 4.38. После разработки детализированных спецификаций на элементы программно-аппаратной структуры устройства работа по его созданию разбивается на две параллельные ветви: первая — включает разработку всех необходимых аппаратных средств и компоновку аппаратурного макета МП-устройства; вторая — содержит все работы по проектированию программных средств МП-системы. После завершения указанных выше работ производится объединение аппаратной и программной частей устройства. Выполняется тестирование и отладка макета МП-устройства, выявляются ошибки в аппаратных и программных модулях. В случае обнаружения существенных ошибок может потребоваться корректировка аппаратурного комплекса или же программных средств системы. По завершении комплексных испытаний макета разра-

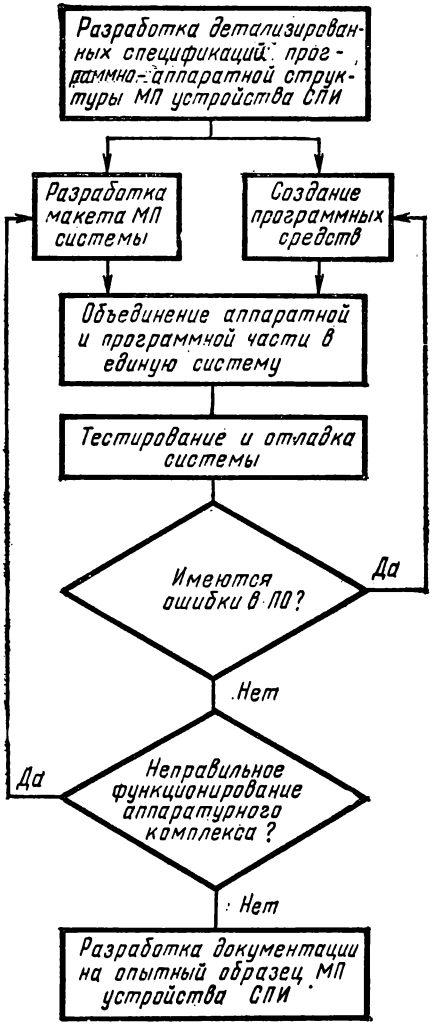



Рис. 4.38. Основные этапы технического проектирования МП устройства СПИ

батывается документация на изготовление опытного образца устройства СПИ.

Конкретизируем отдельные наиболее важные блоки схемы 4.38. В том случае, когда для проектирования устройства используется комплект МП БИС, а не серийная микроЭВМ, разработка начинается с построения структурной схемы макета, на которой детализируется и уточняется применительно к особенностям МПК БИС ранее выбранная физическая структура устройства $W$. Далее процедура делится на ряд параллельных ветвей. Практически независимо могут создаваться модуль процессора, подсистема памяти, система ввода — вывода, пульт МП-системы, а также разрабатываться диагностические процедуры (тесты) для последующей проверки всего комплекса. Каждая из подсистем проверяется автономно. Затем производится сборка макета из модулей и его откладка с помощью диагностических процедур.

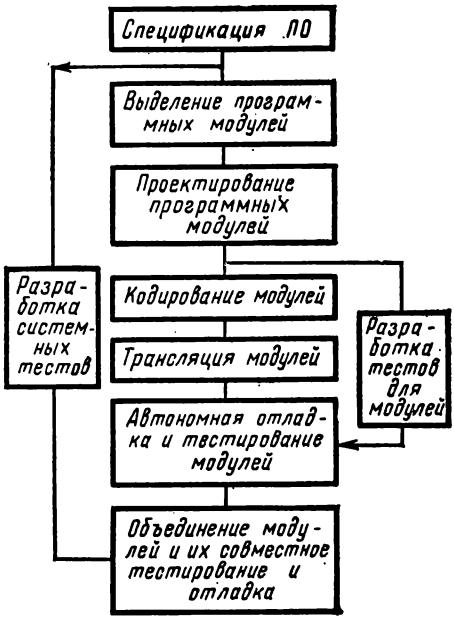



Рис. 4.39. Детализация этапа разработки программных средств МП системы

При использовании же серийной микроЭВМ, создание макета значительно упрощается. Разрабатываются только дополнительные аппаратные средства (сопряжения с УВВ, модули памяти) и проверяется их работа совместно с микроЭВМ.

Одновременно с разработкой макета создаются программные средства МП-устройства СПИ. Формулирование требований к ПО выполняется еще на этапе системного проектирования МП-устройства. Непосредственная разработка ПО начинается с составления внешних спецификаций (рис. 4.39). Определяются входы и выходы

программы и их связи. Далее осуществляется декомпозиция общей задачи, которую необходимо решить, на отдельные фрагменты путем выделения набора взаимодействующих программных модулей высокого уровня. При этом создаются внутренние спецификации — определяются входы и выходы каждого модуля и его взаимосвязи с другими модулями.

Затем проектируется каждый программный модуль в отдельности. В процессе разработки может осуществляться дальнейшая декомпозиция модуля на подмодули. Проектирование модуля включает выбор алгоритма решения поставленной перед модулем задачи и определение структур данных. При создании внешних, внутренних спецификаций и проектировании модулей производится подробное документирование разработки.

На этапе проектирования выполняется кодирование каждого из программных модулей на языке ассемблера или на одном из языков высокого уровня. Далее осуществляется трансляция полученной программы и ее откладка и тестирование, после чего для каждого из программных модулей производится их сборка в единую программу, над которой выполняется аналогичная процедура откладки и тестирования с использованием системных тестов.

Разработка ПО ведется обычно параллельно с изготовлением аппаратурного макета (см. рис. 4.38). В связи с этим при каждой разработке актуальна задача выбора или создания редакционно-отладочного комплекса для ускорения работ по программированию. В ряде случаев часть работ по отладке может вестись с использованием больших ЭВМ. Сокращение срока разработки ПО дает применение в современных МП-системах языков программирования высокого уровня. Качество создаваемых программных устройств в большой степени зависит от правильного документирования ПО и его полного тестирования. Остановимся кратко на основных особенностях перечисления процедур этапа технического проектирования.

### Редакционно-отладочные комплексы

Одним из методов ускорения процесса проектирования МП-системы является использование редакционно-отладочных комплексов на базе микроЭВМ (их часто называ-

ют также инструментальными микроЭВМ, специализированными системами проектирования микроЭВМ).

Редакционно-отладочные комплексы (РОК), используются для проверки аппаратных и программных средств разрабатываемой МП-системы и обеспечивают аппаратное и программное обслуживание всей процедуры проектирования, начиная от разработки программ и кончая отладкой прототипа (макета) системы.

Существующие РОК строятся обычно на том же типе МП, который будет работать в проектируемом устройстве. В настоящее время такие отладочные комплексы имеются для большинства моделей МП. РОК предоставляют разработчику разнообразные средства для отладки программ (в том числе в реальном масштабе времени); текущего исправления ошибок, редактирования исходных и отлаженных текстов; формирования входных перфолент с текстами, подготовленными для реализации в БИС ПЗУ; отображения и редактирования информации на экране дисплея.

Подсистема РОК «диалог с отладчиком» дает программисту широкие возможности контроля выполнения отлаживаемой программы с обширной индикацией ее контролируемых элементов. Подсистемы «диалог с корректором дисплея» и «диалог с редактором программ» осуществляют редактирование исходных и отлаженных текстов программ, управляемое функциональной клавиатурой.

Широкое распространение получили в настоящее время РОК, которые наряду с перечисленными выше функциями позволяют производить также и отладку аппаратуры разрабатываемого комплекса. Рассмотрим типичную структуру такого РОК (рис. 4.40). РОК представляет собой магистральную МП-систему, где в качестве центрального процессора используется МП такого типа, на обеспечение которого рассчитан данный РОК. Система содержит кроме ОЗУ (32...64 Кбайт) память на гибком магнитном диске (емкость одного диска около 250 Кбайт).

Операционная система РОК включает системный монитор (хранится в ПЗУ) и ряд системных программ, размещенных на гибком диске. Основные функции монитора — управление аппаратными и программными ресурсами и обеспечение взаимодействия пользователя с системой. В качестве средства связи пользователя с системой используется дисплей. В монитор входят програм-

мы управления файлами и подпрограммы ввода—вывода. Системные программы включают редактор текста, ассемблер, интерпретаторы и компиляторы для языков БЕЙСИК, ФОРТРАН, *PL/M*, ПАСКАЛЬ и т. д., редактор связей и набор утилитов (подпрограмма отладки в реальном масштабе времени, подпрограмма управления программатором ППЗУ и др.).

В состав РОК входит устройство печати, обеспечивающее документирование программ и данных. Для работы

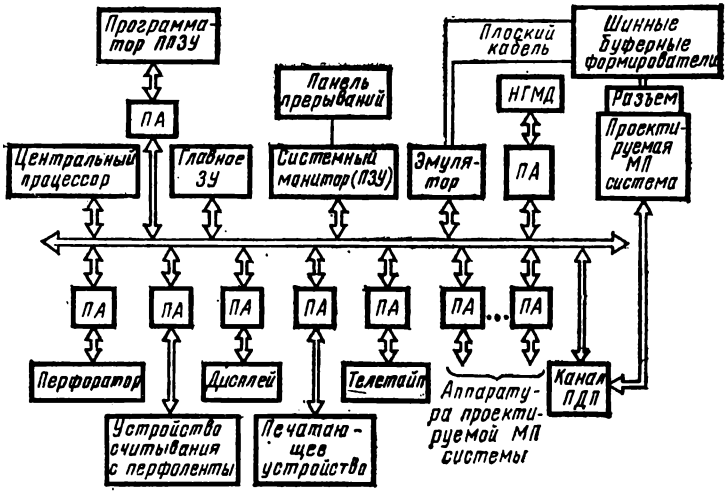



Рис. 4.40. Типичная структура РОК на базе МП

с перфолентами используются устройства ввода и вывода на перфоленту. В качестве УВВ может применяться также телетайп.

Отлаженные программные модули с помощью РОК могут быть занесены в БИС ППЗУ для последующей установки в прототипе. БИС ППЗУ подключается к РОК через соответствующие разъемы на передней панели комплекса. Для облегчения работы с программатором он иногда комплектуется управляющим модулем с МП. Это позволяет с помощью одного приказа записать в ППЗУ блок данных от РОК.

Специализированным модулем РОК является *эмулятор*. Главные компоненты эмулятора — это МП, эмулирующий процессор проектируемой МП системы (прото-

типа), схема управления для прослеживания и контроля процессом эмуляции, а также схемы интерфейса с РОК. Функцией эмулятора является замена МП прототипа и выполнение команд МП в среде прототипа под управлением РОК. Для проведения эмуляции в макете разрабатываемой МП-системы (прототипе) удаляется МП из разъема на плате и вместо него подсоединяется разъем, подключаемый к эмулятору через шинные буферные формирователи и плоский кабель. С использованием эмулятора прототип может при отладке пользоваться всеми аппаратными и программными ресурсами РОК. РОК имеет ПА для подключения устройств прототипа и контроллер ПДП для организации прямого доступа к памяти к прототипу.

С помощью РОК такого типа можно производить различные процедуры технического проектирования МП-устройства СПИ, например: отладку программных модулей с использованием центрального процессора РОК; отладку ПО в комплексе с помощью ЦП РОК в реальном масштабе времени; поэтапную отладку аппаратурного комплекса прототипа с постепенным переключением на прототип модулей ЗУ, ввода—вывода и др. (МП-прототипа заменяется эмулятором). После выполнения таких процедур производится замена в прототипе эмулятора на МП и окончательная проверка макета МП-устройства.

Большинство РОК рассчитаны на отладку систем с одним конкретным типом МП. Однако разрабатываются и более сложные комплексы (например, 8002 фирмы Tektronics), которые могут использоваться при отладке МП-систем на базе различных МП семейств.

### Использование больших ЭВМ для решения задач программирования

Проектирование ПО часто ведется с использованием больших (или мини) ЭВМ, имеющихся в распоряжении разработчика. Совокупность служебных программ, предназначенных для создания рабочих программ на больших или миниЭВМ, называется *кросс-программным обеспечением микроЭВМ,* а ЭВМ, используемая для этой цели, — *технологической ЭВМ.* В состав кросс-средств для разработки микроЭВМ входят программы кросс-ассемблер, кросс-комплиляторы с языков высокого уровня (*PL/M*, БЕЙСИК, ПАСКАЛЬ, ЗЕНИТ и т. д.), загрузчики и отладчики, моделирующая программа (имитатор). Общая

структурная схема процедуры разработки и отладки ПО с использованием кросс-средств приведена на рис. 4.41.

Работа кросс-средств начинается с трансляции исходной программы (модуля), написанной программистом. Она может быть представлена на языке ассемблера или же на одном из языков высокого уровня (рис. 4.42). Программа на языке ассемблера транслируется с помощью *кросс-ассемблера* на машинный язык, образуя *объектный модуль*. Если исходный модуль написан на языке высокого уровня, то трансляция производится с помощью соответствующей программы *кросс-компилятора*. В этом слу-

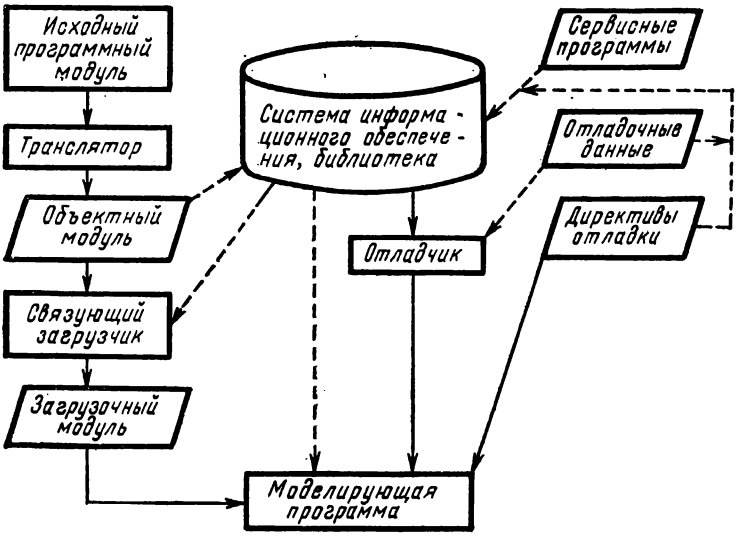



Рис. 4.41. Структурная схема использования кросс-средств для отладки ПО микроЭВМ

чае один оператор исходной программы дает от 5 до 10 команд. Программы кросс-ассемблера и кросс-компилятора выдают листинг программы, на котором приводятся обе версии программы — исходная и объектная, список сообщений об ошибках и другие виды диагностической информации.

Программа, транслированная с языка высокого уровня, является, как правило, неоптимальной как в смысле времени исполнения, так и по используемому объему памяти. В ряде кросс-компиляторов пользователю предоставля-

ется возможность просмотреть программу, оттранслированную на язык ассемблера, с целью ее доработки и оптимизации, после чего производится трансляция уже с языка ассемблера на машинный язык.

Полученные объектные модули заносят в систему информационного обеспечения. Следующей процедурой является объединение отдельно оттранслированных программ и их настройка на конкретные адреса. Эту операцию производит *программа загрузчик*. Результат работы также выводится на печать. Машинная программа, полученная после загрузки (загрузочный модуль), отлаживается с помощью *моделирующей программы*.

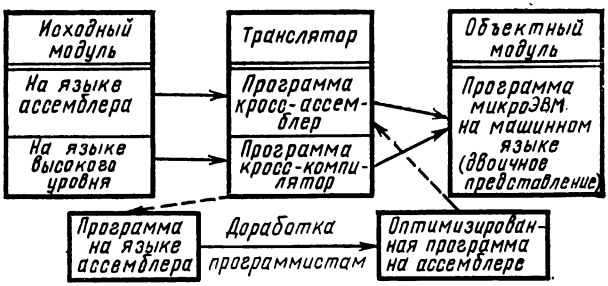



Рис. 4.42. Трансляция программы на кросс-средствах

Моделирование МП — это процесс выполнения программы, составленной в системе команд данного МП, на другой ЭВМ. Архитектура МП моделируется блоком памяти в большой машине, т. е. счетчик команд, регистр состояния, рабочие регистры и память МП представляются ячейками памяти, а каждая команда моделируется несколькими командами большой ЭВМ. Моделирующая программа использует загрузочный модуль с программой пользователя, отладочные данные, вводимые через программу-отладчик и сервис, предоставляемый системой информационного обеспечения. Моделирование выполняется на основании директив отладки пользователя. Моделирующие команды обычно позволяют выводить на дисплей или печать содержимое памяти моделируемой микроЭВМ и ее регистров, указывать контрольные точки в программе, в которых она должна быть приостановлена при достижении некоторого адреса или при чтении или записи в определенную ячейку памяти, выдавать листинг с печатью каждой команды, принадлежащей области адресов,

указанной в директиве, предоставлять пользователю информацию о времени выполнения программы, например число команд и (или) машинных тактов, выполненных от начала работы программы до ее останова.

Кросс-средства могут использоваться на больших ЭВМ в двух режимах — пакетном и диалоговом. Диалоговая отладка с помощью терминала значительно повышает эффективность работы. Рассмотрим работу диалогового отладочного комплекса на примере системы СИДОП, предназначенной для разработки программ микроЭВМ серии «Электроника С5» (рис. 4.43).

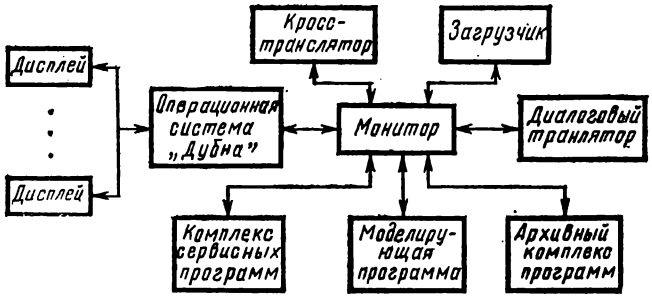



Рис. 4.43. Структура диалогового отладочного комплекса СИДОП на базе ЭВМ БЭСМ-6

Система СИДОП реализована на ЭВМ БЭСМ-6 с операционной системой «Дубна». Управление выполнением процедур в режиме разделения времени осуществляет программа монитор. Запросы (ответы) пользователей СИДОП проходят синтаксический и семантический контроль и переводятся во внутреннее представление системы программой диалоговый транслятор. В состав СИДОП входят архивный комплекс программ, реализующий прямой доступ к внешней памяти ЭВМ БЭСМ-6, и комплекс сервисных программ, дающий возможность диалогового редактирования. Кроме того, в СИДОП входят кросс-средства отладки: транслятор в систему команд микроЭВМ, загрузчик и моделирующая программа. СИДОП позволяет проводить в интерактивном режиме процедуры отладки с временным разделением ресурсов между терминалами. При отладке выполняется трансляция программ ассемблера, их объединение и загрузка в моделируемую память, моделирование процесса работы программы для микроЭВМ, формирование (редактирование)

необходимых текстовых файлов и сервисное обслуживание.

С помощью директив пользователя с дисплея обеспечиваются все возможности интерактивного режима работы — динамическое управление процессами трансляции, загрузки и исполнения программ микроЭВМ; реализуются необходимые процедуры взаимодействия с отлаживаемой программой, используемые программистом при работе за пультом реальной микроЭВМ; предусматриваются режимы диалога между пользователем и ЭВМ как по инициативе пользователя, так и по инициативе ЭВМ; осуществляется синтаксический и семантический контроль запросов (ответов) пользователей в соответствии со словарем языка общения.

Применение кросс-средств позволяет пользователю провести на начальных этапах технического проектирования МП-устройств СПИ трансляцию, сборку и частичную отладку ПО без использования РОК и прототипа устройства. Следует отметить, что моделирующие программы различных производителей значительно отличаются по своим возможностям, однако ни одна из них не может полностью заменить отладку программы на самой микроЭВМ, так как специфические временные соотношения и условия внешнего окружения аппаратуры микроЭВМ невозможно смоделировать полностью. Поэтому речь может идти только о частичной отладке. Окончательная же отладка программ должна вестись на РОК и (или) прототипе МП-устройства СПИ.

### Применение языков высокого уровня

Стремление ускорить процесс программирования привело к появлению в настоящее время множества языков высокого уровня для МП-систем. Наибольшее распространение получили языки высокого уровня для МП, построенные на базе языка PL/1 : PL/M, PL/Z, PL/μs, PL/W и др. В последнее время все чаще пользуются стандартными языками БЕЙСИК, ФОРТРАН, КОБОЛ и ПАСКАЛЬ. Предпочтение отдается языку ПАСКАЛЬ, обладающему мощными средствами описания структур данных и организации процедур.

Другим подходом является разработка простых языков высокого уровня, ориентированных на узкую область применения. Примером специализированного языка для программирования задач реального масштаба времени яв-

ляется язык CS, который использовался при разработке ПО сетевого протокола по рекомендации МККТТ *X*.25. Преимуществом этой группы языков, разрабатываемых непосредственными пользователями МП, является простота и приспособленность к конкретной области.

При использовании языков высокого уровня при программировании МП-устройств сокращается время написания программ в 5...10 раз; уменьшаются затраты на программирование; при переходе от одного МПК БИС к другому не нужно составлять программы заново. Программы на языках высокого уровня обладают свойством самодокументирования. Если в ассемблерной программе почти каждую строку приходится сопровождать комментарием, то операторы языков высокого уровня почти не нуждаются в них; процедуры проверки и отладки прикладных программ на языках высокого уровня занимают гораздо меньше времени по сравнению с программами на языке ассемблера; легче осуществляется сопровождение программ.

Необходимо отметить также и негативные моменты использования языков высокого уровня. Порождаемые транслятором объектные программы менее эффективны, чем эквивалентные программы, составленные на ассемблере. По средним оценкам длина прикладных программ увеличивается на 20...50 %. При использовании языка высокого уровня трудно выдерживать жесткие временны́е ограничения в операциях ввода—вывода и манипулировать данными, формат которых не предусмотрен соглашениями языка.

Возможны три основных случая использования языков высокого уровня в МП-системах:

1) рабочие программы хранятся в памяти системы на языке высокого уровня, отдельные операторы языка переводятся в машинные коды непосредственно в процессе работы;

2) язык высокого уровня является входным для МП-системы, полученная программа компилируется резидентным транслятором, а затем выполняется;

3) язык высокого уровня является лишь средством подготовки программ, которые затем транслируются на большой ЭВМ в машинный язык МП, и полученная программа затем переносится в МП-систему.

В первом случае в МП системе постоянно должна находиться специальная *программа-интерпретатор*, переводящая отдельные предложения на языке высокого уровня

в машинные коды. Интерпретация — это достаточно трудоемкий процесс, поэтому решение задачи в системах такого типа идет во много раз медленнее, чем при использовании машинных кодов. К достоинствам таких систем относится компактность записи программ, в связи с чем уменьшается требуемый объем оперативной памяти. Интерпретаторы применяются обычно в МП-системах, предназначенных для решения научно-технических задач, в которых основным требованием является снижение трудоемкости составления и отладки программ. В интерпретирующих системах применяются языки БЕЙСИК, APL и упрощенные версии ФОРТРАНА.

Во втором случае требуется значительный объем памяти микроЭВМ для хранения *резидентного транслятора.* Наибольшее применение этот подход находит в редакционно-отладочных комплексах, однако современные МП-системы очень часто имеют необходимый для транслятора объем памяти.

Третий случай — использование языков высокого уровня с последующей трансляцией на большой ЭВМ — применяется достаточно широко.

### Документирование программного обеспечения

Для снижения стоимости программных систем и повышения производительности труда программистов широко используются методы программирования, регламентирующее написание программ независимо от языка, операционной системы, ЭВМ и решаемой задачи. Такие методы получили название технологии программирования. Каждая из существующих технологий, как правило, включает методы построения программного обеспечения, его документирования, написания, отладки и внесения изменений.

При сравнении технологий программирования предпочтение целесообразно отдать той, для которой документация является побочным продуктом процесса разработки программ, не требует усилий программиста, рождается постепенно в процессе разработки ПО, является полной, наглядна, в тонкостях отражает все особенности программ, оформлена стандартным образом. Такие достоинства, в частности, присущи Q-технологии структурного программирования.

С точки зрения общего описания документация представляет собой иерархию книг, которые связаны между со-

бой посредством диаграмм, задающих иерархию связи и сборки. Книги объединены по уровням иерархии. Каждая книга описывает один программный модуль ПО. Для организации ссылок на модуль используется номер уровня иерархии и номер книги с описанием модуля внутри уровня (например, номер модуля — 4.2: это значит, что модуль принадлежит четвертому уровню иерархии и его порядковый номер среди модулей четвертого уровня — 2).

Каждая книга описания модуля состоит из предисловия, структуры, описания данных, комментариев.

Предисловие включает наименование модуля, описание входной и выходной информации, перечень ссылок на диаграммы связи и сборки, где присутствует данный модуль, содержательное описание алгоритма, включающее определение всех требований спецификаций данного модуля, т. е. что́ должен делать данный модуль, классификацию модуля с точки зрения процесса тестирования и перечень некоторых примеров со ссылкой на их описание.

Записи, определяющие функции модуля, которые реализуются программным модулем более низкого уровня, обводят в рамку с указанием имени детализирующего модуля.

Структура отражает последовательность детализации алгоритма. Для организации ссылок описание структуры разбивается на зоны, каждая из которых имеет свои координаты, задающие номер листа описания, идентификатор строки, номер столбца. В каждой зоне выделяются поля для стилизованного изображения стандартной структуры, записи имени блока предыдущего уровня детализации, записи координат зоны, в которой расположена детализирующая структура, записи ссылки на описание входной информации, записи ссылки на описание выходной информации для данной зоны. Объем описания структуры ограничен объемом модуля, который не превышает обычно 50...60 операторов.

Описание данных включает индекс данных (переменная, массив, файл); идентификатор, используемый в программе; описание размерности; категорию данных (элемент данных первоначально определен этим модулем, или может быть модифицирован этим модулем, или используется этим модулем); содержательное описание данных; дополнительные сведения (вхождение описываемой информационной единицы, область изменений и т. д.).

Ссылка на описание данных содержит номер листа и номер строки описания данных.

Комментарии описывают диалектику текста программы и содержание этапов тестирования данного модуля.

Для описания стандартных схем детализации в Q-технологии вводятся понятия *класса работ* и *модификаторы.* Выделяют следующие классы работ:

B — вычислительный — это задачи, имеющие целью расчет по формулам;

C — формирования — это формирование констант, векторов, массивов и т. д. с заданными свойствами;

V — выделения — это задачи поиска некоторых элементов множества, обладающих определенными свойствами;

O — обеспечения — это задачи управления, характерной чертой которых является обеспечение какого-либо действия до поступления отменяющего сигнала.

Тип стандартной схемы для детализации определяется модификатором, указывающим на однократность и многократность выполнения работ. Модификатор и работа определяют схему, которая детализирует данную зону.

Полный пакет документации на программу должен также содержать *тест-программы,* которые оформляются отдельным пакетом книг, состоящим из описания тест-программ, которое включает: содержательное название программы, исходя из цели ее функционирования, с указанием имен проверяемых модулей; перечень функций данной тест-программы (что тестируется); способ задания входных условий; текст тест-программы или ссылку на его местоположения; входную информацию; ожидаемые результаты работы тест-программы; дневник работы тест-программы, в которой отражены анализ получаемых результатов функционирования тест-программы, возможная корректировка тестируемой программы и т. д.

Пакет книг тестирования располагается после последнего уровня иерархии.

Диаграммы связи и сборки представляют собой иерархическую блок-схему и задают иерархию связи и сборки книг на модули. Диаграммы отражают структуру функций ПО и иерархические связи между ними. Поскольку число диаграмм не ограничено, то каждая диаграмма имеет идентификатор и порядковый номер. Имея диаграммы, легко найти нужный уровень информации о программе и соответствующие документы.

## Тестирование программного обеспечения

Одним из основных факторов, определяющих стоимость и уровень бездефектности ПО микропроцессорной системы, является рациональная организация системы тестирования, предназначенной для выявления наличия ошибок или убедительной демонстрации отсутствия определенных ошибок, поскольку, как указывал Дийкстра, «тестирование выявляет только наличие ошибок, но не отсутствие ошибок».

Процесс тестирования как один из этапов разработки ПО, занимает по оценкам специалистов от 30 до 50 % общего времени разработки, поэтому все большее значение приобретают вопросы его организации: планирование тестирования; разработка тестовых средств. *Средства тестирования* — это совокупность тест-программ и инструкций, определяющих методологию процесса тестирования.

Тест-программой может быть либо специальная программа, написанная для испытания одного или нескольких (одновременно) элементов ПО, либо непосредственно элемент ПО, контролируемый на определенном множестве входных данных при тех или иных условиях. Чтобы помочь программисту организовать процесс тестирования, сделать его планомерным, формализованным, существуют инструкции по тестированию, охватывающие вопросы выбора метода тестирования, планирования его, определение объемов испытаний, классификацию комплекса программ ПО, тестов, тестовых данных, разработку стандартных схем тестирования программных модулей, анализ получаемых результатов. Иными словами, совокупность инструкций — это методология тестирования, на основе которой с помощью тест-программ достигается цель тестирования — выявление наличия ошибок.

Стратегия тестирования обычно создается на основе одного или двух методов: либо традиционного — снизу вверх, либо более современного — тестирование сверху вниз.

*Тестирование снизу вверх* предполагает первоначальное написание и проверку модулей самого нижнего уровня. Затем программируются и тестируются элементы более высокого уровня и т. д. до тех пор, пока не будет завершено написание всей программы. Этот метод не дает возможности выявлять серьезные ошибки в ПО почти до момента окончания разработки проекта. Также при каждом новом тестировании элементов различного уровня

требуются новые тестовые программы, тестовые данные, что приводит к большому объему работ по программированию.

*Тестирование сверху вниз* является одним из этапов процесса проектирования сверху вниз. При таком методе тестирования ПО вначале имеется основная программа, а незапрограммированные модули низкого уровня заменяются имитирующими их программами — заглушками. Такая программа может быть испытана в отсутствии вызываемых программ и даже без каких-либо данных.

Установив работоспособность основной программы, начинают наращивать саму систему, добавляя по одному новому модулю более низкого уровня и проверяя его отдельно.

Выбор метода тестирования определяется спецификой конкретного проекта. Тестирование сверху вниз имеет ряд преимуществ перед тестированием по восходящему способу, так как в этом случае стержневая логика программы тестируется на раннем этапе и повторяется многократно с добавлением новых модулей; тестирование оказывается распределенным во все время разработки ПО; возможно получаются результаты раньше, и пользователей можно ознакомить с предварительной версией проекта; упрощается отладка (легче идентифицировать ошибку); обеспечиваются естественные условия испытаний.

В ряде случаев фиктивные модули могут оказаться такими же сложными, как и настоящие. Поэтому на практике редко используют тот или иной метод в чистом виде.

Тестирование ПО можно разделить на три основных этапа.

*Тестирование спецификаций* предполагает проверку полноты и ясности спецификаций, правильности сформулированных требований. Результат этапа — уточнение и изменение проектной документации.

*Тестирование отдельных модулей* позволяет произвести отладку конкретного модуля. Подготовленный к тестированию модуль включается в итоговую программу способом сверху—вниз, заменяя при этом используемую ранее вместо него заглушку. Результатом этапа может явиться изменение алгоритма (программы) тестируемого модуля.

*Тестирование межмодульных связей* ставит своей целью проверку правильности передачи информации от модуля к модулю. В процессе тестирования производится

стыковка отлаженных программных модулей и проверяется логика функционирования ПО.

Планирование тестирования включает ряд предварительных классификаций.

*Классификация отладочных данных* предназначена для устранения дублирования испытаний; для этого все множество входной информации разделяют на классы и проверяют программы на одном элементе из каждого класса. Входные данные объединяются в классы по следующему признаку: их действие на входе того или иного модуля вызывает одинаковый эффект. Так, например, можно выделить класс данных, соответствующих нормальному режиму работы; класс, соответствующий граничным условиям; класс ошибочных значений и т. п.

*Классификация модулей* позволяет программисту по принадлежности модуля к тому или иному классу в кратчайший срок определить: как необходимо тестировать данный модуль, что именно проверять при тестировании, какие отладочные данные необходимо подготовить для тестирования. Например, множество модулей, составляющих программу, можно разбить на следующие классы:

$\{M^{Ф}\}$ — класс функциональных модулей. $M_i \in \{M^{Ф}\}$ — модуль, выполняющий определенную функцию, не связанную с обработкой или анализом входной информации. Результатом работы таких модулей может быть запись слова в буфер ввода, вывод слова из буфера вывода и т. п.;

$\{M^{п}\}$ — класс преобразующих модулей. $M_i \in \{M^{п}\}$ — модуль, находящийся на нижнем уровне иерархии и выполняющий действия, связанные с непосредственным преобразованием входной информации (например, исключение нуля после пяти подряд идущих единиц при обработке принятого информационного слова);

$\{M^{Фп}\}$ — класс функционально-преобразующих модулей. $M_i \in \{M^{Фп}\}$ — модуль, выполняющий действия, косвенно связанные с преобразованием информации (посредством вызова модулей, принадлежащих $\{M^{п}\}$), или с преобразованием входной информации, но подлежащие дальнейшей детализации;

$\{M^{э}\}$ — класс элементарных модулей. $M_i \in \{M^{э}\}$ — модули, не имеющие условных операторов, т. е. такие модули, не зависящие от входных условий.

*Классификация тест-программ* производится в соответствии с типом ошибок, которые выявляются при работе данного теста. С этой целью выделяют категории оши-

бок, наиболее часто допускаемых при разработке ПО микропроцессора, рассматриваемого в качестве элемента системы реального времени. Это прежде всего: логические ошибки — ошибки несоответствия программы решаемой задаче; ошибки перегрузки, возникающие при условии перегрузки внутренних таблиц или буферов: временны́е ошибки — логические ошибки, которые нельзя легко повторить, т. е. такие ошибки, которые зависят от временны́х режимов или от совпадающих сочетаний событий внутри программы; ошибки быстродействия и емкости, когда программа может выдавать правильный результат, но не укладывается в ограничения по времени выполнения или (и) объему памяти; ошибки в документации.

В соответствии с выделенными категориями ошибок и спецификой разрабатываемого ПО выделяют классы тестов. Например: $\{Т^{Ф}\}$ — класс функциональных тестов, выявляющих логические ошибки в программе и ошибки перегрузки; $\{Т^{п}\}$ — класс преобразующих тестов, выявляющих правильность передачи и преобразования информации; $\{Т^{в}\}$ — класс временны́х тестов, выявляющих ошибки временны́е, быстродействия и емкости; $\{Т^{д}\}$ — класс тестов по документации, направленных на выявление ошибок в документации, тесты представляют собой инструкции по ручному тестированию, проверке спецификаций технического задания.

*Стандартная схема тестирования* предполагает, что модули, принадлежащие одному классу, будут тестироваться по одинаковой схеме. Так, например, при тестировании модулей классов:

$\{М^{Ф}\}$ проверяется правильность выбора ветви программы в соответствии с входными условиями;

$\{М^{п}\}$ проверяется правильность обработки или анализа входной информации в зависимосит от тех или иных входных условий;

$\{М^{Фп}\}$ выделяется два этапа. На первом этапе тестирование производится аналогично испытанию преобразующих модулей, когда все детализирующие модули отлажены, на втором — аналогично испытанию функциональных модулей. Для испытания достаточен лишь тот объем информации, на котором проверяется согласованность работы данного модуля с детализирующими его;

$\{М^{э}\}$ проверяется правильность включения в работу данного модуля и соответствие выполняемой им функции, возложенной на него. Модуль класса $\{М^{э}\}$ нецелесообразно тестировать отдельно.

Применение вышеизложенных спецификаций облегчает процесс планирования тестирования, делает его более типовым, стандартным, подчиненным определенной схеме, определяет тип данных, которые необходимо варьировать при испытаниях модуля. А использование документации определяет диапазон изменения этих данных, а также, какие конкретные значения необходимо взять в соответствии с требуемым качеством тестирования и ограничением на стоимостные и временные ресурсы.

Результат данного планирования сводят в таблицу, в которой для каждого тестируемого модуля указано имя теста, его тип, значение входных условий и входной информации. Для некоторых модулей потребуется несколько наборов тестов, в других случаях один тест проверяет одновременно несколько модулей.

В процессе планирования также определяют, какие заглушки необходимы для тестирования модулей и какие требования необходимо предъявлять к заглушкам. Здесь возможны случаи, когда программирование заглушки сложно, и поэтому целесообразно использовать сам модуль. Результат такого шага также целесообразно отразить в соответствующей таблице.

При планировании тестирования межмодульных связей используют листы описания данных между модулями и планируют тест, проверяющий правильность передачи данных от одного модуля к другому. Результат этого этапа также отражается таблицей.

При оформлении тест-программ руководствуются теми же правилами, что и при документировании основного ПО.

# ГЛАВА 5
# ВОПРОСЫ ПРОГРАММНОЙ РЕАЛИЗАЦИИ НА МИКРОЭВМ ФУНКЦИИ ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ

## § 5.1. ОБЩИЕ СВЕДЕНИЯ

В этой главе рассмотрены некоторые вопросы, связанные с программной реализацией на базе микроЭВМ функции кодовой защиты как относительно наиболее простой и автономной и в то же время присущей каждому узлу СПИ, реализирующему протокол информационного канала.

Функция защиты от ошибок в оконечных и промежуточных пунктах (ОКП и ПРП), оборудованных соответствующими типами связных процессоров, может быть реализована на основе аппаратной логики или в самих связных процессорах программным способом. В большинстве случаев к ОКП и ПРП подсоединяются как источники, так и получатели информации, что обусловливает в ОКП и ПРП одновременное наличие кодеров и декодеров. Поэтому их совместно целесообразно рассматривать как *кодеки,* выполняющие функцию помехоустойчивого кодирования — декодирования.

Функционирование кодеков на уровне «черного ящика» можно описать соответственно операторами кодирования и декодирования. Для описания работы кодеков в промежуточных пунктах введем понятие оператора декодирования—кодирования (кодовой ретрансляции). Таким образом, в общем случае *оператор защиты от ошибок* (ОЗО) включает в себя операторы кодирования, декодирования и кодовой ретрансляции. Операторы декодирования могут реализовываться с обнаружением ошибок (или) стираний, с исправлением ошибок и (или) стираний или с одновременным обнаружением и исправлением ошибок и (или) стираний. Таким образом, если для задания операторов кодирования достаточно определить тип кода и его параметры, то для операторов декодирования этого недостаточно. Операторы декодирования должны задаться дополнительно режимом реализации кода. Следовательно, ОЗО также задается типом кода и режимом его реализации. Тогда на основе одного кода в общем случае можно сформулировать несколько ОЗО с различными показателями качества.

Для более подробного описания кодека необходимо перейти к описанию его *алгоритма защиты от ошибок* (АЗО). В общем случае этот переход не однозначен, так как одному ОЗО может соответствовать множество АЗО, каждый из которых допускает различные способы реализации. Таким образом, процессу построения кодека должен предшествовать выбор ОЗО (тип кода и режим его реализации), алгоритма и способа его реализации.

Для организации связи между ОКП и ПРП современных СПИ и АСУ используются различные каналы, которые могут обладать существенно различными, а зачастую и нестационарными распределениями характеристик. С другой стороны, требования к верности передачи для различных потребителей часто не совпадают. Поэтому не

будут совпадать и в общем случае ОЗО, число которых определит необходимое количество различных кодеков для каждого ОКП и ПРП СПИ.

Следовательно, в общем случае пункты СПИ оборудованы некоторым числом различных кодеков, которые целесообразно рассматривать совместно. Это приводит к понятиям составного и простого кодексов, где *простой кодек* — кодек, обслуживающий один или несколько каналов в соответствии с конкретными ОЗО, *составной кодек* — кодек, объединяющий некоторое число простых кодеков.

Очевидно,при введении понятий простого и составного кодеков будут иметь место и понятия простого и составного ОЗО.

В настоящее время при реализации составных кодеков на основании аппаратной логики обычно каждый из составляющих простых кодеков строится независимо на отдельном оборудовании. В этом случае количество оборудования составного кодека является суммой количеств оборудования составляющих простых кодеков. Развитие СПИ потребует дополнительного оборудования и (или) замены старого оборудования новым, что сопряжено с большими материальными затратами. Поэтому реализация составных кодеков перспективна при использовании универсального оборудования. Алгоритмической универсальностью при жесткой аппаратной структуре обладают программируемые автоматы, какими являются микроЭВМ. Поэтому необходимо в модульной структуре связного процессора с функциональной организацией предусмотреть функциональный модуль, для которого функция защиты от ошибок является основной. Кодеки, реализуемые с помощью такого модуля, будем называть *программируемыми кодеками* (ПК).

Поскольку структура функционального модуля СП жестко определена (см. 2.3) и включает в себя микроЭВМ, локальную память и адаптер связи, процесс проектирования ПК существенно упрощается в сравнении с общей методикой (см. § 3.3).

Этап системного проектирования ПК включает такую последовательность шагов:

1. Определение и обоснование исходных данных и требований к проектируемому ПК. Обоснование критерия оптимизации кодека.

2. Формирование множества допустимых вариантов кодека. Каждый вариант ПК однозначно задается

некоторым конкретным ОЗО и алгоритмом его реализации.

3. Выбор оптимального варианта (из множества допустимых) проектируемого ПК, т. е. выбор оптимального ОЗО и алгоритма его реализации.

4. Оценка требуемого быстродействия кодирующего процессора, т. е. процессора, реализующего АЗО.

5. Выбор универсальной микромашины для организации ПК.

Этап технического проектирования ПК включает проектирование его программного и аппаратного обеспечения. Программное обеспечение ПК содержит программные средства, осуществляющие обработку информации, и программные средства, управляющие этой обработкой. Аппаратное обеспечение ПК включает в себя микромашину и интерфейс с каналами (адаптер).

На первом шаге системного этапа необходимо знать характер ошибок в каналах (модели каналов), допустимые вероятности потерь информации из-за воздействия помех в каналах, критерий оценки качества возможных вариантов проектируемого кодека.

На втором шаге системного этапа решается вопрос формирования множества допустимых вариантов разрабатываемого кодека. Допустимыми следует считать варианты, удовлетворяющие исходным требованиям. Для составного кодека необходимо провести объединение алгоритмов составляющих простых ОЗО. При этом часто бывает неизвестна конкретная микроЭВМ, на которой предстоит реализовать ПК. Поэтому для оценки вариантов ПК целесообразно использовать модель класса микроЭВМ, которая обладает его характерными особенностями.

Оценка требуемого быстродействия процессора для ПК может быть вычислена по выражениям границ, по быстродействию и нагрузке кодирующего процессора.

На последнем шаге системного этапа проектирования ПК из доступного парка микроЭВМ выбирается машина, удовлетворяющая требованиям по быстродействию и памяти с учетом рекомендуемой разрядности.

Таким образом, специфика проектирования ПК, как функционального модуля СП, связана с конкретизацией общего критерия эффективности МПУ СПИ (см. § 4.1) по отношению к реализации функции кодовой защиты,

построением границ по быстродействию и нагрузке кодирующего процессора, объединением алгоритмов составляющих простых ОЗО, оценкой вариантов ПК с использованием, например, модели класса микроЭВМ.

## § 5.2. КРИТЕРИЙ ОЦЕНКИ КАЧЕСТВА ПРОГРАММИРУЕМЫХ КОДЕКОВ

В общем случае ПК характеризуются множеством показателей качества, которые желательно учитывать при выборе оптимальных вариантов кодеков. Для оценки вариантов ПК воспользуемся критерием эффективности МПУ СПИ, представленным в 4.1, конкретизировав его составляющие в соответствии с особенностями ПК.

Как и в общем случае (4.1), множество показателей качества ПК целесообразно разбить на два подмножества: показателей эффективности $K_{ц}$ и показателей качества $K_{т}$. Под *показателями эффективности* понимают показатели, характеризующие степень выполнения основного назначения кодека, а под *показателями качества* — показатели, характеризующие технико-экономические качества кодека при выполнении его основного назначения. Тогда задача выбора оптимального варианта ПК формулируется чаще всего в следующем виде: оптимальным из множества вариантов является тот, который обладает лучшими показателями качества $K_{т}$ при фиксированных значениях показателей эффективности $K_{ц}$.

С учетом основного назначения кодеков в качестве показателя эффективности целесообразно принять *вероятность $P_{п}$ потери информации* из-за воздействия помех в каналах связи.

Так как структура ПК, как функционального модуля СП, фиксирована, то нет необходимости сравнивать различные варианты ЦК по показателю сложности $S$ (см. § 4.1).

С другой стороны, поскольку ПК может быть построен на основе различных ОЗО, то важным специфическим показателем ПК является *избыточность передачи по каналу* $\gamma$. Показатель $\gamma$ зависит от метода введения избыточности и определяет относительное увеличение длины передаваемого по каналу сигнала за счет помехоустойчивого кодирования.

Поскольку даже один и тот же ОЗО допускает различные алгоритмы реализации, то существенными показателями качества при сравнении вариантов ПК являют-

ся *время обработки на блок T* и требуемый *объем памяти при обработке блоков V*.

Таким образом, в общем случае показатели эффективности и качества составного ПК могут быть записаны в виде комплексных показателей

$$K_{\text{ц}} = K_{\text{ц}}(P_{\text{п}_1}, P_{\text{п}_2}, \ldots, P_{\text{п}_i}, \ldots, P_{\text{п}_m});$$

$$K_T = K_T(\gamma_1, T_1, V_1, \ldots, \gamma_i, T_i, V_i, \ldots, \gamma_m, T_m, V_m),$$

где $m$ — число простых кодеков в составе составного; $P_{\text{п}i}$ — вероятность потери информации из-за воздействия помех для $i$-го простого кодека; $\gamma_i$, $T_i$, $V_i$ — показатели качества $i$-го простого кодека.

Поскольку зафиксирован комплексный показатель эффективности, то необходимо определить допустимые вероятности потерь информации $P^{D}_{\text{п}_i}$ для каждого из простых кодеков. Очевидно, что превышение допустимой вероятности для одного из простых кодеков не может быть скомпенсировано уменьшением вероятности для остальных. Поэтому на вероятности $P^{D}_{\text{п}_i}$, $i=\overline{1, m}$ накладывается условие самостоятельной значимости. Тогда комплексный показатель эффективности в функциональном отношении вырождается в набор своих аргументов $K_{\text{ц}} = (P_{\text{п}1}, \ldots, P_{\text{п}i}, \ldots, P_{\text{п}m})$. Таким образом, при разработке составного ПК необходимо обеспечить вероятности потерь не выше допустимых для каждого из простых кодеков.

Из трех выделенных параметров только $V$ составного ПК не является в общем случае суммой $V_i$ простых кодеков из-за объединения программ. Но учет этого фактора при выборе оптимального варианта на этапе системного проектирования ведет к неоправданным усложнениям. Поэтому вполне разумно считать, что оптимальный вариант составного ПК имеет место при оптимальном построении каждого из простых кодеков. Следовательно, в качестве аргументов комплексного показателя качества $K_T$ составного ПК можно рассматривать комплексные показатели качества простых кодеков $K_i$: $K_T = (K_1, \ldots, K_i, \ldots, K_m)$, где $K_i = f_i(\gamma_i, T_i, V_i)$; $i=\overline{1, m}$.

Тогда с учетом отрицательного инградиента для показателей качества *критерий оптимизации составного ПК* может быть записан следующим образом:

$$\forall_i K_i = \max_{\{B_{ij}\}} f_i(\gamma_{ij}, T_{ij}, V_{ij}) \text{ при } P_{\text{п}\,ij} \leqslant P^{\text{д}}_{\text{п}_i},$$

где $\{B_{ij}\}$ — множество допустимых вариантов для $i$-го простого кодека, а $P_{\text{п}\,ij}$, $\gamma_{ij}$, $T_{ij}$, $V_{ij}$ — показатели $j$-го допустимого варианта $i$-го простого кодека.

Поскольку вывод математической модели, описывающей взаимосвязь показателей качества ПК в общем случае отсутствует, то в соответствии с 4.1 для формирования интегрального показателя качества $K_i$ ПК целесообразно брать *средневзвешенный геометрический показатель,* записываемый в виде

$$K_{ij} = (\gamma_{i\min}/\gamma_{ij})^{\nu_{i\gamma}} (T_{i\min}/T_{ij})^{\nu_{iT}} (V_{i\min}/V_{ij})^{\nu_{iV}},$$

где $\gamma_{i\min}$, $T_{i\min}$, $V_{i\min}$ — минимальные значения соответствующих показателей качества по всем допустимым вариантам $i$-го простого кодека; $\nu_{\gamma i}$, $\nu_{iT}$, $\nu_{iV}$ — весовые коэффициенты приоритетности соответствующих показателей, которые чаще всего определяются в результате эксперимента. При этом

$$\forall_i 0 \leqslant \{\nu_{i\gamma}, \nu_{iT}, \nu_{iV}\} \text{ и } (\nu_{i\gamma} + \nu_{iT} + \nu_{iV}) = 1.$$

Отметим физический смысл выражения для интегрального показателя: $1/(TV)$ показывает, сколько сообщений (блоков) в единицу времени приходится на единицу занимаемой памяти. Следовательно, $1/(TV)$ можно назвать пропускной способность занимаемой памяти.

Тогда $(\gamma_{\min} T_{\min} V_{\min})/(\gamma TV)$ — относительная полезная пропускная способность занимаемой памяти.

В зависимости от весовых коэффициентов показателей качества, входящих в интегральный критерий $K_{ij}$, можно строить ПК, оптимизируя их по одному из следующих параметров: избыточности, пропускной способности, затратам памяти.

В отличие от выражения для интегрального критерия эффективности МПУ СПИ, представленного в § 4.1, в выражении $K_{ij}$ отсутствует относительный показатель стоимости варианта ПК. Это обусловлено следующими соображениями. Стоимость реализации готовой программы определяется стоимостью времени занятия процессора и стоимостью занимаемого блока памяти. Поскольку для ПК, как функционального модуля СП, процессор предоставлен в постоянное пользование, сравнительная стоимость варианта ПК будет определяться стоимостью

занимаемого блока памяти. А в интегральный критерий входит относительный вес блока памяти. Следовательно, вариант ПК, оптимальный по затратам памяти, можно считать оптимальным по стоимости.

## § 5.3. ГРАНИЦЫ ПО БЫСТРОДЕЙСТВИЮ И НАГРУЗКЕ КОДИРУЮЩИХ ПРОЦЕССОРОВ

При проектировании составных ПК представляет интерес оценка требуемого быстродействия составного ПК для поддержания заданной нагрузки либо оценка максимальной нагрузки, которую может принять на себя составной ПК без потери информации. Под *быстродействием* составного ПК понимаем быстродействие его процессора, измеряемое в циклах памяти в секунду (ц/с).

Для определения границ по быстродействию и нагрузке кодирующих процессоров предположим, что передача информации ведется помехоустойчивыми блочными $(n, k)$-кодами непрерывно, т. е. на достаточно большом промежутке времени. Это означает, что при недостаточном быстродействии кодирующего процессора очередь на обслуживание будет расти и будут иметь место потери информации.

Ввод (вывод) блоков в (из) кодирующий(его) процессор(а) осуществляется посылками длины $l_{вв}$ ($l_в$) двоичных символов. Обозначим через $Q$ быстродействие кодирующего процессора, ц/с; $N_{вв}$ — требуемое число циклов для ввода одной посылки; $N_в$ — требуемое число циклов для вывода одной посылки; $v$ — скорость модуляции в канале (нагрузка); $y$ — минимальный интервал между поступающими из канала блоками в двоичных символах; П — пропускную способность кодирующего процессора, блок/с; $T$ — время обработки, ц/блок.

Введем $y_{вв}$ и $y_в$ и определим их как $y_{вв}=k$, $y_в=n$ при кодировании, $y_{вв}=n$, $y_в=k$ при декодировании, $y_{вв}=y_в=n$ при ретрансляции. Тогда скорость поступления информации в блоках за секунду будет $v/(y_{вв}+y)$.

Чтобы не происходили потери информации при непрерывном ее поступлении, должно выполняться неравенство

$$v(y_{вв}+y)^{-1} \leqslant \text{П}. \tag{5.1}$$

Отсюда получаем выражение для определения требуемого быстродействия кодирующего процессора

$$Q \geqslant v(y_{вв}+y)^{-1}\left(T+\left\lceil\frac{y_{bb}}{l_{вв}}\right\rceil N_{вв}+\left\lceil\frac{y_в}{l_в}\right\rceil N_в\right). \tag{5.2}$$

Здесь $\lceil x \rceil$ — ближайшее целое не меньше $x$. Правая часть неравенства (5.2) определяет границу по требуемому быстродействию кодирующего процесора. Отсюда можно определить и максимальную допустимую нагрузку $v$, которую может принять на себя кодирующий процессор с данным быстродействием $Q$ без потери информации.

В тех случаях, когда передача информации ведется сеансами, за счет использования буферной памяти можно снизить границу по быстродействию или повысить границу по нагрузке кодирующего процессора.

Пусть используется внешняя буферная память и неравенство (5.1) является обратным. Тогда выражение $[v(y_{вв}+y)^{-1}—П]$ представляет собой скорость заполнения буферной памяти в блоках за секунду. Требуемый объем буферной памяти $V_б$ в блоках определяется из соотношения

$$V_б = [v(y_{вв} + y)^{-1} — П] T_{сс}, \tag{5.3}$$

где $T_{сс}$ — длительность сеанса связи в секундах.

Из (5.3) после преобразования получаем выражение для границы по быстродействию кодирующего процессора в виде

$$Q \geqslant \left(T + \left\lceil \frac{y_{вв}}{l_{вв}} \right\rceil N_{вв} + \left\lceil \frac{y_в}{l_в} \right\rceil N_в\right)\left(\frac{v}{y_{вв} + y} - \frac{V_б}{T_{сс}}\right). \tag{5.4}$$

Очевидно, граница по быстродействию снижается с уменьшением длительности сеанса связи и (или) увеличением объема буферной памяти. Аналогично, для верхней границы по нагрузке

$$v \leqslant \left\{\frac{Q}{\left(T + \left\lceil \frac{y_{вв}}{l_{вв}} \right\rceil N_{вв} + \left\lceil \frac{y_в}{l_в} \right\rceil N_в\right)} + \frac{V_б}{T_{сс}}\right\}(y_{вв} + y).$$

Теперь рассмотрим случай, когда в качестве буферной применяется оперативная память. Чтобы не было потерь информации, операциям ввода следует придать высший приоритет. Тогда доступные для обработки циклы кодирующего процессора с учетом высшего приоритета операций ввода определяются как

$$Q — v(y_{вв} + y)^{-1}\left\lceil \frac{V_{вв}}{l_{вв}} \right\rceil N_{вв}.$$

Пропускная способность кодирующего процессора будет

$$\Pi = \left(Q - \frac{v}{y_{вв} + y}\left\lceil \frac{y_{вв}}{l_{вв}} \right\rceil N_{вв}\right)\left(T + \left\lceil \frac{y_в}{l_в} \right\rceil N_в\right)^{-1}.$$

Выражение для определения требуемого объема буферной памяти принимает в этом случае вид

$$V_б = \left(\frac{v}{y_{вв} + y} - \frac{Q - v\,(y_{вв} - y)^{-1}\left\lceil \frac{y_{вв}}{l_{вв}} \right\rceil N_{вв}}{T + \left\lceil \frac{y_в}{l_в} \right\rceil N_в}\right) T_{сс}.$$

Отсюда после преобразований получим выражение для границ по быстродействию и соответственно по нагрузке кодирующего процессора:

$$Q \geqslant \left(\frac{v}{y_{вв} + y} - \frac{V_б}{T_{сс}}\right)\left(T + \left\lceil \frac{y_в}{l_в} \right\rceil N_в\right) + \frac{v}{y_{вв} + y}\left\lceil \frac{y_{вв}}{l_{вв}} \right\rceil N_{вв};$$

$$v < (y_{вв} + y)\left(\frac{V_б}{T_{сс}} + \frac{Q - v\,(y_{вв} + y)^{-1}\,[y_{вв}/l_{вв}]\,N_{вв}}{T + [y_в/l_в]\,N_в}\right).$$

Применение внешней и буферной памяти для согласования нагрузки с пропускной способностью кодирующего процессора может чаще всего иметь место при работе без обратной связи. Накопленные в буферной памяти блоки будут обрабатываться после окончания сеанса связи. Необходимое число циклов для этой обработки будет характеризовать мертвое время кодирующего процессора, которое может оказаться определяющим фактором при расчете требуемого быстродействия. По нему можно определить $V_б$ и далее в соответствии с формулами (5.4) или (5.5) — $Q$. При равенстве мертвого времени нулю необходимо пользоваться формулой (5.2).

Сказанное выше в этом подразделе относится к случаю монопольного обслуживания каналов кодирующим процессором, который может выполнять при этом роль как простого, так и составного ПК. В случае составного ПК расчеты требуемого быстродействия необходимо проводить с учетом показателей того из простых кодеков, реализация которого требует максимального быстродействия.

Теперь рассмотрим случай коллективного обслуживания каналов кодирующим процессором.

Монопольное и коллективное обслуживания рассматриваются как возможные варианты обслуживания в ре-

жиме разделения времени. Под *монопольным обслуживанием каналов* понимаем такое обслуживание, при котором отношения времен передачи блоков по каналам к длительности выделяемых квантов на обслуживание каналов близки к нулю. *Коллективное обслуживание* каналов имеет место, когда длительности времен передачи блоков и выделяемых квантов на обслуживание соизмеримы.

Пропускная способность кодирующего процессора при коллективном обслуживании каналов определится по формуле

$$\Pi = QH\left[\sum_{i=1}^{H}\left(T_i + \left[y_{\text{вв}i}/l_{\text{вв}i}\right]N_{\text{вв}i} + \left[y_{\text{в}i}/l_{\text{в}i}\right]N_{\text{в}i}\right)\right]^{-1},$$

где $H$ — число обслуживаемых кодирующим процессором каналов.

Отметим, что в данном случае с точки зрения определения границ необходимо предположить одновременную работу всех каналов в течение длительного промежутка времени. При этом скорость поступления информации в блоках за секунду определится как

$$\left(H\sum_{i=1}^{H}v_i\right)\Big/\sum_{i=1}^{H}\left(y_{\text{вв}i} + y_i\right).$$

Из условия $\left(H\sum_{i=1}^{H}v_i\right)\Big/\sum_{i=1}^{H}\left(y_{\text{вв}i}+y_i\right) \leqslant \Pi$ получаем выражения для определения границ по быстродействию и по суммарной нагрузке кодирующего процессора

$$Q \geqslant \left(\sum_{i=1}^{H}v_i \Big/ \sum_{i=1}^{H}\left(y_{\text{вв}} + y_{\text{в}}\right)\right)\sum_{i=1}^{H}\left(T_i + \left[y_{\text{вв}i}/l_{\text{вв}i}\right]N_{\text{вв}i} + \left[y_{\text{в}i}/l_{\text{в}i}\right]N_{\text{в}i}\right).$$

$$\sum_{i=1}^{H}v_i < \sum_{i=1}^{H}\left(y_{\text{вв}i} + y_i\right)\Pi/H.$$

Представленные формулы позволяют оценить на этапе системного проектирования требуемое быстродействие кодирующего процессора. Расчеты проводятся по показателям выбранного оптимального варианта ПК. По полученной оценке требуемого быстродействия на за-

ключительном шаге этапа системного проектирования ПК выбирается конкретная микромашина, на основе которой строится кодирующий процессор. При этом необходимо учитывать целесообразную разрядность и необходимый объем памяти, которые получаются в результате исследования на модели микроЭВМ.

## § 5.4. ОБЪЕДИНЕНИЕ АЛГОРИТМОВ И ЭКОНОМИЯ ПАМЯТИ ПРИ РЕАЛИЗАЦИИ ПРОГРАММИРУЕМЫХ КОДЕКОВ

При проектировании математического обеспечения составных ПК возникает задача объединения алгоритмов составляющих ОЗО в составной алгоритм, отражающий специфику решения в частных ситуациях. При этом возникает сопутствующая задача минимизации требуемого объема памяти для реализации алгоритмов, составленных из отдельных кусков.

Объединение алгоритмов осуществляется в основном по способам Карпа и Лазарева-Пийль. Однако эти способы можно усовершенствовать, как показано ниже.

### Способы объединения двух алгоритмов с использованием частичных матричных схем

Суть предложенного Р. М. Карпом способа объединения двух алгоритмов, представленных в виде блок-схем, с использованием *матричных схем алгоритмов* (МСА), когда в объединенном алгоритме каждый оператор встречается только один раз, заключается в следующем.

Предположим, что необходимо объединить два алгоритма А и В, заданных в виде блок-схем и имеющих общие операторы. Для указания того, какой из двух вариантов составного алгоритма должен реализовываться, вводится двоичная переменная $\Sigma$.

Пусть алгоритму А соответствует значение $\Sigma=1$, а алгоритму В — значение $\overline{\Sigma}=1$.

Процедура объединения состоит в следующем:

1. По исходным блок-схемам строятся соответствующие матрицы (МСА) — $\|a_{ij}\|$ и $\|b_{ij}\|$, элементы которых представляют собой булевые выражения, определяющие условия перехода из оператора $i$ в оператор $j$.

2. Строится матрица $\|m_{ij}\|$ составного алгоритма, элементы которой $m_{ij}=\Sigma a_{ij}\vee\overline{\Sigma}b_{ij}$.

3. Матрица, полученная на втором этапе, упрощается:

а) если $m_{ij}=\Sigma\vee\overline{\Sigma}$, то $m_{ji}=1$;

б) если $\Sigma$ (или $\overline{\Sigma}$) не встречается в некоторой строке, то удаляются все появления $\overline{\Sigma}$ (или $\Sigma$) в этой строке.

Полученная после этого матрица (МСА) является полным представлением составного алгоритма и от нее можно легко перейти к блок-схеме. В рассмотренной процедуре преобразования алгоритмов количество операций возрастает с ростом произведения $(n_A+q)(n_B-q)$, где $q$ — число общих операторов, а $n_A$ и $n_B$ — число всех операторов в алгоритмах $A$ и $B$ соответственно. С увеличением $n_A$ и $n_B$ приходится манипулировать матрицами большой размерности, что неудобно, особенно при машинной реализации процедуры.

Матрицы $\|a_{ij}\|$, $\|b_{ij}\|$, $\|m^*_{ij}\|$ имеют соответственно следующие размерности: $(n_A\times n_A)$, $(n_B\times n_B)$, $(n_A+n_B-q)^2$. Количество операций при объединении алгоритмов может быть сокращено за счет использования для преобразования частичных матричных схем алгоритмов (ЧМСА), т. е. таких схем, в которых отражены связи только между частью операторов исходных алгоритмов. Процедура способа с использованием ЧМСА состоит в следующем:

1. По исходным блок-схемам строятся частичные матрицы $\|a^*_{ij}\|$ и $\|b^*_{ij}\|$, отражающие связи только общих операторов. Размерность этих матриц соответственно $(q\times n_A)$ и $(q\times n_B)$.

2. Строится частичная матрица $\|m^*_{ij}\|$ составного алгоритма размерностью $q\times(n_A+n_B-q)$ такая, что

$$m^*_{ij}=\Sigma a^*_{ij}\vee\overline{\Sigma}\,b^*_{ij}.$$

3. Полученная на втором этапе частичная матрица упрощается выделением по возможности дизъюнктивного выражения $(\Sigma\vee\overline{\Sigma})$, и если $m^*_{ij}=\Sigma\vee\overline{\Sigma}$, то $m^*_{ij}=1$.

От окончательной частичной матрицы переходим к частичной блок-схеме составного алгоритма. Далее, принимая поочередно $\Sigma=1$ и $\overline{\Sigma}=1$, приписываем к этой блок-схеме оставшиеся части блок-схем алгоритмов A и B и получаем полную блок-схему составного алгоритма.

Проиллюстрируем процедуру объединения на основе ЧМСА на примере алгоритмов A и B, блок-схемы которых представлены на рис. 5.1, *а*, *б*, а общие операторы

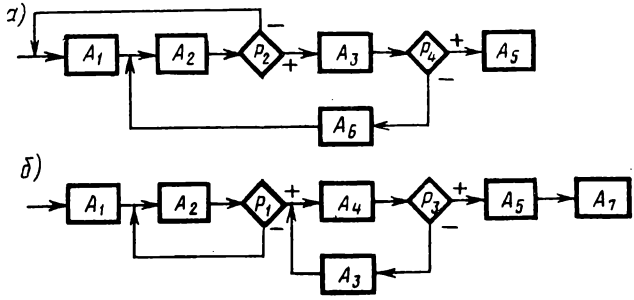



Рис. 5.1. Структурная схема объединяемых алгоритмов

показаны одинаковыми символами. Здесь $q=4$, $n_A=5$ и $n_B=6$.

После выполнения первого этапа процедуры получаем следующие частичные матрицы:

$$\|a^*_{ij}\| = \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array} \begin{array}{c} \begin{array}{ccccc} A_1 & A_2 & A_3 & A_5 & A_6 \end{array} \\ \left\| \begin{array}{ccccc} 0 & 1 & 0 & 0 & 0 \\ \bar{P}_2 & 0 & P_2 & 0 & 0 \\ 0 & 0 & 0 & P_4 & \bar{P}_4 \\ 0 & 0 & 0 & 0 & 0 \end{array} \right\| \end{array};$$

$$\|b^*_{ij}\| = \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array} \begin{array}{c} \begin{array}{cccccc} A_1 & A_2 & A_3 & A_5 & A_4 & A_7 \end{array} \\ \left\| \begin{array}{cccccc} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \bar{P}_1 & 0 & 0 & P_1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right\| \end{array}.$$

Результатом второго этапа является частичная матрица составного алгоритма:

$$\|m^*_{ij}\| = \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_5 \end{array} \begin{array}{c} \begin{array}{ccccccc} A_1 & A_2 & A_3 & A_5 & A_6 & A_4 & A_7 \end{array} \\ \left\| \begin{array}{ccccccc} 0 & \Sigma\wedge\bar{\Sigma} & 0 & 0 & 0 & 0 & 0 \\ \Sigma\bar{P}_2 & \bar{\Sigma}\,\bar{P}_1 & \Sigma P_2 & 0 & 0 & \bar{\Sigma}P_1 & 0 \\ 0 & 0 & 0 & \Sigma P_4 & \Sigma\bar{P}_4 & \bar{\Sigma} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \bar{\Sigma} \end{array} \right\| \end{array}.$$

После замены $(\Sigma\vee\bar{\Sigma})$ можно построить частичную блок-схему, представленную на рис. 5.2 сплошными линиями. Пунктирными линиями показано дополнение до полной блок-схемы составного алгоритма.

Сравнение двух рассмотренных процедур позволяет сделать выводы в пользу способа с использованием ЧМСЛ, а именно: сокращается число операций первого этапа, поскольку строятся только частичные матрицы; сокращается число операций второго этапа, потому что необходимо манипулировать матрицами меньшей размер-

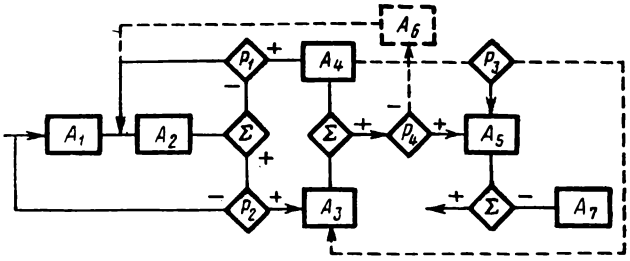



Рис. 5.2. Структурная схема составного алгоритма

ности; сокращается также число операций третьего этапа.

В некоторых случаях может потребоваться получение полной матричной схемы составного алгоритма. Для этого к частичной матричной схеме составного алгоритма следует приписать без изменений оставшиеся части матричных схем обоих алгоритмов. Как показывает анализ, в этом случае способ с ЧМСА выигрывает по сравнению со способом Карпа.

Кроме того, в процедуре Карпа существует неточность, которая проявляется в случаях, подобных рассмотренному примеру, когда общий оператор в одном алгоритме является конечным, а в другом нет (в примере оператор $A_5$). Следуя процедуре Карпа, на третьем этапе в п. *б* пришлось бы убрать из последней строки $\|m_{ij}\|$ символ $\Sigma$, заменяя его на единицу. Тогда в составном алгоритме оператор $A_7$ следовал бы безусловно за оператором $A_5$, что неверно.

### Экономия памяти программируемых кодеков

Проблема минимизации требуемого объема памяти для реализации алгоритмов рассматривается в литературе в двух аспектах — экономии программной памяти и экономии рабочей памяти. При разработке СПИ наряду с ограничениями по памяти команд и оперативной памя-

ти в качестве основных целевых параметров выбираются и времена задержки, что налагает ограничения на времена реализации разрабатываемых алгоритмов и программ ОЗО. В этих условиях возможна задача минимизации требуемого программами защиты от ошибок объема памяти при соблюдении допустимых времен реализации.

В данном параграфе рассматривается задача уменьшения суммарного объема памяти, занимаемого множеством программ при ограничениях на времена реализации каждой из них. При этом предполагается, что в логических схемах отдельных программ имеются общие части. Это условие выполняется во многих случаях, в частности при разработках составных ПК.

Пусть даны логические схемы $n$ программ, для которых соблюдаются допустимые времена реализации $T_i^д$. $i=\overline{1, n}$. С целью уменьшения объема занимаемой памяти общие части могут быть выделены в виде подпрограмм $s_1, s_2, \ldots, s_\lambda$ которые записываются в памяти только один раз.

Включение некоторой подпрограммы в работу программ осуществляется посредством операций, обеспечивающих связь подпрограмм с программами, и операций, обеспечивающих ввод необходимой информации в подпрограмму и вывод результатов. Выполнение этих операций ведет к увеличению времен реализации программ, использующих подпрограммы, и требует некоторого увеличения числа ячеек памяти. Поэтому множество подпрограмм должно быть построено оптимальным образом с учетом временны́х ограничений. Эту задачу можно сформулировать как *задачу двоичного линейного программирования.*

Обозначим $V_j$ объем памяти, занимаемой $j$-общей частью, а через $W_j$ — необходимое число ячеек памяти для включения $s_j$ в работу некоторой из программ. При этом можно представить $W_j=V_j^{(0)}+V_j^{(1)}$, где $V_j^{(0)}$ — часть $V_j$, относящаяся к каждой из программ, использующих $s_j$, а $V_j^{(1)}$ — часть $V_j$, которая относится только к подпрограмме $s_j$.

Очевидно, что для выигрыша в памяти при выделении подпрограмм минимальное число программ $k_j$, которые будут использовать подпрограмму $s_j$, определится из неравенства

$$k_j V_j \geqslant V_j + V_j^{(0)} k_i + V_i^{(1)}. \qquad (5.6)$$

Отсюда $k_j \geqslant (V_j+V_j^{(1)})/(V_j-V_j^{(0)})$ или $k_j=[(V_j+V_j^{(1)})/(V_j-V_j^{(0)})]$. Если $j$-я часть объемом $V_j$ является общей для программ, число которых меньше $k_j$, то выигрыша в памяти от выделения ее в подпрограмму не будет.

Определим бинарную матрицу связности $\|a_{ij}\|$, где элемент $a_{ij}$ равен единице в случае, когда $i$-я программа нуждается в применении $s_j$-й подпрограммы, и нулю — в противном случае.

Через $T_j=\tau_j+\tau_{jo}$ обозначим дополнительное время, на которое увеличивается время реализации программы при однократном обращении к подпрограмме $s_j$. Полное время, на которое увеличивается время реализации $j$-й программы, будет зависеть от частоты $f_{ij}$ ее обращения к $s_j$-й подпрограмме. При этом время $\tau_{jo}$ прибавляется только один раз, а время $\tau_j$ — пропорционально $f_{ij}$.

Двоичные переменные $x_{ij}$ определим следующим образом: $x_{ij}=1$, если обращение $j$-й программы к $s_j$-й подпрограмме целесообразно; $x_{ij}=0$ в противном случае. Число неизвестных $x_{ij}$ равняется числу единиц в матрице связности $\|a_{ij}\|$. Решением задачи является оптимальная матрица связности

$$\|a_{ij}^*\| = \|a_{ij}\| \wedge \|x_{ij}\|.$$

Теперь можно записать функцию цели $Z$ и ограничения задачи двоичного линейного программирования

$$Z = \sum_{j=1}^{\lambda}\left[(V_j+V_j^{(0)})\sum_{i=1}^{n}x_{ij} - (V_j+V_j^{(1)})\right] \to \max;$$

$$\forall_i,\ T_i^{\text{р}} + \sum_{j=1}^{\lambda} x_{ij}(f_{ij}\tau_j+\tau_{j0}) \leqslant T_i^{\text{д}}; \qquad (5.7)$$

$$x_{ij}=0,1;\ \forall_j;\ \sum_{i=1}^{n}x_{ij} \geqslant k_j.$$

Через $T_j^{\text{р}}$ обозначено время реализации $i$-й программы до выделения общих частей.

После решения поставленной задачи получим матрицу $\|x_{ij}\|$. Равенство $x_{ij}$ нулю означает, что $i$-я программа не должна обращаться к подпрограмме $s_j$, и, независимо от того, что эта подпрограмма может существовать для обращения к ней других программ, в $i$-й программе общая часть повторяется.

Рассмотренная постановка минимизации суммарного объема памяти применима для всех $n \geqslant 2$. Однако при

$n=2$ можно получить более простую постановку, особенно с точки зрения ограничений. В этом случае функция цели $Z$ и ограничения запишутся следующим образом:

$$Z=\sum_{j=1}^{\lambda} x_j\left(V_j-2V_j^{(0)}-V_j^{(1)}\right)\to\max \qquad (5.8)$$

при

$$T_1^{\text{р}}+\sum_{j=1}^{\lambda} x_j(f_{1j}\tau_j+\tau_{j0})\leqslant T_1^{\text{д}};$$

$$T_2^{\text{р}}+\sum_{j=1}^{\lambda} x_j(f_{2j}\tau_j+\tau_{j0})\leqslant T_2^{\text{д}}.$$

Число переменных $x_j$ равняется числу подпрограмм. Неравенство (5.6) имеет следующий вид: $2V_j\geqslant V_j+2V_j^{(0)}+V_j^{(1)}$. Отсюда выигрыш в объеме памяти от выделения $j$-й общей части в подпрограмму $s_j$ равняется

$$V_j-2V_j^{(0)}-V_j^{(1)}.$$

В тех случаях, когда ограничения (5.7) или (5.8) удовлетворяются для всех $x_{ij}=1$ или $x_j=1$, решение соответствует матрицам $\|a_{ij}\|$ или $\|a_j\|$, т. е.

$$\|a_{ij}^*\|=\|a_{ij}\| \text{ или } \|a_j^*\|=\|a_j\|.$$

Схемы связей программы с подпрограммами для рассматриваемой задачи экономии памяти при $n>2$ и $n=2$ представлены соответственно на рис. 5.3 и 5.4.

Рассмотрим разновидность задачи экономии памяти с учетом временны́х ограничений при $n=1$.

Пусть дана написанная и отлаженная программа $\pi$, удовлетворяющая с некоторым запасом допустимое вре-

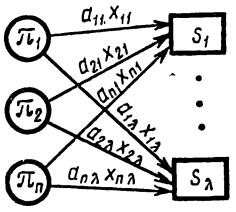



Рис. 5.3. Схема связей для задачи экономии памяти при $n>2$

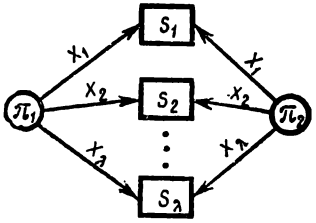



Рис. 5.4. Схема связей для задачи экономии памяти при $n=2$

мя реализации $T^{д}$ и содержащая $\lambda$ общих частей. Пронумеруем в программе условно целыми числами от 1 до $\theta_1$ все места, содержащие первую общую часть; целыми числами от 1 до $\theta_2$ — все места, содержащие вторую общую часть, и т. д. Нумерация как общих частей, так и мест в программе осуществляется случайным образом.

Для удобства воспользуемся уже введенными обозначениями, но со следующим смыслом: $W_j$ — необходимое число ячеек памяти для включения подпрограммы $s_j$ в некотором месте программы $\pi$; $V_j^{(0)}$ — часть $V_j$, которая относится к каждому из мест, в которых используется $s_j$-я подпрограмма; $V_j^{(1)}$ — часть, относящаяся только к подпрограмме $s_j$ (сохраняется прежний смысл); $k_j$ — минимальное число мест в программе, в которых используется подпрограмма $s_j$: $f_{ij}$ — частота обращения программы $\pi$ в $i$-м месте к $s_j$-й подпрограмме; $x_{ij}=1$; если обращение программы в $i$-м месте к $s_j$-й подпрограмме целесообразно; $x_{ij}=0$ в противном случае.

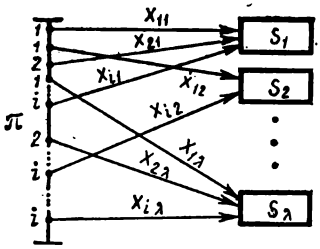



Рис. 5.5. Схема связей для задачи экономии памяти при $n=1$

Схема связей задачи экономии памяти при $n=1$ представлена на рис. 5.5. Очевидно, что при данной разновидности задачи выигрыш в памяти от выделения $j$-й общей части в подпрограмму будет иметь место тогда, когда программа обращается к ней не меньше, чем в $k_j$ местах, где $k_j$ определяется из (5.6).

*Функция цели* и *ограничения* в этом случае записываются следующим образом:

$$Z=\sum_{j=1}^{\lambda}\left[\left(V_j-V_j^{(0)}\right)\sum_{i=1}^{\theta_j}x_{ij}-\left(V_j+V_j^{(1)}\right)\right];$$

$$\sum_{j=1}^{\lambda}\sum_{i=1}^{\theta_j}x_{ij}\left(f_{ij}\tau_j+\tau_{j0}\right)\leqslant T^{д}-T^{р};$$

$$\forall_j,\ \sum_{i=1}^{\theta_j}x_{ij}\geqslant k_j;\quad x_{ij}=0,\ 1.$$

Рассмотренная в данном параграфе задача с разновидностями является, по сути дела, задачей экономии программной памяти. Ее применение в конкретной ситуации позволит с учетом временных ограничений дополнительно минимизировать объем памяти для хранения множества программ при наличии в них общих частей.

## § 5.5. АНАЛИЗ АЛГОРИТМОВ ПРОГРАММНОЙ РЕАЛИЗАЦИИ НА МИКРОЭВМ ПОМЕХОУСТОЙЧИВЫХ КОДОВ

В настоящее время в СПИ наиболее широко применяются ОЗО на основе блочных линейных двоичных кодов. Для них оцениваются и анализируются показатели качества: время обработки в циклах на блок $T(n, k)$, (ц/блок) и $V(n, k)$ — требуемый объем программной и рабочей памяти в ячейках при различных алгоритмах реализации.

Для сравнения возможных вариантов программной реализации ОЗО на этапе системного проектирования воспользуемся моделью класса микроЭВМ (гипотетической микромашиной — ГММ), для которой определен ассемблер с учетом характерных особенностей систем команд распространенных микроЭВМ (см. табл.). Это абсолютный ассемблер, соответствующий машинным командам. Мнемонические обозначения выбраны в расчете наибольшего приближения к выполняемым действиям. В последней графе таблицы представлены времена выполнения команд в циклах (ц). Для адресных команд времена перед скобками соответствуют случаю прямой адресации, а в скобках — случаю одноуровневой косвенной адресации. Признаком косвенной адресации будет являться наличие значка : 1 : в форматах адресных команд. В большинстве микромашин использу-

Абсолютный ассемблер ГММ

| | Команды чтения и записи | | | Время, цикл |
|---|---|---|---|---|
| Чтение из памяти | ЧИТ | адрес | (Σ) : = (адрес) | 2 (3) |
| Запись в память | ЗАП | адрес | (адрес) : = (Σ) | 2 (3) |
| | Арифметические команды | | | |
| Сложение | СЛЖ | адрес | (Σ) : = (адрес) + + (Σ) | 2 (3) |
| Вычитание | ВЫЧ | адрес | (Σ) : = (адрес) — —(Σ) | 2 (3) |

*Продолжение*

Логические команды

| | | | | |
|---|---|---|---|---|
| Инверсия | ИНВ | | $(\Sigma) := (\overline{\Sigma})$ | 1 |
| Конъюнкция | КОН | адрес | $(\Sigma) := (\text{адрес}) \wedge \wedge (\Sigma)$ | 2 (3) |
| Дизъюнкция | ДИЗ | адрес | $(\Sigma) := (\text{адрес}) \vee \vee (\Sigma)$ | 2 (3) |
| Обнуление сумматора | ОБН | | $(\Sigma) := (\Sigma)$ | 1 |
| Обнуление регистра РР | ОБР | | $(РР) := 0$ | 1 |
| Сдвиг правый | СДП | | $(\Sigma) \overset{1}{\rightarrow}$ | 1 |
| Сдвиг левый | СДЛ | | $\overset{1}{\leftarrow} (\Sigma)$ | 1 |
| Циклический сдвиг правый | ЦСП | | | 1 |
| Циклический сдвиг левый | ЦСЛ | | | 1 |
| Исключающее ИЛИ | МОД | адрес | $(\Sigma) := (\text{адрес}) + + (\Sigma)$ | 2 (3) |

Команды передачи управления

| | | | | |
|---|---|---|---|---|
| Безусловный переход | БЗП | адрес | | 1 (2) |
| Переход с запись адреса возврата | ПЗВ | адрес | | 2 (3) |
| Пропуск по плюсу | ППП | | Пропуск следующей команды, если $(\Sigma) \geqslant 0$ | 1 |
| Пропуск по равенству нулю | ПРН | | Пропуск следующей команды, если $(\Sigma) = 0$ | 1 |
| Пропуск по нулю регистра РР | ПНР | | Пропуск следующей команды, если $(РР) = 0$ | 1 |
| Пропуск по сложению | ПСЛ | адрес | (адрес) := (адрес) + 1 и если (адрес) = 0, то пропуск следующей команды | 2 (3) |

Команды обмена информации

| | | | |
|---|---|---|---|
| Ввод из буферного регистра БР | ВВД | адрес БР | $(\Sigma) := (БР)$ |
| Вывод в буферный регистр БР | ВЫВ | адрес БР | $(БР) := (\Sigma)$ |
| Пропуск по «флажку» | ППФ | адрес БР | Пропуск следующей команды, если «флажок» поднят |
| Запрет прерывания | ЗПР | адрес БР | |
| Разрешение прерывания | РПР | адрес БР | |

ется единичная глубина, что увеличивает время на реализацию команд на один цикл. Требуемое время процессора ГММ при несовмещенном обмене одной посылкой составляет $10\tau_{\text{ц}}$, при совмещенном программном — $20\tau_{\text{ц}}$ и $1\tau_{\text{ц}}$ по каналу прямого доступа.

### Матричный способ задания линейных кодов

Линейный $(n, k)$-код может быть задан порождающей $G=\|I_k \vdots \mathrm{P}\|$ или проверочной $H=\|-\mathrm{P}^T \vdots I_{n-k}\|$ матрицами, а также с помощью таблицы кодирования. Так как рассматриваются линейные коды над двоичным полем Галуа, то $\mathrm{P}^T=-\mathrm{P}^T$, поскольку в этом случае элементы поля совпадают со своими обратными элементами.

Каждый вектор $\mathbf{u}_n$ систематического линейного $(n, k)$-кода может быть записан в виде $\mathbf{u}_n=(u_k, u_{n-k})$. Оператор кодирования в этом случае реализуется по одному из следующих соотношений: $u_n=(u_k, u_{n-k})=u_kG$; $u_n=[u_k(0)]+[(0), u_k\mathrm{P}]=(u_k, u_k\mathrm{P})$.

Для декодирования используем соотношение $u_n \cdot H=0$. По синдрому $s$ из $n-k$ компонент, определяемому по соотношению $\overset{*}{u_n}H=s$, где $\overset{*}{\mathbf{u}_n}$ — принятый из канала вектор, можно принимать решение, соответствующее наиболее вероятному исходу в зависимости от принятого режима декодирования.

Данный способ задания (матричный) является общим для всех линейных кодов. На его основе оценим введенные показатели качества при программной реализации класса линейных систематических двоичных кодов в режиме обнаружения ошибок. Рассмотрим два основных случая: 1) все символы кодового вектора и элементы матрицы P записываются в отдельные последовательные ячейки памяти; 2) информационный и кодовый векторы, столбцы матрицы P записываются в последовательные ячейки памяти посылками длиной $l$.

Алгоритм кодирования — декодирования *для первого случая* представлен на рис. 5.6, где $B[i]$, $i=\overline{1, k}$ — массив символов информационного вектора; $B[j]$, $j=\overline{1, (n-k)}$ — массив проверочных символов; $A[j[i]]$ — массив элементов матрицы $\mathrm{P}^T$. Логическая переменная $\varepsilon$ соответствует кодированию при $\varepsilon=1$ и декодированию при $\varepsilon=0$. Алгоритм декодирования отличается от алгоритма кодирования только одним логическим усло-

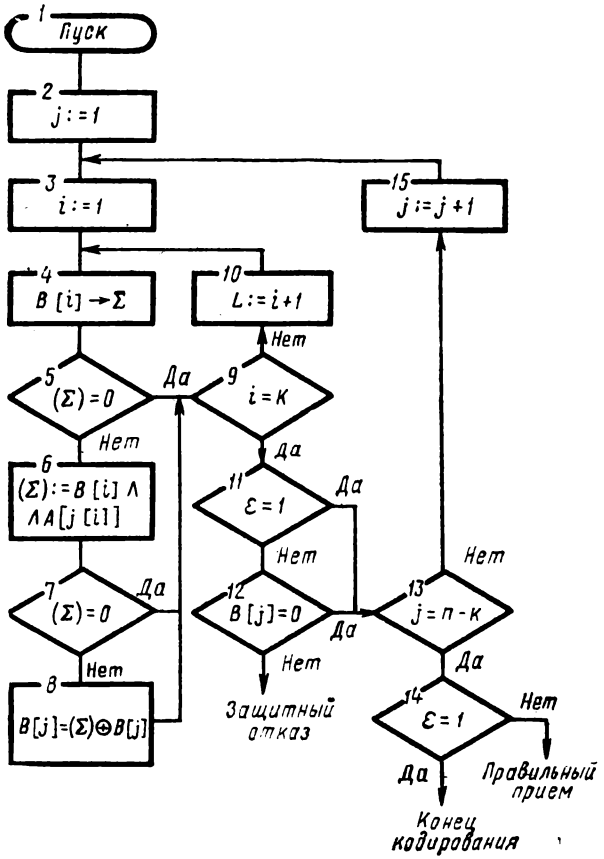



Рис. 5.6. Алгоритм кодирования — декодирования при записи символов кодового вектора и элементов матрицы ρ в отдельные ячейки памяти

вием, которое проверяется после скалярного произведения вектора информационных символов и каждой строки матрицы $P^T$.

По данному алгоритму составлена программа на ассемблере ГММ. После анализа программы получены следующие оценки показателей качества для кодирования:

$$V_{1k}(n,\ k) = k(n-k) + n + 29,\ \text{ячеек},$$

$$T_{1k}(n,\ k) = 22k(n-k) + 13(n-k),\ \text{ц/блок};$$

и соответственно для декодирования

$$V_{1д}(n,k) = V_{1к}(n,\ k) + 3,\ \text{ячеек},$$
$$T_{1д}(n,\ k) = T_{1к}(n,\ k) + 4(n-k),\ \text{ц/блок}.$$

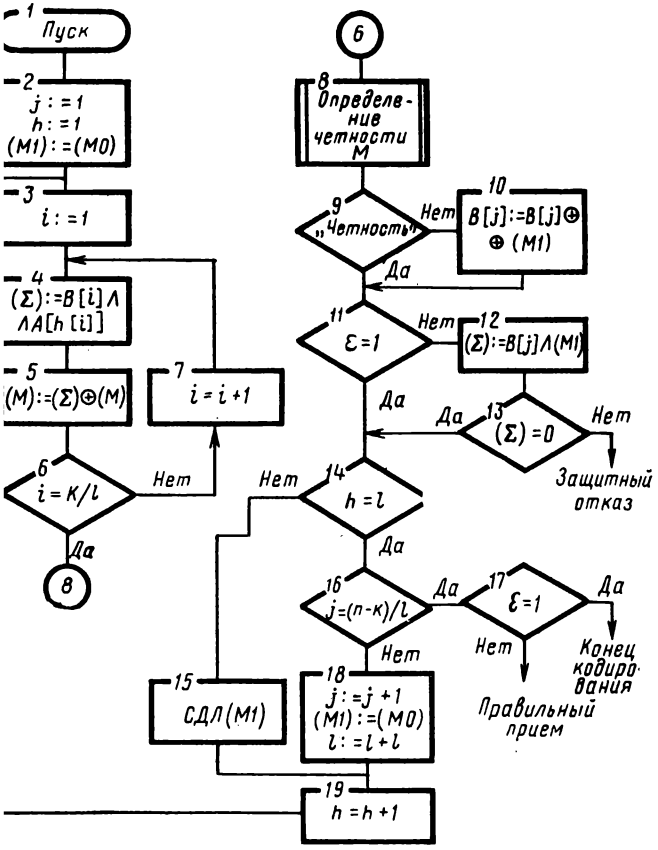
Для достаточно больших $n$ и $k$ являются целесообразными оценки $V_{1к}\ (n,\ k) \cong (n-k)$ и $T_{1к}\ (n,\ k) \cong 22k\ (n-k)$.

Значения показателей $V_{1к}\ (n,\ k)$ и $V_{1д}\ (n,\ k)$ определяются в основном матрицей P, которую необходимо хранить постоянно в памяти кодека.

Алгоритм кодирования — декодирования *для второго случая* представлен на рис. 5.7, где $B[i]$, $i=\overline{1,\ \lceil k/l \rceil}$ — массив посылок информационного вектора; $B[j]$, $j=\overline{1,\ \lceil (n-k)/l \rceil}$ — массив посылок проверочных символов; $A\ [h[i]]$, $h=\overline{1,\ (n-k)}$ — массив посылок столбцов матрицы $P^T$. Для удобства можно принять $\lceil k/l \rceil = k/l$ и $\lceil (n-k)/l \rceil = (n-k)/l$. Несоблюдение указанных соотношений при обработке кодов посылками во всех случаях приводит к усложнению алгоритма кодирования — декодирования и (или) алгоритма обмена с внешней средой. Поэтому, например, в одном из существующих ПК, в котором обработка ведется посылками длиной $l=12$, используемый БЧХ код (63,51) укорачивается до (60,48).

Алгоритм (рис. 5.7) начинается выбором из памяти первой посылки информационного вектора, после чего осуществляется конъюнкция с первой посылкой первого столбца матрицы P, потом выбирается вторая посылка информационного вектора и т. д. до тех пор, пока не будут обработаны все посылки информационного вектора. Результаты операции коньюнкции суммируются по модулю два и в конце определяется четность полученной суммы. Данный процесс повторяется для всех столбцов матрицы P. Остановимся на подпрограмме определения четности содержимого ячейки M, представляющего собой последовательность нулей и единиц длины $l$. В микроЭВМ в большинстве случаев отсутствует команда «Четность», поэтому данная операция должна осуществляться по подпрограмме. Проведенный анализ алгоритмов и подпрограмм определения четности на ГММ показал, что при $l \leqslant 16$ оптимальной по затратам памяти и времени является линейная (без циклов) подпрограмма, построенная на следующей проце-

выделяются $\lfloor l/2 \rfloor$ старших разрядов содержимого ки, ($\lfloor x \rfloor$ — ближайшее целое, не большее $x$) и при-няются по модулю два к оставшимся $\lceil l/2 \rceil$ разрядам, ые перед этим сдвигаются влево на $\lceil l/2 \rceil$ разрядов; олученных $[l/2]$ разрядов выделяются $\lfloor \lceil l/2 \rceil /2 \rfloor$ раз-в и прибавляются к сдвинутым влево оставшимся ]/2] разрядам и т. д. до тех пор, пока не останется разряд, значение которого и будет определять чет-ь или нечетность содержимого ячейки. Для реализа-




5.7. Алгоритм кодирования — декодирования при записи олов информационного и кодового векторов, столбцов матрицы **P** в ячейки памяти посылками длины $l$

ции данной подпрограммы требуется $(6\lceil\log_2 l\rceil + l - 1)$ ячеек памяти и $(10\lceil\log_2 l\rceil + l - 2)$ циклов.

По алгоритму кодирования—декодирования на рис. 5.7 составлена программа для модели микроЭВМ. После анализа программы получены следующие оценки качества:

$$V_{2k}(n,\ k) = \lceil k/l\rceil (n-k) + \lceil k/l\rceil + \left\lceil \frac{n-k}{l} \right\rceil + \\ + 6\lceil\log_2 l\rceil + l + 45;$$

$$T_{2k}(n,\ k) = \left\lceil \frac{n-k}{l} \right\rceil l\,[26\lceil k/l\rceil + 10\lceil\log_2 l\rceil + l + 16;$$

$$V_{2д}(n,\ k) = V_{2k}(n,\ k) + 6;$$

$$T_{2д}(n,\ k) = T_{2k}(n,\ k) + 9l\left\lceil \frac{n-k}{l} \right\rceil.$$

Второй алгоритм обладает лучшими показателями качества по сравнению с первым. Полученные оценки по времени при первом алгоритме являются верхними оценками реализуемости класса линейных систематических двоичных кодов в режиме обнаружения ошибок в классе микроЭВМ.

### Табличный способ задания линейных кодов

Оценим показатели качества при самом общем способе задания линейных кодов — табличном. В этом случае в памяти необходимо хранить таблицу из $2^k$ кодовых или проверочных векторов. При кодировании по принятому информационному вектру из таблицы выбирается кодовый $\mathbf{u}_n$ или проверочный $\mathbf{u}_{n-k}$ векторы. При декодировании по принятому из канала вектору $\mathbf{u}_k^*$ выбирается из таблицы соответствующий ему кодовый или проверочный вектор, который сравнивается с принятым кодовым или проверочным векторами.

При $n>p$ ($p$ — разрядность машины) использование табличного метода можно считать возможным, если $k<p$. В противном случае для записи таблицы потребуется объем памяти, не меньший объема всего основного запоминающего блока машины.

Оценки показателей качества в этом случае следующие: $V_{зк}(n,\ k)=2^k+9$; $T_{зк}(n, k)=11$. Программа декодирования включает в себя программу кодирования,

соответствующие оценки получаются в виде

$$V_{зд}(n, k) = 2^k + 12; \quad T_{зд}(n, k) = 14.$$

По времени обработки данный алгоритм самый быстрый, но требует большого объема памяти и применим только к коротким кодам. Оценки памяти при этом являются верхними оценками реализуемости двоичных линейных кодов в классе мини- и микромашин.

Для реализации более длинных кодов данный алгоритм может быть изменен следующим образом: информационный вектор делится на подблоки, для каждого подблока составляется своя таблица кодирования, в которой проверочные векторы соответствуют информационным, образованным данным подблоком и нулями на местах остальных подблоков. При кодировании по подблокам информационного вектора отыскиваются из таблиц промежуточные проверочные векторы, которые складываются по модулю два, в результате получается проверочный вектор для исходного информационного. С использованием данного метода составлен алгоритм кодирования — декодирования для общего случая, когда информационный и проверочный векторы записываются посылками в последовательные ячейки памяти. Этот алгоритм (четвертый) ориентирован на класс микромашин, которые отличаются небольшой разрядностью, и поэтому кодирование в микромашинах при используемых на практике длинах приведет к необходимости записывать информационные и проверочные векторы в несколько ячеек, так как чаще всего $k$ и $n—k$ больше $p$. Естественно, что в этих случаях для записи в таблицы кодирования каждого промежуточного проверочного вектора потребуется $(n—k)/l$ ячеек памяти. Поэтому по информационной посылке, как по адресу, невозможно сразу отыскать весь соответствующий промежуточный проверочный вектор. Запись таблиц кодирования предлагается производить следующим образом: все первые посылки векторов первой таблицы записываются в последовательные ячейки памяти, начиная с некоторого начального адреса; непосредственно за первой таблицей записываются в последовательные ячейки первые посылки векторов второй таблицы и т. д.; после последней таблицы записываются вторые посылки векторов первой таблицы, потом вторые посылки векторов второй таблицы и на последнем листе — последние посылки векторов последней таблицы. По разработанно-

му алгоритму для модели микроЭВМ составлена программа, после анализа которой получены следующие оценки показателей качества:

$$V_{4k}(n,\ k) = \left\lceil \frac{n-k}{l} \right\rceil \left( \left\lceil \frac{k}{l} \right\rceil 2^l + \gamma 2^{k - \left\lceil \frac{k}{l} \right\rceil l} \right) + \\ + \left\lceil \frac{k}{l} \right\rceil + \left\lceil \frac{n-k}{l} \right\rceil + 29;$$

$$T_{4k}(n,\ k) = \left( 20 \left\lceil \frac{n-k}{l} \right\rceil + 26 \right) [k/l] + 13;$$

$$V_{4д}(n,\ k) = V_{4k}(n,\ k) + 10;\ \ T_{4д}(n,\ k) = \\ = T_{4k}(n,\ k) + 9 \left\lceil \frac{n-k}{l} \right\rceil + 8,$$

где $\gamma=0$, если $k/l$ — целое число; $\gamma=1$ в противном случае.

При конкретных реализациях с использованием этого алгоритма необходимо определить оптимальное значение $l (2 \leqslant l \leqslant p)$, поскольку увеличение $l$ при заданных $n$ и $k$ приводит к уменьшению времени обработки и увеличению требуемого объема памяти. Длины $l>10$ нецелесообразны, так как они приводят к резкому увеличению объема требуемой памяти. Длины посылок $l \leqslant 10$ хорошо согласуются с малой разрядностью микроЭВМ. При этом, однако, в некоторых 12- и 16-разрядных машинах загрузка используемой памяти будет неполная.

Сравнение рассмотренных четырех алгоритмов показывает, что основными алгоритмами реализации ОЗО на основе двоичных линейных кодов с обнаружением ошибок в классе микроЭВМ являются второй и четвертый алгоритмы. Выбор оптимального из них может быть произведен по комплексному показателю качества, так как второй алгоритм оптимален по памяти, а четвертый — по времени обработки.

### Циклические коды

Широко используемым классом линейных кодов по возможностям защиты от ошибок являются *циклические коды*, способные обнаруживать подавляющее большинство зависимых и независимых ошибок. Для обнаружения ошибок в системах передачи и хранения информации МККТТ рекомендован циклический $(n,\ n-16)$-код, где $n=256$, 496 или 976 двоичным символам. Широкое ис-

пользование циклических кодов обусловлено еще и разработанными для них относительно простыми способами аппаратурной реализации. Поэтому, не останавливаясь подробно на алгоритмах, укажем на несколько путей при программной реализации циклических кодов в классе микромашин.

Первый путь основан на том, что циклические коды являются линейными, поэтому для них правомерно все, что относится к линейным кодам.

Второй путь основывается на возможности реализации процесса деления многочленов в двоичном поле Галуа при помощи операций сдвига и сложения по модулю два одновременно над многими символами информационного вектора. Программная реализация в этом случае осуществляется просто, если $k \leqslant p$ и $(n-k+1) \leqslant p$. Если и (или) $(n-k+1) > p$, то информационный и (или) проверочный полиномы займут по несколько ячеек памяти. Последнее часто имеет место в микромашинах, что усложняет программирование и увеличивает время обработки. Показатель по времени обработки в этом случае меньше по сравнению с показателем второго алгоритма и больше по сравнению с показателем четвертого алгоритма первого пути. В то же время показатель по памяти будет минимальным.

Третий путь основывается на усовершенствованном по затратам памяти варианте четвертого (с использованием нескольких таблиц) алгоритма первого пути. Этот вариант базируется на свойствах циклического кода и известен как алгоритм с использованием одной таблицы частичных остатков.

На примере ГММ можно показать, что при реализации циклических кодов в классе микромашин показатели по времени обработки алгоритма с использованием одной таблицы частичных остатков и четвертого алгоритма первого пути будут отличаться незначительно. В то же время показатели по затратам памяти могут существенно отличаться в пользу алгоритма с использованием одной таблицы частичных остатков.

Таким образом, при реализации циклических кодов с обнаружением ошибок в классе микромашин целесообразны в основном второй и третий путь. Выбор между ними может быть осуществлен только с учетом комплексного показателя качества, потому что второй путь характеризуется меньшим объемом требуемой памяти, а третий — меньшим временем обработки.

Для реализации второго пути рассмотрим алгоритм, который предполагает формирование частичных остатков в процессе обработки и поэтому не требует таблицы остатков. Назовем его *алгоритмом с частичным делением.* Суть этого алгоритма в следующем. В сумматоре Σ записана $i$-я посылка информационного (кодового) вектора длины $l$ (здесь $i=1$, $(k'/l)$, где $(k'/l)=[k/l]$ — число посылок информационного вектора в общем случае, когда $(k/l)$ — не целое число. Замена $k$ на $k'$ (соответственно и $n$ на $n'$) означает, что со стороны старших разрядов прибавлено $(k'—k)$ нулей, и информационный вектор рассматривается фиктивно, как имеющий $k'$ компонент. Очевидно, что $n'—k'=n—k$. В некоторой ячейке М хранится постоянно порождающий многочлен без самого старшего разряда, который всегда является значащим и подразумевается. Таким образом, в ячейку с разрядностью $p$ можно записывать порождающие многочлены степени $(n'—k')\leqslant p$. После левого сдвига содержимого сумматора старший разряд $i$-й посылки окажется в регистре РР расширителя сумматора. Содержимое регистра РР проверяется на равенство нулю (командой типа ПНР), и, если оно не равно нулю, то к содержимому сумматора прибавляется по модулю два содержимое ячейки М. Если содержимое регистра РР равно нулю, то осуществляется сдвиг. Таким образом, после сдвигов в сумматоре будет находиться частичный остаток от деления многочлена $i$-й посылки, умноженного на $X^{n'-k'}$ на порождающих многочлен. Далее к содержимому сумматора прибавляется по модулю два $(i+1)$-я посылка, и процесс повторяется. После обработки $(k'/l)$-й посылки в сумматоре получится проверочный полином.

По данному алгоритму составлена программа на ассемблере ГММ и получены следующие выражения для показателей качества:

$$V_{\text{бд}}(n,k) \simeq V_{\text{бк}}(n,k) = |k/l| + 2{,}5,$$

$$T_{\text{бд}}(n,k) \simeq T_{\text{бк}}(n,k) = (10l + 12)\,|k/l| + 15.$$

Рассмотренный алгоритм сравнительно легко реализуем в классе микромашин при помощи расширителя сумматора, если $(n'—k')\leqslant l$. При $(n'—k')>l$ прямое деление на порождающий многочлен реализуется с существенными затратами машинного времени.

Реализация на ассемблере ГММ алгоритмов с использованием одной таблицы частичных остатков харак-

теризуется следующими выражениями для показателей качества:

$$\text{при } (n'-k') \leqslant l$$

$$V_{\text{бд}}(n,k) \simeq V_{\text{бк}}(n,k) = 2^l + 23 + \lfloor k/l \rfloor,$$

$$T_{\text{бд}}(n,k) \simeq T_{\text{бк}}(n,k) = 19\lfloor k/l \rfloor + 13;$$

$$\text{при } (n'-k') > l:$$

$$V_{\text{бк}}(n,k) = \lceil (n-k)/l \rceil (2^l + 1) + \lfloor k/l \rfloor + 27,$$

$$T_{\text{бк}}(n,k) = (20 \cdot \lceil (n-k)/l \rceil + 24)\lfloor k/l \rfloor + 12,$$

$$V'_{\text{бд}}(n,k) = V'_{\text{бк}}(n,k) + 9,$$

$$T_{\text{бд}}(n,k) = T_{\text{бк}}(n,k) + 9\lceil (n-k)/l \rceil + 6.$$

Выражение для границы по быстродействию, (с/ц), при использовании алгоритма с частичным делением имеет вид

$$\tau_{\text{ц}} \leqslant n/\{v[30 + (10l + 32)\lfloor k/l \rfloor]\}. \qquad (5.9)$$

Выражения для границ по быстродействию при использовании алгоритма с таблицей частичных остатков имеет вид

$$\text{при } (n-k) \leqslant 8$$

$$\tau_{\text{ц}} \leqslant n/[v(25 + 39\lfloor k/8 \rfloor)], \qquad (5.10)$$

$$\text{при } (n-k) > 8$$

$$\tau_{\text{ц}} \leqslant n/\left\{v\left[18 + \left(20\left\lceil\frac{n-k}{8}\right\rceil + 44\right)\lfloor k/8 \rfloor + 19\left\lceil\frac{n-k}{8}\right\rceil\right]\right\}. \qquad (5.11)$$

Приведенные выражения получены в предположении, что обмен информацией осуществляется через программно-управляемый канал, т. е. $N_{\text{в}} = N_{\text{вв}} = 10$ ц/посылку. Длина посылки в случае алгоритма с таблицей остатков принята для определенности равной 8. Такая длина наиболее целесообразна при исследовании в общем, поскольку она согласуется с разрядностью микромашин — 8, 12, 16 или 32 и требует сравнительно небольшого объема памяти для хранения таблицы остатков. В случае алгоритма с частичным делением длина посылки меняется в зависимости от параметров кодов $n$ и $k$.

Оценки допустимых времен цикла $\tau_{\text{ц.доп}}$, мкс/ц, по формуле (5.9) для некоторых циклических кодов при различных значениях нагрузки $v$ приведены в табл. 5.1 с точностью до второго значащего разряда.

Таблица 5.1

| Код $l$ | $v$, бод | | | | | | |
|---|---|---|---|---|---|---|---|
| | 50 | 100 | 600 | 1200 | 4800 | 9600 | 47 500 |
| (15,8), 8 | 2100 | 1100 | 180 | 88 | 22 | 11 | 2,2 |
| (31,16), 16 | 2800 | 1400 | 230 | 120 | 29 | 15 | 2,9 |
| (45,30), 15 | 2300 | 1100 | 190 | 95 | 24 | 12 | 2,4 |
| (60,48), 12 | 1900 | 940 | 160 | 78 | 20 | 9,8 | 2,0 |
| (63,56), 8 | 1500 | 770 | 130 | 64 | 16 | 8,0 | 1,6 |
| (127,120), 8 | 1500 | 740 | 120 | 62 | 15 | 7,7 | 1,6 |
| (127,120), 15 | 1700 | 850 | 140 | 71 | 18 | 8,9 | 1,8 |
| (256,240), 16 | 1800 | 880 | 150 | 73 | 18 | 9,2 | 1,9 |
| (496,480), 16 | 1700 | 860 | 140 | 71 | 18 | 8,9 | 1,8 |
| (976,960), 16 | 1700 | 850 | 140 | 70 | 18 | 8,8 | 1,8 |

Из приведенных расчетов следует, что алгоритм с использованием частичного деления может быть реализован в классе микромашин при большинстве применяемых на практике скоростях модуляции в каналах связи. Объем памяти, который потребуется при этом, незначителен.

Оценки допустимых времен цикла $\tau_{ц.доп}$, мкс/ц по формулам (5.10) и (5.11) для алгоритма с таблицей остатков приведены в табл. 5.2 с точностью до второго значащего разряда. Для сравнения с оценками алгоритма на основе частичного деления использованы те же самые циклические коды, что и в табл. 5.1.

Таблица 5.2

| Код $l$ | $v$, бод | | | | | | |
|---|---|---|---|---|---|---|---|
| | 50 | 100 | 600 | 1200 | 4800 | 9600 | 47 500 |
| (15,8) | 4700 | 2300 | 390 | 200 | 49 | 24 | 4,9 |
| (31,16) | 2800 | 1400 | 230 | 120 | 29 | 15 | 2,9 |
| (45,30) | 2300 | 1100 | 190 | 96 | 24 | 12 | 2,4 |
| (60,48) | 2100 | 1100 | 180 | 89 | 22 | 11 | 2,3 |
| (63,56) | 4200 | 2100 | 350 | 180 | 44 | 22 | 4,5 |
| (127,120) | 4200 | 2100 | 350 | 170 | 43 | 22 | 4,4 |
| (256,240) | 2000 | 1000 | 170 | 83 | 21 | 10 | 2,1 |
| (496,480) | 1900 | 970 | 160 | 81 | 20 | 10 | 2,0 |
| (976,960) | 1900 | 960 | 160 | 80 | 20 | 10 | 2,0 |

На основе формул (5.9) и (5.10) можно получить асимптотические границы по быстродействию, (с/ц), записанные в следующем виде:

$$\lim_{k\to\infty} \tau_{\text{ц.доп}}(k) = l/[v(10l+32)], \qquad (5.12)$$

$$\lim_{k\to\infty} \tau_{\text{ц.доп}}(k) = 0{,}2/v. \qquad (5.13)$$

Данные выражения означают, что, начиная с некоторой длины информационного вектора, дальнейшее увеличение ее не приведет к изменению требований к быстродействию кодирующего процессора.

Число проверочных символов рассматриваемых кодов не больше 16. С учетом этого можно определить асимптотическую границу по быстродействию и на основе формулы (5.11):

$$\lim_{k\to\infty} \tau_{\text{ц.доп}}(k) = 0{,}1/v. \qquad (5.14)$$

Сравнивая выражения (5.13) и (5.14), можно сказать, что при переходе от случая $(n-k)\leqslant 8$ к случаю $8<(n-k)\leqslant 16$ требование к быстродействию процессора усиливается в два раза.

Аналогичные расчеты можно провести в случаях обмена по прерыванию работы процессора или через КПД для выявления влияния способа обмена на требуемое быстродействие процессора.

### Простой итеративный код

В СПИ для АСУ во многих случаях целесообразно использовать ОЗО на основе простого итеративного кода в режиме обнаружения ошибок. Простой итеративный код с точки зрения удобства последующей алгоритмизации целесообразно записать в виде

$$\begin{array}{cccc|c}
a_1 & a_2 & \dots & a_\eta & r_{k+1} \\
a_{\eta+1} & a_{\eta+2} & \dots & a_{2\eta} & r_{k+2} \\
\vdots & & & & \\
a_{k-\eta+1} & a_{k-\eta+2} & \dots & a_k & r_{n-\eta-1} \\
\hline
r_{n-\eta} & r_{n-\eta+1} & \dots & r_{n-1} & r_n
\end{array}$$

где $a_1 \dots a_k$ — символы информационного вектора; $r_{k+1} \dots r_{n-\eta-1}$ — символы проверок по строкам; $r_{n-\eta} \dots r_{n-1}$ — символы проверок по столбцам; $r_n$ — символ проверки проверок.

Проверки могут осуществляться либо на четность, либо на нечетность. Символ может быть результатом проверки как по строкам, так и по столбцам.

При программной реализации удобно каждую строку кодовой матрицы рассматривать как посылку и хранить в отдельной ячейке памяти, причем последовательные посылки — в последовательных ячейках. Таким образом, для записи кодового вектора необходимо $(k/\eta+1)$ ячеек памяти, начиная с некоторого начального адреса. При этом длина посылок ввода при кодировании будет $\eta$, а при декодировании — $(\eta+1)$.

Алгоритм кодирования-декодирования простого итеративного кода включает в себя те же операторы и процедуру определения четности, что и алгоритм, представленный на рис. 5.7. Отметим некоторые особенности алгоритма простого итеративного кода, и приведем выражения для оценки качества, полученные на ГММ.

При определении четности массив посылок кодового вектора $\mathbf{B}[i]$ $i=\overline{1,\ (k/l+1)}$ в случае кодирования рассматривается как $\eta$-разрядная, а в случае декодирования—как $(\eta+1)$-разрядная посылки. Если используется составной алгоритм, т. е. алгоритм кодирования-декодирования, то целесообразно иметь только подпрограмму определения четности, рассчитанную на $(\eta+1)$ разрядов, что сократит объем программной памяти.

Оценки показателей качества для рассматриваемого ОЗО имеют вид:

$$V_{7k}(n,k) = 21 + (k/\eta) + 6\lceil\log_2\eta\rceil + \eta,$$

$$T_{7k}(n,k) = 8 + (k/\eta)\cdot 10\lceil\log_2\eta\rceil + \eta + 22,$$

$$V_{7д}(n,k) = 16 + (k/\eta) + 6\lceil\log_2(\eta+1)\rceil + \eta,$$

$$T_{7д}(n,\ k) = 11 + (k/\eta)\cdot 10\lceil\log_2(\eta+1)\rceil.$$

Показатель по времени в случае кодирования при конкретных $n$ и $k$ является постоянной величиной, а в случае декодирования — случайной. Полученное выражение для $T_{7д}(n,\ k)$ определяет максимальное время обработки, имеющее место при необнаружении ошибок. При приеме с защитным отказом время обработки будет меньше и достигнет минимума при обнаружении ошибки в первой посылке. Поскольку вероятность отказа в реальных СПИ очень мала, то математическое ожидание времени обработки несущественно меньше $T_{7д}(n,\ k)$.

Показатели качества по времени обработки значительно ухудшаются за счет времени выполнения подпрограммы определения четности. Для улучшения этого показателя целесообразно в системы команд микромашин ввести команду «четность».

При заданной длине блока $n$ можно построить различные таблицы для простого итеративного кода. Однако, с точки зрения избыточности лучший вариант имеет место, когда $(\eta+1)^2=n$. Очевидно, что в простом итеративном коде с минимальной избыточностью

$$k=(\sqrt{n}-1)^2 \quad \text{и} \quad (n-k)=2\sqrt{n}-1.$$

Анализ на ГММ рассмотренных ОЗО и ряда других (кодов Питерсона — Файра в режиме обнаружения одиночных пачек ошибок, кодов Рида — Маллера в режиме исправления многократных ошибок, итеративного кода с циклическим кодом в первой ступени и проверкой на четность во второй ступени, простого итеративного кода в режиме обнаружения и исправления ошибок) показывает их реализуемость в оперативном режиме в классе современных микромашин при большинстве применяющихся на практике скоростях модуляции в каналах связи.

## ЗАКЛЮЧЕНИЕ

При разработке системы передачи информации — одной из важнейших в современных автоматизированных системах управления — требуется решать задачи выбора структуры системы, протоколов обмена информацией, элементной базы реализации связных устройств, а также проектирования устройств для конкретной системы. В настоящем пособии изложены сведения, необходимые разработчику системы передачи информации, относительно принципов функционирования сетей передачи информации и их основной элементной базы — микропроцессорных средств, раскрыты основные этапы проектирования микропроцессорных устройств для систем передачи информации, дан материал о важнейших математических аппаратах и средствах, используемых в процессе поиска оптимальных решений.

Быстрое развитие микроэлектронной технологии приводит к ускоренной смене и модернизации элементной

базы, используемой разработчиками АСУ. В подсистеме передачи информации осуществляется переход от мини- к микроЭВМ и от аппаратных устройств управления к программируемым микроконтроллерам на БИС. Вместе с тем сложность процесса проектирования устройств связи не снижается из-за роста функциональных задач и требований к подсистеме передачи. Поэтому актуальной задачей на сегодняшний день является развитие методов и средств автоматизированного проектирования.

Настоящее учебное пособие поможет студентам глубже ознакомиться с проблематикой разработки систем передачи информации АСУ, получить конкретные сведения об алгоритмах, структуре и устройствах систем и сетей передачи информации, принципах построения микропроцессорных средств.

# СПИСОК ЛИТЕРАТУРЫ


1. *Аветов Ю. В., Головин Ю. А., Кутузов О. И., Петрович В. И.* Применение микро-ЭВМ «Электроника С5-01» в судовых абонентских пунктах. — М.: Электронная промышленность, 1978, вып. 5.

2. *Аветов Ю. В., Головин Ю. А.* Формализация и оценка эффективности протокола X.25/2. — В кн.: Рекомендация МККТТ X.25 и ее применение в информационно-вычислительных сетях, ч. I. — М.: МЦНТИ, 1983.

3. *Аветов Ю. В., Кутузов О. И., Советов Б. Я.* Программируемые кодеки на микро-ЭВМ. — Л.: ЛЭТИ, 1983.

4. *Аветов Ю. В., Головин Ю. А.* Анализ структур микропроцессорных устройств сетей ЭВМ с использованием $E$-сетевых моделей.— В кн.: Девятая всесоюзная школа-семинар по вычислительным сетям, ч. 1.2. — М. — Пущино, 1984.

5. *Антонов П. Ц., Захаров Ю. В., Шаповалов В. И.* О выборе показателя для комплексной оценки качества продукции. — В кн.: Методы контроля в комплексной системе управления качеством. — Л.: ЛДНТП, 1978.

6. *Балашов Е. П., Пузанков Д. В.* Микропроцессоры и микропроцессорные системы. — М.: Радио и связь, 1981.

7. *Брайли Б. Е., Дуй У. Н.* Процессоры в системах связи. — М.: ТИИЭР. 1977, т. 65, № 9.

8. *Вайда Ф., Чакань В.* Микро-ЭВМ. — М.: Энергия, 1980.

9. *Вальков В. М.* Микроэлектронные управляющие вычислительные комплексы: системное проектирование и конструирование. — Л.: Машиностроение, 1979.

10. *Гальперин М. П. и др.* Микро-ЭВМ «Электроника С5» и их применение. — М.: Сов. радио, 1980.

11. *Геминтерн В. И., Каган Б. М.* Методы оптимального проектирования. — М.: Энергия, 1980.

12. *Гибсон Г., Лю Ю-Ч.* Аппаратные и программные средства микро-ЭВМ. — М.: Финансы и статистика, 1983.

13. *Головин Ю. А., Кутузов О. И., Аветов Ю. В.* Анализ возможности использования микропроцессоров в устройствах передачи данных — В кн.: Комплексные радиоэлектронные системы управления / Под ред. *В. А. Бесекерского.* — Л.: ЛЭТИ, 1977.

14. ГОСТ 25007-81. Стык С1 системы передачи данных. Параметры сопряжения.

15. ГОСТ 26113-84. Процедура управления звеном передачи данных.

16. *Дрожжинов В. И., Мямлин А. Н.* Сети коммутации пакетов с интерфейсом X.25. — В кн.: Рекомендация МККТТ X.25 и ее применение в информационно-вычислительных сетях, ч. 1. — М.: МЦНТИ, 1983.

17. *Дэвис Д., Барбер Д., Прайс У., Соломонидес С.* Вычислительные сети и сетевые протоколы. — М.: Мир, 1982.

18. *Дей Дж. Д., Зиммерман Ю.* Эталонная модель взаимосвязи открытых систем (ВОС). — ТИИЭР, т. 71, 1983, № 12.

19. *Каган Б. М., Сташин В. В.* Микропроцессоры в цифровых системах. — М.: Энергия, 1979.

20. *Калашников В. В.* Сложные системы и методы их анализа.— М.: Знание, 1980.

21. *Калашников В. В.* Организация моделирования сложных систем. — М.: Знание, 1982.

22. *Клингман Э.* Проектирование микропроцессорных систем. — М.: Мир, 1980.

23. *Котов В. Е.* Сети Петри. — М.: Наука, 1984.

24. *Крылов Е. И.* Однокристальные микро-ЭВМ серий К1814, К1820, К1816. — Микропроцессорные средства и системы, 1985, № 2.

25. *Крюков А. М., Мартынов Ю. М., Разгон В. Л.* Математическое обеспечение сетей ПД. — М.: Связь, 1978.

26. *Малашевич Б. М., Шахнов В. А., Коночкин Э. И.* Термины и определения: Микропроцессорные средства и системы. Микропроцессорные интегральные микросхемы. — Микропроцессорные средства и системы, 1984, № 3.

27. *Малашевич Б. М., Шахнов В. А., Коночкин Э. И.* Термины и определения: Микропроцессорные модули. — Микропроцессорные средства и системы. — М., 1984, № 4.

28. *Мамиконов А. Г. Основы построения АСУ.*—М.: Высшая школа, 1981.

29. *Мартынов Ю. М., Губарев Ю. А., Разгон В. Л.* Исследование вопросов построения перспективных абонентских пунктов в сетях передачи данных. — В кн.: Вычислительные средства в технике и системах связи/Под ред. *С. Д. Пашкеева.* — М.: Связь, 1979. Вып. 4.

30. Микро-ЭВМ / Под ред. *А. Дирксена.* — М.: Энергоиздат, 1982.

31. Мультиплексоры передачи данных / Под ред. *В. С. Лапшина, А. И. Корчинского.* — М.: Энергия, 1980.

32. *Овчинников Г. Р.* К расчету среднего времени задержки пакетов в центре коммутации. — Техника средств связи. Серия ТПС, М., 1984. Вып. 2.

33. *Окунев Ю. Б., Плотников В. Г.* Принпицы системного подхода к проектированию в технике связи. — М.: Связь, 1976.

34. *Питерсон Дж.* Теория сетей Петри и моделирование систем.— М.: Мир, 1984.

35. *Прангишвили И. В.* Микропроцессоры и микро-ЭВМ. — М.: Энергия, 1979.

36. *Прангишвили И. В.* Микропроцессоры и локальные сети микро-ЭВМ в распределенных системах управления. — М.: Энергоатомиздат, 1985.

37. *Пугачев В. Н.* Комбинированные методы определения вероятностных характеристик. — М.: Сов. радио, 1973.

38. *Рахимов Т. Н., Советов Б. Я., Заикин О. А.* Основы построения АСУ. — Ташкент: Укитувчи, 1984.

39. Рекомендация МККТТ Х.25 и ее применение в информационно-вычислительных сетях. Ч. 2. Описание рекомендации Х.25. — М.: МЦНТИ, 1983.

40. *Секстон Дж.* Мультимикропроцессоры в сетях пакетной коммутации. — Автоматика и вычислительная техника, 1981, № 3.

41. *Соботка З., Стары Я.* Микропроцессорные системы. — М.: Энергоиздат, 1981.

42. *Советов Б. Я., Рухман Е. Л., Яковлев С. А.* Системы передачи информации от терминалов к ЦВМ. — Л.: ЛГУ, 1978.

43. *Советов Б. Я., Стах В. М.* Построение адаптивных систем передачи информации для автоматизированного управления. — Л.: Энергоиздат, 1982.

44. *Советов Б. Я., Йонев С., Рухман Е. Л.* Оценка эффективности алгоритмов адаптивного управления в сетях обмена данными. — Автоматика и вычислительная техника, 1983, № 3.

45. *Станьоне Д. С.* Микропроцессоры в системах связи. — М., ТИИЭР, т. 66, 1978, № 2.

46. *Суздалев А. В.* Сети передачи информации АСУ. — М.: Радио и связь, 1983.

47. *Уокерли Дж.* Архитектура и программирование микро-ЭВМ. — М.: Мир., 1984.

48. *Фрибель В. и др.* Программирование микропроцессоров. — М.: Энергоиздат, 1982.

49. *Хильбурн Дж., Джулич П.* Микро-ЭВМ и микропроцессоры. — М.: Мир, 1979.

50. *Шрайбер Т. Дж.* Моделирование на GPSS. — М.: Машиностроение, 1980.

51. *Четвериков В. Н.* Подготовка и телеобработка данных в АСУ. — М.: Высшая школа, 1981.

52. *Янбых Г. Ф., Эттингер Б. Я.* Методы анализа и синтеза сетей ЭВМ. — Л.: Энергия, 1980.

53. *Якубайтис Э. А.* Архитектура вычислительных сетей. — М.: Статистика, 1980.

## ОГЛАВЛЕНИЕ

*Учебное издание*

**Борис Яковлевич Советов, Олег Иванович Кутузов, Юрий Алексеевич Головин, Юрий Вартанович Аветов**

ПРИМЕНЕНИЕ МИКРОПРОЦЕССОРНЫХ СРЕДСТВ В СИСТЕМАХ ПЕРЕДАЧИ ИНФОРМАЦИИ

Зав. редакцией Н. И. Хрусталева. Редактор И. А. Кузьмина. Мл. редактор Е. В. Богуславская. Художник В. И. Мешалкин. Технический редактор Н. А. Битюкова. Корректор Г. А. Чечеткина

ИБ № 6072

Изд. № СТД—509. Сдано в набор 20.11.86. Подп. в печать 24.03.87. Т-08388. Формат 84×108$^1/_{32}$. Бум. тип. № 2. Гарнитура литературная. Печать высокая. Объем 13,44 усл. печ. л. 13,55 усл. кр.-отт. 13,83 уч.-изд. л. Тираж 37 000 экз. Зак. № 716. Цена 50 коп.

Издательство «Высшая школа», 101430, Москва, ГСП-4, Неглинная ул., д. 29/14.

Владимирская типография Союзполиграфпрома при Государственном комитете СССР по делам издательств, полиграфии и книжной торговли 600000, г. Владимир, Октябрьский проспект, д. 7