АКАДЕМИЯ НАУК ССС
СИБИРСКОЕ ОТДЕЛЕНИ

Э. В. ЕВРЕИНОВ, Ю. Г. КОСАРЕ

ОДНОРОДНЫЕ УНИВЕРСАЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ ВЫСОКОЙ ПРОИЗВОДИТЕЛЬНОСТИ

ИЗДАТЕЛЬСТВО «НАУКА» • СИБИРСКОЕ ОТДЕЛЕНИ
НОВОСИБИРСК • 196

Ответственные редакторы:

Канд. физ.-матем. наук *Ю. С. Завьялов*,
канд. техн. наук *Н. Г. Загоруйко*

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

В данной работе излагаются основные результаты исследований авторов по проблеме существенного повышения производительности электронных вычислительных машин. Авторы исходили из следующих основных принципов:

параллельности — основное повышение производительности должно достигаться не путем повышения быстродействия работы физических элементов, а путем параллельного выполнения большого числа операций. Это означает переход от отдельных вычислительных машин к вычислительным системам;

универсальности — вычислительная система должна быть универсальным средством, позволяющим достигать высокой производительности для любых классов задач;

надежности — вычислительная система должна обладать максимально высокой структурной надежностью;

технологичности — при разработке вычислительных систем основной целью должно быть максимальное упрощение технологии изготовления.

Основываясь на этих принципах, авторы пришли к двум дополняющим друг друга идеям:

однородной универсальной вычислительной системы, состоящей из одинаковых и одинаково соединенных друг с другом универсальных вычислительных машин (элементарных машин) и обладающей возможностью программного изменения структуры как системы коммутаций, так и основных параметров элементарных машин.

Создание подобной системы может оказаться выгодным не только для построения вычислительных систем миллиардного диапазона, но и для вычислительных систем меньшей производительности. В последнем случае в качестве элементарных машин могут служить универсальные ЭВМ, изготовленные из обычных

стандартных элементов, используемых в современной вычислительной технике;

однородной вычислительной среды, состоящей из одинаковых и одинаково соединенных друг с другом универсальных элементов, программно настраивающихся на выполнение любой логической функции из полного набора логических функций, памяти и любого соединения со своими соседями, как основы микроструктуры элементарных машин.

Работа подразделена на восемь глав. В первой рассматривается необходимость разрабатывать средства вычислительной техники высокой производительности. Во второй главе обсуждаются пути повышения производительности вычислительной техники и обосновывается принцип параллельности. В третьей и четвертой главах для обоснования принципа универсальности рассматриваются описанные в литературе проекты проблемно ориентированных вычислительных систем и существующие универсальные вычислительные системы. В пятой главе рассматриваются универсальные вычислительные системы с программно изменяемой структурой, в шестой — свойства вычислительных сред. В седьмой главе обсуждаются особенности решения задач на универсальных вычислительных системах, рассматривается алгоритмический язык для описания алгоритмов решения задач. В восьмой главе приводятся примеры решения основных классов вычислительных задач.

Авторы считают своим долгом выразить признательность академику С. Л. Соболеву за поддержку работ по вычислительным системам высокой производительности в Институте математики СО АН СССР, академикам А. А. Дородницыну, В. М. Глушкову и канд. техн. наук В. К. Левину, со стороны которых авторы всегда встречали поддержку и внимание к своей работе, акад. С. А. Лебедеву, чл.-кор. А. А. Маркову, докторам физ.-матем. наук Ю. Г. Решетняку и Ю. И. Журавлеву, канд. физ.-матем. наук Ю. С. Завьялову, канд. техн. наук Н. Г. Загоруйко за полезные обсуждения.

Авторы также выражают глубокую благодарность коллективу Отделения вычислительной техники Института математики СО АН СССР, в котором выполнялась данная работа.

Глава 1

ОСНОВНЫЕ ЧЕРТЫ ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН ВЫСОКОЙ ПРОИЗВОДИТЕЛЬНОСТИ

1.1. Вычислительная техника как основа научно-технического прогресса

Применение электронных вычислительных машин (ЭВМ) во многих областях науки и техники позволило увеличить как темпы, так и экономичность многих научных исследований, производственных и других процессов. Поэтому за последние годы появилось много новых областей применения вычислительной техники. Так, наряду с применением ЭВМ для научных расчетов, управления производственными процессами, диспетчеризации, статистики, экономических исследований ЭВМ начинают применяться в медицине (для установления диагноза и наиболее эффективной стратегии лечения, для управления аппаратурой, поддерживающей и контролирующей состояние организма больного в ходе операции) [1]; в юриспруденции и криминалистике (для установления личности, анализа улик, выявления противоречий в действующих законах и т. п.) [2]; в исторических науках (для анализа неизвестных письменностей, археологических памятников, в историко-экономических исследованиях и т. п.) [3—6].

Электронные вычислительные машины начали играть большую роль в обучении в качестве тренажера, экзаменатора, лектора, что позволяет сократить сроки обучения благодаря учету индивидуальных особенностей каждого обучающегося [7, 8].

В исследованиях процессов творчества ЭВМ используются для установления особенностей творческих стилей писателей [9—11] и композиторов [12—13], для моделирования процессов научного мышления (например, доказательства математических теорем) [14], для исследования процессов обучения и самообучения [15—18] и т. п., что позволяет глубже проникнуть в тайны творчества и дает один из подходов к пониманию законов мышления.

Этот перечень можно было бы продолжить. С ростом числа областей применения ЭВМ (рис. 1) происходит процесс более глубокого внедрения ЭВМ в уже известные области применения [19]. Характерен переход от решения отдельных задач к решению на ЭВМ целых комплексов вопросов, составляющих существо той или иной проблемы. Например, в области проектирования ЭВМ, если в конце 50-х годов машины использовались только для составления монтажных схем, то теперь ЭВМ применяются на всех этапах проектирования от выбора варианта путем моделирования до выдачи заводской документации [20]. В области экономики решаются не только отдельные задачи по данному предприятию, отдельной отрасли производства, но и задачи в целом по экономическому району, одной или нескольким странам. Аналогичное положение и в области автоматизации производства.

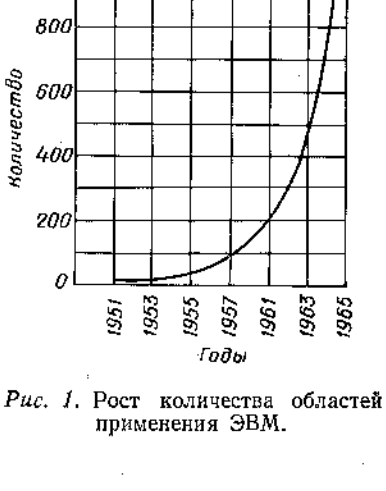


Рис. 1. Рост количества областей применения ЭВМ.

Нужно отметить, что появление ЭВМ способствовало возникновению новых областей науки. Так, автоматизация перевода с одного языка на другой, процессов обучения и других работ, где требуются большие объемы памятей и сложные процессы обработки информации, стала возможной только после появления ЭВМ.

Во многих важнейших областях, например в управлении народным хозяйством страны, при существующих темпах роста производства уже нельзя обходиться без применения ЭВМ [21], так как существующий аппарат, работающий в сфере планирования и управления экономикой, способен выполнить в год работу по переработке информации, которая эквивалентна около 10^{12} арифметическим операциям, что примерно в 10 000 раз меньше числа операций, требуемых для оптимального управления экономикой [22, 23].

В таком же положении находится и научно-техническая информация. Число работ по всем отраслям знаний увеличивается во все возрастающем темпе. Даже по узкой области науки или техники специалисту без ЭВМ зачастую труднее найти готовое

решение в литературе (если даже известно, что оно есть), чем провести самостоятельно исследование или конструктивную разработку [24].

Без применения ЭВМ трудно ожидать прогресса в некоторых областях науки. Это прежде всего области, связанные с исследованием быстропротекающих процессов, управление которыми требует выполнения сложных алгоритмов, в частности исследование явлений микромира, биологических и других объектов, для которых характерны необратимые процессы с трудновоспроизводимыми начальными условиями.

Электронно-вычислительные машины кроме универсальных свойств, обеспечивающих их широкое применение, обладают высокой экономической эффективностью. Затраты на ЭВМ, составляя малую долю общих затрат в той области, где они применяются, существенно повышают эффективность основных затрат. Так, применение ЭВМ в металлургии, где их стоимость составляет ничтожную долю от стоимости домен, конверторов, прокатных станов и другого оборудования, окупается в первые же месяцы эксплуатации [25].

Аналогично обстоит дело с применением ЭВМ в энергетике. Например, применение ЭВМ на электростанции средней мощности дает чистую экономию в 1 250 000 долларов [19]. Еще больший эффект дает применение ЭВМ для управления электрическими сетями, объединяющими электростанции экономического района, страны или нескольких стран.

Большую экономию дает применение ЭВМ в связи. Например, применение ЭВМ для автоматизации телеграфной связи позволяет увеличивать пропускную способность имеющихся каналов связи более чем в 100 раз и сократить дорогостоящее строительство новых линий связи [26].

Применение ЭВМ на транспорте позволяет существенно улучшить план перевозок [27].

В научных исследованиях ЭВМ позволяют уменьшить затраты на эксперименты за счет расчетов и моделирования, а также сократить сроки разработок, что позволяет более эффективно использовать научные силы [28, 29].

Не удивительно, что по темпам роста вычислительная техника намного опережает другие отрасли народного хозяйства. Среднегодовые темпы роста составляют примерно 50—100% [19].

Интересно проследить динамику развития вычислительной техники по наиболее развитым капиталистическим странам.

Ведущее положение среди капиталистических стран в области производства и использования ЭВМ занимают США, где эксплуатируется более 12 000 универсальных ЭВМ [30]. Объем про-

дажи ЭВМ в США резко растет (рис. 2) и в 1965 г. составил около 2 млрд. долларов [19].

США гораздо раньше других стран начали развивать вычислительную технику. Еще в 1960 г. в этой области работало свыше 1 200 000 чел., а число фирм, занятых производством ЭВМ, было более 600 [19]. Мощность парка ЭВМ в США растет по кривой, близкой к экспоненте (рис. 3).

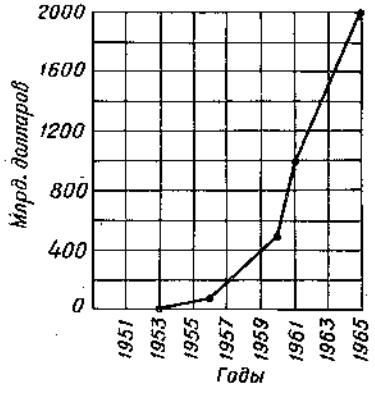


Рис. 2. Рост объема продажи ЭВМ в США.

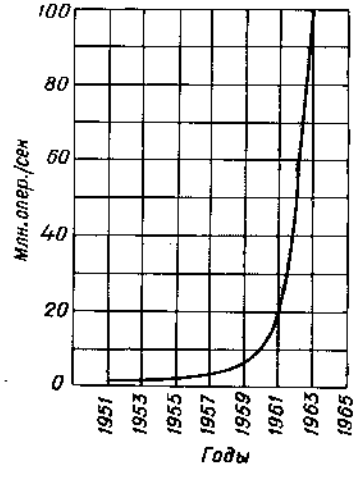


Рис. 3. Мощность парка ЭВМ в США.

Другие капиталистические страны, начавшие развивать вычислительную технику с запозданием, за последние годы резко увеличили производство ЭВМ. Англия и Франция по темпам развития вычислительной техники догнали США, а Япония и ФРГ имеют даже более высокие темпы [19].

Итак, применение вычислительной техники имеет следующие положительные качества:

1) оказывается весьма эффективным во всех областях науки и отраслях народного хозяйства;

2) вызывает появление новых важных областей науки и техники;

3) дает возможность развиваться некоторым областям науки и народного хозяйства, зашедшим в тупик;

4) дает большой экономический эффект и быструю окупаемость.

Дальнейший научно-технический прогресс был бы не мыслим без вычислительной техники.

Универсальность ЭВМ общего назначения — одна из их особенностей как инструмента, определяющего темпы научно-технического прогресса. Это позволяет применять одну и ту же ЭВМ в различных областях хозяйства и науки и для разных целей, легко сочетать работу ЭВМ с другими объектами, обеспечивая высокую динамичность при быстро изменяющихся условиях работы.

Применение электронных вычислительных машин благодаря общности подхода и использованию единого формального аппарата для описания и решения различных задач способствует также взаимному проникновению и слиянию наук и отраслей техники.

Одним из основных показателей уровня развития той или иной современной страны служит суммарная мощность ее парка ЭВМ аналогично тому, как энергетическая вооруженность оценивается по суммарной мощности всех энергетических установок. Иногда это называют *интеллектуальной мощностью* государства и оценивают суммарным числом операций (типа сложения двух чисел), производимых в секунду.

Области применения ЭВМ в зависимости от трудоемкости задач можно разделить на две группы.

I. Массовые проблемы со сравнительно небольшим объемом вычислений, такие как задачи управления производственными процессами, задачи, связанные с проектно-конструкторскими работами, большинство научных, экономических задач и задач управления народным хозяйством и т. п.

Для успешного развития этой группы задач важна *суммарная производительность всего парка ЭВМ*.

II. Проблемы, которые требуют для своего решения выполнения огромного числа операций: сложные научно-технические задачи, задачи ядерной физики, задачи, связанные с освоением космоса, динамической метеорологией, проектированием с выбором оптимального варианта и т. п. Успех в решении проблем второй группы зависит от достигнутого уровня производительности ЭВМ. При этом производительность ЭВМ определяет не только круг задач, поддающихся решению, но и общие темпы научных исследований. Дело в том, что основные усилия исследователей тратятся, как правило, на создание методов, которые сводили бы решение данной задачи к определенному числу операций, поддающемуся выполнению на имеющейся технике.

По мере проникновения в глубины атомного ядра, космоса, тайны живой природы, процессов мышления и т. п. потребуется решать все более и более сложные задачи. Поэтому *повышение производительности ЭВМ как незаменимого инструмента познания будет всегда одной из основных проблем науки и техники.*

1.2. О необходимости значительного повышения производительности ЭВМ

По-видимому, всегда будут возникать задачи, для которых существующий уровень вычислительной техники будет недостаточен. Это прежде всего те, которые определяют дальнейшее развитие науки. Еще в 1956 г. М. В. Келдыш, А. А. Ляпунов и М. Р. Шура-Бура [31], отмечая данное положение, указывали, что решение двумерных и трехмерных задач математической физики, в частности трехмерных задач газовой динамики и расчет гетерогенных котлов, требуют создания ЭВМ с быстродействием в несколько миллионов операций в секунду, что на два-три порядка превышало быстродействие лучших образцов ЭВМ того времени.

В 1961 г. на основании анализа имеющихся задач академик А. А. Дородницын пришел к выводу, что необходимо создавать ЭВМ с быстродействием в сотни миллионов операций в секунду, что опять-таки превышает параметры лучших образцов машин того времени на два-три порядка [32].

Опыт последующих лет все больше подтверждает этот вывод. Так, для решения задачи оптимального управления экономикой требуется выполнять 10^{18} операций в год, что соответствует примерно 10^9 операций в секунду [23].

Следовательно, уже сейчас необходимы средства вычислительной техники с производительностью 10^9 *опер/сек*.

Производительность вычислительной техники определяется не только быстродействием, т. е. чистой скоростью выполнения операций, но и другими факторами.

Нередко ЭВМ с меньшим быстродействием, но с большим объемом памяти, большей скоростью обмена информацией, большей надежностью или более удачно выбранным списком команд имеет большую производительность, чем более быстродействующая ЭВМ. Наиболее полное определение производительности ЭВМ с учетом влияния различных факторов (универсальный критерий эффективности) дан академиком В. М. Глушковым [33].

Между быстродействием и другими параметрами, определяющими производительность ЭВМ, существуют определенные зависимости, характер которых трудно поддается точной оценке, так как он в сильной мере определяется классами рассматриваемых задач.

Попытаемся оценить нижнюю границу значения указанных параметров на основе рассмотрения некоторых классов задач.

1. *Объем памяти ЭВМ* для данной задачи определяется максимальным объемом информации, которую надо одновременно

хранить в ЭВМ в ходе решения задачи. Эта информация состоит из программы вычислений, исходных, промежуточных и результирующих данных. Удобно измерять этот объем общим количеством двоичных разрядов.

Каждая задача характеризуется, прежде всего, общим числом операций L, которое надо выполнить для получения решения, и наибольшим объемом информации v в двоичных единицах, который надо хранить в ходе решения задачи. Отношение между этими параметрами, которое характеризует среднее число операций, приходящееся на единицу информации, назовем *коэффициентом активности информации*:

$$\alpha = \frac{L}{v}\left[\frac{опер}{дв.\ ед.}\right]. \tag{1.1}$$

Рассмотрим, как изменяется величина α с ростом L и v для различных задач. Так как на получение точной зависимости трудно надеяться без детального анализа задач, то ограничимся асимптотическими оценками. При $L, v \to \infty$ возможны три случая:

$$\alpha \to \infty;$$

$$\alpha \to 0;$$

$$c_1 \leqslant \alpha \leqslant c_2,$$

где c_1 и c_2 — константы.

В первом случае рост числа операций обгоняет рост объема информации (например, при решении задач комбинаторного анализа методом перебора; при решении задач методом Монте-Карло с высокой точностью; при вычислении многомерных интегралов с помощью кубатурных формул и т. п.).

Во втором случае число операций растет медленнее, чем объем информации (например, при решении задач типа машинного перевода со словарями большого объема; при решении задач, связанных с выбором элемента из конечного упорядоченного множества и т. п.).

В третьем случае рост числа операций и объема информации имеет один и тот же порядок величины (например, при решении уравнений в частных производных методом сеток; при решении задач распознавания образов; решении систем обыкновенных дифференциальных уравнений и т. п.).

Для второго и третьего случаев объем информации играет существенную роль. Для некоторых задач, относящихся к первому случаю, объем информации (хотя и будет расти медленнее,

чем число операций) также может составить значительную величину. Так, в задачах линейной алгебры (умножение матриц, обращение матриц и решение систем линейных уравнений точными методами и т. п.) с ростом их порядка число операций растет пропорционально кубу, а объем информации — квадрату, т. е. α растет линейно.

Таким образом, для большинства задач увеличение быстродействия ЭВМ в широких пределах должно сопровождаться и ростом объема запоминающих устройств.

Для каждой ЭВМ имеется предельное число операций

$$N_{\text{пр}} = Pt_{\text{пр}}, \tag{1.2}$$

где P — производительность ЭВМ;

$t_{\text{пр}}$ — допустимое время решения задачи,

и предельная величина хранимой информации $v_{\text{пр}}$.

Пусть $N_{\text{пр}} = 10^{14}$ операций, что соответствует производительности ЭВМ порядка 10^9 *опер/сек* (при этом предполагается, что $t_{\text{пр}} = 24$ *час*). Тогда для наиболее массовых задач, таких как решение систем обыкновенных дифференциальных уравнений, дифференциальных уравнений в частных производных и многих других, $\alpha \approx 10^2 - 10^3$ *опер/дв. ед.*, что соответствует при производительности 10^9 *опер/сек* объему памяти порядка 10^{11} *дв. ед.* Примерно такое же отношение наблюдается и у существующих машин. Сделанное выше рассмотрение позволяет предполагать, что это соотношение должно сохраниться и для будущих высокопроизводительных ЭВМ.

2. Установим, каким должно быть соотношение между быстродействием, объемом памяти и *скоростью ввода и вывода* информации. При определении этого соотношения будем исходить из требования, чтобы время, затрачиваемое на ввод и вывод, не превышало основного времени работы. При $t_{\text{пр}} = 24$ *час*, $P = 10^9$ *опер/сек*, $v_{\text{пр}} = 10^{11}$ *дв. ед.* ввод — вывод информации для заполнения всего объема памяти должен выполняться со скоростью около 10^6 *дв. ед./сек*. Эта скорость не чрезмерна и может быть достигнута даже с помощью имеющихся средств, например магнитных лент или оптических устройств. В некоторых задачах, например для распознавания образов, может потребоваться большая скорость ввода—вывода (около $5 \times 10^6 - 10^7$ *дв. ед./сек*).

Ввод и вывод большого объема информации потребует непосредственного обмена с активными источниками информации, такими как другие ЭВМ, человек, источники зрительных, звуковых и других образов, т. е. требуется, чтобы ЭВМ могла непосредственно принимать и выдавать оптические, звуковые и электромагнитные сигналы в реальном масштабе времени.

Сейчас ведутся активные работы по использованию этих средств для уже существующих ЭВМ [34, 35].

3. С увеличением числа элементов и быстродействия резко растут требования к *надежности ЭВМ*.

Следует отметить, что при большом числе элементов (10^{11} и выше) уже трудно рассчитывать на ремонт ЭВМ с заменой неисправных элементов. В принципе возможны два подхода построения надежных неремонтируемых ЭВМ:

а) использование элементов с высокой надежностью; при этом требуется, чтобы в течение нескольких лет не вышел из строя ни один элемент (считается, что выход хотя бы одного элемента ведет к необратимой порче ЭВМ); на создание таких элементов, по-видимому, трудно рассчитывать;

б) применение методов построения надежных схем из ненадежных элементов, предложенных Д. Нейманом и К. Шенноном [36, 37]. В этом случае повышение надежности достигается за счет избыточного числа элементов. Достаточная надежность достигается при увеличении числа элементов по оценкам К. Шеннона не менее чем на 2 порядка, по оценкам Д. Неймана — не менее чем на 4—5 порядков.

4. Высокое быстродействие может не дать ожидаемого эффекта при решении конкретной задачи, если ЭВМ будет иметь неудачный для этой цели *набор команд*, т. е. здесь сохраняется то же противоречие между стремлением сделать набор операций универсальным и требованием учета специфики каждой задачи, что и у существующих универсальных машин. Здесь возможны два подхода: либо путем повышения быстродействия компенсировать завышенный расход операций при фиксированном наборе команд, либо строить ЭВМ с переменным набором команд.

5. Весьма важно оценить *экономические затраты* на создание ЭВМ указанной выше производительности 10^9 *опер/сек*.

Если предположить, что стоимость такой ЭВМ будет возрастать пропорционально росту производительности, т. е. сохранится отношение производительности существующих машин к затратам на их изготовление (критерий цены эффективности быстродействия по В. М. Глушкову [33]), то для изготовления такой ЭВМ потребуется сумма, сравнимая с годовым бюджетом страны. Следовательно, при построении подобных ЭВМ учет экономических факторов приобретает первостепенное значение. При создании такой ЭВМ необходимо удешевлять конструкции элементов, уменьшать технологические затраты и количество расходуемого материала, снижать общую стоимость разработки. Эта задача может быть решена на основе микроминиатюризации и применения ЭВМ на всех стадиях разработки и изготовления [38].

Укажем основные параметры ЭВМ высокой производительности:

1) быстродействие не ниже 10^9 *опер/сек*;

2) объем памяти не менее 10^{10}—10^{11} *дв. ед.*;

3) скорость ввода и вывода 10^6—10^7 *дв. ед./сек*. Должен быть также предусмотрен непосредственный ввод и вывод зрительных, звуковых и электромагнитных сигналов;

4) система команд должна меняться в зависимости от класса решаемой задачи;

5) надежность должна обеспечить непрерывную работу в течение 3—5 лет;

6) конструктивно ЭВМ должна быть выполнена на основе микроминиатюризации и приспособлена для полностью автоматизированной технологии.

Построение подобной ЭВМ — проблема сложнейшая, которая не может быть решена в короткий срок. Оценить точный срок подобной разработки трудно. Для создания современных ЭВМ требуется 3—5 лет, для разработки данной ЭВМ потребуется времени по крайней мере в 2—3 раза больше.

Работы в этой области в некоторых странах уже идут полным ходом.

В США на протяжении многих лет ведутся работы над проектом «Молния», результатом которого должно явиться создание ЭВМ с быстродействием 10^8 *опер/сек* [39]. В фирме «Вестингауз Электрик Корпорэйшен» разрабатывается система СОЛОМОН с быстродействием при решении некоторых задач 10^8 *опер/сек* [40]. В Стенфордском научно-исследовательском институте ведутся разработки ЭВМ с быстродействием 10^{10}—10^{11} *опер/сек* и объемом памяти 10^{12} *дв. ед.* [41].

Решение такой сложной проблемы требует уделить особое внимание методике разработки. По своему характеру эта проблема относится к сложным системам [42] и должна опираться на современные методы планирования [43—45] и другие современные средства исследования, основанные на применении ЭВМ на всех стадиях разработки.

Глава 2

ОСНОВНЫЕ ПУТИ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ ЭВМ

2.1. Увеличение быстродействия элементов

Современные ЭВМ обычно состоят из наборов стандартных элементов трех типов: 1) реализующих элементарные функции, образующие полный набор логических функций, и временну́ю задержку сигналов; 2) формирующих, приводящих сигналы к стандартной форме; 3) элементов памяти, служащих для хранения информации.

Существует много наборов стандартных элементов, отличающихся как по своим параметрам, так и по используемым физическим явлениям. Рассмотрим основные характеристики стандартных элементов: быстродействие, надежность, потребляемую мощность, габариты и стоимость.

Под быстродействием стандартного элемента (или его тактовой частотой) понимается максимальное число переключений в секунду в рабочем режиме.

Для определения надежности существуют два основных подхода: а) надежность характеризуется вероятностью безотказной работы в течение заданного времени p; б) средним временем работы между двумя отказами в часах $t_{ср}$ или его обратной величиной — средней частотой отказа λ.

Рассмотрим теперь, какие параметры элементов должны быть у ЭВМ высокой производительности, если идти только по пути повышения быстродействия элементов.

Тактовая частота работы элементов ЭВМ должна быть примерно на один-два порядка выше быстродействия ЭВМ, измеряемого в операциях в секунду. Это объясняется тем, что каждая операция выполняется за определенное число последовательных тактов, которое тем больше, чем сложнее операция. Таким образом, тактовая частота работы элементов в нашем случае должна быть не менее 10^{10}—10^{11} переключений в секунду.

Надежность элементов из расчета на среднее время работы ЭВМ без повреждений 3—5 лет, при общем числе элементов 10^{10}—10^{11} должна соответствовать $\lambda \approx 10^{-14}$ —10^{-16} 1/*час*.

При повышении тактовой частоты работы элементов начинают играть первостепенную роль их габариты. Как известно, между тактовой частотой работы ЭВМ и ее размерами существует определенная зависимость, обусловленная запаздыванием сигналов из-за конечной скорости их распространения. Чтобы избежать учета запаздывания сигналов в линиях связей внутри машины, необходимо выполнить соотношение

$$d_{\text{пр}} \ll \frac{c}{\nu}, \tag{2.1}$$

где $d_{\text{пр}}$ — наибольшая длина связей внутри машины;

c/ν — длина волны, соответствующая тактовой частоте работы элементов ν;

c — скорость света, или

$$d_{\text{пр}} = \frac{1}{\gamma} \cdot \frac{c}{\nu}, \tag{2.2}$$

где γ — коэффициент превышения длины волны над длиной связи. Величина этого коэффициента характеризует допустимую относительную величину фазового сдвига в линиях связей машины. Обычно $\gamma = 10$—100. От предельной длины связей не трудно перейти к предельному объему машины

$$V_{\text{пр}} = a d_{\text{пр}}^3 = a \left(\frac{c}{\gamma}\right)^3 \frac{1}{\nu^3}, \tag{2.3}$$

где a — коэффициент, учитывающий геометрическую конфигурацию машины.

Средний объем элемента

$$v_{\text{эл}} = h \frac{V_{\text{пр}}}{n}, \tag{2.4}$$

где n — общее число элементов машины;

h — коэффициент, учитывающий относительные затраты объема на элементы.

Величина h лежит в пределах 0,1—0,01. Основные затраты объема машины приходятся на каналы связи.

Для заданных параметров ЭВМ предельные линейные размеры машины $d_{\text{пр}} = 0{,}3$—$0{,}03$ *см*, предельный объем ЭВМ $V_{\text{пр}} = 10^{-2}$ —10^{-5} *см*^{3}, а предельный объем элемента $v_{\text{пр}} = 10^{-14}$—10^{-20} *см*^{3}, т. е. плотность упаковки элементов с учетом связей должна быть 10^{13}—10^{18} 1/*см*^{3}.

Рассмотрим теперь, какие же параметры у элементов современных машин. Для существующих машин характерно применение ламповых, полупроводниковых и магнитных элементов. В серийных машинах достигнуты следующие уровни быстродействия элементов: для магнитных — несколько сот тысяч переключений в секунду, для ламповых — несколько миллионов, для полупроводниковых — несколько десятков миллионов.

Существующая технология изготовления этих элементов не позволяет надеяться на заметное увеличение быстродействия. К тому же они не обладают высокой надежностью. Средняя частота отказа λ у ламповых элементов 10^{-3}—10^{-4} 1/*час*, у полупроводниковых и магнитных—10^{-4}—10^{-6} 1/*час* [1, 2].

Укажем примерные габариты существующих элементов. Ламповые элементы занимают объем 500—600 *см*^{3}, полупроводниковые—50—60 *см*^{3}, магнитные элементы функциональные и формирующие — 25—50 *см*^{3}, памяти — 0,001 *см*^{3}. Потребляемая мощность ламповых элементов составляет десятки ватт, у магнитных — от сотен милливатт до единиц ватт, у полупроводниковых — десятки милливатт.

Ориентировочная стоимость ламповых элементов 5—10 руб., полупроводниковых — 10—20, магнитных функциональных и формирующих — 1—3, памяти — 0,1—0,2 руб. Существующие элементы весьма далеки по своим параметрам от требуемых.

Достигнутый уровень быстродействия отдельных образцов новых элементов составляет 10^8—10^9 переключений в секунду. Разрабатываются они с ориентацией на микроминиатюризацию, которая важна не только сама по себе как способ борьбы с запаздыванием сигналов, но и как средство для повышения надежности. Дело в том, что микроминиатюризация сопровождается, как правило, полной автоматизацией изготовления как самих стандартных элементов, так и их связей. Это ведет к уменьшению разброса параметров и исключению соединений типа пáек или разъемных контактов, являющихся одним из основных источников неисправностей. Кроме того, микроминиатюрные размеры элементов и ЭВМ в целом позволяют решить вопрос о полной герметизации машины.

Количественных данных, позволяющих оценить возможное повышение надежности в результате микроминиатюризации, пока нет. Однако можно предположить, что величина λ уменьшится на несколько порядков.

Достижимые плотности упаковки деталей (триодов, диодов, сопротивлений, емкостей и т. п.) в зависимости от технологии

изготовления составляют 30—3000 *дет./см*^{3} [3—9]. Плотность упаковки элементов будет соответственно 3—300 *элем./см*^{3}.

Применение микроминиатюрных элементов кроме разработки технологии усложняется трудностями получения элементов с малой мощностью рассеяния. Минимально достигнутые мощности рассеяния даже для частот переключения 10^7—10^8 для полупроводниковых элементов составляют 10^{-3} *вт*, у туннельных при 10^8—10^9 *перекл./сек* — 10^{-6} *вт* [10].

Эти элементы пока намного дороже обычных, однако можно надеяться, что по мере автоматизации производства их стоимость будет снижаться и со временем станет намного ниже, чем у существующих элементов.

Таким образом, новые элементы (даже с учетом ближайших перспектив их развития) не удовлетворяют сформулированным выше требованиям к элементам ЭВМ высокой производительности в случае повышения быстродействия ЭВМ только путем увеличения тактовой частоты работы.

Следует отметить, что указанные выше параметры новых элементов отражают больше современное состояние исследований в этой области, чем принципиально возможные значения параметров элементов.

За последние годы были сделаны попытки установить верхние границы значений некоторых параметров на основании общих законов физики.

В работе [11] даны оценки предельно возможного быстродействия элементов на основании принципа неопределенности Гейзенберга и конечности скорости света. Автор по предельно допустимой величине запаздывания сигнала установил предельный объем элемента и подсчитал, что для максимальных значений плотности вещества время переключения системы из двух элементов будет 10^{-26} *сек*.

Эта величина с ростом числа элементов будет увеличиваться.

В работе [12] была сделана попытка оценить минимальные размеры двоичного элемента памяти, устойчиво хранящего информацию в течение длительного времени (100 лет). На основании предположения о том, что причиной разрушения информации служат тепловые флюктуации, автор пришел к выводу, что каждый такой элемент должен состоять не менее чем из 100 независимых физических единиц (диполей, спинов и т. п.).

Оценка минимального уровня сигнала для переключения элемента дана в работе [13]. Автор исходит из требования, чтобы уровень сигнала превышал уровень тепловых шумов, равный kT.

Разрыв между этими оценками и достигнутыми значениями параметров может быть объяснен как слишком грубыми теоретическими оценками, так и недостаточным уровнем практических исследований в этой области. Важно отметить, что теоретические оценки делались для каждого параметра в отдельности. Вполне понятно, что комплексное рассмотрение этих параметров может значительно снизить теоретически достижимый уровень быстродействия и микроминиатюризации. В частности, оценка быстродействия элемента (а следовательно, величины энергии, подводимой для изменения его состояния) при одновременном учете предельного уровня энергии, рассеиваемой единицей объема, и взаимодействия многих элементов (а не только двух) может снизить оценку предельно допустимого быстродействия и сократить разрыв между теоретическими и практическими значениями.

Не исключено также, что комплекс требований к параметрам элементов, возникающий в связи с необходимостью достигнуть высокую производительность путем увеличения быстродействия элементов, окажется если и не превышающим теоретически достижимый предел, то близким к нему. Возможно, этим объясняется трудность в создании подобных элементов. Можно с достаточной уверенностью сказать, что в ближайшие годы подобные элементы вряд ли будут созданы.

2.2. Увеличение числа элементов

Вторая тенденция повышения производительности ЭВМ состоит в стремлении к совмещению во времени возможно большего числа частей вычислительного процесса. Это совмещение достигается путем увеличения числа элементов. При этом предполагается, конечно, что имеющиеся элементы распределены между частями ЭВМ оптимальным образом, т. е. никаким перераспределением элементов между частями ЭВМ (при постоянстве общего числа элементов) нельзя уменьшить общее время решения задач.

Установим требования к уровню совмещения во времени частей вычислительного процесса для достижения заданной производительности. Если предположить, что элементарная операция типа сложения двух одноразрядных чисел выполняется за один рабочий такт (как у машин последовательного действия), то производительность ЭВМ может быть представлена в виде

$$P = \frac{L_9}{m} \nu, \qquad (2.5)$$

где ν — тактовая частота работы элементов ЭВМ;

m — количество двоичных разрядов в числах, над которыми производятся операции;

$L_э$ — среднее число совмещенных элементарных операций.

Из этого соотношения следует, что для достижения производительности 10^9 *опер/сек* при $\nu = 10^6$—10^7 *1/сек* и $m = 50$ требуемая величина $L_э \approx 5\times10^3$—5×10^4.

Тенденция совмещения проявляется на различных уровнях.

Совмещение микроопераций внутри операций. Все операции, содержащиеся в списке команд машины, являются, как правило, сложными и могут быть разбиты на более мелкие части, называемые обычно микрооперациями. В каждой ЭВМ имеется свой набор микроопераций, из комбинаций которых составлены все команды [14—16]. Микрооперации могут быть довольно сложными. Для наших целей рассмотрим простейший набор микроопераций, в который включим элементарные операции типа арифметических и логических операций над одноразрядными двоичными числами. Будем считать, что каждая такая микрооперация выполняется за один микротакт. Далее под микрооперацией будем понимать только микрооперации данного типа. Любую операцию можно характеризовать общим числом микроопераций, которые необходимо выполнить для ее реализации. Эта величина может служить количественной оценкой сложности операции. Так, операция сложения двух m-разрядных двоичных чисел будет иметь сложность m. Умножение тех же чисел — m^2 и т. д. Без совмещения каждая операция может быть выполнена за число микротактов, равное сложности.

Путем совмещения микроопераций за счет пропорционального увеличения оборудования можно сократить число микротактов. Предельный выигрыш во времени будет, очевидно, достигнут при полном совмещении микроопераций, т. е. когда каждая операция выполнится за один микротакт. При этом для каждой операции оборудование должно быть увеличено пропорционально ее сложности, т. е. для операции сложения двух m-разрядных чисел оборудование должно быть увеличено в m раз, для умножения этих чисел — в m^2 раз и т. д.

Таким образом, затраты оборудования будут определяться в основном операциями с наибольшей сложностью g_{max}. Выигрыш во времени будет пропорционален средней величине сложности операций:

$$g = \sum_i p_i g_i, \qquad (2.6)$$

где p_i — относительная частота встречаемости i-й операции;

g_i — сложность этой операции.

Как правило, чем сложнее операция, тем она реже встречается, поэтому g будет существенно меньше g_{max}.

Совмещение микроопераций широко используется в современных ЭВМ. Как правило, применяется m-кратное совмещение, где m — число двоичных разрядов в коде. Совмещение большей кратности допускается редко, так как при этом выигрыш во времени достигается ценой непропорционально больших затрат оборудования. Это объясняется тем, что наиболее часто встречающиеся операции имеют сложность порядка m и время их выполнения не изменяется при увеличении оборудования более чем в m раз.

Следовательно, совмещение микроопераций выше того уровня, который достигнут в современных ЭВМ, хотя и может дать некоторое увеличение производительности, но это увеличение не того порядка, который требуется для создания ЭВМ высокой производительности.

Совмещение частей различных операций. Это совмещение широко используется в современных ЭВМ. Например, одновременно с выполнением одной операции проводятся подготовительные микрооперации следующей операции: выборка команды (частичное совмещение) либо выборка исходных чисел (полное совмещение) [17, 18]. Это совмещение выполняется в основном путем лучшего использования имеющегося оборудования блоков ЭВМ, которые обычно устроены так, что различные типы микроопераций выполняются различными блоками. Таким путем можно получить выигрыш в несколько раз.

Совмещение операций. Аналогично совмещению частей различных операций в некоторых ЭВМ совмещаются операции, выполняемые различными блоками. Например, арифметические операции совмещаются с логическими операциями, управляющими ходом вычислений, с обменом кодов между памятями, с вводом и выводом информации и т. п. [19]. При этом выигрыш во времени получается только при определенных последовательностях операций и вряд ли может быть сделан больше чем в несколько раз.

Совмещение решения нескольких задач или частей одной задачи. Подобное совмещение наблюдается у наиболее производительных современных машин [20—22]. Эти машины состоят из нескольких ЭВМ, каждая из которых имеет свою память, свой набор команд, арифметический блок и блок управления, что позволяет каждой ЭВМ работать автономно над решением своей час-

ти задачи. Эти ЭВМ обычно специализированы в одной из трех областей: выполнение арифметических и логических операций; переработка информации, управляющей ходом вычислений; обмен информацией с внешними объектами и простейшая ее переработка.

Такая специализация дает экономию в оборудовании. Выигрыш во времени примерно пропорционален числу ЭВМ, из которых состоит данная машина. Увеличение числа ЭВМ позволяет уменьшить время решения задач. Однако выигрыш во времени зависит от структуры задачи и обусловливает плохое использование оборудования при решении широкого круга задач. Помимо увеличения оборудования это направление ведет к усложнению проектирования, изготовления и эксплуатации, обусловленного в основном разнородностью как самих ЭВМ, так и их блоков, что препятствует созданию ЭВМ высокой производительности на этой основе.

При решении какой-либо массовой задачи или узкого класса задач можно добиться повышения производительности, отказавшись от универсальности и перейдя к построению специализированной машины, где для каждой операции или группы операций отводится специальное устройство. Этим достигается одновременное выполнение большого числа операций. Производительность специализированных ЭВМ обычно на один-два порядка выше, чем универсальных машин с тем же объемом оборудования [23, 24].

При существующем уровне вычислительной техники построение специализированных машин зачастую оказывается невыгодным, так как за время их разработки и изготовления нередко изменяются условия задач либо возникают более эффективные методы решения этих задач на универсальных электронных вычислительных машинах.

Следует отметить, что при повышении производительности путем совмещения операций приходится сталкиваться со следующей принципиальной трудностью. Совмещать во времени можно только независимые операции, каждую из которых можно выполнять, не ожидая результатов другой. Таким образом, повышение производительности путем параллельного выполнения большого числа операций зависит от числа независимых операций при решении данной задачи, которое, в свою очередь, зависит от применяемого алгоритма.

В связи с этим возникает новое направление в теории алгоритмов — разработка так называемых параллельных алгоритмов (см. главу 7).

2.3. Сравнение путей увеличения производительности

В начале данной главы было введено понятие о предельном объеме ЭВМ для заданной частоты работы элементов. От предельного объема нетрудно перейти к предельному числу элементов ЭВМ:

$$n_{\text{пр}} = \rho V_{\text{пр}}, \tag{2.7}$$

где ρ — максимально допустимое число элементов в единице объема, определяемое уровнем технологии.

На основании (2.3)

$$n_{\text{пр}} = \rho a \left(\frac{c}{\gamma}\right)^3 \frac{1}{v^3}, \tag{2.8}$$

или

$$n_{\text{пр}} = A \frac{1}{v^3}, \tag{2.9}$$

где через A обозначена часть выражения, не зависящая от рабочей частоты.

Производительность ЭВМ можно считать пропорциональной общему числу n элементов ЭВМ. При этом предположении оценка величины производительности не завышается. Действительно, если, например, число элементов увеличивается в k раз, то можно считать, что число машин, а значит, и производительность увеличиваются в k раз.

Кроме того, производительность прямо пропорциональна рабочей частоте, т. е. можно считать, что

$$P = bnv, \tag{2.10}$$

где b — коэффициент пропорциональности.

На основании (2.9) и (2.10) получаем, что предельно возможная производительность ЭВМ может быть увеличена путем уменьшения частоты при одновременном увеличении числа элементов до предельно допустимого $n_{\text{пр}}$. Тогда

$$P_{\text{пр}} = B \frac{1}{v^2}, \tag{2.11}$$

где $B = bA$.

Уменьшение рабочей частоты, например, на один порядок позволяет увеличивать число элементов в машине на три порядка и получить тем самым увеличение производительности примерно в 100 раз. Иными словами, производительность ЭВМ при таком подходе уже не лимитируется рабочей частотой и может быть существенно увеличена. Предельная производительность

будет определяться числом элементов, которые можно изготовить за практически приемлемое время, надежностью и т. п.

Попытаемся установить соотношение между числом элементов в различных устройствах ЭВМ в зависимости от производительности. Пусть общее число элементов ЭВМ будет

$$n = n_{\text{п}} + n_{\text{л}}, \tag{2.12}$$

где $n_{\text{п}}$ — число элементов памяти;

$n_{\text{л}}$ — число элементов, затрачиваемое на выполнение арифметических, логических и других операций, а также операций управления.

В свою очередь,

$$n_{\text{п}} = n_{\text{з}} + n_{\text{о}}, \tag{2.13}$$

где $n_{\text{з}}$ — число запоминающих элементов;

$n_{\text{о}}$ — число элементов обрамления памяти, осуществляющего процессы выборки, записи и считывания информации.

Как уже говорилось ранее (см. 1.2), для большинства задач число запоминающих элементов должно расти пропорционально производительности ЭВМ (и, следовательно, не зависит от способа, каким эта производительность достигнута), т. е.

$$n_{\text{з}}(P) = c_{\text{з}}P, \tag{2.14}$$

где $c_{\text{з}}$ — коэффициент пропорциональности.

На основании (2.5)

$$n_{\text{з}}(P) = c_{\text{з}}\frac{L_{\text{э}}}{m}\text{v}. \tag{2.15}$$

Число элементов обрамления памяти с увеличением объема памяти также должно расти. Этот рост может носить логарифмический характер, если одновременно растет и нагрузочная способность элементов (число запоминающих элементов, на которое работает элемент обрамления). Если нагрузочная способность элементов обрамления не изменяется, то их число должно расти пропорционально увеличению объема памяти, т. е.

$$n_{\text{о}}(P) = c_{\text{о}}P = c_{\text{о}}\frac{L_{\text{э}}}{m}\text{v}, \tag{2.16}$$

где $c_{\text{о}}$ — коэффициент пропорциональности.

При росте производительности, достигнутой путем увеличения физического быстродействия элементов, время обращения к памяти должно пропорционально уменьшаться. При совмещении операций и микроопераций время обращения к памяти может оставаться тем же, но должно расти число независимых обращений к памяти. В таком случае должно пропорционально увеличиваться число элементов обрамления памяти. Если память при этом делается в виде независимых блоков, то увеличение числа элементов будет выражаться тем же соотношением (2.16).

При росте производительности, достигнутой в результате увеличения частоты работы, $n_{\text{л}}$ остается почти тем же. При совмещении операций и микроопераций оно растет. Если предположить, что $n_{\text{л}}$ растет пропорционально увеличению производительности, то при этом делается ошибка в сторону завышения числа элементов. Действительно, увеличение $n_{\text{л}}$ в целое число раз эквивалентно тому, что вместо одного устройства стало несколько таких же устройств. Обычно такого же увеличения производительности можно добиться путем увеличения оборудования только в некоторых «узких» местах. Для наших целей достаточно будет и явно завышенной оценки, согласно которой рост числа элементов $n_{\text{л}}$ пропорционален увеличению числа совмещаемых операций и микроопераций, т. е.

$$n_{\text{л}} = c_{\text{л}}L_{\text{э}}, \tag{2.17}$$

где $c_{\text{л}}$ — коэффициент пропорциональности.

Таким образом, общее число элементов без учета затрат на ввод и вывод на основании (2.15) — (2.17) будет равно:

$$n(P) = c_{\text{з}}\frac{L_{\text{э}}}{m}\text{v} + c_{\text{о}}\frac{L_{\text{э}}}{m}\text{v} + c_{\text{л}}L_{\text{э}} \tag{2.18}$$

или

$$n(P) = c_{\text{з}}P + c_{\text{о}}P + c_{\text{л}}L_{\text{э}}. \tag{2.18а}$$

Первые два члена, относящиеся к памяти, растут пропорционально производительности ЭВМ независимо от того, в результате чего произошел этот рост (частоты или распараллеливания). Третий член, характеризующий затраты на арифметические и логические операции, растет только в том случае, если производительность увеличивается путем распараллеливания.

В современных ЭВМ коэффициент $c_{\text{з}}$ на два-три порядка больше $c_{\text{о}}$ и $c_{\text{л}}$. Коэффициенты $c_{\text{о}}$ и $c_{\text{л}}$ по величине одного порядка.

У большинства ЭВМ запоминающие элементы делаются на много проще и дешевле, чем элементы обрамления и логические.

Затраты на последние примерно эквивалентны друг другу. С учетом затрат на запоминающие элементы обычно затраты на арифметический блок и блок управления не превышают общих затрат на память.

Таким образом, при повышении производительности путем распараллеливания общие затраты возрастут не более чем вдвое по сравнению со случаем, когда повышение производительности достигается увеличением частоты работы элементов. Если учесть, что стоимость быстродействующих элементов намного выше стоимости медленнодействующих, первый путь (частотный) не оказывается экономичнее второго (параллельного) и в этом случае. Общее число элементов во втором случае примерно то же, что и в первом, так как прирост числа логических элементов составляет ничтожную долю от общего числа элементов.

Сейчас появилась тенденция строить запоминающие устройства, обрамление и логические блоки из одних и тех же элементов. Так, машина «Сетунь» построена полностью на магнитных элементах [25], японская машина «Мусасино» полностью параметронная [26]. Ведутся разработки машин на криотронах [27].

В ходе применения принципов микроминиатюризации данная тенденция станет, по-видимому, основной, причем для упрощения технологии главное внимание будет уделяться универсальным приборам, с помощью которых могут быть построены как запоминающие, так и логические элементы, например туннельные диоды, индуктивные и емкостные параметроны на пленках, элементы, основанные на явлении сверхпроводимости (криотрон, персистор, криосар и др.). В этом случае основные затраты независимо от пути повышения производительности будут приходиться на элементы памяти, число которых на несколько порядков превышает число логических элементов, так как при микроминиатюризации затраты на изготовление элемента памяти и логического элемента становятся соизмеримыми.

Итак, *увеличение в известных пределах числа логических элементов для повышения производительности ЭВМ мало сказывается как на общем числе элементов, так и на их суммарной стоимости.*

Рассмотрим теперь случай построения ЭВМ для таких задач, у которых рост числа операций не сопровождается значительным увеличением информации (см. 1.2). Ограничимся двумя вариантами:

1) при росте производительности, полученном в результате увеличения частоты, общее число логических элементов остается того же порядка, что и у существующих ЭВМ;

2) при росте производительности, полученном в результате распараллеливания, число логических элементов увеличивается в $L_э$ раз.

При сравнении этих вариантов следует учесть, что хотя число элементов в первом варианте меньше, требуемый уровень микроминиатюризации заметно выше, чем во втором варианте; что особенность изготовления микроминиатюрных конструкций с большим числом элементов такова, что основные затраты занимают разработка физических основ, технологии, создание автоматизированной системы и т. п. Затраты на создание самой системы сравнительно невелики, поэтому различие в числе элементов даже на несколько порядков не играет существенной роли.

И в этом случае, по-видимому, выгоднее идти по пути повышения производительности от распараллеливания, а не повышения частоты.

Итак, приходим к следующим выводам.

1. Повышение производительности путем совмещения во времени частей вычислительного процесса по сравнению с повышением производительности от поднятия частоты работы элементов снижает требования к частоте работы элементов, а, следовательно, и к уровню микроминиатюризации на несколько порядков при несущественном увеличении общего числа элементов, что делает реальным создание ЭВМ высокой производительности в ближайшие годы.

2. Переход к параллельному выполнению большого числа операций налагает некоторые ограничения на алгоритмы решаемых задач и требует разработки новых классов алгоритмов (так называемых параллельных алгоритмов).

2.4. Особенности ЭВМ, реализующих массовое распараллеливание

Существующие ЭВМ в той или иной форме позволяют совмещать во времени выполнение операций или микроопераций (см. 2.2). Уровень этого совмещения сравнительно невысок и дает повышение производительности на 1—2 и реже 3 порядка, что значительно ниже уровня совмещения, который необходим для достижения производительности 10^9 *опер/сек*. В дальнейшем условимся называть совмещение, необходимое для достижения производительности 10^9 *опер/сек* и выше, *массовым совмещением* (или *массовым распараллеливанием*). Как указывалось ранее, при массовом совмещении требуется одновременное выполнение 10^4 — 10^5 операций или микроопераций.

Отсюда следует, что алгоритмы задач, которые будут эффективно решаться таким способом, должны допустить разделение процесса решения на большое число независимо выполняемых операций. Для тех задач, которые не обладают подобным свойством, этот путь повышения производительности будет малоэффективным. Подчеркнем, что здесь речь идет о задачах, требующих выполнения не менее 10^{10} операций, так как с меньшим объемом вычислений могут успешно справиться и существующие ЭВМ. Этот путь может также оказаться бесполезным при управлении быстротекущими процессами, если требуемое время решения задачи менее некоторой величины $t_{\min}$, определяемой тактовой частотой работы элемента. Для указанных типов задач потребуется создать ЭВМ, работающие на высоких частотах. Следует отметить, что перечисленные выше типы задач встречаются редко, поэтому на них останавливаться не будем.

Анализ задач и методов их решения на современных ЭВМ (см. гл. 8) показывает, что многие методы решения задач уже в настоящем виде пригодны для массового распараллеливания. Это прежде всего методы решения задач линейной алгебры, линейного программирования и других задач, сводящихся к действиям над матрицами. При умножении матриц n-го порядка можно одновременно выполнять n^3 операций умножения и $n^3/\log_2 n$ операций сложения.

При решении систем дифференциальных уравнений в частных производных с помощью сеток итерационными методами можно одновременно выполнять число операций, которое не меньше числа узлов в сетке.

При решении задач методом Монте-Карло можно все испытания считать одновременно. Внутри каждого испытания могут также допускаться совмещения.

Аналогично обстоит дело с вычислением интегралов, задачами распознавания образов, машинным переводом и т. д. Отсюда видно, что разработка массово распараллеливаемых алгоритмов в заметном числе случаев не потребует больших усилий. По-видимому, если вопросу разработки параллельных алгоритмов будет уделено достаточное внимание, то класс массово распараллеливаемых алгоритмов может охватить большинство практических задач.

Заметим, что многие задачи, как это можно видеть из указанных примеров, допускают уровень распараллеливания более высокий, чем это требуется для достижения производительности 10^9 *опер/сек*, и не налагают существенных ограничений на число параллельно работающих устройств M, которое определяется в основном общей величиной требуемой производительности P и частотой работы элементов ν. Действительно, пусть каждое из M устройств выполняет за один рабочий такт m элементарных операций, тогда общее число выполняемых за один такт операций сложности m будет равно

$$L = \frac{L_э}{m}. \tag{2.19}$$

Очевидно,

$$M = \beta L, \tag{2.20}$$

где β — коэффициент использования устройств.

На основании (2.5), (2.19) и (2.20)

$$M = \beta \frac{P}{\nu}. \tag{2.21}$$

Для $P \approx 10^9$ *опер/сек*, $\nu = 10^6$ и $\beta = 1$ $M = 1000$.

Если считать, что частота работы элементов и совмещение микроопераций обеспечивают производительность 10^6 *опер/сек*, то уровень эффективного распараллеливания, необходимый для достижения требуемой производительности, будет 10^3. Иными словами, вычислительный процесс должен в среднем состоять из 1000 параллельных ветвей, выполняемых независимо при скорости вычисления в каждой из ветвей 10^6 *опер/сек*.

Если при решении определенной задачи объем информации, которую надо хранить в процессе ее решения, равен v, то на каждую ветвь вычислений в среднем придется объем информации

$$v_в = \frac{v}{M} \tag{2.22}$$

или, согласно (1.1),

$$v_в = \frac{L}{\alpha M}. \tag{2.23}$$

Для многих задач $\alpha = 10$—100, т. е. $v_в$ будет 10^7—10^8 *дв. ед.* при $M = 1000$.

Таким образом, *устройства ЭВМ, предназначенные для выполнения параллельных ветвей в процессе вычисления задачи, должны для большинства задач обладать значительным объемом памяти*, определяемым (2.23).

Как уже указывалось, чтобы быстродействие ЭВМ (*опер/сек*) совпало по величине с производительностью, необходимо, чтобы для каждой из выполняемых задач имелись соответствующие

список команд и разрядность кодов. Это условие трудно выполнить даже для имеющихся задач вследствие их большого разнообразия, тем более трудно учесть требования тех задач, которые появятся в будущем.

Отсюда вытекает требование к ЭВМ, чтобы они были либо проблемно ориентированными, т. е. рассчитаны на решение определенных классов задач, либо обладали избыточным быстродействием для компенсации излишних затрат операций, либо, наконец, обладали возможностью изменять свою структуру, список команд и разрядность в зависимости от задачи.

Первый путь связан со значительными затратами на построение большого числа проблемно ориентированных ЭВМ и требует создания для вновь возникающих классов задач новых ЭВМ, которые должны быть построены в короткий срок.

Второй путь связан с усложнением программирования и избыточной затратой логических элементов.

Третий путь предъявляет некоторые дополнительные требования к логической структуре ЭВМ.

Существующая в вычислительной технике терминология и классификация приспособлены в основном для ЭВМ, в которых в каждый момент времени выполняется одна какая-либо операция. С появлением новых типов ЭВМ, в которых одновременно выполняются несколько операций, возникло и новое понятие — вычислительная система.

Под *вычислительной системой* понимается преобразователь дискретной информации, в различных устройствах которого одновременно может выполняться несколько частей вычислительного процесса.

Вычислительные системы подразделяются:

а) *по назначению* на проблемно ориентированные, предназначенные для решения определенного круга задач, и на универсальные, предназначенные для решения широкого круга задач;

б) *по конструкции* на однородные, образованные многократным повторением одного и того же элемента и его связей (модуля, блока или машины), на неоднородные, образованные из различных и различно соединенных элементов, и на смешанные, построенные из одного или нескольких конструктивных элементов (блоков, модулей и т. п.);

в) *по структуре* на вычислительные системы с жесткой структурой, у которых система связей между модулями или машинами, объем памяти модуля или машины, разрядность кодов и система команд не изменятся в процессе решения задачи, и на вычислительные системы с переменной структурой, у которых система связей между модулями или машинами, объем памяти модуля или машины, разрядность кодов и система команд могут автоматически изменяться во время решения задачи или перед ее решением;

г) *по уровню распараллеливания* на малые вычислительные системы с уровнем совмещения операций менее 100, на средние — с уровнем совмещения операций 10^2—10^3 и на большие вычислительные системы с уровнем совмещения свыше 10^3 операций.

Глава 3

ПРОБЛЕМНО ОРИЕНТИРОВАННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Рассмотрим наиболее характерные проекты проблемно ориентированных ВС, предназначенных для реализации массового распараллеливания частей вычислительного процесса.

3.1. Система Унгера

В 1959 г. С. Унгер предложил проект вычислительной системы для решения пространственных проблем типа распознавания зрительных образов [1]. Эти проблемы отличаются тем, что одни и те же операции выполняются над большими массивами чисел. При этом числа задаются и получаются в виде матриц, и при вычислении данного числа используются числа, находящиеся в соседних ячейках матрицы. Например, при распознавании зрительных образов информация обычно задается в виде двумерной сетки, в каждом узле которой содержится одноразрядное двоичное число. Процесс решения задачи разбивается на ряд последовательных шагов. Первый шаг сводится к выполнению одних и тех же операций над первоначальными числами, на втором шаге выполняются опять-таки одинаковые операции над полученной промежуточной сеткой чисел и т. д. до получения результата. При вычислениях каждого числа используются результаты, полученные в соседних узлах сетки на предыдущем шаге.

Такого рода проблемы позволяют использовать простую логическую схему, имеющую одно центральное устройство управления и большое число одинаковых логических модулей, расположенных в узлах однородной сетки (рис. 4). Центральное устройство управления представляет собой, по сути, ЭВМ с памятью большого объема. Логический модуль состоит из одноразрядного двоичного накопительного сумматора с соответствующими логическими схемами для выполнения определенного набора операций и нескольких одноразрядных ячеек памяти. Каждый

модуль соединен каналами связи со своими четырьмя соседями, центральным устройством и внешним источником информации. Все логические модули выполняют одновременно одну и ту же операцию.

Набор операций данной ВС можно разделить на три группы:

1) пять логических операций — замена числа в *сумматоре* его отрицанием; его дизъюнкцией или конъюнкцией с числом либо из определенной ячейки памяти данного модуля, либо из

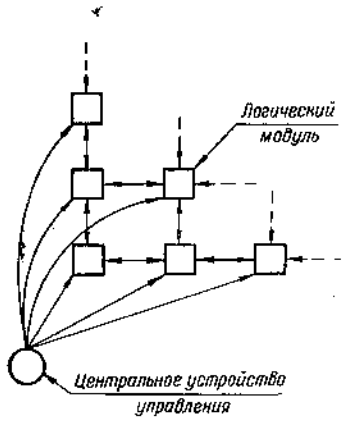


Рис. 4. Блок-схема системы Унгера.

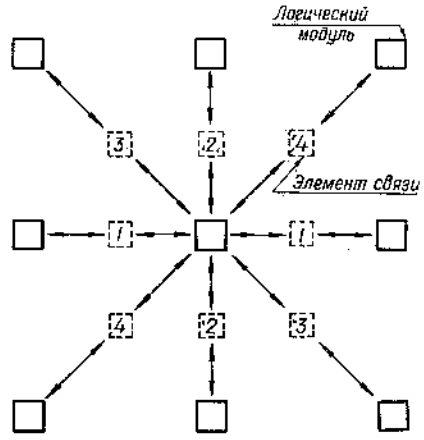


Рис. 5. Схема связи между логическими модулями системы Унгера.

сумматора одного из четырех соседних модулей; замена числа в *ячейке памяти* модуля его дизъюнкцией или конъюнкцией с числом из сумматора данного модуля;

2) две операции, специфичные для проблем распознавания образов,— операции связи и распространения, с помощью которых можно выявлять линии (образованные единицами в сумматорах модулей), идущие в заданном направлении из данных точек (модулей). Эти операции осуществляются с помощью специальных элементов связи (рис. 5), которые соединяют каждый модуль с его восемью соседями (включая модули, расположенные по диагонали). Элементы связи могут находиться в двух состояниях: возбужденном, когда они соединяют прилежащие модули, и невозбужденном. Операция связи приводит в возбужденное состояние все элементы связи, в обоих сумматорах прилежащих модулей которых находятся единицы. При выполнении операции распространения каждый из модулей, содержащий единицу

в сумматоре, заносит ее в сумматоры всех модулей, которые связаны с ним линией из возбужденных элементов связи. Направление этой линии (горизонтальное, вертикальное или вдоль одной из диагоналей) указывается в команде распространения. Типовой порядок использования этих операций следующий: а) вводится образ путем записи единиц и нулей в сумматоры модулей; б) выполняется операция связи; в) вместо первого образа вводится новый образ (совокупность точек, связность которых хотят установить); г) выполняется операция распространения;

3) операция условного перехода, осуществляющая переход к другой подпрограмме, если в сумматорах ни одного из модулей не содержится единица; операция безусловного перехода; операции сдвига:

а) влево, вправо, вверх, вниз, с помощью которых можно передавать информацию от модуля к модулю в нужном направлении. При этом информация, содержащаяся в сумматоре данного модуля, записывается в сумматор соответствующего соседнего модуля. Последовательно применяя эти операции, можно информацию, содержащуюся в данном модуле, передать в любой модуль системы;

б) кругового сдвига, при выполнении которой информация, содержащаяся в сумматоре каждого из модулей, передается к его соседу справа, кроме крайних правых модулей, содержимое которых передается к крайним левым модулям вышележащего ряда. При этом все модули образуют цепочку для передачи информации от крайнего левого модуля нижнего ряда, в который информация передается извне, до крайнего правого модуля верхнего ряда, содержимое которого выводится. Эта операция позволяет одновременно вводить новую информацию и выводить старую.

Логическая схема модуля с учетом элементов связи содержит около 170 вентилей и 11 элементов памяти (рис. 6)[1]. Когда триггер находится в состоянии 1, $T = 1$, в состоянии $0 - \bar{T} = 1$. В модуль входит пять командных шин по числу логических операций первой группы. Если от центрального устройства подается сигнал в одну из этих шин, то выполняется соответствующая операция первой группы. Операции третьей группы образуются с помощью комбинации логических операций первой группы. В этом случае сигналы от центрального устройства подаются по нескольким шинам. Для указания адресов сумматоров соседних модулей служат четыре шины и еще по одной шине приходится на каждую ячейку памяти данного модуля. Операции

[1] Для последующих рисунков приняты условные обозначения рис. 6.

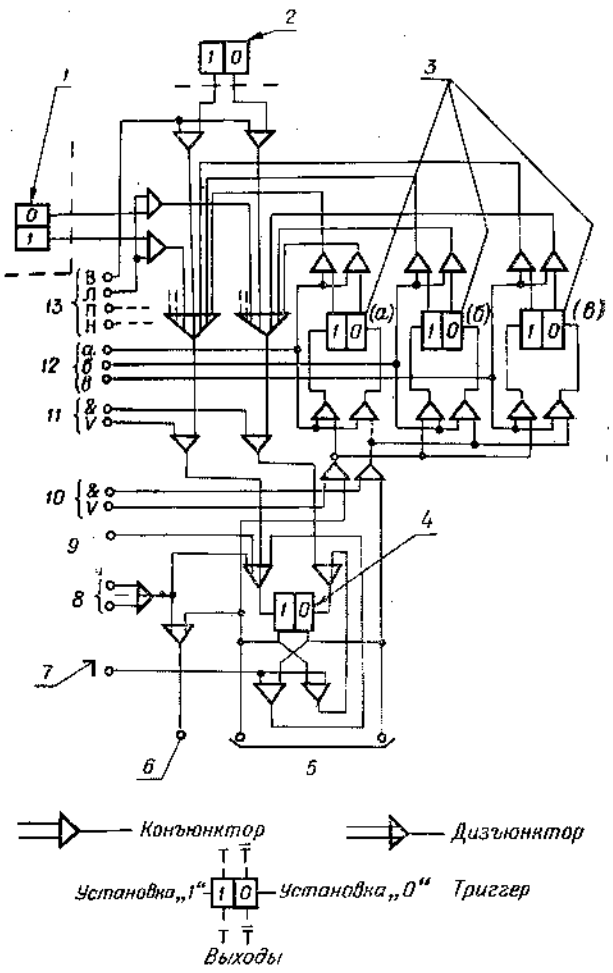


Рис. 6. Принципиальная схема логического модуля системы Унгера.

1, 2 — сумматоры соседних модулей; *3* — регистры памяти; *4* — сумматор; *5* — сигналы от схем связи соседних модулей; *6* — сигнал к схемам связи соседних модулей; *7* — операция отрицания; *8* — сигналы от схем связи; *9* — вход связи; операции с ячейками памяти: *10* — запомнить; *11* — записать или сдвинуть; *12* — адреса ячеек памяти; *13* — адреса соседних модулей (верхний, левый, правый, нижний).

второй группы выполняются, когда из центрального устройства подается сигнал в особую шину связи.

То что все модули одинаковы и имеют одинаковые связи, позволяет увеличивать их число без изменения логической структуры ВС. С помощью такой ВС могут решаться задачи по распознаванию образов, требующие для своего решения как сотен, так и тысяч и десятков тысяч модулей. Применение большого числа модулей в принципе позволяет ускорять процесс распознавания образов до нужных пределов путем массового распараллеливания процесса вычислений.

Экономические затраты на создание такой системы при применении микроминиатюризации могут быть невелики.

Область применения этой ВС ограничивается небольшим кругом проблем из-за следующих ее особенностей: 1) во всех модулях выполняется одна и та же операция; 2) все модули являются активными, т. е. нельзя какие-либо модули исключить из выполнения операции; 3) объем памяти каждого модуля мал (несколько двоичных единиц); 4) мала разрядность чисел, над которыми выполняются операции; 5) набор операций невелик.

3.2. Система СОЛОМОН

Дальнейшим развитием идеи Унгера является проект вычислительной системы СОЛОМОН[1], разрабатываемой фирмой «Вестингауз Электрик Корпорэйшн» (США) [2—4]. СОЛОМОН предназначается для решения систем линейных уравнений, обращения матриц, вычисления корреляционных функций, численного решения уравнений в частных производных и других задач, для которых допускается одновременное выполнение одной и той же операции над множеством чисел. Эти задачи в отличие от задач распознавания образов требуют запоминания значительно большего объема информации, оперирования с многоразрядными кодами и выполнения более сложных операций.

Как и в предыдущем проекте, вычислительная система СОЛОМОН состоит из центрального устройства управления и сетки однородных и одинаково соединенных модулей (обрабатывающих устройств). В сетке 1024 модуля, расположенных в виде квадрата 32×32. Каждый модуль имеет арифметическое устройство, выполняющее последовательно, разряд за разрядом арифметические и логические операции, и два блока памяти емкостью по 2048 двоичных единиц каждый (рис. 7). Каждый модуль соединен каналами связи со своими четырьмя соседями. Кроме того, имеется общий канал связи, с помощью которого все модули соединены с центральным устройством и могут считывать с его выходного регистра константы, требующиеся в процессе вычисления.

[1] Simultaneous Operation Linked Ordinal MOdular Network.

Свободные каналы связи крайних модулей используются для соединения с выводными устройствами. Большое число параллельных каналов связи обеспечивает высокую скорость обмена информацией с внешними устройствами. В СОЛОМОНЕ предусматриваются две системы внешних устройств. Первичная — на магнитных дисках и лентах, работающая со скоростями, близкими к скорости работы модулей, и вторичная, состоящая из быстропечатающих и других стандартных устройств для ввода и вывода информации. Первичная система обменивается информацией непосредственно с модулями и может служить также в качестве вспомогательной памяти для задач с большим объемом данных. Вторичная система служит для связи потребителя с вычислительной системой, которая осуществляется через первичную систему ввода — вывода.

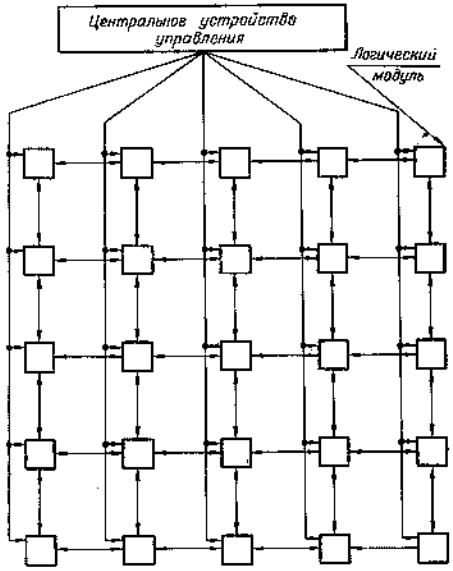


Рис. 7. Блок-схема системы СОЛОМОН.

Все модули в один и тот же момент могут выполнять только одну операцию над числами, хранящимися в их ячейках памяти с одними и теми же адресами. Однако в отличие от проекта Унгера в СОЛОМОНЕ не все модули должны обязательно выполнять команды, поступающие из центрального устройства. Часть из них может игнорировать полученную команду и оставаться в пассивном состоянии. Это достигается путем применения многомодальной логики. В каждом модуле содержится регистр модальности, имеющий четыре состояния (рис. 8). Центральное устройство вместе с командой задает и модальность. Команда выполняется только теми модулями, модальность которых сов-

падает с модальностью, задаваемой центральным устройством. Модальность модулей задается программно с помощью специальных команд, меняющих состояние регистров модальности каждого из модулей. Использование многомодальных операций позволяет также индивидуально управлять передачей информации,

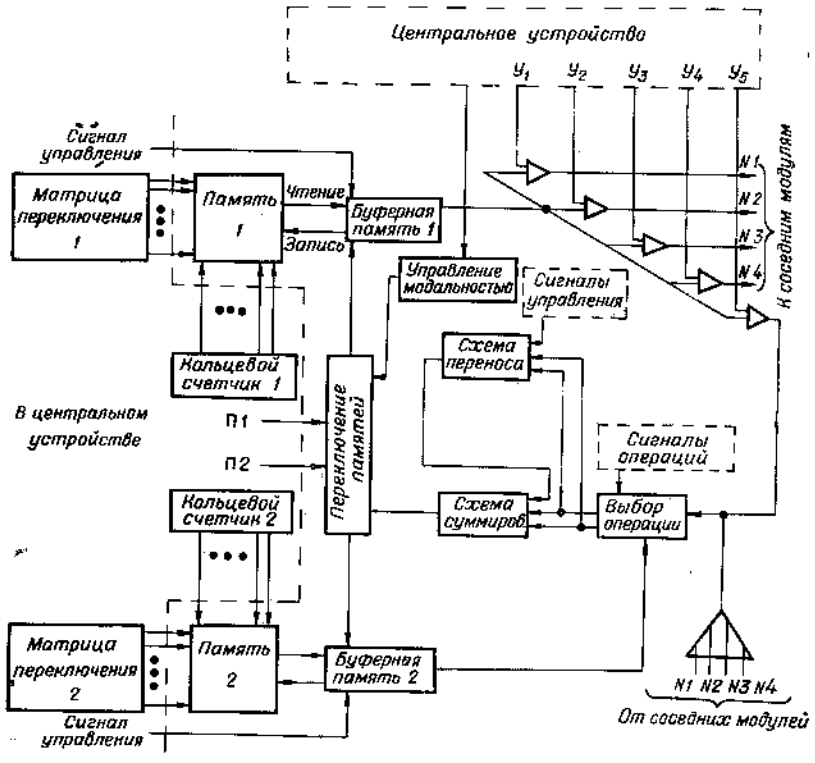


Рис. 8. Схема логического модуля системы СОЛОМОН.

так как модули, у которых состояние модальности отлично от задаваемого центральным устройством, не воспринимают информацию.

Кроме обычных логических и арифметических операций в СОЛОМОНЕ предусматриваются специальные операции для матриц, с помощью которых выбираются заданные строки и колонки. Невыбранные ряды элементов при этом передают требуемые коды, не меняя содержимого памяти. Для выполнения этих

операций в центральном устройстве предусмотрены специальные регистры, управляющие выбором соответствующих строк и колонок.

По сравнению с обычными ЭВМ система СОЛОМОН имеет более высокую производительность благодаря следующим особенностям:

1) одновременному выполнению большого числа операций (до 1024);

2) ничтожно малым затратам на выборку из памяти команды, так как одна команда выбирается не для каждой операции, а сразу для 1024;

3) матричному расположению модулей, что позволяет исключить индексные операции;

4) возможности использования непосредственных связей между модулями;

5) использованию переменной длины кода.

В то же время из-за последовательного принципа работы отдельная операция выполняется модулем системы СОЛОМОН дольше, чем ЭВМ параллельного действия с такой же тактовой частотой. Эффективность заметно снижается для тех задач, у которых в каждый момент времени выполняется менее 1024 операций. С учетом этих факторов система СОЛОМОН, по оценкам авторов, будет иметь производительность в 60—200 раз выше, чем лучшие современные ЭВМ. Эта производительность будет, конечно, достигаться только для тех задач, для которых несущественны ограничения, определяемые особенностями СОЛОМОНА, а именно:

1) во всех 1024 модулях одновременно может выполняться только одна операция, т. е. вычислительный процесс должен делиться на большое число *одинаковых* ветвей;

2) объем памяти каждого модуля составляет 128 тридцатидвухразрядных чисел, т. е. коэффициент активности информации α должен быть достаточно высоким, чтобы время ввода и вывода не составило основную долю общего времени решения задач.

3.3. Итеративные системы Холланда

Заметным вкладом в разработку принципов построения вычислительных систем являются работы Холланда [5, 6] по итеративным вычислительным системам. В отличие от двух вышеописанных ВС, которые предназначались для решения задач, допускающих представление вычислительного процесса в виде большого числа *одинаковых* параллельных ветвей, итеративные ВС Холл-

ланда предназначаются для реализации вычислительных процессов как с одинаковыми, так и с *различными* параллельными ветвями вычислений.

Таким образом, если в ВС Унгера и СОЛОМОНЕ в каждый момент времени должна выполняться одна программа (задаваемая центральным устройством управления и реализуемая модулями), то в системе Холланда может одновременно выполняться *много различных* программ.

Вычислительная система Холланда представляет собой однородную сетку из одинаковых модулей. В случае двумерной сетки каждый модуль соединен со своими четырьмя соседями. Функции модулей заключаются не только в выполнении операций, но и в *управлении* ходом вычислений. Каждый модуль имеет запоминающий регистр объемом $n + 14$ двоичных единиц (табл. 1)

Таблица 1

Распределение разрядов запоминающего регистра системы Холланда

$n+14$	$n+13 \ldots 14$	13...10	9 8 7	6 5	4 3	2	1
y_n	$y_{n-1} \ldots y_0$	$d_3 \ldots d_0$	$i_2 i_1 i_0$	$s_1 s_2$	$q_1 q_2$	p	a

с соответствующими логическими схемами и несколько вспомогательных регистров. Модули работают синхронно во времени, которое рассматривается как дискретная величина с шагом дискретности (тактом) $t = 0, 1, 2 \ldots$ В каждом такте модуль может быть в одном из двух состояний: активном или пассивном.

В активном состоянии модуль интерпретирует число в разрядах i_0, i_1, i_2 запоминающего регистра как команду и переходит к ее выполнению. В каждый данный момент может быть произвольное число активных модулей. Обычно, если модуль $m(i, j)$ с координатами i, j активен в такте t, то в такт $t + 1$ он переходит в неактивное состояние, а один из четырех соседних модулей: $m(i + 1, j)$, $m(i - 1, j)$, $m(i, j + 1)$ или $m(i, j - 1)$ — становится активным. (Исключение из этого правила бывает только при выполнении команд условного перехода.) Последующий активный модуль определяется разрядами s_1, s_2 запоминающего регистра модуля $m(i, j)$. Таким образом, можно образовывать цепочки активных модулей и тем самым задавать подпрограмму работы. Одновременно может быть произвольное число активных модулей, что позволяет параллельно выполнять

несколько подпрограмм. Кроме команд, управляющих ходом вычислений, имеются команды, изменяющие содержимое запоминающего регистра, поэтому число подпрограмм может меняться во времени. Эти изменения управляются одной или несколькими специальными подпрограммами.

Действия модуля во время каждого такта делятся на три последовательные фазы.

1. Во время начальной фазы в запоминающий и вспомогательные регистры могут вводиться извне числа. У большинства модулей ввод чисел будет происходить только на первых тактах работы вычислительной системы (ввод программы) или на некоторых определенных тактах (ввод данных).

2. Во время второй фазы активный модуль определяет местонахождение операнд (чисел, над которыми будет выполняться данная команда). Активный модуль делает это посредством прокладывания пути к модулю, содержащему операнды. Прокладывание пути обеспечивается двумя свойствами модулей:

а) каждый модуль установкой единицы в разряде p регистра может быть приведен в особое состояние, при котором он становится источником возникновения путей. В таком состоянии модуль называется P-модулем;

б) каждый модуль имеет своего предшественника, указанного в разрядах q_1, q_2 регистра. Последовательность предшественников связывает активный модуль с ближайшим P-модулем. В течение начальной части второй фазы разряды регистра активного модуля $y_0, \ldots, y_n$ и $d_0, \ldots, d_3$, определяющие путь, передаются по линии своих предшественников к ближайшему P-модулю. Содержимое этих разрядов потом используется P-модулем для образования пути к числам, над которыми должна быть выполнена команда, записанная в разрядах i_0, i_1, i_2 регистра активного модуля. Каждый путь благодаря расположению модулей в виде прямоугольной сетки состоит из отрезков прямых, параллельных осям сетки. Эти отрезки, называемые сегментами, состоят из некоторого начального модуля с координатами (i, j), промежуточных модулей с координатами $(i + b_1, j + b_2)$, $(i + 2b_1, j + 2b_2)$, $\ldots$, $(i + (y - 1)b_1, j + (y - 1)b_2)$ и конечного модуля с координатами $(i + yb_1, j + yb_2)$. Здесь $b_1 = \pm 1$ или 0, а $b_2 = \pm (1 - |b_1|)$.

В каждом модуле имеются четыре вспомогательных $*$-регистра, каждому $*$-регистру соответствует одно из направлений, определяемое числами (b_1, b_2). Так как эти регистры независимы, то каждый модуль может принадлежать четырем путям. Однажды включенный $*$-регистр сохраняет определяемый им путь до его выключения.

Каждый сегмент образуется от одного единственного активного модуля. Однако от одного активного модуля благодаря ветвлению может образовываться много сегментов. После того как с помощью чисел $y_n, \ldots, y_0, d_3, \ldots, d_0$ был проложен путь к ближайшему P-модулю вдоль линии предшественников активного модуля, на концах каждой ветви в P-модулях возникают новые сегменты. Ветвление управляется числами $d_3, \ldots, d_0$. Каждому из этих четырех чисел соответствует один из четырех соседей в каждом конце ветви.

Путь разветвляется только в тех случаях, когда разряд регистра $y_n = 0$, тогда разряды $y_{n-1}, \ldots, y_0$ указывают общую длину новых сегментов, а разряды $d_3, \ldots, d_0$ — их направление. При $y_n = 1$ направления, указанные в разрядах $d_3, \ldots, d_0$, стираются, а разряды $y_{n-1}, \ldots, y_0$ не используются.

Чтобы не возникали перекрещивания различных путей, установлены определенные правила приоритета.

3. В течение третьей фазы выполняется команда, содержащаяся в разрядах регистра i_2, i_1, i_0 активного модуля. Кроме активного модуля, в этом участвуют модули, являющиеся окончанием ближайшего пути, запоминающие регистры которых содержат операнды, и модуль, который выполняет функцию арифметического устройства (A-модуль). В качестве такого модуля служит первый A-модуль, встречающийся в ряду предшественников данного активного модуля. Для перевода модуля в A-модуль в разряды p и a его регистра должны быть соответственно засланы 0 и 1.

Список команд итеративной вычислительной системы состоит из следующих операций:

1) ИЛИ/СЛОЖИТЬ. Числа, хранимые в соответствующих модулях, передаются по ветвям в P-модули. В точках ветвления образуется поразрядная дизъюнкция чисел из разных ветвей. Результат передается в ближайший A-модуль, где число суммируется с содержимым запоминающего регистра A-модуля;

2) И/СЛОЖИТЬ. Выполняется аналогично предыдущей операции, но вместо поразрядной дизъюнкции образуется поразрядная конъюнкция;

3) ЗАПОМНИТЬ. Содержимое ближайшего A-модуля передается в запоминающие регистры всех модулей, содержащих операнды;

4) ИЗМЕНИТЬ ПО МИНУСУ. Выполнение этой команды зависит от числа в запоминающем регистре ближайшего A-модуля. Если в этом числе разряд $y_n = 0$, то в конце третьей фазы активный модуль становится неактивным, а следующий за ним модуль становится активным. Если $y_n = 1$, то все модули, содержащие числа, над которыми производятся вычисления, становятся активными;

5) НЕ ВЫПОЛНЯТЬ ОПЕРАЦИЮ. При этом третья фаза проходит без выполнения команды;

6) СТОП. Активный модуль становится неактивным без последующей передачи состояния активности следующему за ним модулю.

Вычислительная система Холланда обладает двумя существенными преимуществами перед ранее описанными системами: а) в ней может выполняться одновременно большое число *различных* подпрограмм; б) вычислительная система строится только из сравнительно *простых однородных* модулей.

Недостатки данной вычислительной системы в том, что она имеет малый объем памяти модуля (это приводит к неэффективному использованию оборудования в большинстве задач) и ограниченный набор команд.

3.4. Система для поиска информации

С. Ли и М. Паулл [7] предложили проект вычислительной системы для поиска информации.

Предыдущие вычислительные системы были ориентированы на решение вычислительных задач. В связи с этим у них основное внимание уделялось структуре арифметических устройств. Для задачи поиска информации первостепенное значение имеет организация и объем памяти.

Авторы исходили из следующих требований к памяти:

1) длина цепочки информации, которая может храниться в памяти, не ограничена. Это означает, что в данном случае нет смысла говорить о ячейке памяти, состоящей из фиксированного числа разрядов;

2) скорость выборки информации не зависит от количества информации, содержащейся в памяти. Увеличение скорости достигается путем параллельной работы элементов, а не путем увеличения их быстродействия;

3) память единообразна и имеет модульную конструкцию. Ее объем может регулироваться добавлением или удалением идентичных модулей;

4) применяется принцип адресации по содержимому, что позволяет избавиться от таких операций, как сканирование, поиск, счет.

Вычислительная система, предлагаемая авторами, состоит из управляющей вычислительной машины, в которой содержится

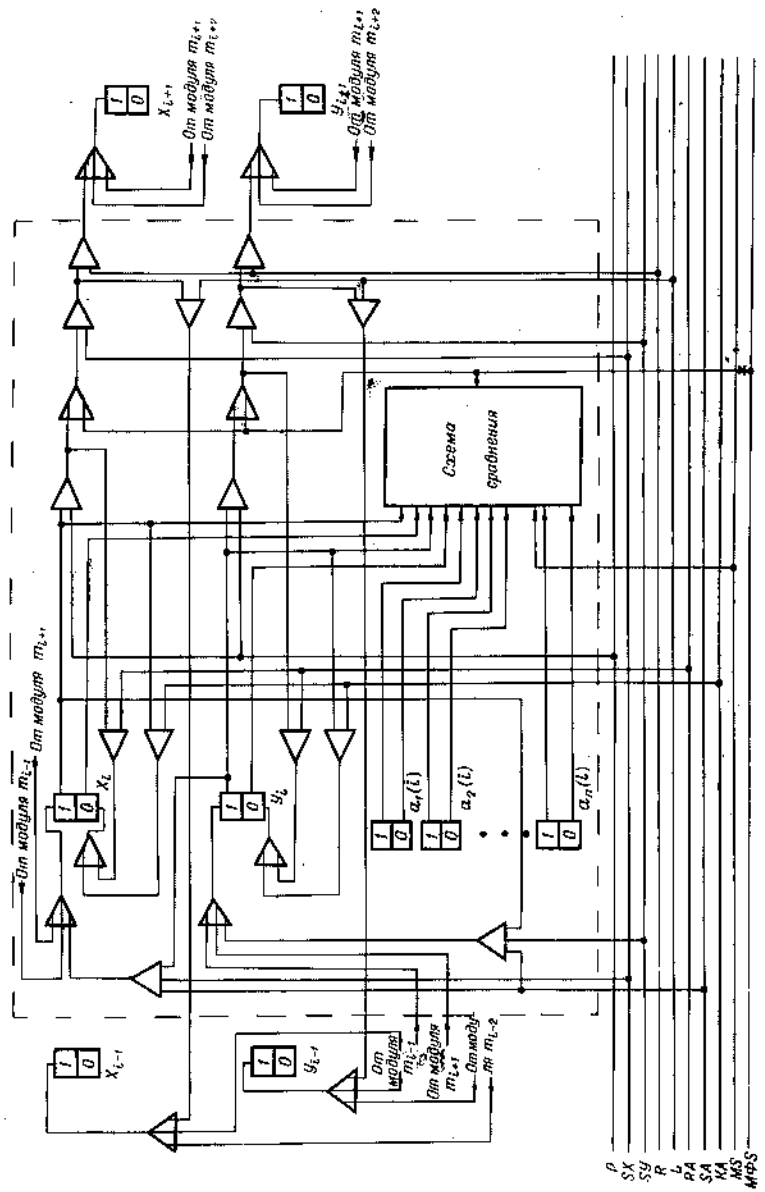


Рис. 9. Схема логического модуля системы для поиска информации.

программа работы, и большого числа небольших вычислительных машин (модулей), вытянутых в линию. Каждый модуль состоит в основном из запоминающих двоичных элементов (типа триггеров), которые делятся на два типа: элементы состояния модуля и элементы символа модуля (запоминаемых чисел). Элементов состояния модуля два: элемент X-активности, элемент Y-активности. Элементов символа модуля может быть несколько в зависимости от количества символов в принятом алфавите. Под содержимым модуля понимается распределение единиц во всех элементах модуля. Это содержимое можно также называть комплексным символом данного модуля. Комплексный символ включает в себя как модули состояния, так и символы модуля.

Типовой модуль m_i имеет два триггера активности X_i и Y_i и n разрядов символа $a_1(i)$, $a_2(i)$, . . ., $a_n(i)$ (рис. 9). Состояние модулей управляется центральной вычислительной машиной с помощью управляющих шин. По шине P (распространения) посылается сигнал передачи активности от данного модуля к соседним. Для выбора типа активности X или Y служат соответственно шины SX и SY. Направлением распространения активности управляют шины R (вправо) и L (влево). Шина SA служит для установления в активное состояние триггеров. По шине RA подается сигнал установки на нуль триггеров активности после действия сигнала распространения. Шина KA служит для уничтожения активности модулей, выбранных с помощью шин SX и SY. При появлении сигнала в шине MS выполняется операция сравнения. Шина $МФS$ выходная. На ней появляется сигнал P_i, когда при выполнении операций сравнения в модуле m_i имеется равенство сравниваемых величин.

У каждого модуля имеется $2n$ входных шин a_1, a_1', a_2, a_2', ..., a_n, a_n', через которые вводится входная информация (входной символ) (рис. 10, *а*). Модуль m_i принимает входной символ, если подан сигнал в шину IS и хотя бы один из триггеров X_i или Y_i активен. X- или Y-активности выбираются программно с помощью шин SX или SY. При запоминании входного символа входные импульсы подаются на триггеры символов $a_1(i)$, $a_2(i)$,... ..., $a_n(i)$. Источником символов может быть как память центральной вычислительной машины, так и память самих модулей. Для запоминания символов, поступающих из памяти центральной машины либо из памяти модулей, в системе имеется буферная память.

Выходная схема модуля аналогична входной (рис. 10, *б*). Модуль m_i выдает содержащийся в ней символ, по $2n$ выходным шинам Φ_1, Φ_1', Φ_2, Φ_2', . . ., Φ_n, Φ_n', если подан сигнал в шину

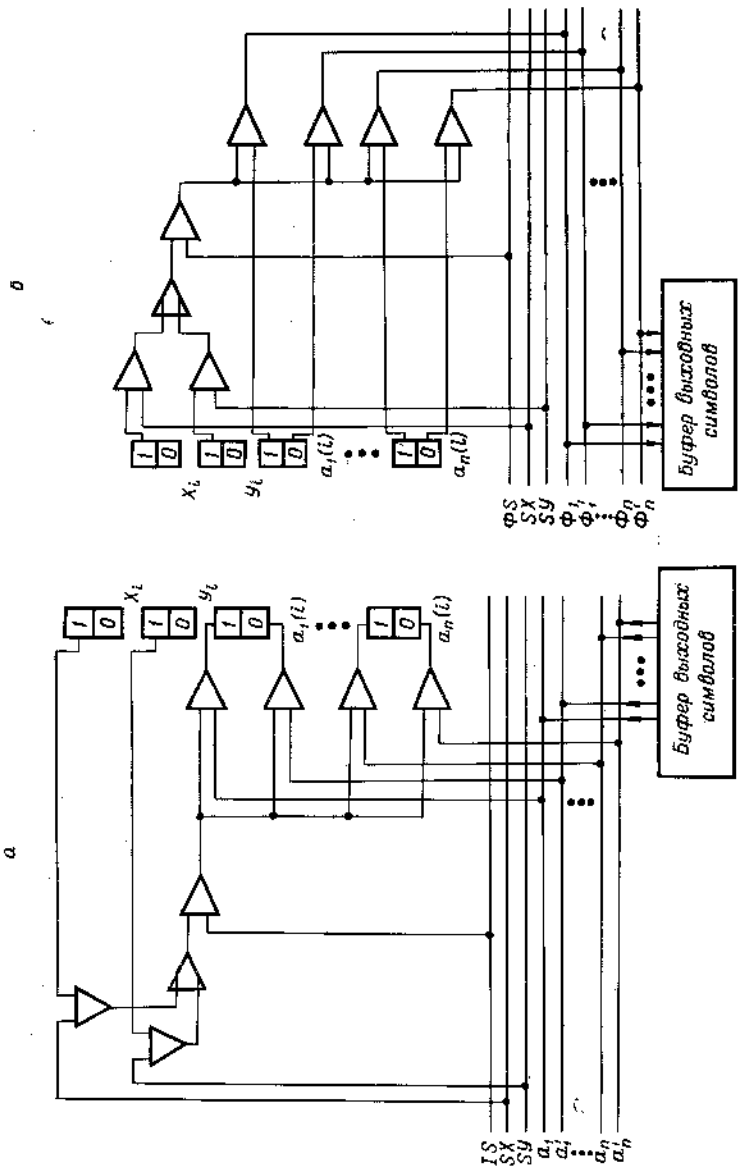


Рис. 10. Схема модуля системы для поиска информации.
а — входная; *б* — выходная.

ΦS и если возбужден триггер X_i и шина SX, либо триггер Y_i и шина SY. Во всех случаях выходной символ запоминается в буферной памяти.

Операция сравнения выполняется с помощью специальной схемы (рис. 11). После поступления сигнала в шину M происходит сравнение содержимого каждого элемента всех модулей с

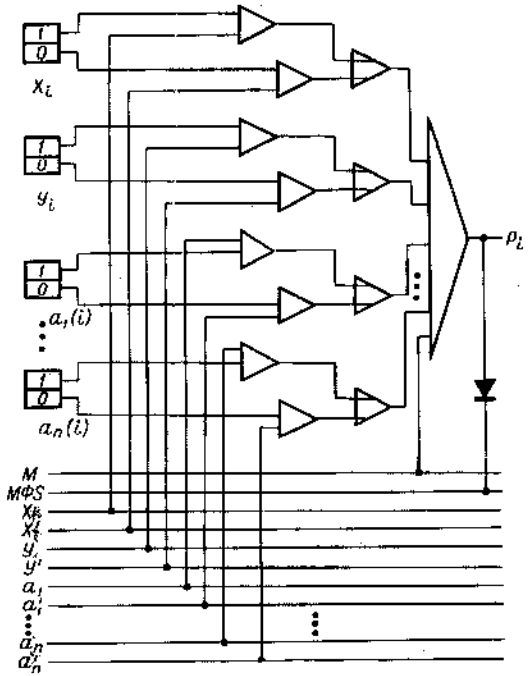


Рис. 11. Схема сравнения модуля системы для поиска информации.

информацией, поступающей по соответствующим шинам: X, X', Y, Y', a_1, a_1', . . ., a_n, a_n'.

Те ячейки, содержимое которых совпадает с поступающей информацией, выдают сигнал в шину $M\Phi S$. Эти сигналы поступают на пороговую схему, которая устанавливает, сколько сигналов (ни одного, один или много) поступило от модулей.

Модули содержат также схемы пошагового распространения (рис. 12). Эти схемы имеют две выходные шины $Q_{i,L}$ и $Q_{i,R}$ для передачи состояния возбуждения соответственно к соседним левому и правому модулям и две входные шины $Q_{i+1,L}$ и $Q_{i-1,R}$,

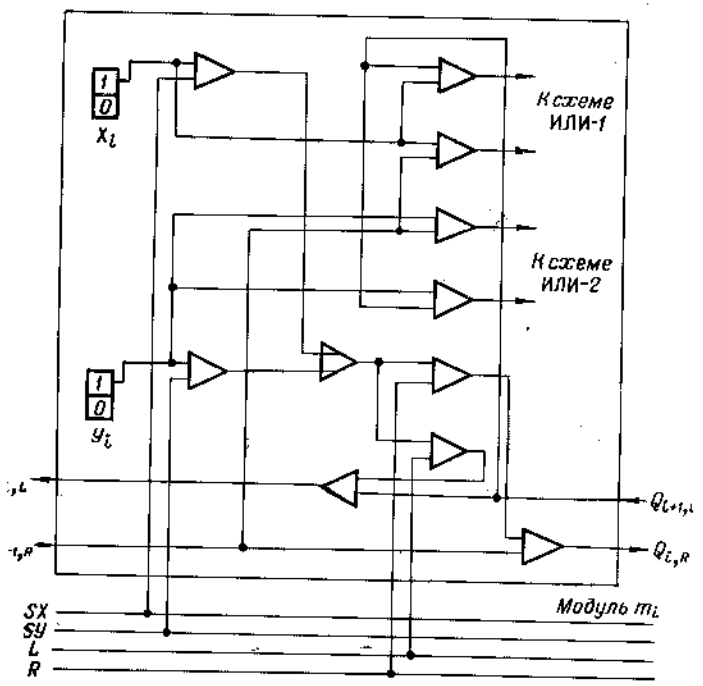


Рис. 12. Схема распространения модуля системы для поиска информации.

воспринимающие сигналы распространения соответственно от левого и правого соседа. Левый крайний модуль (первый) и правый крайний модуль (n-й) имеют шины $Q_{0,R}$ и $Q_{n+1,L}$, по которым поступают сигналы распространения от центральной машины. Если, например, сигнал пошагового распространения появился в n-м модуле и шины SX и L активные, то сигнал будет распространяться шаг за шагом влево, пока не достигнет первого X-активного модуля. Если этим модулем будет m_i, то сигнал в шине SX вызовет появление сигнала на выходе модуля. Так как шина L возбуждена, то действие сигнала распространения прекращается на данном модуле. Далее, при возбужденном триггере X_i и сигнале в шине $Q_{i+1,L}$ становится X-активным модуль m_{i-1}.

В данной вычислительной системе имеется 16 команд для управления работой модульной памяти. Эти команды можно разделить на два класса: непосредственно воздействующие на модули и связанные с работой центральной машины. Каждая команда

может иметь до четырех параметров: α, β, γ и λ. Параметр α указывает направление распространения (влево или вправо); β — тип активности (X или Y либо X и Y); γ — сложность символа. В сложном символе кроме числа еще содержится указание на типы активности, в простом, который обозначается γ^*, содержится только число, λ служит для указания адреса в программе управления.

Приведем несколько примеров команд:

1) KA, β — уничтожить активность во всех модулях с β-активностью. Если $\beta = X$, то во всех модулях будет уничтожена X-активность;

2) S, β, γ^* — запомнить, символ γ^* во всех модулях с β-активностью. Например, команда $SX(Q)$ означает, что символ Q запоминается во всех модулях с X-активностью.

Применение подобной вычислительной системы дает увеличение скорости выполнения информационных задач. Решение других задач, например вычислительных, будет, конечно, малоэффективно.

3.5. Оптическая вычислительная система

Дж. Хаукинс и С. Манси [8] в 1963 г. предложили проект оптической вычислительной системы для распознавания образов. В отличие от ранее описанных вычислительных систем в основе данной ВС лежит более простая логика и более гибкая система связей между модулями. При этом авторы исходили из следующих факторов:

1) число двоичных разрядов в изображении образа во многих практических применениях составляет 10^4—10^5 и может возрастать до 10^7;

2) изготовление сеток из такого числа сложных элементов (модулей) наталкивается на трудности, для преодоления которых потребуется много лет;

3) в большинстве случаев для обработки образов функции модуля могут быть значительно упрощены.

Авторами предложена система, состоящая из двумерных сеток. В каждой сетке S модулей. В простейшем случае имеются две сетки: входная и выходная (рис. 13). Свойства этой системы таковы:

1) сигналы распространяются только в одном направлении, от входной сетки к выходной;

2) каждый модуль выходной сетки принимает сигналы от N входных модулей, где $N \ll S$ (заметим, что связи не ограничиваются только ближайшими соседями);

3) конфигурация связей между входными и выходными модулями одна и та же для всех модулей;

4) на каждом шаге обработки все выходные модули выполняют одну и ту же логическую операцию над входными сигналами;

5) выходные модули передают свои результаты в память, входные получают матрицу двоичных единиц либо из памяти, либо извне. При этих передачах сохраняется первоначальное относительное расположение разрядов образа;

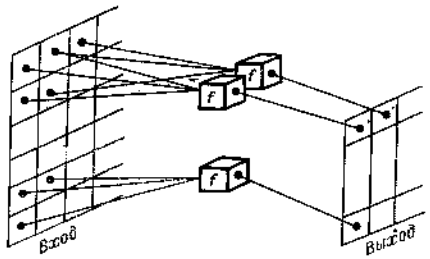


Рис. 13. Простейшая схема планарной оптической ВС.

6) на каждом шаге работы команда кроме адресов входного источника и адресов, по которым посылается результат, состоит еще из двух частей, задаваемых таблично, логической функции (таблица истинности) и относительного положения переменных (рис. 14);

7) на каждом шаге могут выполняться только такие логические функции, которые являются линейной функцией переменных x_i

$$f = \frac{1}{2}\left\{1 - \operatorname{sgn}\left[c_0 + \sum_{i=1}^{N} c_i x_i\right]\right\}, \qquad (3.1)$$

где $c_0, c_1, \ldots, c_N$ — любые действительные числа, знак sgn означает, что функция $\operatorname{sgn} z = -1$ при $z < 0$, 0 при $z = 0$, $+1$ при $z > 0$. Переменные x_i и функция f принимают значения 0 и 1.

Для выполнения, например, функции $\neg \bigvee_{i=1}^{N} x_i$ (отрицание дизъюнкции) нетрудно убедиться, что коэффициентам должны

Таблица 2

Число представлений функций логики

N	2^{2^N}	Линейная логика	И или ИЛИ
2	16	14	14
3	256	104	48
4	65536	1881	154
5	4294967296	94572	476
6	16×10^{18}	15028134	1446

быть приданы следующие значения: $c_0 = -1/2$, $c_1 = c_2 = \ldots = c_N = 1$. Естественно, что не все функции логики могут быть представлены с помощью (3.1). Однако число представимых линейных функций довольно велико (табл. 2). Кроме того, эти функции образуют полный набор функций логики (в частности, свойством полноты, как известно, обладает функция $\neg \bigvee_{i=1}^{N} x_i$) и все функции, не представимые в форме (3.1), могут быть образованы с помощью представимых функций за определенное число шагов посредством подачи выходных сигналов снова на вход или путем применения многосеточных конструкций, когда выходные сигналы одной сетки служат входными для другой.

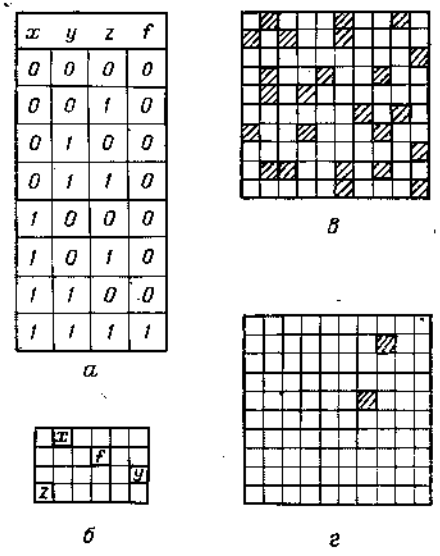


Рис. 14. Пример реализации конъюнкции от трех переменных на оптической ВС.

а — таблица истинности $f=xyz$; *б* — взаимное расположение переменных и функции; *в* — входная матрица; *г* — выходная матрица.

Основная особенность данной вычислительной системы в том, что она позволяет реализовать большое число связей. Если S соответствует числу разрядов, необходимому для достижения высокой разрешающей способности при распознавании образа, то даже при небольших N общее число связей будет 10^8—10^9. Такое число связей трудно выполнить на существующей технике. Однако высокая степень параллелизма и итеративная природа логики позволяют использовать для реализации функций вида (3.1) оптические устройства.

С помощью простой оптической системы можно получить такое двумерное распределение энергии, которое соответствовало бы в каждой точке величине, равной $c_0 + \sum_{i=1}^{N} c_i x_i$ (рис. 15). Пусть входная сетка представляет двоичные величины в форме интенсивности некогерентного света. Поместим между входной и выходной сетками транспорант T, который везде непрозрачен, кро.

ме определенных мест, соответствующих относительному расположению входных переменных выполняемой функции. В этих местах сделаем величину передачи пропорциональной коэффициентам c_i, требуемым для реализации функции. Если при этом выходную сетку освещать равномерно светом с интенсивностью, пропорциональной c_0, то в каждой точке выходной сетки будет происходить суммирование интенсивностей. Заметим, что это справедливо только для некогерентных источников света, когда нет интерференции. Суммарная интенсивность света будет пропорциональна $c_0 + \sum_{i=1}^{N} c_i x_i$, что согласуется с (3.1).

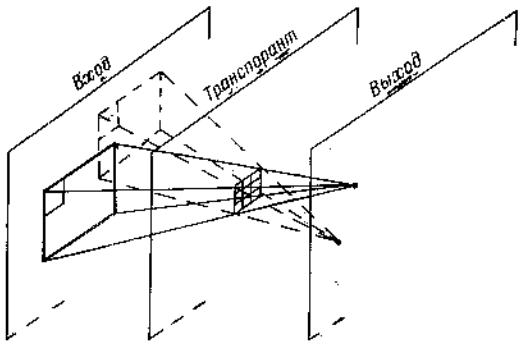


Рис. 15. Принцип работы оптической ВС.

Для завершения логического процесса необходимо выполнить операцию sgn. В обычных системах эта операция выполняется с помощью инвертирующего усилителя-ограничителя. По аналогии выходное поле должно усиливать падающий световой поток. Это усиление должно осуществиться независимо для каждой точки решетки. Должна существовать некоторая минимальная интенсивность I_m. Если интенсивность в какой-либо точке меньше I_m, то считается, что в этой точке находится 0, а если больше или равно, то 1. Максимальное значение интенсивностей равно $\sum_{i=0}^{N} c_i$, наименьшая величина изменения входного сигнала должна быть $1 \Big/ \sum_{i=0}^{N} c_i$. В качестве светоусилительных устройств могут применяться планарные устройства (различные трубки, у которых изображение усиливается на параллельной основе, телевизионные камеры, у которых усиление достигается электронным путем на основе последовательного сканирования, различные электростатические и химические способы репродукции и т. д.).

Исследование возможности применения данной вычислительной системы к распознаванию геометрических форм, определению объектов в реальных изображениях и т. п. показывает их высокую эффективность. Однако применение этой системы для решения других задач малоэффективно. Основные конструктивные достоинства этой системы заключаются в высокой однородности и простоте логической структуры, в отсутствии обычных каналов связи между элементами. Однако очень трудно создать набор транспорантов.

* * *

Рассмотрение проблемно ориентированных систем позволяет сделать следующие выводы:

1) знание конкретной проблемы позволяет упростить конструкцию основных элементов и связей между ними, что снижает требования к технологии и уровню микроминиатюризации;

2) в конструктивном отношении вычислительные системы строятся по модульному принципу. В основе вычислительной системы лежит логический модуль, функции которого определяются решаемой проблемой. Вычислительная система образуется повторением одного и того же модуля. При этом связи между модулями везде идентичны;

3) объем памяти модуля колеблется от одной двоичной единицы до нескольких тысяч;

4) проблемно ориентированные вычислительные системы, хотя и обладают, как правило, полным набором операций, могут эффективно решать только определенный круг задач.

В целом проблемно ориентированные вычислительные системы позволяют решать проблему повышения производительности, как правило, только по тем классам задач, на которые они рассчитаны. Поэтому повысить производительность можно путем создания большого числа проблемно ориентированных вычислительных систем по количеству различных проблем. Недостаток такого направления в том, что при возникновении новой проблемы, не сводящейся к уже известным, требуется создавать новую вычислительную систему. Кроме того, сложные проблемы носят комплексный характер, т. е. в одной проблеме содержатся части различных типов проблем. Поэтому возникает необходимость объединять проблемно ориентированные системы в общую систему соответствующими средствами обмена между ними.

Следует заметить, что в проблемно ориентированных системах из-за стремления к упрощению модуля объем памяти в каждом модуле незначителен, вследствие чего эффективное быстродействие системы получается намного ниже номинального. А это признак того, что оборудование между логическими устройствами и устройствами памяти распределено не оптимальным образом.

Глава 4

УНИВЕРСАЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

4.1. Универсальные системы малой и средней мощности

Как указывалось выше, применение проблемно ориентированных вычислительных систем ограничивается в основном недостаточным объемом памяти логических модулей, отсутствием возможности (у большинства проектов) одновременно выполнять различными модулями различные подпрограммы и ограниченным набором команд. Расширение возможностей модулей для устранения указанных ограничений превращает модуль в вычислительную машину. Поэтому вполне понятна наметившаяся тенденция построения сложных универсальных вычислительных систем путем объединения обычных ЭВМ.

Первые вычислительные системы подобного типа были объединением двух или трех *различных* по своим параметрам вычислительных машин.

Простейшую подобную вычислительную систему представляет объединение машин «Сеак» и «Дисеак» [1, 2]. Эти машины различны по своей конструкции, но имеют одинаковую систему кодирования чисел и команд. Объединяются эти машины как через посредство обычных входных и выходных устройств, так и с помощью каналов связи.

Другим примером служит вычислительная система «Унивак-Ларк» [3], состоящая также из двух различных машин: вычислительной и обрабатывающей. Обе машины универсальны, но ориентированы на выполнение определенных функций. Обрабатывающая — на управление входными-выходными устройствами и передачей данных между вспомогательной и основной памятями, а также на первичную обработку массивов информации (сортировка, перекодировка, обработка программ и т. д.). Вычислительная — на выполнение арифметических операций.

Более совершенна система «Стретч» [4], состоящая из трех различных машин: машины ввода-вывода, работающей непосредственно с устройствами ввода и вывода информации и каналами связи, машины последовательного действия для обработки потока вводных и выводных данных и, наконец, быстродействующей машины параллельного действия для выполнения арифметических и логических операций. Связаны эти машины через общую многоступенчатую память.

К данному типу вычислительных систем относится и система «Пайлот» [5], предназначенная для эффективного проведения экспериментальных работ по исследованию проблем, связанных с обработкой больших массивов данных. «Пайлот» позволяет реализовать характеристики различных существующих устройств обработки данных и выявлять требования к новым устройствам. Благодаря гибкой логической структуре «Пайлот» может работать с большим числом внешних устройств и приспосабливаться к решению широкого круга проблем: обработки научных данных, перевода, автоматического управления электрическими и механическими инструментами, автоматической диспетчеризации воздушных и других сообщений и т. п. «Пайлот» может взаимодействовать одновременно с многими объектами, в том числе удаленными на большие расстояния. Система «Пайлот» состоит из трех машин: быстродействующей для выполнения арифметических и логических операций; вычислительной, работающей в качестве помощника первой (следит за различными частями программы, обрабатывает данные и снабжает ими по мере необходимости первую машину), и машины, управляющей внешними устройствами, которая получает данные об ожидаемых потребностях первой машины во внешних данных, отыскивает данные во внешней памяти или входных устройствах, придает им соответствующую форму и в нужный момент передает в первую машину. Аналогичным образом третья машина выполняет и обратный процесс: переработку и вывод полученных результирующих данных.

Объединять в систему большое число различных машин трудно из-за сложности как разработки и изготовления, так и технической и математической эксплуатации их. Поэтому большие системы стремятся строить из одинаковых ЭВМ. Примером такой вычислительной системы служит СДС-3600 [6], в основу построения которой положен модульный принцип. Имеются три типа высокоскоростных модулей: вычислительный модуль 3604, модуль памяти 3603, модуль связи 3602, а также пульт управления 3601 с входными и выходными устройствами, комбинируя которые можно строить различные конфигурации системы. Модуль 3604 представляет собой вычислительное устройство с развитой системой команд, оперирующее над 48-разрядными двоичными числами и имеющее непосредственный доступ к памяти (модуль

3603). Он может также включать входные и выходные устройства.

Модуль 3603 состоит из двух независимых секций памяти с произвольной выборкой по 16 384 слова (51-разрядных) каждая.

Модуль 3602 управляет обменом информации с входными и выходными устройствами. Он представляет собой ЭВМ с блоком памяти, арифметическим устройством, устройством управления и

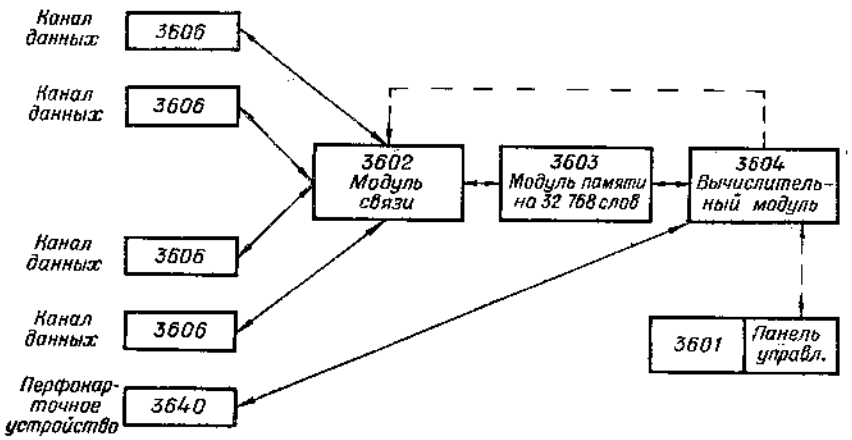


Рис. 16. Основной вариант системы СДС-3600.

четырьмя двусторонними каналами передачи информации. Предусмотрена также возможность подключения еще четырех каналов. Операции обмена информацией могут выполняться независимо и асинхронно с операциями в вычислительном модуле. Обмен осуществляется непосредственно с модулем памяти.

Основной вариант системы СДС-3600 состоит из одного модуля каждого типа (рис. 16). Число модулей памяти может быть увеличено до 8, число блоков связи — до 4 (рис. 17). С каждым модулем 3602 связано не более 8 каналов данных, т. е. в системе может быть 32 канала данных. Каждый модуль памяти подключен не больше чем к пяти независимым каналам с асинхронным управлением.

Предусмотрено также для решения сложных задач объединение нескольких систем в одну общую (рис. 18).

Подобные системы по сравнению с неоднородными кроме преимуществ в разработке, изготовлении и эксплуатации обладают и более высокой надежностью, так как функции машины или модуля, вышедших из строя, могут взять на себя другие такие же машины или модули.

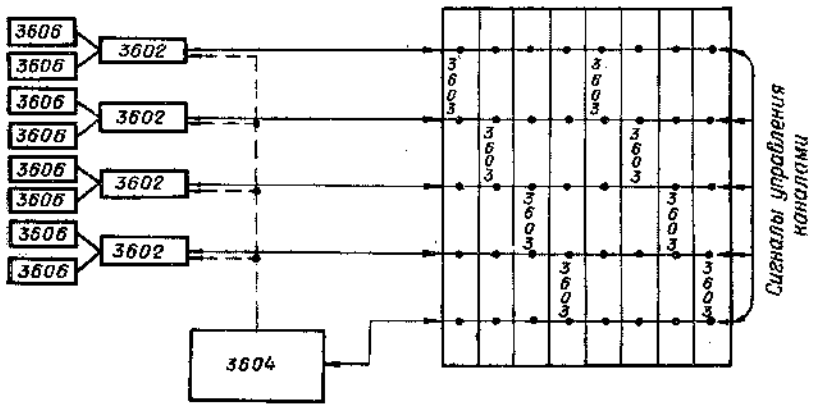


Рис. 17. Полная схема вычислительной системы СДС-3600.

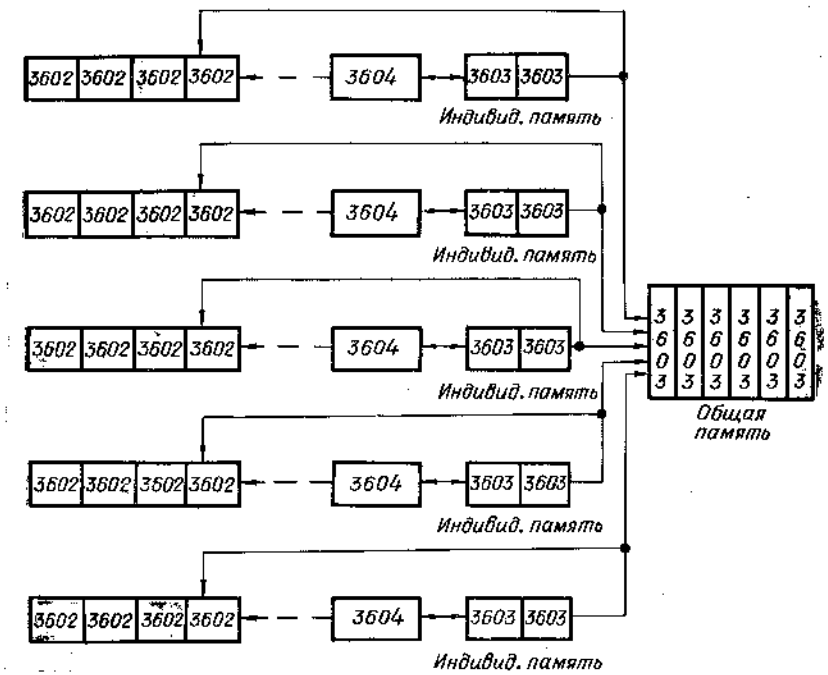


Рис. 18. Объединение систем СДС-3600 в общую систему.

4.2. Сети вычислительных машин

За последние годы появилась тенденция объединять ЭВМ в масштабах какого-либо района или всей страны с помощью каналов связи в единую информационную сеть аналогично единым энергетическим системам. Информационные сети позволяют ЭВМ оперативно обмениваться информацией, минуя промежуточные инстанции. Важно, что при такой работе заметно повышается коэффициент использования ЭВМ и оперативность решения задач, так как загрузкой ЭВМ можно управлять с единого диспетчерского пункта и пересылать программы, данные, подлежащие обработке, и результаты вычислений по каналам связи.

Эти достоинства сетей ЭВМ общеизвестны, и мы на них останавливаться не будем. Менее известно применение сетей ЭВМ для решения трудоемких задач, с которыми не может справиться одна даже наиболее производительная ЭВМ. В этом случае задача разбивается на части, которые одновременно выполняются различными ЭВМ. После окончания своей части задачи машина передает результат по каналу связи другим ЭВМ и сама получает новые данные для выполнения следующей части. Таким образом, вычислительная сеть работает в режиме вычислительной системы и может рассматриваться как ее частный случай.

Вычислительные сети отличаются от других вычислительных систем тем, что из-за удаленности ЭВМ друг от друга время обмена информацией между ЭВМ может стать сравнимым, а если не предпринимать специальных мер, то и значительно превышающим время вычислений.

Одной из мер уменьшения времени обмена может служить такое разделение задачи на части, при котором объем информации, передаваемой между ЭВМ, был бы минимален.

Другой способ сокращения времени обмена, разобранный в [7], заключается в наиболее полном использовании каналов связи. В идеальном случае вычислительный процесс строится таким образом, что в канал связи все время поступает такой поток информации, который соответствует предельной пропускной способности канала. Это может быть достигнуто, например, следующим способом. Для простоты рассмотрим сеть из двух ЭВМ, соединенных двумя каналами связи. По одному идет передача информации от первой ЭВМ ко второй, а по другому — от второй к первой. Предположим, что на протяжении данного числа циклов каждая ЭВМ выполняет какую-либо подпрограмму над результатами, полученными другой ЭВМ. Если при этом последовательности чисел, выходящие из каждой ЭВМ, могут быть разбиты на группы, над любой из которых может выполняться подпрограмма независимо от других групп, и если скорости вычислений и передачи по линии связи согласованы, то дополнительные затраты времени на обмен информацией благодаря достаточно полному совмещению процесса обмена с выполнением операций практически будут равны нулю.

Согласование во времени может быть достигнуто как путем соответствующего расчленения задачи, так и посредством применения большого числа ЭВМ. Согласование скоростей работы при неравномерном поступлении данных может достигаться использованием буферных памятей. Одновременно применение буферных памятей позволяет ослабить требования к членению вычислительного процесса, так как становится возможным существенное увеличение размеров каждой группы.

В тех случаях, когда время обмена между ЭВМ удается сделать небольшим, сеть по сравнению с одной ЭВМ может дать выигрыш во времени решения задачи, пропорциональный отношению суммарной производительности ЭВМ сети к производительности одной ЭВМ. Для сетей, состоящих из одинаковых ЭВМ, выигрыш во времени будет пропорционален числу ЭВМ.

В некоторых случаях этот выигрыш может быть и несколько больше. Действительно, пусть объем данных v решаемой задачи будет больше объема оперативной памяти S одной ЭВМ и меньше суммарного объема оперативных памятей всех M машин сети $S < v < SM$. Если в ходе решения задачи эти данные требуются многократно, то при решении на одной ЭВМ время на ввод и вывод данных может составить основную долю затрат времени. Этого может не быть у сети машин, так как все данные разместятся в оперативных памятях.

Применение вычислительных сетей в качестве вычислительных систем тормозится отставанием как в разработке общих методов решения задач на основе массового распараллеливания, так и в вопросах, специфичных для сетей. К последним относятся методы уменьшения относительных затрат времени на обмен информацией между ЭВМ (по сравнению со временем счета), повышения скорости и надежности передачи: разработка рациональных систем коммутаций между ЭВМ сети (речь идет о создании такой системы связи между ЭВМ, которая была бы проще, чем система, где каждая ЭВМ связана с каждой, и которая не отличалась бы существенно от нее по эффективности); методы эксплуатации сетей. Это довольно важно, так как сети из современных ЭВМ не могут использоваться в качестве вычислительных систем, если не будет уделено достаточное внимание вопросу надежности. Как известно, надежность современных ЭВМ не особенно велика и падает по мере удаления от момента профилактики. Поэтому в качестве одной из мер

повышения надежности сети может быть, например, одновременность проведения профилактики для всех ЭВМ сети и использование сети в качестве вычислительной системы сразу же после профилактики.

Решение этих вопросов ускорило бы использование сетей ЭВМ в качестве вычислительных систем и дало бы инструмент для решения задач, объем вычислений у которых превышает возможности наиболее производительных существующих ЭВМ.

4.3. Универсальные системы с переменной структурой

Как мы видели (гл. 3), большинство проектов вычислительных систем предназначается для решения определенного круга проблем и малоэффективно при решении других задач. Возникает вопрос, может ли данное направление стать основным при построении вычислительных систем высокой производительности? Для этого необходимо, чтобы для каждого класса задач была создана своя вычислительная система. Иными словами, необходимо создать парк проблемно ориентированных вычислительных систем, который смог бы удовлетворить все нужды науки и техники.

На этом пути, как уже упоминалось, встречаются, по крайней мере, две трудности:

1) в ходе научно-технического прогресса постоянно возникают новые задачи и проблемы. По мере их возникновения необходимо будет разрабатывать все новые и новые вычислительные системы. При этом сроки разработки таких систем начинают играть первостепенную роль, так как именно от них начинают зависеть и сроки решения самих проблем. В идеале время от появления новой задачи до создания вычислительной системы должно быть соизмеримо со временем решения задачи. Таким образом, возникает необходимость обеспечить очень высокую скорость изготовления вычислительных систем, состоящих из числа элементов на несколько порядков большего, чем у самых крупных современных ЭВМ;

2) многие проблемы комплексные и требуют решения задач различных классов. Нередко эти задачи взаимозависимы и должны решаться совместно. Значит, возникает необходимость объединять различные проблемно ориентированные системы в единый комплекс либо переходить к разработке вычислительных систем с более широкими возможностями, т. е. к универсальным вычислительным системам.

Универсальные вычислительные системы свободны и от первого недостатка, так как на их изготовление вполне допустимы

гораздо большие затраты времени. Однако из-за необходимости удовлетворять требованиям различных задач универсальные ВС сложнее проблемно ориентированных.

Возникает также вопрос, как в достаточно полной мере выполнить требование универсальности.

Под термином *универсальная вычислительная машина или система следует понимать такое устройство, производительность которого не испытывает больших колебаний при решении задач различных классов*. Под это определение, строго говоря, не подходят имеющиеся вычислительные машины и системы, которые создавались как универсальные, но не обладающие возможностью изменять параметры (устройства с *жесткой структурой*). У этих машин соотношение между быстродействием и объемом памяти, между быстродействием и скоростью обмена информацией с другими объектами, система команд и другие параметры, определяющие производительность, устанавливаются при разработке их конструктором и не могут изменяться перед решением той или иной задачи. Если даже эти параметры выбраны с учетом имеющихся классов задач, то нет никакой гарантии, что не появятся такие задачи (или методы их решения), для которых эти параметры окажутся неудовлетворительными и которые вследствие этого не будут эффективно решаться данной системой или машиной. Положение усугубляется тем, что их параметры устанавливаются на основе весьма приближенных сведений о требованиях, предъявляемых задачами. Дело в том, что пока нет достаточно полного анализа имеющихся задач, а поэтому и не существует строго обоснованных требований к вычислительным устройствам.

Анализ задач, безусловно, позволил бы улучшить параметры вычислительных средств. Однако этим путем можно только расширить область применения данной электронной вычислительной машины или вычислительной системы, но не сделать ее в полном смысле универсальной.

Выходом из указанного положения может служить переход к созданию универсальных вычислительных систем с *переменной структурой*, т. е. таких вычислительных систем, основные параметры которых (отношение объема памяти к быстродействию, скорость обмена информацией внутри системы и с внешними объектами, система команд) могут *программно* изменяться до или во время решения задачи. Заметим, что словом «программно» из данной категории исключаются вычислительные системы, у которых указанные выше параметры могут изменяться вручную путем перестройки схем и добавлением новых блоков, как у СДС-3600 (см. 4.1).

Перестройка структуры вручную — это, безусловно, шаг вперед по сравнению с устройствами с жесткой структурой. Но вряд ли это может решить вопрос как из-за длительности процесса перестройки, так и вследствие ограниченности возможностей изменения параметров. В дальнейшем под термином вычислительная система с переменной структурой будут пониматься системы с программной настройкой.

Одним из первых примеров программной настройки структуры служит ЭВМ ТХ-2 [8], в которой предусмотрена возможность программно управлять изменением длин слов, делая их по желанию 36-, 27-, 18- или 9-разрядными, числом одновременно выполняемых операций (при этом можно выполнить операцию над 36-разрядным словом, либо над двумя 18-, либо 27- и 9-разрядными словами, либо над четырьмя 9-разрядными словами), выбором двух любых блоков памяти из четырех.

Возможность построения ЭВМ с переменной структурой исследуется в работах [9—11]. В их основу положена идея построения ЭВМ из двух основных частей: постоянной, представляющей собой ЭВМ общего назначения с жесткой структурой, и переменной, состоящей из совокупности вычислительных устройств, которые могут программно перестраивать свою структуру. Эти две части объединяются с помощью специального блока управления.

Одно из возможных направлений в создании вычислительных систем с переменной структурой предложено в работе [12]. Дальнейшее развитие это направление получило как в работах авторов, так и в других работах, выполненных под руководством авторов и опубликованных в основном в сборниках трудов Института математики Сибирского отделения АН СССР «Вычислительные системы». В этих работах была поставлена задача и исследуются пути разработки такой вычислительной системы, которая была бы универсальной и рассчитана на решение задач произвольного класса с производительностью 10^9 *опер/сек* и выше, максимально простой в изготовлении, надежной, простой в применении.

Для удовлетворения этим требованиям в основу данного направления были положены следующие принципы:

1) однородность структуры. Это означает, что вычислительная система строится из одинаковых и одинаково связанных друг с другом элементов;

2) в качестве основного элемента ВС служит универсальная вычислительная машина, имеющая свою систему управления, арифметическое устройство, оперативную память и устройства ввода и вывода. Такие машины далее будем называть *элементарными машинами* (ЭМ);

3) набор команд не фиксирован жестко, а может изменяться в зависимости от требований задачи. В систему команд включены также операции, управляющие каналами связи;

4) объем памяти и разрядность чисел каждой элементарной машины могут изменяться;

5) может изменяться само число элементарных машин.

Создание универсальной вычислительной системы с переменной структурой как одного из основных направлений повышения производительности ЭВМ — проблема сложная, требующая широкого фронта исследований во многих областях науки и техники и прежде всего в физике, математике, химии, вакуумной технике, радиоэлектронике.

В свою очередь проблема создания ВС зависит от решения других проблем, особенно разработки технологии, которая позволила бы создавать большое число элементов в едином автоматизированном процессе. Все остальные проблемы зависят от ее успешного решения и направлены в основном на облегчение требований и снятие различных препятствий при решении проблемы технологии. Чтобы создать вычислительные системы, необходимо разработать следующее:

1) методы решения задач на ВС (теорию параллельных алгоритмов, анализ и классификацию задач, которые предстоит решать на ВС, теорию программирования для ВС, входные языки, трансляторы, библиотеки и подпрограммы, вопросы математической эксплуатации и др.);

2) логические основы построения ВС (должны быть рассмотрены основные принципы построения ВС, исследованы возможные варианты ВС, структура элементарных машин, разработана система коммутаций между машинами, дана оценка надежности, производительности, экономичности и т. п.);

3) средства обмена информацией между ВС и внешним миром (принципы построения систем для распознавания зрительных и звуковых образов, вопросы приема и передачи информации по каналам связи и т. д.);

4) физико-технологические основы построения ВС (исследования различных физических явлений, пригодных для создания на их основе элементов ВС с заданными свойствами, оценка предельных возможностей элементов по быстродействию, размерам, потребляемой мощности, создание и исследование новых технологических способов производства большого числа элементов, выяснение возможности их автоматизации и т. д.);

5) методы и средства автоматизации процессов проектирования исследований и изготовления ВС (разработка машинных методов проектирования, расчета элементов, синтеза схем, автома-

тизация физических и технологических исследований, производства элементов и ЭМ, методики исследования сложных систем).

Хотя все эти проблемы объединены общей целью, каждая из них может разрабатываться в значительной мере независимо от других и представить самостоятельный интерес, выходящий за рамки построения ВС.

Представление алгоритмов в виде совокупности параллельно выполняемых операторов, несомненно, может оказаться полезным для теории алгоритмов.

Создание системы программирования для ВС включает в себя проблемы управления системой машин, работающих над выполнением определенного алгоритма. Эта проблема возникает и на более низких уровнях организации ЭВМ, например в сетях ЭВМ и системах, имеющих ряд самостоятельных арифметических и логических блоков и т. п.

Разработка основ построения ВС с переменной структурой может оказаться полезной при создании специализированных устройств со структурой, меняющейся в зависимости от изменения параметров и характера решаемой задачи.

Методы распознавания звуковых и зрительных образов имеют большое самостоятельное значение. Кроме того, полученные результаты могут быть приложены к решению различных проблем, формулируемых как проблемы распознавания образов.

Разработка физических и технологических основ построения ВС — кардинальная задача всей радиоэлектроники. Ее решение позволяет создать новую техническую базу для изготовления многочисленной радиоэлектронной аппаратуры. Для вычислительной техники это создает возможность наладить массовый выпуск недорогих ЭВМ, предназначенных для решения задач средней трудности, что увеличит общую мощность парка ЭВМ. Появляется также возможность построить большой парк специализированных ЭВМ для решения типовых задач.

Значение автоматизации проектирования, физических и технологических исследований, производства элементов ВС далеко выходит за рамки вычислительной техники. На этой же основе могут проектироваться и изготавливаться многие радиоэлектронные приборы.

Некоторые вопросы, например достижение максимальной надежности, моделирование сложной системы на ЭВМ и другие,— общие. С ними приходится сталкиваться при проектировании не только электронных приборов, но и сложных машин, установок, самолетов, судов и т. д. Автоматизация научных исследований имеет большое значение для повышения эффективности исследований различных физических, химических и других явлений.

Глава 5

МАКРОСТРУКТУРА УНИВЕРСАЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ С ПРОГРАММНО ИЗМЕНЯЕМОЙ СТРУКТУРОЙ

5.1. Общие сведения

В универсальной вычислительной системе с программно изменяемой структурой можно изменять программным способом число элементарных машин (ЭМ) в системе, систему коммутаций между ЭМ, структуру ЭМ (набор операций, разрядность кода, объем памяти). В зависимости от того, предусмотрена возможность программного изменения структуры ЭМ или нет, различаются *вычислительные системы с полностью изменяемой структурой и с частично изменяемой структурой*. В дальнейшем изложении (для сокращения) под термином универсальная вычислительная система (УВС) будем понимать универсальную вычислительную систему с полностью или частично программно изменяемой структурой.

На макроструктурном уровне рассмотрения УВС будем различать три основные части системы: элементарные машины, коммутатор и каналы связи. Коммутатор и каналы связи вместе образуют систему связи УВС. УВС могут программно приспосабливаться наилучшим образом к особенностям данной задачи путем изменения системы коммутации, числа ЭМ, выбора соответствующей системы операций, объема памяти, разрядности кода. При этом устанавливается соответствие между параметрами и характеристиками задачи и структурой УВС.

На макроструктурном уровне рассмотрения вычислительные системы с полностью и с частично изменяемой структурой не различаются. Глава 5 посвящена изложению основ построения вычислительных систем с частично изменяемой структурой. Особенности построения элементарной машины с переменной структурой изложены в 5.4 и гл. 6.

5.2. Основные определения

Пусть дан конечный автомат К с *входными каналами* x_{il} и *выходными каналами* z_{il} ($i = 0,1, \ldots, n$, $l = 1, 2, 3$). Пусть каждый из каналов может пребывать в u попарно различимых состояниях. Поставим в соответствие каждому такому состоянию символ из некоторого конечного множества символов A (число символов равно u). Множество A будем называть *алфавитом*, а отдельные символы этого множества — *буквами*. *Словом* в алфавите A будем называть всякую упорядоченную последовательность букв из множества A. Будем предполагать, что в каждый дискретный момент $t = 1, 2, \ldots$ в каждый входной канал поступает и из каждого выходного канала выдается по одной букве. Будем обозначать через $x_{il}(t)$ и $z_{il}(t)$ буквы, проходящие в момент t по каналам x_{il}, z_{il} соответственно.

Назовем *алфавитом состояний* автомата К алфавит $B = \{H_d C_{ij}^{(l)}\}$, где

$$d = 1, 2;\ l = 1, 2, 3;\ i, j = 0,1, \ldots, n;\ H_d \in \{0,1\};\ C_{ij}^{(l)} \in \{0,1\}.$$

Пусть даны алфавиты $A_{i1} = \{a_1, a_2, \ldots, a_m\}$; $A_{i2} = \{p_1, p_2\}$, $A_{i3} = \{\alpha_1, \alpha_2, S_{rs}^{(q)}\}$, составленные из символов, поступающих во входные каналы x_{i1}, x_{i2}, x_{i3} соответственно; $q = 1, 2, 3$; $r, s = 0, 1, \ldots, n$.

Назовем *коммутационно-настроечным* конечный автомат К, заданный системой канонических уравнений:

$$\left.\begin{aligned}
&z_{jl}(t) = x_{il}(t) \,\&\, C_{ij}^{(l)}(t) \,\&\, H_2(t), \quad i, j = 0, 1, \ldots, n;\ l = 1,2;\\
&z_{j3}(t) = x_{i3}(t) \,\&\, C_{ij}^{(3)}(t) \,\&\, H_2(t) \quad \text{для } x_{i3} \in \{\alpha_2, S_{rs}^{(q)}\};\\
&z_{j3}(t) = x_{i3}(t) \,\&\, C_{ij}^{(3)}(t) \,\&\, H_1(t) \text{ для } x_{i3} \in \{\alpha_1\};\\
&H_1(t+1) = H_1(t) \vee (x_{i3}(t) = \alpha_1);\\
&H_2(t+1) = H_2(t) \vee (x_{i3}(t) = \alpha_2);\\
&C_{ij}^{(l)}(t+1) = x_{i3}(t) \,\&\, H_1(t) \vee C_{ij}^{(l)}(t) \,\&\, \overline{H_1(t)} \text{ для } x_{i3} \in \{S_{rs}^{(q)}\};\\
&\qquad\qquad S_{rs}^{(q)} \in \{0, 1\};\\
&\qquad i, j = 0, 1, \ldots, n;\ l = 1, 2, 3.
\end{aligned}\right\} \quad (5.1)$$

Концы каналов, не премыкающие к автомату, назовем *внешними полюсами* автомата, концы каналов, примыкающие к автомату, — *внутренними полюсами*. Будем различать также *входные* и *выходные* полюса.

Под *соединением* будем понимать отождествление внутренних полюсов входных каналов с внутренними полюсами выходных каналов. В матрице *соединений* $[C_{ij}^{(l)}]$ строки соответствуют внутренним входным полюсам, а столбцы — выходным. В случае отождествления x_{il} внутреннего входного полюса с z_{jl} внутренним выходным полюсом $C_{ij}^{(l)} = 1$, а при отсутствии отождествления полюсов $C_{ij}^{(l)} = 0$.

Пусть дан программный автомат В. М. Глушкова [1] с универсальным набором операций, т. е. в наборе имеются операции ввода и вывода, пересылки, переадресации на ± 1 и условного перехода по значению предиката ω ($\omega = 1$ — при точном совпадении слов, $\omega = 0$ — в противном случае) или им эквивалентные. Добавим к нему входные и выходные каналы y_{0l} и ω_{0l} ($l = 1, 2, 3$).

Введем также дополнительный набор операций $\{\pi_1, \pi_2, Y, H\}$. Пусть $b + \gamma c$ ($\gamma = 1, 2, \ldots, m$) — номера слов, составленных из букв алфавита A_{i1}.

Операцию π_1, которая сводится к последовательной передаче слов с номерами $b + \gamma c$ из памяти автомата в выходной канал ω_{01}, назовем *операцией передачи*:

$$\pi_1 : [b + \gamma c] \Rightarrow \omega_{01},$$

где b и c — константы.

Операцию π_2, которая сводится к последовательной передаче слов с номерами $b + \gamma c$ в память автомата по входному каналу y_{01}, назовем *операцией приема*:

$$\pi_2 : y_{01} \Rightarrow [b + \gamma c].$$

Операцию Y, при которой совершается переход к очередной команде в случае выполнения условия $P = 1$ и к команде с номером b_i — в противном случае, назовем *операцией обобщенного условного перехода*.

Это условие определяется следующим образом. Будем полагать, что в выходном канале ω_{02} программного автомата появляется символ p_1, если $\varphi(\omega, y_{02}) = 1$, где φ — функция алгебры логики от двух переменных.

Пусть имеется m программных автоматов. Введем функцию $P(\omega_1, \ldots, \omega_m) \in \{0,1\}$, где ω_i — значение предиката в i-м программном автомате.

Схема, реализующая функцию $P(\omega_1, \ldots, \omega_m)$, есть суперпозиция схем, реализующих функции $\varphi(\omega_i, y_{02}^i)$ в каждом автомате. При $P(\omega_1, \ldots, \omega_m) = 1$ будем полагать, что во входном канале y_{02} каждого программного автомата появляется символ p_2.

Операцию Н, при которой из памяти программного автомата в выходной канал w_{03} передается последовательность слов вида либо $\alpha_1 \Rightarrow w_{03}$; $\{S_{rs}^{(q)}\} \Rightarrow w_{03}$; $\alpha_2 \Rightarrow w_{03}$, либо $\{S_{rs}^{(q)}\} \Rightarrow w_{03}$; $\alpha_2 \Rightarrow w_{03}$ ($r, s = 0, 1, \ldots, n$; $q = 1, 2, 3$), назовем *операцией настройки*.

Будем считать возможным прием в память программного автомата информации, поступающей во входной канал y_{03}.

Программный автомат с универсальным набором операций, конечным объемом памяти, с дополнительными входными y_{0l} и выходными w_{0l} каналами, дополнительным набором операций $\{\pi_1, \pi_2, Y, H\}$ будем называть *функциональным автоматом* Ф.

Отождествим полюсы x_{0l}, z_{0l} коммутационно-настроечного автомата К с полюсами w_{0l}, y_{0l} функционального автомата Ф.

Схему из автоматов Ф и К, полученную в результате такого отождествления, назовем *элементарной машиной* (ЭМ).

В элементарной машине полюсы x_{il}, где $i = 1, 2, \ldots, n$, будем называть *входными*, а полюсы z_{il} — выходными.

Установим следующий способ композиции элементарных машин: входные полюсы x_{il} и выходные z_{il} одной ЭМ попарно отождествляются с соответствующими выходными и входными полюсами z_{jl} и x_{jl} другой ЭМ ($l = 1, 2, 3$). Такие ЭМ будем называть соседними.

Будем говорить, что элементарные машины m_1 и m_M соединены цепью, если имеется такая последовательность машин m_1, m_2, $\ldots$, m_{M-1}, m_M, что для каждой m_i ($i = 2, \ldots, M - 1$) машины m_{i-1} и m_{i+1} будут соседними.

Под универсальной вычислительной системой с переменной структурой будем понимать такую совокупность элементарных машин, в которой две любые элементарные машины соединены хотя бы одной цепью.

Универсальные вычислительные системы с переменной структурой, состоящие из однотипных элементарных машин, будем называть однородными, если каждая элементарная машина имеет одинаковое число соседних.

5.3. Основные свойства универсальных вычислительных систем

Функциональная целостность. Для систем важную роль играет способ достижения взаимно согласованных действий частей системы, или, иначе говоря, то объединяющее начало, благодаря которому система может функционировать как единое целое.

Вычислительные системы, отличающиеся только способом организации взаимодействия элементарных машин (модулей), вообще говоря, могут обладать различной производительностью.

Рассмотрим способы, с помощью которых достигается функциональная целостность у известных вычислительных систем.

Жесткое управление. Примером ВС, реализующей этот способ управления, может служить система Унгера (см. 3.1). В этой ВС все модули должны одновременно выполнять одну и ту же команду, задаваемую центральной машиной.

Жесткое управление с частичной автономией. В системе СОЛОМОН (см. 3.2) модули, как и в системе Унгера, должны одновременно выполнять одну и ту же команду, поступающую из центральной машины, но некоторым модулям при определенных условиях дается право игнорировать эту команду.

Для указанных способов управления характерно, что функции управления сосредоточены в центральной машине. Обратная связь от модулей к центральной машине осуществляется через информацию, которую она собирает, руководствуясь заложенной в ней программой. Обмен информацией производится между соседними модулями по команде из центральной машины.

Ясно, что такой способ управления эффективен при решении только некоторых задач.

Система управления с «директором». При этом весьма распространенном способе управления в системе имеется центральная машина — «директор». В отличие от системы с жестким управлением машина-«директор» передает остальным не отдельные команды, а подпрограммы, вообще говоря, различные для различных машин. После выполнения своей подпрограммы машина посылает сигнал в машину-«директор» и ждет ее дальнейших «распоряжений». Непосредственный обмен информацией между машинами обычно отсутствует и происходит через машину-«директор» или через общую память.

Организация взаимодействия машин через общую память. Этот способ согласования работы машин, реализованный в системе СДС-3600 (см. 4.1), в известной мере противоположен описанным выше. Система состоит из равноправных машин и общей памяти, к которой может обращаться любая из машин. Моменты обращения и адреса ячеек общей памяти задаются программами, заложенными в каждой из машин.

Согласование работы машин осуществляется через информацию, которую машины помещают и берут из общей памяти.

Управление работой машин в таких системах не носит оперативного характера. При интенсивном обмене информацией даже при сравнительно небольшом числе машин в системе производительность может существенно снизиться. Применение такого способа взаимодействия оправдано у ВС, предназначен-

ных для выполнения вычислительных процессов, разделяющихся на малозависимые части.

Управление с помощью схем прерывания. За последнее время наметилась тенденция согласовывать работу машин с помощью схем прерывания, первоначально разработанных для согласования работы ЭВМ с внешними источниками информации. У каждой машины имеются выходные каналы, по которым она в определяемые программой моменты времени передает информацию в другие машины, и входные каналы. Поступающую по ним информацию машина может в зависимости от своего состояния либо принимать, прерывая процесс вычислений, либо игнорировать. В каждый момент времени могут выполняться парные обмены между машинами. Схема прерывания позволяет не только обмениваться информацией, но и согласовывать работу машин. Появление информации в одном или нескольких указанных в программе входных каналах может восприниматься как выполнение условия, определяющего переход к новой подпрограмме. Осуществление такого способа управления для большого числа машин может встретить значительные трудности.

Из рассмотренного видно, что все приведенные способы организации взаимодействия машин в системе не обладают достаточной гибкостью и могут эффективно применяться только для проблемно ориентированных ВС. Большинство из них к тому же трудно осуществить на практике при большом числе машин. Поэтому указанные способы мало пригодны для универсальных вычислительных систем высокой производительности. Для УВС необходимо, чтобы допускался обмен информацией одновременно между большим числом машин при произвольных конфигурациях схемы обмена и чтобы была возможность гибкого изменения хода вычислений в зависимости от результатов, полученных в различных машинах.

Основная трудность организации обмена связана с заданием адресов: откуда информация должна быть взята и куда помещена. С увеличением объемов памятей и числа машин размеры адресов становятся практически неприемлемыми.

При организации взаимодействия большого числа машин основная трудность заключается в том, что функция, определяющая значение обобщенного условия, зависит от многих переменных, число которых может быть равно числу машин, и имеет сложный вид.

Одним из путей преодоления указанных трудностей может быть переход к программно изменяемой структуре схем управления и обмена информацией. При этом процесс управления и обмена выполняется в два этапа. Сначала с помощью специальных команд настройки выделяются машины, участвующие в выработке обобщенного условия, и машины, переходящие к новым подпрограммам при выполнении этого условия; передающие машины соединяются каналами связи с принимающими.

Затем выполняется собственно изменение хода вычислений и обмен. В соответствующих командах управления уже нет необходимости указывать, какие машины управляющие, а какие исполняющие. При обмене передающая машина посылает информацию в канал связи, не указывая адресата, а принимающие машины получают информацию из канала, не «заботясь» об адресе отправителя.

Покажем, как эти процессы могут выполняться с помощью операций H, π_1, π_2, Y, введенных ранее (см. 5.2).

Операция настройки H изменяет матрицы соединений C^1_{ij}, C^2_{ij}, C^3_{ij}, которые управляют соответственно коммутациями каналов $x_{11}, \ldots, x_{n1}$; $z_{11}, \ldots, z_{n1}$ для обмена информацией, каналов $x_{12}, \ldots, x_{n2}$; $z_{12}, \ldots, z_{n2}$ для передачи информации, определяющей обобщенный условный переход, каналов $x_{13}, \ldots, x_{n3}$; $z_{13}, \ldots, z_{n3}$ для передачи информации, необходимой для выполнения операции настройки. Тем самым операция настройки может реализовывать всевозможные конфигурации схем обмена, выделять управляющие и исполняющие машины, устанавливать пути передачи информации, управляющей настройкой, для последующей операции настройки. В частности, команда настройки может разбить систему на независимо работающие подсистемы. Команда может быть выполнена из любой машины.

Для выполнения обмена в программе передающей машины должна стоять команда π_2, а во всех машинах, принимающих эту информацию, команда π_1. Согласование этих команд во времени (синхронизация) выполняется с помощью команды обобщенного условного перехода Y. Команда Y устанавливает значение функции, определяющей обобщенное условие, и осуществляет переход к новой подпрограмме.

Вид функции может быть задан произвольным, но, как показывает анализ задач (см. гл. 8), по-видимому, можно ограничиться двумя случаями, когда образуется конъюнкция и дизъюнкция от M переменных, где M — число машин в системе. При этом всем переменным, соответствующим машинам, не участвующим в выработке обобщенного условия, с помощью настройки придается значение в первом случае 1, во втором — 0.

Изменение вида функции может осуществляться либо с помощью операции настройки (коммутацией соответствующих входных и выходных каналов), либо введением двух команд обобщенного условного перехода Y_1 и Y_2.

Набор команд $\{H, \pi_1, \pi_2, Y\}$ следует рассматривать как минимальный полный набор операций. На практике для облегчения программирования этот набор, по-видимому, должен быть расширен. Удобно, например, включить в него команды обобщенного безусловного перехода, с помощью которых можно из одной машины изменять ход вычислений в любой другой.

Примеры схем, с помощью которых можно реализовать операции системы, приводятся в данной главе. Примеры использования этих команд для построения алгоритмов приведены в гл. 7 и 8. Как будет видно из этих примеров, ни реализация схем, ни построение соответствующих алгоритмов не встречают больших затруднений.

Универсальность вычислительной системы. В. М. Глушковым [1] доказано, что программный автомат с универсальным набором операций и неограниченной памятью реализует нормальный алгоритм А. А. Маркова, т. е. является универсальным.

Теорема 1. Универсальная вычислительная система с частично переменной структурой при отсутствии ограничений на число элементарных машин универсальна в смысле В. М. Глушкова.

Выделим любую из элементарных машин системы. По определению она обладает универсальным набором операций. Покажем, что она может использовать неограниченный объем памяти. Действительно, суммарный объем памятей ЭМ системы может быть сделан сколь угодно большим, так как на число машин по условию теоремы не наложены ограничения. С помощью операций настройки H, обобщенного условного перехода Y, операций приема π_1 и передачи π_2 можно осуществить обмен информацией между данной ЭМ и памятями всех остальных ЭМ системы, что и доказывает теорему.

Быстродействие (производительность) **вычислительной системы.** Скорость работы элементарной машины можно измерять числом рабочих циклов (цикл соответствует времени выполнения команды), выполняемых ЭМ в течение секунды. Так как в общем случае рабочие циклы (в зависимости от команды) могут иметь различную длительность, то при определении быстродействия ЭМ подсчитывается среднее число команд в единицу времени в предположении, что ЭМ работает все время с оперативным запоминающим устройством, без учета ввода, вывода и работы с внешним запоминающим устройством.

По аналогии с понятием номинального быстродействия автомата [1] введем понятие номинального быстродействия элементарной машины.

Номинальным быстродействием $v_н$ элементарной машины будем называть такое быстродействие, которое осуществляется ЭМ с оперативным запоминающим устройством и выражается средним числом операций (команд), выполняемых ЭМ в секунду.

Пусть дана универсальная вычислительная система из M элементарных машин ($M = 1,2, .$). Назовем *номинальным быстродействием системы* $V_н$ произведение числа машин в системе на номинальное быстродействие ЭМ:

$$V_н = v_н M. \tag{5.3}$$

Номинальное быстродействие системы $V_н$ *— величина переменная и изменяется дискретно с шагом дискретности* $v_н$.

Учитывая время, затрачиваемое на ввод и вывод информации, на обращение к внешним запоминающим устройствам, на обнаружение и устранение неисправностей в системе, на обмен информацией между ЭМ в системе, на выполнение различных по своей сложности операций и т. д., выразим быстродействие системы числом выполняемых ею в единицу времени стандартных операций (через которую выражены все операции) и назовем его *эффективным быстродействием* (производительностью) системы.

Эффективное быстродействие зависит от задач, которые решаются на УВС. В связи с этим целесообразно ввести понятие *среднего эффективного быстродействия* $V_э$ *УВС* и *среднего эффективного быстродействия* $v_э$ *ЭМ*. Среднее эффективное быстродействие ЭВМ определяется следующим образом [1].

Выделяется конечное множество типовых задач $R_1, R_2, \ldots, R_m$ с весами $p_1, p_2, \ldots, p_m$, $\sum_{i=1}^{m} p_i = 1$, пропорциональными времени их решения. Тогда среднее эффективное быстродействие v ЭВМ определяется по формуле

$$\frac{1}{v} = \frac{p_1}{v_1} + \frac{p_2}{v_2} + \cdots + \frac{p_m}{v_m}, \tag{5.4}$$

где $v_1, v_2, \ldots, v_m$ — эффективные быстродействия ЭВМ при решении задач $R_1, R_2, \ldots, R_m$. Аналогично для вычислительной системы с числом машин M среднее эффективное быстродействие системы будет равно

$$\frac{1}{V_э(M)} = \frac{p_1}{V_{э1}(M)} + \frac{p_2}{V_{э2}(M)} + \cdots + \frac{p_m}{V_{эm}(M)}, \tag{5.5}$$

где $V_{э1}$ (M), $V_{э2}(M), \ldots, V_{эm}(M)$ — эффективные быстродействия системы при решении задач $R_1, R_2, \ldots, R_m$.

Среднее эффективное быстродействие ЭМ

$$v_э(M) = \frac{V_э(M)}{M} \tag{5.6}$$

зависит от числа машин в системе. Эта зависимость возникает главным образом из-за изменения относительных затрат времени на выполнение операций обобщенного условного перехода (синхронизацию) и обмена как между ЭМ, так и между внешней и оперативной памятью внутри каждой ЭМ.

Под *критерием цены эффективного быстродействия системы* $Q_э$ будем понимать отношение среднего эффективного быстродействия ЭМ к стоимости изготовления ее $C_э$:

$$Q_э = \frac{v_э}{C_э}. \tag{5.7}$$

Остановимся теперь на сравнении быстродействия УВС с быстродействием универсальной вычислительной машины, имеющей тот же набор операций и то же время их выполнения, что и в элементарной машине. При сравнении будем использовать два типа универсальных вычислительных машин (УВМ), различающихся по соотношению объемов внешней и оперативной памяти. Будем предполагать, что общий объем памяти УВМ для обоих типов равен объему памяти УВС. Для первого типа УВМ будем предполагать, что вся память является оперативной. Обозначим через $S_{мо}$ объем оперативной памяти УВМ (в двоичных разрядах), а через $S_{эм}$ — объем памяти элементарной машины. Тогда для первого типа УВМ объем памяти $S_{мо} = S_{эм} M$, а номинальное быстродействие УВМ $v_м = v_н$. Если предположить, что для широкого круга задач существуют параллельные алгоритмы, допускающие расчленение вычислений на M ветвей [8], то отношение быстродействия УВС к быстродействию УВМ

$$R = \frac{V_н}{v_м} = \frac{v_н M}{v_н} = M. \tag{5.8}$$

Для второг типа УВМ будем предполагать, что общий объем памяти

$$S_м = S_{мо} + S_{мв}, \tag{5.9}$$

где $S_{мо} = S_{эм}$, а объем внешней памяти $S_{мв} = (M - 1) S_{эм}$. Пусть время выборки слова из внешней памяти составляет $\tau_в$, а $\tau_о$ — время, затрачиваемое в среднем на обработку этого слова. Обозначим через k отношение времени выборки слова из внешней памяти ко времени обработки слова:

$$k = \frac{\tau_в}{\tau_о}. \tag{5.10}$$

Тогда отношение быстродействия УВС к быстродействию УВМ составит

$$R = \frac{V_н}{v_м} = (1 + k) M. \tag{5.11}$$

Надежность вычислительной системы. Под надежностью системы будем понимать способность сохранять заданные свойства при заданных условиях работы в течение определенного периода [9].

Количественной оценкой надежности является вероятность (P) того, что система обладает заданными свойствами при заданных условиях работы в течение определенного времени T.

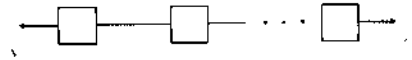
Работу системы можно характеризовать также вероятностью выхода ее из строя Q за определенный промежуток времени T. Так как ненадежность системы и ее надежность — события несовместимые, то

$$Q = 1 - P. \tag{5.12}$$

Пусть p — вероятность исправной работы элементарной машины, а $q = 1 - p$ — вероятность ее выхода из строя. В силу однородности структуры элементарных машин УВС, а также системы коммутации между ними можно считать, что выход из строя одной элементарной машины не сказывается на работе остальных ЭМ. В связи с этим будем предполагать, что выходы элементарных машин из строя — независимые события. Благодаря переменной структуре и возможности обнаружить неисправности ЭМ программным способом процесс обнаружения неисправной ЭМ и замены ее исправной может происходить автоматически и за пренебрежимо малое время.

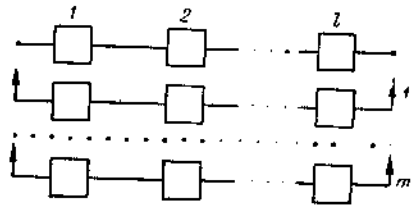
Пусть дана вычислительная система из M элементарных машин. Пусть для решения задачи требуется одновременная работа l машин. Определим надежность вычислительной системы с различными схемами резервирования (дублирования) [10], [11]:

а) надежность УВС без резервирования и ремонта.



$$P = p^l; \tag{5.13}$$

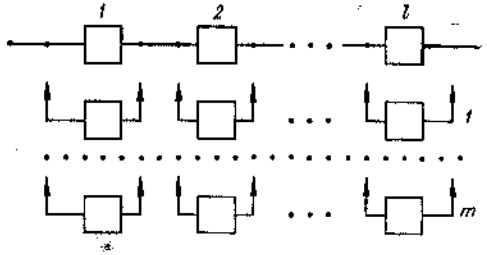
б) надежность УВС при резервировании всей системы



Пусть применяется $m = \frac{M}{l} - 1$ резервных цепей для всей системы в целом. Тогда надежность системы будет

$$P = 1 - (1 - p^l)^{m+1}; \qquad (5.14)$$

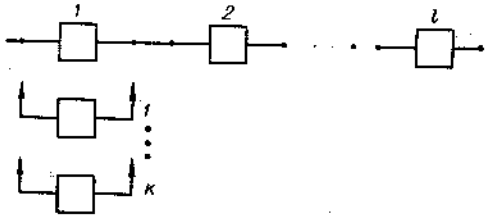
в) надежность УВС при резервировании системы по элементарным машинам



Пусть для каждой элементарной машины используется $m = \frac{M}{l} - 1$ резервных машин. Тогда надежность системы будет

$$P = [1 - (1 - p)^{m+1}]^l; \qquad (5.15)$$

г) надежность УВС при автоматическом ремонте неисправностей в системе



При выходе из строя одной из l машин неисправная ЭМ исключается из системы, а к оставшимся исправным машинам добавляется одна машина из резерва. Между ЭМ системы задача перераспределяется. По мере выхода из строя машин в системе из l машин процесс повторяется. Пусть $k = M - l$ машин используется в качестве резерва. Тогда надежность системы будет

$$P = \sum_{i=0}^{i=k} C_M^i p^i (1 - p)^{M-i}. \qquad (5.16)$$

Стоимость вычислительной системы. Стоимость УВС может быть выражена либо общим количеством эквивалентных элементов оборудования, взятых за единицу измерения, либо в других единицах измерения (например, в рублях). Будем сравнивать стоимость $C_с$ вычислительной системы со стоимостью $C_м$ универсальной вычислительной машины, которая эквивалентна вычислительной системе по своему быстродействию, объему памяти и надежности.

Отношение стоимости машины к стоимости системы будем называть *коэффициентом экономичности системы*:

$$k_с = \frac{C_м}{C_с}. \qquad (5.17)$$

Рассмотрим два случая:

1) все элементы одинаковы;

2) элементы памяти машины более просты по сравнению с остальными элементами.

Пусть все элементы, из которых выполнены система и машина, одинаковы, т. е. устройство памяти, арифметические устройства, устройства управления и т. д. составлены из одних и тех же элементов. Пусть $C_п$ — общее число элементов в памяти элементарной машины, а $C_л$ — число элементов во всей остальной части УВМ. Тогда коэффициент экономичности $k_с$ для системы из M машин будет

$$k_{с1} = \frac{C_м}{C_с} = \frac{MC_п + C_л}{MC_п + MC_л} = \frac{C_п + \frac{C_л}{M}}{C_п + C_л} = \frac{1 + \beta/M}{1 + \beta}, \qquad (5.18)$$

где $\beta = \frac{C_л}{C_п}$.

Но так как обычно $\beta \ll 1$, то в случае применения одинаковых элементов

$$k_{с1} = \frac{1}{1 + \beta}. \qquad (5.19)$$

Обозначим отношение стоимости элемента машины к стоимости элемента системы через α. Тогда

$$k_{c2} = \frac{(1 + \beta/M)\alpha}{1 + \beta} \approx \frac{\alpha}{1 + \beta}. \quad (5.20)$$

Пусть элементы собственно памяти более просты по сравнению с другими элементами, из которых реализуются система и машина. Обозначим через C_0 — общее число элементов, затрачиваемых на выборку слов из памяти ЭМ, запись в память, считывание из памяти. Элементы, затрачиваемые на реализацию собственно памяти ЭМ, не будем учитывать при определении k_c. Тогда коэффициент экономичности k_c для системы из M машин будет

$$k_{c1} = \frac{C_м}{C_c} = \frac{MC_0 + C_л}{MC_0 + MC_л} = \frac{C_0 + C_л/M}{C_0 + C_л} = \frac{1 + \gamma/M}{1 + \gamma}, \quad (5.21)$$

где $\gamma = \frac{C_л}{C_0}$.

Но так как практически $\gamma \approx 1$, то

$$k_{c1} \approx \frac{1}{1 + \gamma}, \quad (5.22)$$

или, с учетом различия стоимости элементов машины и элементов системы,

$$k_{c2} = \frac{(1 + \gamma/M)\alpha}{1 + \gamma} \approx \frac{\alpha}{1 + \gamma}. \quad (5.23)$$

Синхронизация вычислительной системы. Работа УВС может быть синхронизирована многими способами. В качестве примера приведем два: способ командной синхронизации и способ подпрограммной синхронизации.

При командной синхронизации система работает следующим образом. В каждой элементарной машине выполняется команда. После выполнения ее ЭМ работу прекращает до завершения операций во всех остальных ЭМ системы. Чтобы приступить к выполнению очередной команды, необходимо завершить выполнение предыдущей команды во всех машинах.

При подпрограммном способе синхронизации в каждой ЭМ независимо выполняется своя подпрограмма. После завершения подпрограмм во всех машинах по команде обобщенного условного перехода производится переход к новым подпрограммам.

Синтез вычислительной системы. Как уже описывалось выше, для построения универсальной вычислительной системы необходимо задать элементарную машину и размерность сетки. Тогда синтез вычислительной системы можно свести к двум этапам: к синтезу элементарной машины, необходимой для создания УВС с заданной размерностью сетки, и к синтезу универсальной УВС из элементарных машин для решения данной задачи или данного класса задач.

Вполне очевидно, что на первом этапе, в силу того, что элементарная машина фактически является конечным автоматом, применимы все методы синтеза, развиваемые в теории конечных автоматов.

На втором этапе для решения данной задачи или данного класса задач УВС из ЭМ может быть реализована программной настройкой, поэтому на втором этапе приемлемы методы синтеза, развиваемые в теории программирования.

5.4. Элементарные машины

Различают два типа элементарных машин: с фиксированной и с переменной структурой.

Структура первого типа аналогична структуре существующих ЭВМ и на ней останавливаться не будем.

В ЭМ с переменной структурой должны программно изменяться набор операций, разрядность кода и объем памяти.

Реализация переменного набора операций не должна встретить непреодолимых трудностей. Практически это можно выполнить на основе принципа микропрограммирования [12], согласно которому любая операция ЭВМ может быть составлена программно из некоторого фиксированного универсального набора микроопераций, что позволит в зависимости от требований задачи выбирать те или иные наборы операций.

Операции над кодами с переменным числом разрядов можно выполнить, например, применяя последовательный способ, как это делается, в частности, в вычислительной системе СОЛОМОН [6], где в коде операции указывается длина кода. При этом допускается изменение длины кода в широких пределах.

Объем памяти может быть изменен путем применения, например, модульной конструкции, допускающей подключение к памяти различного числа модулей, как в системе СДС-3600 [13].

Таким образом, есть возможность создать ЭМ с переменной структурой путем использования отдельных устройств, блоков, модулей и элементов с фиксированной структурой, объединенных

в единое целое с помощью программно управляемого коммутатора, осуществляющего те или иные связи между частями ЭМ. При таком подходе структура ЭМ аналогична структуре УВС, состоящей из переменного коммутатора и ЭМ с фиксированной структурой. По аналогии подобный подход к построению ЭМ будем называть макроструктурным.

На основании требований технологичности макроструктура элементарной машины должна быть по возможности однородной и простой.

Основное количество элементов ЭМ приходится на устройство памяти. Затраты оборудования на арифметическое устройство (АУ), на устройства управления (УУ) и коммутатор (К) составляют незначительную часть. Устройства памяти проще по структуре и однороднее, чем АУ, УУ и К, имеющие обычно сложную логическую структуру и состоящие из разнообразных логических схем.

Высокая логическая сложность устройств АУ, УУ и К и большое разнообразие схем предъявляют высокие требования как к процессу изготовления таких схем, так и к автоматической установке для производства ЭМ. Особенно большие трудности возникают при изготовлении связей между элементами, если эти связи неоднородны. Возникает необходимость строить такие схемы АУ, УУ и К, которые были бы по сложности одного порядка со схемами устройства памяти и, следовательно, не предъявляли бы высоких требований к технологии и средствам автоматизации изготовления.

Отметим, что устройство памяти из-за ограничений, накладываемых на рабочую частоту элементов машины, требования большого объема памяти и достаточно высокой скорости выполнения операций должно допускать произвольную выборку информации. Это усложняет структуру устройства памяти. Поэтому требование, чтобы сложность АУ, УУ и К была одного порядка с устройством памяти, не слишком жестко.

Рассмотрим возможные пути упрощения структуры ЭМ [14].

1. Построение ЭМ с ограниченным набором операций. Уменьшение числа различных операций в ЭМ упрощает структуру устройства управления. Выбор же самих операций влияет на простоту структуры арифметического устройства.

В работах [15, 16] указывается, что возможно построить универсальные машины с весьма ограниченным набором операций (например, сдвиг, сложение, условный переход или вычитание (сложение) и условный переход). Такие наборы при относительной простоте схем АУ и УУ позволяют реализовать любые алгоритмы. Следует отметить, что ограниченный набор операций не позволяет обеспечить достаточно высокую производительность.

Свободной от указанного недостатка могла бы быть элементарная машина, реализующая операцию выборки слова. Выполнение этой операции может быть представлено следующим образом.

Пусть требуется выполнить некоторую операцию R над словами a и b и получить результат $R(a, b) = c$. Тогда в памяти машины для каждой возможной пары слов a, b по соответствующему адресу $A(a, b)$ следует заранее запомнить результат операции c. В этом случае операция сводится к составлению адреса из слов, над которыми выполняется операция, и выборки из памяти результата по составленному адресу, т. е. в конечном счете — к реализации табличной операции (аналогично таблицам умножения, деления и т. п.). При достаточном объеме памяти можно выполнить любую операцию и получить максимально высокую производительность.

Операция выборки может быть также в принципе распространена и на n слов, т. е. $R(a_1, \ldots, a_n) = c$. Операция выборки не может привести к схемам более сложным, чем схемы устройства памяти, так как операции записи в память и считывания с памяти включают в себя операцию выборки. Следовательно, применение операции выборки позволяет строить схемы ЭМ со сложностью, не превышающей сложности схем устройства памяти.

Необходимый объем памяти для запоминания таблиц операций сильно растет с увеличением разрядности слов. Поэтому нерационально запоминать таблицы для выполнения операций над многоразрядными числами. Приходится использовать таблицы для выполнения операций над словами с меньшим числом разрядов, хотя это и приводит к некоторому снижению производительности ЭМ.

2. Построение ЭМ с ограниченным числом одновременно выбираемых или обрабатываемых разрядов слова. Требование одновременной выборки и обработки произвольного числа разрядов слова чрезмерно усложняет структуру ЭМ.

Здесь возможны два варианта:

1) выборка и обработка производится по одному двоичному разряду; при этом максимально упрощается структура ЭМ, но увеличивается время выполнения операций, а следовательно, уменьшается производительность машины;

2) выбирается максимальная длина слова, которая требуется для выполнения операций с заданной точностью; при этом, хотя структура машины относительно проста, при сохранении максимально возможной производительности затраты оборудования возрастут.

Первый вариант применяется в машинах последовательного действия [17]; второй — в машинах параллельного действия [18].

Промежуточный вариант представляет машина параллельно-последовательного действия [19]. В ней удается сочетать простоту структуры машин последовательного действия с высокой производительностью машин параллельного действия. Здесь допускается одновременная выборка и обработка некоторого ограниченного количества двоичных разрядов слова. Удобно это количество разрядов принять за элемент слова — «букву». Каждая буква далее неделима, что позволяет избежать операций сдвига, и адресуема, что удобно при обработке информации. Адресуемость букв позволяет выполнять операции над словами произвольной длины и эффективно использовать память [20].

Снижение производительности при использовании последовательного или параллельно-последовательного кода может быть в значительной мере компенсировано путем параллельного выполнения операций. В частности, по-видимому, очень эффективен метод, предложенный В. Шуманом [21]. Вместо обычного способа размещения данных он предлагает помещать в каждой ячейке памяти разряды разных слов, что позволяет выполнять операцию сразу над несколькими словами. Выигрыш во времени при этом получается тем больше, чем короче слова. Сложность структуры не больше, чем в машинах последовательного действия.

3. Построение ЭМ с ограниченным набором микроопераций. Простая структура элементарной машины может быть получена введением в устройства АУ и УУ некоторого набора микроопераций, из которых можно программным способом образовать более сложные операции [12]. Схемы АУ и УУ могут быть представлены в виде матриц, управляющих друг другом. В одной матрице могут быть записаны микрооперации, а в другой — последовательность выполнения и типы микроопераций, необходимые для реализации данной сложной операции. Структура ЭМ получается не сложнее структуры памяти, так как матрицы — это небольшие устройства постоянной памяти.

При достаточно большом наборе микроопераций можно образовывать широкий класс сложных операций и обеспечивать необходимую производительность. Если предусмотреть возможность записи информации в матрицы программным способом, то можно в некоторых границах менять набор микроопераций и последовательность их выполнения. При этом можно выбирать (в известных пределах) такие операции, которые обеспечивают решение данной задачи с наибольшей эффективностью.

4. Построение ЭМ из простейших машин, автоматов, модулей. Перечисленные пути построения ЭМ основаны на представлении машины в виде единого функционального целого. При этом сложность структуры различных устройств определялась сложностью

наибольшего по затратам оборудования устройства, в частности устройства памяти. Сейчас наметился и другой путь, при котором вся вычислительная машина состоит из некоторых простейших машин, соединенных по определенному закону. Такие соединения однородных машин получили название итеративных цепей [22].

Одним из возможных вариантов подобного вида служит построение ЭМ из машин с минимальным числом ячеек памяти с произвольной выборкой. В частности, возможно построение машины, подобной универсальной машине Тьюринга с двумя внутренними состояниями [23], но с конечным объемом внешней памяти. В таком случае в качестве внешней памяти может быть использована своего рода динамическая память с хранением информации в некоторой среде (например, в волноводах, линиях задержки и т. п.). Структура памяти при этом становится исключительно простой, так как нет необходимости в произвольной выборке слов, а значит, и в соединениях для связи с каждым элементом памяти. Простота остальной части машины обусловлена минимумом внутренних состояний. Если каждая из простейших машин будет предназначена для выполнения какой-нибудь операции или микрооперации, то можно получить достаточно высокую производительность при относительно простой структуре ЭМ.

Возможен также вариант построения ЭМ из некоторых простейших автоматов [24].

Большое развитие получило направление, основанное на построении итеративных цепей из однородных модулей [4,25]. В качестве модуля используется вычислительное устройство, приспособленное для выполнения некоторого ограниченного набора операций и имеющее определенное число ячеек памяти для хранения информации. Каждый модуль имеет каналы связи с соседними модулями для обмена информацией.

Возможны различные варианты построения элементарной машины из модулей.

При централизованном управлении модулями команды задаются из единого центрального устройства управления. В каждый момент выполняется одна и та же операция всеми модулями [25]. При таком подходе к построению ЭМ производительность ее может быть достаточно высокой, но эффективность использования модулей при решении многих задач будет низкой.

При децентрализованном управлении каждый модуль имеет регистр для хранения команды и может выполнять в каждый данный момент любую операцию из заданного набора независимо от операций, выполняемых другими модулями. Каждый модуль

служит вычислительным устройством с ограниченным объемом памяти [22]. При этом эффективность использования модулей и производительность ЭМ выше, чем при централизованном управлении.

Рассмотренные пути построения ЭМ показывают, что макроструктурный подход ЭМ позволяет получить однородную макроструктуру. В первых трех случаях сложность структуры ЭМ определяется сложностью структуры памяти, а в четвертом — сложностью структуры модуля.

Хотя при макроструктурном подходе и удается повысить производительность вычислений, однако трудно гарантировать достижение максимальной производительности для любого класса задач. Следовательно, можно сделать вывод, что макроструктурный подход не снимает необходимости знать классы решаемых задач для достижения высокой производительности. Для получения максимально простой структуры ЭМ требуется провести исследования по анализу вариантов построения структуры памяти или модулей. При этом может оказаться, что найденные варианты будут слишком сложны для автоматического процесса производства.

Второй подход к построению ЭМ с переменной структурой, названный микроструктурным [3], основывается на идее применения однородной вычислительной среды. Этот подход подробно рассматривается в гл. 6.

5.5. Система связи УВС

Система связи УВС состоит из каналов связи и коммутатора.

Каналы связи служат для обмена информацией в процессе решения задачи ЭМ друг с другом и с внешними объектами. Этот обмен может осуществляться либо в параллельном, либо в последовательном, либо смешанном параллельно-последовательном кодах. В параллельном коде для передачи m-разрядного кода требуется m каналов связи, в последовательном — один, в параллельно-последовательном — некоторое промежуточное число каналов между 1 и m. На скорость передачи кода налагается очевидное условие, чтобы время передачи кода не превышало времени выполнения операции над этим кодом в ЭМ. По-видимому, это условие не предъявляет невыполнимых требований к скорости передачи информации по каналу связи.

В УВС могут применяться каналы как с односторонней, так и с двухсторонней передачей информации.

В качестве каналов связи могут применяться как обычные линии связи, так и радиоканалы [26]. В первом случае затраты определяются линиями связи, во втором — приемно-передающей аппаратурой.

В дальнейшем будем исходить из возможности использования как линейных, так и радиочастотных каналов. Будем также считать, что внешние каналы, связывающие УВС с другими объектами, построены на тех же принципах, что и внутренние.

Создание коммутатора — одна из важнейших и наиболее сложных проблем, возникающих при реализации УВС с большим числом элементарных машин. Коммутатор должен реализовать матрицу соединений $[C_{ke}]$, которая построена в предположении, что ЭМ выполняют только одно- и двуместные операции и, следовательно, имеют один входной и один выходной полюсы, и в каждый данный момент времени к одному входному полюсу подключается не более одного выходного.

Непосредственная реализация такой матрицы приводит к построению коммутатора с $Q = M + R$ входами и Q выходами с числом элементов порядка Q^2 (M — число ЭМ, R — число внешних входов и выходов у УВС). При росте числа ЭМ эта величина может превысить общее число остальных элементов УВС. Вполне понятно, что коммутатор усложнится еще больше, если применять p-местные операции при $p \geqslant 3$.

Ограничение, наложенное на число выходных полюсов, подсоединяемых к данному входному полюсу, позволяет применять многоступенчатые коммутаторы. В связи с этим возникает необходимость выбрать оптимальную схему многоступенчатого коммутатора, реализующего матрицу соединений $[C_{ke}]$. Математическая формулировка этой задачи для p-местных операций сделана Ю. Г. Решетняком [27].

Определение. Пусть S — произвольное конечное множество, состоящее из $Q > 0$ элементов. Для простоты будем полагать, что S совпадает с отрезком $(1, 2, \ldots, Q)$ натурального ряда.

Коммутацией в множестве S будем называть некоторое многозначное отображение $f(x)$ его подмножества S' в множество S. Здесь $f(x)$ означает совокупность всех элементов, отвечающих элементу x. Будем говорить, что элемент x управляет множеством $f(x)$. Коммутация $f(x)$ называется простой, или однократной, если при $x \neq y$ множества $f(x)$ и $f(y)$ не имеют общих элементов ($f(x) \cap f(y) = \Lambda$).

Коммутация $f(x)$ называется p-кратной, если никакой элемент множества S не может принадлежать более чем p различным множествам $f(x)$, причем существует элемент, принадлежащий в точности p-множествам.

Пусть имеется вычислительная система, содержащая M элементарных машин m_i ($i = 1, 2, \ldots, M$). Предположим, что выделены некоторые машины $m_{i_1}, m_{i_2}, \ldots, m_{i_k}$, где $k \leqslant M$. Машине m_{i_S} сопоставлено некоторое множество машин A_S, $S = 1, 2, \ldots, k$.

Допустим, что для каждого S в машине m_{i_S} вырабатывается некоторый сигнал σ_S, который передается во все машины множества A_S. В этом случае будем говорить, что в вычислительной системе задана некоторая коммутация, а машину m_{i_S} будем называть машиной, управляющей множеством A_S. Задание коммутации в вычислительной системе равносильно заданию коммутации на множестве, образованном номерами машин ВС. При этом S есть множество номеров $i_1, i_2, \ldots, i_k$; $f(i_S)$ — множество номеров машин, входящих в A_S.

Простая коммутация соответствует случаю, когда в каждую машину системы может поступить сигнал не больше чем из одной машины. p-Кратная коммутация соответствует случаю, когда в каждую машину поступает не более p сигналов из остальных машин системы, причем есть машина, в которую поступает в точности p сигналов.

Если предположить, что все элементарные машины УВС выполняют только одно- или двуместные операции, то в каждую машину должно поступать не более одного кода, так как можно считать, что второй код уже хранится в данной машине, т. е. в каждый данный момент каждая ЭМ должна быть связана не более чем с одной другой ЭМ. При трехместных операциях число связей с другими ЭМ равно двум и т. п. Следовательно, математические требования к УВС заключаются в том, чтобы в ней имелась такая система связей, которая позволила бы легко осуществлять различные коммутации какой-либо определенной кратности.

Ю. Г. Решетняк подсчитал общее число коммутаций кратности не выше p:

$$\Gamma_{p,\,M} = \left(\sum_{k=0}^{p} c_M^k\right)^M. \tag{5.24}$$

Асимптотическая оценка для числа $\Gamma_{p,\,M}$ при фиксированном p и достаточно большом M имеет вид:

$$\Gamma_{p,\,M} = \frac{M^{pM}}{(p!)^M} e^{-\frac{p^2-3p}{2}}\left[1 + O\left(\frac{1}{M}\right)\right]. \tag{5.25}$$

Остановимся теперь на понятии информационного графа вычислительной системы. Пусть каналы связи, посредством которых соединяются элементарные машины УВС, будут ориентированными, т. е. допускающими передачу информации в одном определенном направлении. Совокупность всех каналов связи образует некоторый ориентированный граф, вершинами которого служат ЭМ, составляющие ВС, а дугами — каналы связи. Такой граф будем называть *информационным графом* вычислительной системы.

Для произвольной машины m_i обозначим через K_i^+ совокупность всех каналов связи, подключенных к машине m_i своими выходами. По этим каналам информация поступает в машину m_i. Через K_i^- обозначим совокупность всех каналов связи, подключенных к m_i своими входами. По этим каналам связи передается информация из машины m_i. Число каналов связи, принадлежащих множествам K_i^+ и K_i^-, обозначим через k_i^+ и k_i^- соответственно. Нетрудно видеть, что $\sum_{i=1}^{M} k_i^+ = \sum_{i=1}^{M} k_i^- = k$, где k общее число каналов связи. Информация из машины m_i вычислительной системы в машину m_j может передаваться следующим образом. Найдем цепочку каналов связи $l_1, l_2, \ldots, l_k$ такую, что каналы l_r и l_{r+1} ($r = 1, 2, \ldots, k-1$) присоединены к одной ЭМ, причем один из них присоединен к этой ЭМ своим входом, а другой — своим выходом. При этом вход канала l_1 подключен к ЭМ m_i, а выход канала l_k — к ЭМ m_j. Существование такой цепочки каналов связи — необходимое условие для того, чтобы какая-либо информация могла быть передана из ЭМ m_i в ЭМ m_j. Процесс передачи можно выполнять как за $k-1$ тактов, так и за один такт.

Многотактная передача. В первом такте сигнал из ЭМ m_i передается в m_{i_1}, к которой подключен выход канала l_1. Во втором такте этот сигнал подается из m_{i_1} на вход канала l_2, по которому сигнал поступает в следующую ЭМ m_{i_2} и т. д. Потактную передачу информации можно использовать для обработки передаваемого сигнала, чтобы определить по адресной части сигнала дальнейшее направление передачи информации. Таким образом, цепочки каналов связи находятся непосредственно в процессе передачи информации.

Однотактная передача. При этом способе передачи выход канала l_S непосредственно соединен со входом канала l_{S+1}, передаваемый сигнал проходит по цепочке каналов связи в один такт. Вся цепочка каналов связи работает как единый канал. Путь, по которому передается сигнал, в данном случае определяется заранее.

Предположим, что в вычислительной системе задана некоторая коммутация $f(x)$. Пусть $f(x)$ определена для $x = i_1, i_2, \ldots, i_k$

Положим $f(i_S) = A_S$. Допустим, что из m_{i_S} передается сигнал σ_S. Чтобы этот сигнал был принят машинами множества A_S, нужно построить линию связи l_S, исходящую из ЭМ m_{i_S} и соединяющую ее со всеми ЭМ множества A_S. Построим соответствующие линии связи l_S. Передача сигналов из управляющих ЭМ m_{i_S} будет возможна по этим линиям l_S в том и только в том случае если никакой канал связи не входит одновременно в две линии l_{S_1} и l_{S_2}. Иначе будет происходить смешение различных сигналов и передача из машин станет невозможна.

Для реализации передачи сигналов, соответствующих произвольной коммутации, посредством информационного графа мы должны иметь в каждой ЭМ m_i устройство, которое обеспечивает возможность соединения любого выхода каналов группы K_i^+ с любым входом группы K_i^-. Сложность такого коммутационного устройства зависит от значений k_i^+ и k_i^-. При достаточно больших значениях этих чисел коммутационное устройство может быть очень сложным.

Таким образом, приходим к следующему требованию, которому должен удовлетворять оптимальный информационный граф вычислительной системы: для каждого i числа k_i^+ и k_i^- должны быть не слишком велики. Для более точной формулировки положим $k_i = k_i^+ + k_i^-$ и пусть $k(M) = \max k_i$. Допустим, что информационный граф вычислительной системы $i = 1, 2, \ldots, M$ устроен так, что в нем может быть реализована передача сигналов, отвечающая любой коммутации кратности не выше p. Будем говорить, что граф оптимален, если величина k (p) в нем достигает наименьшего из возможных значений, причем ни одна дуга информационного графа не может быть устранена без того, чтобы граф не потерял того свойства, что в нем может быть реализована любая коммутация заданной кратности.

Пусть Q — произвольный ориентированный граф. Простым путем в графе Q называется всякая последовательность дуг графа $\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_p$, такая, что никакая дуга не встречается в этой последовательности дважды и начало дуги $\bar{a}_i$ совпадает с концом дуги $\bar{a}_{i-1}$ для всех $i = 1, 2, \ldots, p$.

Пусть информационный граф ВС устроен так, что в нем может быть реализована любая простая коммутация. Тогда из него легко построить информационный граф, в котором реализуется любая p-кратная коммутация. Для этого достаточно каждую дугу графа Q заменить на p дуг той же ориентации и с теми же концами.

Построенный граф, хотя и не будет оптимальным, из соотношения $\frac{\log_2 \Gamma_{p,M}}{\log_2 \Gamma_{1,M}} \to p$ при $M \to \infty$ следует, что данный способ соединения асимптотически близок к оптимальному. В качестве тривиального примера графа, в котором может быть реализована любая простая коммутация, можно привести так называемый ориентированный полный граф. Это граф Q, в котором любые две вершины соединены двумя (и только двумя) дугами противоположного направления. В ориентированном полном графе может быть реализована любая (M — 1)-кратная коммутация, где M — число вершин графа.

Итак, соединение машин по способу полного ориентированного графа с точки зрения функционирования вычислительной системы наиболее выгодно. В тех случаях, когда M невелико, по-видимому, машины УВС целесообразно соединять по способу полного ориентированного графа.

Для тех случаев, когда M велико, Ю. Г. Решетняк предлагает применять P_n-граф.

Рассмотрим n-мерный куб, и пусть P_n означает ориентированный граф, который получается, если в графе из одномерных ребер куба каждое ребро заменить двумя, имеющими взаимно противоположные направления. Общее число вершин графа P_n равно 2^n. Количество дуг k^-, исходящих из каждой вершины, равно числу дуг, входящих в нее, и равно n, $k = k^+ = k^- = n$. Общее число дуг графа P_n равно $n \cdot 2^n$. Число k равно $\log_2 M$, где $M = 2^n$ — число вершин P_n.

В связи с этим высказывается следующая гипотеза: *в графе P_n может быть реализована любая коммутация кратности один и притом с использованием только простых путей.* В общем виде это утверждение не доказано. Для случая $n = 2$ удается проверить его справедливость перебором всех возможных коммутаций. Рассмотрение примеров реализации коммутации показывает, что наибольшие трудности встречаются, как правило, при реализации простых коммутаций вида $f(x)$, где $f(x)$ — взаимооднозначное отображение. Коммутации такого рода будем называть перестановками. При этом в каждой вершине должен заканчиваться, по крайней мере, один путь (из числа реализующих коммутацию) и, по крайней мере, один путь должен исходить из нее. При реализации перестановок повышается «расход» дуг. Максимальный расход дуг достигается при перестановке, для которой вершины x и $f(x)$ максимально удалены друг от друга. Длина пути, соединяющего вершины x и $f(x)$, должна быть в этом случае не меньше n. Так как число всех путей равно 2^n, то общее количество дуг, используемых при реализации данной перестановки, не

меньше $n \cdot 2^n$, и поскольку $n \cdot 2^n$ есть общее число ребер графа P_n, оно должно быть равно $n \cdot 2^n$. При реализации данной коммутации используются все дуги графа.

Хотя сделанное выше рассмотрение не учитывает связей с внешними полюсами УВС, оно не теряет своей общности, так как можно считать, что введение внешних полюсов эквивалентно соответствующему увеличению числа ЭМ.

Таким образом, оптимальной схемой коммутатора для малого числа ЭМ будет схема типа полного графа, а для большого числа ЭМ — схема, построенная по принципу P_n -графа.

Граничное число ЭМ, при котором становится выгодно применять схему P_n-графа, зависит от типа кода (параллельного, последовательного или смешанного) и технической реализации. Например, при использовании радиочастотного принципа, когда каждая ЭМ имеет передатчик с фиксированной частотой и приемник, настраивающийся на прием Q различных частот, схема полного графа может оказаться выгоднее при большем числе ЭМ, чем при соединении ЭМ обычными линиями связи [26].

При числе ЭМ 10^3—10^4, которым можно ограничиться для достижения производительности УВС 10^9 *опер/сек*, затраты на коммутатор, построенный по схеме полного графа, составят небольшую долю общих затрат на УВС. Если ЭМ увеличить до количества, которое будет больше указанного числа, это утверждение может оказаться несправедливым, и потребуется применять структурно более сложную схему P_n-графа. При этом предполагается, что число непосредственных связей, реализуемых P_n-графом, $n = \log_2 Q$. Это соответствует тому, что в каждый момент времени может быть реализовано соединение любых Q пар полюсов, т. е. предполагается соединение между собой любых сколь угодно удаленных друг от друга элементарных машин. Из анализа задач (гл. 8) следует, что практически для всех известных важных классов задач можно ограничиться значениями $n \ll \ll \log_2 Q$. Это позволяет реализовать простые схемы коммутаторов, построенных по принципу P_n-графов. В частности, можно реализовать схемы, в которых каждая ЭМ соединена только с 2, 4 либо 6 своими соседями, что соответствует расположению ЭМ в одно-, дву- и трехмерной системах координат.

5.6. Основные типы УВС

В связи с изучением УВС возникает необходимость в систематизации и классификации различных типов УВС по конструктивным признакам. Сейчас, когда только начинается систематическое

исследование вычислительных систем, дать достаточно полную классификацию УВС трудно, тем более, что подобной классификации не существует и для ЭВМ. Поэтому ограничимся некоторым, не претендующим на полноту вариантом классификации, в основу которого положены следующие признаки: характер пространственного размещения ЭМ (система координат и расстояние между ЭМ); тип коммутатора; организация схем, управляющих изменением структуры ЭМ и коммутатора (структура схемы настройки, способ выборки настраиваемых ЭМ); способ управления ходом вычислений; характер изменения структуры; система ввода и вывода данных и т. п.

По характеру пространственного размещения ЭМ будем различать *одно-*, *дву-* и *многомерные* УВС, которые соответствуют представлению ЭМ в виде точек на линии, поверхности и n-мерном пространстве. В первом случае для задания ЭМ достаточно одной координаты. Каждая ЭМ будет иметь двух соседей. Во втором случае положение ЭМ задается двумя координатами и каждая ЭМ имеет четырех соседей. В n-мерном случае положение ЭМ задается n координатами и каждая ЭМ имеет $2n$ соседей.

По расстоянию между ЭМ будем различать *сосредоточенные и распределенные* УВС. У сосредоточенных УВС время распространения сигналов между двумя наиболее удаленными друг от друга ЭМ существенно меньше времени выполнения операции элементарной машиной. В этом случае запаздывание сигналов в каналах связи практически не влияет на производительность УВС. У распределенных УВС время распространения сигналов между ЭМ сравнимо либо превышает время выполнения операции ЭМ. В этом случае запаздывание сигналов в каналах связи может существенно сказаться на производительности УВС, если не предприняты специальные меры.

По типу коммутатора будем различать УВС с коммутаторами, реализующими *соединения* между ЭМ *по типу полного графа*, когда каждая ЭМ соединена каналом связи с каждой ЭМ, и *по типу P_n-графа*, когда каждая ЭМ соединена только со своими соседями в n-мерной системе координат.

В зависимости от структуры схемы настройки различают УВС *с фиксированной и переменной системами настройки*. При фиксированной системе настройки схема, управляющая настройкой, не может программно изменять своей структуры. При переменной системе можно программно перестраивать структуру схем настройки и изменять пути передачи настроечной информации, что позволяет создавать УВС с высокой надежностью.

В зависимости от способа выборки настраиваемых ЭМ будем различать УВС *с произвольной системой выборки настраиваемых*

ЭМ (произвольная настройка) и системы *с упорядоченной выборкой* (упорядоченная настройка). При произвольной настройке в каждый данный момент настраивается одна ЭМ и ее соединения с другими ЭМ. При упорядоченной настройке ЭМ настраиваются в строго заданной последовательности и обычно сразу для всех ЭМ УВС или групп ЭМ.

По способу управления ходом вычислений будем различать УВС *с иерархической структурой управления* и *с однородной структурой*. При иерархической структуре управления программа работы УВС находится либо в одной ЭМ, которая посылает команды управления во все остальные ЭМ, либо в определенной группе ЭМ. При этом устанавливаются иерархическая подчиненность и правила приоритета команд управления, поступающих от различных ЭМ. При однородной структуре управления все ЭМ равнозначны, программа решения задачи распределена между всеми ЭМ и ход вычислений в зависимости от полученных результатов изменяется с помощью операции обобщенного условного перехода, учитывающей состояния всех ЭМ.

По характеру изменения структуры будем различать УВС *с частично изменяемой структурой*, когда изменяется только структура коммутатора, и *с полностью изменяемой структурой*, когда изменяется как структура коммутатора, так и структура ЭМ.

По характеру обмена информацией между ЭМ внутри УВС и между УВС и внешними объектами будем различать УВС *с последовательной*, *параллельной* и *последовательно-параллельной передачей* слов. При параллельной передаче все разряды слова передаются за один такт, при последовательной — слово передается разряд за разрядом от одной ЭМ к другой, при последовательно-параллельной — частями слов. Для каждой части слова одновременно передаются все разряды за один такт.

Большинство приведенных признаков независимы и могут сочетаться друг с другом в произвольных комбинациях. Хотя этими признаками классификация далеко не исчерпывается, их комбинации охватывают, по-видимому, основные типы возможных вариантов УВС.

5.7. Система настройки УВС

Как уже говорилось (см. 5.6), в зависимости от структуры схемы настройки различают УВС с фиксированной и переменной системами настройки, а по способу выборки настраиваемых ЭМ — с упорядоченной и произвольной выборкой. Рассмотрим более

подробно различные варианты системы настройки УВС. Настройка УВС с полностью изменяемой структурой по принципу организации мало отличается от настройки УВС с частично изменяемой структурой. Поэтому для простоты ограничимся рассмотрением УВС с частично изменяемой структурой. Будем также предполагать, что управлять настройкой можно на каждом этапе решения задачи любой ЭМ. Возможно также управление настройкой одновременно несколькими ЭМ, если у УВС более одного иерархического уровня управления работой ЭМ.

Пусть в вычислительной системе из M элементарных машин, каждая ЭМ имеет один входной канал для приема информации от других ЭМ и один выходной канал для передачи информации в другие ЭМ. Пусть в каждый данный момент входной канал может быть соединен только с одним выходом любой ЭМ вычислительной системы. Выход же данной ЭМ может соединяться с любым числом входных каналов других ЭМ. Таким образом, в данный момент в элементарную машину вводится информация только от одной какой-либо ЭМ вычислительной системы и выводится из данной ЭМ в любое число ЭМ. Другими словами, относительно входов ЭМ в вычислительной системе реализуется однократная (простая) коммутация. Коммутацию в вычислительной системе будем представлять в виде квадратной матрицы

$$\begin{bmatrix} C_{11}(t) & C_{12}(t) \dots C_{1l}(t) \dots C_{1M}(t) \\ C_{21}(t) & C_{22}(t) \dots C_{2l}(t) \dots C_{2M}(t) \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ C_{k1}(t) & C_{k2}(t) \dots C_{kl}(t) \dots C_{kM}(t) \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ C_{M1}(t) C_{M2}(t) \dots C_{Ml}(t) \dots C_{MM}(t) \end{bmatrix} \tag{5.26}$$

элементы которой $C_{kl}(t)$ принимают значение 0 или 1. Равенство $C_{kl}(t) = 1$ означает, что входной канал ЭМ m_k присоединен в момент t к выходу ЭМ m_l. Время, как и ранее, считается дискретным и принимает значения $t = 0, 1, 2, \dots$ Матрица $[C_{kl}(t)]$ обладает следующими свойствами:

$$\bigvee_{\substack{l,j=1 \\ l \neq j}}^{M} C_{kl} \& C_{kj} = 0; \tag{5.27}$$

$$\underset{k=l}{C_{kl}} = 0. \tag{5.28}$$

На определенных тактах вычислений, когда требуется изменить систему соединений между ЭМ, выполняется операция настройки

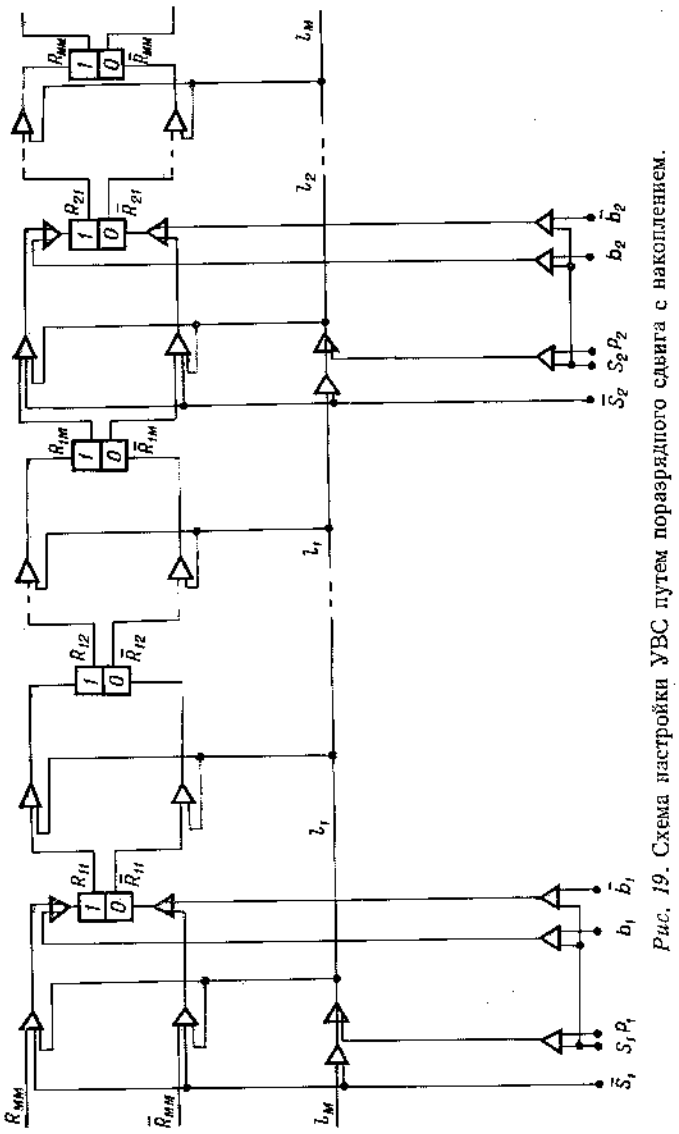


Рис. 19. Схема настройки УВС путем поразрядного сдвига с накоплением.

изменяющая значения элементов матрицы $C_{kl}(t)$. Будем понимать под H ($[C_{kl}(t)]$) операцию настройки, изменяющую в момент t значения элементов матрицы $[C_{kl}(t)]$.

Рассмотрим сперва примеры УВС с фиксированной системой настройки.

Настройка УВС путем поразрядного сдвига с накоплением. Поставим в соответствие каждому элементу матрицы $C_{kl}(t)$ двоичный элемент памяти R_{kl} и построим логическую сеть (рис. 19), в которой элементы R_{kl} принимают сигналы от элемента $R_{k,l-1}$ ($l \neq 1$) либо от элемента $R_{k-1,M}$ ($l=1$), кроме элемента R_{11}, который принимает сигналы от элемента R_{MM}. Элементам R_{u1}, $R_{u2}, \ldots, R_{uM}$ ($u = 1, \ldots, M$) ставится в соответствие ЭМ m_u, выходы которой S_u, $\overline{S_u}$, b_u, $\overline{b_u}$ и P_u поданы на элемент R_{u1}. Данная схема предусматривает, что в каждый данный момент любая из ЭМ может быть управляющей. Для этого она должна содержать в своей памяти все M^2 элементов матрицы $[C_{kl}(t)]$, которые в ходе выполнения операции настройки надо ввести в соответствующие элементы памяти R_{kl}. Выполняется это следующим образом. Пусть какая-либо из машин, например m_u, будет управляющей, что соответствует значениям

$$S_u = 1,\ S_k = 0\ (k = 1,\ 2, \ldots,\ u-1,\ u+1, \ldots,\ M). \qquad (5.29)$$

В этом случае для машины m_u преграждается поступление сигналов из предыдущей машины m_{u-1} и открываются пути для подачи через выходы b_u, $\overline{b_u}$ соответствующих значений элементов матрицы $C_{kl}(t)$ и тактовых импульсов P_u. На первом такте в R_{u1} поступает элемент матрицы $C_{u-1,M}$, на втором — $C_{u-1,M-1}$, на третьем — $C_{u-1,M-2}$ и т. д. и, наконец, на такте M^2 — элемент C_{u1}. Одновременно на каждом такте все R_{kl} будут передавать свое содержимое соседу справа. В результате за M^2 тактов все элементы C_{kl} будут введены в соответствующие памяти R_{kl}.

Данная логическая сеть может быть описана с помощью следующей системы уравнений:

$$R_{kl}(t+1) = R_{k,l-1}(t)\ l_k(t+1) \qquad (l \neq 1);$$

$$\overline{R_{kl}}(t+1) = \overline{R_{k,l-1}}(t)\ l_k(t+1) \qquad (l \neq 1);$$

$$R_{k1}(t+1) = R_{k-1,M}(t)\ \overline{S_k}(t+1) \vee b_k(t+1)\ S_k(t+1) (k \neq 1);$$

$$\overline{R_{k_1}}(t+1) = \overline{R_{k-1,M}}(t)\,\overline{S_k}(t+1) \vee \overline{b_k}(t+1)\ S_k(t+1) \qquad (k \neq 1);$$

$$R_{1_1}(t+1) = R_{MM}(t)\ S_1(t+1) \vee b_1(t+1)\ S_1(t+1);$$

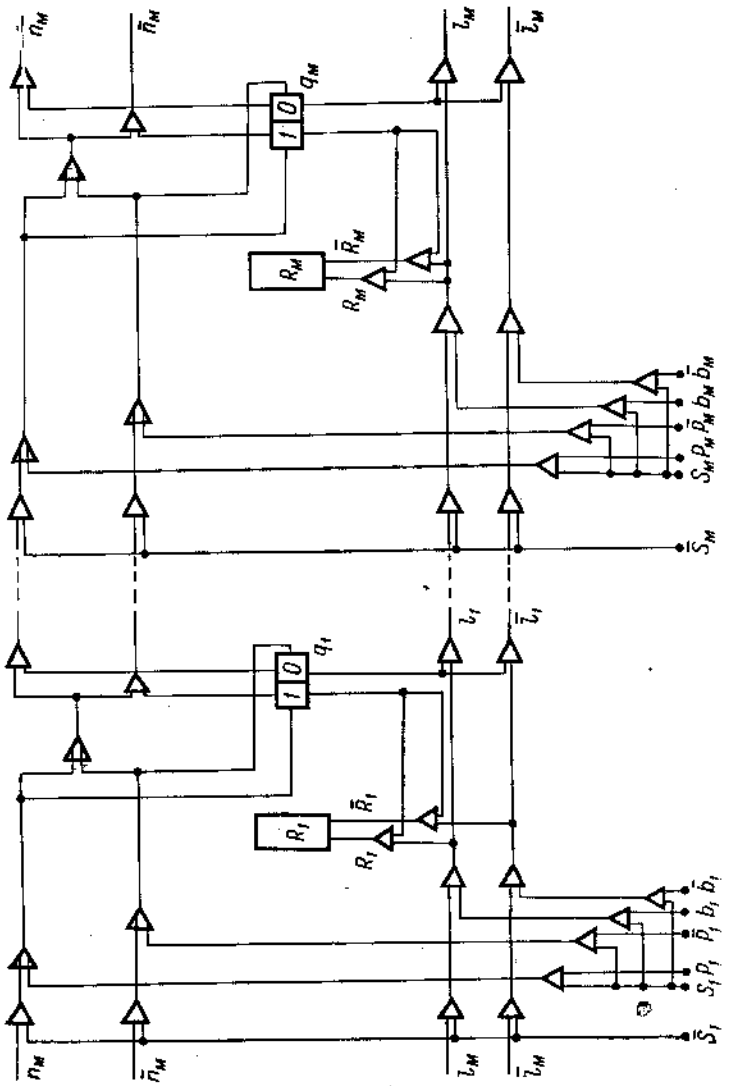


Рис. 20. Схема пошаговой настройки УВС.

$$\overline{R}_{11}(t+1) = \overline{R}_{MM}(t)\overline{S}_1(t+1) \vee \overline{b}_1(t+1)S_1(t+1);$$
$$l_k(t) = l_{k-1}(t)\,\overline{S}_k \vee P_k S_k;$$
$$l_1(t) = l_M(t)\,\overline{S}_1 \vee P_1 S_1. \qquad (5.30)$$

Время настройки может быть уменьшено, если настройкой управляют несколько ЭМ. В частности, если в каждой ЭМ m_k содержится k-я строка матрицы $[C_{kl}(t)]$, то все M машин могут быть сделаны управляющими, для чего надо положить все $S_k(t) = 1$ $(k = 1, 2, \ldots, M)$.

Тогда настройка может быть осуществлена всего за M тактов.

Пошаговая настройка использует свойство матрицы $[C_{kl}(t)]$ (5.27), согласно которому в каждой строке матрицы не может быть более одного ненулевого элемента. Это позволяет определить матрицу $[C_{kl}(t)]$ не более чем M числами, каждое из которых указывает адрес соответствующего ненулевого элемента матрицы. Обозначим их соответственно $a_1, a_2, \ldots, a_M$. Каждое из a_k будет, очевидно, иметь $m = \log_2 M$ двоичных разрядов. Пусть в каждой ЭМ m_k имеется память R_k объемом m двоичных разрядов для хранения соответствующего a_k. Тогда выполнение операции настройки сведется к засылке в памяти R_k соответствующих a_k, что может быть реализовано, например, с помощью схемы (рис. 20), соответствующей следующей системе уравнений:

$$R_k(t) = q_k(t)\,[l_{k-1}(t)\overline{S}_k(t) \vee b_k(t)\,S_k(t)];$$
$$\overline{R}_k(t) = q_k(t)\,[\overline{l}_{k-1}(t)\overline{S}_k(t) \vee \overline{b}_k(t)\,S_k(t)];$$
$$l_k(t) = \overline{q}_k(t)\,[l_{k-1}(t)\overline{S}_k(t) \vee b_k(t)\,S_k(t)];$$
$$\overline{l}_k(t) = \overline{q}_k(t)\,[\overline{l}_{k-1}(t)\overline{S}_k(t) \vee \overline{b}_k(t)\,S_k(t)]; \qquad (5.31)$$
$$q_k(t) = n_{k-1}(t)\overline{S}_k(t) \vee P_k(t)\,S_k(t) \vee q_k(t);$$
$$\overline{q}_k(t) = \overline{n}_{k-1}(t)\overline{S}_k(t) \vee \overline{P}_k(t)\,S_k(t) \vee \overline{q}_k(t);$$

$$n_k(t) = q_k(t)\,[n_{k-1}(t)\overline{S}_k(t) \vee P_k(t)\,S_k(t) \vee \overline{n}_{k-1}(t)\overline{S}_k(t) \vee \overline{P}_k(t) \times S_k(t)];$$

$$\overline{n}_k(t) = \overline{q}_k(t)\,[n_{k-1}(t)\overline{S}_k(t) \vee P_k(t)\,S_k(t) \vee \overline{n}_{k-1}(t)\overline{S}_k(t) \vee \overline{P}_k(t) \times S_k(t)].$$

Как и в предыдущем случае (5.30), все машины при выполнении операции настройки оказываются расположенными вдоль зам-

кнутой линии. В каждой машине m_k имеется триггер q_k, который находится в состоянии $q_k = 1$ в той ЭМ, которая в данный момент настраивается. Операция настройки выполняется следующим образом.

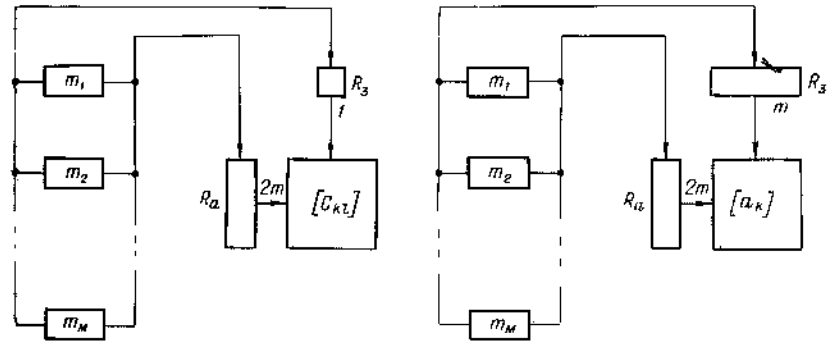
Пусть машина m_u будет управляющей и содержит в своей памяти значения всех $a_1, a_2, \ldots, a_M$. И пусть, как и ранее, выполняется условие (5.29), соблюдение которого означает, что шины схемы настройки управляющей машины оказываются запертыми для внешних сигналов и подключены к источникам настроенной информации через входы b_u, $\bar{b}_u$ и тактовых толкающих импульсов через входы P_u, $\bar{P}_u$. Тактовые толкающие импульсы подаются через интервалы времени mt, где t — рабочий такт, необходимый для ввода одного двоичного разряда кода. Первый толкающий импульс подается на вход P_u, все $M - 1$ последующих импульсов — на вход $\bar{P}_u$. Первый толкающий импульс устанавливает значение триггера $q_u = 1$, что отпирает входы памяти R_u, и в нее за m последовательных тактов вводится код a_u. Второй толкающий импульс устанавливает значение триггера следующей машины $q_{u+1} = 1$ и гасит триггер q_u. Тогда очередной код a_{u+1} воспримется только машиной m_{u+1}. Третий толкающий импульс возбуждает триггер q_{u+2} и т. д. Через M шагов по m тактов каждый все машины будут настроены.

С помощью данной схемы можно изменять настройку только у некоторых произвольно выбранных ЭМ. Можно также одновременно настраивать несколько ЭМ с одинаковыми значениями a_k. Достигается это подбором соответствующих последовательностей толкающих импульсов. Например, если ЭМ m_k не нужно перенастраивать, то когда подходит ее очередь, подаются подряд два импульса толкания. Если, например, $a_{\frac{M}{2}+1} = a_1$, $a_{\frac{M}{2}+2} = a_2, \ldots$, $a_M = a_{M/2}$, то сперва подается серия толкающих импульсов $P(t = 1) = 1$, $P(t = 2) = 0$, $P(t = 3) = 0, \ldots$, $P(t = M/2) = 0$, $P(t = M/2 + 1) = 1$, после чего начинается ввод кодов $a_1, a_2, \ldots, a_{M/2}$ обычным порядком.

Данная схема, как и предыдущая, допускает одновременную настройку с помощью нескольких управляющих машин.

Координатная настройка. В двух предыдущих способах порядок настройки ЭМ был задан жестко. Рассмотрим теперь примеры схем настройки с произвольным порядком выборки настраиваемых ЭМ. Схемы, допускающие произвольный порядок записи и считывания информации, широко применяются в оперативных памятях ЭВМ. В этих схемах выборка необходимой ячейки

памяти достигается путем подачи ее адреса в виде определенной совокупности сигналов в соответствующие координатные шины. Аналогичным образом может быть построена и схема настройки. Матрице $[C_{kl}(t)]$ ставится в соответствие матричная память объемом M^2 двоичных ячеек. В этом случае при двумерной системе координат для задания адреса любой ячейки потребуется регистр R_a объемом $2m$ разрядов, где $m = \log_2 M$ (рис. 21, *а*). Если же



а *б*

Рис. 21. Схема координатной настройки УВС.

а — при запоминании всех элементов матрицы; *б* — при запоминании только ненулевых элементов.

запоминать только адреса ненулевых элементов, то можно обойтись памятью объемом Mm двоичных ячеек и для задания адреса достаточно будет регистра объемом m двоичных разрядов (рис. 21, *б*). Однако в этом случае объем регистра записи R_3 увеличится до m разрядов вместо одного в первом случае.

При координатном способе настройкой управляет одна из ЭМ, например m_u. Операция настройки выполняется следующим образом. Из ЭМ m_u в регистр R_a подается адрес настраиваемого элемента, а в регистр R_3 — соответствующий код настройки. В каждый момент настраивается только один элемент матрицы. Время настройки будет при этом составлять M^2, когда запоминаются C_{kl}, либо Mm тактов, когда запоминаются a_k.

Адресная настройка. Код настройки, сопровождаемый адресом, посылается из управляющей ЭМ в общую систему связи, соединяющую все ЭМ системы. Каждая ЭМ анализирует адрес, сопровождающий код настройки, и при совпадении адреса с ее собственным воспринимает код настройки. В результате изменяются соответствующие коммутации в блоке K (рис. 22). Вход и выход каждой элементарной машины подключен к общей шине O.

К коммутатору K каждой ЭМ подходит M входных каналов. В каждый данный момент к ЭМ с помощью коммутатора K *может быть подключено не более одного входного канала.*

Схема коммутатора состоит из регистра R, управляющего коммутациями и собственно коммутатора, в котором происходит подключение к входу ЭМ только одного входного канала, опре-

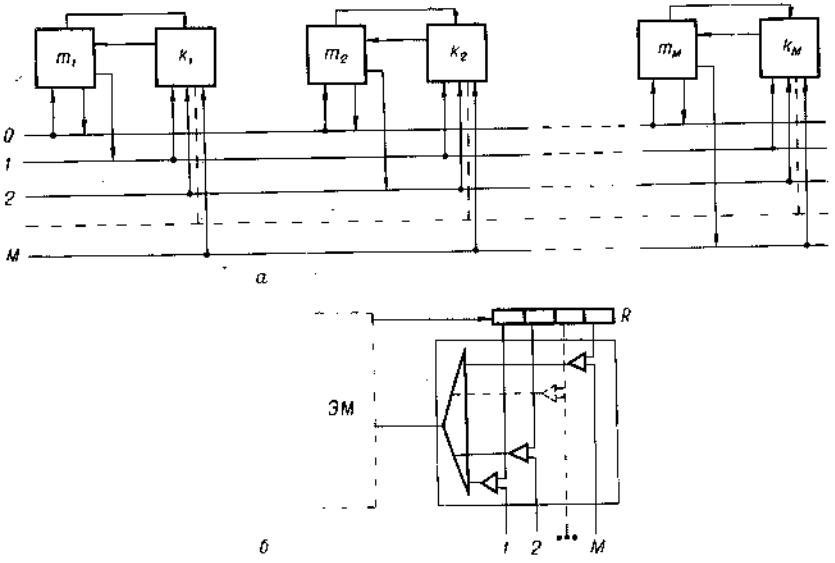


Рис. 22. Схема адресной настройки УВС.
а — блок-схема УВС; *б* — схема коммутационного устройства.

деляемого содержимым регистра R (рис. 22, *б*). Информация в регистр R подается из ЭМ.

При адресном способе настройка выполняется следующим образом. Пусть в некоторой машине m_u хранится информация, определяющая коммутацию вычислительной системы. Процесс настройки разделяется на две фазы. I. Из ЭМ m_u по каналу 0 информация настройки каждой ЭМ передается в соответствующую ЭМ. Это можно выполнить с помощью специальных подпрограмм, хранящихся в каждой ЭМ. По признаку принадлежности информации настройки данной ЭМ информация выделяется из общего потока и помещается в памяти ЭМ для управления коммутацией по второй фазе. II. Информация настройки по специальной команде подается на регистр коммутационного устройства. В результате соответствующий входной канал подключается к данной ЭМ.

При адресном способе настройки можно одновременно настраивать все ЭМ, если у них нужно установить одинаковую систему соединений. Возможна также одновременная настройка с помощью нескольких управляющих ЭМ. Для этого можно, воспользовавшись ранее установленной системой коммутации, сперва передать в различные ЭМ соответствующую настроечную информацию, а затем дать сигнал перехода к новой системе коммутации, т. е. сперва для всех ЭМ выполнить первую фазу настройки, а затем вторую.

Адресный способ отличается большой гибкостью и может быть реализован за сравнительно небольшое время. В то же время он не требует больших дополнительных схем. Способы настройки с произвольной выборкой особенно выгодны, если изменяется структура только некоторых ЭМ.

Системы настройки с переменной структурой. Все четыре рассмотренных примера имеют фиксированную систему настройки, что делает их недостаточно надежными. Выход из строя одной или нескольких ЭМ или локальные неисправности каналов связи могут существенно снизить производительность УВС и сделать ее непригодной для дальнейшего использования до устранения неисправностей. При большом числе ЭМ время на ремонт может уменьшить полезное время работы УВС до практически неприемлемой величины. Для УВС, изготовленных на основе микроминиатюризации как единое целое, ремонт сильно затруднен и, возможно, от него придется вообще отказаться. Все это указывает на необходимость принять меры к повышению надежности УВС. Безусловно, прежде всего необходимо требовать увеличения надежности ЭМ и каналов связи. Однако трудно надеяться, что одни эти меры смогут обеспечить надежную работу таких многомашинных систем, как УВС.

Поэтому необходимо предпринять шаги, которые обеспечили бы сохранение производительности УВС на достаточно высоком уровне при выходе из строя отдельных ЭМ и каналов связи. В качестве одной из таких мер можно указать на переход к переменной структуре схем настройки, которая обеспечивает локализацию неисправностей в небольшой области и сводит ее воздействие к исключению из УВС некоторого числа ЭМ при сохранении всех остальных свойств УВС.

Рассмотрим в качестве примера двумерную УВС. Пусть каждая ЭМ с координатами (i, j) может получать коды настройки от любой, но в каждый данный момент только одной соседней ЭМ, и передавать коды любой соседней ЭМ с координатами $(i, j - 1)$, $(i - 1, j)$, $(i, j + 1)$ или $(i + 1, j)$. В дальнейшем для простоты будем данную ЭМ и соответствующие ей параметры обозна-

чать индексом 0, а ее соседей—индексами 1, 2, 3, 4 соответственно.

Для задания системы коммутаций достаточно, чтобы для каждой ЭМ было указано, в какую соседнюю ЭМ она должна передавать свою информацию. Это позволяет все ЭМ системы объединить в единую цепочку передачи информации, начиная с некоторой управляющей ЭМ, либо образовать несколько таких цепочек, если имеется несколько иерархических уровней управления. Порядок следования ЭМ в цепочках должен храниться в памяти управляющих ЭМ и последовательно передаваться в другие ЭМ по мере прокладывания пути. Схема, реализующая данный вариант настройки, содержит регистр $R_н$, в котором запоминается номер последующей ЭМ в цепочке передачи информации (рис.

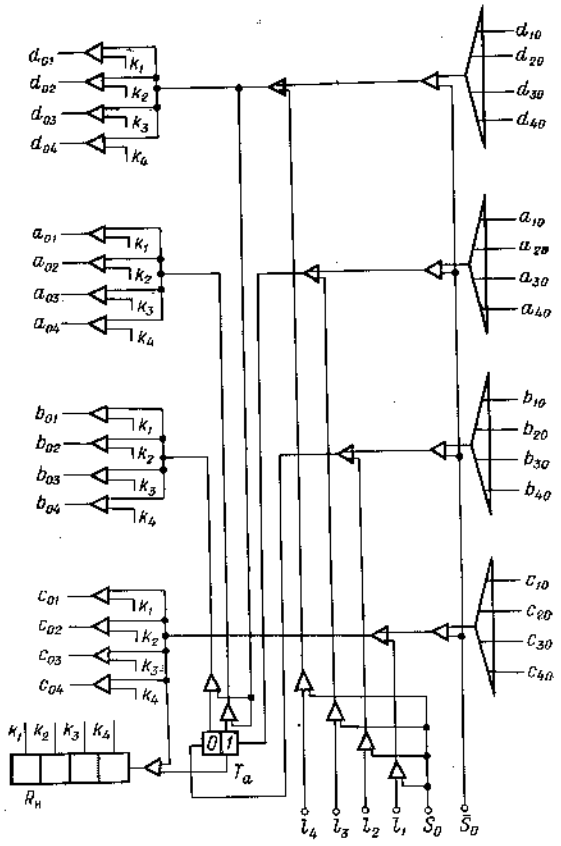


Рис. 23. Схема УВС с переменной структурой настройки.

23). Изменением состояния регистра $R_н$ управляет триггер активности T_a. В состоянии $T_a=1$ код настройки, поступающий либо извне через один из входов c_{10}, c_{20}, c_{30} или c_{40}, либо из памяти самой ЭМ через вход l_1, поступает в регистр $R_н$, что приводит к настройке данной ЭМ.

Выполнение операции настройки начинается с настройки регистра $R_н$ самой управляющей ЭМ. Для этого через вход l_3 подается сигнал, переводящий триггер активности в состояние $T_a = 1$. После этого в регистр $R_н$ засылается код настройки. Этот код отпирает выходы только к одной из соседних ЭМ. Подачей импульса через вход l_4 возбуждается триггер следующей ЭМ (через выходы d_{01}, d_{02}, d_{03} или d_{04}) и одновременно гасится триггер T_a данной ЭМ. Затем через выходы c_{01}, c_{02}, c_{03} или c_{04} передается код настройки следующей ЭМ и т. д., пока не будет образована цепочка из всех ЭМ. Если настройку желательно прекратить на какой-либо ЭМ, то со входа e_2 транзитом через выходы b_{01}, b_{02}, b_{03} или b_{04} всех предшествующих ЭМ посылается сигнал сброса, приводящий триггер активности данной ЭМ в состояние $T_a = 0$ без возбуждения триггера следующей машины.

Нетрудно видеть, что данная схема позволяет реализовать переменную структуру настройки и посредством изменения порядка следования машин в цепочке изолировать неисправные ЭМ либо поврежденные линии связи.

5.8. Система обмена информацией с внешними объектами

В связи с усложнением структуры ВС и увеличением объема памяти появляется возможность возложить на ВС решение все более сложных задач. В этих условиях возникает необходимость в таких новых средствах общения человека с ВС, как обмен слуховой и зрительной информацией. Возможность хранения в ВС достаточно большой информации позволяет снизить требования к скорости ввода и вывода информации в ВС при решении различных частных задач данной области. Вместе с тем решение сложных задач требует совершенствования известных устройств ввода и вывода, таких как печатающее и различные входные устройства. Требование совместной работы в реальном масштабе времени со многими объектами, в том числе и с другими ВС, требует использования параллельно работающих каналов для ввода и вывода информации. Наконец, работа ВС в системе единой информационной сети требует, чтобы она могла работать в системе связи.

Систему обмена информацией можно разделить на четыре группы устройств: обмена зрительными, слуховыми образами, обычного ввода — вывода и многоканального обмена информацией. Остановимся на каждом из них.

Устройства *обмена зрительными образами* можно подразделить на узлы восприятия образа, воспроизведения образа и обработки информации. В качестве узла, воспринимающего изображение, могут служить, например, иконоскоп, матрица фотоэлементов, а в качестве узла, воспроизводящего образ,— кинескоп, слой люминофора, возбуждаемый в результате приложения разности напряжений. В связи с восприятием и воспроизведением информации возникает необходимость в преобразовании воспринимаемого образа в дискретное множество электрических сигналов (дискретную информацию) и обратном преобразовании дискретной информации в некоторый воспроизводимый образ. Эти преобразования сравнительно легко могут быть выполнены (в воспринимающем и воспроизводящем узлах).

Значительно сложнее обстоит дело с построением устройства обработки информации. Алгоритмы для переработки некоторой дискретной информации в зрительный образ сравнительно просты. Основные трудности возникают при переработке информации воспринимаемого изображения, т. е. при решении проблемы распознавания образов. Для решения этой проблемы сложились два основных направления, различаемых по виду используемых устройств. Одно из направлений основано на применении вычислительных устройств (ЭВМ, специализированных машин), работающих по жестко заданному алгоритму. Второе — на применении вычислительных устройств, построенных по принципу самообучения (персептронов). Применение ЭВМ для распознавания образов сейчас не эффективно, так как нет хороших алгоритмов и достаточно быстродействующих ЭВМ. Применение персептронов выгодно в том отношении, что не требуется разрабатывать алгоритмы. Однако известные типы персептронов имеют недостаточное количество элементов, чтобы решать сложные задачи распознавания образов. Увеличение числа элементов станет возможным на основе технологии микроминиатюризации.

При наличии вычислительных систем они сами могут применяться в качестве устройств обработки информации. В этом случае к вычислительной системе добавляется лишь воспринимающее и воспроизводящее устройства (рис. 24). Роль узла обработки информации выполняет вычислительная система. При этом возможны два варианта реализации. Первый вариант сводится к тому, что задача распознавания решается по заданному алгоритму на

ВС со структурой, ориентированной на эту проблему. Во втором варианте для распознавания образов создается некоторая ЭМ с переменной структурой, реализующая алгоритм самообучения, как это делается в персептроне.

Устройство *обмена слуховыми образами* должно состоять из узлов: воспринимающего, воспроизводящего и обработки информации. Воспринимающий и воспроизводящий узлы обмена слуховыми образами проще соответствующих устройств обме-

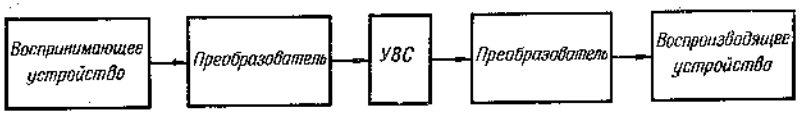


Рис. 24. Схема обмена УВС с внешними объектами.

на зрительными образами, так как звуковые сигналы изменяются только во времени, а зрительные — и во времени, и в пространстве. Это существенно упрощает требования к устройствам восприятия и воспроизведения слуховых образов. Для передачи образов в узел обработки информации и выдачи ее в форме слуховых образов необходимы преобразователи непрерывных сигналов в дискретные и дискретных в непрерывные. Как и для зрительных образов, алгоритмы синтеза слуховых образов проще алгоритмов анализа или распознавания образов, т. е. основная трудность заключается в решении проблемы распознавания образов. Здесь также возможно использование как жестких алгоритмов на основе применения вычислительных устройств, так и алгоритмов самообучения. В обоих вариантах, как и для зрительных образов, решение проблемы распознавания и синтеза слуховых образов может быть возложено на вычислительную систему.

Обычные устройства ввода — вывода данных. Как уже говорилось, по мере повышения производительности ВС для некоторых классов задач возрастают требования к устройствам ввода — вывода информации. Устройства ввода могут включать в себя внешние памяти большого объема, различные читающие устройства на перфокартах, на перфолентах, устройства ввода данных от различных датчиков. К устройствам вывода относятся различные типы печатающих устройств, устройства вывода на перфокарту, перфоленту, фотовывод, вывод данных на различные исполнительные органы, вывод данных во внешнюю память и другие устройства.

К этим устройствам предъявляется общее требование — повышение их надежности, производительности.

Устройства обмена информацией по каналам. Для всех типов устройств ввода — вывода характерно, что они подключаются ко многим каналам ВС. Если учесть еще, что ВС должны включаться в единую информационную сеть и объединяться каналами связи в единую вычислительную систему, то можно отметить большое значение многоканального обмена информацией. Наличие каналов обмена позволяет отделить ВС от остальных устройств ввода — вывода данных. В этом случае для обмена информацией с внешними устройствами наиболее целесообразно применять частотный принцип. При этом можно иметь столько каналов, сколько ЭМ в ВС.

Каждая ЭМ будет иметь свою заданную частоту работы, на которую должны настраиваться внешние источники и приемники информации. Такими источниками и приемниками могут быть другие ВС, информационные сети, отдельные ЭВМ, различные устройства преобразования информации, устройства восприятия и воспроизведения зрительных образов, речи. Все эти устройства для связи с ВС должны иметь приемники и передатчики. Обмен информации на основе использования радиочастотных каналов допускает выполнение ВС и устройств ввода и вывода данных на разной конструктивной основе: ВС — в микроминиатюрном варианте, а системы ввода и вывода данных — в обычном исполнении. Согласование этих устройств посредством радиоканалов не представляет труда. При таком подходе ВС можно рассматривать как систему с обменом информации по радиоканалам без включения в нее устройств ввода и вывода.

5.9. Распределенные УВС

Согласно приведенной ранее классификации (см. 5.6), к распределенным УВС относятся системы, ЭМ которых находятся на таких расстояниях друг от друга, что время распространения сигнала между ними превышает время выполнения операции элементарной машиной. К подобным системам относятся сети, образованные соединением с помощью линий связи обычных ЭВМ, расположенных на значительных расстояниях друг от друга, и УВС, размеры которых не удовлетворяют соотношению (2.1).

Вполне понятно, что основной проблемой для распределенных УВС считается отыскание путей уменьшения относительных затрат времени на передачу информации до таких значений, когда они не сказываются существенно на общем времени решения задач. Некоторые из таких путей рассмотрены в работе [28].

На общем времени решения задач распределенными УВС может сказываться запаздывание сигналов не только в каналах связи, соединяющих ЭМ, но и внутри самих ЭМ при передаче сигналов между отдельными блоками и, более того, запаздывание при передаче сигналов между отдельными узлами внутри данного блока. Когда имеется запаздывание сигналов на всех трех уровнях, можно рассматривать каждый из уровней независимо от нижележащих, включая затраты времени на запаздывание в общее время работы данного блока или машины.

Таким образом, достаточно рассмотреть три случая: 1) запаздывание сигналов между ЭМ существенно, а внутри ЭМ пренебрежимо мало; 2) запаздывание сигналов между блоками ЭМ существенно, а внутри каждого блока мало; 3) запаздывание сигналов между узлами блока существенно, а внутри узла мало.

Рассмотрим каждый из этих случаев.

1. Запаздывание сигналов в каналах связи между ЭМ сказывается в тех случаях, когда возникает необходимость выполнения какой-либо операции над кодом, находящимся в другой ЭМ. В общем случае время выполнения такой операции

$$T = t_0 + 2\tau_{ij}, \tag{5.32}$$

где t_0 — время выполнения данной операции, когда коды находятся в самой ЭМ;

τ_{ij} — время распространения сигнала между ЭМ m_i и m_j.

При этом для простоты считается, что затраты времени на посылку запроса на требуемую информацию те же, что и на ее получение.

Пусть теперь ЭМ m_i в процессе решения задачи выполняет последовательность операций $k_1, k_2, \ldots, k_N$. Время в общем случае будет при этом

$$T_N = tN + 2\tau N, \tag{5.33}$$

где t — среднее время выполнения операции;

τ — среднее время передачи кода из одной ЭМ в другую.

Предположим для простоты, что все операции имеют одинаковую длительность, равную t, и будем рассматривать процесс решения задачи в квантованные промежутки времени $0, t, 2t, 3t, \ldots$

Пусть

$$2\tau = rt, \tag{5.34}$$

где $r > 0$ — целое число, тогда (5.33) можно заменить

$$T_N = (1 + r)\, tN. \tag{5.35}$$

Соотношение (5.35) справедливо, если запрос посылается в тот момент, когда возникает необходимость в данном коде.

Время выполнения операции T может быть существенно сокращено, если запрос посылать заранее с тем, чтобы требуемый код поступил в данную ЭМ до начала соответствующей операции, если программно предусмотреть отсылку кодов в соответствующую машину сразу, как только они возникают в результате выполнения какой-либо операции, или в определенные моменты, выбираемые таким образом, чтобы информация поступала в данные ЭМ до начала соответствующих операций (работа по схеме без запросов информации).

В обоих случаях влияние затрат времени на передачу кодов на общее время решения задачи будет существенно снижено. При схеме работы без запросов память ЭМ должна иметь дополнительный объем для хранения поступающих кодов.

В схеме работы с запросами можно в принципе обойтись без дополнительного объема памяти, но при этом налагается некоторое ограничение на последовательность операций $k_1, k_2, \ldots, k_N$, а, именно, каждая операция k_i не должна зависеть от результатов выполнения r предшествующих ей операций $k_{i-r}, k_{i-r+1}, \ldots, k_{i-1}$. Далее будем называть такие последовательности независимыми в интервале r.

Предположим теперь, что данная последовательность операций выполняется по следующей схеме. На каждом из первых $N - r$ тактах работы ЭМ m_i посылает запросы на требуемую ей информацию в соответствующие ЭМ. Обозначим через $p_1, p_2, \ldots, p_N$ коды, необходимые для выполнения операций $k_1, k_2, \ldots, k_N$ соответственно, а через $[p_1], [p_2], \ldots, [p_N]$ — соответствующие запросы. Пусть запрос $[p_i]$ посылается в момент времени $(i - 1)\, t$ $(i = 1,2,\ldots, N)$, тогда на первых r тактах будут только посылаться запросы, на $r + 1$ такте будет выполнена операция k_1, на $r + 2$ такте — k_2, на $r + i$ такте — k_i и, наконец, на $r + N$ такте будет выполнена операция k_N. Такая схема работы названа конвейерной.

Время выполнения последовательности операций $k_1, k_2, \ldots, k_N$ при конвейерной схеме работы будет, очевидно,

$$T_N^{\text{к}} = tr + tN$$

или

$$T_N^{\text{к}} = (r/N + 1)\, tN. \tag{5.36}$$

На основании (5.35) и (5.36) видно, что конвейерная схема дает выигрыш во времени

$$\frac{T_N}{T_N^{\text{к}}} = \frac{r + 1}{r/N + 1}, \tag{5.37}$$

увеличивающийся с ростом r, величина которого пропорциональна запаздыванию в каналах связи, и увеличением общего числа операций N.

При $\frac{r}{N} \ll 1$ время решения задачи у распределенных УВС будет тем же, что и у сосредоточенных.

Если данная ЭМ m_i должна получать информацию не от одной ЭМ m_j, а от различных ЭМ, то могут быть два случая:

1) когда время τ_{ij} $(j = 1, 2, \ldots, M)$ одинаково для всех машин, (все машины находятся на одинаковом расстоянии от данной). Этот случай не отличается от рассмотренного выше;

2) когда τ_{ij} различны для различных j. При этом $\tau_i = \frac{1}{M} \sum_{j=1}^{M} \tau_{ij}$.

Во втором случае можно также применять конвейерную схему работы, если посылать запросы в такие моменты, чтобы запрашиваемый код поступал в машину m_i в момент выполнения соответствующей операции. Предварительную посылку запросов можно сделать, так как все τ_{ij} известны. При этом, конечно, усложнится программа работы и потребуется согласование загрузок каналов связи. Этого можно в значительной мере избежать, если разрешить, чтобы код поступал в машину m_i в произвольный момент до выполнения соответствующей операции, т. е. поступающие коды направляются в память данной ЭМ, откуда потом затребуются по мере надобности. Заметим, что необходимый объем памяти будет меньше, чем при схеме работы без запросов.

Требование независимости последовательности операций в интервале r может быть выполнено для большинства задач, о чем можно судить на основании анализа различных типов задач, рассмотренных в гл. 8. Действительно, любая программа решения задачи может быть разбита на участки — линейные, ветвящиеся и циклические.

Линейный участок характеризуется тем, что для него заранее известна последовательность операций. Для ветвящегося участка характерно, что в точке разветвления выбор направления вычислений определяется результатом предыдущей операции.

Циклические участки могут обладать свойствами как линейных, так и ветвящихся участков. Для наших целей представляют интерес два случая:

а) для цикла заранее известно число повторений;

б) число повторений цикла заранее не известно, но вероятность выхода из цикла в течение r операций мала.

Аналогичен второму случаю для цикла ветвящийся процесс вычислений с преобладающей ветвью вычислений, вероятность ухода с которой на боковые ветви в течение r операций мала.

Для получения последовательностей из r операций необходимо комбинировать линейные, циклические и ветвящиеся участки путем последовательного присоединения участков с независимыми операциями. В тех случаях, когда операций одной программы недостаточно для получения участков длиной r, чтобы не допустить снижения производительности, можно перейти к одновременному решению нескольких задач.

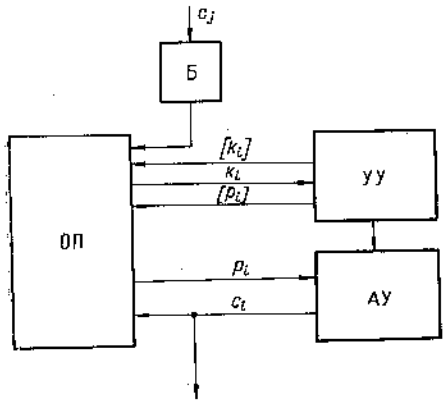


Рис. 25. Схема работы при наличии значительного запаздывания сигналов между блоками ЭВМ.

2. Случай, когда имеется заметное запаздывание в каналах связи между блоками ЭВМ, может рассматриваться аналогично предыдущему. Пусть арифметическое устройство (АУ) и устройство управления (УУ) соединены с оперативной памятью (ОП) каналами связи (рис. 25), время передачи сигналов по которым значительно превышает длительность выполнения операции t_0. Время выполнения операции над хранящимся в ОП кодом, если их выполнять как в обычных ЭВМ, будет

$$T = \tau_1 + t_в + \tau_1 + t_к + \tau_1 + t_в + \tau_2 + t_0 + \tau_2 + t_в, \quad (5.38)$$

где τ_1 — время передачи кода между ОП и УУ;
$t_в$ — время выборки команды из ОП;
$t_к$ — время анализа команды в УУ;
τ_2 — время передачи кода между ОП и АУ;
t_0 — время выполнения операции.

Это соответствует посылке запроса на выборку из ОП очередной команды, самой выборке, пересылке команды в УУ, расшифровке команды, посылке запроса на выборку кода, выборке кода, пересылке кода в АУ, собственно операции, отсылке результата в ОП, запись в ОП. Совершенно очевидно, что такая схема работы при больших значениях τ_1 и τ_2 неудовлетворительна. Для уменьшения влияния запаздывания сигналов в линиях связи, как и в предыдущем случае, может быть применена конвейерная схема.

Пусть, как и ранее, имеется последовательность операций $k_1, k_2, \ldots, k_N$, независимых в интервале r, и положим

$$rt_0 \geqslant T. \quad (5.39)$$

Предположим также, что длительности t_0 у всех операций одинаковы. Пусть теперь запросы на очередные команды $[k_i]$ посылаются в канал связи, не ожидая выполнения предыдущей операции через равномерные промежутки времени. Тогда из ОП в УУ будет поступать ответный поток команд k_i. В УУ эти команды анализируются и в ОП посылается последовательность запросов на коды $[p_i]$, которому соответствует поток кодов p_i, и, наконец, из АУ в ОП отправляется поток результирующих кодов c_i. Будем учитывать следующие ограничения, обычные для ЭВМ:

1) в промежутке времени $t_к$ допускается выбор и выполнение только одной команды; 2) за период t_0 допускается выполнение только одной операции; 3) за период $t_в$ допускается выбор только одного кода; 4) в промежутке времени $1/p$ допускается передача по каналу связи только одного кода, где p — пропускная способность канала, определяемая числом кодов, передаваемых за единицу времени.

В течение цикла выполнения одной операции требуются три обращения к ОП. Кроме того, ОП должна принимать коды и от других ЭВМ (c_j). (Будем предполагать, что ЭВМ обмениваются между собой кодами по схемам без запросов.) Эти коды c_j поступают сперва на буферный регистр (Б), что позволяет синхронизировать их ввод в ОП. Обозначим через $\alpha t_в$ — среднее время на ввод c_j. Учитывая эти ограничения, нетрудно видеть, что каждая последующая операция должна отстоять от предыдущей не менее чем на

$$t_m = \max\left\{t_к, t_0, (3+\alpha)\, t_в, \frac{1}{p}\right\}. \quad (5.40)$$

Общее время выполнения N операций при конвейерной схеме работы составит

$$T_N^к = T_1 + (N-1)\, t_m, \quad (5.41)$$

где T_1 — время от посылки первого запроса $[k_1]$ до выполнения операции k_1. Выигрыш во времени для конвейерной схемы составит

$$\frac{T_N}{T_N^к} = \frac{T_1 N}{T_1 + t_m (N-1)}. \qquad (5.42)$$

При больших N

$$\frac{T_N}{T_N^к} \approx \frac{T_1}{t_m}. \qquad (5.43)$$

Таким образом, и в этом случае конвейерная схема позволяет уменьшить влияние времени передачи кодов на общее время работы.

Требование независимости последовательности операций в интервале r может быть снято, если в АУ добавить дополнительную память объемом r кодов. Тогда результат операций k_i, который потребуется для любой из операций k_{i+1}, k_{i+2},..., k_{i+r}, запоминается в дополнительной памяти и извлекается из нее в соответствующий момент. Введение небольших дополнительных памятей на входе или выходе каждого из каналов позволяет работать в синхронном режиме даже при несинхронной работе блоков ЭВМ, например при различной длительности выполнения операций.

Можно сделать вывод, что построение распределенных вычислительных систем с конвейерной схемой обработки информации позволит эффективно компенсировать влияние запаздывания сигналов на производительность вычислительной системы, увеличить допустимые размеры УВС и ослабить требования к микроминиатюризации. Становится реальным построение ЭМ из отдельных блоков, соединенных каналами связи, ВС на базе имеющегося парка вычислительных машин, объединение в систему машин, удаленных друг от друга на большие расстояния. Общая производительность такой системы будет близкой к суммарной производительности отдельных ЭВМ.

Наконец, открываются возможности для объединения отдельных вычислительных систем в единую вычислительную систему. При этом каждую отдельную ВС можно рассматривать как некоторую сложную ЭВМ.

Естественно, что в случае создания единой вычислительной системы общая производительность может быть получена сколь угодно большой.

Рассмотрим теперь возможные варианты структуры распределенных вычислительных систем. В качестве простейшего примера распределенной вычислительной системы можно назвать ВС, построенную путем объединения ЭВМ, расположенных на больших расстояниях друг от друга. В этом случае из-за высокой стоимости каналов связи целесообразно применять наиболее простые типы УВС, например одномерные двухсторонние УВС (см. 5.10). Естественно, что в структуре одномерной ВС должны быть учтены особенности работы в условиях запаздывания.

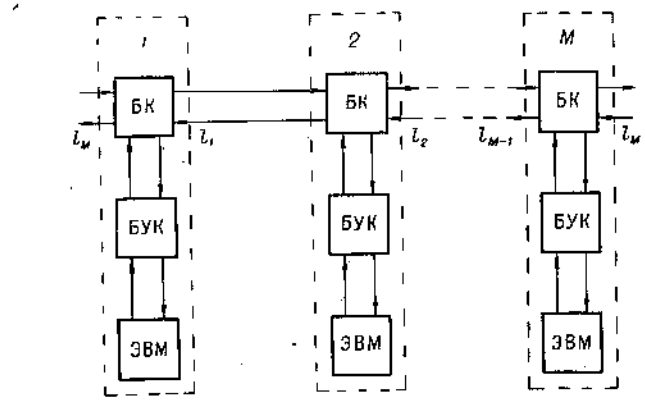


Рис. 26. Блок-схема кольцевой УВС с переменной структурой коммутаций.

В случае построения единой вычислительной системы требования к скорости обмена информацией между входящими в нее ВС повышаются. Растет также требование к надежности каналов обмена информацией. Все это приводит к необходимости создавать более сложные вычислительные системы.

Рассмотрим, в частности, некоторые варианты построения единой вычислительной системы.

Кольцевая вычислительная система (рис. 26). По своей структуре она наиболее проста: состоит из ВС, соединенных двухсторонними каналами связи. В состав каждой ВС входит блок коммутатора (БК) и блок управления коммутацией (БУК), которые обеспечивают возможные варианты коммутаций для вывода или ввода информации. При этом возможна передача данных от одной ВС ко всем остальным или к любой другой ВС, одновременная передача данных всеми ВС своим ближайшим соседям. Предусматривается также возможность управления из данной ВС всеми остальными ВС кольцевой ВС. Скорость обмена информацией должна быть такой, чтобы за приемлемое по условиям ре-

шения задачи время, происходил полный обмен информации в данной ВС.

Кольцевая система ВС обладает повышенной надежностью работы, так как при выходе из строя одной ВС или линии связи

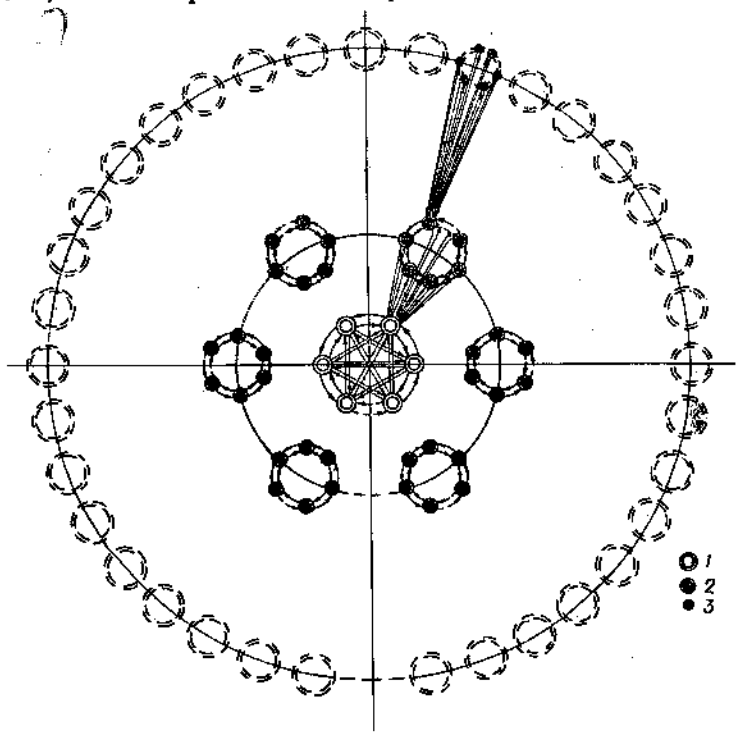


Рис. 27. Схема единой вычислительной сети.

1 — главные вычислительные системы; *2* — областные вычислительные системы; *3* — местные вычислительные системы.

на одном из участков между двумя соседними ВС работа ВС в целом не нарушается.

Единая вычислительная сеть ВС (рис. 27). Единую вычислительную сеть можно представить состоящей из трех иерархических уровней: главных вычислительных систем (ГВС), областных вычислительных систем (ОВС) и местных вычислительных систем (МВС), объединенных каналами связи.

ГВС представляют собой мощные высокопроизводительные вычислительные системы, соединенные друг с другом каналами связи по принципу полного графа.

ОВС соединяются каналами связи с ГВС, и, кроме того, ОВС, тяготеющие к одной и той же ГВС, образуют между собой кольцевую вычислительную систему.

Наконец, МВС соединяются каналами связи с ОВС и имеют каналы связи друг с другом, образуя одномерную вычислительную систему. Чем выше уровень, тем должна быть выше пропускная способность каналов и надежность системы, что выражается в переходе от одномерных ВС на двух нижних уровнях к соединению по принципу полного графа. Здесь так же, как и в случае с кольцевой системой, может быть предусмотрена возможность управления из любой ВС всеми остальными. В единой вычислительной сети ВС благодаря соответствующей организации системы связи может быть достигнута высокая производительность при обеспечении надежности и экономичности.

Многомерная единая вычислительная система. По мере удешевления каналов связи можно надеяться на возможность перехода к созданию двумерных и многомерных распределенных ВС. Роль элементарной машины в этом случае будут выполнять ВС, между которыми организуется система связи в двумерной или n-мерной системе координат. Естественно, что производительность, надежность и гибкость такой распределенной единой вычислительной системы будет выше, чем у систем с одномерной структурой связей.

5.10. Варианты реализаций УВС

Для иллюстрации свойств и возможностей УВС рассмотрим некоторые варианты реализации УВС.

Одномерные УВС, элементарные машины которых соединены с соседними по принципу P_n-графа. Рассмотрение одномерных УВС кроме теоретического интереса может иметь и практическое значение, в частности, для распределенных УВС, образованных из расположенных друг от друга на значительных расстояниях обычных ЭВМ, когда стоимость каналов связи играет существенную роль Применение УВС из обычных ЭВМ, как уже указывалось ранее [26, 28], может рассматриваться как первый этап построения УВС и явиться одним из способов решения сложных задач, требующих выполнения большего числа операций, чем это допускается в отдельных ЭВМ (в том числе и наиболее производительных).

Пусть имеется M вычислительных машин, произвольно расположенных относительно друг друга. Образуем из них последовательность, придав машинам номера от 1 до M. Будем считать,

что каждая из машин m_i соединена с каждым из двух своих соседей m_{i-1} и m_{i+1} двумя каналами связи: входным и выходным. Первая и последняя машины могут быть либо соединены друг с другом (кольцевая система), либо не соединены (линейная система). Будем также считать, что информация одновременно может передаваться по обоим каналам. Такие УВС будем называть одномерными двусторонними УВС.

Предположим далее, что УВС состоит из обычных ЭВМ, соединенных друг с другом программно изменяемой системой коммутаций (УВС с частично переменной структурой). Будем также исходить из того, что каждая ЭВМ может совмещать ввод и вывод информации с работой по основной программе. Иначе говоря, каждая ЭВМ может одновременно выполнять три программы: ввода информации, вывода информации и счета.

Будем считать, что в данной УВС применяется иерархический принцип управления вычислениями и на каждом этапе вычислений одна из машин управляет работой и настройкой всех остальных. При этом каждая ЭВМ либо выполняет команды, поступающие от управляющей машины, либо, если их нет, команды, хранящиеся в памяти самой ЭВМ.

Подобную УВС можно построить из обычных ЭВМ, если добавить к каждой из них блок коммутации и блок управления коммутацией (см. рис. 26). Блок коммутации каждой машины содержит три входных канала связи: от самой машины m_i и от машин m_{i-1} и m_{i+1} и три выходных канала к тем же машинам. Схема коммутации, выполненная, например, на девяти конъюнкторах и трех дизъюнкторах (рис. 28), позволяет соединить входной канал любой из этих трех машин с любым, но только одним выходным каналом другой машины $z_{i\pm k}$ ($k, l = 0, 1, 2$).

Работа блока коммутации может быть описана системой булевых функций:

$$x_k = \bigvee_{l=0}^{2n} C_{kl} \& z_l; \qquad (5.44)$$

$$\bigvee_{\substack{i,j=0 \\ i \neq j}}^{2n} C_{ki} \& C_{kj} = 0,$$

где n — размерность пространственного расположения УВС. Для одномерных УВС $n = 1$.

Блок управления коммутацией служит для управления блоком коммутации путем подачи сигналов на входы C_{kl} и для обмена информацией между ЭВМ и блоком коммутации. В соответствии с этим он состоит из трех узлов: узла управления коммутацией (УУК), узла ввода информации (УВВ) и узла вывода информации (УВ) (рис. 29).

Узел управления коммутацией состоит из регистра R_5, в котором хранится информация, управляющая блоком коммутации, и схемы управления коммутацией и синхронизацией (СУК), с помощью которой выполняется синхронизация ввода, вывода, пуск и останов ЭВМ, приоритет ввода над выводом и т. п.

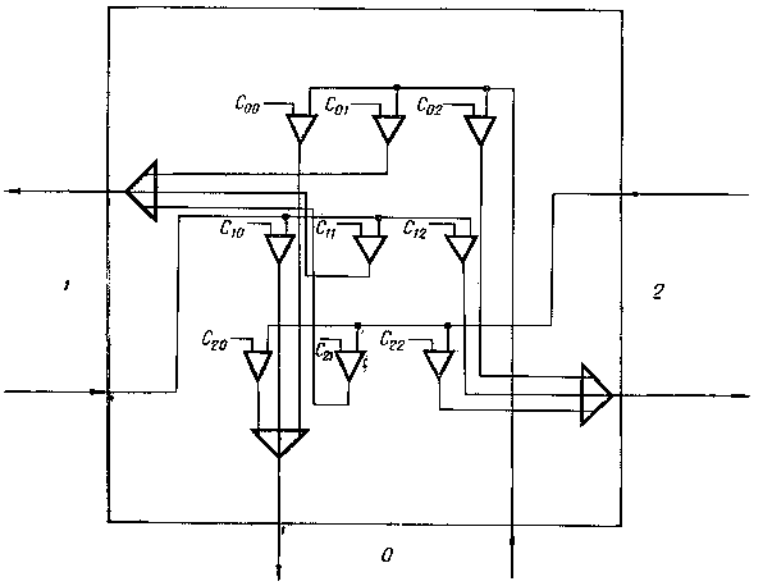


Рис. 28. Схема блока коммутации.

Узел ввода информации состоит из командного регистра ввода R_3, регистра R_4, в который поступает информация из канала x_0, и схемы управления вводом (СУВВ). Информация, поступающая в регистр R_4, имеет адресную часть, состоящую из признака q, указывающего, что информация предназначена всем ЭВМ ($q = 1$) либо только одной ($q = 0$), номер которой n указывается далее, а также номера регистра r, в который должна быть введена данная информация. Схема управления вводом анализирует адрес информации и в тех случаях, когда полученная информация предназначается для данной машины, открывает пути ее передачи в соответствующие регистры.

Узел вывода информации состоит из командного регистра вывода R_1, где хранится очередная команда вывода, регистра R_2,

куда из оперативной памяти данной ЭВМ поступает информация, подлежащая выводу через канал z_0, и схемы управления выводом (СУВ), которая анализирует информацию, поступившую в R_2, и в зависимости от ее адреса открывает пути для ее передачи либо в канал z_0, либо в один из регистров данной ЭВМ.

Возможны и другие варианты схемы блока управления коммутацией, в частности вариант, в котором одна и та же сравнительно простая схема управляет и вводом и выводом информации.

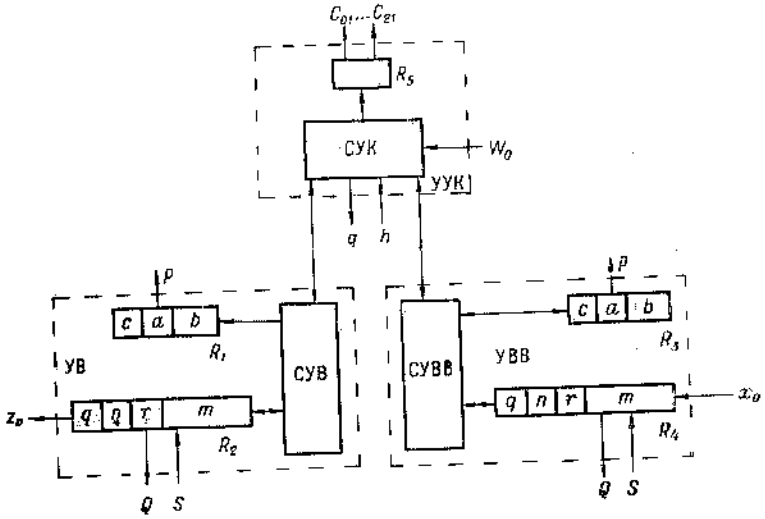


Рис. 29. Схема блока управления коммутацией.

Блок-схема ЭВМ (рис. 30) типовая и состоит из устройства управления (УУ) с регистром команд R_6 и счетчиком команд R_7; арифметического устройства (АУ); оперативной памяти (ОП) с регистром информации R_8, регистром адреса R_9 и самой памятью (F); блока ввода — вывода (БВВ). Для определенности будем предполагать, что машина — обычная серийная ЭВМ, в схему которой внесены небольшие изменения, связанные с подключением блока управления коммутацией.

Многие современные ЭВМ рассчитаны на работу с каналами связи и на режимы совмещения ввода и вывода с работой основной программы, что существенно упрощает построение одномерной двусторонней УВС и фактически сводит его лишь к добавлению относительно простых блоков коммутации и подключению ЭВМ к каналам связи.

Рассмотрим теперь выполнение операций, специфичных для УВС: настройки, ввода, вывода и обобщенного условного перехода.

Состояние коммутатора каждой машины задается командами настройки, поступающими на регистр R_5 либо из данной ЭВМ, либо из ЭВМ, управляющей ходом вычислений. В первом случае на регистре команд R_6 устанавливается команда передачи содержимого ячейки оперативной памяти f, хранящей соответствующую команду настройки, в регистр R_5. Эта команда выполняется

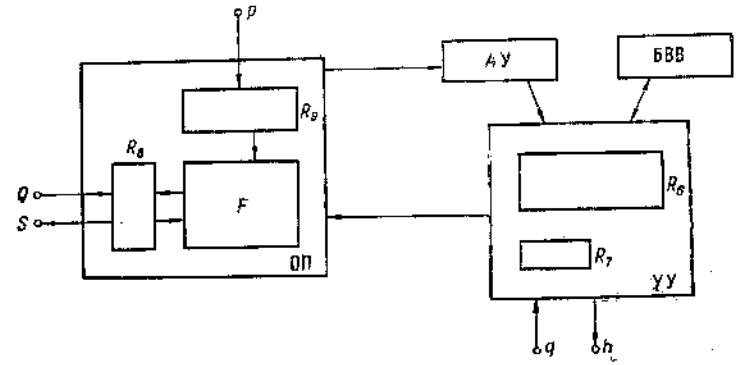


Рис. 30. Блок-схема ЭВМ.

в тот момент, когда нет приема или передачи информации. Во втором случае машина, выполняющая функции управления, транзитом через все машины посылает команду настройки, сопровождаемую или предваряемую специальным сигналом настройки W_0, который включает регистры R_4 всех ЭВМ на прием команды настройки. Этот сигнал может передаваться по особому каналу, проходящему через все ЭВМ (см. рис. 29), либо по тем же каналам, что и остальная информация, в виде особого сигнала. Поступившая в R_4 информация анализируется схемами СУВВ и СУК, и если удовлетворяются соответствующие условия, то информация передается в регистр R_5. В результате изменяется набор сигналов C_{kl}, отпирающих соответствующие вентили блока коммутации, и устанавливается требуемая система коммутаций.

Для описания выполнения операции настройки введем функцию

$$\psi_{\text{н}}^{\text{ц}} = \{q \vee \psi([R_4(n)], i)\} \,\&\, \psi([R_4(r)], r_5), \qquad (5.45)$$

где

$$\psi(x, y) = \begin{cases} 1 \text{ при } x = y; \\ 0 \text{ при } x \neq y; \end{cases} \quad (5.46)$$

$[R_j(k)]$ — содержимое части k регистра с номером j;
r_i — номер регистра R_i;
i — номер данной ЭВМ.

Если $\psi_и = 0$ и в регистре R_6 находится команда передачи содержимого ячейки памяти f в регистр R_5,

$$[R_6] = \Pi f R_5. \quad (5.47)$$

При этом содержимое ячейки f замещает содержимое регистра R_5:

$$[f] = : R_5. \quad (5.48)$$

При $\psi_и = 1$

$$[R_4(m)] = : R_5. \quad (5.49)$$

Для выполнения операции ввода информации на регистре R_3 устанавливается команда ввода c, в ней указывается номер ячейки, начиная с которой размещаются в памяти коды вводимой информации a и число вводимых кодов b. Ввод выполняется порциями, размер которых определяется емкостью регистра R_4. После того как регистр R_4 целиком заполняется поступающей через вход x_0 информацией, его содержимое передается в оперативную память по адресу, содержащемуся в R_3. На это время оперативная память прерывает свою работу по основной программе. Затем регистр R_4 начинает принимать новую порцию информации, а ЭВМ переключается на продолжение основной программы и т. д., пока не будет принята вся информация. После завершения данной операции ввода ЭВМ приступает к выполнению следующей команды ввода, если таковая имеется.

Команда ввода может быть установлена в R_3 либо программным способом из данной ЭВМ, либо путем передачи соответствующей команды из ЭВМ, управляющей вычислительным процессом.

Операция ввода информации может быть описана с помощью функции

$$\psi_{вв} = \{q \vee \psi([R_4(n)], i) \& \psi([R_4(r)], r_3), \quad (5.50)$$

в которой приняты те же обозначения, что и в (5.45).

Тогда при $\psi_{вв} = 0$ и

$$[R_6] = \Pi f R_3 \quad (5.51)$$

выполняется пересылка информации

$$[f] = : R_3, \quad (5.52)$$

а при $\psi_{вв} = 1$

$$[R_4(m)] = : R_3. \quad (5.53)$$

После того как в регистре R_3 оказывается команда ввода, регистр R_4 принимает информацию, поступающую из канала x_0, и порциями, равными емкости регистра R_4, отправляет ее в ОП

$$[x_0] = : R_4; \quad (5.54)$$

$$[R_4] = : a + k, \quad (5.55)$$

где $k = 0, 1, \ldots, b - 1$.

При $k = b - 1$ процесс ввода оканчивается.

Операция вывода информации выполняется аналогично операции ввода. Команда вывода заносится в регистре R_1 либо из самой ЭВМ, либо из ЭВМ, управляющей ходом вычислений, через регистр R_4. В команде вывода информации указывается номер первой ячейки участка памяти a, хранящего выводимую информацию, и общее число считываемых кодов b. После того как все содержимое регистра R_2 передано в канал связи z_0, в регистр R_2 передается следующая порция информации. На этот период оперативная память выключается из основной программы. После завершения данной операции вывода ЭВМ переходит к выполнению следующей команды вывода, если таковая имеется.

Операция вывода информации может быть описана с помощью функции

$$\psi_в = \{q \vee \psi([R_4(n)], i) \& \psi([R_4(r)], r_1), \quad (5.56)$$

где обозначения те же, что и в (5.50).

Тогда при $\psi_и = 0$ и

$$[R_6] = \Pi f R_1 \quad (5.57)$$

в регистр R_1 засылается команда, содержащаяся в ячейке f оперативной памяти

$$[f] = : R_1, \quad (5.58)$$

а при $\psi_в = 1$ — команда, переданная по каналам связи,

$$[R_4(m)] = : R_1. \quad (5.59)$$

После этого начинается вывод информации в канал z_0 порциями, равными емкости регистра R_2:

$$[a + k] = : R_2;\ k = 0,\ 1,\ \ldots,\ b - 1; \tag{5.60}$$

$$[R_2] = : z_0. \tag{5.61}$$

При $k = b - 1$ процесс вывода заканчивается.

Работа каждой ЭВМ УВС по основной программе также может быть описана с помощью аналогичных формул:

$$\psi_0 = \{q \vee \psi\,([R_4(n)],\ i)\}\ \ \&\psi\,([R_4(r)],\ r_6). \tag{5.62}$$

Тогда при $\psi_0 = 0$ выполняется очередная команда, содержащаяся в памяти самой ЭВМ, а при $\psi_0 = 1$ — команда, поступившая из канала x_0,

$$[R_4(m)] = : R_6. \tag{5.63}$$

Выполнение операции обобщенного условного перехода при иерархической системе управления можно представить себе следующим образом. Когда наступает время выполнения команды обобщенного условного перехода, содержащегося в программе машины, которая управляет работой системы, то из этой машины во все остальные поступает специальная команда. Эта команда может передаваться либо по специальному каналу, либо по обычным каналам в виде особой команды. По этой команде каждая машина в управляющую посылает информацию о состоянии, в котором она находится. Управляющая машина анализирует информацию о состоянии всех машин и в зависимости от этого изменяет ход выполнения программы.

При однородной системе управления выполнение операции обобщенного условного перехода можно представить следующим образом. Пусть требуется выполнить операцию обобщенного условного перехода при выполнении какого-либо условия каждой из ЭМ системы, которые, вообще говоря, могут быть различны для различных ЭМ. В программе каждой ЭМ m_i эти условия могут выполняться в виде обычной операции условного перехода, снабженного особым признаком, который указывает, что эта операция участвует в образовании операции обобщенного условного перехода.

Рассмотрим некоторые возможные модификации операции обобщенного условного перехода, отличающиеся условием перехода УВС в новое состояние. Пусть условием перехода будет следующее:

1) достижение всеми ЭМ операции условного перехода, отмеченной признаком. При этом машины, которые раньше достигли этой операции, ждут последнюю машину;

2) кроме предыдущего условия у всех машин выполняется некоторое дополнительное условие;

3) достижение любой из ЭМ операции условного перехода с признаком;

4) кроме предыдущего условия в любой из ЭМ выполняется некоторое дополнительное условие.

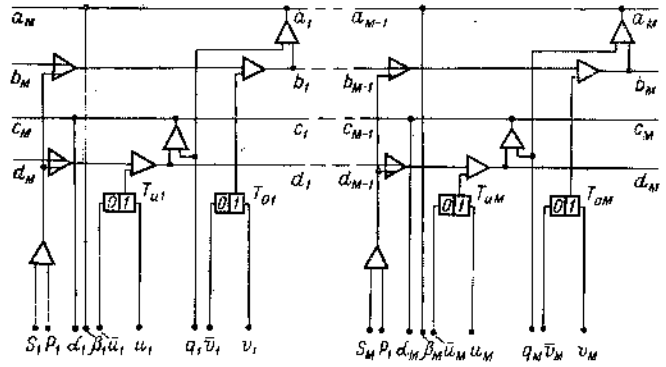


Рис. 31. Вариант схемы, реализующей операции обобщенного условного перехода.

Все четыре случая могут быть реализованы сравнительно простой схемой (рис. 31), приданной каждой ЭМ. Все ЭМ объединяются четырьмя общими каналами связи a, b, c и d.

Схема каждой ЭМ m_i содержит два триггера: T_{ui}, который приходит в состояние $T_{ui} = 1$, когда наступает момент выполнения операции условного перехода с признаком, и T_{0i}, который приходит в состояние $T_{0i} = 1$, когда выполняется некоторое условие, содержащееся в команде условного перехода с признаком. На входы S_i и q_i подаются потенциальные сигналы, отпирающие соответствующие вентили. На входы P_i подаются через определенные промежутки времени импульсы.

Первый случай реализуется подачей сигнала S_1 у первой ЭМ m_1 и сигнала q_M у последней ЭМ m_M и только у них. Тогда после того, как все триггеры T_{ui} через входы u_i будут приведены в состояние $T_{ui} = 1$, импульс со входа P_1 поступит в шину d, пройдет через все вентили и попадет в шину c, откуда в виде сигнала на входы α_i поступит во все ЭМ, которые его воспримут как сигнал выполнения обобщенного условного перехода.

Второй случай отличается от первого только тем, что сигналом обобщенного условного перехода будет одновременный приход сигналов на входы α_i и β_i всех ЭМ. Приход сигнала только на входы α_i означает, что условие, необходимое для выполнения операции обобщенного условного перехода, не осуществилось.

В третьем случае на входы S_i и q_i всех ЭМ поданы потенциальные сигналы. При поступлении сигнала в любой из машин на вход u_i триггер T_{ui} переходит в состояние $T_{ui}=1$. В шине c появляется сигнал, попадающий на входы α_i всех машин.

Четвертый случай отличается от третьего только тем, что сигналом обобщенного условного перехода служит импульс, поступающий в ЭМ через входы β_i.

Соответствующие значения S_i и q_i устанавливаются с помощью операции настройки. Заметим, что не обязательно, чтобы все ЭМ участвовали либо воспринимали операцию обобщенного условного перехода. У тех ЭМ, которые желательно исключить из рассмотрения, с помощью операции настройки триггера T_{ui} и T_{0i} блокируются в двух первых случаях в состоянии 1, в двух последних в состоянии 0, а также могут запираться входы α_i и β_i, через которые поступают сигналы, управляющие переходом. В первых двух случаях роль первой и последней ЭМ могут выполнять любые две ЭМ, соседние, либо такие, между которыми находятся ЭМ, не участвующие в образовании обобщенного перехода.

Подчеркнем некоторые общие свойства одномерных двусторонних систем рассматриваемого типа.

Все машины системы равноправны. Любая из них или все вместе могут управлять ходом вычислительного процесса. При этом не обязательно, чтобы одна и та же машина управляла решением данной задачи от начала до конца. Допустимо разделение процесса решения на этапы, каждым из которых управляет своя машина.

С помощью операций настройки УВС может быть разбита на подсистемы, в каждой из которых находятся ЭВМ с номерами, совпадающими с отрезком натурального ряда. В одномерных системах в каждой такой подсистеме в любой момент только одна ЭВМ может выводить информацию в канал связи. Все остальные машины подсистемы могут при этом принимать либо не принимать данную информацию. ЭВМ, выводящая информацию, может одновременно вводить информацию от какой-либо ЭВМ, не входящей в данную подсистему.

Двумерные УВС с коммутациями по типу P_n-графа. Пусть все ЭМ расположены в узлах однородной квадратной сетки. Будем положение каждой ЭМ характеризовать двумя координатами i, j.

Для каждой ЭМ m_{ij} будем называть соседними $m_{i,j-1}$, $m_{i-1,j}$, $m_{i,j+1}$, $m_{i+1,j}$. Условимся далее при рассмотрении данной ЭМ приписывать ей индекс 0, а ее соседям 1, 2, 3, 4 соответственно. Как и ранее, будем считать, что каждая ЭМ имеет блок коммутации, к которому от каждой соседней ЭМ подходят двусторонние каналы связи. Отличие данного блока от соответствующего блока одномерных УВС (см. рис. 28) сводится к тому, что коммутатор выполняет любое соединение между пятью парами входов и выходов, а не тремя. Работа блока коммутации может быть описана теми же функциями (5.44).

Блок управления коммутацией при двумерном расположении ЭМ будет отличаться от показанного на рис. 29 только бо́льшим объемом регистра R_5, содержащего соответствующие значения элементов матрицы соединений $[C_{kl}]$. Работа блока управления коммутации, выполнение операций ввода и вывода, настройки, обобщенного условного перехода могут быть такими же, как и у одномерных УВС.

У двумерных УВС появляются некоторые особенности, которых не было у одномерных. Например, имеется возможность передавать информацию от одной из ЭМ m_i к любой другой m_j многими путями, которые определяются выбором значений соответствующих элементов матрицы соединений $[C_{kl}]$. Это свойство имеет важное значение для увеличения гибкости использования УВС и, что особенно важно, для повышения надежности УВС (см. 5.7).

Аналогичным образом могут быть построены и трехмерные УВС. В этом случае каждая ЭМ будет соединена с шестью соседними, что вызывает соответствующие изменения в блоках коммутации, управления коммутацией и в каналах связи.

Глава 6

МИКРОСТРУКТУРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

6.1. Основные направления при разработке микроструктуры вычислительных систем

В гл. 5 вычислительная система рассматривалась как совокупность трех элементов: элементарных машин, коммутаторов и каналов связи, которые были названы макроэлементами.

Обсудим теперь следующий уровень организации ВС: структуру макроэлементов, которую будем называть *микроструктурой* вычислительных систем.

При рассмотрении микроструктуры ВС устанавливаются некоторые первоначальные, рассматриваемые как неделимое целое элементы, из которых построены все макроэлементы. Эти элементы мы будем далее называть микроэлементами.

Такой подход, когда отдельные блоки и устройство в целом создаются из набора некоторого числа элементов, фигурирующих как неделимое целое, характерен для вычислительной техники, как впрочем, по-видимому, и для большинства сложных систем. Этот же подход нашел отражение и в теории конечных автоматов [1—3]. Конструкцию автомата принято задавать путем формального определения схемы, описывающей, каким образом сложный автомат построен из неподлежащих дальнейшему расчленению элементарных автоматов или других элементов. Каждый элемент описывается системой канонических уравнений и имеет входы и выходы для соединения с другими элементами или внешними объектами. Внешние концы этих входов и выходов принято называть *полюсами*.

Схема автомата строится из элементов путем отождествления полюсов в соответствии с определенными правилами. Никаких ограничений на длину каналов связи между элементами в теории автоматов не накладывается. Другими словами, допускается отождествление полюсов элементов, расположенных друг от друга сколь угодно далеко. Не налагаются также ограничения

на число входных полюсов, отождествляемых с выходным полюсом данного элемента. Отсутствие указанных ограничений отражает особенности существующих ЭВМ, у которых число элементов сравнительно невелико, а длина каналов связи может меняться в широких пределах и практически мало влияет на сложность и стоимость ЭВМ, определяемых в основном функциональными элементами. Обычно различают три типа функциональных элементов: логические, усилительные и запоминающие, которые отличаются как по конструкции, так и по сложности и стоимости.

При переходе к построению сложных ЭВМ и ВС, состоящих из большого числа элементов, картина становится иной. Во-первых, в сложных системах допущение соединения друг с другом любых элементов настолько усложняет конструкцию, что это может оказаться основным препятствием для ее реализации. Во-вторых, при большом числе элементов разнообразие типов элементов и связей между ними также может сделать конструкцию трудноосуществимой.

Таким образом, для сложных ЭВМ и ВС первоочередное значение приобретают максимально возможное укорочение связей между элементами и уменьшение как числа типов элементов, так и разнообразия соединений между ними. С этой точки зрения очень перспективно построение вычислительных систем в виде итеративных цепей [4]. Итеративные цепи обладают следующими свойствами:

1) все элементы имеют одну и ту же структуру (конструктивная однородность элементов);

2) все элементы соединены одинаковыми связями с другими (обычно соседними) элементами (однородность соединений между элементами);

3) имеется возможность наращивать схему путем присоединения к крайним элементам новых.

Применение итеративных структур особенно выгодно для микроминиатюрных конструкций. С одной стороны, итеративные структуры существенно облегчают производство схем, которое при этом сводится к многократному повторению одного и того же элемента и его связей. С другой — использование новых физических явлений и технологии микроминиатюризации создает необходимые предпосылки для применения итеративных структур. Из работ по исследованию особенностей технологии микроминиатюризации [5—11], можно сделать вывод, что при микроминиатюризации стирается грань между различными типами функциональных элементов (логическими, формирующими и запоминающими). Эти элементы становятся близкими по сложности, и их выгодно делать на одной основе. В качестве примера можно

привести криотронные ЭВМ [12], у которых все функциональные элементы строятся из одного и того же элемента — криотрона.

Увеличение роли соединений при уменьшении размеров элементов привело к возникновению принципиально нового типа элементов *нейристоров* [13]. Эти элементы служат одновременно и соединительным каналом и логическим элементом, элементом памяти и формирующим элементом. По сути, они представляют собой канал связи, по которому могут распространяться сигналы с постоянной скоростью и без потери энергии.

Рассмотрение сложившихся направлений в области микроминиатюризации показывает, что на изготовление соединений между элементами требуется не меньше технологических операций, чем на изготовление функциональных элементов. Если же учесть соединительные каналы между далеко расположенными элементами, то затраты на каждый такой канал будут превышать затраты на элементы, которые он соединяет. Если принять во внимание, что связи занимают бо́льший объем, чем элементы, то фактически *структура микроминиатюрных схем* будет в основном *определяться связями*. В этих условиях целесообразно наряду с функциональными элементами ввести элементы связи, выполняющие функции как каналов связи, так и коммутаторов.

Каждо́му функциональному элементу придается элемент связи, посредством которого (и только через него) данный функциональный элемент соединяется с другими элементами. Такое сочетание функционального и соединительного элементов имеет смысл рассматривать как один функционально-соединительный элемент и считать, что схема образована многократным повторением одного элемента, выполняющего функции соединительного, логического, формирующего и запоминающего элемента. Такой элемент далее будет называться *универсальным*.

Требование простоты структуры универсального элемента связано с применением принципа близкодействия, согласно которому непосредственно воздействуют друг на друга только соседние элементы.

Универсальный элемент должен также обладать свойствами, обеспечивающими возможность программного изменения макроструктуры ВС, т. е. числа и свойств элементарных машин, каналов связи и коммутаторов в зависимости от требований решаемых задач.

Это требование может быть реализовано либо путем введения относительно сложных элементов с переменной структурой, реализующих широкий класс функций, либо применением относительно простых элементов с переменной структурой соединений между ними. Второй путь более реален, так как сводится к созда-

нию функционального элемента, реализующего логически полную систему функций, и элемента соединения, позволяющего реализовать любую конфигурацию связей схемы.

Из учета требований технологии целесообразно строить микроструктуры ВС из одинаковых и одинаково соединенных элементов. Будем называть такие микроструктуры вычислительными средами.

Вычислительные среды нашли широкое применение в вычислительной технике непрерывного действия [14—16]. В технике непрерывного действия для моделирования полей используются два метода: метод сплошных сред и метод сеток. Метод сплошных сред обычно реализуется путем создания электрического поля в сплошной проводящей среде. Метод сеток основан на дискретном представлении элементарных объемов моделируемой области и замещении их электрическими цепями с сосредоточенными параметрами, в результате чего образуется электрическая сетка [17].

Эти среды предназначаются для решения вполне определенных задач и в этом отношении их можно считать *специализированными*.

Для специализированных вычислительных сред характерна высокая однородность как элементов, так и соединений между ними. При этом элементы соединены друг с другом по принципу близкодействия. Указанные свойства обеспечивают высокую надежность сред. Кажется весьма заманчивым построение УВС из сред такого типа. Однако этому препятствуют следующие их свойства: 1) все элементы среды выполняют при моделировании одну и ту же функцию, передают и принимают информацию по фиксированным направлениям; 2) элементы среды не универсальны и могут выполнять только одну определенную (хотя, может быть, и сложную) функцию.

Универсальность ВС требует применения вычислительных сред, рассчитанных на широкий круг задач. Такие среды будем далее называть *универсальными*.

Основные свойства универсальных вычислительных сред были сформулированы в работе [18]. Они сводятся к следующему:

о д н о р о д н о с т ь — вычислительная среда состоит из одних и тех же элементов с одинаковыми конфигурациями соединений между ними;

п р и н ц и п б л и з к о д е й с т в и я — элементы имеют связи только со своими ближайшими соседями;

у н и в е р с а л ь н о с т ь э л е м е н т о в — каждый элемент среды выполняет полный набор логических функций

(например, функции дизъюнкции $\vee$, конъюнкции & и отрицания $\neg$), задержку сигнала и функцию соединения с другими элементами;

возможность настройки элементов — каждый универсальный элемент может настраиваться на выполнение той или иной функции, а также на задание различных направлений обмена информацией.

Из двух первых свойств следует простота технической реализации ВС, которая сводится к многократному повторению одного микроэлемента.

Третье свойство позволяет делать любые схемы, а следовательно, любую ЭВМ или ВС.

Благодаря четвертому свойству можно быстро программным способом перестраивать схемы ЭВМ и ВС как заранее (в зависимости от класса решаемых задач), так и в процессе вычислений.

По-видимому, существует большое количество самых разнообразных вычислительных сред, удовлетворяющих заданным требованиям. Представляет большой практический и теоретический интерес отыскание универсальных континуальных вычислительных сред.

Следует отметить, что возможность моделирования «поведенческих актов» управляющих систем континуальными средами показана в работе [19], где говорится, что континуальная среда обладает свойствами запоминания, если у ее элементов возникает спонтанная активность. Сейчас континуальные среды используются при создании отдельных элементов [13], строить же ЭМ на основе континуальной среды довольно трудно из-за сложности и недостаточной изученности явлений в среде.

Более реально использовать для ЭМ дискретные вычислительные среды. Возможность применения дискретной среды для моделирования простейших актов поведения показана в работе [20]. Здесь выполнение той или иной сложной функции определяется групповым взаимодействием элементов, каждый из которых выполняет одну и ту же функцию (в том числе и функцию соединения). Будем называть такую дискретную среду *универсальной вычислительной средой с коллективным поведением элементов*.

В качестве другого вида дискретной среды можно назвать среду, в которой возможна индивидуальная настройка любого отдельного элемента среды на выполнение какой-либо одной из полного набора функций. При этом под функцией, выполняемой элементом, понимается функция задержки, функции $\vee$, &, $\neg$ и т. д., прием и передача информации по заданным направлениям. Такой вид среды будем называть *универсальной вычислительной средой с индивидуальным поведением элементов*.

Применение универсальных вычислительных сред позволяет по-новому подойти к разработке проблем, связанных с созданием ВС высокой производительности.

Разработка самой ВС перестает быть связанной с разработкой методов решения задач и методов программирования на ВС. Благодаря существенному упрощению структуры ВС снижаются требования к технологии изготовления ВС. Однородность структуры позволяет сосредоточить все усилия на разработке элемента среды.

Применение вычислительных сред позволяет объединить достоинства проблемно-ориентированных и универсальных вычислительных систем, так как возможность наилучшим образом использовать имеющееся оборудование сочетается с возможностью построить ВС заранее.

Вычислительные среды позволяют обеспечить построение ВС с максимально высокой производительностью, надежностью и экономичностью.

Рассмотрим теперь более подробно основные типы универсальных вычислительных сред: континуальные (см. 6.2); дискретные с коллективным поведением элементов (см. 6.3); дискретные с индивидуальным поведением элементов (см. 6.4—6.12).

Основное внимание при этом уделим средам с индивидуальным поведением элементов, так как среды такого типа, по-видимому, при данном состоянии техники легче реализовать.

6.2. Континуальные среды

Начало изучению свойств континуальных сред было положено в 1946 г. Н. Винером и А. Розенблютом [21]. Основная цель их работы — исследовать распространение импульсов в сети из связанных возбудимых элементов, в частности в сердечной мышце. Сердечная мышца при распространении импульсов может рассматриваться как единая однородная структура, возбуждение в любой части которой может распространиться на всю мышцу. В качестве объекта исследования сердечная мышца выбрана потому, что она и больше экспериментально изучена, чем, например, нервная ткань.

В сердце помимо биений может иногда наблюдаться и другой тип сокращения, называемый *трепетанием*, или *флаттером*. Трепетание представляет собой непрерывно циркулирующую волну, бегущую по определенному замкнутому пути.

Математическая модель среды строится исходя из следующих свойств:

1) импульсы, однажды возникнув, распространяются с постоянной во всех направлениях скоростью (т. е. среда однородна, и составляющие ее части имеют повсюду одинаковые свойства);

2) амплитуда колебательного процесса, вызывающего распространение возбуждения, остается постоянной. По величине она превышает порог смежных областей, если они находятся в состоянии покоя;

3) каждая область среды может находиться в одном из трех состояний:

а) активное, бывает только на мгновенном фронте волны;

б) рефрактерное, наступающее сразу после активного состояния. Рефрактерному состоянию приписывается постоянная продолжительность. Область, находящаяся в этом состоянии, не возбуждается импульсами и не передает их;

в) состояние покоя, следующее непосредственно за рефрактерным состоянием и длящееся до следующего возбуждения.

За каждым свободно движущимся фронтом волн активности следует полоса фиксированной ширины, внутри которой проходит восстановительный процесс. Ширину этой полосы авторы назвали *длиной волны*.

Рассмотрим процессы, происходящие в средах с указанными выше свойствами.

Пусть дано одиночное волокно с двумя свободными концами, поперечные размеры которого меньше длины волны, т. е. дана одномерная разомкнутая система. Если на обоих концах волокна возникнут импульсы, которые будут распространяться к противоположному концу, тогда в соответствии со сформулированными выше свойствами могут быть три случая:

1) если второй импульс возникает через время $t \geqslant (l + w)/v$ после первого, где l — длина волокна, v — скорость распространения, w — длина волны (ширина рефрактерной области), то импульсы не будут взаимодействовать и *оба импульса достигнут противоположного конца*;

2) если $l/v < t < (l + w)/v$, то *второй импульс не пройдет*, так как на его пути встретится участок волокна, находящийся в состоянии рефрактерности;

3) если $t < l/v$, то *оба импульса* встретятся внутри волокна и *взаимно погасятся*, так как и вправо и влево от точки встречи волокно будет находиться в рефрактерном состоянии.

Как увидим далее, указанные свойства одномерных разомкнутых систем позволяют реализовать полный набор логических функций, а также передачу информации от элемента к элементу. Однако в разомкнутой одномерной цепи активность не может самостоятельно поддерживаться неопределенно долго, иными словами, такая цепь не может реализовать функцию памяти.

Рассмотрим теперь одномерный *замкнутый* путь. В этом случае оказываются справедливыми утверждения, что устойчивая циркуляция импульса по замкнутому волокну не может быть вызвана путем:

1) единичного раздражения, нанесенного в любой точке волокна. В этом случае от точки раздражения возникнут две противоположно направленные волны, которые затем встретятся в какой-либо точке замкнутого волокна и погасят друг друга;

2) последовательным нанесением раздражения в одной и той же точке. Так как скорость распространения волн одинакова, то этот случай не отличается от предыдущего;

3) одновременным нанесением раздражений в нескольких точках. Действительно, если рассмотреть две любые соседние точки, то волны от них будут идти навстречу друг другу и взаимно погасятся.

Для образования устойчивой циркуляции импульса необходимо образовать только одну волну или несколько волн, движущихся в одном направлении. Н. Винер и А. Розенблют, чтобы достигнуть этого, предложили следующее. В области точки r_1 замкнутого волокна в момент t наносится раздражение, вызывающее появление двух волн (рис. 32, *а*). Через время $t + \Delta t$ в области точки r_2, находящейся на границе между покоящейся и рефрактерной областью правой волны, наносится новое раздражение. Так как справа от точки r_2 находится рефрактерная область, то волна образуется только в одном направлении (рис. 32, *б*). После того, как волны, образованные первым раздражением, встретятся и взаимно погасятся, в волокне останется только одна волна, которая будет беспрепятственно циркулировать по нему. Для образования одной волны, как легко видеть, должны быть выполнены определенные соотношения между моментом времени нанесения второго раздражения t и размерами стимулированной области λ около точки r_2

$$(w + \lambda/2)/v < t \leqslant (w + \lambda)/v. \qquad (6.1)$$

Соблюдение (6.1) обеспечивает то, что участок длиной λ, на который наносится второе раздражение, захватывает часть рефрактерной области правой волны и не захватывает рефрактерной области левой волны. Можно установить и обратное соотношение

$$tv - w - \varepsilon_1 < h < tv - w + \varepsilon_2, \qquad (6.2)$$

где h — совокупность точек, в которых надо нанести раздражение в момент t;

ε_1 и ε_2 — положительные числа; $\varepsilon_1 + \varepsilon_2 = \lambda$.

Соотношения (6.1) и (6.2) позволяют объяснить явления трепетания для одномерных замкнутых структур, а для технических применений указывают, что на этой основе возможно создавать элементы памяти.

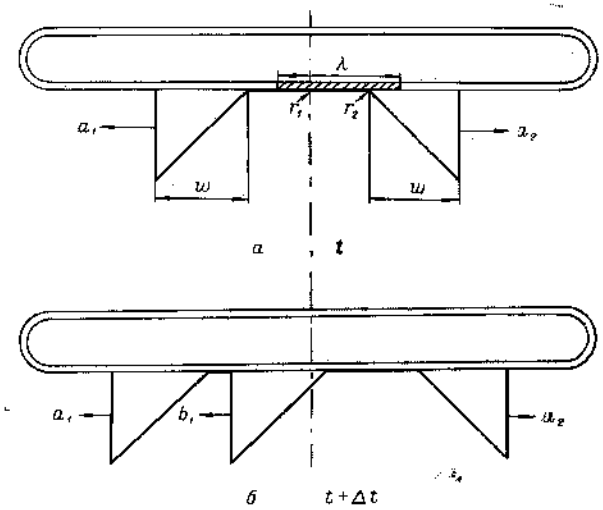


Рис. 32. Образование незатухающей волны в замкнутом волокне.

Исследование однородной двумерной системы показывает, что возбуждение распространяется в ней по принципу Гюйгенса: последовательные фронты волны перпендикулярны воображаемой системе лучей из натянутых линий, которые начинаются в возбужденной точке и огибают все препятствия. Задний фронт волны рефрактерности — кривая той же формы — следует за передним фронтом волны на расстоянии w, отложенном вдоль лучей. Фронт волны может распространяться только в область, находящуюся в состоянии покоя.

На бесконечной поверхности без отверстий и других препятствий передний фронт волны, распространяющейся из точки, будет представлять собой окружность с центром в этой точке, уходящую в бесконечность. В конечной выпуклой области без препятствий раздражение одиночной точки образует расширяющийся круговой фронт волны, исчезающий без отражения на границах. При наличии препятствий круговая форма фронта волны может изменяться. В этом случае фронт волны в области за препятствием распространяется по эвольвенте граничной кривой препятствия.

Трепетание, связанное с возникновением стационарных незатухающих волн, движущихся в одном направлении, может возникнуть благодаря лишь тем волнам, у которых передние и задние фронты являются эвольвентами препятствий. Так как для распространения волны задний фронт не должен интерферировать с передним, периметр препятствия должен быть больше длины волны. Это означает, во-первых, что в односвязной области (область без препятствий) активность не может самовоспроизводиться и, во-вторых, что периметр выпуклого препятствия, которое нарушает односвязность, должен быть больше длины волны. Это утверждение имеет силу и для препятствий с вогнутыми участками. Исследование двумерных систем с двумя и более препятствиями, а также трехмерных систем с однородной проводимостью позволило обобщить результаты на случай возникновения трепетаний в однородной изотропной спонтанно неактивной проводящей среде.

Эти результаты сводятся к следующему.

1. Трепетание не может быть вызвано одиночным раздражением из точки или области.

2. Трепетание не может быть вызвано также одновременной стимуляцией любого числа точек или областей.

3. Для возникновения трепетания необходимы, по крайней мере, два раздражения, приложенные в различных областях и разделенные соответствующим промежутком времени. Второе раздражение должно быть приложено к области, пересекающей задний фронт волны. Поэтому промежуток времени, разделяющий эти два раздражения, равен сумме рефрактерного периода и времени распространения волны между двумя возбужденными областями.

4. Нанесение последующего раздражения должно быть отделено от предыдущего, по крайней мере, на время, равное продолжительности рефрактерного периода. Передний фронт волны в произвольной двумерной системе есть незамкнутая или замкнутая кривая. Если эта кривая не замкнута, то ее свободные концы должны лежать или на геометрических границах или на задних фронтах волн.

5. По истечении одного рефрактерного периода после нанесения последнего раздражения в системе не увеличивается число пересечений задних и передних фронтов волн.

6. Число передних и задних фронтов волн в системе может уменьшаться.

7. Во всякой односвязной области, в которой по истечении рефрактерного периода после окончания раздражения нет пересечений передних и задних фронтов волн, трепетание невозможно.

Дальнейшее развитие теории континуальных сред сделано И. М. Гельфандом и М. Л. Цетлиным [19]. Ими рассмотрены свойства континуальных сред при следующих условиях:

1) скорость распространения возбуждения зависит от времени, прошедшего с момента возбуждения;

2) каждая точка среды обладает свойством спонтанной активности, т. е. через время T после последнего возбуждения каждая точка вновь возбуждается самопроизвольно.

Таким образом, каждая точка среды находится в колебательном режиме с периодом T. При воздействии на точку одиночным импульсом она будет продолжать колебания с тем же периодом, но уже с иной фазой, т. е. такая среда обладает способностью к запоминанию.

Каждая точка x, не подвергающаяся внешнему воздействию, будет находиться в одной и той же фазе $\tau(x, t)$ в моменты $t = 0, T, 2T, \ldots, nT, \ldots$

Рассмотрим, как изменяются фазы точек среды, если подать возбуждение на одну точку x_0 в момент $t_0 > R + nT$, где R — длительность рефрактерного периода.

Допустим, что точку x_0 возбудили в момент $t = 3$ (рис. 33). Тогда линии одинаковых фаз τ_i ($i = 0, 1, 2, \ldots$) покажут развитие картины во времени, соответствующем скорости распространения возбуждения v. Фронт возбуждения имеет вид конуса, который расширяется до момента T, после чего форма и размеры конуса, образующего фронт волны возбуждения, сохраняются неизменными во времени. В момент T все точки, на которые не успело распространиться возбуждение, начинают самовозбуждаться и фронт волны, возникшей из точки x_0, достигает их в момент, когда они находятся в рефрактерном периоде. Непостоянство скорости распространения возбуждения v сказывается при этом только на форме фронта волны.

На основании исследований, сделанных в работах [19—21], можно сделать вывод, что континуальные среды могут быть использованы для создания микроэлементов ВС.

Действительно, возможность образования незатухающих волн с постоянной амплитудой позволяет выполнять логические функции, функции соединения и функции памяти. Как показано в работе [13] на основе континуальных сред могут быть созданы сравнительно простые элементы — нейристоры. Нейристор можно представить в виде канала, по которому распространяется сигнал в форме разряда без затухания и с постоянной скоростью. Для его реализации необходимо иметь распределенный источник энергии, распределенное накопление энергии; распределенный активный элемент.

Участки канала позади движущегося фронта возбуждения находятся в состоянии рефрактерности.

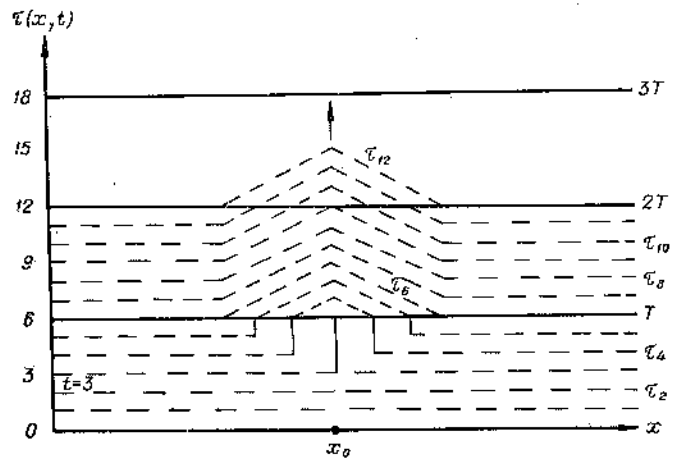


Рис. 33. Распространение единичного возбуждения в среде со спонтанной активностью.

Возбуждение в любом месте нейристора распространяется на соседние участки.

Для построения нейристорной логики достаточно двух типов соединений нейристорных линий: T- и R-соединения (рис. 34).

В T-соединении разряд, идущий слева направо по каналу *1*, возбуждает каналы *2*, *3*, ..., *n*.

В R-соединении разряд, идущий по каналу *1—1* (или *2—2*), не вызывает разряда в канале *2—2* (или *1—1*). На время рефрактерного периода канал *2—2* (или *1—1*) запирается. В течение рефрактерного периода после каждого импульса в канале *1—1* (или *2—2*) импульсы в канале *2—2* (или *1—1*) пройти не могут и затухают.

Используя эти два соединения, можно строить различные функциональные элементы:

клапан (рис. 35, *а*) — по каналу *A—B* импульсы идут только в том случае, если в канал *C* не поступают импульсы;

вентиль (рис. 35, *б*) — импульсы могут передаваться только от *B* к *A*, т. е. эта схема эквивалентна диоду;

кольцевой элемент памяти (рис. 35, *в*) — если на вход *A* (*B*) подан импульс, то благодаря *R*-соединению в кольце возникнет только одна волна возбуждения, которая будет циркулировать

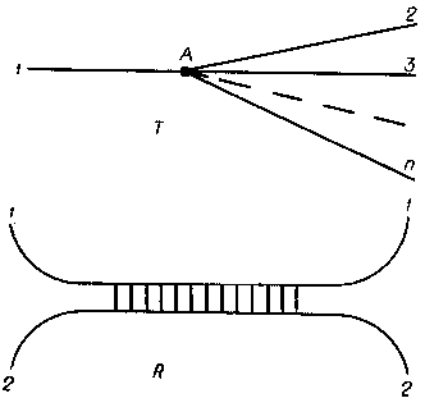


Рис. 34. Основные элементы нейристорной логики.

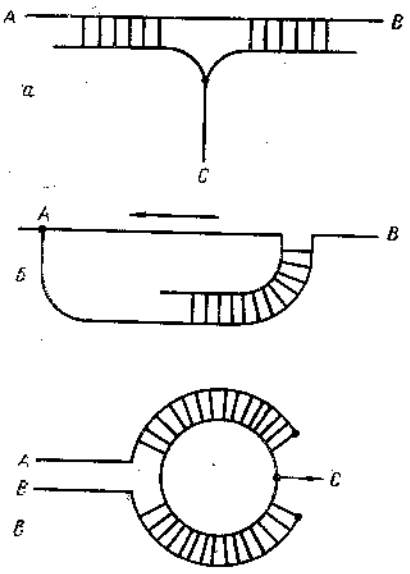
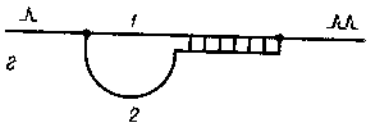


Рис. 35. Нейристорные функциональные элементы.

а — клапан; *б* — вентиль; *в* — память; *г* — умножитель числа импульсов.



сколь угодно долго, что эквивалентно запоминанию. Считывание информации производится с выхода *C*. Гашение достигается подачей импульса на другой вход *B* (*A*).

Из нейристорных элементов можно строить обычные элементы вычислительной техники. Свойства нейристоров позволяют также строить элементы, обладающие и другими качествами. Например, можно получить простую схему умножителя количества импульсов (рис. 35, *г*). Пусть канал *2* длиннее канала *1*. Тогда на выходе количество импульсов будет удваиваться.

Применение однородных непрерывных сред в качестве основы для построения элементов вычислительной техники весьма перспективно. Главные достоинства такого направления в высокой технологичности, надежности, простоте схем.

Реализация с помощью континуальной среды какого-либо сложного логического устройства требует применения весьма сложных конфигураций среды и учета большого числа временных соотношений, поэтому такое заманчивое направление применения континуальных сред вряд ли может быть реализовано в ближайшее время.

В заключение следует отметить, что несмотря на большой срок, прошедший со времени опубликования работы Н. Винера и А. Розенблюта [21], практических результатов в применении континуальных сред в вычислительной технике не получено.

По-видимому, теория континуальных сред может быть распространена на все направления разработки элементов и устройств вычислительной техники, в основе которых лежат принципы взаимодействия волн (в том числе световые элементы, элементы СВЧ и т. п.).

6.3. Среды с коллективным поведением элементов

И. С. Балаховским [20] была исследована возможность моделирования простейших актов поведения с помощью дискретных однородных сред. Каждый элемент среды располагается в узлах однородной сетки и соединяется со своими восемью соседями одинаковыми связями с двухсторонней проводимостью (рис. 36). Элемент может находиться в трех состояниях: возбужденном, рефрактерном и покоя. Каждый возбужденный элемент передает возбуждение всем своим соседям за одно и то же время τ. Время нахождения элемента в возбужденном состоянии $\mu \ll \tau$. Время рефрактерного состояния $2\tau < R < 3\tau$. Состояние системы рассматривается только в квантованные моменты времени: 0, τ, 2τ, ..., $n\tau$, ..., что позволяет избежать трудностей, связанных с синхронизацией.

Элемент не может возбудить тот элемент, от которого он возбудился сам. Если предполагать, что каждый элемент возбуждается одним импульсом, полученным от любого из соседей, то возбуждение будет распространяться во все стороны, пока не выйдет за пределы рассматриваемого участка.

Если возбуждение будет происходить только при одновременном приходе двух импульсов, то картина будет иной. На рис. 37 показан процесс передачи возбуждения для пяти последовательных моментов указанных цифрами внутри элементов. При возбуждении нескольких соседних вертикально расположенных

элементов возникает волна возбуждения, распространяющаяся в горизонтальном направлении. Этот пример показывает, что выбором элементов, возбуждаемых в начальный момент, можно управлять направлением передачи возбуждения.

На этой же основе можно реализовать функции алгебры логики и функции памяти. Например, операции дизъюнкции $C = A \vee B$ и конъюнкции $C = A \,\&\, B$ могут быть реализованы

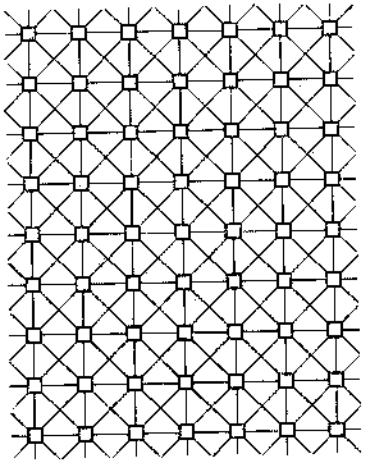


Рис. 36. Схема однородной дискретной среды.

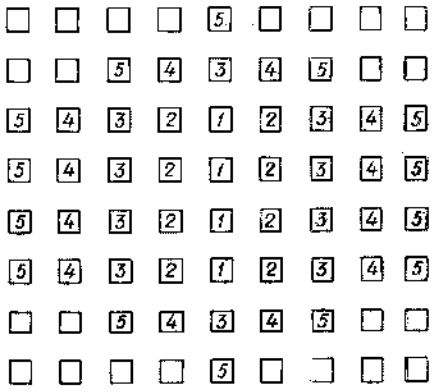


Рис. 37. Распространение возбуждения в среде с порогом два.

так (рис. 38, *а*, *б*). Запоминание информации может быть реализовано путем создания незатухающей серии возбуждений (рис. 38, *в*). Одновременное возбуждение элементов группы *1* сменяется возбуждением элементов группы *2*, которые возбуждают элементы группы *3*, а они вновь возбуждают элементы группы *1* и т. д. Для получения подобной картины достаточно возбудить четыре элемента, помеченных звездочкой.

Указанные выше среды могут быть использованы и в вычислительной технике. В дальнейшем мы их будем называть *средами с коллективным поведением элементов*.

Основная трудность реализации схем вычислительных машин с помощью сред с коллективным поведением элементов заключается в необходимости одновременно возбуждать большое число элементов в определенной конфигурации. Эта трудность частично снимается, если ограничиться построением относительно простых схем, т. е. создавать на этой основе не целые устройства,

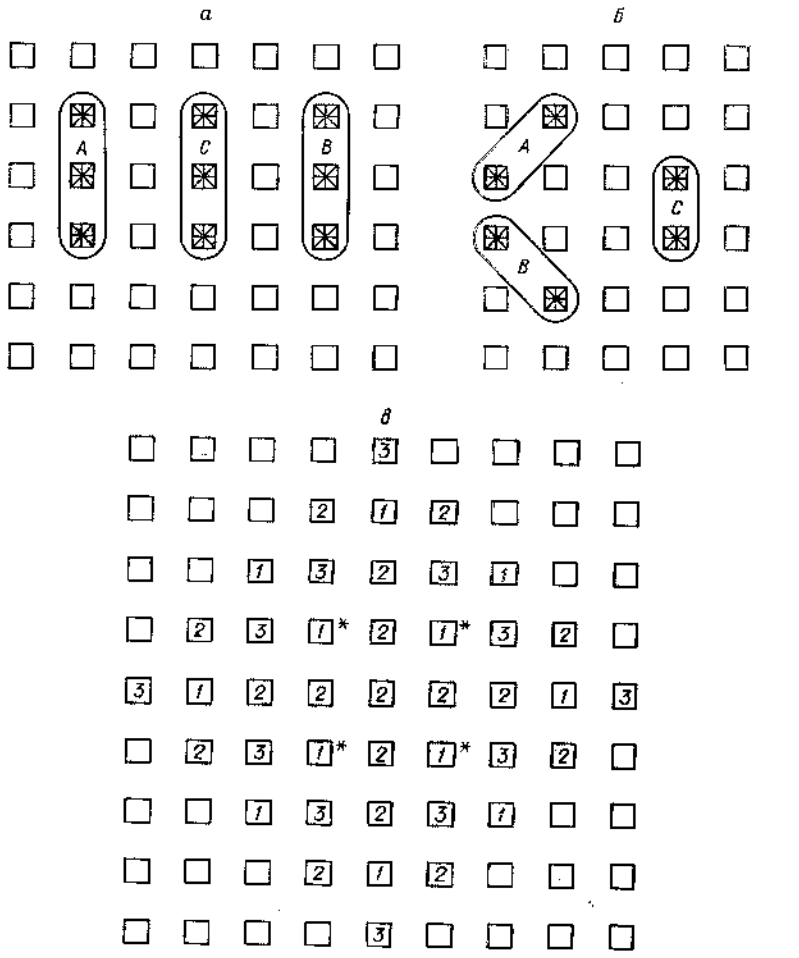


Рис. 38. Реализация функций на среде с порогом два.
а — дизъюнкции $C = A \vee B$; *б* — конъюнкции $C = A \,\&\, B$; *в* — запоминания.

а, например, применить вычислительную среду для построения универсального элемента с возможностью индивидуальной настройки на заданную функцию и заданное направление передачи или приема информации. В этом случае универсальный элемент может быть образован из слоев вычислительной среды и слоев, обеспечивающих одновременное возбуждение элементов среды для выполнения той или иной функции.

Применение вычислительных сред с коллективным поведением элементов позволяет уменьшить требования к технологии микроминиатюризации, так как элемент среды довольно прост и имеет постоянные непереключаемые связи с соседними элементами.

6.4. Основные свойства элементов решетчатых структур

Рассмотрение вычислительных сред с индивидуальным поведением элементов требует более подробного исследования некоторых аспектов построения схем, реализующих функции конечных автоматов с учетом роли соединительных элементов.

В работах [1—3, 22, 23] понятия «схемы» и «логической сети» используются для описания того, каким образом данный сложный автомат построен из функциональных элементов, не подлежащих расчленению. Каждому элементу поставлена в соответствие функция, реализуемая данным элементом. Для входных и выходных каналов элементов определены правила их отождествления при построении схемы или логической сети.

Для уточнения конструкции сложного автомата необходимо задать в некоторой системе координат расположение его функциональных элементов [24]. Далее конструкцию автомата можно уточнить введением элементарных каналов или «соединительных элементов», не подлежащих дальнейшему расчленению, из которых в соответствии с некоторыми правилами можно образовать входные и выходные каналы функциональных элементов [18].

Как уже указывалось ранее, при микроминиатюризации стоимости функциональных и соединительных элементов примерно одинаковы. А так как при построении автоматов соединительных элементов требуется существенно больше, чем функциональных, то при переходе к сложным микроминиатюрным автоматам соединительные элементы начинают играть первостепенную роль.

Задание конструкции сложного автомата в этом случае сводится к заданию схемы (логической сети), дополненной правилами отождествления соединительных элементов и положения соединительных и функциональных элементов в некоторой системе координат. Большой интерес представляет изучение структур, обладающих свойством однородности элементов и соединений между ними.

В качестве математического аппарата при изучении однородной структуры можно использовать аппарат, применяющийся при описании структуры кристаллов [25—27]. Следуя работе

[27], введем понятие целочисленной двумерной и трехмерной решеток.

Будем рассматривать системы параллельных прямых на плоскости, которые разбивают ее на полосы равной ширины. Две такие системы разбивают плоскость на равные параллелограммы и образуют плоскую *решетку*. Вершины этих параллелограммов называются *узлами решетки*, а сами параллелограммы — се *основными параллелограммами*. Аналогично в трехмерном пространстве рассматривают системы параллельных плоскостей, разбивающих пространство на слои равной толщины. Три такие системы параллельных плоскостей разбивают пространство на равные параллелепипеды и образуют трехмерную решетку. Вершины параллелепипедов суть *узлы решетки*, а сами параллелепипеды — *основные параллелепипеды* решетки.

Для двумерной решетки справедливо следующее утверждение: *выпуклые фигуры с центром в узле решетки, заполняющие плоскость без просветов и налеганий, суть или центрально симметрические шестиугольники или параллелограммы.* Эти фигуры получили название *параллелогонов*, так как при заполнении плоскости они располагаются параллельно друг другу. Будем называть эти фигуры в дальнейшем *элементами решетки*. Кроме названных фигур других параллелогонов быть не может. Роль параллелогонов для пространства играют многогранники, называемые *параллелоэдрами*. Существует пять типов параллелоэдров: куб (6-гранник), призма с гексагональным основанием (8-гранник), ромбододекаэдр (12-гранник), вытянутый ромбододекаэдр (12-гранник), кубооктаэдр (14-гранник). Прикладывая друг к другу равные параллелоэдры равными гранями так, чтобы они совпадали друг с другом, можно заполнить пространство без промежутков, причем все параллелоэдры окажутся ориентированными параллельно друг другу.

Элементы перечисленных решеток суть центрально симметрические многоугольники (многогранники). Каждой стороне (грани) центрально симметрического многоугольника (многогранника) отвечает параллельная ей равновеликая сторона (грань) с противоположным направлением внешней нормали. Будем называть такие стороны (грани) *противоположными*, а стороны, имеющие общие вершины (или грани, имеющие общие ребра),— *соседними*.

Двумерные (трехмерные) решетки будем использовать для описания конструкции сложных автоматов. В этом случае каждому параллелогону (параллелоэдру) можно поставить в соответствие или функциональный, или соединительный элемент. Тогда входам и выходам элемента будут соответствовать стороны

многоугольника (грани многогранника). Координаты центра многоугольника (многогранника) будут координатами соединительного или функционального элемента. Задание конструкции сложного автомата можно свести к заданию логической сети [2] из функциональных и соединительных элементов типа решетки и функции, ставящей в соответствие функциональным и соединительным элементам логической сети элементы решетки.

Ниже будут исследоваться вопросы построения схем из соединительных элементов, схем из соединительных и функциональных элементов, схем из элементарных автоматов и соединительных элементов, схем из элементов вычислительной среды.

6.5. Схемы из соединительных элементов

Для определения схемы из соединительных элементов нам потребуются некоторые понятия из теории графов [28].

Говорят, что дан граф, если даны:

а) непустое множество X;

б) отображение Γ множества X в X.

Будем изображать элементы множества X точками плоскости, а пары точек x и y, для которых $y \in \Gamma x$, соединять непрерывной линией со стрелкой, направленной от x к y.

Каждый элемент множества X называется *вершиной* графа, а пара элементов (x, y), в которой $y \in \Gamma x$ — *дугой* графа. Множество дуг графа будем обозначать через V. Граф (X, V) называется *симметрическим*, если $(x, y) \in V \Rightarrow (y, x) \in V$. Граф (X, V) называется *антисимметрическим*, если $(x, y) \in V \Rightarrow (y, x) \notin V$.

Путем в графе (X, V) называется такая последовательность дуг, что конец каждой предыдущей дуги совпадает с началом следующей.

Пусть дан граф $G = (X, V)$ и пусть $x_1, \ldots, x_N$ — его вершины. Обозначим через b_j^i число дуг V, идущих из x_i в x_j.

Квадратная матрица (b_j^i) с N строками и N столбцами называется *матрицей смежности* графа, где b_j^i — элемент, стоящий на пересечении i-й строки и j-го столбца. В дальнейшем рассмотрении b_j^i обычно принимает значения 0, 1.

Рассмотрим графы $T=(X, U)$, у которого множество вершин X можно представить в виде $X=\{X_1, X_1', X_2, X_2', \ldots, X_s, X_s'\}$, где X_i и X_i' — подмножества, содержащие одинаковое количество вершин: $x_{i_1}, \ldots, x_{ik_i}$ и $x'_{i_1}, \ldots, x'_{ik_i}$. Такие подмножества будем называть подмножествами противоположных вершин,

а вершины $x_{ij} \in X_i$ и $x'_{ij} \in X_i'$ — противоположными вершинами. Общее количество вершин множества X будет $K = 2(k_1 + k_2 + \ldots + k_i + \ldots + k_s)$.

Пусть дан элемент E решетки с числом сторон (или граней) n, где n принимает значения 4, 6 (или 6, 8, 12, 14).

Обозначим стороны (грани) элемента E решетки следующим образом. Пусть некоторая сторона (грань) элемента решетки будет обозначена a_1, тогда параллельную ей равновеликую сторону (грань) с противоположным направлением внешней нормали обозначим a_1'.

Выберем из оставшихся сторон (граней) элемента любую сторону (грань) и обозначим ее через a_2, а параллельную ей с противоположным направлением внешней нормали сторону (грань) обозначим через a_2'. Процесс продолжаем до тех пор, пока не обозначим все n сторон (граней). Каждую пару сторон (граней) a_i, a_i' разобьем на k_i равных частей: $a_{i_1}, a_{i_1}', \ldots, a_{ik_i}, a_{ik_i}'$, которые будем называть полюсами. Такие элементы решетки с выделенными полюсами назовем Λ-*элементами*. Распространим эти обозначения на все Λ-элементы данной решетки.

Поставим в соответствие Λ-элементу граф T с $2s = n$ подмножествами вершин и пусть a_i стороне (грани) элемента соответствует подмножество вершин X_i, а a_i' — подмножество X_i' $(i = 1, \ldots, s)$. Каждой из вершин x_{ij} подмножества X_i поставим во взаимно однозначное соответствие a_{ij} $(j = 1, \ldots, k_i)$. Соединим пары вершин x, $y \in X$ дугами $(x, y) \in U$.

Граф, поставленный в соответствие элементу решетки, называется *графом, реализуемым этим элементом*.

Λ-Элемент с поставленным ему в соответствие графом будем называть *соединительным элементом*; его полюсы, которым поставлены в соответствие вершины графа T, будем называть *полюсами* соединительного элемента. Λ-Элемент с поставленным ему в соответствие симметрическим (антисимметрическим) графом будем называть *симметрическим* (*антисимметрическим*) *соединительным элементом*. Полюсы, соответствующие противоположным частям сторон (граней) Λ-элемента, будем называть *противоположными*.

Два Λ-элемента с общей стороной (гранью) назовем *соседними*. У них будем считать отождествленными полюсы, соответствующие совпадающим частям сторон.

Определим *схему из Λ-элементов*.

1. Λ-Элемент есть схема.

2. Две схемы, у которых хотя бы для одного Λ-элемента имеется соседний Λ-элемент из другой схемы, есть схема.

Неотождествленные полюсы Λ-элементов суть *полюсы схемы*.

Схему, в которой все Λ-элементы суть соединительные, назовем *схемой из соединительных элементов*.

Схемы из соединительных элементов можно классифицировать в зависимости от размерности решетки (двумерные и трехмерные); от количества сторон в элементе для двумерной решетки (4 и 6-сторонние) или от количества граней в элементе трехмерной решетки (6, 8, 12, 14-гранники); от количества вершин графа, соответствующих данной стороне (грани) элемента решетки; от типа графа (симметрические и антисимметрические).

К наиболее простым соединительным элементам относятся те, которые получены из элементов двумерной решетки — параллелограммов. Им поставлены в соответствие графы с общим числом вершин — 4, так что каждой стороне соответствует одна вершина. Заметим, что 4-полюсные элементы могут быть получены и из других типов элементов путем соответствующего выбора четырех полюсов.

Теорема 1. *Всякий конечный граф может быть реализован схемой из четырехполюсных соединительных элементов.*

Пусть дан произвольный граф G с вершинами $x_1, x_2, \ldots, x_N \in X$ и дугами $v_1, \ldots, v_M \in V$.

В решетке из 4-полюсных соединительных элементов, у которых соединены полюсы a_1 с a_1' и a_1' с a_1, выделим N строк элементов и каждой из них поставим во взаимно однозначное соответствие одну из вершин графа G. В результате множество вершин графа X будет взаимно однозначно отображено на множество C строк решетки.

Покажем теперь, что любой дуге графа G (x_k, x_l) может быть поставлен во взаимно однозначное соответствие путь из строки решетки c_k к строке c_l $(c_k, c_l \in C)$.

Пусть для определенности $k < l$.

Рассмотрим три возможных случая (рис. 39).

1. (x_k, x_l). Нетрудно видеть, что если у элементов какого-либо столбца i, принадлежащих строкам c_k и c_l, соединить полюсы a_2' и a_2 с a_1 и a_1', а у всех остальных элементов этого же столбца соединить полюс a_2' с a_2 (и только с ним), то строка c_k окажется соединенной путем со строкой c_l и никаких других путей между этими строками и какими-либо другими не возникает, т. е. проложенный путь может быть поставлен во взаимно однозначное соответствие дуге (x_k, x_l).

2. Случай (x_l, x_k) аналогичен предыдущему с единственным отличием, что у всех элементов j-го столбца, не принадлежащих строкам c_k и c_l, полюс a_2 будет соединен с полюсом a_2'.

3. В случае (x_k, x_k), когда данная вершина x_k является и входом и выходом дуги (x_k, x_k) (петля), ее взаимно однозначное отображение на решетке может быть осуществлено, если у элементов в столбцах u и $(u+1)$, принадлежащих строке c_k

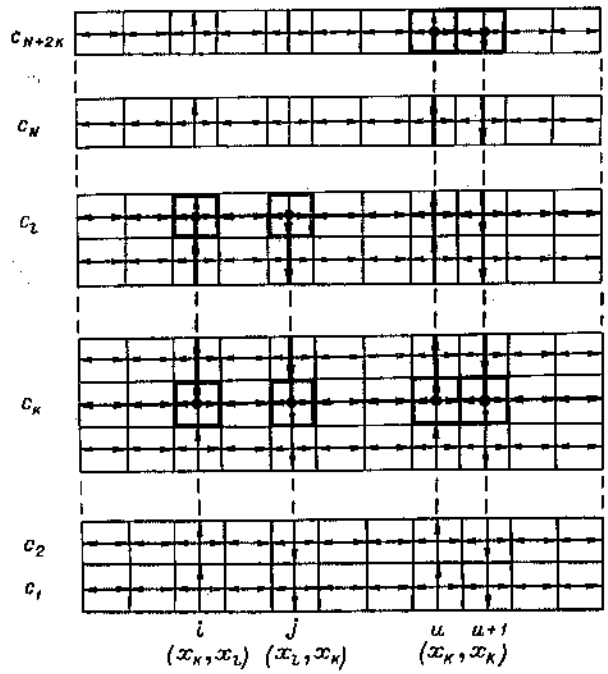


Рис. 39. Реализация графа с помощью решетки соединительных элементов.

Обозначение полюсов 

и некоторой строке $c_{N+2k} \in C$, полюсы a_2 и a_2' соединить с полюсами a_1 и a_1', а у всех остальных элементов u-го столбца соединить полюс a_2' с полюсом a_2, а $(u+1)$-го столбца — полюс a_2 с a_2'. При этом образуется замкнутый путь, соответствующий дуге (x_k, x_k), и не возникает никаких других соединений между строками множества C. Таким образом, отводя для каждой дуги типа $(x_k, x_l)_{k \neq l}$ столбец элементов, а каждой дуге типа (x_k, x_k) — два столбца элементов и одну дополнительную строку, с помощью конечного числа элементов можно реализовать

произвольный конечный граф G, что и требовалось доказать.

Из способа доказательства теоремы 1 вытекают следствия.

Следствие 1. *Для реализации любого графа G_i с N_i вершинами, M_i дугами, R_i из которых являются петлями вида (x_k, x_k), достаточно выделить часть решетки из 4-полюсных элементов размером $N_i + R_i$ строк и $M_i + R_i$ столбцов.* Эта часть решетки может быть изолирована от остальной решетки с помощью элементов с изолированными полюсами, матрицы смежности которых будут иметь вид:

$$0 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Тогда вся остальная часть решетки может быть использована для размещения схем, реализующих другие графы.

Следствие 2. При доказательстве теоремы 1 были использованы соединительные элементы с матрицами смежности:

$$P = \begin{array}{c} \\ a_1 \\ a_1' \\ a_2 \\ a_2' \end{array} \begin{array}{c} \begin{array}{cccc} a_1 & a_1' & a_2 & a_2' \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{array} \qquad D' = \begin{array}{c} \\ a_1 \\ a_1' \\ a_2 \\ a_2' \end{array} \begin{array}{c} \begin{array}{cccc} a_1 & a_1' & a_2 & a_2' \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \qquad D'' = \begin{array}{c} \\ a_1 \\ a_1' \\ a_2 \\ a_2' \end{array} \begin{array}{c} \begin{array}{cccc} a_1 & a_1' & a_2 & a_2' \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Нетрудно видеть, что если все элементы решетки, не входящие в поле размером $(N + R)(M + R)$, будут иметь соединения типа D' и D'', то никаких новых путей между строками решетки не появится, откуда следует, что *любой граф может быть реализован набором из трех видов 4-полюсных элементов.*

Следствие 3. *Любое конечное число конечных графов может быть реализовано решеткой из трех видов 4-полюсных соединительных элементов.*

Из теоремы 1 и следствий 1 и 2 вытекает, что любой граф G_i можно реализовать, используя $N_i + R_i$ строк элементов и $M_i + R_i$ столбцов, причем от элементов строк и столбцов, не лежащих в поле S_i из $N_i + R_i$ строк и $M_i + R_i$ столбцов, требуется, чтобы они имели соединение типа D' или D''. Если поле S_{i+1}, реализующее граф G_{i+1}, расположить по диагонали от S_i так, что у него не будет общих строк и столбцов с полем S_i, то оба графа могут быть реализованы независимо друг от друга (рис. 40). Аналогично могут быть выбраны в направлении диагонали поля для всех других графов.

Следствие 4. *Любой симметрический граф может быть реализован решеткой из двух видов 4-полюсных симметрических соединительных элементов.*

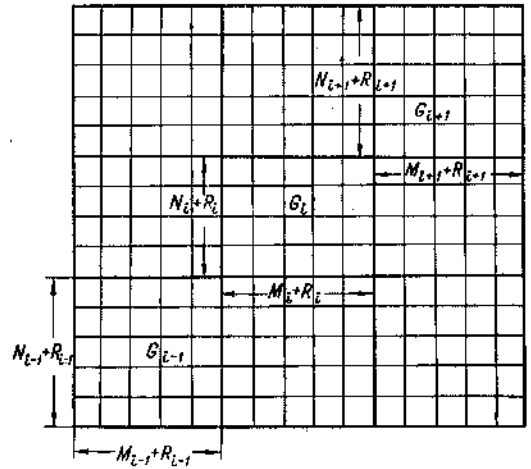


Рис. 40. Реализация произвольного числа графов с помощью решетки соединительных элементов.

По определению симметрического графа каждой дуге (x_k, x_l) должна соответствовать дуга (x_l, x_k). Это будет выполнено, если вместо матриц смежности D' и D'' взять матрицу смежности

$$D = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Из следствий 3 и 4 непоседственно вытекает следствие 5.

Следствие 5. Так как 4-полюсный соединительный элемент может быть получен из K-полюсного изоляцией пар противоположных всех избыточных полюсов, то *теорема 1 и ее следствия могут быть распространены на K-полюсные элементы.*

Систему соединительных элементов $S_1, \ldots, S_k$ будем называть *полной*, если любой конечный граф может быть реализован схемой из этих элементов.

Теорема 2. *Минимальная полная система 4-полюсных симметрических соединительных элементов для отображения симметрических графов содержит два элемента.*

Достаточность вытекает из следствия 4 теоремы 1. Необходимость следует из того, что схема, образованная многократным повторением одного и того же элемента, может соответствовать только одному определенному типу графа. Действительно, рассмотрим, например, две группы случаев: 1) когда ни один полюс соединительного элемента не изолирован и хотя бы один полюс соединен не менее чем с двумя другими; 2) все остальные случаи.

В первой группе все полюсы всех элементов соединены друг с другом путем, т. е. реализуется один-единственный полный граф и только он. Во второй группе в схеме будут отсутствовать разветвления, т. е. нельзя реализовать граф типа дерева.

6.6. Схемы из соединительных и функциональных элементов

Пусть дана система функций алгебры логики

$$\left.\begin{aligned}
z_{11} &= f_{11}(x_{11}, \ldots, x_{1k_1}, \ldots, x_{s1}, \ldots, x_{sk_s});\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
z_{1k_1} &= f_{1k_1}(x_{11}, \ldots, x_{1k_1}, \ldots, x_{s1}, \ldots, x_{sk_s});\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
z_{ik_i} &= f_{ik_i}(x_{11}, \ldots, x_{1k_1}, \ldots, x_{s1}, \ldots, x_{sk_s});\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
z_{s1} &= f_{s1}(x_{11}, \ldots, x_{1k_1}, \ldots, x_{s1}, \ldots, x_{sk_s});\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
z_{sk_s} &= f_{sk_s}(x_{11}, \ldots, x_{1k_1}, \ldots, x_{s1}, \ldots, x_{sk_s}).
\end{aligned}\right\} \qquad (6.3)$$

Поставим в соответствие полюсу a_{ij} Λ-элемента переменную x_{ij}, а полюсу a'_{ij} — функцию z_{ij}.

Λ-Элемент с поставленной ему в соответствие системой функций алгебры логики (6.3) будем называть *функциональным элементом*, или *истинностным автоматом*, а полюсы a_{ij} и a'_{ij} с поставленными им в соответствие переменными x_{ij} и функциями z_{ij} — его *входными и выходными полюсами*.

Множество функциональных элементов будем называть *полным*, если соответствующие им функции алгебры логики образуют функционально полную систему, т.е. если с их помощью может быть построена любая функция алгебры логики.

Схему, в которой Λ-элементы суть либо соединительные, либо функциональные, назовем *схемой из соединительных и функциональных элементов*.

Теорема 3. *Каждая функция алгебры логики может быть реализована схемой из трех типов функциональных и двух типов симметрических соединительных 4-полюсных элементов.*

Пусть даны функциональные элементы F_1, F_2 и F_3, реализующие функции алгебры логики соответственно:

$$\begin{aligned}
&F_1 \quad z_1 = \bar{x}_1;\ z_2 = \bar{x}_2;\\
&F_2 \quad z_1 = z_2 = x_1 \,\&\, x_2;\\
&F_3 \quad z_1 = z_2 = x_1 \vee x_2
\end{aligned} \qquad (6.4)$$

и симметрические соединительные 4-полюсные элементы P и D.

Из алгебры логики известно, что любая функция $f(x_1, \ldots, x_m)$, принимающая два значения 0 и 1 и определенная для всевозможных значений аргументов, принимающих также два значения 0 и 1, может быть выражена в совершенной дизъюнктивной нормальной форме (с.д.н.ф.)

$$f(x_1, \ldots, x_m) = \bigvee_{k=1}^{N} x_1^{\sigma_1^k} x_2^{\sigma_2^k}, \ldots, x_m^{\sigma_m^k}. \qquad (6.5)$$

В формуле (6.5) логическое суммирование ведется по тем наборам $\sigma_1^k, \sigma_2^k, \ldots, \sigma_m^k$, для которых $f(\sigma_1^k, \sigma_2^k, \ldots, \sigma_m^k) = 1$ ($x_i^{\sigma_i^k} = \bar{x}_i$ при $\sigma_i^k = 0$, $x_i^{\sigma_i^k} = x_i$ при $\sigma_i^k = 1$). Индекс набора $k = 1, \ldots, N = 2^m$.

Схема, реализующая функцию $f(x_1, \ldots, x_m)$, может быть построена из четырех блоков A, B, C и G (рис. 41).

1. Входы блока A есть переменные $x_1, x_2, \ldots, x_m$, а выходы — N наборов тех же переменных. Блок реализуется в виде решетки, у которой m столбцов и Nm строк. Каждый входной полюс x_i соединен путем только со всеми x_i выходными полюсами.

2. Блок B реализует системы наборов $x_1^{\sigma_1^k}, x_2^{\sigma_2^k}, \ldots, x_m^{\sigma_m^k}$. Входы блока B отождествлены с выходами блока A, а выходы суть наборы переменных $x_1^{\sigma_1^k}, x_2^{\sigma_2^k}, \ldots, x_m^{\sigma_m^k}$. Блок представляет собой столбец решетки, состоящий из элементов F_1 и D общим числом Nm.

3. Блок C реализует все конъюнкции вида $x_1^{\sigma_1^k} x_2^{\sigma_2^k} \ldots x_m^{\sigma_m^k}$. Входы блока есть выходы блока B, а выходы — конъюнкции

от m переменных. Блок представляет собой решетку из m столбцов и mN строк, в которой используются элементы F_2, P и D. Выходами конъюнкций служат выходы элементов F_2, соответствующие переменным x_m.

4. Блок G реализует дизъюнкцию всех конъюнкций. Входами блока являются выходы блока C. Блок представляет собой столбец решетки, состоящий из элементов F_3, P, D. Выход элемента столбца, т. е. выход последней конъюнкции, в N-м наборе, соответствует функции $f(x_1, \ldots, x_m)$. Теорема доказана.

Систему соединительных элементов будем называть *коммутационно полной*, если с ее помощью можно реализовать произвольную заданную матрицу соединений полюсов любого конечного числа Λ-элементов хотя бы для одного варианта их размещения в решетке.

Теорема 4. *Для каждого типа Λ-элементов существует коммутационно полная система соединительных элементов.*

Доказательство теоремы громоздко, поэтому укажем только его идею. Разместим Λ-элементы так, чтобы было не более одного Λ-элемента в строке и между этими строками было K строк без Λ-элементов. Под строкой понимается последовательность элементов решетки вдоль нормали к их общим сторонам. Всего в решетке $n/2$ семейств параллельных строк.

Произвольность размещения Λ-элементов в решетке следует из определения коммутационной полноты. Поставим в соответствие каждому из K полюсов по строке из каждого семейства и так заполним решетку соединительными элементами, чтобы элементы строк, не содержащие Λ-элементов, соединились путем друг с другом и с одним и только с одним из полюсов Λ-элемента. Например, для 4-полюсного Λ-элемента с координатами (α, β) достаточно поместить в элементы решетки $(\alpha, \beta \pm 1)$, $(\alpha \pm 1, \beta \mp 1)$, $(\alpha \pm 2, \beta \mp 2)$, $(\alpha \pm 2, \beta)$ соединительный элемент P, а во все остальные — D. После этого для реализации матрицы соединений достаточно соединить путем соответствующие пары строк. Для 4-полюсного Λ-элемента это достигается заменой элемента D на P в местах пересечения соответствующих строк.

Следствие 1. *В решетке может быть реализована любая схема из функциональных и соединительных элементов.*

Следствие 2. *Полная система функциональных и коммутационно полная система соединительных элементов образуют полную систему функциональных и соединительных элементов, с помощью которой реализуется любая функция алгебры логики.*

Теорема 5. *Минимальная коммутационно полная система 4-полюсных симметрических соединительных элементов состоит из двух элементов.* Доказательство теоремы очевидно.

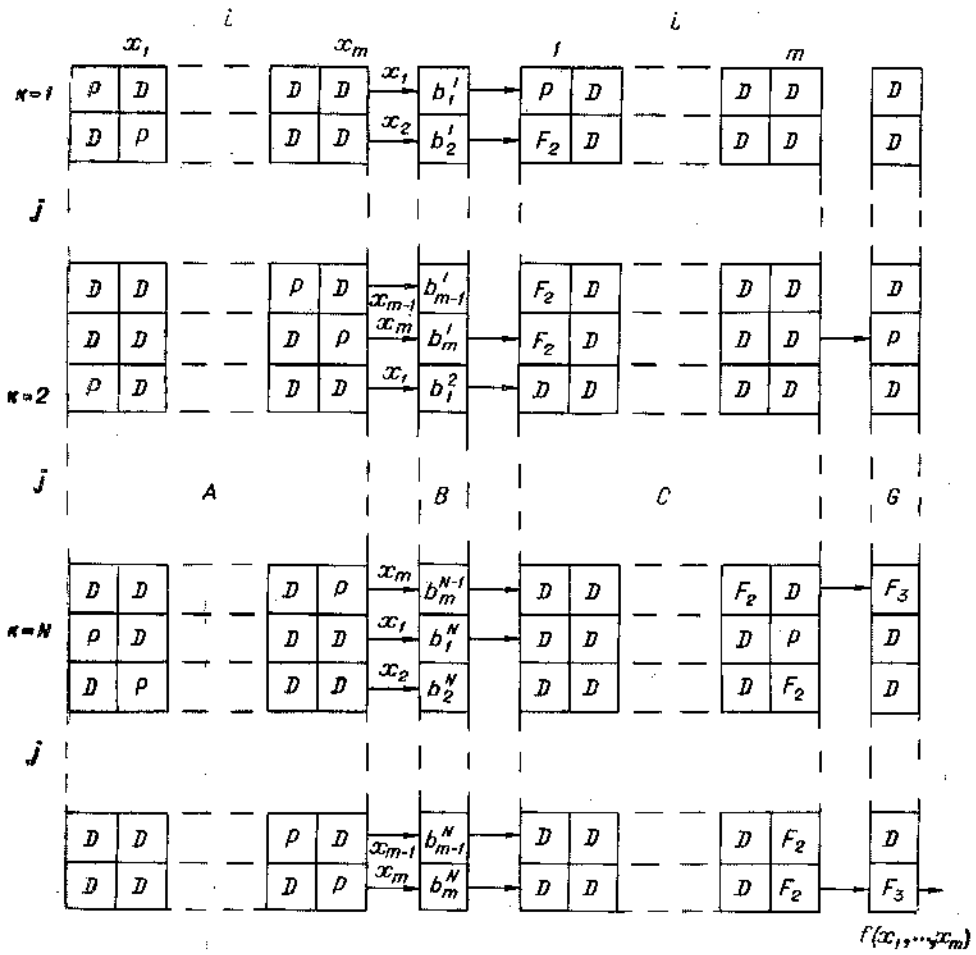


Рис. 41. Схема реализации булевой функции от m переменных.

$$a_{ij}^k = \begin{cases} D \text{ при } i=j; \\ P \text{ при } i \neq j; \end{cases} \qquad b_j^k = \begin{cases} F_1 \text{ при } \sigma_j^k = 0; \\ D \text{ при } \sigma_j^k = 1; \end{cases}$$

$$c_{ij}^k = \begin{cases} P \text{ при } i=k,\ j=1; \\ F_2 \text{ при } i=k,\ j>1; \\ D \text{ при } i \neq k; \end{cases} \qquad g_j^k = \begin{cases} P \text{ при } j=m,\ k=1; \\ F_3 \text{ при } j=m,\ k>1; \\ D \text{ при } j \neq m \end{cases}$$

$(k = 1,2,\ldots, N)\ (i, j = 1,2,\ldots, m)$.

Как уже упоминалось, при переходе к микроминиатюризации стоимости функциональных и соединительных элементов становятся одинаковыми. Следовательно, при синтезе схем из функциональных и соединительных элементов следует учитывать возросшую роль соединительных элементов.

О п р е д е л е н и е. *Сложностью* схемы S из функциональных и соединительных элементов будем называть общее число ее элементов.

Пусть $L_\Phi(f)$ — наименьшая из сложностей схем, реализующих функцию f (сложность простейшей схемы для функции f). Пусть $L_\Phi(m) = \max L_\Phi(f_i)$. Максимум берется по всем функциям $f_i(x_1, \ldots, x_m)$ от m аргументов, т. е. $L_\Phi(m)$ есть наименьшее число элементов, достаточное для реализации любой функции от m аргументов $x_1, \ldots, x_m$. Рассмотрим некоторые методы синтеза схем из функциональных и соединительных элементов, реализующих функции алгебры логики $f(x_1, \ldots, x_m)$, и оценим сложность этих схем при $m \to \infty$, как для обычных схем [29, 30].

1. Моделирование с.д.н.ф. (6.5). Схема реализации функции $f(x_1, \ldots, x_m)$ через ее с.д.н.ф. представлена на рис. 42.

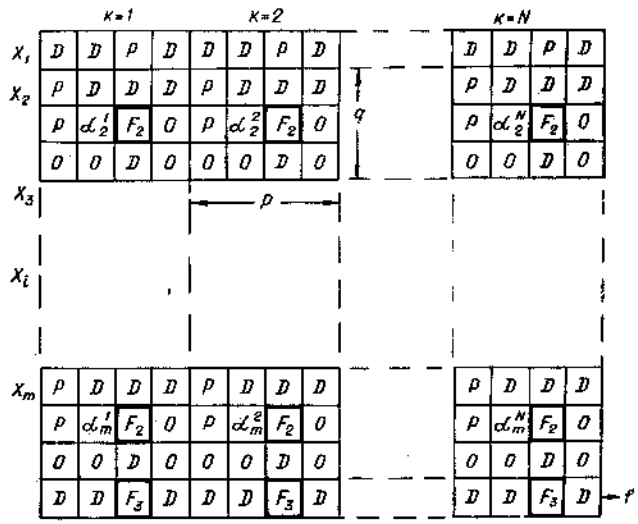


Рис. 42. Схема реализации с.д.н.ф. функции от m переменных.

$$\alpha_i^k = \begin{cases} F_1 \text{ при } \sigma_i^k = 0; \\ D \text{ при } \sigma_i^k = 1 \end{cases}$$

$(k = 1, 2, \ldots, N)$
$(i = 1, 2, \ldots, m)$.

Для простоты рассматривается случай реализации $f(x_1, \ldots, x_m)$ схемой из 4-полюсных элементов: F_1, F_2, F_3, P, D и O. Элемент α_i^k заменяется либо на F_1 при $\sigma_i^k = 0$, либо на D при $\sigma_i^k = 1$:

$$L_1(f) < pq\,(m-1)\,N + 2p\,N, \tag{6.6}$$

где p, q — количество элементов в стандартном блоке; N не превышает 2^m — числа всех наборов из m аргументов.

При $m \to \infty$ справедливо неравенство

$$L_1(f) < pqm2^m\,[1 + o(1)]. \tag{6.7}$$

Эта оценка может быть улучшена, если учесть, что либо сама функция f, либо ее отрицание $\bar{f}$ имеет с.д.н.ф. с числом конъюнктивных членов $\leqslant \frac{2^m}{2}$. Моделирование $\bar{f}$ или f с меньшим числом членов с.д.н.ф. и взятие отрицания снижает оценку (6.7) вдвое

$$L_1(f) < \frac{pqm2^m}{2}\,[1 + o(1)]. \tag{6.8}$$

2. В первом методе основное количество элементов приходится на формирование всех конъюнкций вида $x_1^{\sigma_1^k} \ldots x_m^{\sigma_m^k}$.

Это количество может быть уменьшено, если каждую конъюнкцию представить в виде

$$x_1^{\sigma_1^k} \ldots x_m^{\sigma_m^k} = (x_1^{\sigma_1^k} \ldots x_{\frac{m}{2}}^{\sigma_{\frac{m}{2}}^k})(x_{\frac{m}{2}+1}^{\sigma_{\frac{m}{2}+1}^k} \ldots x_m^{\sigma_m^k}) \tag{6.9}$$

и реализовать каждое выражение в скобках отдельной схемой. Тогда затраты элементов на эти схемы не превысят $pqm2^{m/2}$.

Для получения полных конъюнкций понадобится еще не более чем $pq2^m$ элементов (рис. 43). При реализации функции $f(x_1, \ldots, x_m)$ таким способом

$$L_2(f) < pqm2^{m/2} + pq2^m \tag{6.10}$$

и при $m \to \infty$

$$L_2(f) < pq2^m[1 + o(1)]. \tag{6.11}$$

На рис. 43 обозначения те же, что и в пункте 1. Элемент β_k заменяется либо на F_2 для случая, когда $f(\sigma_1^k, \ldots, \sigma_m^k) = 1$, либо на 0, когда $f(\sigma_1^k, \ldots, \sigma_m^k) = 0$.

Методы Шеннона и Лупанова для оценки сложности схем существенно используют следующие положения:

элементы, реализующие функции алгебры логики, могут соединяться каналами неограниченной длины;

один и тот же выход элемента может подключаться к неограниченному количеству входов других элементов.

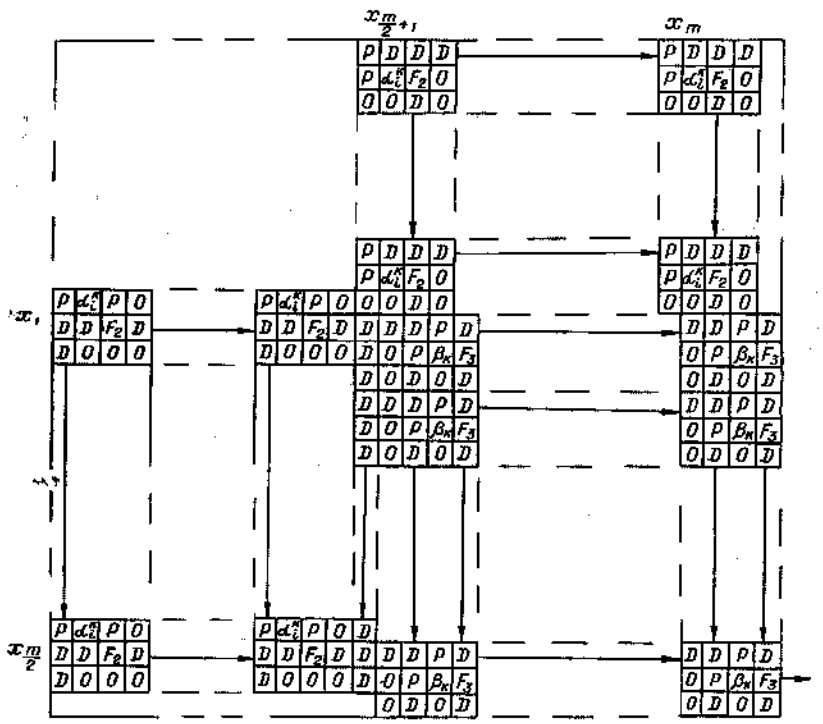


Рис. 43. Схема реализации булевой функции от m переменных:

$$\alpha_i^k = \begin{cases} D \text{ при } \sigma_i^k = 1; \\ F_1 \text{ при } \sigma_i^k = 0; \end{cases} \qquad \beta_k = \begin{cases} F_2 \text{ при } f(\sigma_1^k, \ldots, \sigma_m^k) = 1; \\ 0 \text{ при } f(\sigma_1^k, \ldots, \sigma_m^k) = 0. \end{cases}$$

В нашем случае ни одно из указанных положений не справедливо. Это не позволяет применять их оценки для схем из функциональных и соединительных элементов.

Можно высказать предположение, что оценка сложности схем из функциональных и соединительных элементов

$$L(f) < C2^m[1 + o(1)] \tag{6.12}$$

не может быть существенно улучшена.

6.7. Схемы из элементарных автоматов и соединительных элементов

В этом параграфе потребуются дополнительные сведения из теории цифровых автоматов [1—3]. Абстрактным автоматом называется объект, способный принимать различные состояния из некоторого фиксированного множества (множества внутренних состояний автомата), переходить из одного состояния в другое под воздействием сигналов из некоторого фиксированного конечного множества (множества входных сигналов, называемого также входным алфавитом автомата) и выдавать выходные сигналы из некоторого фиксированного конечного множества выходных сигналов (выходного алфавита автомата).

Абстрактный автомат определяется заданием трех множеств (множества внутренних состояний $\mathfrak{A}$, входного алфавита $\mathfrak{X}$, выходного алфавита $\mathfrak{B}$), элемента a_1 множества $\mathfrak{A}$ (начального состояния) и двух функций (функций переходов φ и функции выходов ψ), определяющих функционирование автомата в дискретном времени.

Функция переходов определяет состояние $a(t+1)$ автомата в момент $t+1$ в зависимости от его состояния $a(t)$ и входного сигнала $x(t)$ в момент t:

$$a(t+1) = \varphi[a(t), x(t)]. \tag{6.13}$$

Функция выходов определяет зависимость выходного сигнала $y(t)$ от состояния автомата и входного сигнала в тот же самый момент t:

$$y(t) = \psi[a(t), x(t)]. \tag{6.14}$$

Абстрактный автомат, у которого множество $\mathfrak{A}$ внутренних состояний конечно, называется *конечным автоматом*. При изучении конструкции автомата используется понятие схемы, которая предназначена для описания того, каким образом данный конечный автомат построен из *простейших автоматов*, не подлежащих дальнейшему расчленению, или просто *элементов*.

Каждый элемент определяется заданием:

1) ячейки, употребляющей внутренний алфавит Q;

2) m входных каналов, употребляющих входные алфавиты X_m $(m = 0, 1, 2, \ldots)$;

3) h выходных каналов, употребляющих выходные алфавиты Z_h $(h = 0, 1, 2, \ldots)$;

4) системы канонических уравнений:

$$\begin{aligned} z_1(t) &= \Phi_1[x_1(t), x_2(t), \ldots, x_m(t), q(t)];\\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ z_h(t) &= \Phi_h[x_1(t), x_2(t), \ldots, x_m(t), q(t)];\\ q(t+1) &= \psi[x_1(t), x_2(t), \ldots, x_m(t), q(t)]. \end{aligned} \tag{6.15}$$

Функции $x_l(t)$, $z_j(t)$, $q(t)$ принимают значения соответственно из алфавитов X_m, Z_h и Q, $t = 0, 1, 2, \ldots$

Концы каналов, не примыкающие к ячейке, называются полюсами элемента (соответственно входными и выходными).

Схемой над множеством элементов $\{\mathfrak{M}_i\}$ называется любой набор элементов из $\{\mathfrak{M}_i\}$, для которых указано, какие их полюсы считаются отождествленными, причем отождествлять можно лишь такие полюсы, к которым отнесен один и тот же алфавит.

Представляет интерес изучение класса схем, так называемых логических сетей, определяемых следующим образом:

1) всякий элемент из данного множества элементов $\{\mathfrak{M}_i\}$ является логической сетью над $\{\mathfrak{M}_i\}$;

2) совокупность двух логических сетей без общих полюсов есть логическая сеть;

3) в результате отождествления двух входных полюсов логической сети получается логическая сеть;

4) в результате отождествления выходного полюса z_i логической сети с таким ее входным полюсом x_l, которому z_i не подчинен, получается логическая сеть;

5) других логических сетей не существует.

Неподчиненность полюса z_i полюсу x_j означает, что выход $z_i(t)$ не зависит от входа $x_j(t)$.

Оператор $g(t) = \theta[f(t)]$ называется детерминированным оператором, если при любом t буква $g(t)$ — однозначная функция от слова

$$f(1)f(2)\ldots f(t-1)f(t).$$

Детерминированный оператор называется ограниченно детерминированным, если он принимает конечное число значений при неограниченном увеличении t. Понятие конечного автомата определяется как совокупность понятий логической сети и ограниченно детерминированного оператора.

Для любого ограниченно детерминированного оператора θ существует логическая сеть, реализующая θ. Любая логическая сеть может быть построена при наличии элементов, реализующих полную систему функций алгебры логики и элемента задержки. Другими словами, любой конечный автомат может быть задан с помощью ограниченно детерминированного оператора θ и логической сети, реализующей θ.

Нас будут интересовать вопросы реализации логической сети из простейших автоматов и соединительных элементов.

Пусть дан простейший конечный автомат, входным полюсам которого приписаны функции $x_{ij}(t)$, а выходным — $z_{ij}(t)$ ($i = 1, \ldots, s = n/2$; $j = 1, \ldots, k_i'$). Поставим в соответствие каждому полюсу a_{ij} А-элемента входной полюс автомата с функцией $x_{ij}(t)$, а полюсу a_{ij}' — выходной полюс с функцией $z_{ij}(t)$. А-Элемент с поставленным ему в соответствие простейшим конечным автоматом будем называть *элементарным автоматом*, а полюсы a_{ij} и a_{ij}' с поставленными им в соответствие функциями $x_{ij}(t)$ и $z_{ij}(t)$ — его *входными и выходными полюсами*.

Схему, в которой все А-элементы — элементарные автоматы или соединительные элементы, будем называть *схемой из элементарных автоматов и соединительных элементов*.

Множество элементарных автоматов будем называть *полным*, если соответствующие им конечные автоматы образуют автоматно полную систему, т. е. если с их помощью может быть реализован любой конечный автомат.

Из теории автоматов известно, что автоматы, реализующие функционально полную систему функций алгебры логики от двух переменных и задержку на один такт, образуют автоматно полную систему. Поэтому соответствующее им множество элементарных автоматов будет полным.

Канонические уравнения, соответствующие элементарному автомату, реализующему функцию задержки, имеют вид:

$$\left.\begin{aligned} &z_{11}(t) = q_{11}(t) \qquad q_{11}(t+1) = x_{11}(t);\\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ &z_{1k_1}(t) = q_{1k_1}(t) \qquad q_{1k_1}(t+1) = x_{1k_1}(t);\\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ &z_{i1}(t) = q_{i1}(t) \qquad q_{i1}(t+1) = x_{i1}(t);\\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ &z_{ik_i}(t) = q_{ik_i}(t) \qquad q_{ik_i}(t+1) = x_{ik_i}(t);\\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ &z_{s1}(t) = q_{s1}(t) \qquad q_{s1}(t+1) = x_{s1}(t);\\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ &z_{sk_s}(t) = q_{sk_s}(t) \qquad q_{sk_s}(t+1) = x_{sk_s}(t). \end{aligned}\right\} \tag{6.16}$$

Теорема 6. *Любой конечный автомат может быть реализован схемой из 4-полюсных элементарных автоматов и соединительных элементов.*

Пусть даны симметрические соединительные элементы P, D, элементарные (истинностные автоматы F_1, F_2, F_3) (6.4) и 4-полюсный элементарный автомат M_0, реализующий функцию задержки

$$z_1(t) = q_1(t);\ q_1(t+1) = x_1(t);$$
$$z_2(t) = q_2(t);\ q_2(t+1) = x_2(t).$$

Причем функции $x_1(t)$ и $x_2(t)$ соответствуют входным полюсам автомата, совмещенных с полюсами А-элемента a_1 и a_2, а функции $z_1(t)$ и $z_2(t)$ — выходным полюсам, совмещенных с полюсами a_1' и a_2'.

Справедливость теоремы следует из того, что элементарные автоматы M_0, F_1, F_2 и F_3 образуют автоматно полную систему, а соединительные элементы P, D — коммутационно полную симметрическую систему.

Покажем, что автоматно полная система может содержать только один элементарный автомат. Действительно, функционально полная система может состоять из одного истинностного автомата, реализующего функцию штрих Шеффера или стрелку Пирса.

При реализации функции от двух переменных достаточно трех полюсов, что позволяет использовать четвертый полюс в качестве выходного полюса автомата M_0. Такой автомат будет реализовать следующие функции:

$$z_1(t) = \bar{x}_1(t) \vee \bar{x}_2(t); \qquad (6.17)$$
$$z_2(t) = q(t);\ q(t+1) = z_1(t).$$

Аналогично элементарный K-полюсный автомат будет реализовать функции

$$z_1(t) = z_2(t) = \ldots = z_{K/2}(t) = \bar{x}_1(t) \vee \bar{x}_2(t) \ldots \vee \bar{x}_{K/2-1}(t); \qquad (6.18)$$
$$z_{K/2}(t) = q(t);\ q(t+1) = z_1(t).$$

Так как коммутационно полная симметрическая система 4-полюсных соединительных симметрических элементов может содержать два элемента, то очевидно следующее утверждение.

Теорема 7. *Полная система 4-полюсных элементарных автоматов и соединительных элементов может содержать один элементарный автомат и два симметрических соединительных элемента.*

6.8. Схемы из элементов вычислительной среды

В предыдущих параграфах исследовались схемы с фиксированной структурой.

Представляет большой интерес исследование схем, построенных на основе вычислительных сред с переменной структурой. Эта вычислительная среда состоит из универсальных элементов, каждый из которых реализует полную систему функций алгебры логики, задержку сигнала и полную систему функций соединений с другими элементами и обладает возможностью настройки элементов на выполнение указанных функций.

Дадим формальное определение схемы из элементов вычислительной среды.

Пусть дан некоторый конечный автомат A, который определяется заданием:

1) множества реализуемых им элементарных автоматов $\mathfrak{M}_1, \ldots, \mathfrak{M}_l \in \mathfrak{M}$, образующих функциональную полную систему, каждый из которых обладает $q_{j\omega_j}$-внутренними состояниями ($\omega_j \geqslant 2$);

2) множества реализуемых им соединительных элементов $S_{l+1}, \ldots, S_{l+k} \in S$, образующих полную систему соединений;

3) множества состояний реализации $P_1, \ldots, P_r$ ($r = l + k$). В каждом из состояний P_j автомат A реализует либо элементарный автомат $\mathfrak{M}_j$ ($j = 1, \ldots, l$), либо соединительный элемент S_i ($i = l+1, \ldots, l+k$);

4) множества входных и выходных каналов первого типа $x_1, \ldots, x_m$ и $z_1, \ldots, z_m$ соответственно. Входные $x_{j1}, \ldots, x_{jm}$ и выходные $z_{j1}, \ldots, z_{jn}$ каналы у всех элементарных автоматов множества $\mathfrak{M}$ и соединительных элементов множества S отождествлены с соответствующими входными и выходными каналами первого типа автомата A;

5) множества входных и выходных каналов второго типа $y_1, \ldots, y_g$ и $u_1, \ldots, u_g$ соответственно;

6) систем канонических уравнений для каждого элементарного автомата $\mathfrak{M}_j$:

$$z_{j1}(t) = \mathfrak{M}_{j1}[x_{j1}(t), \ldots, x_{jm}(t), q_j(t)];$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$z_{jn}(t) = \mathfrak{M}_{jn}[x_{j1}(t), \ldots, x_{jm}(t), q_j(t)]; \qquad (6.19)$$
$$q_j(t+1) = \psi_j[x_{j1}(t), \ldots, x_{jm}(t), q_j(t)];$$

7) систем канонических уравнений для каждого соединительного элемента S_i:

$$z_{i1}(t) = S_{i1}[x_{i1}(t), \ldots, x_{im}(t)];$$
$$\ldots\ldots\ldots\ldots\ldots\ldots \quad (6.20)$$
$$z_{im}(t) = S_{im}[x_{i1}(t), \ldots, x_{im}(t)];$$

8) систем канонических уравнений для каждого состояния реализации P_j:

$$x_{j1}(t) = \eta[x_1(t), P_j(t)];$$
$$\ldots\ldots\ldots\ldots$$
$$x_{jm}(t) = \eta[x_m(t), P_j(t)]; \quad (6.21)$$
$$z_1(t) = \zeta[z_{j1}(t), P_j(t)];$$
$$\ldots\ldots\ldots\ldots$$
$$z_m(t) = \zeta[z_{jm}(t), P_j(t)];$$

9) систем канонических уравнений для изменения состояний реализации:

$$u_1(t) = f_1[y_1(t), \ldots, y_g(t), R(t)];$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$u_g(t) = f_g[y_1(t), \ldots, y_g(t), R(t)]; \quad (6.22)$$
$$P(t+1) = \varphi[y_1(t), \ldots, y_g(t), R(t), P(t)];$$
$$R(t+1) = \mu[y_1(t), \ldots, y_g(t), R(t)],$$

где $P(t)$ определено на множестве $P_1, \ldots, P_r$.

Пусть дан автомат A с входными и выходными полюсами первого и второго типа, которым приписаны функции $x_{\alpha\gamma}(t)$ и $z_{\alpha\gamma}(t)$; $y_{\alpha\delta}(t)$ и $u_{\alpha\delta}(t)$ соответственно:

$$\alpha = 1, \ldots, s = \frac{n}{2}; \quad \gamma = 1, \ldots, k'_\alpha; \quad \delta = 1, \ldots, k''_\alpha;$$

$$k'_\alpha \geqslant 1, \; k''_\alpha \geqslant 0; \quad k'_\alpha + k''_\alpha = k_\alpha; \; \sum_{\alpha=1}^{s} k_\alpha = K.$$

Поставим в соответствие каждому полюсу $a_{\alpha\beta}$ А-элемента входной полюс первого типа с функцией $x_{\alpha\gamma}(t)$ при $\beta = 1, \ldots, k'_\alpha$ или второго типа с функцией $y_{\alpha\delta}(t)$ при $k'_\alpha < \beta \leqslant k_\alpha$, а полюсу $a'_{\alpha\beta}$ — выходной полюс первого типа с функцией $z_{\alpha\gamma}(t)$ при $\beta = 1, \ldots, k'_\alpha$ или второго типа с функцией $u_{\alpha\delta}(t)$ при $k'_\alpha < \beta \leqslant k_\alpha$.

А-Элемент с поставленным ему в соответствие автоматом A будем называть *элементом вычислительной среды*, а его полюсы $a_{\alpha\beta}$ и $a'_{\alpha\beta}$ с поставленными им в соответствие функциями $x_{\alpha\gamma}(t)$, $z_{\alpha\gamma}(t)$ и $y_{\alpha\delta}(t)$, $u_{\alpha\delta}(t)$ — его *входными и выходными полюсами*

первого и второго типа. Обратим внимание на то, что изменять функцию соединения в соответствии с уравнениями (6.20) можно только для входных и выходных полюсов первого типа. Коммутация входных и выходных полюсов второго типа определяется внутренним состоянием $R(t)$.

Системы уравнений (6.21, 6.22) описывают процесс настройки элемента вычислительной среды либо на выполнение одной из заданных функций $\mathfrak{M}_1, \ldots, \mathfrak{M}_l$, либо на выполнение одного из типов соединений $S_{l+1}, \ldots, S_{l+k}$. Полюсы $y_1, \ldots, y_g$ используются для подачи сигналов на входы элемента. В зависимости от комбинации сигналов, подаваемых на входы, элемент переходит в одно из возможных состояний, определяющих тип функции или тип соединения.

Полюсы $u_1, \ldots, u_g$ используются для передачи сигналов настройки на входные полюсы $y_1, \ldots, y_g$ соседних элементов. Вид функций $f_1, \ldots, f_g, \mu$ и φ зависит от принятого конкретного способа организации настройки вычислительной среды. Настройка производится дискретными тактами $t = 1, 2, 3, \ldots$.

Система уравнений (6.19) описывает функции, реализуемые элементом вычислительной среды.

Функция $\mathfrak{M}_j$ реализуется элементом вычислительной среды при условии, что $P(t) = P_j$ $(j \leqslant l)$, т. е. элемент находится в состоянии, соответствующем настройке элемента на функцию $\mathfrak{M}_j$. При этом полюсы, соответствующие переменным $x_1(t), \ldots, x_m(t)$, являются входными, а $z_1(t), \ldots, z_m(t)$ — выходными. Работа элемента, реализующего функцию $\mathfrak{M}_j$, происходит дискретными тактами $t = 1, 2, 3, \ldots$, каждому из которых соответствует определенное состояние его внутренней памяти $q_{j\omega_j}$.

Система уравнений (6.20) описывает типы соединений, реализуемые элементом вычислительной среды. Тип соединений S_i реализуется элементом вычислительной среды при условии, что $P(t) = P_i$ $(i > l)$, т. е. элемент находится в состоянии, соответствующем настройке элемента на S_i. При этом полюсы, соответствующие переменным $x_1(t), \ldots, x_m(t)$ и $z_1(t), \ldots, z_m(t)$, являются полюсами соединительного элемента S_i.

Теорема 8. *Каждая схема из соединительных элементов, каждая схема из соединительных и функциональных элементов, каждая схема из элементарных автоматов и соединительных элементов может быть реализована схемой из элементов вычислительной среды.*

Элемент вычислительной среды реализует коммутационно полную систему соединительных элементов, полную систему функциональных элементов, полную систему элементарных автоматов. Доказательство теоремы следует из теоремы 4.

Определение. Система элементов вычислительной среды $l_1, \ldots, l_S$ называется *полной*, если любой граф, любую функцию алгебры логики, любой конечный автомат можно представить в виде суперпозиции элементов $l_1, \ldots, l_S$.

Теорема 9. *Минимальная полная система K-полюсных элементов вычислительной среды содержит один элемент.*

Доказательство следует из того, что может быть задан элемент вычислительной среды, описываемый уравнениями (6.19)—(6.22).

Следствие 1. Из теоремы 7 следует, что *минимальное число состояний реализации* $r_{\min} = l_{\min} + k_{\min} = 3$ при $l_{\min} = 1$, $k_{\min} = 2$.

Следствие 2. *Минимальное общее число состояний элемента среды:*

$$r^*_{\min} = \sum_{j=1}^{l_{\min}} q_{j\omega_j} + k_{\min} = 4, \tag{6.23}$$

так как

$$l_{\min} = 1,\ \omega_j = 2,\ k_{\min} = 2.$$

Приведем в качестве простейшего примера построение элемента вычислительной среды из обычных и двухобмоточных поляризованных реле. Для простоты будем полагать, что нам требуется построить 12-полюсный элемент вычислительной среды (рис. 44).

Процесс работы данного элемента вычислительной среды может быть описан уравнениями:

$$\left.\begin{aligned} z_1(t) &= x_1(t) \,\&\, v;\\ z_2(t) &= x_1(t) \,\&\, \bar{v} \,\&\, w_3;\\ V(t+1) &= x_2(t) \,\&\, w_4; \end{aligned}\right\} \tag{6.24}$$

$$z_1(t) = [x_1(t) \,\&\, w_2] \vee [x_2(t) \,\&\, w_1 \,\&\, w_3 \,\&\, \bar{v} \,\&\, w_2] \vee [z_2(t) \,\&\, w_3 \,\&\, \bar{v} \,\&\, w_2];$$

$$z_2(t) = [x_1(t) \,\&\, \bar{v} \,\&\, w_3] \vee [x_2(t) \,\&\, w_1] \vee [z_1(t) \,\&\, w_2 \,\&\, \bar{v} \,\&\, w_3];$$

$$x_1(t) = [x_2(t) \,\&\, w_1 \,\&\, w_3 \,\&\, \bar{v}] \vee [z_1(t) \,\&\, w_2] \vee [z_2(t) \,\&\, w_3 \,\&\, \bar{v}];$$

$$x_2(t) = [x_1(t) \,\&\, \bar{v} \,\&\, w_3 \,\&\, w_1] \vee [z_1(t) \,\&\, w_2 \,\&\, \bar{v} \,\&\, w_3 \,\&\, w_1] \vee [z_2(t) \,\&\, w_1]; \tag{6.25}$$

$$u_1(t) = y_1(t); \quad u_2(t) = y_2(t); \quad u_3(t) = y_3(t); \quad u_4(t) = y_4(t); \tag{6.26}$$

$$W_1(t+1) = y_1(t) \,\&\, y_3(t); \quad W_2(t+1) = y_1(t) \,\&\, y_4(t);$$
$$W_3(t+1) = y_2(t) \,\&\, y_3(t); \quad W_4(t+1) = y_2(t) \,\&\, y_4(t),$$

где w_1, w_2, w_3, w_4 — нормально разомкнутые контакты реле W_1, W_2, W_3, W_4 соответственно;

v и $\bar{v}$ — контакты реле V нормально разомкнутый и нормально замкнутый соответственно.

Уравнения (6.24) описывают работу элемента вычислительной среды в качестве функционального. Функциональный элемент выполнен на реле V. С помощью реле, как известно, можно

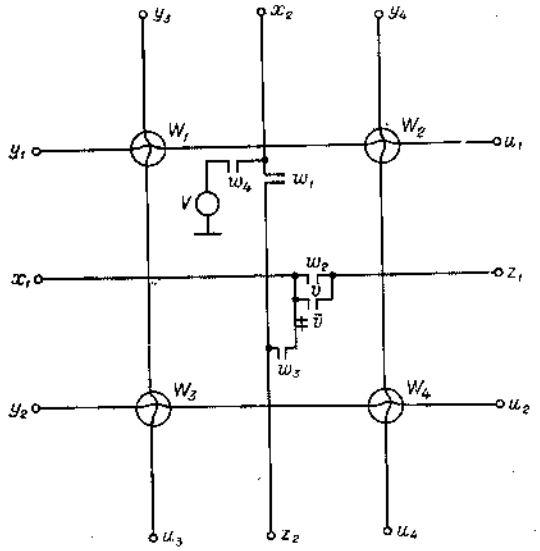


Рис. 44 Схема элемента вычислительной среды.

W_1 — W_4 — двухобмоточные поляризованные реле, используемые для настройки элемента; V — обычное неполяризованное реле, используемое в качестве функционального элемента.

реализовать функционально полную систему элементарных автоматов. Уравнения (6.25) описывают работу элемента вычислительной среды в качестве симметрических соединительных элементов. При этом $x_1(t)$, $x_2(t)$, $z_1(t)$ и $z_2(t)$ могут служить как входами, так и выходами элемента. Набор возможных вариантов задания соединительных элементов является полным. Действительно, при $\bar{w}_1, \bar{w}_2, \bar{w}_3, \bar{w}_4$ образуется соединительный элемент O, при $w_1, w_2, w_3, \bar{w}_4$ — P, при $w_1, w_2, \bar{w}_3, \bar{w}_4 - D$.

Уравнения (6.26) описывают процесс настройки элемента вычислительной среды на работу в качестве функционального или соединительного элементов.

Элемент вычислительной среды с переменной структурой сочетает в себе как функциональное устройство, реализующее функции элементарных автоматов, функции коммутации, так и устройство настройки, которое при подаче на входы элемента определенной комбинации сигналов воспринимает информацию об изменении функционального состояния. Иными словами, в вычислительной среде с переменной структурой нужно указывать адреса элементов, функциональное состояние которых хотят изменить, и информацию, характеризующую это изменение.

6.9. Настройка вычислительной среды

Настройка вычислительной среды заключается в следующем. Пусть имеется вычислительная среда из L элементов, расположенных в n-мерной системе координат ($n = 1, 2, 3$). Каждый элемент имеет память, в которой запоминается одно из возможных состояний реализации элемента ($r \geqslant 3$). Вне вычислительной среды (или данного участка вычислительной среды) имеется программа настройки элементов среды, содержащая информацию о состоянии реализации, в которое должен быть переведен каждый из настраиваемых элементов среды.

Настройка сводится к передаче информации, содержащейся в программе, в памяти соответствующих элементов. Способы настройки можно разделить на два основных типа: с фиксированной структурой и с переменной структурой.

Под первой будем понимать схемы, которые не могут быть изменены программно, под второй — схемы настройки, которые могут изменяться программным путем.

Схемы настройки вычислительной среды с *фиксированной структурой* аналогичны существующим схемам запоминающих устройств. Как и у них, имеются шины выборки элемента по заданному адресу и шины для передачи кода информации. Могут применяться как схемы с произвольной выборкой, например координатные, так и схемы с упорядоченной выборкой типа шагового накопителя, когда ячейки памяти элементов соединяются в цепочку. На каждом шаге i-й элемент цепочки переходит в состояние $(i - 1)$-го элемента, а $(i + 1)$-й элемент в состояние i-го. Как число координат, так и число таких цепочек может быть произвольным.

В качестве примера рассмотрим схему настройки с двухкоординатной выборкой элементов и двухкоординатным заданием кода настройки (рис. 45). Для выбора элемента на входы $a_1, \ldots, a_n, b_1, \ldots, b_m$ подаются сигналы в одну из шин a_i и в одну из шин b_j. При этом выбирается элемент, который находится

на пересечении шин a_i и b_j. Число шин для выборки элемента при этом равно $n + m$. Число выбираемых элементов $L \leqslant nm$. Код настройки подается по шинам $c_1, \ldots, c_s, d_1, \ldots, d_p$ общим для всех элементов. В каждый момент времени подаются сигналы в одну из шин $c_1, \ldots, c_s$ и в одну из шин $d_1, \ldots, d_p$. Число

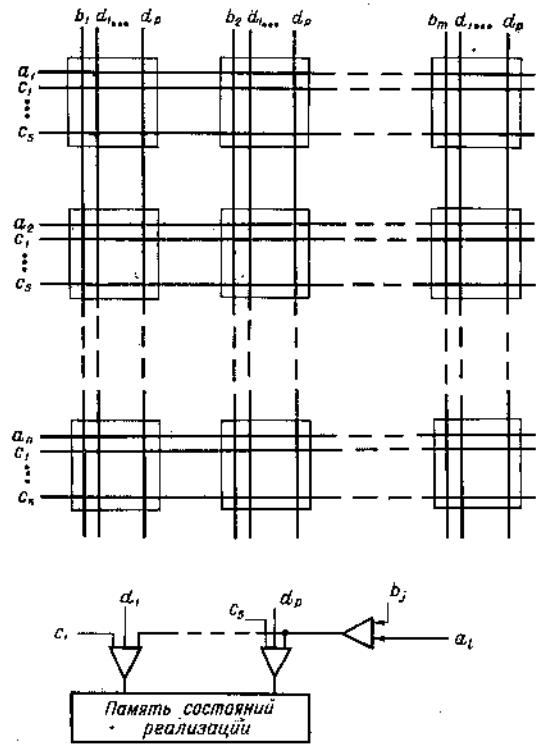


Рис. 45. Схема настройки двухкоординатной выборкой элементов.

$a_1, \ldots, a_n$; $b_1, \ldots, b_m$; $c_1, \ldots, c_s$; $d_1, \ldots, d_p$ — шины, соответствующие входам и выходам второго типа $y_1, \ldots, y_g$; $u_1, \ldots, u_g$ (см. 6. 8)

шин равно $s + p$, $sp = r$. При этом выбирается элемент и передается код настройки за один такт. Число шин может быть уменьшено. Если передавать информацию последовательным двоичным кодом за $\log_2 r$ тактов, то в этом случае $sp = \log_2 r$.

В качестве второго примера рассмотрим схему с упорядоченной выборкой (рис. 46). Памяти состояний реализации R

элементов среды (или части элементов) соединяются в цепочку. Первый элемент цепочки получает информацию извне. На каждом шаге по шине a_0 поступает импульс, отпирающий вентили между памятями соседних элементов, в результате происходит передача содержимого памяти от каждого i-го элемента в $(i+1)$-й. Число шин $a_1, \ldots, a_n$ при кодировании информации двоичным кодом равно $\log_2 r$.

Схемы с упорядоченной выборкой проще схем с произвольной выборкой: у них меньше шин и более проста структура элементов. Эти схемы удобнее, когда надо перенастраивать все элементы среды. При этом время настройки будет тем же или меньше, чем у координатных схем, и пропорционально общему числу элементов среды L.

При настройке только части элементов оказывается более удобным координатный способ, при котором время настройки пропорционально числу настраиваемых элементов среды.

Общий недостаток указанных способов настройки — их невысокая надежность. При выходе любой шины нормальная работа этих схем нарушается, а при упорядоченной выборке нормальная работа нарушена даже при выходе из строя хотя бы одного из элементов. Выход из строя шины приводит к тому, что выпадает группа элементов, через которые проходит данная шина, либо группа состояний реализации. Выход из строя элемента при координатном способе мало сказывается на возможностях среды, так как при этом могут исключаться из работы только его ближайшие соседи. При полном выходе из строя элемента при упорядоченной выборке среда полностью или частично теряет способность к настройке. Следует учесть, что вычислительные среды будут в основном применяться в микроминиатюрных конструкциях, которые, как правило, не ремонтируются.

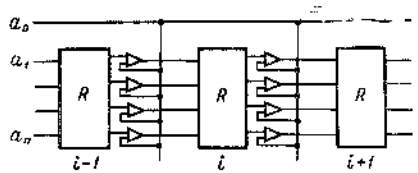


Рис. 46. Схема настройки с упорядоченной выборкой.

Для увеличения надежности схем настройки можно идти двояким путем: либо строить избыточные схемы, либо перейти к схемам с переменной структурой.

В схемах настройки *с переменной структурой* при появлении какой-либо неисправности должна быть предусмотрена возможность такой перестройки структуры схемы, которая локализовала бы влияние этой неисправности в небольшой области. Возникновение неисправности должно приводить только к некоторому уменьшению общего числа элементов вычислительной среды при сохранении всех основных ее свойств.

Ограничимся рассмотрением таких схем с переменной структурой, при которых неисправность одиночного элемента среды локализуется перенастройкой только его ближайших соседей.

Один из примеров схемы настройки с переменной структурой приведен в работе [18]. В этой схеме все элементы представляются расположенными в виде схемы информационного дерева. Для каждого элемента указывается его предшественник, от которого он получает настроечную информацию, и два элемента, следующих за ним. В каждом элементе имеется триггер выбора направления, указывающий, какому из двух последующих элементов должна быть передана настроечная информация. Более общий случай показан на рис. 47.

Каждый элемент двумерной вычислительной среды с координатами (i, j) может получать и передавать настроечную информацию любому из четырех своих соседей. Для простоты предшествующий элемент обозначается индексом k, последующий индексом l. Оба индекса принимают значения 1, 2, 3, 4, обозначающих соответственно соседей элемента (i, j), $(i, j-1)$, $(i-1, j)$, $(i, j+1)$, $(i+1, j)$.

В каждом элементе кроме памяти состояний реализации содержится память выбора направления, имеющая четыре состояния, которые соответствуют выбору одного из четырех последующих элементов, и нулевое состояние. Состояние этой памяти изменяется программно, когда триггер активности возбужден (находится в состоянии 1). В каждый данный момент в состоянии возбуждения может находиться триггер только одного из элементов среды (или участка среды).

Состояние возбуждения последовательно передается от предшествующего элемента к последующему. Все элементы среды от первого, соединенного с внешним источником информации, до последнего (L-го) образуют цепочку. На первом шаге возбуждается триггер первого элемента среды, соединенного с внешним источником информации (шины a). В память выбора направления засылается информация, определяющая второй элемент цепи (шины c). В память состояний реализации засылается код настройки для последнего (L-го) элемента среды. На втором шаге возбуждается триггер активности второго элемента, выбор которого обусловлен содержимым памяти выбора направления.

Триггер активности первого элемента переходит в состояние 0 и отпирает каналы $c_1, \ldots, c_5$ для передачи информации о выборе направления в последующие элементы. Когда состояние возбуждения триггера активности доходит до элемента (i, j), то он

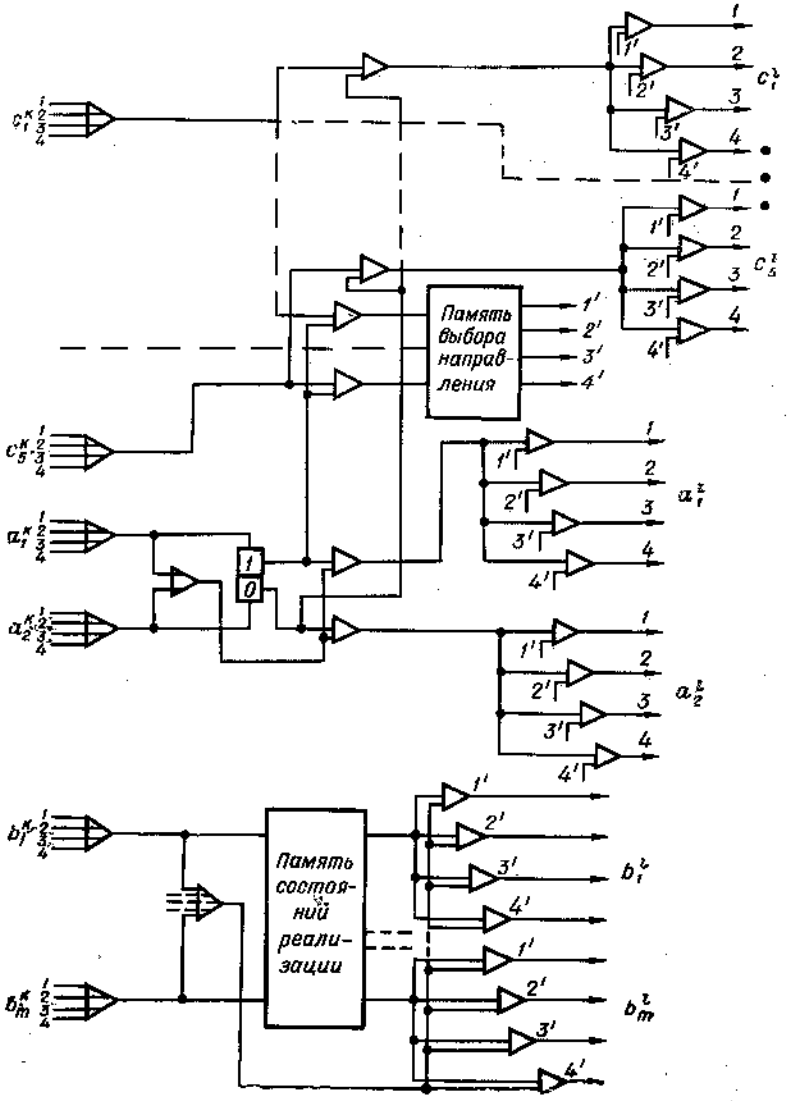


Рис. 47. Схема настройки с переменной структурой.

оказывается соединенным цепочкой из предшествующих ему элементов с источником внешней информации. Поступающая от этого источника на входы $c_1^k, \ldots, c_5^k$ информация запоминается в памяти выбора направления, которая, в свою очередь, отпирает путь к одному из соседей. Одновременно с этим в память состояний реализации поступает код настройки L-го элемента. Через L шагов, когда состояние активности достигает последнего элемента среды (или того элемента, на котором хотят закончить настройку), настройка заканчивается подачей сигнала в шину c_5. В данной схеме коды настройки передаются аналогично схеме шагового накопителя.

Нетрудно видеть, что в указанной схеме можно изменять конфигурацию путей передачи информации и локализовать любую неисправность в элементе или совокупности элементов. Другое достоинство данной схемы — ее гибкость — позволяет образовывать большое число одновременно настраиваемых цепочек элементов, что приводит к сокращению времени настройки.

Вариантов схем настройки с переменной структурой можно предложить много, они могут отличаться способом передачи кодов настройки, количеством выбираемых направлений и т. п. Однако все варианты будут иметь несколько большую сложность по сравнению со схемами настройки с фиксированной структурой.

6.10. Физическая реализация элементов вычислительной среды с переменной структурой

Для отыскания наиболее подходящих физических реализаций элемента вычислительной среды попытаемся представить

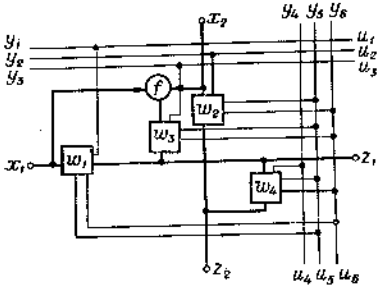


Рис. 48. Схема элемента вычислительной среды с симметрическими соединительными элементами.

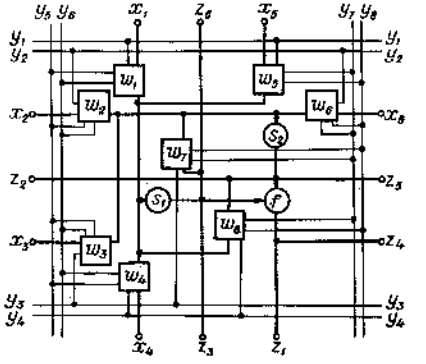


Рис. 49. Схема элементов вычислительной среды с антисимметрическими соединительными элементами.

его функциональную схему в таком виде, чтобы она состояла из небольшого числа по возможности однотипных функциональных элементов. Для этого рассмотрим варианты схем двух типов: с симметрическими и антисимметрическими соединительными элементами. Предположим для определенности, что у обеих схем выборка элементов среды и передача кода настройки выполняются координатным способом в двумерной системе координат.

Указанные функциональные схемы (рис. 48, 49) описываются соответственно системами уравнений:

$$\begin{cases} z_1(t) = x_1(t)\,\&\,w_1(t) \vee x_2(t)\,\&\,w_2(t)\,\&\,w_1(t) \vee \\ \qquad \vee z_2(t)\,\&\,w_4(t) \vee f(x_1(t), x_2(t))\,\&\,w_3(t); \\ z_2(t) = [x_2(t)\,\&\,w_2(t)] \vee (x_1(t)\,\&\,w_1(t)\,\&\,w_1(t)\&w_4(t)) \vee \\ \qquad \vee (z_1(t)\,\&\,w_4(t)) \vee (f(x_1(t), x_2(t))\,\&\,w_3(t)\,\&\,w_4(t)); \\ x_1(t) = (z_1(t)\,\&\,w_1(t)) \vee (z_2(t)\,\&\,w_1(t)\,\&w_4(t)) \vee \\ \qquad \vee (x_2(t)\,\&\,w_1(t)\,\&\,w_2(t)\,\&\,w_4(t)); \\ x_2(t) = (z_2(t)\,\&\,w_2(t)) \vee (z_1(t)\,\&\,w_2(t)\,\&\,w_4(t)) \vee \\ \qquad \vee (x_1(t)\,\&\,w_1(t)\,\&\,w_2(t)\,\&\,w_4(t)); \\ w_1(t+1) = y_1(t)\,\&\,y_5(t); \quad \bar{w}_1(t+1) = y_1(t)\,\&\,y_6(t); \\ w_2(t+1) = y_2(t)\,\&\,y_5(t); \quad \bar{w}_2(t+1) = y_2(t)\,\&\,y_6(t); \\ w_3(t+1) = y_3(t)\,\&\,y_5(t); \quad \bar{w}_3(t+1) = y_3(t)\,\&\,y_6(t); \\ w_4(t+1) = y_4(t)\,\&\,y_5(t); \quad \bar{w}_4(t+1) = y_4(t)\,\&\,y_6(t); \\ u_1(t) = y_1(t); \qquad t = 0, 1, 2, \ldots; \\ \ldots\ldots\ldots \\ u_8(t) = y_6(t); \qquad f(x_1(t), x_2(t)); \end{cases} \tag{6.27}$$

$$\begin{cases} z_1(t) = z_4(t) = f(s_1(t), s_2(t)); \\ z_2(t) = z_5(t) = s_2(t) = (x_2(t)\,\&\,w_2(t)) \vee (x_3(t)\,\&\,w_3(t)) \vee \\ \qquad \vee (x_6(t)\,\&\,w_6(t)) \vee (s_1(t)\,\&\,w_7(t)); \\ z_3(t) = z_6(t) = s_1(t) = (x_1(t)\,\&\,w_1(t)) \vee (x_4(t)\,\&\,w_4(t)) \vee \\ \qquad \vee (x_5(t)\,\&\,w_5(t)) \vee (s_2(t)\,\&\,w_8(t)); \\ w_1(t+1) = y_1(t)\,\&\,y_5(t); \quad \bar{w}_1(t+1) = y_1(t)\,\&\,y_6(t); \\ w_2(t+1) = y_2(t)\,\&\,y_5(t); \quad \bar{w}_2(t+1) = y_2(t)\,\&\,y_6(t); \\ w_3(t+1) = y_3(t)\,\&\,y_5(t); \quad \bar{w}_3(t+1) = y_3(t)\,\&\,y_6(t); \\ w_4(t+1) = y_4(t)\,\&\,y_5(t); \quad \bar{w}_4(t+1) = y_4(t)\,\&\,y_6(t); \end{cases} \tag{6.28}$$

$$\begin{cases} w_5(t+1) = y_1(t)\,\&\,y_7(t); \quad \bar{w}_5(t+1) = y_1(t)\,\&\,y_8(t); \\ w_6(t+1) = y_2(t)\,\&\,y_7(t); \quad \bar{w}_6(t+1) = y_2(t)\,\&\,y_8(t); \\ w_7(t+1) = y_3(t)\,\&\,y_7(t); \quad \bar{w}_7(t+1) = y_3(t)\,\&\,y_8(t); \\ w_8(t+1) = y_4(t)\,\&\,y_7(t); \quad \bar{w}_8(t+1) = y_4(t)\,\&\,y_8(t); \\ u_1(t) = y_1(t); \qquad t = 0, 1, 2, \ldots; \\ \ldots\ldots\ldots \\ u_8(t) = y_8(t); \qquad f(s_1(t), s_2(t)). \end{cases}$$

В этих уравнениях и на соответствующих им схемах входы и выходы элемента $x_1, \ldots, x_6$; $z_1, \ldots, z_6$; $y_1 \ldots, y_8$; $u_1, \ldots, u_8$ те же, что и в п. 6.8.

Схемы образованы элементами трех типов:

$s_1(t)$, $s_2(t)$ — усилительные элементы с односторонней проводимостью, функцией которых является восстановление сигнала до стандартной формы;

$f(x_1, x_2)$ и $f(s_1, s_2)$ — полные системы функциональных элементов;

$w_1, \ldots, w_8$ — элементы, функции которых могут быть описаны следующей системой уравнений:

$$g_{xz}(w(t)) = g_{zx}(w(t)) = w(t);$$
$$w(t+1) = v_3(t)\&v_2(t), \quad \bar{w}(t+1) = v_1(t)\&v_2(t); \tag{6.29}$$
$$t = 0, 1, 2, \ldots,$$

где g — функция проводимости между полюсами x и z, принимающая два значения $g_{xz} = 1$, когда существует путь между полюсами x и z, и $g_{xz} = 0$, когда этого пути нет. На языке теории графов это соответствует матрице смежности $\begin{pmatrix} 0 & w(t) \\ w(t) & 0 \end{pmatrix}$.

Назовем элемент, описываемый уравнениями (6.29), *триггерным переключателем*, или *триггерным контактом*, сокращенно *трикон* (рис. 50).

Трикон выполняет функции управляемого контакта, соединяющего полюса двухполюсника. Контакт хранит свое состояние до прихода сигнала о его изменении. Состояние контакта управляется ко-

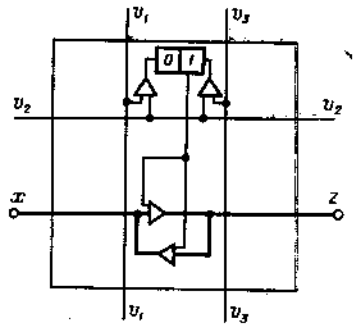


Рис. 50. Трикон из обычных элементов.

ординатными шинами. При задании положения трикона в двумерной системе координат этих шин три (v_1, v_2, v_3), что позволяет индивидуально управлять состоянием любого трикона.

Элементы s_1 и s_2 могут быть реализованы с помощью общеизвестных усилителей-формирователей самых различных типов.

Способы реализации функциональных элементов также общеизвестны. Различные варианты реализации элементов, образующих функционально полную систему, показаны, например, в работе [2]. Для их построения могут использоваться самые различные физические принципы: электрические, магнитные, механические, оптические, пневматические, химические и др. Аналогично обстоит дело с реализацией временной задержки (памяти). В частности, элемент памяти можно выполнить схемно. Триггер, например, может быть построен из двух инверторов, которые используются для реализации логического отрицания.

Сейчас наметилась тенденция строить схемы вычислительных машин из универсальных элементов. Универсальный элемент реализует функционально полную систему логических функций и временную задержку. Кроме того, он обладает способностью приводить сигнал к стандартной форме, т. е. играет роль усилителя-формирователя. Часто функция памяти реализуется путем отождествления соответствующих полюсов универсальных элементов.

В качестве примера универсального, обладающего функциональной полнотой элемента можно назвать элемент, реализующий функцию $z = \overline{x_1 \vee x_2} = \bar{x}_1 \& \bar{x}_2$ (стрелка Пирса). Он может быть, например, выполнен на одном транзисторе и сопротивлениях (рис. 51). Соответствующим отождествлением полюсов нескольких таких элементов можно реализовать дизъюнкцию, конъюнкцию, а также триггер. Наличие транзистора позволяет также усиливать (формировать) сигналы.

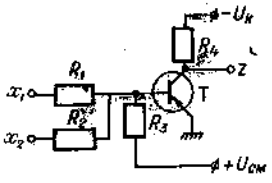


Рис. 51. Схема универсального функционального элемента.

Аналогично реализуются универсальные элементы на феррито-диодных, феррито-транзисторных ячейках, параметронах, криотронах и т. п.

Рассмотрим различные варианты физической реализации трикона. Сделаем это на примере трех типов физических приборов.

Для приборов первого типа характерно отсутствие электрической связи между управляющими и рабочими полюсами и то, что они не изменяют своего состояния после прекращения действия управляющего сигнала и даже после полного отключения их

от источников питания. К ним относятся поляризованные реле, трансфлюксоры, биаксы и т. п. Свойства этих приборов позволяют строить триконы, используя только один прибор и, как правило, без дополнительных элементов.

У приборов второго типа также нет электрической связи между управляющими и рабочими полюсами, однако в отличие от приборов первого типа в них нет запоминания состояния. После снятия внешнего сигнала они возвращаются в начальное состояние. К таким элементам принадлежат обычные электромагнитные реле, криотроны, оптотроны, термисторы и т. п. Эти приборы представляют собой четырехполюсники с двумя управляющими (a, b) и двумя рабочими полюсами (x, z) (рис. 52). Воздействуя на полюсы a и b, можно изменять величину сопротивления R, через которое соединены рабочие полюсы x и z. Величина сопротивления резко изменяется под воздействием внешнего сигнала: от высокоомного (состояние 0) до низкоомного (состояние 1). Для построения трикона необходимо использовать несколько таких приборов либо прибегать к различным схемным ухищрениям.

Рис. 52. Схема трикона на управляемом сопротивлении.

Для приборов третьего типа характерно, что они, как правило, двух- и трехполюсные и что у них имеется электрическая связь между управляющими и рабочими полюсами. Представителями данного типа приборов являются обычные ламповые и полупроводниковые триоды, управляемые диоды типа p- n- p- n, туннельные диоды, тиратроны и т. п.

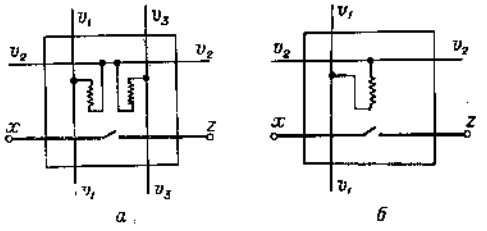


Рис. 53. Схема трикона на поляризованном реле.

а — двухобмоточном; *б* — однообмоточном.

Рассмотрим примеры схем триконов на физических приборах всех трех типов.

Трикон на двух- и однообмоточном поляризованном реле. Свойство поляризованных реле сохранять свое состояние после снятия сигнала позволяет легко реализовать на них трикон (рис. 53). У двухобмоточного реле при подаче сигнала в одну обмотку трикон переходит в состояние 1 (контакт замкнут), при подаче в другую — в состояние 0 (кон-

такт разомкнут). В однообмоточном реле для переключения трикона подают сигналы в обмотку то одной, то другой полярности.

Трикон на трансфлюксоре. Трансфлюксор представляет собой сердечник из феррита с прямоугольной петлей гистерезиса, имеющий в простейшем случае два отверстия (рис. 54). С помощью обмотки u_1, проходящей через центральное отверстие, сердечник может быть приведен в состояние 0 либо 1 (рис. 54, *а* и *б*). В состоянии 0 переменный сигнал в обмотке, проходящей через малое отверстие, может только усиливать магнитный поток вокруг этого отверстия, а так как сердечник находится в состоянии насыщения, то сигнал не изменяет магнитного потока и не вызывает сигнала в другой обмотке. В состоянии 1 переменный сигнал может уменьшить магнитный поток вокруг малого отверстия и тем самым вызвать сигнал в другой обмотке, т. е. между обеими

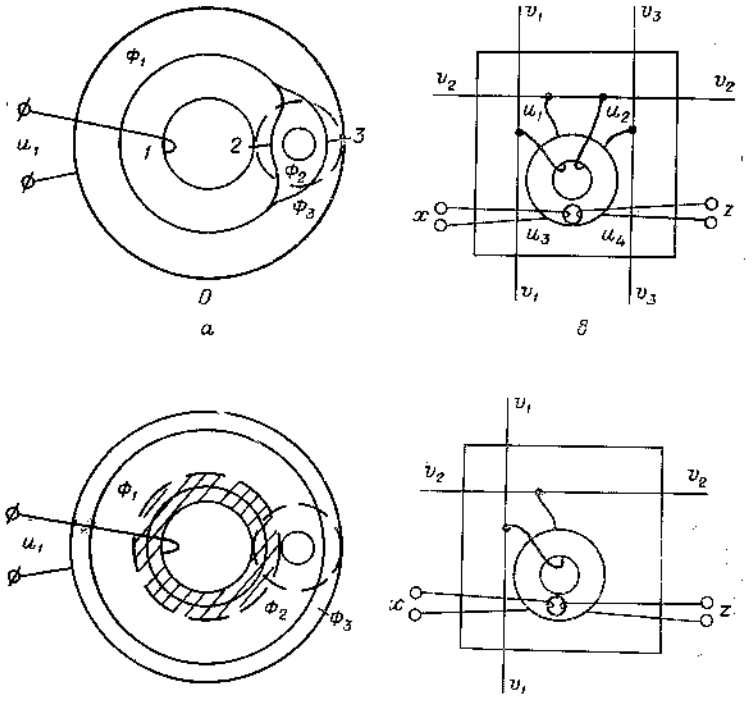


Рис. 54. Схема трикона на трансфлюксоре.

а — трансфлюксор в состоянии 0; *б* — трансфлюксор в состоянии 1; *в* — трикон на трансфлюксоре с двумя управляющими обмотками; *г*—трикон на трансфлюксоре с одной управляющей обмоткой.

обмотками u_3 и u_4 появляется трансформаторная связь. Это свойство позволяет создать трикон на одном трансфлюксоре. В трансфлюксоре с двумя управляющими обмотками (рис. 54, *в*) переход трикона из одного состояния в другое происходит при подаче сигнала в соответствующую обмотку. При одной управляющей обмотке (рис. 54, *г*) в нее подаются сигналы различной полярности.

Аналогичным образом может быть реализован трикон на биаксах и других элементах подобного типа.

Трикон на обычном реле. В простейшем случае трикон может быть построен на обычном реле, которое имеет одну контактную

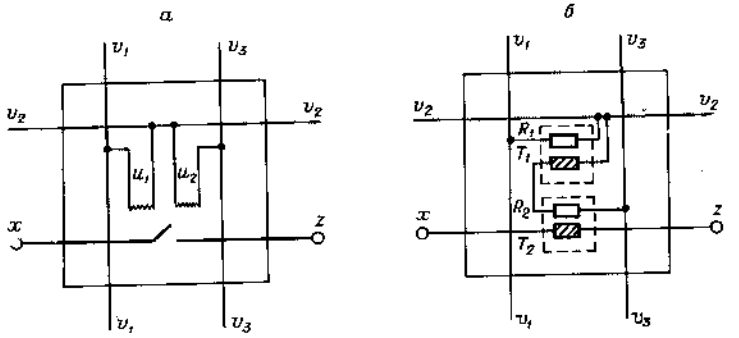


Рис. 55. Схема трикона.

а — на обычном реле; *б* — на термисторах с косвенным подогревом.

пару и две обмотки (рис. 55, *а*). Обмотка u_1 служит для задания состояния, обмотка u_2 — для поддержания состояния реле при замкнутом контакте. При подаче сигнала на обмотку u_1 реле возбуждается и его контакт замыкается (состояние 1). В обмотке u_2 течет непрерывно ток, величина которого достаточна для удержания реле в состоянии 1 после снятия сигнала в обмотке u_1, но недостаточна для возбуждения реле, если оно находится в состоянии 0 (контакт разомкнут). Для перевода реле в состояние 0 нужно обесточить обмотку u_2. Это можно сделать с помощью дополнительного реле, которое при включении будет обесточивать либо все триконы среды, либо триконы только данного элемента среды.

Возможность поддержания реле в возбужденном состоянии можно реализовать и другими способами, не прибегая к двухобмоточным реле, например с помощью блокировки через собственный дополнительный контакт и т. п.

Трикон на термисторах с косвенным подогревом. Сопротивление термистора велико при обычной температуре и уменьшается на несколько порядков при нагреве. Это позволяет построить трикон на двух термисторах (T_1 и T_2), снабженных для их подогрева сопротивлениями R_1 и R_2 (рис. 55, *б*). Управляющий сигнал подается на полюсы v_1, v_2. К полюсам v_3, v_2 приложено напряжение. При подаче управляющего сигнала в течение некоторого времени происходит нагрев сопротивления R_1. В результате сопротивление термистора T_1 уменьшается, ток через него усиливается и термистор T_1 переходит в состояние высокой проводимости. Одновременно возрастает ток в сопротивлении R_2 и термистор T_2 также становится высокопроводящим. Полюсы x и z оказываются соединенными через низкоомное сопротивление (состояние 1). Это состояние сохраняется после снятия управляющего сигнала. Чтобы вывести трикон из этого состояния, нужно снять напряжение. Это делается либо для всех элементов вычислительной среды одновременно, либо только для триконов одного элемента. При этом термисторы остывают, величина их сопротивления резко возрастает и полюсы x и z оказываются соединенными через высокоомное сопротивление (состояние 0).

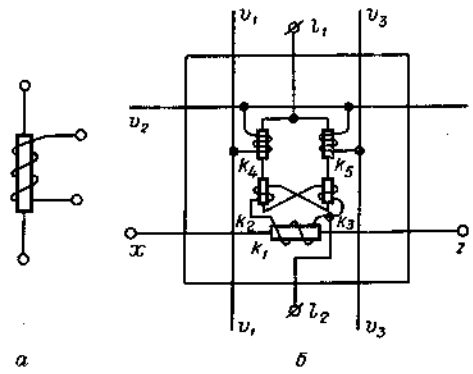


Рис. 56. Схема трикона на криотронах.
а — криотрон; *б* — трикон.

Трикон на криотронах. Криотрон обычно изготовляется в виде стержня, вокруг которого нанесена обмотка (рис. 56, *а*). И стержень и обмотка делаются из сверхпроводящих материалов, например, стержень из тантала, а обмотка из ниобия. Криотрон помещается в криостат, в котором поддерживается температура жидкого гелия. При этом как тантал, так и ниобий находятся в сверхпроводящих состояниях. При пропускании тока соответствующей величины через ниобиевую обмотку тантал переходит в состояние обычной проводимости. Ниобий, у которого критическая точка перехода в обычное состояние выше, чем у тантала, остается в сверхпроводящем состоянии. При прекращении тока в обмотке танталовый стержень снова возвращается в состояние

сверхпроводимости. Это свойство криотрона позволяет построить трикон на криотронах (рис. 56, *б*).

Состояние 1 трикона соответствует состоянию сверхпроводимости танталового стержня криотрона k_1. При этом полюсы x и z оказываются короткозамкнутыми. Состояние 0 соответствует состоянию обычной проводимости стержня. Тогда полюсы x и z оказываются соединенными друг с другом через сопротивления. Для изменения состояния подаются сигналы на обмотки криотронов k_4 и k_5, которые изменяют состояние триггера, сделанного на криотронах k_2 и k_3. Нетрудно видеть, что ток от источника (l_1, l_2) течет через обмотку криотрона k_1 и переводит его в состояние обычной проводимости (состояние 0) только тогда, когда криотрон k_3 находится в сверхпроводящем состоянии, а k_2 — в обычном.

Трикон в состояние 0 переключается подачей сигнала в шины v_1, v_2. Сигнал в шинах v_3, v_2 переключает трикон в состояние 1. Благодаря триггеру эти состояния запоминаются и хранятся после снятия внешнего сигнала.

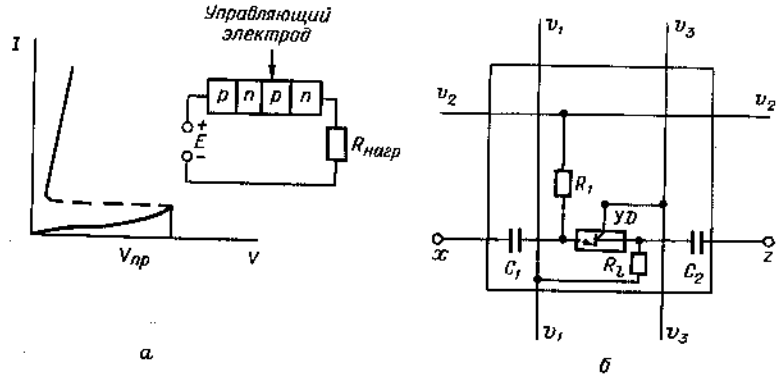


Рис. 57. Схема трикона на полупроводниковом управляемом диоде с *p-n-p-n*-переходами.
а — вольтамперная характеристика диода; *б* — принципиальная схема трикона.

Трикон на полупроводниковом управляемом диоде типа p-n-p-n. Диоды с *p-n-p-n*-переходами (например, кремниевые) имеют *S*-образную вольтамперную характеристику с участком отрицательного сопротивления (рис. 57, *а*). Ток через диод мал, пока напряжение на нем не достигнет пробивного значения $V_{пр}$. При этом напряжении сопротивление диода скачком меняет свое значение. Напряжение на диоде падает, ток растет, и его величина ограничивается в основном сопротивлением нагрузки.

Подачей сигнала на управляющий электрод можно уменьшать значение $V_{пр}$ и тем самым переводить диод в состояние с высокой проводимостью. В этом состоянии диод остается и после прекращения сигнала. После пробоя управляющий электрод теряет способность управлять состоянием диода. Схема переходит в состояние низкой проводимости при отключении источника напряжения. В этом отношении управляемый диод аналогичен тиратрону. Трикон может быть построен на одном управляемом диоде (УД), двух сопротивлениях и двух емкостях (рис. 57, *б*). К шинам v_1, v_2 подключен источник напряжения E.

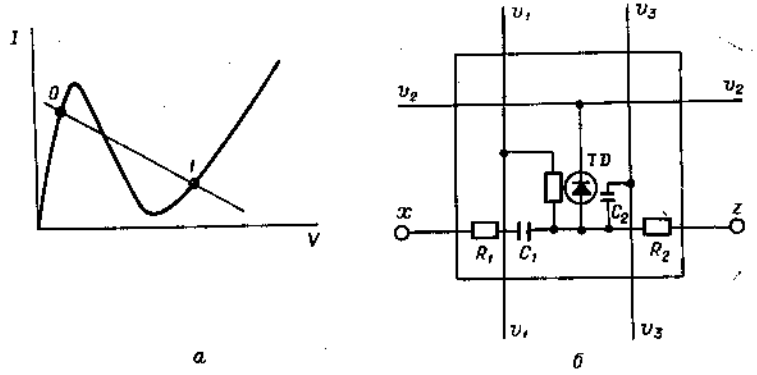


Рис. 58. Схема трикона на туннельном диоде.

а — вольтамперная характеристика туннельного диода; *б* — принципиальная схема трикона.

Переключение трикона в состояние высокой проводимости (состояние 1) производится подачей сигнала в шины v_3, v_2. При этом создается возможность для прохождения сигналов от x к z. Чтобы сигналы проходили и от z к x, можно подавать двухполярные сигналы либо использовать еще один трикон. Для приведения трикона в состояние непроводимости необходимо снять напряжение с шин v_1, v_2, что может быть выполнено соответствующей схемой, снимающей напряжение либо с триконов всех элементов среды, либо только данного элемента.

Трикон на туннельном диоде. Вольтамперная характеристика туннельного диода имеет N-образную форму с участком отрицательного сопротивления (рис. 58, *а*), что позволяет строить на одном приборе схемы с двумя устойчивыми состояниями (0 и 1 на рис. 58, *а*). В состоянии 0 сопротивление диода мало, а в состоянии 1 велико. Трикон может быть построен на одном туннельном диоде (ТД), сопротивлениях и емкостях (рис. 58, *б*). На

шины v_1, v_2 подается напряжение. В состоянии 0 туннельный диод отперт и шунтирует сопротивление R_1 и сигнал между рабочими полюсами x и z не проходит. В состоянии 1 туннельный диод имеет высокое сопротивление и не препятствует прохождению сигналов между полюсами x и z. Трикон из одного состояния в другое переключается подачей в шины v_3, v_2 сигналов соответствующей полярности.

Аналогичным образом могут быть построены триконы на неоновых лампах, параметронах, термисторах, без косвенного подогрева и т. п.

Следовательно, схемы триконов могут строиться из тех же физических приборов, что и схемы функциональных элементов при одинаковой их сложности.

Простота трикона существенно зависит от того, насколько хорошо сочетаются свойства используемых физических приборов с особенностями трикона. Для создания триконов могут оказаться перспективными химические элементы, элементы, основанные на пробое, элементы, использующие частотный и фазово-импульсный принципы. Весьма выгодным может оказаться и применение нейристоров.

Свойства трикона весьма универсальны. На них может быть построена также полная система функциональных элементов и память. Триконы могут также обладать и усилительными свойствами, т. е. они могут выполнять функции усилителя-формирователя. Таким образом, элемент вычислительной среды может быть создан из одних триконов путем соответствующего отождествления их полюсов.

Заметим, что требование, чтобы трикон обладал свойством двусторонней проводимости между рабочими полюсами, не обязательно, тем более что трикон с двусторонней проводимостью может быть сделан из двух триконов с односторонней проводимостью.

6.11. Логические схемы из элементов вычислительной среды

В качестве основы для построения схем возьмем двухмерную вычислительную среду с координатной настройкой и элементом вычислительной среды с антисимметрическими соединительными элементами (см. рис. 49), описываемым системой уравнений (6.28).

Пусть элемент реализует функцию алгебры логики $f[s_1(t), s_2(t)] = \overline{s_1(t) \vee s_2(t)}$ (стрелку Пирса), образующую полную систему функциональных элементов. Обозначим ненастроенный (с изолированными полюсами) элемент вычислительной среды

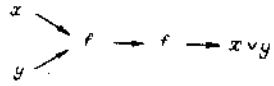
пустым квадратом. При обозначении элемента, настроенного на выполнение функции $f[s_1(t), s_2(t)]$, будем использовать символ f.

При обозначении элемента, настроенного на выполнение коммутаций между полюсами элемента, будем использовать либо графическое обозначение коммутации, либо указывать символом, какие полюсы элемента коммутируются. При построении логических схем будем описывать эти схемы формулами, структурными схемами и в виде двухмерной решетки вычислительной среды. Для упрощения изображения шины настройки элементов среды приводиться не будут.

Реализация в вычислительной среде простейших функций алгебры логики дизъюнкции $\vee$, *конъюнкции* &, *отрицания* $\neg$ (рис. 59). Дизъюнкция от двух переменных x и y может быть представлена через $f(x, y)$ как

$$x \vee y = f(f(x, y)\ ,\ ) = f(\ ,\ f(x, y)) = f(f(x, y), f(x, y)) \quad (6.30)$$

или при представлении структурной схемой



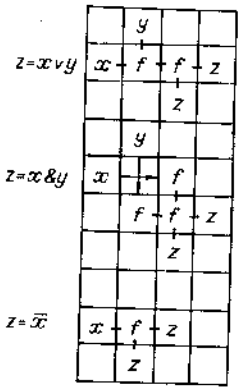
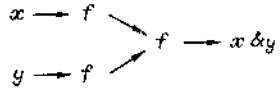


Рис. 59. Реализация функций алгебры логики в вычислительной среде.

Аналогично конъюнкция тех же переменных имеет вид:

$$x \,\&\, y = f(f(\ ,x), f(\ , y)) = f(f(x,\ ),\ f(\ , y)) = f(f(x,\ ), f(y,\ )) = f(f(\ , x), f(y,\ )) = f(f(x, x),\ f(y, y)) \quad (6.32)$$

или



Функция отрицания

$$\bar{x} = f(x, x) = f(x,\ ) = f(\ , x) \quad (6.34)$$

или

$$x \xrightarrow{f} \bar{x}. \quad (6.35)$$

Реализация в вычислительной среде неравнозначности, равнозначности, одноразрядного преобразователя и двойного переключателя. Схема неравнозначности

$$z = (x \vee y)\,\overline{xy} \quad (6.36)$$

представляется структурной схемой через $f(x, y)$ (рис. 60, *а*):

(6.37)

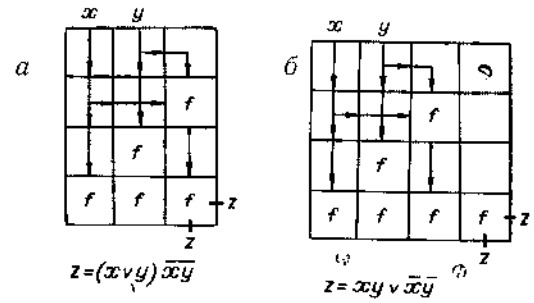


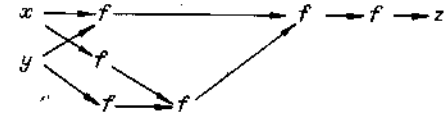
Рис. 60. Реализация схемы в вычислительной среде.

а — неравнозначности; *б* — равнозначности.

Схема равнозначности описывается формулой

$$z = (x \,\&\, y) \vee (\bar{x} \,\&\, \bar{y}) \quad (6.38)$$

и имеет вид структурной схемы в представлении через $f(x, y)$ (рис. 60, *б*):



(6.39)

Схема одноразрядного преобразователя (рис. 61) —

$$z = x_1 y \vee y\,\bar{x}_2 \quad (6.40)$$

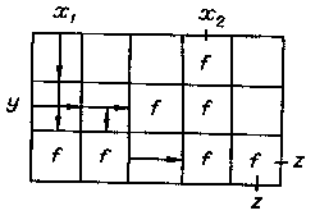
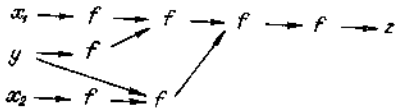


Рис. 61. Схема одноразрядного преобразователя.

в структурной схеме имеет вид:

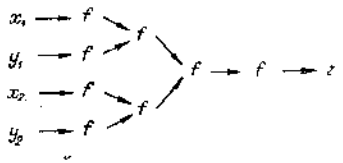


(6.41)

Схема двойного переключателя описывается формулой

$$z = x_1y_1 \vee x_2y_2, \tag{6.42}$$

и представляется структурной схемой (рис. 62):



(6.43)

Реализация регистров и счетчиков различных типов в вычислительной среде. Рассмотрим схемы двоичного счетчика, регистра параллельного действия, накопительного регистра, регистра

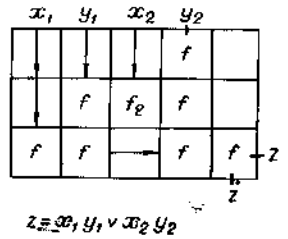
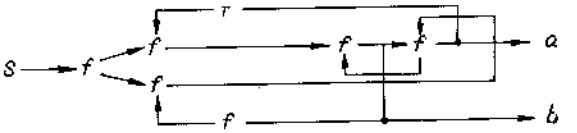


Рис. 62. Схема двойного переключателя.

для преобразования параллельного кода в последовательный и обратный, которыми охватываются практически все типы счетчиков и регистров в вычислительных машинах.

Основой для построения указанных схем служит схема триггера (рис. 63).

Триггер может быть получен соединениями двух схем, реализующих функцию $f(x, y)$, таким образом, что вход каждой схемы соединен с выходом другой. Структурная схема триггера со счетным входом (рис. 64) имеет следующий вид:



(6.44)

Схема n-разрядного двоичного счетчика (рис. 65) построена из схем триггера (см. рис. 63). В n-разрядном регистре параллельного действия (рис. 66,*а*) при подаче разрешения на шину S_3

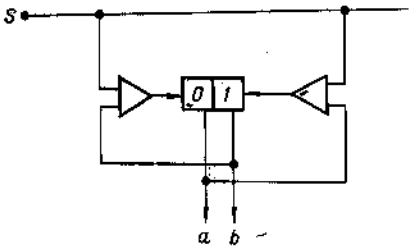


Рис. 63. Схема триггера со счетным входом и двумя вентилями (конъюнкторами).

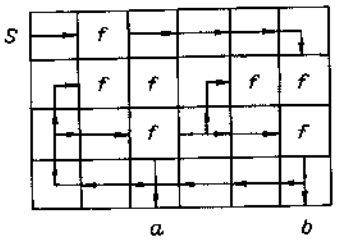


Рис. 64. Реализация схемы триггера в вычислительной среде.

параллельно записывается в регистр n-разрядное число, поступающее по шинам $k_1, \ldots, k_n$. При подаче сигнала на шину S_2 регистр устанавливается в положение 0. При подаче сигнала на шину S_1 выдается n-разрядное число по шинам $L_1, \ldots, L_n$.

Аналогично реализуется и накопительный n-разрядный регистр (рис. 67). По шинам a, b подается информация на накопительный регистр. Одновременно поступает сигнал на шину S. В результате первый разряд заполняется новой информацией, а его прежнее содержимое передается второму разряду, содержимое которого передается третьему и т. д.

Схема реализации регистра для преобразования параллельного кода в последовательный и обратно приведена на рис. 68. При подаче сигнала по шине S_1 записывается параллельный код,

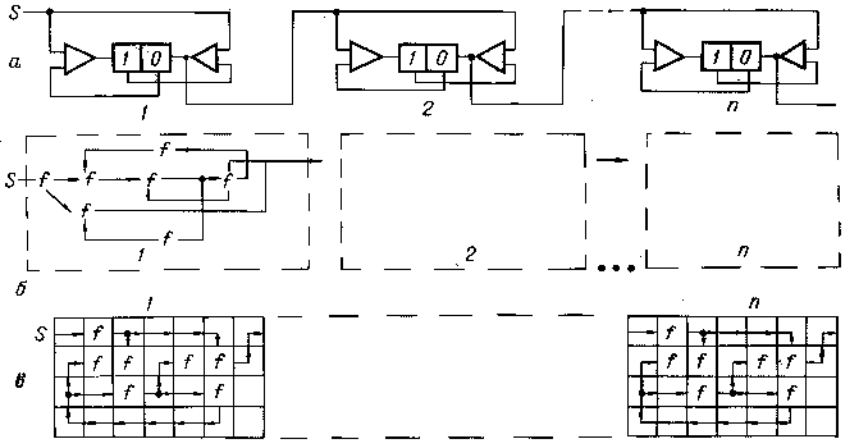


Рис. 65. Схема n-разрядного двоичного счетчика.

а — в обычном изображении; *б* — построенная с помощью одного функционального элемента f (стрелка Пирса); *в* — реализованная в вычислительной среде.

поступающий в регистр по шинам $k_1, \ldots, k_n$. При подаче сигналов по шине S_2 параллельный код, запомненный в регистре, преобразуется в последовательный и выдается по шинам c и d. Для преобразования в последовательный код подается сигнал на шины a и e при одновременной подаче сигналов в S_2. При подаче сигнала разрешения в шину S_1 выдается параллельный код по шинам $L_1, \ldots, L_n$. Регистр устанавливается в состояние 0 подачей сигнала в S_3.

Реализация дешифраторов в вычислительной среде. В переключательных схемах вычислительной техники широко распро-

странены различного рода дешифраторы. Их назначение заключается в том, чтобы каждую комбинацию входных переменных (код) преобразовать в выходной сигнал одной из выходных шин дешифратора. Рассмотрим реализацию в вычислительной среде

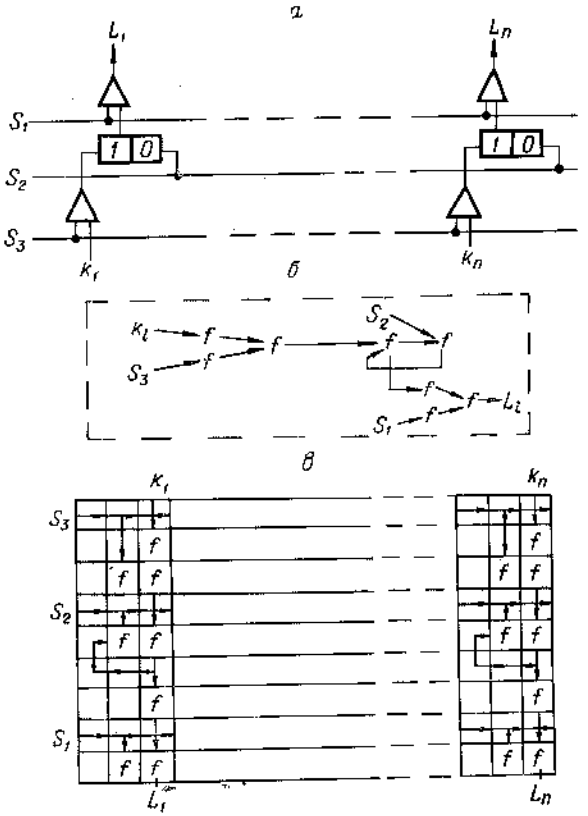


Рис. 66. Схема n-разрядного регистра параллельного действия.

а — логическая схема n-разрядного регистра; *б* — структурная схема одного разряда регистра; *в* — реализация n-разрядного регистра в вычислительной среде.

двух типов дешифраторов: пирамидального и прямоугольного. Другие типы дешифраторов могут быть реализованы аналогично.

Реализация пирамидального дешифратора на восемь выходов показана на рис. 69. Комбинация входных сигналов x_1, x_2, x_3 запоминается в триггерах T_1, T_2, T_3. При подаче сигнала в шину

S на выходе одной из шин $z_1, \ldots, z_8$, соответствующей поступившей комбинации, появляется сигнал.

В прямоугольном дешифраторе (рис. 70) сигнал на выходе z_i появляется при условии поступления по соответствующим каналам x_i, y_i двух входных сигналов $\bar{x}$ и $\bar{y}$ на вход схемы совпадения. Сигналы, поступающие на входы x_i, y_i, могут вырабатываться в схемах пирамидальных дешифраторов, как это было рассмотрено выше.

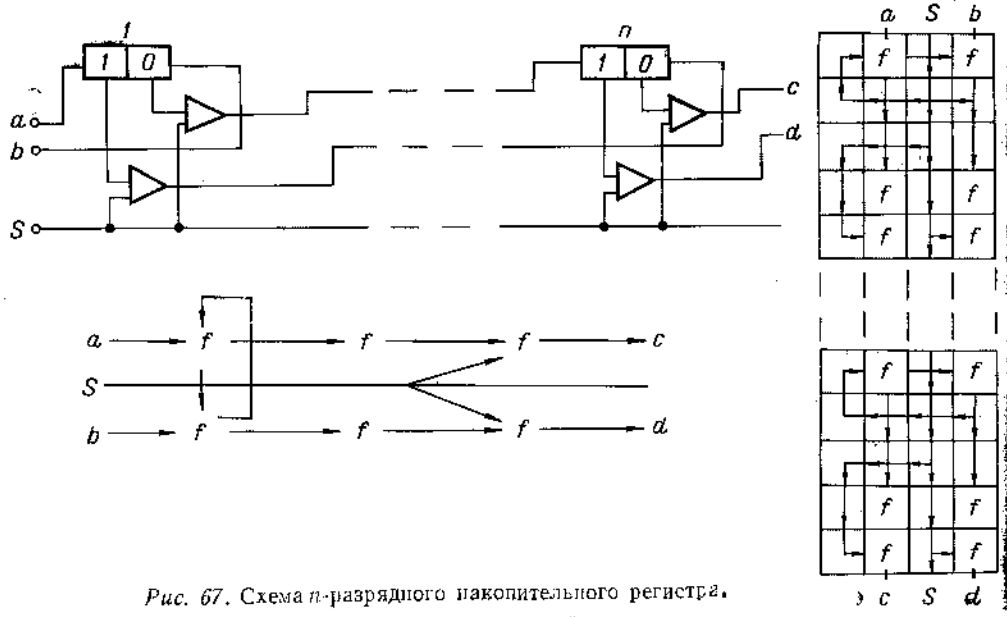
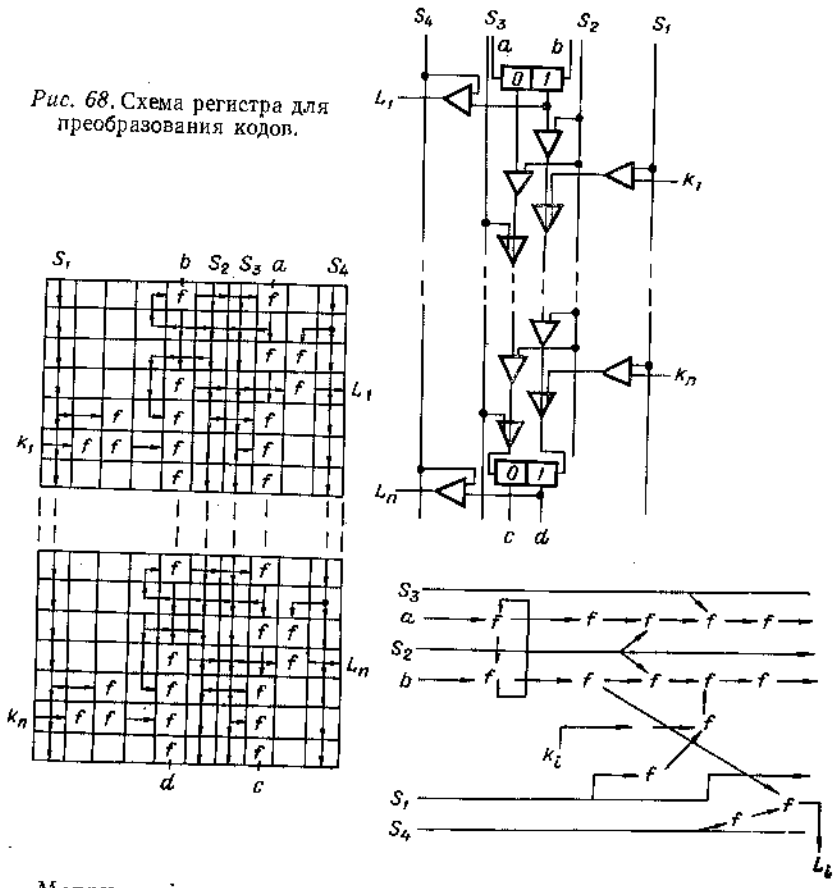


Рис. 67. Схема n-разрядного накопительного регистра.

Реализация устройства памяти в вычислительной среде. Рассмотрим в качестве примера вариант реализации матричной памяти в среде.

Память матричного типа можно представить состоящей из n матриц, где n — число разрядов в числе. Таким образом, каждая матрица содержит одни и те же разряды всех чисел. Каждому числу соответствует ячейка, разряды которой находятся в матрицах с координатами, соответствующими данному числу. На рис. 71 изображена схема и реализация с помощью среды одного разряда ячейки памяти. При выборе данного разряда по шинам x_i, y_i подаются сигналы. Для записи 1 или 0 подается сигнал соответственно по шине 1 или 0. Для считывания достаточно подавать только сигналы выборки x_i, y_i.

Рис. 68. Схема регистра для преобразования кодов.



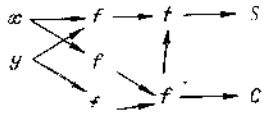
Матрицу i-х разрядов чисел можно выполнить из элементов, реализующих разряд ячейки памяти. Так как шины 1 и 0 общие для всех разрядов, то для их объединения предусматривается дополнительно по одному ряду элементов сверху и снизу общей матрицы, а для объединения всех шин считывания данного разряда — ряд элементов справа общей матрицы. Общий вид матрицы с n^2 разрядами показан на рис. 72.

Реализация схем сумматоров в вычислительной среде. Рассмотрим одноразрядные сумматоры на два и на три входа и комбинационный сумматор с параллельным вводом и переносом.

Одноразрядный сумматор на два входа описывается системой уравнений

$$S = (x \vee y)\,\overline{xy}; \quad C = xy. \tag{6.45}$$

Его структурная схема (рис. 73, *а*) имеет вид:



(6.46)

Одноразрядный сумматор на три входа (рис. 73, *б*) описывается системой уравнений

$$S = xyz \vee (x \vee y \vee z)\,\overline{(xy \vee xz \vee yz)}; \qquad (6.47)$$

$$C = xy \vee xz \vee yz.$$

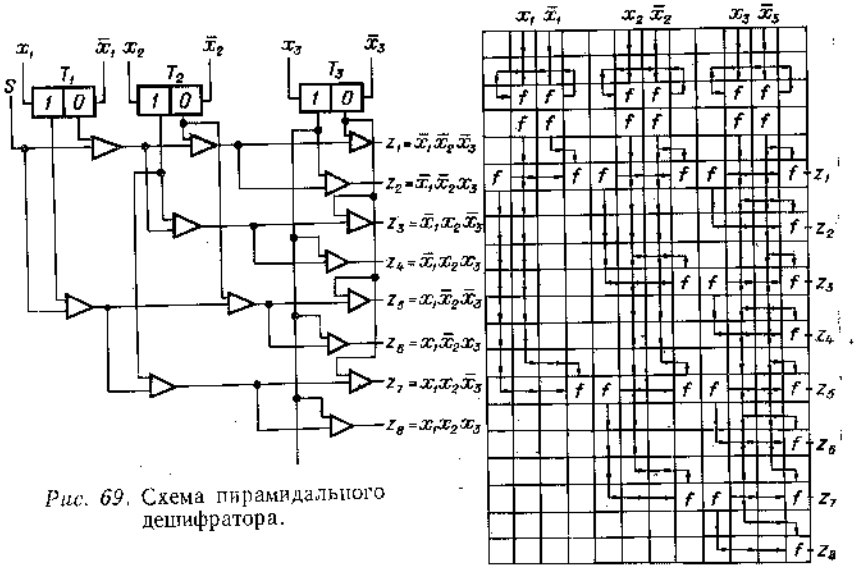


Рис. 69. Схема пирамидального дешифратора.

Комбинационный сумматор с параллельным вводом и параллельным переносом может быть также получен из одноразрядных сумматоров с двумя входами (рис. 74).

Рассмотренные варианты реализации различных схем показывают, что есть возможность построить самые различные схемы вычислительной техники в вычислительной среде посредством соответствующей настройки элементов среды.

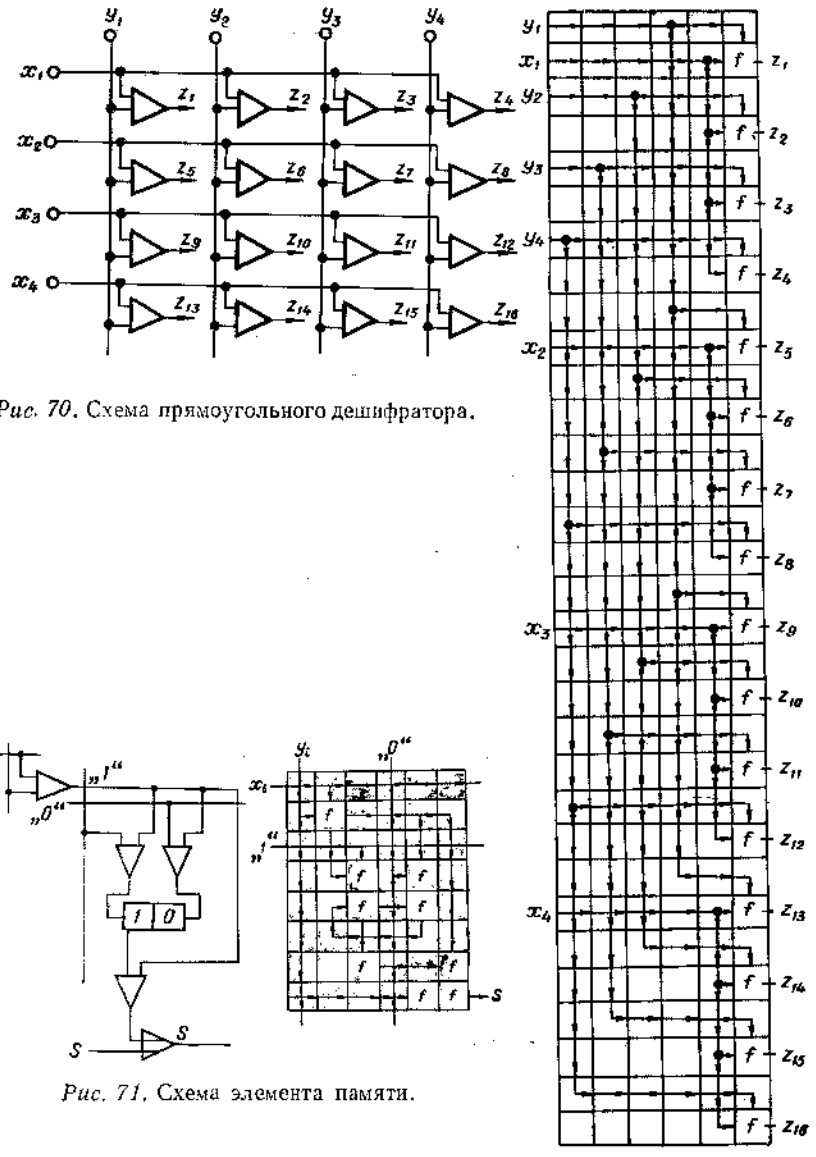


Рис. 70. Схема прямоугольного дешифратора.

Рис. 71. Схема элемента памяти.

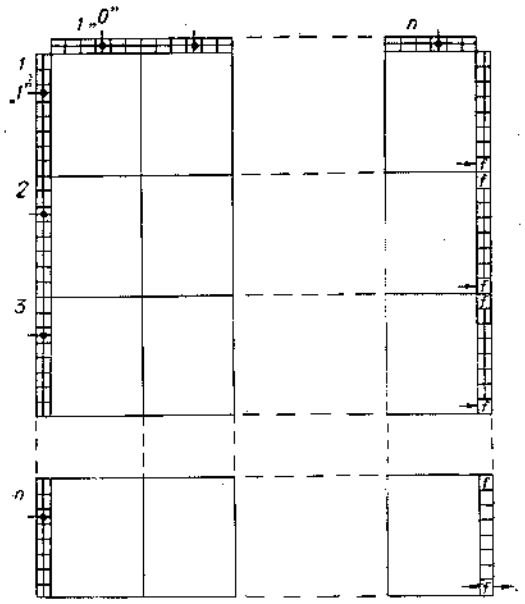


Рис. 72. Схема матрицы памяти.

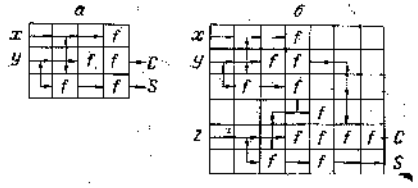


Рис. 73. Схема одноразрядного сумматора.
а — на два входа; *б* — на три входа.

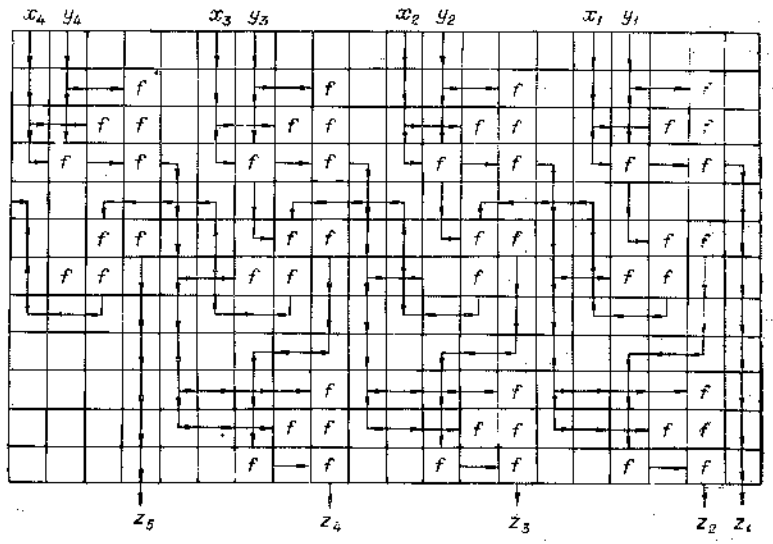


Рис. 74. Схема четырехразрядного сумматора с параллельным вводом и параллельным переносом.

При построении описанных выше примеров схем не преследовалась цель минимизировать количество элементов среды либо число типов элементов. Построение оптимальных схем требует разработать соответствующие методы синтеза схем из элементов вычислительной среды.

6.12. Основные свойства вычислительных сред

Основные достоинства вычислительной среды—высокая технологичность, гибкость, экономичность, надежность.

Высокая технологичность вычислительной среды обусловлена тем, что все элементы среды одинаковы, одинаково соединены друг с другом, просты по своей структуре. Это позволяет предъявлять минимальные требования к технологии, так как производство среды сводится к многократному повторению одного и того же элемента. Немаловажно, что при изготовлении вычислительной среды можно допускать известный процент брака. Неисправные участки среды при ее использовании могут быть обойдены путем изменения конфигурации схемы с помощью операции настройки.

Гибкость среды обусловлена возможностями настройки ее элементов на выполнение различных функций. Это позволяет реализовать схемы любой сложности—различные типы универсальных и специализированных машин. Вычислительная среда обладает возможностью быстрой перестройки ее отдельных частей в процессе решения задачи. Это позволяет практически решать вопрос о создании структур самоорганизующихся систем.

Экономичность среды. Хотя при реализации схем в вычислительной среде затраты элементов в несколько раз больше, чем при использовании обычных элементов и сами элементы сложнее, эти недостатки перекрываются другими достоинствами среды: высоким коэффициентом работающих элементов, так как из одних и тех же элементов с помощью настройки можно создавать различные схемы по мере необходимости, а не делать их заранее; низкой стоимостью элементов среды, обусловленной однородностью технологического процесса и возможностью использования низкочастотных схем элементов среды.

Высокая надежность схем из элементов среды обеспечивается возможностью автоматического ремонта. Неисправные элементы среды исключаются из схемы путем изменения ее конфигурации с помощью настройки. Кроме того, однородность структуры среды облегчает построение схем с автоматическим исправлением ошибок.

Из всего сказанного можно сделать вывод, что построение вычислительной среды — одно из наиболее перспективных направлений создания микроструктуры универсальных вычислительных систем высокой производительности.

* * *

В первых шести главах показана необходимость достижения производительности вычислительной техники свыше 10^9 *опер/сек*; обосновано утверждение, что наиболее реальный путь решения такой проблемы — это применение ВС, одновременно выполняющих много ветвей вычислений; изложены логические основы построения ВС высокой производительности на макро- и микроуровнях.

Создание подобных ВС — сложная проблема, которую трудно решить в короткий срок даже при больших затратах сил и средств. Для ее реализации можно предложить поэтапный план, предусматривающий использование на практике результатов каждого этапа.

1 этап. А. Разработка ВС малой и средней производительности на базе серийных ЭВМ. Под руководством авторов конструкторским бюро во главе с Г. П. Лопато была разработана ВС «Минск-222» из 2—16 машин «Минск-2/22». Эксплуатация

первого образца ВС, вступившего в строй в апреле 1966 г., подтвердила правильность теоретических основ, изложенных в предыдущих главах. Одновременно выяснилось, что ВС «Минск-222» представляет практический интерес. По сравнению с разрозненными ЭВМ система обладает новым качеством — возможностью решения более сложных задач. Кроме того, многие задачи ВС из M машин решает в kM раз быстрее, чем одна ЭВМ ($k \geqslant 1$). Производительность ВС может наращиваться по мере необходимости простым подключением новых ЭВМ.

Оказалось, что ВС «Минск-222» дешевле ЭВМ той же производительности. Затраты на системные устройства ничтожны (менее 0,5% стоимости машин). Наладка ВС проще и сводится фактически к наладке отдельных небольших машин. (Наладка первого образца системы заняла менее 10 дней). Проще и эксплуатация ВС. Выход из строя отдельных устройств или даже машин не нарушает работоспособности системы, а только снижает ее производительность. Все это говорит о целесообразности серийного выпуска подобных ВС.

Б. Создание элементов и отдельных блоков вычислительных сред на обычных элементах. Эта задача в основном решена. Разработка макетов сред подтвердила правильность теоретических положений, изложенных в главе 6. Оказалось также, что небольшие макеты из 25—100 элементов могут применяться на практике как логические конструкторы и обучающие машины, а применение макетов из 1000 и более элементов в качестве дополнительных блоков ЭВМ сокращает время решения некоторых типов задач в сотни раз.

2 этап. А. Построение ВС на 10^7—10^8 *опер/сек* из микроминиатюрных ЭВМ.

Б. Разработка микроминиатюрных блоков вычислительных сред.

Состояние вычислительной техники и технологии микроминиатюризации позволяет надеяться на завершение этого этапа в 3—5 лет.

3 этап. Создание ВС производительностью свыше 10^9 *опер/сек* на основе вычислительной среды. Успех этого этапа зависит от технологии массового изготовления элементов среды. Изложенные в предыдущих главах теоретические результаты, подкрепленные практическими работами 1 этапа, делают вполне реальными требования к рабочей частоте, к уровню микроминиатюризации, к проценту выхода годных элементов. От технологии требуется многократное воспроизведение одного элемента и одного типа соединений. Это позволяет надеяться на решение данной сложнейшей задачи в недалеком будущем.

Глава 7

МЕТОДИКА РЕШЕНИЯ ЗАДАЧ НА УНИВЕРСАЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

7.1. Особенности решения задач на УВС

УВС по сравнению с обычными ЭВМ обладают двумя основными особенностями, которые сказываются на решении задач: одновременностью выполнения большого числа операций и возможностью программного изменения структуры как самих элементарных машин, так и связей между ними.

Первая особенность (см. 2.4) налагает, вообще говоря, ограничения на задачи, решаемые на УВС. Однако на практике это ограничение не имеет сколько-нибудь большого значения, так как на УВС не могут решаться только такие задачи, в процессе решения которых требуется выполнить последовательность более чем из 10^{10}—10^{13} операций, причем каждую из этих операций нельзя начать до окончания предыдущей. Авторам пока не известно ни одной практической задачи подобного рода. Трудно предположить, что значительное число таких задач появится в будущем. С меньшим же числом операций (чем 10^{10} *опер/сек*) могут справиться и обычные ЭВМ. Следует отметить, что поскольку для каждой задачи существует обычно не единственный метод решения и тем более не один алгоритм, то указанное ограничение может быть обойдено путем выбора соответствующего алгоритма.

Вопрос о свойствах алгоритмов с точки зрения возможности представления процесса их реализации в виде большого числа параллельно выполняемых операций до сих пор изучался мало. Основное внимание уделялось алгоритмам, рассчитанным на последовательное выполнение операций. Некоторые подходы к рассмотрению алгоритмов с параллельными ветвями вычислений (параллельных алгоритмов) и особенностей их реализации на вычислительных системах были рассмотрены ранее [1]. В данной главе в след за [2] содержится детализация и дальнейшее развитие этих подходов, иллюстрируемых в гл. 8 примерами решения на УВС наиболее важных классов задач.

Вторая особенность УВС, программное изменение ее структуры, сейчас применяется мало. Какой-либо теории использования подобных устройств еще нет. Программное управление структурой УВС открывает широкие возможности для увеличения производительности УВС путем приспособления УВС под любую задачу и облегчает программирование задач в нынешнем понимании этого термина благодаря возможности записывать программу каждой задачи на наиболее подходящем для нее языке.

Преимущества переменной структуры: возможность произвольно изменять число ЭМ, увеличивать объем памяти ЭМ либо усложнять их логические схемы для выполнения более сложных команд, менять коммутации между ЭМ и т. п., несомненны и, как мы увидим на конкретных задачах, могут давать существенный эффект даже при составлении программ настройки вручную простейшим, не претендующим на оптимальность способом. Безусловно, разработка оптимальных методов настройки структуры под данный алгоритм решения задачи может дать больший эффект. В этой области, однако, трудно ожидать быстрых результатов, так как возникающая проблема в качестве составных элементов содержит проблему оптимального программирования и проблему оптимального синтеза вычислительных систем, каждая из которых еще далека от завершения.

В качестве задач, возникающих при этом, следует назвать выбор списка операций для каждой ЭМ (исходя из конкретного алгоритма); составление плана распределения частей вычислительного процесса между ЭМ и их объединениями на различных иерархических уровнях; распределение потоков информации по каналам связи и составление программы работы УВС.

Все эти задачи тесно связаны друг с другом и должны решаться совместно, как это делается в сложных системах [3], самой УВС. Для решения первой задачи должны быть разработаны алгоритмы, которые анализировали бы решаемую задачу и на этой основе определяли оптимальный набор команд каждой ЭМ. Вторая задача по своему характеру аналогична задачам линейного и динамического программирования. Для ее решения, по-видимому, можно будет в значительной мере использовать готовый математический аппарат [4, 5]. Задача распределения потоков информации между каналами связи сходна с транспортными задачами, рассматриваемыми в математической экономике. Для них также имеется развитый математический аппарат [6, 7]. Четвертая задача включает в себя программу настройки, программу работы каждой ЭМ и УВС в целом, программу обмена информацией между ЭМ и внешними объектами.

Сейчас, когда мы еще далеки от создания оптимальных методов решения задач подобного рода, приходится ограничиваться

теми приемами и методами, которые выработаны в процессе ручного программирования и ручного синтеза вычислительных машин и систем. Однако, как говорилось выше, этот путь уже позволяет достаточно эффективно решать на УВС практические задачи.

7.2. Основные понятия и определения

1. Согласно А. А. Ляпунову [8], любой алгоритм может быть представлен в операторной форме или в терминологии Ю. И. Янова [9] — в виде логической схемы. Логическая схема алгоритма представляет собой конечную строчку, составленную из заданного набора операторов $(A_1, A_2, \ldots, A_n)$, предикатов и двух вспомогательных символов $\underset{i}{\lfloor}$, $\underset{i}{\rfloor}$ $(i = 1, 2, \ldots)$, называемых соответственно левой и правой полускобками. При этом строчка $A_{i_1}A_{i_2}\ldots A_{i_s}$ означает, что должны быть последовательно выполнены операторы $A_{i_1}, A_{i_2}, \ldots, A_{i_s}$. В строчке $A_{i_1}\alpha\underset{m}{\lfloor} A_{i_2}\ldots\underset{m}{\rfloor} A_{i_s}\ldots$ α — некоторый предикат, означающий, что после выполнения оператора A_{i_1}, при $\alpha = 1$, должен быть выполнен оператор A_{i_2}, стоящий непосредственно за левой полускобкой, при $\alpha = 0$ — оператор A_{i_s}, стоящий за правой полускобкой.

Часто употребляют более наглядную запись логической схемы $A_{i_1}\alpha\overset{\downarrow}{\underset{\uparrow}{\ }}A_{i_2}\ldots A_{i_s}$. При этом стрелку, соответствующую $\alpha = 1$, помещают над строчкой, а стрелку, соответствующую $\alpha = 0$, — под строчкой. Стрелку, соответствующую переходу к соседу справа, обычно опускают. Если переход от какого-либо оператора к его соседу справа отсутствует, то их принято разделять точкой с запятой.

2. На операторы $A_1, A_2, \ldots, A_n$ и предикаты не налагаются какие-либо ограничения, кроме их выполнимости [1]. Вопрос о выполнимости оператора и алгоритма в целом требует уточнения. В теории алгоритмов под требованием выполнимости алгоритма понимается, что он может быть реализован за конечное число шагов, на каждом из которых выполняется какая-либо элементарная операция из заданного набора. При этом вопрос о самом числе шагов игнорируется. На несовершенство такого подхода применительно к теории автоматов указал еще Дж. Нейман [10]. Конечность числа шагов еще не означает практической реализации данного алгоритма, так как необходимо, чтобы это

[1] Далее для краткости мы будем пользоваться термином «оператор», подразумевая под этим и предикаты, если это не оговаривается особо.

число было не только конечно, но и меньше некоторого числа $N_{\text{пр}}$, определяемого предельным количеством элементарных операций, которое может быть осуществлено на данном уровне вычислительной техники. Кроме того, в теории алгоритмов [11] обычно не налагается ограничение на предельный объем памяти $V_{\text{пр}}$, потребный для запоминания исходной и промежуточной информации, т. е. считается, что память может быть сколь угодно большой. На практике величина $V_{\text{пр}}$ ограничена достигнутым техническим уровнем. Величины $N_{\text{пр}}$ и $V_{\text{пр}}$ не постоянны и увеличиваются по мере появления более производительных ЭВМ и ВС. Эти величины нельзя рассматривать оторванно друг от друга, поэтому, называя величины предельного числа операций $N_{\text{пр}}$ и предельный объем памяти $V_{\text{пр}}$, необходимо указывать конкретное техническое средство, их реализующее. Кроме $N_{\text{пр}}$, требуется также указать допустимое время работы над одним алгоритмом (1.2). Таким образом, можно для каждого вычислительного устройства M_l указать $N_{\text{пр}l}$ — предельно выполнимое им число операций, содержащихся в списке команд, и $V_{\text{пр}l}$ — предельный объем хранимой информации. Записав алгоритм в виде последовательности команд данного вычислительного устройства (программы), подсчитав общее число команд N, которое должно быть выполнено для его реализации, и наибольший объем хранимой информации V, а затем сравнив их соответственно с $N_{\text{пр}l}$ и $V_{\text{пр}l}$, можно сказать, реализуем ли данный алгоритм данным вычислительным устройством. Условимся в этом случае говорить, что данный алгоритм *выполним* данным вычислительным устройством при

$$N \leqslant N_{\text{пр}l};\ V \leqslant V_{\text{пр}l} \tag{7.1}$$

и *невыполним*, если хотя бы одно из условий (7.1) не выполняется. Алгоритм, невыполнимый ни одним из существующих вычислительных устройств, назовем *практически невыполнимым*.

3. В некоторых случаях удобно, чтобы в схемах алгоритмов правая полускобка всегда была правее левой полускобки с тем же индексом, т. е. чтобы логические схемы алгоритмов не имели петель.

Очевидно, что любая логическая схема практически выполнимого алгоритма может быть преобразована в беспетлевую. Действительно, для этого достаточно повторить всю часть схемы, заключенную между полускобками, принадлежащими данному предикату, столько раз, сколько это требуется по условию задачи. Из практической выполнимости данного алгоритма вытекает, что такое число повторений должно быть заведомо меньше $N_{\text{пр}}$, что и указывает на справедливость данного утверждения.

При построении беспетлевой схемы из схем с петлями каждая петля с известным числом циклов се повторения выписывается столько раз, сколько имеется циклов. Если число циклов является функцией от результатов вычислений, то в этом случае па основании требования выполнимости должно быть известно предельно допустимое число циклов c_m, при котором не нарушаются условия (7.1). Для построения беспетлевой схемы достаточно написать данный цикл в схеме c_m раз. Случай, когда имеются петли с переменным числом циклов, про которые известно, что они в совокупности не нарушают условия (7.1), может быть описан беспетлевой схемой, в которой каждый цикл повторен столько раз, сколько он максимально может встретиться при решении данной задачи. То, что при этом число *записанных* операций в схеме может превысить $N_{\text{пр}}$, не имеет в данном случае существенного значения, так как нас интересует соблюдение условий (7.1) только для числа *выполняемых* операций.

4. Введем понятие зависимости между операторами. Будем говорить, что *оператор B непосредственно информационно зависит от оператора A, если оператор B выполняется над результатом оператора A; оператор A непосредственно управляет операторами B и C, если результат оператора A определяет, какой из этих двух операторов должен выполняться, и если при этом оператор A непосредственно предшествует операторам B и C.*

Заметим, что нередко между операторами могут быть одновременно и информационные и управляющие связи. Пусть, например, требуется выполнить над числом $z - a$ оператор B, если $z < a$, и оператор C — в противном случае. Оператор A (сравнение числа z с a) в этом случае предшествует оператору B по информации и непосредственно управляет операторами B и C.

Для каждого алгоритма можно построить либо две схемы (для информационных и управляющих связей), либо одну объединенную схему с указанием типа связи. Такие схемы могут быть определены в виде графов. Условимся далее для большей наглядности рисовать эти граф-схемы в беспетлевой форме (в виде дерева).

Рассмотрим в качестве примера умножение прямоугольных матриц A и B

$$AB=\begin{bmatrix} a_{11}a_{12}\dots a_{1n} \\ a_{21}a_{22}\dots a_{2n} \\ \dots\dots\dots \\ a_{m1}a_{m2}\dots a_{mn} \end{bmatrix}\times\begin{bmatrix} b_{11}b_{12}\dots b_{1p} \\ b_{21}b_{22}\dots b_{2p} \\ \dots\dots \\ b_{n1}b_{n2}\dots b_{np} \end{bmatrix}=\begin{bmatrix} c_{11}c_{12}\dots c_{1p} \\ c_{21}c_{22}\dots c_{2p} \\ \dots\dots \\ c_{m1}c_{m2}\dots c_{mp} \end{bmatrix} \quad (7.2)$$

где

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{in}b_{nj}\ (i = 1, 2, \dots, m;\ j = 1, 2, \dots, p). \quad (7.3)$$

Для простоты будем считать, что $n = 2^s$ (s — целое). Обозначим $d_{ijk} = a_{ik}\,b_{kj}$ ($i = 1, 2, \dots, m;\quad j = 1, 2, \dots, p;\ k = 1, 2, \dots, n$). Тогда

$$c_{ij} = d_{ij1} + d_{ij2} + \dots + d_{ijn-1} + d_{ijn}. \quad (7.4)$$

Граф-схема информационных связей при вычислении элемента матрицы c_{ij} показана на рис. 75, где вершинами служат либо исходные данные (обозначены точками), либо результаты умножения или сложения двух чисел (обозначены прямоугольниками со знаком соответствующей операции).

Заметим, что хотя по форме граф-схемы связей близки к граф-схемам алгоритмов, предложенных Л. А. Калужниным [12], но они отражают другие свойства алгоритмов. Граф-схемы Л. А. Калужнина устанавливают *порядок* выполнения операторов (в терминологии автора—операторов и распознавателей). В то же время граф-схемы связей показывают, *какие операторы должны быть предварительно выполнены для того, чтобы стало возможным выполнение данного оператора.* В граф-схеме Л. А. Калужнина возможны такие изменения порядка выполнения отдельных операторов, которые никак не отражаются на результатах. Граф-схема связей как раз и устанавливает возможные границы подобных изменений.

Граф-схемы связей аналогичны сетевым графикам, применяемым в системе планирования типа PERT [13—15]. Как те, так и другие указывают, какие операции (работы) должны быть окончены к началу данной операции (работы).

Для ввода в ЭВМ удобно связи между операторами записывать в виде таблицы, в которой для каждого оператора указываются непосредственно предшествующие или управляющие операторы. Естественно, если таких операторов несколько, то они не должны зависеть друг от друга, т. е. ни один из них не предшествует и не управляет ни одним другим.

5. При заданном наборе операторов данный алгоритм может быть представлен различными логическими схемами. Например, выражения

$$\begin{aligned} &P_1 \underset{1}{\lfloor\!\lfloor} \underset{2}{\rfloor\!\rfloor} A_1P_2 \underset{2}{\lfloor\!\lfloor} P_3 \underset{3}{\lfloor\!\lfloor} \underset{1}{\rfloor\!\rfloor} A_2 \underset{3}{\rfloor\!\rfloor} A_3A_4A_5\ \text{Я}; \\ &\overline{P}_1 \underset{1}{\lfloor\!\lfloor} A_2P_3 \underset{2}{\lfloor\!\lfloor} \underset{1}{\rfloor\!\rfloor} \underset{2}{\rfloor\!\rfloor} A_1P_2 \underset{3}{\lfloor\!\lfloor} \underset{2}{\rfloor\!\rfloor} A_3A_5A_4\ \text{Я} \end{aligned} \quad (7.5)$$

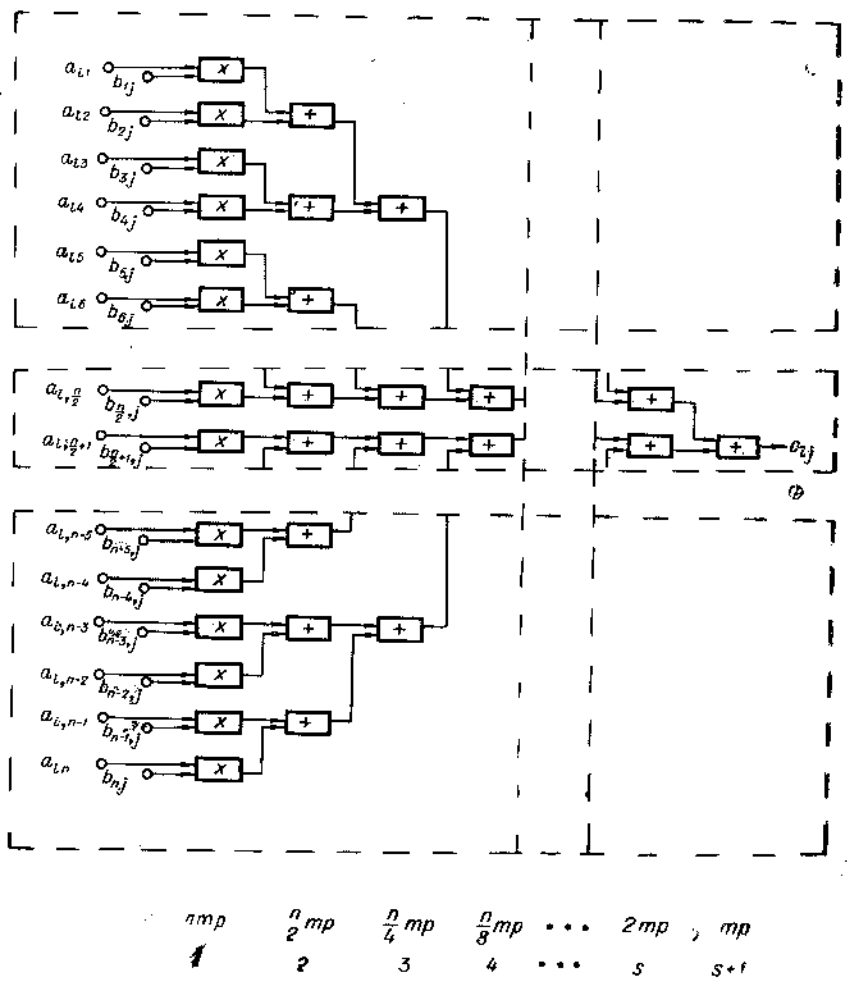


Рис. 75. Граф-схема информационных связей при умножении прямоугольных матриц ($(m \times n)$ и $(n \times p)$).

с предикатами P_1, $\overline{P}_1$ и P_2, тождественно равном нулю предикатом P_3 и оператором конца Я представляют различные схемы записей одного и того же алгоритма.

Чтобы отличать различные схемные записи одного алгоритма от схем различных, но эквивалентных по результатам алгоритмов, условимся в том случае, если двум схемным записям соответствует одна и та же схема связей, считать их эквивалентными схемами одного и того же алгоритма и схемами различных алгоритмов в противоположном случае. Нетрудно убедиться, что схемы в приведенном выше примере изображают один и тот же алгоритм, так как им соответствует одна и та же схема связей

(7.6)

где P равно P_1, либо $\overline{P}_1$.

Как видно из примера, схемные записи одного и того же алгоритма могут отличаться друг от друга значениями предикатов и порядком выполнения во времени взаимно независимых операторов.

Обычно используемые наборы операторов избыточны, т. е. последовательность операторов может быть заменена другой, эквивалентной по результатам последовательностью операторов, принадлежащих этому же набору. Условимся, что при такой замене получается новый алгоритм. Этот случай не следует смешивать с заменой некоторых последовательностей операторов одним оператором, который обычно называют *обобщенным*, что нужно рассматривать как подробную и сокращенную формы записи одного и того же алгоритма.

В соответствии с установившейся традицией будем различать схему алгоритма и схему программы. Принципиальное отличие между ними состоит в том, что первая явно не зависит от конкретного устройства, ее реализующего, а вторая приспособлена к данной ЭВМ (или ВС) и может содержать операторы, учитывающие ее особенности, например операторы обмена информацией между оперативной и вспомогательной памятями, изменение содержимого индекс-регистров и т. п. Применительно к УВС условимся в схемы алгоритмов включать операторы обмена информацией между элементарными машинами, операторы настройки и обобщенного условного перехода (см. гл. 5).

6. Введем некоторые определения. При этом будем использовать термин «операция», подразумевая, что все сказанное может быть отнесено к оператору, реализуемому данной операцией.

Простой операцией (или просто *операцией*) будем называть одно- или двухместную операцию над m-разрядными двоичными числами (или кодами), которая не превышает по сложности операции умножения либо деления двух m-разрядных чисел.

Под *сложностью* операции будем понимать наименьшее число функций $f_1(x_1, x_2) = \overline{x_1} \vee \overline{x_2}$ (штрих Шеффера), либо $f_2(x_1, x_2) = \overline{x_1} \cdot \overline{x_2}$ (стрелка Пирса), необходимых для ее реализации за m рабочих тактов. Под рабочим тактом понимается дискретное автоматное время $t = 0, 1, 2, \ldots$, под m — наибольшее количество двоичных разрядов в числах, над которыми выполняются операции. В дальнейшем рассмотрении будем предполагать, что величина m фиксирована.

Возможность представления любой операции с помощью функций штрих Шеффера, либо стрелки Пирса следует из свойства полноты этих функций. В число простых операций попадают все операции, входящие в список команд обычных ЭВМ, за исключением $\sqrt{x}$, $\log x$ и т. п.

Множество простых операций содержит подмножества, элементы которых образуют полный набор логических функций. Например, функция отрицания и конъюнкция двух чисел, функция отрицания и дизъюнкция двух чисел, штрих Шеффера, стрелка Пирса. Следовательно, *любой алгоритм может быть представлен с помощью совокупности простых операций.*

При заданном наборе назовем *кортежем простых операций*, или просто *кортежем*, любую последовательность простых операций, в которой каждая операция может зависеть только от исходных данных и результатов предшествующих операций.

Очевидно, что при заданном наборе простых операций данный алгоритм может быть представлен различными кортежами простых операций, отличающимися порядком выполнения во времени некоторых взаимно независимых операций.

7. Совокупность *одновременно* и *независимо* друг от друга выполняемых простых операций будем называть p-операцией.

Число входящих в нее простых операций назовем *высотой* l p-операции.

Любую последовательность p-операций, в которой каждая операция, входящая в данную p-операцию, может зависеть только от исходных данных и результатов операций, входящих в предшествующие p-операции, будем называть *кортежем p-операций*, или *p-кортежем*. Каждой p-операции присвоим номер, соответствующий ее месту в последовательности.

Число p-операций, входящих в p-кортеж, назовем его *длиной* h, а наибольшую высоту образующих его p-операций — его *шириной* L:

$$L = \max(l_1, l_2, \ldots, l_h). \tag{7.7}$$

Рис. 75 без изображения связей можно рассматривать как некий p-кортеж, состоящий из $s + 1$ p-операций (длина p-кортежа), первая из которых состоит из n простых операций (ширина p-кортежа), вторая из $n/2$ и т. д. и $s + 1$-я содержит одну операцию. (В полной задаче умножения матриц число операций в каждой из p-операций больше в mp раз.)

Суммарное число простых операций, входящих в p-операции, назовем *объемом p-кортежа*

$$\mathcal{L} = \sum_{j=1}^{h} l_j. \tag{7.8}$$

Зависимость между высотой p-операции и ее номером в p-кортеже будем называть *функцией пошагового распределения*

$$l_j = \varphi(j), \tag{7.9}$$

где j пробегает последовательность значений 1, 2, ..., h. Для рассмотренного примера функция φ p-кортежа, показанного на рис. 75, изображена на рис. 76.

Будем называть p-кортеж *сжатым*, если в нем первая p-операция образована всеми операциями, зависящими только от исходных данных, а любая из последующих p-операций состоит из тех и только тех операций, которые зависят хотя бы от одной операции, входящей в непосредственно предшествующую p-операцию. Именно такой p-кортеж изображен на рис. 75. Сжатый p-кортеж удобно совмещать с граф-схемой связей между операциями данного алгоритма.

Если в каждой из p-операций p-кортежа содержится только по одной простой операции, то такой p-кортеж будем называть *вырожденным*.

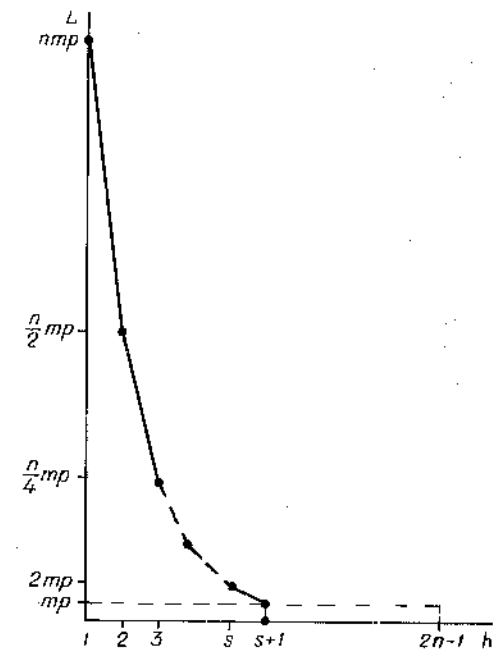


Рис. 76. Функция пошагового распределения для сжатого p-кортежа при умножении прямоугольных матриц.

8. Алгоритм, представленный в виде кортежа p-операций, будем называть *параллельным алгоритмом*, или *p-алгоритмом*.

Алгоритм может быть представлен различными p-кортежами, множество которых будем называть *семейством* p-кортежей данного p-алгоритма.

В соответствии с определением алгоритма все p-кортежи семейства могут отличаться друг от друга только иным распределением операций между p-операциями и числом самих p-операций. Ясно, что все p-кортежи семейства имеют один и тот же объем.

Очевидна справедливость утверждения, что *каждое семейство p-кортежей содержит единственный сжатый* p-кортеж. Ему соответствует наименьшая длина h_m.

Наибольшая длина $h_{\text{пр}}$ из всех p-кортежей семейства у вырожденного p-кортежа.

Семейство p-кортежей отражает особенности соответствующего p-алгоритма, поэтому можно говорить о наименьшей длине h_m p-алгоритма, наибольшей его длине $h_{\text{пр}}$, совпадающей с его объемом $\mathcal{L}$, ширине p-алгоритма при заданной его длине и т. п.

Если семейство состоит только из одного вырожденного p-кортежа, то такой p-алгоритм будем называть *вырожденным*.

9. Поставим в соответствие первой p-операции p-кортежа некоторую исходную информацию v_0, второй p-операции — информацию v_1, образованную из результатов выполнения первой p-операции и той части исходной информации v_0, которая потребуется для выполнения всех последующих p-операций. Аналогично v_j есть сумма объема информации, полученного при выполнении p-операции Q_j, и той части объема информации v_{j-1}, которая потребуется для последующих p-операций, считая с Q_{j+1}. Назовем соответствующие объемы информации v_j, измеряемые в двоичных единицах, *существенными объемами информации p-операций*.

Наибольший из существенных объемов информации среди всех p-операций p-кортежа назовем *существенным объемом информации p-кортежа*:

$$V = max\ (v_0,\ v_1,\ \ldots,\ v_h). \tag{7.10}$$

10. Рассмотрим некоторый p-кортеж длиной h и шириной L, состоящий из p-операций высотой $l_1, l_2, \ldots, l_h$. Дополним p-операции пустыми операциями с тем, чтобы все они стали высотой L, а p-кортеж — прямоугольным. Под *пустой* операцией будем понимать операцию, результат которой для выполнения данного алгоритма безразличен.

Операции, входящие в p-кортеж, расположены теперь в виде прямоугольной матрицы с h перенумерованными столбцами и L строками, которые условимся нумеровать сверху вниз. Каждая строка такой матрицы представляет собой кортеж простых операций.

Из определения p-операции следует, что образующие ее простые операции могут переставляться между собой произвольным образом, тем не менее p-операции, а значит и p-кортеж, считаются одними и теми же. Очевидно, что число возможных представлений p-операции высотой L с числом пустых операций $L - l_j$ равно $L!/(L - l_j)!$. При этом пустые операции не различаются между собой.

Все возможные кортежи из h простых операций, получающиеся при таких перестановках, образуют *множество* кортежей данного p-кортежа. Число элементов множества, как нетрудно видеть,

$$\prod_{j=1}^{h} b_j,\ b_j = \begin{cases} l_j & \text{при } l_j = L; \\ l_j + 1 & \text{при } l_j < L. \end{cases} \tag{7.11}$$

Из этого множества выберем подмножества из L кортежей, таких, что каждая операция p-операций входила бы в данное подмножество один и только один раз. Общее число подмножеств, очевидно, равно

$$\frac{(L!)^h}{L \prod_{j=1}^{h} (L - l_j)!}. \tag{7.12}$$

Нетрудно видеть, что каждое такое подмножество определяет рассматриваемый p-кортеж. Такие подмножества кортежей в дальнейшем будем называть *покрытиями* данного p-кортежа.

11. Рассмотрим кортежи k_i и k_j, принадлежащие некоторому покрытию. Будем говорить, что k_i *не зависит* от k_j, если ни одна операция кортежа k_i непосредственно не зависит ни от одной из операций кортежа k_j. В качестве *меры зависимости* k_i от k_j возьмем число операций кортежа k_j, от которых непосредственно зависят операции кортежа k_i. Это число обозначим символом c'_{ij}.

Если рассматривается все покрытие, то числа c'_{ij} образуют квадратную матрицу порядка L. Примем, что диагональные элементы матрицы $c'_{ij} = 0$ $(i = j)$. Сумма

$$C'_i = \sum_{j=}^{L} c'_{ij}. \tag{7.13}$$

дает для k_i общее число операций остальных кортежей покрытия, от которых непосредственно зависят операции данного кортежа. Сумму

$$C' = \sum_{i=1}^{L} C'_i \tag{7.14}$$

назовем *связностью покрытия по операциям*, а матрицу (c'_{ij}) — *матрицей связности по операциям*.

Распределим исходную информацию v_0 между кортежами покрытия. В процессе вычислений, возможно, придется передавать исходную информацию из кортежа k_j в кортежи k_i. Обозначим количество различных передаваемых кодов c''_{ij}. Числа c''_{ij} образуют квадратную матрицу порядка L, у которой элементы $c''_{ij} = 0$ $(i = j)$. Матрицу (c''_{ij}) условимся называть *матрицей связности покрытия по исходной информации*, а

$$C'' = \sum_{i,\,j=1}^{L} c''_{ij} \tag{7.15}$$

— *связностью покрытия по исходной информации*.

Матрицу

$$(c_{ij}) = (c'_{ij}) + (c''_{ij}) \tag{7.16}$$

будем называть *матрицей связности*, а

$$C = C' + C'' \tag{7.17}$$

— *связностью покрытия*.

Представление p-кортежа в виде покрытия эквивалентно представлению процесса выполнения данного p-алгоритма в виде множества одновременно (вообще говоря, зависимо) выполняемых последовательных алгоритмов, обмен информацией между которыми описывается матрицей связности.

12. Хотя предыдущее рассмотрение велось без учета предикатов и, следовательно, возможных разветвлений в процессе выполнения p-алгоритма, оно может быть легко распространено и на этот случай. Действительно, пусть предикат α обусловливает выполнение либо p_1-подкортежа с шириной L_1 и длиной h_1, либо p_2-подкортежа с шириной $L_2 = L_1$ и длиной $h_2 \leqslant h_1$, каждый из которых уже не содержит предикатов (под p-подкортежем понимается последовательность p-операций, входящая в p-кортеж). Тогда условимся брать в качестве основного p-подкортежа

p_1-подкортеж с большей длиной. Иными словами, общая длина p-кортежа, содержащего разветвления, будет определяться более длинными ветвями. Объем p-кортежа будет при этом определяться числом операций, входящих в длинные ветви. Число выполняемых p-операций в этом случае будет обычно меньше длины p-кортежа. Поправочный коэффициент можно при необходимости вычислить, если знать вероятности, с которыми предикаты принимают то или другое значение, либо, что бывает зачастую проще, определять его путем моделирования.

13. Будем говорить, что *схема* p-алгоритма решения задачи построена, если определены:

1) p-кортеж, представляющий данный алгоритм, в том числе функция пошагового распределения;
2) покрытие p-кортежа простыми кортежами;
3) распределение исходной информации между кортежами;
4) матрицы связности (c_{ij}) и связности по операциям (c'_{ij}).

7.3. Характеристическая функция p-алгоритма

1. В п. 7.2 было указано на неоднозначность представления p-алгоритма p-кортежами, множество которых мы назвали семейством p-кортежей. Рассмотрим этот вопрос подробнее.

Все p-кортежи семейства, имеющие длину h, будем называть *h-подсемейством* p-кортежей.

p-Кортеж h-подсемейства назовем *минимальным по ширине*, если он имеет наименьшую ширину из всех p-кортежей подсемейства

$$L^* = \min_r L_r, \tag{7.18}$$

где r — индекс p-кортежа h-подсемейства.

L^* будем называть *минимальной шириной h-подсемейства*.

Все p-кортежи семейства, имеющие ширину L, назовем *L-подсемейством* p-кортежей.

p-Кортеж L-подсемейства будем называть *минимальным по длине*, если он имеет наименьшую длину из всех p-кортежей подсемейства

$$h^* = \min_q h_q, \tag{7.19}$$

где q — индекс p-кортежа L-подсемейства.

h^* назовем *минимальной длиной L-подсемейства*.

p-Кортеж будем называть *оптимальным*, если он одновременно минимален и по длине и по ширине.

Мы уже видели, что длина p-кортежей не может быть меньше длины h_m сжатого p-кортежа. В h_m-подсемействе выделим p-кортежи с минимальной шириной L_m^* и максимальной шириной $\hat{L}_m$. Очевидно, что все L-подсемейства при $L_m^* \leqslant L \leqslant \hat{L}_m$ имеют минимальную длину h_m. L-подсемейства с $L > \hat{L}_m$ хотя, вообще говоря, и могут существовать, но рассматривать их практически не интересно.

Минимальный по ширине в $h_{\text{пр}}$-подсемействе p-кортеж (с $L = 1$) в подсемействе $L = 1$ будет минимальным по длине, а следовательно,— оптимальным.

Таким образом, на плоскости h, L, имеются два множества точек $\mathfrak{M}_1 = \{(h, L^*)\}$ и $\mathfrak{M}_2 = \{(h^*, L)\}$, где $h_m \leqslant h \leqslant h_{\text{пр}}$, $\hat{L}_m \geqslant L \geqslant 1$. Рассмотрим их объединение $\mathfrak{M} = \mathfrak{M}_1 \bigcup \mathfrak{M}_2$.

Теорема 1. *Множество $\mathfrak{M}$ на плоскости h, L представляет собой монотонную последовательность точек*

$$a_1(h_1, L_1) = a_m(h_m, \hat{L}_m), \ldots, a_l(h_l, L_l), a_{l+1}(h_{l+1}, L_{l+1}), \ldots$$
$$\ldots, a_n(h_n, L_n) = a_{\text{пр}}(h_{\text{пр}}, 1),$$

где $h_{l+1} \geqslant h_l$; $L_{l+1} \leqslant L_l$.

Докажем сначала некоторые леммы.

Лемма 1. *Любую p-операцию, состоящую более чем из одной операции, можно заменить последовательностью из двух или более p-операций, составленных из тех же операций, что и исходная.* Справедливость леммы очевидна и непосредственно вытекает из взаимной независимости операций, входящих в одну p-операцию.

Лемма 2. *В каждом h-подсемействе $h_m \leqslant h \leqslant h_{\text{пр}}$ имеется не менее одного p-кортежа.* Действительно, рассмотрим сжатый p-кортеж с длиной h_m и шириной L_m. В согласии с леммой 1 заменим какую-либо из его p-операций, содержащую более одной операции, двумя и тем самым увеличим длину p-кортежа на 1. Применяя эту процедуру многократно, получим последовательность p-кортежей, начиная от сжатого и кончая вырожденным, каждый из которых будет иметь длину, большую на единицу, чем у предыдущего, что и доказывает лемму.

Лемма 3. *В каждом L-подсемействе $1 \leqslant L \leqslant \hat{L}_m$ имеется не менее одного p-кортежа.* Возьмем за исходный p-кортеж с шириной $\hat{L}_m$. В соответствии с леммой 1 все p-операции с высотой $\hat{L}_m$ заменим на две так, чтобы хотя бы одна из p-операций осталась

высотой $\hat{L}_m - 1$. Повторяя эту процедуру многократно, получим последовательность p-кортежей с шириной от $\hat{L}_m$ до 1, каждый из которых будет иметь ширину на единицу меньшую, чем у предыдущего, что и требовалось доказать.

Лемма 4. *L-подсемейство с минимальной длиной h^* содержит p-кортежи длиной h^*, $h^* + 1$, $h^* + 2$, ..., $h_{\text{пр}} - L + 1$,* где $h_{\text{пр}}$ — длина вырожденного p-кортежа. Действительно, выделим в минимальном по длине p-кортеже p-операцию высотой L и заменим любую другую p-операцию с высотой больше 1 двумя в согласии с леммой 1. Это даст нам p-кортеж длиной $h^* + 1$. Повторяя эту процедуру, получим p-кортеж длиной $h^* + 2$ и т. д. до тех пор, пока все p-операции, кроме выбранной, не станут высотой 1. Очевидно, что длина p-кортежа при этом будет равна $h_{\text{пр}} - L + 1$, и это и есть наиболее длинный p-кортеж в L-подсемействе. Лемма доказана.

Лемма 5. *h-подсемейство с минимальной шириной L^* содержит p-кортежи с шириной $L^*, L^* + 1, L^* + 2, \ldots, L^* + k$,* где $L^* + k = \min\{\hat{L}_m, h_{\text{пр}} - h + 1\}$. Возьмем за исходный минимальный по ширине p-кортеж h-подсемейства. Перенесем в его первую p-операцию одну за другой операции, входящие в первую p-операцию p-кортежа с шириной $\hat{L}_m$, пока ее высота не станет равной $L^* + 1$. Если этого не произойдет, а все возможные операции уже перенесены, то начинаем строить таким же способом вторую p-операцию и т. д. Если при этом где-либо образуется пустая p-операция, то в согласии с леммой 1 разделим на две любую p-операцию, кроме достраиваемой. Подобное деление возможно, пока имеются p-операции (кроме выбранной) с высотой больше 1, что, очевидно, будет, пока $L^* + k < h_{\text{пр}} - h + 1$. Наибольшая ширина p-кортежа h-подсемейства будет при этом равна $L^* + k = \min\{\hat{L}_m, h_{\text{пр}} - h + 1\}$. Лемма доказана.

Лемма 6. *Множеству $\mathfrak{M}$ не могут одновременно принадлежать три точки:* $a_1(h, L)$, $a_2(h', L)$ и $a_3(h, L')$, где $h' < h$, $L' < L$. Действительно, предположим противное. Тогда, поскольку a_2 принадлежит множеству $\mathfrak{M}$ ($a_2 \in \mathfrak{M}$), то a_1 не принадлежит множеству $\mathfrak{M}_2$ ($a_1 \notin \mathfrak{M}_2$). Аналогично, так как $a_3 \in \mathfrak{M}$, то $a_1 \notin \mathfrak{M}_1$, а следовательно, $a_1 \notin \mathfrak{M}$.

Лемма 7. *Множеству $\mathfrak{M}$ не могут одновременно принадлежать точки* $a_1(h, L)$ и $a_0(h', L')$, где $h' < h \leqslant h_{\text{пр}}$, то $L' < L \leqslant \hat{L}_m$. В самом деле, предположим противное. Тогда по лемме 4 в L'-подсемействе должен быть p-кортеж с длиной $h > h'$, если только $h \leqslant h_{\text{пр}} - L' + 1$. Но из $a_1 \in \mathfrak{M}$ и $L > L'$ следует, что $h \leqslant h_{\text{пр}} - L + 1 < h_{\text{пр}} - L' + 1$, т. е. указанный p-кортеж существует. Это значит, что $a_1 \notin \mathfrak{M}_1$.

Далее, по лемме 5 в h'-подсемействе должен быть p-кортеж шириной $L > \hat{L}'$, если только $L \leqslant \min\{\hat{L}_m, h_{\text{пр}} - h + 1\}$. Но так как $a_1 \in \mathfrak{M}$ и $h > h'$, то $L \leqslant \hat{L}_m$ и $L \leqslant h_{\text{пр}} - h + 1$. Отсюда $L < h_{\text{пр}} - h' + 1$, т. е. и такой p-кортеж существует, а это означает, что $a_1 \notin \mathfrak{M}_2$. Следовательно, $a_1 \notin \mathfrak{M}$. Лемма доказана.

Приступим теперь к доказательству теоремы.

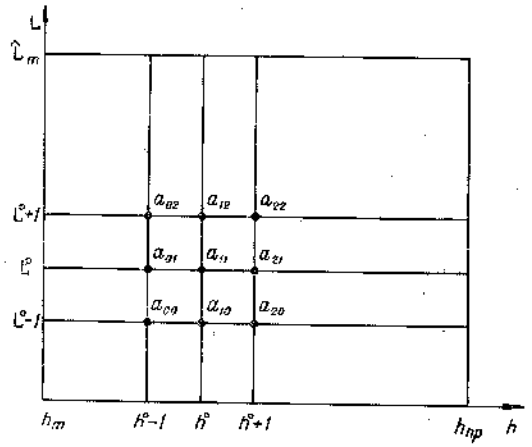


Рис. 77. Определение вида характеристической функции.

Пусть точка $a_{11}(h^0, L^0) \in \mathfrak{M}$, где $h^0 < h_{\text{пр}}$, $1 < L^0$ (рис. 77). Тогда по лемме 7 точки $a_{22}(h^0 + 1, L^0 + 1) \notin \mathfrak{M}$ и $a_{00}(h^0 - 1, L^0 - 1) \notin \mathfrak{M}$. Кроме того, по лемме 6 множеству $\mathfrak{M}$ не могут одновременно принадлежать точка a_{21} и точка a_{20}, а по лемме 7 — точка a_{21} и точка a_{10}. По лемме 2 множеству $\mathfrak{M}$ должна принадлежать хотя бы одна точка с $h = h^0 + 1$, а по лемме 7 в этой точке $L \leqslant L^0$. Тогда это либо одна из точек a_{21} или a_{20} (т. е. утверждение теоремы верно), либо точка с $L < L^0 - 1$, причем при $L^0 = 2$ возможно только первое. Однако по лемме 3 множеству $\mathfrak{M}$ должна принадлежать хотя бы одна точка с $L = L^0 - 1$, а по лемме 7 в этой точке $h \geqslant h^0$. Тогда это либо одна из точек a_{10} или a_{20}, либо обе вместе (в этих случаях утверждение верно), либо $h > h^0 + 1$, причем при $h^0 = h_{\text{пр}} - 1$ возможно только первое. Очевидно, что утверждение может быть неверно только в случае одновременной принадлежности множеству $\mathfrak{M}$ точек $(h^0 + 1,\ L < L^0 - 1)$ и $(h > h^0 + 1,\ L = L^0 - 1)$, но это невозможно по лемме 7.

Таким образом, должна реализовываться одна из четырех возможностей, а именно: множеству $\mathfrak{M}$ принадлежат либо одна из точек a_{21}, a_{20}, a_{10}, либо пара точек a_{10}, a_{20}. В последнем случае по леммам 6 и 7 точки $(h^0,\ L^0 - 2) \notin \mathfrak{M}$, $(h^0 + 1, L' - 2) \notin \mathfrak{M}$.

Условимся считать, что точка (h', L') следует за точкой (h, L), если $h' \geqslant h, L' \leqslant L$. Тогда будем рассматривать четыре указанных варианта как переходы $a_{11}a_{21}$, $a_{11}a_{20}$, $a_{11}a_{10}$, $a_{11}a_{10}a_{20}$. В этих переходах значения h не убывают, а значения L не возрастают.

Применим к точке $a_1 = a_m$ выводы, сделанные для точки a_{11}. В результате получаем переход из двух или трех точек, принадлежащих множеству $\mathfrak{M}$. Затем применяем ту же процедуру к последнему члену перехода и т. д. (На первых шагах до точки $(h_m, \hat{L}_m)$ будут реализовываться переходы только типа $a_{11}a_{10}$.) В результате получаем монотонную последовательность точек. Вследствие монотонности она дойдет либо до значения $h = h_{\text{пр}} - 1$, либо до $L = 2$. Далее могут быть или переходы типа $a_{11}a_{10}$ (при $h = h_{\text{пр}} - 1$), или $a_{11}a_{21}$ (при $L = 2$) и так до точки $(h_{\text{пр}} - 1, 2)$, ибо на линиях $h = h_{\text{пр}}$ и $L = 1$ множеству $\mathfrak{M}$ принадлежит только одна точка, которой и будет заканчиваться данная последовательность.

Очевидно, что все точки множества вошли в последовательность. Действительно, по леммам 2 и 3 на каждой линии h $(h_m \leqslant h \leqslant h_{\text{пр}})$ и L $(1 \leqslant L \leqslant \hat{L}_m)$ содержится хотя бы одна точка последовательности. Таким образом, для любой точки $a\ (h, L)$, не принадлежащей последовательности, найдется по крайней мере одна точка $a'\ (h', L')$ такая, что либо $h' > h$, $L' > L$, либо $h' < h$; $L' < L$ (и в том и в другом случае $a \notin \mathfrak{M}$ по лемме 7) или пара точек $a'\ (h', L')$ и $a''\ (h'', L'')$, таких, что $h = h'$, $L > L'$ и $h > h''$, $L = L''$ (тогда $a \notin \mathfrak{M}$ по лемме 6).

Таким образом, теорема 1 доказана полностью.

Построенную последовательность можно рассматривать как дискретную функцию

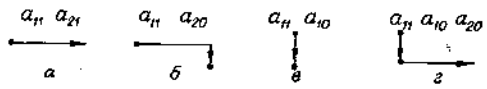
$$L = F(h), \tag{7.20}$$

определенную для $h = h_m, h_m + 1, \ldots, h_{\text{пр}}$ и принимающую значения $\hat{L}_m, \hat{L}_m - 1, \ldots, 1$.

Назовем эту функцию характеристической функцией p-алгоритма.

Условимся изображать характеристическую функцию в виде ориентированного графа, ребра которого представляют собой отрезки, параллельные горизонтальной и вертикальной осям плоскости h, L, при этом каждый из четырех упомянутых пере-

ходов будет изображаться следующим образом:



Тогда характеристическая функция будет иметь вид (рис. 78), где кривая I соответствует суммированию n слагаемых, а кривая II — вычислению таблицы из n чисел, каждое из которых — результат одной операции над исходными числами.

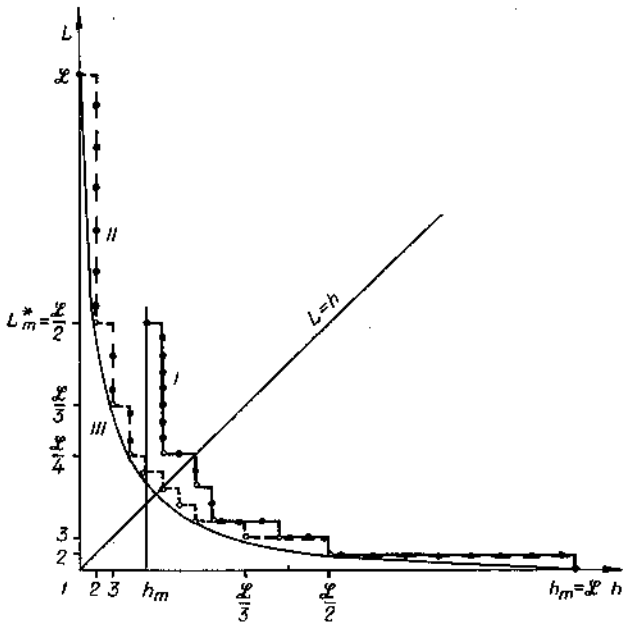


Рис. 78. Примеры характеристических функций.

I — суммирование n чисел; II — вычисление таблицы из n чисел; III — кривая эффективных точек.

Из этих примеров видно, что в характеристической функции могут быть все четыре перехода, упомянутые выше. По лемме 6 после переходов *а* и *г* невозможны переходы *в* и *г*. Примеры остальных 12 пар переходов можно видеть на рис. 78.

Очевидны следующие свойства характеристической функции:

вершины углов, соответствующие переходам *б*, не содержат точек характеристической функции;

точки, соответствующие минимальным по длине p-кортежам, находятся на вертикальных участках, а точки, соответствующие минимальным по ширине p-кортежам,— на горизонтальных;

точки, соответствующие оптимальным p-кортежам, и только они находятся в вершинах углов перехода *г*;

знания всех оптимальных точек и $\hat{L}_m$ достаточно для определения всех других точек характеристической функции.

Характеристическая функция полностью определяется объемом $\mathscr{L}$ p-алгоритма и граф-схемой связей. При одном и том же $\mathscr{L}$ можно указать некоторые особенности характеристических функций:

1) вырожденный p-алгоритм имеет единственную точку ($h_m = h_{пр}$, $\hat{L}_m = 1$), которая, очевидно, оптимальна. Наибольшее число оптимальных точек у характеристической функции p-алгоритма, все операции которого не зависят друг от друга (рис. 78, кривая II). В последнем случае характеристическая функция симметрична относительно диагонали $L = h$, а оптимальными точками являются точки с $h \leqslant d$ или $L \leqslant d$, где d есть округленное до большего целого значения $\sqrt{\mathscr{L}}$;

2) если U — p-алгоритм с независимыми операциями, то для произвольной точки $a(h, L)$ характеристической функции любого другого A p-алгоритма (с зависимыми операциями) справедливы соотношения:

$$\text{при } h = h',\ L \geqslant L';$$
$$\text{при } L = L'',\ h \geqslant h'',$$

где $u'(h', L')$ и $u''(h'', L'')$ — точки характеристической функции p-алгоритма U. Справедливость этого утверждения следует из того, что при фиксированном h (или L) для p-алгоритма U может быть образован p-кортеж с L^* (или h^*) не большей, чем для p-алгоритма A (см. рис. 78).

2. Отношение объема p-алгоритма к произведению длины p-кортежа на его ширину будем называть *эффективностью* данного p-кортежа:

$$\delta = \frac{\mathscr{L}}{hL}. \qquad (7.21)$$

Наибольшая эффективность $\delta = 1$ будет при $L = \mathscr{L}/h$, когда все p-операции имеют одинаковую высоту. Такие p-кортежи и соответствующие им точки характеристической функции будем называть *эффективным и*.

У неэффективных p-кортежей имеются p-операции с высотой $l_j < L$. Если дополнять эти операции до высоты L пустыми, то $hL = \mathscr{L} + \Lambda$, где Λ — общее число пустых операций в p-кортеже,

$$\delta = \frac{\mathscr{L}}{\mathscr{L}+\Lambda} = \frac{1}{1+\Lambda/\mathscr{L}}. \qquad (7.22)$$

Отсюда видно, что δ убывает по гиперболическому закону с ростом пустых операций.

Ясно, что все эффективные точки являются также и оптимальными. На характеристической функции (7.20) им соответствуют точки, лежащие на гиперболе $L = \mathscr{L}/h$ (см. рис. 78, кривая III).

Если характеристическую функцию изобразить в координатах $(L, \mathscr{L}/h)$.

$$\mathscr{L}/h = f(L), \qquad (7.23)$$

то эффективные точки располагаются на прямой $\mathscr{L}/h = L$.

При представлении характеристической функции в виде (7.23) ее точкам соответствует эффективность

$$\delta = \frac{1}{L} f(L). \qquad (7.24)$$

Нетрудно видеть, что поскольку точки характеристической функции при заданном L соответствуют наименьшему h, а при заданном h — наименьшему L, то им будут отвечать наибольшие значения δ в h(или L)-подсемействах.

Для рассматриваемого примера умножения матриц функции $f(L)$ и $\delta(L)$ имеют вид, показанный на рис. 79.

Среди эффективных точек важную роль играет точка с наибольшим значением L, которую будем называть *главной эффективной точкой*, или просто *главной точкой*, и обозначать буквой g. Условимся часть характеристической функции, которой соответствуют $1 \leqslant L \leqslant L_g$, называть *эффективной* областью, а часть, которой соответствуют $L_g < L \leqslant \hat{L}_m$,— *неэффективной*.

Будем называть L-*развертной* p_c-кортежа p-кортеж, образованный путем замены каждой p-операции p_c-кортежа последовательностью p-операций, которые все имеют высоту L, а последняя не больше чем L. Возможность применения L-развертки к p-кортежу с шириной $L_c > L$ вытекает из леммы 1. Условимся в L-развертке дополнять все p-операции до высоты L пустыми операциями.

Очевидно следующее важное утверждение.

Теорема 2. *Пусть дан p-кортеж и функция его пошагового распределения $l_j = \varphi(j)$, $(j = 1, 2, \ldots, h)$. Тогда, если существует общий делитель L чисел l_j, то из данного можно построить эффективный p-кортеж.*

Действительно, построим L-развертку заданного p-кортежа. Так как все l_j кратны L, то каждая из p-операций заменится последовательностью, состоящей из равновысоких (высотой L) p-операций. Таким образом, будет получен прямоугольный p-кортеж с $\delta = 1$.

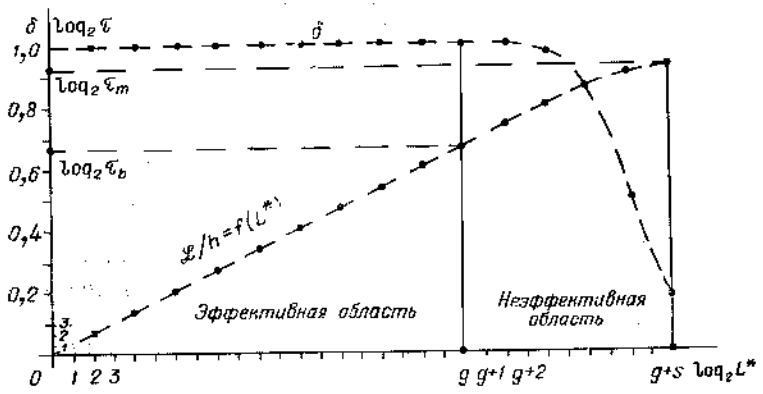


Рис. 79. Вид характеристической функции и изменения коэффициента эффективности δ для умножения матриц.

$$\tau = \frac{\mathscr{L}}{h} = \frac{(2n-1)\,mp}{h}; \quad \tau_g = \frac{(2n-1)\,mp}{2n-1}; \quad \tau_m = \frac{(2n-1)\,mp}{\log_2 n + 1};$$

$$mp = 2^g; \quad n = 2^s.$$

Из этого утверждения вытекает следствие: *если известна какая-либо эффективная точка e (h_e^*, L_e^*) и $L_e^* = L_a L_c$, где L_a и L_c — целые, то характеристическая функция имеет эффективные точки a $(h_a = L_c h_e, L_a)$ и c $(h_c = L_a h_e, L_c)$.*

В рассмотренном примере умножения матриц для $n = m = p = 2^{10}$ эффективные точки будут соответствовать точкам, координаты которых являются степенями двойки, а главной точке будет соответствовать $\mathscr{L}/h_g^* = L_g^* = mp = 2^{20}$ (рис. 79).

Указанным способом можно получить также значения координат точек, для которых δ мало отличается от 1 (*квазиэффективные точки*). Действительно, дополним некоторые p-операции p-кортежа пустыми операциями так, чтобы у нового набора l_j появились общие делители. Тогда, если общее число добавляемых операций $\Lambda \ll \mathscr{L}$, то, согласно (7.22), $\delta \approx 1$.

В частности, если исходный p_e-кортеж был эффективным, а требуется найти p-кортеж с шириной L, которой не кратно L_e^*, то добавим ко всем p-операциям p_e-кортежа λ пустых с тем, чтобы

$$L_e^* + \lambda = dL,$$

где $d > 0$ — целое, $1 \leqslant \lambda < L$. Тогда L-развертка содержит $\Lambda = \lambda h_e^*$ пустых операций и, согласно (7.22),

$$\delta = \frac{1}{1 + \lambda/L_e^*}. \qquad (7.25)$$

Из (7.25) видно, если L_e^* лежит в области больших значении L, например $L_e^* = L_g^*$, так что $L \ll L_e^*$, то $\lambda/L_e^* \ll 1$ и δ будет близко к 1.

Теорема 3. *При L-развертке p_c-кортежа с шириной L_c, когда L_c кратно L, величина коэффициента эффективности не убывает, то есть $\delta \geqslant \delta_c$.*

Действительно, при составлении L-развертки, так как L_c кратно L, то не нужно добавлять пустых операций, поэтому их число в новом p-кортеже по сравнению с p_c-кортежем может только уменьшиться за счет отбрасывания p-операций, состоящих сплошь из пустых операций, откуда на основании (7.22) и следует справедливость неравенства $\delta \geqslant \delta_c$.

Следствие. *Ширина таким образом полученного p-кортежа $h \leqslant \frac{L_c}{L} h_c$.*

3. Рассмотренные свойства L-разверток можно использовать для определения (хотя бы приближенно) вида характеристической функции.

В практических задачах обычно нетрудно составить сжатый p-кортеж и из него получить другие p-кортежи h_m-подсемейства. Если существуют общие делители всех высот p-операций в каких-либо p-кортежах (либо этого можно добиться добавлением небольшого числа пустых операций), то, согласно теореме 2, тем самым определяются эффективные (либо квазиэффективные) точки. Среди этих точек для нас важны точки с большими значениями L (малыми h). Практически достаточно знать хотя бы одну такую точку, которую назовем *базовой* и обозначим буквой b. В частности, базовой точкой может быть главная эффективная точка. В ряде задач выбор базовой точки непосредственно подсказывается видом вычислительного алгоритма. Так, в задачах

линейной алгебры L_b обычно равно порядку матриц, с которыми приходится оперировать.

Из p_b-кортежа строим различные L-развертки. При этом для каждого L получается значение $h \geqslant h^*$, где h^* — значение характеристической функции при заданном L. Таким образом, получается верхнее значение координаты h^*. Нижней границей h^* является значение $(h^*)_u$ характеристической функции p-алгоритма U с независимыми операциями (см. конец п. 1).

Таким образом, получаем полосу, в которой заключена характеристическая функция. Если в области интересующих нас значений L (или h) эта полоса слишком широка, то можно провести построение, взяв в качестве базовой точки другую точку и т. д.

4. p-Кортежи, принадлежащие h_m-подсемейству, в отличие от всех остальных имеют хотя бы по одной цепочке операций длиной, равной длине h_m p-кортежа, в которой каждая операция непосредственно зависит от предшествующей (кроме, естественно, первой). Эта цепочка, собственно, и ограничивает дальнейшее уменьшение длины p-кортежей. Она аналогична *критическому пути* в системе планирования PERT, и мы будем называть ее этим же термином. Обычно в p-кортеже таких цепочек бывает не одна, а несколько, причем у них, как правило, имеются общие операции. Например, в задаче умножения матриц все операции принадлежат критическому пути (см. рис. 75). Операции, принадлежащие критическому пути (и только они), не могут быть перемещены из одной p-операции в другую при переходе от одного p-кортежа h_m-подсемейства к другому.

Будем каждую операцию характеризовать интервалом перемещения, равным разности номеров самой правой и самой левой p-операций, которым может принадлежать данная p-операция в p-кортежах h_m-подсемейства.

Для определения интервала перемещения каждой операции достаточно построить сжатый p-кортеж и аналогичный ему, *сжатый вправо*. Последний строится так. Отнесем к h_m-й p-операции все операции, от которых не зависит ни одна операция данного p-алгоритма; а к любой из предыдущих p-операций отнесем те и только те операции, от которых зависит хотя бы одна из операций входящих в непосредственно следующую за ней p-операцию.

Знание интервала помогает целенаправленно вести перебор p-кортежей h_m-подсемейства.

Существующая аналогия между схемами параллельных алгоритмов и сетевыми графиками системы PERT естественна, так как и те и другие описывают процессы, состоящие из параллельно выполняемых ветвей. Основное отличие между ними в том, что

параллельные алгоритмы предполагается реализовать на УВС, состоящей из одинаковых элементарных машин, в то время как сетевые графики выполняются, как правило, коллективами, специализирующимися на выполнении однотипных работ. В этом отношении сетевые графики сходны со схемами алгоритмов для вычислительных систем из разнородных машин.

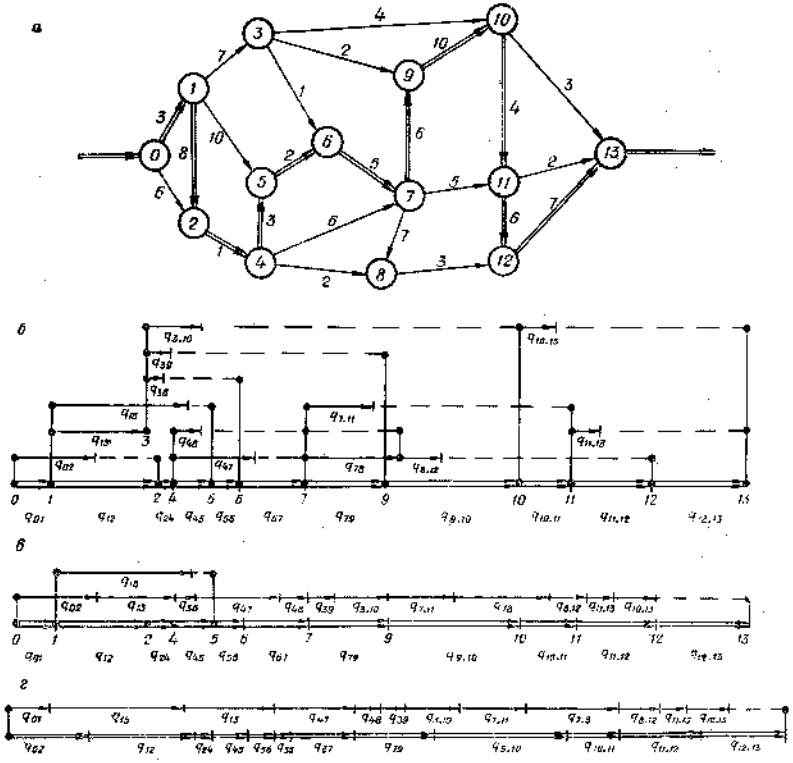


Рис. 80. Представление сетевых графиков в виде p-кортежей.

а — пример сетевого графика; *б* — представление сетевого графика в виде сжатого p-кортежа; *в* — оптимального p-кортежа минимальной длины; *г* — рационального p-кортежа с $\delta \approx 1$.

Указанное отличие, хотя и существенно, но, по-видимому, не помешает применению одинаковых методов. Представляется полезным введение для системы PERT таких понятий, как «ширина», которая характеризует число одновременно выполняемых работ, «сжатый» график, «характеристическая функция» и т. д. Не углубляясь в этот вопрос, укажем, что аналогия становится

более явной, если после того, как составлен сетевой график и выявлен критический путь, совместить его с временным графиком, т. е. отложить вдоль горизонтальной оси время, а вдоль вертикальной — число параллельно выполняемых работ. Проиллюстрируем это (рис. 80) на примере, взятом из работы [15].

7.4. Общая схема решения задач на УВС

Как уже упоминалось, решение задач на УВС имеет характерные особенности, заметно отличающие его от решения задач на обычных ЭВМ. Можно указать восемь основных этапов, которые необходимо выполнить (от формулировки задачи до составления программы):

1) формулировка задачи;
2) выбор алгоритма решения задачи;
3) построение граф-схемы связей алгоритма;
4) исследование характеристической функции и выбор рабочей точки;
5) определение существенного объема информации;
6) выбор покрытия p-кортежа кортежами простых операций;
7) составление схемы p-алгоритма;
8) составление программы решения задачи на УВС.

Рассмотрим каждый из этапов.

Формулировка задачи должна содержать следующее:

а) собственно формулировку задачи с указанием массива исходных данных и вида результирующих данных;

б) метод или методы решения задачи;

в) желаемое время решения задачи. Каждая задача, как правило,— это часть какой-либо проблемы, и время ее решения входит как слагаемое в общее время решения проблемы. На основании этого можно дать две оценки времени решения задачи: минимальное $t_{\min}$ и максимальное $t_{\text{пр}}$. Под $t_{\min}$ понимается такое время, ниже которого его величина перестает существенно сказываться на общем времени решения проблемы. Под $t_{\text{пр}}$ понимается предельно допустимое время решения задачи. Желательно также знать промежуточные значения времени решения задачи $t_{\min} \leqslant t \leqslant t_{\text{пр}}$ с указанием их веса.

Выбор алгоритма решения задачи. Каждый метод решения задачи может быть реализован многими алгоритмами, отличающимися временем выполнения и материальными затратами. В зависимости от требований, предъявляемых к процессу решения задачи, должен быть выбран тот или иной алгоритм. Вопрос о выборе рационального алгоритма весьма сложен. Он не решен

даже для алгоритмов, выполняемых последовательно на одной ЭВМ; практически алгоритмы выбираются в значительной степени интуитивно. Будем считать, что для УВС эта задача также будет решаться пока на таком же уровне; этого, по-видимому, будет достаточно для первого периода использования УВС.

Построение граф-схемы связей алгоритма. Как уже указывалось, граф-схемы связей удобно изображать в виде сжатого p-кортежа (см. рис. 75). Для этого алгоритм записывается с помощью набора простых операторов, для каждого из которых указываются его непосредственные предшественники. Затем строится сжатый p-кортеж, процесс построения которого может быть легко алгоритмизирован. Остается только нанести изображение связей между операторами.

Сжатый p-кортеж представляет собой одну из реализаций p-алгоритма, а полученная граф-схема связей — удобную исходную позицию для его анализа.

Исследование характеристической функции и выбор рабочей точки. Поскольку каждый p-алгоритм может быть представлен многими p-кортежами, то нужно выбрать наиболее рациональный по времени и затратам. Для этого строится характеристическая функция, как это описывалось в 7.3. Если при формулировке задачи известна зависимость между временем решения задачи и ее относительным весом, то выбор соответствующей рабочей точки не представляет больших трудностей, тем более, что известно предельное число машин в системе и выбор производится в сравнительно небольшой области. При этом нужно, конечно, в качестве рабочей точки взять оптимальную.

Определение существенного объема информации. Для рабочей точки (или точек) надо найти существенный объем информации p-кортежа. Эта задача обычно не вызывает больших затруднений. Как будет видно из дальнейшего (гл. 8), во многих задачах существенный объем практически совпадает с объемом исходной информации. В тех задачах, в которых существенный объем возрастает в процессе вычислений, обычно нетрудно найти закономерность, определяющую его увеличение.

Выбор покрытия p-кортежа кортежами простых операций. После того, как выбран p-кортеж, реализующий данный p-алгоритм, и определен существенный объем информации, необходимо найти рациональное покрытие p-кортежа простыми кортежами. Под рациональным будем понимать такое покрытие, при котором достигается некий компромисс между противоречивыми требованиями: уменьшением объема информации, соответствующей каждому кортежу покрытия, и уменьшением связности покрытия. Поясним это.

Прежде всего уточним вопрос об объеме информации. Вся исходная информация v_0 должна быть распределена между кортежами покрытия. К кортежу k_i отнесена информация v_{i0}. При этом допускается вхождение одних и тех же исходных данных в разные кортежи. Тогда фактический объем

$$v_0' = \sum_{i=1}^{L} v_{i0}' \geqslant v_0. \tag{7.26}$$

Замена существенного объема v_0 фактическим v_0' потребует также замены остальных существенных объемов $v_1, v_2, \ldots, v_h$ фактическими $v_1', v_2', \ldots, v_h'$. Величина последних может измениться так же, если результат какой-либо операции одновременно относить к нескольким кортежам. На практике это может встретиться, в частности, при работе на распределенных УВС, в которых, как показано в 5.9, выгоднее работать в режиме, когда информация передается сразу же после ее вычисления.

Аналогично существенному объему информации p-кортежа введем фактический объем информации p-кортежа

$$V' = \max (v_0', v_1', \ldots, v_h') \geqslant V. \tag{7.27}$$

В процессе распределения исходной информации и результатов операций между кортежами покрытия с каждым кортежем k_i на каждом k-м шаге его выполнения связывается некоторый фактический объем информации $v_{i,k-1}'$. Наибольший из этих объемов

$$v_{\text{макс}}' = \max_{i,\,k} v_{i,\,k-1}' \; (i = 1, 2, \ldots, L;\; k = 1, 2, \ldots, h) \tag{7.28}$$

определяет требуемый объем памяти элементарных машин, а связность данного покрытия (7.17) — общий объем информации, которой обмениваются ЭМ. Увеличивая фактический объем, можно в известных пределах уменьшать связность, и наоборот.

Важное свойство покрытия — равномерность распределения фактических объемов между кортежами — может характеризоваться разностью между $v_{\text{макс}}'$ и

$$v_{\text{ср}}' = V'/L. \tag{7.29}$$

Составление схемы p-алгоритма. После того как покрытие p-кортежа выбрано, остается ввести в схему p-кортежа p-операции обмена информацией между кортежами покрытия, обобщенного условного перехода и настройки и записать полученную схему p-алгоритма на формальном, например, матричном p-языке

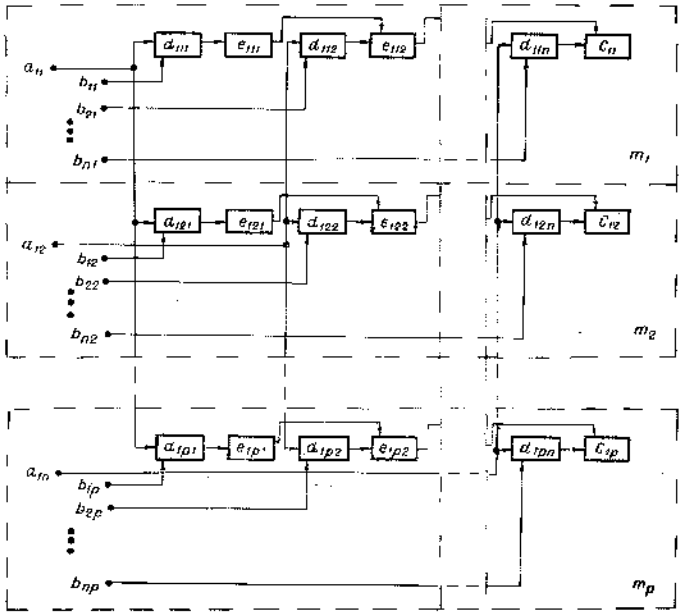


Рис. 81. Схема распределения кортежей между p элементарными машинами для задачи умножения матриц.

$d_{1jk}=a_{1k}b_{kj}$; $e_{1jk}=e_{1jk-1}+d_{ijk}$; $e_{1j0}=d_{1j1}$; $c_{1j}=e_{1jn}$.

[16] или в виде граф-схемы (см. 7,6). Для умножения матриц пример граф-схемной записи для вычисления элементов первой строки матрицы (c_{ij}) показан на рис. 81.

Составление программы решения задачи на УВС. Программирование на УВС само состоит из нескольких этапов.

Для реализации выбранной схемы p-алгоритма необходимо прежде всего распределить кортежи покрытия между элементарными машинами. Будем предполагать, что число ЭМ в УВС равно ширине выбранного p-кортежа. В этом случае число возможных размещений, очевидно, равно $L!$. В УВС обмену информацией между двумя ЭМ m_i и m_j можно поставить в соответствие некото-

рый вес $p_{ij}=p_{ji}$ (при $i=j$ $p_{ij}=0$). Например, в сосредоточенных УВС ЭМ, соединенным непосредственно друг с другом каналами связи, может быть придан вес 1, а всем остальным парам машин — на единицу больший числа ЭМ, через которые они обмениваются информацией. В частности, в одномерных УВС p_{ij} может быть просто равно разности между номерами машин. В распределенных УВС величина p_{ij} должна учитывать в первую очередь время обмена между ЭМ и может быть выбрана пропорциональной ему.

Характеристикой распределения кортежей между машинами с точки зрения обмена информацией будет не связность, а величина

$$PC=\sum_{i,\ j=1}^{L} p_{ij}\vec{c_{ij}}, \tag{7.30}$$

которую назовем *связностью программы.*

Задача заключается в выборе такого распределения кортежей между ЭМ, которое имело бы величину PC возможно меньшей и соответствовало бы возможностям системы коммутаций данной УВС. Например, для одномерных УВС должны быть соблюдены условия, чтобы в каждый момент все ЭМ были разбиты на такие группы, в каждой из которых только одна ЭМ выводила бы информацию, а все другие ЭМ этой же группы могли только принимать ее. ЭМ, выводящая информацию, может одновременно принимать информацию от ЭМ, не входящей в данную группу. Соблюдение этого требования позволяет совместить пересылку кодов с операциями. Для распределенных УВС этот этап имеет особо важное значение. Пример распределения кортежей операторов между ЭМ для задачи умножения матриц приведен на рис. 81. Соблюдение указанных правил упрощает также программы настройки и обмена информацией. Данная задача имет сходство с задачей размещения стандартных элементов, возникающей при проектировании ЭВМ и другой радиотехнической аппаратуры. При размещении стандартных элементов основное требование состоит в минимизации суммарной длины связей между элементами. В общем виде эта задача еще не решена, однако имеются методы, упрощающие ее решение [17—19].

Для УВС, у которых переменной является только система коммутаций, после этого остается собственно программирование, согласно схеме p-алгоритма, записанной, например, на матричном p-языке.

Программирование ведется для каждой элементарной машины отдельно. Однако в большинстве случаев (см. гл. 8) программы

работы всех ЭМ либо полностью идентичны, либо отличаются лишь незначительными деталями. Употребление команд настройки, обмена информацией между ЭМ и обобщенного условного перехода также не приводит к существенному усложнению процесса программирования. Благодаря этому программирование оказывается по сложности того же порядка, что и для одной ЭВМ.

В УВС с полностью переменной структурой, которые могут изменять систему команд и другие параметры, имеются большие возможности повышения эффективности решения задачи. При этом, однако, нужны более сложные методы программирования, обсуждение которых требует отдельного рассмотрения.

7.5. Универсальная программа

В отличие от ЭВМ, у которых время решения задачи однозначно определяется алгоритмом ее решения и параметрами ЭВМ, у вычислительных систем имеется возможность выбирать время решения задачи путем изменения числа выполняющих ее ЭМ. Это свойство особенно важно при использовании УВС для управления какими-либо объектами, когда приходится одновременно решать несколько задач, срочность которых может изменяться в зависимости от ситуации. Решение этого же вопроса требуется также для обеспечения надежного управления объектами, когда функции вышедшей из строя ЭМ берут на себя остальные машины системы. При решении вычислительных задач это позволило бы сократить объем программирования, более оперативно решать задачи и т. п. Чтобы использовать эту возможность на практике, необходимо найти способы построения универсальных программ, которые при изменении числа ЭМ перестраивали бы себя либо сами, либо с помощью некоторой преобразующей программы. Укажем на принципиальную возможность построения простых универсальных программ для p-алгоритмов, базовая точка которых лежит в области больших значений L, что характерно для многих классов задач (см. гл. 8).

Пусть число ЭМ в УВС равно L. Тогда будем строить L-развертки p_b-кортежа. p_b-Кортеж прямоугольный или почти прямоугольный. Его покрытие состоит из L_b-кортежей. Дополним p_b-кортеж λ пустыми кортежами так, чтобы $L_b + \lambda = d \cdot L$, где $d > 0$ целое, $0 \leqslant \lambda < L$, и разобьем дополненный p-кортеж на L полос шириной d. Строя теперь L-развертку, каждую полосу развернем в один простой кортеж. В результате получаем прямоугольный p-кортеж шириной L и длиной $h_b d$.

Покажем, что связность покрытия построенного p-кортежа не превышает связности p_b-кортежа, т. е. $C \leqslant C_b$.

Процессу построения нового p-кортежа шириной L и длиной $h = h_b d$ соответствует следующее:

а) дополнение с двух сторон исходной матрицы связности λ нулевыми строками и столбцами;

б) разбиение матрицы на L^2 клеток размером $d \times d$ элементов;

в) замена клетки новым элементом. При этом диагональные клетки заменяются нулями, а каждая из остальных — элементом, значение которого не превышает суммы элементов соответствующей клетки исходной матрицы.

Отсюда следует, что связность покрытия нового p-кортежа не больше, чем у исходного. То же самое можно сказать и о связности программы. Более того, если исходное покрытие было рациональным, то уменьшение связности программы может быть значительным, так как наиболее связанные друг с другом кортежи объединятся в один.

По аналогичным соображениям, при переходе от p_b-кортежа к его L-развертке фактический объем исходной информации может только уменьшиться и приблизиться по значению к существенному объему. Естественно, что фактический объем информации, приходящийся на каждый кортеж покрытия, может только увеличиваться. Это увеличение происходит из-за того, что общий фактический объем исходной информации делится между меньшим (в d раз) числом кортежей. Кроме того, увеличивается фактический объем промежуточной информации, приходящейся на каждый кортеж вследствие того, что результат каждой операции должен храниться, по крайней мере, до тех пор, пока не начнется выполнение операций, входивших в p_b-кортеже в следующую по порядку p-операцию. При этом дополнительно придется хранить d кодов.

Таким образом, можно сделать общий вывод: если имеется рациональная программа решения задачи на УВС из L_b машин, соответствующим базовой точке, лежащей в области больших значений L, то программа для УВС с числом машин $L < L_b$ может быть получена путем сравнительно простых преобразований программы для базовой точки. При этом, если последняя была рациональной, то и полученная программа будет также рациональной, когда $L \ll L_b$, либо L_b кратно, или близко к кратному L

7.6. Язык для описания схем p-алгоритмов

Параллельные алгоритмы существенно отличаются от известных алгоритмов [11] одновременностью выполнения большого числа операций. Формально отличие состоит в том, что вместо

однокоординатной записи схемы алгоритма (например, в виде строчки) надо применять двухкоординатную, где одна координата указывает последовательность выполнения операций во времени, а вторая — распределение операций между ветвями вычислений. Имеющиеся языки описания алгоримов [8, 9, 12] не пригодны для *р*-алгоритмов.

Для описания схемы *р*-алгоритмов введем матричный язык. Для краткости будем называть его *р-языком*.

В *р*-языке используются в качестве элементов как простые операторы (см. 7.2), так и обобщенные. Под *обобщенным оператором* будем понимать последовательность нескольких операторов, если один и только один из входящих в него простых операторов имеет внешний вход; в каждый момент времени выполняется только один простой оператор; за конечное число шагов после выполнения оператора, имеющего внешний вход, будут выполнены все простые операторы.

Соответственно будем называть *р*-операторы *обобщенными*, если их компонентами являются обобщенные операторы. Обобщенные *р*-операторы будем обозначать либо буквой $\mathbf{Q}_j$, либо $\dot{Q}_j$, либо в виде столбца $\begin{pmatrix} Q_{1j} \\ \vdots \\ Q_{nj} \end{pmatrix}$, где Q_{ij} $(i = 1, 2, \ldots, n)$—обобщенный оператор, принадлежащий i-й ветви вычислений.

Простые *р*-операторы будем обозначать буквами латинского и русского алфавитов, написанными жирным шрифтом, либо с точкой сверху, либо в виде столбца из простых операторов, входящих в него. Для некоторых часто встречающихся стандартных операторов будем употреблять установившиеся обозначения:

А — арифметический оператор, производящий вычисления по формулам;

Р — оператор условного перехода;

F(i) — оператор изменения параметра, изменяющий адреса, либо содержимое определенных ячеек памяти, зависящих от значений параметра *i*;

И — оператор начала, определяющий первую команду программы;

Я — оператор конца, определяющий останов машины;

Ф — оператор формирования, изменяющий другие операторы;

В — оператор восстановления первоначального значения параметра либо содержимого ячейки памяти;

Е — всякий простой оператор, отличный от перечисленных выше.

Λ — пустой оператор, пропускающий выполнение одного шага вычислений.

Аналогично некоторые *р*-операторы, все составляющие которых одинаковы и совпадают с одним из стандартных операторов, будем называть стандартными *р*-операторами и обозначать теми же буквами, что и для его составляющих, но написанных жирным шрифтом или снабженных точкой сверху. К стандартным *р*-операторам будем относить также операторы, специфичные для вычислительной системы (см. гл. 5):

Ḣ — *р*-оператор настройки, производящий изменение состояний коммутаторов элементарных машин либо параметров, управляющих структурой ЭМ;

Ȯ — оператор обмена информацией между ЭМ, который указывает для каждой ЭМ, какой код (содержащийся в ячейке ОП, арифметическом устройстве или устройстве управления) должен быть послан во внешний канал связи либо принят из канала связи и направлен в один из блоков ЭМ, либо то и другое одновременно;

Ṗ — *р*-оператор обобщенного условного перехода, устанавливающий изменение хода вычислительного процесса в системе в зависимости от выполнения некоторого условия (или условий) в одной или всех ЭМ.

Введем также по аналогии с [9] левую и правую полускобки: $\lfloor_k$ и $\rfloor_k$, соответственно. Левая полускобка $\lfloor_k$ будет всегда следовать за *р*-оператором обобщенного условного перехода Ṗ. Правая полускобка $\rfloor_k$ предшествует *р*-оператору, к выполнению которого необходимо перейти в случае невыполнения условия, определяемого Ṗ. В случае выполнения условия осуществляется переход к *р*-оператору, следующему за Ṗ.

Логической схемой *р*-алгоритма будем называть матрицу

$$\Omega_{Lh} = \begin{bmatrix} \dot{\Omega}_{11} & \Omega_{12} \ldots \Omega_{1j} \ldots \Omega_{1h} \\ \Omega_{21} & \Omega_{22} \ldots \Omega_{2j} \ldots \Omega_{2h} \\ \cdots & \cdots \\ \Omega_{i1} & \Omega_{i2} \ldots \Omega_{ij} \ldots \Omega_{ih} \\ \cdots & \cdots \\ \Omega_{L1} & \Omega_{L2} \ldots \Omega_{Lj} \ldots \Omega_{Lh} \end{bmatrix} \qquad (7.31)$$

элементы j-го столбца $(j = 1, 2, \ldots, h)$ которой представляют собой либо операторы (простые или обобщенные) с правой полускобкой или без нее, либо операторы условного перехода с левой полускобкой. Операторы i-й строки $(i = 1, 2, \ldots, L)$ образуют кортеж, соответствующий i-й ветви вычислений, причем в i-й

строке для каждой левой полускобки $\underset{k}{\lfloor\!\_}$ имеется одна и только одна правая полускобка с тем же индексом k, и, наоборот, для каждой правой полускобки $\underset{k}{\_\!\rfloor}$ имеется одна и только одна левая полускобка с тем же индексом. Все столбцы нумеруются слева направо, а строки — сверху вниз.

Будем также использовать векторную форму записи схемы p-алгоритма, представляющую собой строку, составленную из символов p-операторов, левой и правой полускобок.

В некоторых случаях, как и в обычных схемах алгоритмов [8], будем вместо левой полускобки использовать стрелку, исходящую из данного оператора условного перехода, с указанием номера оператора, к выполнению которого нужно перейти в случае невыполнения условия, а правую полускобку заменять входящей стрелкой с указанием номера оператора условного перехода, от которого осуществляется переход к данному.

Иногда для наглядности вместо полускобок или стрелок с адресами будем использовать для описания условных переходов в схемах p-алгоритмов линии, соединяющие оператор условного перехода с оператором, к выполнению которого следует перейти при невыполнении условия.

Для выяснения структуры связей между ветвями вычислений будем также применять запись логических схем p-алгоритмов в виде графов. Вершинами графов служат сами операторы и источники информации, дугами — информационные и управляющие связи.

Рассмотрим в качестве примера запись на p-языке схемы p-алгоритма умножения прямоугольных матриц A и B ((7.2) — (7.4)) на вычислительной системе из $M = mp$ элементарных машин. Пусть для простоты в памяти ЭМ, вычисляющей элемент c_{ij}, хранится строка a_i и столбец b_j.

Схема p-алгоритма на матричном p-языке запишется так

$$\begin{array}{c|cccc|} 1 & \underset{1}{\lfloor\!\_} A_1 & A_2 & P_3 \underset{1}{\lfloor\!\_} & Я_4 \\ 2 & \underset{1}{\lfloor\!\_} A_1 & A_2 & P_3 \underset{1}{\lfloor\!\_} & Я_4 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ M & \underset{1}{\lfloor\!\_} A_1 & A_2 & P_3 \underset{1}{\lfloor\!\_} & Я_4 \end{array}$$

где операторы A_1 и A_2 вычисляют соответственно $d_{ijk} = a_{ik}$ и b_{kj}.

$e_{ijk} = e_{ijk-1} + d_{ijk}$ $(e_{ij0} = 0)$; P_3 — оператор условного перехода (по $k = p$); $Я_4$ — оператор конца.

В векторной форме эта схема примет вид

$$\underset{1}{\lfloor\!\_} A_1 \quad A_2 \quad P_3 \underset{1}{\lfloor\!\_} Я_4$$

либо

$$\overset{\uparrow}{A_1} \quad A_2 \quad P_3 \quad Я_4.$$

Пример граф-схемной записи этого же p-алгоритма приведен на рис. 75

Данный пример и примеры записей схем p-алгоритмов, приведенные в гл. 8, показывают, что p-язык в матричной и граф-схемной формах позволяет сравнительно просто описывать процесс решения задач на вычислительных системах.

Глава 8

РЕШЕНИЕ ЗАДАЧ НА УНИВЕРСАЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

В связи с разработкой универсальных вычислительных систем возникают вопросы, насколько широк круг задач, которые могут на них эффективно решаться, и каковы трудности при составлении схем алгоритмов и программировании.

Сейчас, когда теория параллельных алгоритмов находится в зачаточном состоянии, ответить на эти вопросы можно лишь путем рассмотрения достаточно большого числа конкретных задач. В качестве начала этой большой работы в данной главе рассматривается 16 типов задач, охватывающих основные разделы вычислительной математики.

Цель этого рассмотрения — показать принципиальную возможность решения задач на УВС. Поэтому для каждой конкретной задачи ограничимся записью на p-языке одного из p-алгоритмов, возможно не наилучшего, но обладающего высоким коэффициентом эффективности δ. При выборе алгоритмов используются в основном известные методы решения задач, которые, естественно, не предназначались для УВС.

В дальнейшем будем, как правило, рассматривать сложные задачи, не поддающиеся решению на современных ЭВМ, для решения которых требуется выполнить 10^{10} и более операций. Однако, как будет видно из дальнейшего, те же p-алгоритмы оказываются применимыми и для ускорения процесса решения задач с меньшим числом операций.

Для простоты изложения будем описывать схему p-алгоритма для большого числа параллельных ветвей. Переход к схемам с меньшим числом ветвей может выполняться с помощью простой методики, указанной в 7.5.

Далее будем предполагать, что вычислительная система состоит из M одинаковых машин и каждая ЭМ имеет набор простых операций, определяемый кругом решаемых задач. Структура вычислительной системы аналогична описанным в гл. 5.

8.1. Задачи линейной алгебры

Рассмотрим особенности решения на УВС задач линейной алгебры: систем линейных уравнений, обращение матриц, нахождение собственных значений матриц. Эти задачи относятся к числу наиболее часто встречающихся на практике.

К решению систем линейных уравнений больших порядков сводятся многие задачи физики и техники как непосредственно, так и при решении дифференциальных уравнений (обыкновенных и в частных производных).

Определение матриц, обратных заданным, возникает в задачах распределения электрических токов в схемах с многими соединениями; распределения скоростей потоков воды в сложных гидравлических системах; теории потенциала; решения нормальных систем уравнений, получающихся при применении к различным физическим задачам метода наименьших квадратов; приложения статистического анализа к психологии, социологии, экономике и многих других.

Характеристическое уравнение с его собственными значениями и соответствующими собственными векторами (проблема собственных значений) играет основную роль в теории колебаний, будь то колебания механического или электрического, макроскопического или микроскопического характера.

Примерами применения характеристического уравнения служат упругие колебания моста или другого жесткого сооружения, неустойчивые колебания крыла самолета, неустановившиеся колебания электрической сети, атомарные и молекулярные колебания, рассматриваемые в волновой механике Шредингера, и т. д.

Приведенный далеко не полный перечень задач показывает, какую большую роль играют задачи линейной алгебры. Решению их посвящено очень много работ. Наиболее полно методы решения задач линейной алгебры изложены в работе [1], которой будем придерживаться в дальнейшем.

Специально разработанных методов решения задач линейной алгебры на вычислительной системе нет, хотя некоторая работа в этой области была проделана [2—4].

Анализ известных методов решения задач линейной алгебры показывает, что их можно эффективно использовать для решения задач на УВС. Приведем лишь некоторые из них, а именно: решение системы линейных уравнений методом последовательных приближений, обращение матриц методом пополнения, вычисление коэффициентов характеристического полинома методом А. М. Данилевского.

Решение системы линейных уравнений методом последовательных приближений. Под последовательным приближением понимается следующий итерационный процесс. Система уравнений $AX = G$ записывается в виде

$$X = BX + G, \tag{8.1}$$

где $B = E - A$;
A, B — матрицы;
X и G — векторы;
E — единичная матрица.

Последовательные приближения вычисляются по формуле

$$X^{(k)} = BX^{(k-1)} + G, \tag{8.2}$$

начиная с некоторого исходного приближения X^0, которое может быть выбрано произвольно. Разумеется, условия сходимости предполагаются выполненными. Процесс завершается, если

$$|X^{(k)} - X^{(k-1)}| < \varepsilon, \quad \varepsilon > 0.$$

Этот метод называется методом простой итерации.

Формулу (8.2) представим в развернутом виде:

$$\begin{aligned}
x_1^{(k)} &= g_1 + b_{11}x_1^{(k-1)} + \ldots + b_{1j}x_j^{(k-1)} + \ldots + b_{1n}x_n^{(k-1)};\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
x_i^{(k)} &= g_i + b_{i1}x_1^{(k-1)} + \ldots + b_{ij}x_j^{(k-1)} + \ldots + b_{in}x_n^{(k-1)};\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
x_n^{(k)} &= g_n + b_{n1}x_1^{(k-1)} + \ldots + b_{nj}x_j^{(k-1)} + \ldots + b_{nn}x_n^{(k-1)}.
\end{aligned} \tag{8.3}$$

В качестве нулевого приближения принимается $x_i^0 = g_i$ ($i = 1, 2, \ldots, n$). Для каждого приближения вычисляется разность $\Delta x_i^{(k)} = |x_i^{(k)} - x_i^{(k-1)}|$. Задача считается решенной, когда выполняется условие

$$\max_{1 \leqslant i \leqslant n} \Delta x_i^{(k)} < \varepsilon. \tag{8.4}$$

При вычислении каждого приближения выполняется n^2 операций умножения, n^2 операций сложения, n операций вычитания и n операций сравнения, т. е. всего $2n^2 + 2n$ операций. Функция пошагового распределения $\varphi(j)$ для этого случая, соответствующая сжатому кортежу, имеет вид, показанный на рис 82. На первом шаге выполняются все n^2 умножений, затем за $s = \log_2(n+1)$ последующих шагов выполняются n^2 сложений, затем

выполняются n вычитаний и, наконец, условный переход, т. е. минимальное число шагов для выполнения одного приближения

$$h_m = \log_2(n+1) + 3. \tag{8.5}$$

Это число шагов будет достигнуто, если допускается одновременное выполнение n^2 операций, т. е. ВС содержит n^2 элементарных машин.

Форма функции $\varphi(j)$ получилась при этом далекой от оптимальной — прямоугольной. Нетрудно видеть, что все операции лежат на критическом пути (см. 7.3, п. 4), т. е. форма функции $\varphi(j)$ может быть улучшена только при увеличении числа шагов.

Покажем, что при числе машин n может быть получен p-кортеж с коэффициентом эффективности $\delta = 1$. Действительно, построим процесс вычислений так, что каждая ЭМ m_i производит вычисление одного из $x_i^{(k)}$ и хранит в своей памяти коэффициенты $b_{i1}, \ldots, b_{in}$, g_i и $x_i^{(k-1)}$, а также программу своей работы.

Тогда на первом шаге из машины m_1 во все остальные ЭМ будет передано число $x_1^{(k-1)}$; на втором — образованы произведения $e_{i1}^{(k)} = b_{i1}x_1^{(k-1)}$; на третьем — суммы $s_{i1}^{(k)} = g_i + e_{i1}^{(k)}$.

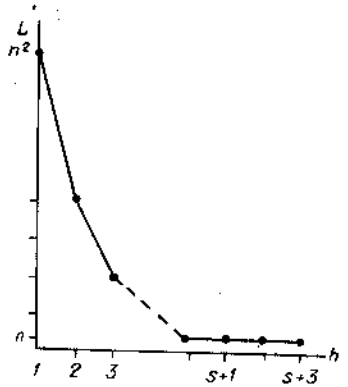


Рис. 82. Вид сжатого p-кортежа для решения системы линейных уравнений методом последовательных приближений. $s = \log_2(n+1)$.

После этого во все машины засылается число $x_2^{(k-1)}$, хранящееся в машине m_2. Затем вычисляются произведения $e_{i2} = b_{i2}x_2^{k-1}$, суммы $s_{i2}^{(k)} = s_{i1}^{(k)} + e_{i2}^{(k)}$ и т. д.

После $3n$ шагов получается $x_i^{(k)} = s_{in}^{(k)}$. Затем вычисляются $\Delta x_i^{(k)} = |x_i^{(k)} - x_i^{(k-1)}|$ и по результату сравнения $\Delta x_i^{(k)}$ с ε делается условный переход. Если $\max\limits_{1 \leqslant i \leqslant n} \Delta x_i^{(k)} < \varepsilon$, то процесс прекращается, в противном случае выполняется следующее приближение.

Общее число шагов для каждого приближения $h(n) = 3n + 2$, а если предусмотреть возможность совмещения пересылки числа с вычислительными операциями (в данном случае умножением), то $h(n) = 2n + 2$.

На каждом шаге будут работать все n машин. В результате за указанное число шагов выполняется все $2n^2 + 2n$ операций. Объем памяти каждой ЭМ должен состоять из $n + 3$ ячеек для хранения коэффициентов $x_i^{(k-1)}$ и $x_i^{(k)}$, из двух рабочих ячеек для хранения $s_{ij}^{(k)}$ и $e_{ij}^{(k)}$, а также из r ячеек для программы. В данном случае программа будет весьма простой и не превысит по-видимому, 100 команд. Таким образом, требуемый объем памяти ЭМ будет:

$$V = (n + 5)m_1 + rm_2, \qquad (8.6)$$

где m_1 и m_2 — количество двоичных разрядов чисел и команд соответственно.

Процесс решения данной задачи на УВС из n элементарных машин может быть проиллюстрирован схемой (рис. 83), на которой операции, выполняемые одновременно, находятся в одной колонке. Стрелками указаны информационные связи, необходимые для выполнения каждой из операций.

Для составления программы работы системы удобнее ввести циклы. Логическая схема соответствующего p-алгоритма может

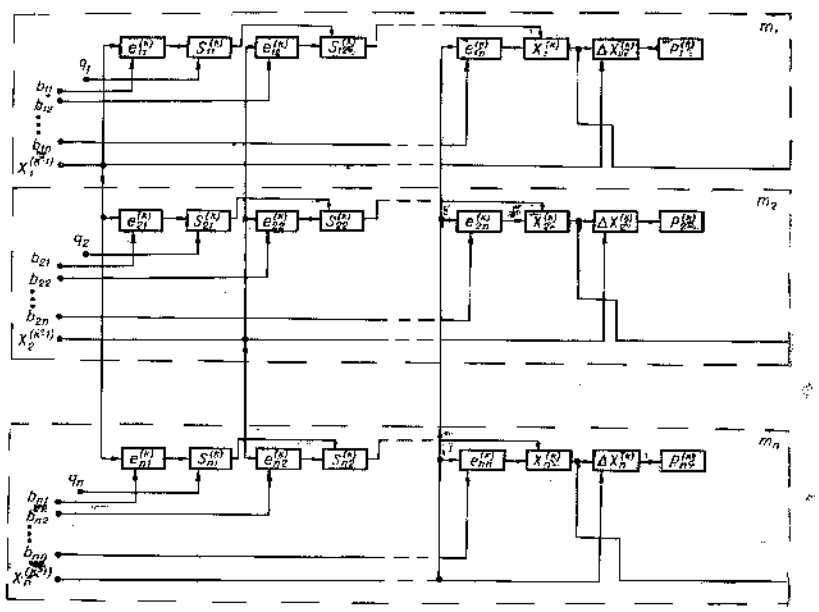


Рис. 83. Схема решения на УВС системы линейных уравнений методом последовательных приближений.

быть записана в виде следующей строчки:

$$\dot{H}_1\dot{F}_2\dot{O}_3^j\dot{A}_4^j\dot{A}_5^j\dot{P}_6(j)\dot{A}_7\dot{P}_8\dot{P}_9\dot{Я}_{10}, \qquad (8.7)$$

где $\dot{H}_1$ — p-оператор настройки, устанавливающий состояния коммутаторов ЭМ. В данной задаче на каждом из этапов вычислений код поочередно из каждой ЭМ (начиная с m_1) пересылается во все остальные. Поэтому p-оператор $\dot{H}_1$ устанавливает все входы и выходы ЭМ в открытом состоянии.

p-Оператор F_2 устанавливает первоначальное значение индекс-регистров и счетчиков циклов во всех ЭМ.

$\dot{O}_3^j$ — p-оператор обмена информацией между ЭМ, зависит от параметра j ($j = 1, 2, \ldots, n$). При $j = 1$ передается код в линию связи из ЭМ m_1, при $j = 2$ — из ЭМ m_2 и, наконец, при $j = n$ — из ЭМ m_n. Коды, поступающие в линию связи, воспринимаются всеми ЭМ.

$\dot{A}_4^j$, $\dot{A}_5^j$ и $\dot{A}_7$ — арифметические p-операторы, соответственно вычисляющие $e_{ij}^{(k)}$, $s_{ij}^{(k)}$ и $\Delta x_i^{(k)}$. Первый из них в зависимости от параметра j (номер цикла) изменяет адрес ячейки ОП, из которых берется соответствующее b_{ij}.

$\dot{P}_6(j)$, $\dot{P}_8$ и $\dot{P}_9$ — p-операторы обобщенного условного перехода. Первый из них управляет изменением параметра j и при $j = n$ переводит систему на выполнение p-оператора $\dot{A}_7$. Второй проверяет условие (8.4), если оно удовлетворено, то указывает на окончание решения задачи, если не удовлетворено, то передает управление p-оператору $\dot{P}_9$, который проверяет, не превысило ли число итераций k заранее заданного числа N. При $k \leqslant N$ система переходит к выполнению p-оператора F_2, при $k > N$ решение прекращается.

Как видно из схемы (см. рис. 83), p-операторы состоят из одинаковых составляющих. Благодаря этому программы работы всех ЭМ совпадают с точностью до адресов.

Расхождение в программах, которое может возникнуть из-за отличия составляющих p-оператора обмена $\dot{O}_3^j$, может быть устранено. Действительно, каждую составляющую p-оператора Q_3^j можно представить в виде логической схемы из трех операторов $\overline{P_{31}(i, j)}\,Q_{32}\,\overline{O_{33}(A_4^j)}$. Оператор $P_{31}(i, j)$ проверяет совпадение номера цикла j с номером машины i. При $j \neq i$ выполняется оператор O_{33}, принимающий код из канала связи и направляющий его в арифметическое устройство. При $j = i$ выполняется

оператор Q_{32}, который отправляет код $x_i^{(k-1)}$, хранящийся в ОП, во внешний канал связи, принимает его из канала связи и отправляет в арифметическое устройство, а затем передает управление оператору A_1^j. Номер машины i у каждой ЭМ может храниться в ячейке с одним и тем же адресом. Таким образом, отличие в номерах ЭМ будет проявляться только в содержимом этой ячейки, а не в программах.

Для составления программы можно воспользоваться логической схемой (8.7), заменяя соответствующие p-операторы простыми операторами. Нетрудно видеть, что все операции, которые использовались при решении данной задачи, являются простыми, т. е. они либо одно-, либо двуместные и по сложности не превосходят операции умножения. Таким образом, мы не допустили противоречия с условиями, которые были положены в основу данного анализа (см. гл. 7).

Выбор этих условий определялся в первую очередь удобствами анализа задач. Это вовсе не означает, что при создании реальных УВС в систему команд ЭМ не должны быть заложены более сложные операции. В некоторых случаях введение подобных операций может оказаться вполне оправданным. Прежде всего это относится к частым операциям, которые не повышают требований к обмену информацией между ЭМ и не требуют усложнения системы связи. Для задач линейной алгебры характерна операция суммирования парных произведений (умножение двух чисел и сложение полученного произведения с содержимым некоторой ячейки). Эта операция трехместная. Она, однако, не увеличивает потока информации между ЭМ, если суммирование выполнять с содержимым ячейки памяти из той же ЭМ.

Обращение матриц методом пополнения. Пусть даны квадратные матрицы n-го порядка: $A = [a_{ij}]$, обратную которой $A^{-1} = [\alpha_{ij}]$ надо найти; $B = [b_{ij}]$, отличающаяся от матрицы A только строкой с номером k; $V = [v_{ij}]$, у которой элементы всех строк, кроме k-й, равны нулю, а элементы k-й строки таковы, что $A = B + V$ и $B^{-1} = [\beta_{ij}]$, обратная B. Тогда, как показано [1], любой j-й столбец матрицы A^{-1} может быть найден по формуле:

$$\alpha_j = \beta_j - \frac{(v_k, \beta_j)}{1 + (v_k, \beta_k)} \beta_k, \tag{8.8}$$

где α_j — столбец матрицы A^{-1} с номером j;
β_j и β_k — столбцы матрицы B^{-1} с номерами j и k соответственно;
v_k — строка матрицы V с номером k;
(v_k, β_j) — скалярное произведение векторов.

При обращении матриц методом пополнения матрица A рассматривается как последний член последовательности $A_0 = E$, $A_1, \ldots, A_n = A$, причем переход от матрицы A_{k-1} к матрице A_k осуществляется посредством замены k-й строки матрицы A_{k-1} на k-ю строку матрицы A. В качестве начальной берется единичная матрица E. Нетрудно видеть, что у матриц A_k^{-1} строки с номерами $i > k$ останутся те же, что и у матрицы E. Матрица A^{-1} получается в результате повторения указанного процесса n раз. Применительно к этому случаю формула (8.8) для перехода на k-м шаге примет вид:

$$\alpha_j^{(k)} = \alpha_j^{(k-1)} - \frac{(w_k, \alpha_j^{(k-1)})}{1 + (w_k, \alpha_k^{(k-1)})} \alpha_k^{(k-1)}, \tag{8.9}$$

где $\alpha_j^{(k)}$ и $\alpha_j^{(k-1)}$ — j-е столбцы матриц A_k^{-1} и A_{k-1}^{-1} соответственно;
$\alpha_k^{(k-1)}$ — k-й столбец матрицы A_{k-1}^{-1};
$w_k = (a_{k1}, \ldots, a_{kk} - 1, \ldots, a_{kn})$ — строка матрицы $W = A - E$. Подробнее (8.9) может быть записано в виде

$$\alpha_{ij}^{(k)} = \alpha_{ij}^{(k-1)} - \frac{\sum_{i=1}^{n} w_{ki}\alpha_{ij}^{(k-1)}}{1 + \sum_{i=1}^{n} w_{ki}\alpha_{ik}^{(k-1)}} \alpha_{ik}^{(k-1)}. \tag{8.10}$$

Формула (8.10) отражает весь процесс обращения матрицы методом пополнения.

Для дальнейшего анализа распишем процесс вычисления по этой формуле более подробно. Введем обозначения для получающихся в ходе вычислений промежуточных величин. Смысл их будет ясен из самих выражений:

$$c_{ij}^{(k)} = w_{kl}\alpha_{ij}^{(k-1)}; \tag{8.11}$$

$$d_{ij}^{(k)} = d_{i-1,j}^{(k)} + c_{ij}^{(k)}; \tag{8.12}$$

$$d_{0j}^{(k)} = 0.$$

Обозначим $e_j^{(k)} = d_{nj}^{(k)}$.

Так как у матрицы A_{k-1}^{-1} строки с номерами $i \geqslant k$ те же, что у матрицы E, то

$$\begin{cases} e_j^{(k)} = d_{k-1,j}^{(k)} & j < k; \\ e_j^{(k)} = d_{k-1,j}^{(k)} + w_{kj} & j \geqslant k; \end{cases} \tag{8.13}$$

$$f^{(k)} = 1 + e_k^{(k)}; \tag{8.14}$$

$$g_j^{(k)} = e_j^{(k)} / f^{(k)}; \tag{8.15}$$

$$h_{ij}^{(k)} = g_j^{(k)} \alpha_{ik}^{(k-1)}; \tag{8.16}$$

$$\alpha_{ij}^{(k)} = \alpha_{ij}^{(k-1)} - h_{ij}^{(k)} \quad i \neq k; \tag{8.17}$$

$$\alpha_{kj}^{(k)} = \alpha_{kj}^{(k-1)} - g_j^{(k)}.$$

Придавая индексам значения $i = 1, 2, \ldots, k$; $j = 1, 2, \ldots, n$, получаем все элементы матрицы A_k^{-1}.

Рассмотрим требуемое число шагов.

Все $c_{ij}^{(R)}$ не зависят друг от друга, поэтому все $(\kappa - 1)n\, c_{ij}^{(k)}$ могут быть вычислены одновременно за один шаг.

Получение всех n сумм $d_{k-1,j}$ $(j = 1, 2, \ldots, n)$ может выполняться за $\log_2(k-1)$ шагов $(k > 4)$ (для первых 4-х этапов это число будет соответственно равно 0, 0,1 и 2).

Вычисление $e_j^{(k)}$ требуется только для $(n - k + 1)$ случаев. На это необходим еще один шаг.

На получение $f^{(k)}$ требуется одна операция, выполняемая за один шаг.

Затем по одному шагу требуется на вычисление всех $g_j^{(k)}$ $h_{ij}^{(k)}$, $\alpha_{ij}^{(k)}$ $(i = 1, 2, \ldots, k;\ j = 1, 2, \ldots, n)$.

Таким образом, каждый этап может быть выполнен за $h_m = 6 + \log_2(k-1)$ шагов, если осуществлять одновременно до n^2 вычислений. При этом коэффициент использования устройств δ будет небольшим. Сжатый p-кортеж в данном случае является и оптимальным (рис. 84).

Нетрудно видеть, что базовая точка на характеристической функции для данного p-алгоритма будет при числе ЭМ, равном n. Действительно, процесс вычисления в этом случае можно распределить между n машинами таким образом, что на каждой ЭМ m_j будут вычисляться элементы j-го столбца $\alpha_{1j}^{(k)}$, $\alpha_{2j}^{(k)}$, ..., $\alpha_{kj}^{(k)}$ матрицы A_k^{-1} (рис. 85). На рис. 85 приведены также зависимости между выполняемыми операциями и схема обмена кодами между ЭМ. При этом предполагается, что в памяти ЭМ m_j $(j = 1, 2, \ldots, n)$ содержатся элементы j-го столбца w_{1j}, w_{2j}, ..., w_{jj} и j-й строки w_{j1}, w_{j2}, ..., w_{jj-1} матрицы W. (В ЭМ m_1 будет храниться один элемент w_{11}, а в ЭМ m_n — $(2n-1)$ элементов.) Кроме того, в каждой ЭМ m_j запоминаются элементы $\alpha_{1j}^{(k-1)}$, ..., $\alpha_{k-1,j}^{(k-1)}$, предыдущей промежуточной матрицы A_{k-1}^{-1}. На схеме представлены арифметические операторы. Операторы, которые

образуют один p-оператор, расположены в одном столбце. При этом из $4k + 1$ операторов только в двух случаях число одновременно выполняемых операций не равно n. При вычислении $e_j^{(k)}$ оно равно $n - k + 1$ операции, при вычислении $f^{(k)}$ — одной операции. (На схеме для единообразия программ введены пустые операции, помеченные звездочкой.) Нетрудно видеть, что коэффициент δ в данном случае отличается от 1 на величину, меньшую $1/2n$, которой можно пренебречь и считать, что УВС из n элементарных машин по сравнению с УВС из одной машины ускоряет выполнение данной задачи в n раз (без учета выигрыша от увеличения суммарного объема оперативной памяти). Беспетлевая схема (см. рис. 85) удобна для выявления связей между операторами. Для программирования эту схему нужно свернуть путем введения циклов и добавить операторы, организующие процесс вычислений на системе. В результате логическая схема p-алгоритма может быть записана в виде следующей строчки p-операторов:

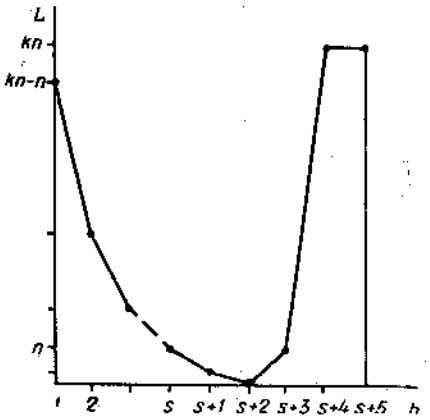


Рис. 84. Вид сжатого p-кортежа для обращения матриц методом пополнения. $s = 1 + \log_2(k-1)$.

$$\dot{H}_1 \dot{F}_2^k \dot{O}_3 \dot{A}_4^k \dot{O}_5^k \dot{A}_6^k \dot{A}_7^k \dot{P}_8^k (k)\, \dot{A}_9^k \dot{Q}_{10}^k \dot{A}_{11}^k \dot{O}_{12}^k \dot{A}_{13}^k \dot{A}_{14}^k \dot{P}_{15}^k (k)\, \dot{A}_{16}^k \dot{P}_{17} \dot{Я}_{18} \dot{H}_{19}^k. \tag{8.18}$$

Операторы $\dot{H}_1$ и $\dot{H}_{19}^k$ $(k = 2, 3, \ldots, n)$ — p-операторы настройки, устанавливающие состояния коммутаторов ЭМ. Из схемы (см. рис. 85) видно, что на каждом этапе обмен кодами между ЭМ осуществляется так, что одна ЭМ (номер которой совпадает с номером этапа) передает код во все остальные ЭМ. Таким образом, к системе связи и конструкции коммутатора предъявляются минимальные требования. Например, вполне достаточно было бы связать ЭМ кольцевым двусторонним каналом связи, к которому каждая ЭМ подключается с помощью коммутатора, состоящего из двух конъюнкторов и одного дизъюнктора

(рис. 86). Кроме информационного канала имеется особый канал для передачи управляющего сигнала ω, включающего все ЭМ для приема информации. В блоке управления коммутацией достаточно помнить значения двух двоичных чисел c_{10} и c_{02}. Естественно, что все более сложные виды системы связи, описанные в гл. 5, также могут реализовать указанные функции. На всех этапах необходимо, чтобы у всех ЭМ входы были открыты $c_{10}^{(j)} = 1$ ($j = 1, 2, \ldots, n$), а выходы закрыты, кроме ЭМ m_k, у которой номер совпадает с номером этапа, $c_{02}^{(j)} = 0$ ($j \neq k$), $c_{02}^{(k)} = 1$. p-Оператор $\dot{H}_1$ можно реализовать за три такта. На первом — всем $c_{10}^{(j)}$ ($j = 1, 2, \ldots, n$) придается значение 1, на втором — всем $c_{02}^{(j)}$ ($j = 1, 2, \ldots, n$) придается значение 0 и на третьем — $c_{02}^{(k)}$ придается значение 1. Управление настройкой может осуществляться машиной m_k, номер которой совпадает с номером этапа. Каждый этап оканчивается p-оператором $\dot{H}_{19}^k$, который можно реализовать следующим образом. В каждой из ЭМ m_j ($j = 1, 2, \ldots, n-1$) в ячейках ОП с одними и теми же адресами будем хранить команды настройки: установить в машине m_{j+1} значение $c_{02}^{j+1} = 1$, а в самой ЭМ m_j — значение $c_{02}^{(j)} = 0$. При выполнении оператора $\dot{H}_{19}^k$ обе команды с адресами машин, которым они предназначаются, посылаются в канал связи. Благодаря тому, что у всех ЭМ, кроме m_k, выходы закрыты, в него поступают только команды, хранящиеся в ЭМ m_k. Первая из этих команд воспринимается и исполняется машиной m_{k+1}, вторая — самой ЭМ m_k.

p-Оператор $\dot{F}_2^k$ изменяет параметр k (номер этапа) на единицу.

p-Операторы обмена информацией между ЭМ $\dot{O}_3^k$, $\dot{O}_5^k$, $\dot{O}_{12}^k$ воздействуют одинаковым образом на все ЭМ. При этом код, хранящийся в ОП ЭМ по заданному адресу, посылается в канал связи. Так как выход открыт только у машины m_k, то в канал связи поступает код только из нее. Этот код из канала связи поступает на входы всех ЭМ (в том числе и самой m_k). Выполнение p-операторов обмена можно представить в виде двух операций: пересылки кода из ОП в канал связи и пересылки кода из канала связи в арифметическое устройство. Их выполнение может совмещаться с выполнением других операторов.

p-Оператор $\dot{A}_4^k$ вычисляет $c_{ij}^{(k)}$ ($j = 1, 2, \ldots, n$).

p-Операторы $\dot{A}_6^k$, $\dot{A}_7^k$, $\dot{A}_{13}^k$, $\dot{A}_{14}^k$, $\dot{A}_{16}^k$ вычисляют соответственно $c_{ij}^{(k)}$, $d_{ij}^{(k)}$ ($i = 2, 3, \ldots, k-1$; $j = 1, 2, \ldots, n$); $h_{ij}^{(k)}$, $\alpha_{ij}^{(k)}$ ($i = 1, 2, \ldots, k-1$; $j = 1, 2, \ldots, n$); $\alpha_{kj}^{(k)}$ ($j = 1, 2, \ldots, n$).

p-Операторы обобщенного условного перехода $\dot{P}_8^{(k)}(k)$ и $\dot{P}_{15}^{(k)}(k)$

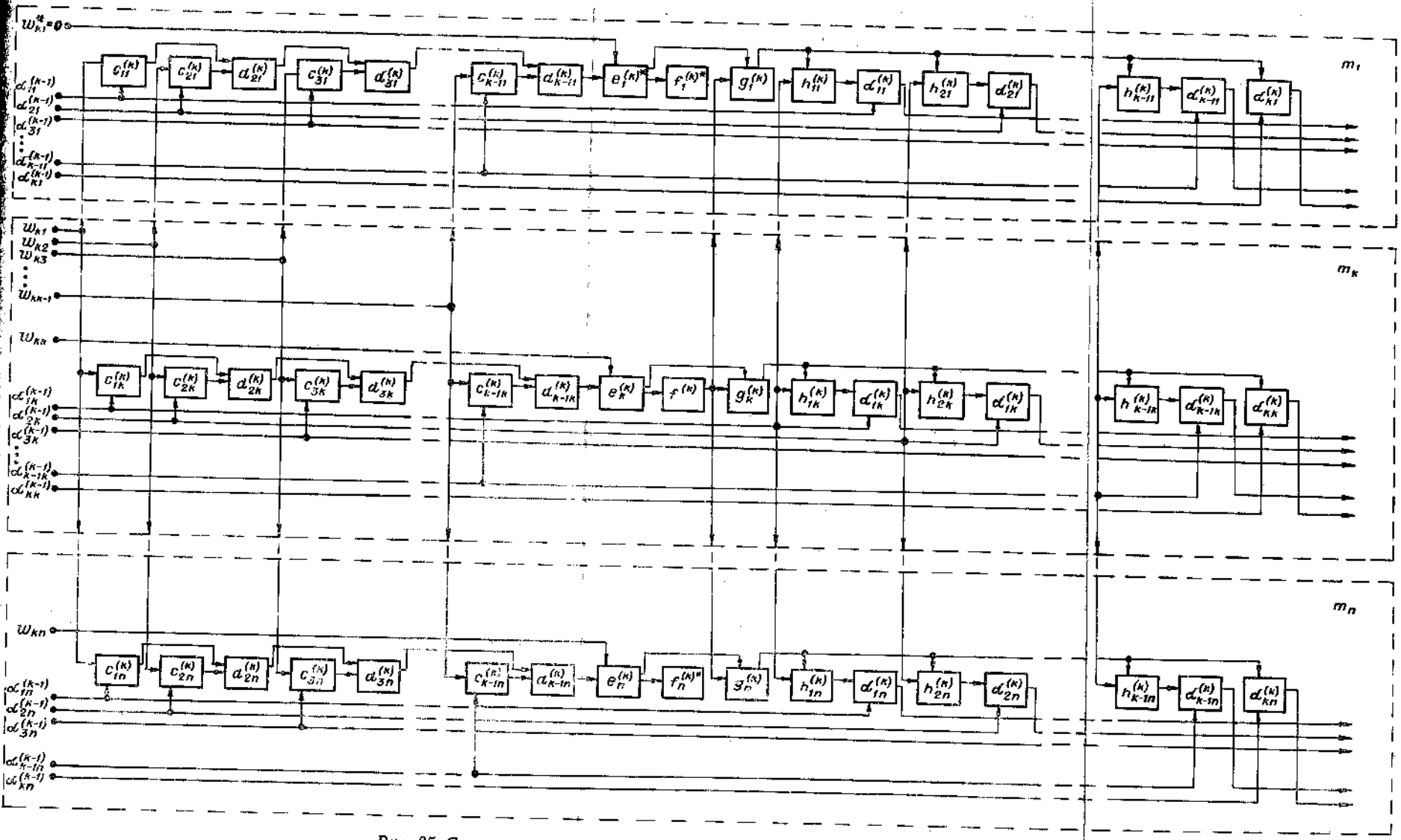


Рис. 85. Схема решения на УВС задачи обращения матриц методом пополнения.

устанавливают конец цикла после того, как он повторился соответственно k—2 и k раз. Число циклов меняется от этапа к этапу (с помощью оператора $\dot{F}_2^k$), поэтому эти p-операторы зависят от параметра k.

p-Оператор $\dot{P}_{17}$ осуществляет то же самое по отношению к числу этапов.

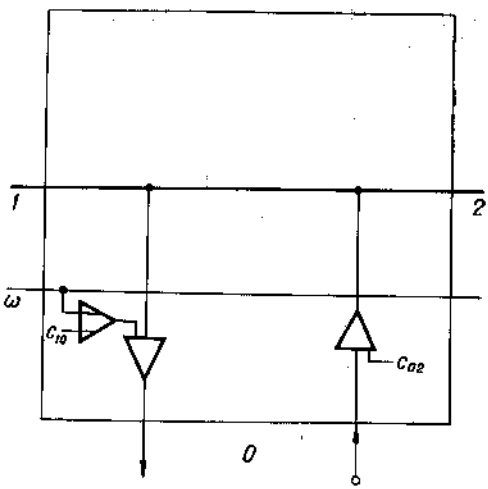


Рис. 86. Схема коммутатора.

p-Операторы $\dot{A}_9^k$, $\dot{Q}_{10}^k$ и $\dot{A}_{11}^k$ вычисляют $e_j^{(k)}$, $f_j^{(k)}$ и $g_j^{(k)}$ ($j = 1, 2, \ldots, n$). Как упоминалось выше, для единообразия программ всех ЭМ введены пустые операции $e_j^{(k)}$ ($j=1, 2, \ldots, k-1$) и $f_j^{(j)}$ ($j = 1, 2, \ldots, k-1, k+1, \ldots, n$). Для этого в памятях соответствующих ЭМ в ячейках, предназначенных для элементов w_{kj} ($j=1, 2, \ldots, k-1; k+1, 2, \ldots, n$) матрицы W, помещены нули. Вычисление $f_j^{(k)}$ во всех ЭМ, а не только в ЭМ m_k также ничего не изменяет.

p-Оператор $\dot{Q}_{10}^k$ результат вычисления функции $f_j^{(k)}$ направляет в канал связи. Благодаря тому, что состояние коммутаторов таково, что выходы всех элементарных машин, кроме m_k, отключены от канала связи, в последний поступит только требуемый код $f_k^{(k)}$.

p-Оператор $\dot{Я}_{18}$ устанавливает конец работы.

Из всего сказанного можно сделать вывод, что программы работы у всех n элементарных машин совпадают с точностью до адресов и значений индекс-регистров. Поэтому составление программы выполнения данного p-алгоритма на УВС мало чем отличается от обычного программирования. Так как все составляющие p-операторов одинаковы, то логическую схему работы системы (8.18) можно рассматривать как логическую схему работы одной ЭМ, заменяя каждый p-оператор соответствующим простым оператором.

Вычисление собственных значений матрицы по методу Данилевского [1]. Задача заключается в нахождении всех корней

λ_i характеристического полинома (собственных значений) матрицы A:

$$|A - Et| = \begin{vmatrix} a_{11} - t & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} - t & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} \dots & & a_{nn} - t \end{vmatrix} =$$

$$= (-1)^n [t^n - p_1 t^{n-1} - \dots - p_n] = 0. \qquad (8.19)$$

Идея метода Данилевского состоит в следующем. Данная матрица A рассматривается как матрица оператора в базисе

$$e_1 = (1, 0, \dots, 0)', \; e_2 = (0, 1, \dots, 0)', \dots, e_n = (0, 0, \dots, 1)'. \qquad (8.20)$$

В формуле (8.20) штрих (знак транспонирования) употребляется для указания, что в скобках находятся элементы столбца. Предполагается, что векторы e_1, $Ae_1, \dots, A^{n-1}e_1$ линейно независимы. Тогда

$$A^n e_1 = p_1 A^{n-1} e_1 + p_2 A^{n-2} e_1 + \dots + p_n e_1, \qquad (8.21)$$

где $p_1, \dots, p_n$ — искомые коэффициенты характеристического полинома.

В базисе e_1, $Ae_1, \dots, A^{n-1}e_1$ рассматриваемый оператор будет иметь матрицу Фробениуса

$$P = \begin{bmatrix} 0 & 0 & \dots & p_n \\ 1 & 0 & \dots & p_{n-1} \\ 0 & 1 & \dots & p_{n-2} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & p_1 \end{bmatrix} \qquad (8.22)$$

содержащую в явном виде искомые коэффициенты характеристического полинома.

Переход от базиса $e_1, e_2, \dots, e_n$ к базису e_1, $Ae_1, \dots, A^{n-1}e_1$ осуществляется за $n - 1$ шагов. Каждый шаг состоит в переходе от базиса $e_1, Ae_1, \dots, A^{k-1}e_1, e_{k+1}, \dots, e_n$ к базису $e_1, \dots, A^{k-1}e_1, A^k e_1, e_{k+2}, \dots, e_n$. При этом предполагается, что все промежуточные системы действительно базисы, т. е. состоят из линейно независимых векторов.

Обозначим через $A^{(k)}$ матрицу, полученную на $k - 1$-м шаге, так что $A = A^{(1)}$, $P = A^{(n)}$. Столбцами матрицы $A^{(k)}$ служат

координаты векторов $Ae_1, A^2e_1, \dots, A^k e_1, Ae_{k+1}, \dots, Ae_n$ в базисе $e_1, Ae_1, \dots, A^{k-1}e_1, e_{k+1}, \dots, e_n$. Поэтому первые $k - 1$ столбцов матрицы $A^{(k)}$ будут совпадать с одноименными столбцами матрицы Фробениуса P. Имеем

$$A^{(k+1)} = S_k^{-1} A^{(k)} S_k, \qquad (8.23)$$

где S_k — соответствующая матрица преобразования координат:

$$S_k = \begin{bmatrix} 1 \dots S_{1,k+1} & \dots 0 \\ 0 \dots S_{2,k+1} & \dots 0 \\ \dots & \dots \\ 0 \dots S_{k+1,k+1} & \dots 0 \\ \dots & \dots \\ 0 \dots S_{n,k+1} & \dots 1 \end{bmatrix} \quad S_k^{-1} = \begin{bmatrix} 1 \dots -\frac{S_{1,k+1}}{S_{k+1,k+1}} & \dots 0 \\ 0 \dots -\frac{S_{2,k+1}}{S_{k+1,k+1}} & \dots 0 \\ \dots & \dots \\ 0 \dots \frac{1}{S_{k+1,k+1}} & \dots 0 \\ \dots & \dots \\ 0 \dots -\frac{S_{n,k+1}}{S_{k+1,k+1}} & \dots 1 \end{bmatrix} \qquad (8.24)$$

где $S_{1,k+1}, \dots, S_{k+1,k+1}, \dots, S_{n,k+1}$ — элементы $a_{ik}^{(k)}$ k-го столбца матрицы $A^{(k)}$.

Матрица $A^{(k+1)}$ вычисляется в два приема. Сначала вычисляется вспомогательная матрица $B^{(k)} = S_{(k)}^{-1} A^{(k)}$, для чего достаточно $(k+1)$-ю строку матрицы $A^{(k)}$ умножить на $\frac{1}{a_{k+1,k}^{(k)}}$, а от каждой из остальных строк отнять полученную $k+1$-ю строку матрицы $B^{(k)}$, умноженную на соответствующий элемент k-го столбца матрицы $A^{(k)}$. Затем матрица $B^{(k)}$ умножается справа на S_k.

Матрицы $A^{(k)}$ и $B^{(k)}$ имеют вид

$$A^{(k)} = \begin{bmatrix} 0 & 0 \dots 0 & a_{1k}^{(k)} & a_{1k+1}^{(k)} & a_{1,k+2}^{(k)} & \dots a_{1n}^{(k)} \\ 1 & 0 \dots 0 & a_{2k}^{(k)} & a_{2k+1}^{(k)} & a_{2,k+2}^{(k)} & \dots a_{2n}^{(k)} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 \dots 1 & a_{kk}^{(k)} & a_{kk+1}^{(k)} & a_{k,k+2}^{(k)} & \dots a_{kn}^{(k)} \\ 0 & 0 \dots 0 & a_{k+1,k}^{(k)} & a_{k+1,k+1}^{(k)} & a_{k+1,k+2}^{(k)} & \dots a_{k+1,n}^{(k)} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 \dots 0 & a_{nk}^{(k)} & a_{n,k+1}^{(k)} & a_{n,k+2}^{(k)} & \dots a_{nn}^{(k)} \end{bmatrix} \qquad (8.25)$$

$$B^{(k)} = \begin{bmatrix} 0\ 0\ldots 0\ 0\ b^{(k)}_{1,\,k+1} & a^{(k+1)}_{1,\,k+2}\ldots a^{(k+1)}_{1n} \\ 1\ 0\ldots 0\ 0\ b^{(k)}_{2,\,k+1} & a^{(k+1)}_{2,\,k+2}\ldots a^{(k+1)}_{2n} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 0\ 0\ldots 1\ 0\ b^{(k)}_{k,\,k+1} & a^{(k+1)}_{k,\,k+2}\ldots a^{(k+1)}_{kn} \\ 0\ 0\ldots 0\ 1\ b^{(k)}_{k+1,\,k+1} & a^{(k+1)}_{k+1,\,k+2}\ldots a^{(k+1)}_{k+1,\,n} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 0\ 0\ldots 0\ 0\ b^{(k)}_{n,\,k+1} & a^{(k+1)}_{n,\,k+2}\ldots a^{(k+1)}_{nn} \end{bmatrix} \qquad (8.26)$$

Переход от матрицы $A^{(k)}$ к матрице $A^{(k+1)}$ происходит по формулам

$$\begin{aligned} b^{(k)}_{k+1,\,j} &= \frac{1}{a^{(k)}_{k+1,\,k}}\,a^{(k)}_{k+1,\,j}; \\ b^{(k)}_{ij} &= a^{(k)}_{ij} - a^{(k)}_{ik}b^{(k)}_{k+1,\,j} \quad (i \neq k+1); \\ a^{(k+1)}_{ij} &= b^{(k)}_{ij} \quad (j \neq k+1); \\ a^{(k+1)}_{i,\,k+1} &= \sum_{j=1}^{n} b^{(k)}_{ij}a^{(k)}_{jk}. \end{aligned} \qquad (8.27)$$

Сначала у матрицы $B^{(k)}$ вычисляются элементы строки с номером $k+1$, а затем — элементы всех остальных строк. Так как первые k столбцов матриц $B^{(k)}$ и $A^{(k+1)}$ совпадают с соответствующими столбцами матрицы Фробениуса, вычисляются только элементы $n-k$ последних столбцов. Последние столбцы матрицы $B^{(k)}$, начиная с $(k+2)$-го, совпадают с соответствующими столбцами искомой матрицы $A^{(k)}$. Поэтому остается определить только элементы $(k+1)$-го столбца.

Для перехода к логической схеме распишем эти формулы так, чтобы все операции были простыми:

$$b^{(k)}_{k+1,\,j} = \frac{a^{(k)}_{k+1,\,j}}{a^{(k)}_{k+1,\,k}} \qquad (j = k+1,\ k+2,\ldots,\ n); \qquad (8.28)$$

$$c^{(k)}_{ij} = a^{(k)}_{ik}b^{(k)}_{k+1,\,j} \qquad \begin{matrix}(i = 1,\ 2,\ldots,k,\ k+2,\ldots,n; \\ j = k+1, k+2,\ldots,n);\end{matrix} \qquad (8.29)$$

$$b^{(k)}_{ij} = a^{(k)}_{ij} - c^{(k)}_{ij} \qquad \begin{matrix}(i = 1,\ 2,\ldots,k, k+2,\ldots,n; \\ j = k+1, k+2,\ldots,n);\end{matrix} \qquad (8.30)$$

$$d^{(k)}_{ij} = b^{(k)}_{ij}a^{(k)}_{jk} \qquad (i = 1,\ 2,\ldots,n;\ j = k+1, k+2,\ldots,n); \qquad (8.31)$$

$$e^{(k)}_{ij} = e^{(k)}_{i,\,j-1} + d^{(k)}_{ij} \quad (i = 1,2,\ldots,n;\quad j = k+2, k+3,\ldots,n); \qquad (8.32)$$

$$\begin{aligned} e^{(k)}_{i,\,k+1} &= 0; \\ a^{(k+1)}_{i,\,k+1} &= e^{(k)}_{in} + a^{(k)}_{i-1,\,k} \quad (i = 2,\ldots,k+1); \\ a^{(k+1)}_{i,\,k+1} &= e^{(k)}_{in} \quad (i = 1, k+2, k+3,\ldots,n). \end{aligned} \qquad (8.33)$$

Нетрудно видеть, что минимальное число шагов, которое необходимо затратить для перехода от матрицы $A^{(k)}$ к матрице $A^{(k+1)}$ равно

$$h_m = 4 + \log_2(n-k), \qquad (8.34)$$

где 4 соответствует затратам на счет по функциям (8.28)—(8.31), а второе слагаемое — по (8.32) — (8.33).

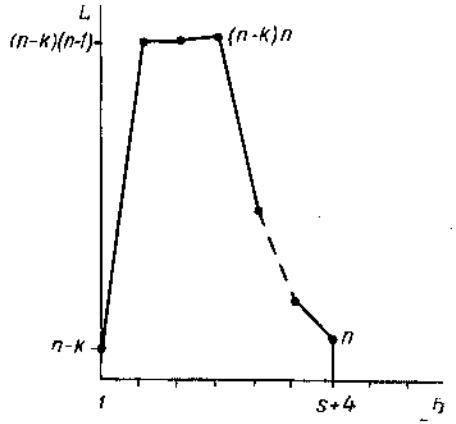


Рис. 87. Функция пошагового распределения минимальной длины для задачи нахождения собственных значений матрицы по методу Данилевского. $s = \log_2(n-k)$.

Сжатый p-кортеж (рис.87) имеет вид, далекий от оптимального. Если, как и в предыдущих задачах, вместо n^2 машин использовать n, то можно получить p-кортеж, близкий к оптимальному. Действительно, если в каждой машине m_i будем вычислять элементы i-й строки матрицы $A^{(k+1)}$ (процесс вычислений можно проиллюстрировать схемой (рис. 88), на которой показаны пути передачи информации), то логическая схема p-алгоритма может быть записана в виде

$$\dot{H}_1\dot{F}^k_2\dot{A}^k_3\dot{O}^k_4\dot{A}^k_5\dot{A}^k_6\dot{O}^k_7\dot{A}^k_8\dot{O}^{kj}_9\dot{A}^{kj}_{10}\dot{A}^{kj}_{11}\dot{O}^{kj}_{12}\dot{A}^{kj}_{13}\dot{A}^{kj}_{14}\dot{P}^k_{15}\,(j)\,\dot{H}_{16}\dot{O}^k_{17}\dot{A}^k_{18}\dot{P}_{19}\,(k)\,\dot{Я}_{20}, \qquad (8.35)$$

(с надстрочными $j=k+2,\ k+3,\ldots,\ n$ и $k=1,2,\ldots,\ n-1$)

где $\dot{H}_1$ и $\dot{H}_{16}$ — p-операторы настройки; первый настраивает коммутаторы всех ЭМ на прием и передачу информации, второй —

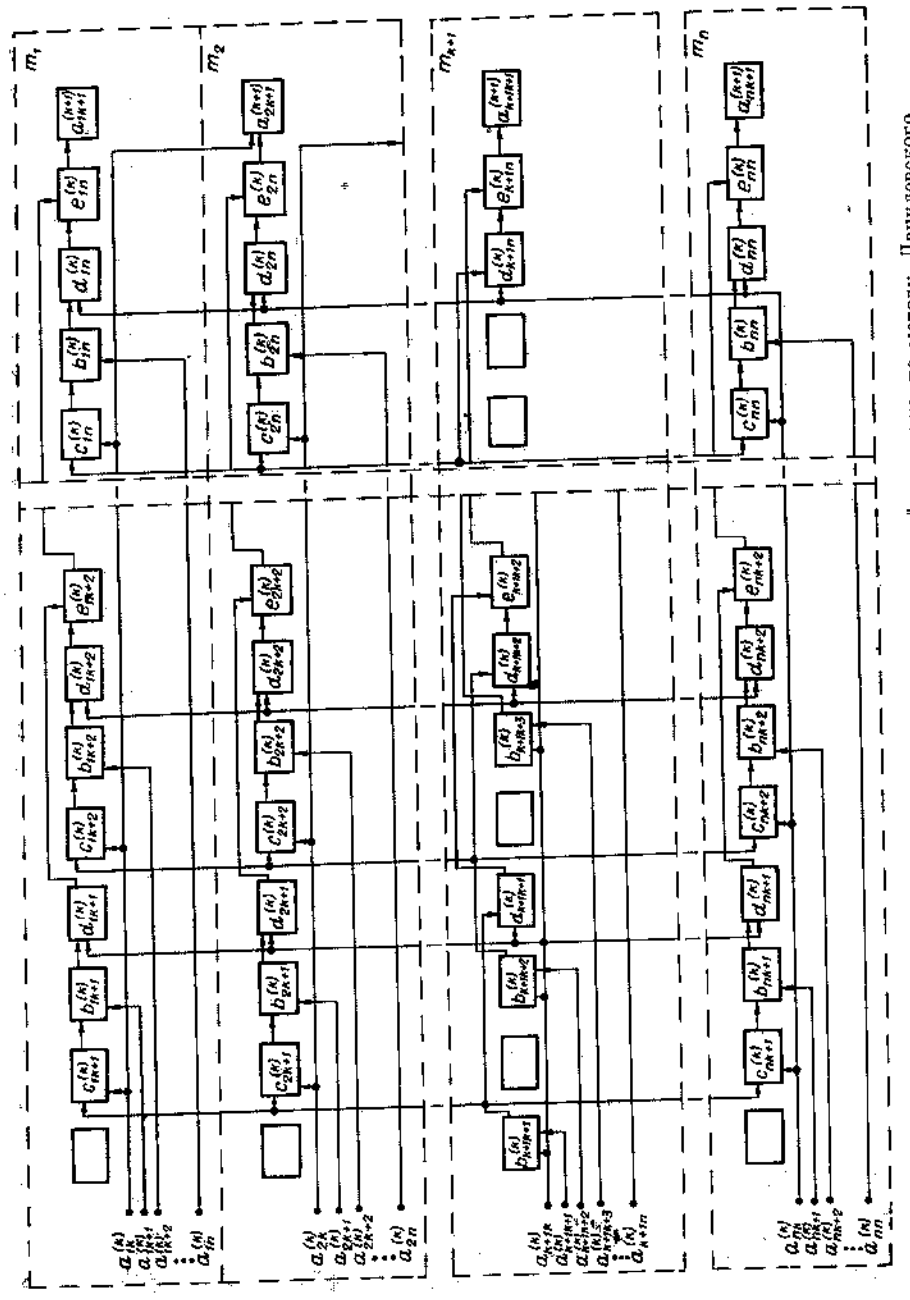


Рис. 88 Схема нахождения на УВС собственных значений матрицы по методу Данилевского.

коммутатор каждой ЭМ m_i на прием кода из ЭМ m_{i-1} и передачу кода в ЭМ m_{i+1};

$\dot{F}_2^k$ — оператор, формирующий очередной этап вычислений;

$\dot{A}_3^k$ — оператор, вычисляющий элемент $b_{k+1,\,k+1}^{(k)}$; только одна его составляющая $(k+1)$-я отлична от нуля;

$\dot{O}_4^k$ $\dot{O}_9^{kj}$ — p-операторы обмена, передающие коды ($b_{k+1,\,k+1}^{(k)}$ и $b_{k+1,\,j}^{(k)}$ $(j = k+2, \ldots, n)$, соответственно) из машины m_{k+1} во все ЭМ системы;

$\dot{A}_5^k$, $\dot{A}_{10}^{kj}$ — арифметические p-операторы, вычисляющие $c_{i,\,k+1}^{(k)}$ и $c_{ij}^{(k)}$ $(j = k+2, \ldots, n)$ соответственно; у этих p-операторов $(k+1)$-я составляющая пустая;

$\dot{A}_6^k$, $\dot{A}_{11}^{kj}$ — арифметические p-операторы, вычисляющие $b_{i,\,k+1}^{(k)}$ и $b_{k+1,\,k+2}^{(k)}$, $b_{ij}^{(k)}$ и $b_{k+1,\,j+1}^{(k)}$ $(i \neq k+1,\ j = k+2, \ldots, n)$ соответственно. Составляющая $(k+1)$-я у этих p-операторов несколько отличается от остальных составляющих.

$\dot{O}_7^k$ и $\dot{O}_{12}^k$ — p-операторы обмена, передающие во все остальные ЭМ соответственно код $a_{k+1,\,k}$, из ЭМ m_{k+1} и код a_{jk} из ЭМ m_j $(j = k+2, \ldots, n)$;

$\dot{A}_8$ и $\dot{A}_{13}^{kj}$ — арифметические p-операторы, вычисляющие $d_{i,\,k+1}^{(k)}$ и $d_{ij}^{(k)}$ $(j = k+2, \ldots, n)$ соответственно;

$\dot{A}_{14}^{kj}$ — арифметический p-оператор, вычисляющий e_{ij} $(i = 1, 2, \ldots, n;\ j = k+2, k+3, \ldots, n)$;

$\dot{P}_{15}^k(j)$ и $\dot{P}_{16}(k)$ — p-операторы обобщенного условного перехода, управляющие циклами по параметрам j $(j = k+2, k+3, \ldots, n)$ и k $(k = 1, 2, \ldots, n-1)$ соответственно;

$\dot{O}_{17}^k$ — p-оператор обмена, принимающий и передающий коды $a_{ik}^{(k)}$ для ЭМ m_i $(i = 2, 3, \ldots, k)$, только передающий для машины m_1 и только принимающий для машины m_{k+1};

$\dot{A}_{18}^k$ — арифметический p-оператор, вычисляющий элементы $a_{i,\,k+1}^{(k+1)}$. В ЭМ m_i $(i = 2, 3, \ldots, k+1)$ это осуществляется сложением $e_{in}^{(k)}$ с $a_{i-1,\,k}^{(k)}$, а во всех остальных ЭМ — пересылкой $e_{in}^{(k)}$ в ячейки памяти $\langle a_{i,\,k+1}^{(k)} \rangle$.

Рассмотрение трех задач линейной алгебры показывает, что процесс их решения может быть представлен в виде одинаковых (или почти одинаковых) ветвей вычисления, что упрощает их программирование. Ветви вычислений могут быть распределены между элементарными машинами таким образом, что хранимая информация V равномерно распределяется между ЭМ, т. е. в системе из n машин, каждая из них должна хранить объем информации V/n. Обмен информацией между ЭМ при этом может

быть осуществлен с помощью простейшей одно- или двухканальной системы связи.

Базовая точка характеристических функций этих задач лежит в области больших значений n (числа машин), что указывает на практически неограниченную возможность уменьшения времени их решения при сохранении максимального значения коэффициента использования машин δ.

Так как рассмотренные задачи типовые, то можно сделать вывод, что задачи линейной алгебры можно эффективно решать на УВС.

8.2. Решение экстремальных задач методами линейного программирования

Существует обширный класс экстремальных задач, широко применяемых на практике. Прежде всего сюда относятся экономические задачи:

распределение производственной программы между несколькими предприятиями; вопросы кооперирования отдельных предприятий; календарное распределение имеющейся программы; распределение затрат между отдельными видами продукции при комплексном ее выпуске; выбор оптимальных вариантов технологии, в частности оценка эффективности нового технологического способа и новой техники; рациональный раскрой промышленных товаров; вопросы транспортировки, в частности рациональное прикрепление пунктов потребления некоторого продукта к имеющимся пунктам производства; планирование в государственном или республиканском масштабах; решение общих задач планового и экономического анализа и т. п.

Методы линейного программирования, разработанные впервые Л. В. Канторовичем [5], широко применяются в различных отраслях промышленности, в сельском хозяйстве, в связи и на транспорте, в военных приложениях, при конструировании, регулировании уличного движения и т. д.

Они широко применяются также при решении задач теории игр для отыскания стратегии, дающей максимально возможный в данной игре выигрыш. Оказалось, что общая задача линейного программирования эквивалентна произвольной игре двух партнеров с нулевой суммой и конечным числом стратегий [6].

В данном параграфе рассматривается решение на вычислительной системе общей задачи линейного программирования, к которой сводятся все задачи линейного программирования, и общей транспортной задачи [7—12].

Решение общей задачи линейного программирования модифицированным симплексным методом. Общая задача линейного программирования заключается в отыскании функции

$$c_1x_1 + c_2x_2 + \ldots + c_jx_j + \ldots + c_nx_n \tag{8.36}$$

при следующих ограничениях:

$$x_j \geqslant 0,\ j = 1,\ 2,\ldots,\ n \tag{8.37}$$

и

$$\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \ldots + a_{1j}x_j + \ldots + a_{1n}x_n = b_1;\\
&a_{21}x_1 + a_{22}x_2 + \ldots + a_{2j}x_j + \ldots + a_{2n}x_n = b_2;\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{ij}x_j + \ldots + a_{in}x_n = b_i;\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mj}x_j + \ldots + a_{mn}x_n = b_m,
\end{aligned} \tag{8.38}$$

где c_j, a_{ij} и b_i — заданные постоянные величины, $b_i \geqslant 0$, $m < n$.

Значения $x_1, x_2, \ldots, x_n$, удовлетворяющие условиям (8.37) и (8.38), принято называть *планом*, а если при этом функция (8.36) принимает минимальное значение, то — *оптимальным планом*.

В модифицированном симплексном методе вводятся дополнительные переменные $x_{n+1}, x_{n+2}, \ldots, x_{n+m} \geqslant 0$, а также x_{n+m+1} и x_{n+m+2}. Здесь

$$x_{n+m+1} = -\sum_{j=1}^{n} a_{m+1,j}x_j, \tag{8.39}$$

где $a_{m+1,j} = c_j$;

$$x_{n+m+2} = b_{m+2} - \sum_{j=1}^{n} a_{m+2,j}x_j, \tag{8.40}$$

где

$$\left.\begin{aligned}
a_{m+2,j} &= -\sum_{i=1}^{m} a_{ij},\ j = 1, 2, \ldots, n;\\
b_{m+2} &= -\sum_{i=1}^{m} b_i.
\end{aligned}\right\} \tag{8.41}$$

Задача сводится к максимизации x_{n+m+1} при условиях:

$$\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + x_{n+1} = b_1;\\
&a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + x_{n+2} = b_2;\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots
\end{aligned}$$

$$
\begin{aligned}
a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n + x_{n+m} &= b_m;\\
a_{m+1,1}x_1 + a_{m+1,2}x_2 + \ldots + a_{m+1,n}x_n + x_{n+m+1} &= 0;\\
a_{m+2,1}x_1 + a_{m+2,2}x_2 + \ldots + a_{m+2,n}x_n + x_{n+m+2} &= b_{m+2}.
\end{aligned}
\tag{8.42}
$$

$$
\left.
\begin{aligned}
x_1 &\geqslant 0;\\
x_2 &\geqslant 0;\\
&\vdots\\
x_n &\geqslant 0;\\
x_{n+1} &\geqslant 0;\\
&\vdots\\
x_{n+m} &\geqslant 0.
\end{aligned}
\right\}
\tag{8.43}
$$

Задача решается в два этапа. На первом добиваются того, чтобы $x_{n+m+2} = 0$. Так как на основании (8.40) и (8.42)

$$
-x_{n+m+2} = x_{n+1} + x_{n+2} + \ldots + x_{n+m} \tag{8.44}
$$

и x_{n+1}, $x_{n+2}, \ldots,$ $x_{n+m} \geqslant 0$, все x_{n+1}, $x_{n+2}, \ldots,$ x_{n+m} равны нулю. Это означает, что x_j $(j = 1, 2, \ldots, m)$ удовлетворяют условию (8.38), т. е. образуют план. На втором этапе этот план рассматривается как первоначальный (опорный), последовательным улучшением которого получают оптимальный план.

Рассмотрим процесс выполнения обоих этапов.

Этап I. Здесь повторяется один и тот же цикл вычислений до тех пор, пока не станет $x_{n+m+2} = 0$, либо не будет установлена невозможность решения. В качестве исходного плана берутся значения переменных $x_j = 0$ $(j = 1, \ldots, n)$ и $x_{n+i} = b_i$ $(i = 1, 2, \ldots, m)$. Основные операции выполняются над элементами матрицы, составленной из коэффициентов системы условий (8.42).

$$
\overline{A} = \begin{bmatrix}
a_{11} & a_{12} & \ldots & a_{1j} & \ldots & a_{1n};\\
a_{21} & a_{22} & \ldots & a_{2j} & \ldots & a_{2n};\\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots\\
a_{i1} & a_{i2} & \ldots & a_{ij} & \ldots & a_{in};\\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots\\
a_{m1} & a_{m2} & \ldots & a_{mj} & \ldots & a_{mn};\\
a_{m+1,1} & a_{m+1,2} & \ldots & a_{m+1,j} & \ldots & a_{m+1,n};\\
a_{m+2,1} & a_{m+2,2} & \ldots & a_{m+2,j} & \ldots & a_{m+2,n}
\end{bmatrix}
\tag{8.45}
$$

и вспомогательных матриц, получающихся на каждом цикле вычислений:

$$
U_s = \left[\begin{array}{ccccc|c}
u_{11}^{(s)} & u_{12}^{(s)} & \ldots & u_{1j}^{(s)} & \ldots & u_{1m}^{(s)} & 0\ 0\\
u_{21}^{(s)} & u_{22}^{(s)} & \ldots & u_{2j}^{(s)} & \ldots & u_{2m}^{(s)} & 0\ 0\\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots\\
u_{i1}^{(s)} & u_{i2}^{(s)} & \ldots & u_{ij}^{(s)} & \ldots & u_{im}^{(s)} & 0\ 0\\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots\\
u_{m1}^{(s)} & u_{m2}^{(s)} & \ldots & u_{mj}^{(s)} & \ldots & u_{mm}^{(s)} & 0\ 0\\
\hline
u_{m+1,1}^{(s)} & u_{m+1,2}^{(s)} & \ldots & u_{m+1,j}^{(s)} & \ldots & u_{m+1,m}^{(s)} & 1\ 0\\
u_{m+2,1}^{(s)} & u_{m+2,2}^{(s)} & \ldots & u_{m+2,j}^{(s)} & \ldots & u_{m+2,m}^{(s)} & 0\ 1
\end{array}\right]
\tag{8.46}
$$

В качестве исходной матрицы U_1 возьмем единичную матрицу E.

Первый этап состоит из следующих шагов.

Шаг. 1. Сравниваем $x_{m+n+2}^{(s)}$ с нулем. Если $x_{m+n+2}^{(s)} = 0$, переходим на второй этап. Если $x_{m+n+2}^{(s)} < 0$, подсчитываем

$$
\delta_j^{(s)} = u_{m+2,1}^{(s)} a_{1j} + u_{m+2,2}^{(s)} a_{2j} + \ldots + u_{m+2,\,m+2}^{(s)} a_{m+2,j}; \quad j = 1, 2, \ldots, n. \tag{8.47}
$$

Шаг 2. Сравниваем $\delta_j^{(s)}$ с нулем. Если все $\delta_j^{(s)} \geqslant 0$, то задача не имеет ни одного плана. Если хотя бы одно $\delta_j^{(s)} < 0$, то определяется

$$
\delta_k^{(s)} = \min \delta_j^{(s)}. \tag{8.48}
$$

Если минимум достигается на нескольких $\delta_j^{(s)}$, то выбираем $\delta_j^{(s)}$ с наименьшим индексом.

Шаг 3. Подсчитываем

$$
x_{ik}^{(s)} = u_{i1}^{(s)} a_{1k} + u_{i2}^{(s)} a_{2k} + \ldots + u_{i,m+2}^{(s)} a_{m+2,k} \tag{8.49}
$$

для $i = 1, 2, \ldots, m + 2$ и определяем

$$
\theta_0^{(s)} = \min_i \frac{x_i^{(s)}}{x_{ik}^{(s)}} = \frac{x_l^{(s)}}{x_{lk}^{(s)}} \quad (i = 1, 2, \ldots, m). \tag{8.50}
$$

Минимум берется по $x_{ik} > 0$. Если минимум достигается одновременно для нескольких значений i, выбираем x_l с наименьшим индексом.

Шаг 4. Определяем новые значения переменных в опорном плане.

$$x_i^{(s+1)} = x_i^{(s)} - \frac{x_l^{(s)}}{x_{lk}^{(s)}} x_{ik}^{(s)} \quad (i \neq k); \tag{8.51}$$

$$x_k^{(s+1)} = \frac{x_l^{(s)}}{x_{lk}^{(s)}}.$$

Затем определяем элементы матрицы U_{s+1} по формулам:

$$u_{ij}^{(s+1)} = u_{ij}^{s} - \frac{u_{lj}^{(s)}}{x_{lk}^{(s)}} x_{ik}^{(s)} \quad \begin{pmatrix} j = 1, 2, \ldots, m \\ i \neq l \end{pmatrix}; \tag{8.52}$$

$$u_{lj}^{(s+1)} = \frac{u_{lj}^{(s)}}{x_{lk}^{(s)}}.$$

После этого снова повторяем шаг 1 первого этапа.

Этап II. *Шаг 1.* Подсчитываем

$$\gamma_j^{(s+1)} = u_{m+1,1}^{(s+1)} a_{1j} + u_{m+1,2}^{(s+1)} a_{2j} + \ldots + u_{m+1,\,m+2}^{(s+1)} a_{m+2,\,j}; \tag{8.53}$$

$$j = 1, 2, \ldots, n,$$

где $u_{m+i,\,i}$ — значение элементов матрицы U_{s+1}, при котором $x_{n+m+2} = 0$.

Шаг 2. Сравниваем $\gamma_j^{(s+1)}$ с 0. Если все $\gamma_j^{(s+1)} \geqslant 0$, то x_{n+m+1} достигает своего максимума и соответствующий план является оптимальным. Если хотя бы одно из $\gamma_j^{s+1} < 0$, то необходимо перейти к новому базису. Пусть

$$\gamma_k^{(s+1)} = \min_j \gamma_j^{(s+1)}. \tag{8.54}$$

Тогда переменную $x_k^{(s+1)}$ следует ввести в новый план. Если при этом минимум достигается одновременно на нескольких $\gamma_j^{(s+1)}$, то берем $\gamma_j^{(s+1)}$ с наименьшим индексом.

Шаг 3. Подсчитываем

$$x_{ik}^{(s+1)} = u_{i1}^{(s+1)} a_{1k} + u_{i2}^{(s+1)} a_{2k} + \ldots + u_{i,\,m+2}^{(s+1)} a_{m+2,k} \tag{8.55}$$

для $i = 1, 2, \ldots, m, m+1, m+2$. Переменную $x_l^{(s+1)}$, подлежащую исключению из старого плана, определяем из соотношения

$$\theta_0 = \min_i \frac{x_i^{(s+1)}}{x_{ik}^{(s+1)}} = \frac{x_l^{(s+1)}}{x_{lk}^{(s+1)}}, \quad i = 1, 2, \ldots, m, \tag{8.56}$$

где минимум берется по $x_{ik}^{(s+1)} > 0$. Если все $x_{ik}^{(s+1)} < 0$, решение прекращаем, так как это означает, что линейная форма задачи не ограничена сверху.

Шаг 4. Определяем новые значения переменных:

$$x_i^{(s+2)} = x_i^{(s+1)} - \frac{x_l^{(s+1)}}{x_{lk}^{(s+1)}} x_{ik}^{(s-1)} \quad \text{для } i \neq k; \tag{8.57}$$

$$x_k^{(s+2)} = \frac{x_l^{(s+1)}}{x_{lk}^{(s-1)}}.$$

Элементы матрицы U преобразуются по той же формуле (8.52). Итерации продолжаем до тех пор, пока не будет определен оптимальный план, либо не будет установлена неограниченность линейной формы задачи.

Вычисления на этапе II отличаются от соответствующих вычислений на этапе I только тем, что вместо строки матрицы U_s с номером $m+2$ берется строка той же матрицы с номером $m+1$.

Представим, как и в предыдущих случаях, процесс решения задачи в виде простых операций. Исключение сделаем для операции нахождения наименьшего из некоторой группы чисел.

Шаг 1. 1. Сравнение $x_{m-n+2}^{(s)}$ с нулем.

2. $d_{ij}^{(s)} = u_{m+2,\,i}^{(s)} a_{ij}$ $(i = 1, 2, \ldots, m+2;\ j = 1, 2, \ldots, n)$. (8.58)

3. $e_{ij}^{(s)} = e_{i-1,\,j}^{(s)} + d_{ij}^{(s)}$ $(i = 1, 2, \ldots, m+2;\ j = 1, 2, \ldots, n)$, (8.59)

$$e_{0j}^{(s)} = 0.$$

Шаг 2. 4. Сравнение $\delta_j^{(s)} = e_{m+2,\,j}^{(s)}$ с предыдущим $\delta_{j-1}^{(s)}$, запоминание наименьшего из них $\delta_k^{(s)} = \min_j \delta_j^{(s)}$ $(j = 1, 2, \ldots, n)$.

5. Сравнение $\delta_k^{(s)}$ с нулем, если $\delta_k^{(s)} \geqslant 0$, то останов.

Шаг 3. 6. $f_{ij}^{(s)} = u_{ij}^{(s)} \cdot a_{jk}$ $(i = 1, 2, \ldots, m+2;\ j = 1, 2, \ldots, m+2)$. (8.60)

7. $g_{ij}^{(s)} = g_{i,j-1}^{(s)} + f_{ij}^{(s)}$ $(g_{i0}^{(s)} = 0)$. (8.61)

8. Сравнение $g^{(s)}_{l,\,m+2} = x^{(s)}_{lk}$ с нулем.

Для $x_{lk} > 0$

9. $$h^{(s)}_{lk} = \frac{x^{(s)}_l}{x^{(s)}_{lk}}. \qquad (8.62)$$

10. Сравнение $h^{(s)}_{lk}$ друг с другом и нахождение наименьшего из них $\theta^{(s)}_0 = \min\limits_i h^{(s)}_{ik}$.

Шаг 4. 11. $p^{(s)}_{ik} = \theta^{(s)}_0 x^{(s)}_{ik}$.

12. $$x^{(s+1)}_l = x^{(s)}_l - p^{(s)}_{ik} \quad (i \neq k). \qquad (8.63)$$

$$x^{(s+1)}_k = \frac{x^{(s)}_l}{x^{(s)}_{lk}} = \theta^{(s)}_0.$$

13. $$q^{(s)}_{lj} = \frac{u^{(s)}_{lj}}{x^{(s)}_{lk}}. \qquad (8.64)$$

14. $t^{(s)}_{ij} = q^{(s)}_{li} x^{(s)}_{ik}$.

15. $$u^{(s+1)}_{ij} = u^{(s)}_{ij} - t^{(s)}_{lj} \quad (i \neq l); \qquad (8.65)$$

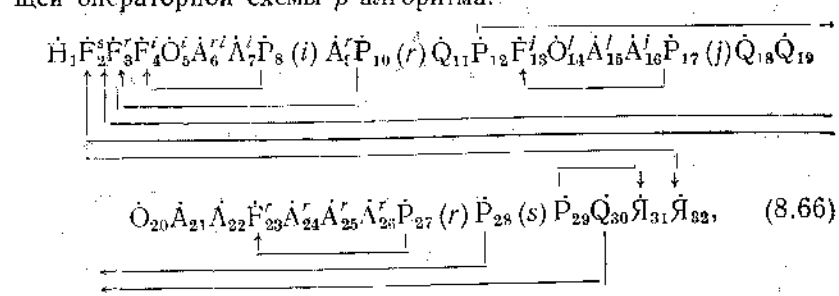
$$u^{(s+1)}_{lj} = q^{(s)}_{lj}.$$

16. Сравнение $x^{(s)}_{m+n+2}$ с нулем. При $x^{(s)}_{m+n+2} < 0$ переходим к очередной итерации.

17. При $x^{(s)}_{m+n+2} = 0$ во всех командах изменяются адреса элементов $u^{(s)}_{m+2,\,i}$ на адреса $u^{(s)}_{m+1,\,i}$ и изменяются признаки остановов, после чего выполняется очередная итерация этапа II.

В данной задаче число машин удобно взять равным $m + 2$. В память каждой машины m_i $(i = 1, 2, \ldots, m + 2)$ поместим i-ю строку матрицы U_s, соответствующие значения компонент $x^{(s)}_i$, номер компоненты плана $i^{(s)}$, а также $\frac{n}{m+2}$ столбцов матрицы A, так что в ЭМ m_1 будут находиться столбцы с номерами 1, 2, ..., $\frac{n}{m+2}$, в ЭМ m_2 — $\frac{n}{m+2} + 1$, $\frac{n}{m+2} + 2$, ..., $\frac{2n}{m+2}$ и т. д.

В каждой ЭМ будем вычислять количество δ_j, равное $\frac{n}{m+2}$. Решение данной задачи может быть представлено в виде следующей операторной схемы p-алгоритма:



$$\dot{H}_1\dot{F}^s_2\dot{F}^r_3\dot{F}^i_4\dot{O}^i_5\dot{A}^{ri}_6\dot{A}^i_7\dot{P}_8(i)\,\dot{A}^r_9\dot{P}_{10}(r)\,\dot{Q}_{11}\dot{P}_{12}\dot{F}^j_{13}\dot{O}^j_{14}\dot{A}^j_{15}\dot{A}^j_{16}\dot{P}_{17}(j)\dot{Q}_{18}\dot{Q}_{19}$$

$$\dot{O}_{20}\dot{A}_{21}\dot{A}_{22}\dot{F}^r_{23}\dot{A}^r_{24}\dot{A}^r_{25}\dot{A}^r_{26}\dot{P}_{27}(r)\,\dot{P}_{28}(s)\,\dot{P}_{29}\dot{Q}_{30}\dot{Я}_{31}\dot{Я}_{32}, \qquad (8.66)$$

где $\dot{H}_1$ — p-оператор настройки, соединяющий входы и выходы всех ЭМ системы с общим каналом связи;

$\dot{F}^s_2$ — p-оператор, формирующий выполнение цикла по s во всех ЭМ;

$\dot{F}^r_3$ — p-оператор, формирующий выполнение цикла по r во всех ЭМ $\left(r = 1, 2, \ldots, \frac{n}{m+2}\right)$;

$\dot{F}^i_4$ — p-оператор, формирующий выполнение цикла по i во всех ЭМ $(i = 1, 2, \ldots, m + 2)$;

$\dot{O}^i_5$ — p-оператор, пересылающий код $u_{m+2,\,i}$ из машины m_{m+2} во все остальные ЭМ;

$\dot{A}^{ri}_6$ — p-оператор, вычисляющий $d^{(s)}_{ir}$ во всех ЭМ;

$\dot{A}^i_7$ — p-оператор, вычисляющий $e^{(s)}_{ir}$ во всех ЭМ;

$\dot{P}_8(i)$ — p-оператор обобщенного условного перехода, осуществляющий при $i \leqslant m + 2$ повторение цикла;

$\dot{A}^r_9$ — p-оператор, выполняющий сравнение $e^{(s)}_{m+2,\,r} = \delta^{(s)}_r$ с $\delta^{(s)}_{r-1}$ и оставляющий в ячейке $\langle\delta^{(s)}_{r-1}\rangle$ наименьшее из этих чисел;

$\dot{P}_{10}(r)$ — p-оператор обобщенного условного перехода, осуществляющий при $r \leqslant \frac{n}{m+2}$ повторение цикла;

$\dot{Q}_{11}$ — обобщенный p-оператор, определяющий $\delta_k = \min\limits_i \delta_i$ $(i = 1, 2, \ldots, m + 2)$ (δ_i находятся в ЭМ с соответствующим номером i);

$\dot{P}_{12}$ — p-оператор обобщенного условного перехода, который сравнивает δ_k с нулем, при $\delta_k \geqslant 0$ осуществляет переход к p-оператору $\dot{Я}_{32}$; при $\delta_k < 0$ — к p-оператору $\dot{F}^j_{13}$;

$\dot{F}^j_{13}$ — p-оператор, формирующий выполнение цикла по j;

$\dot{O}_{14}^{l}$ — p-оператор обмена, посылающий код a_{jk} из ЭМ (той же, в которой находится δ_k) во все машины;

$\dot{A}_{15}^{l}$ — арифметический p-оператор, вычисляющий $f_{ij}^{(s)}$ во всех ЭМ;

$\dot{A}_{16}^{l}$ — арифметический p-оператор, вычисляющий $g_{ij}^{(s)}$ во всех ЭМ;

$\dot{P}_{17}(j)$ — p-оператор обобщенного условного перехода, осуществляющий при $j \leqslant m+2$ повторение цикла по j, при $j > (m+2)$-переход к p-оператору $\dot{Q}_{18}$;

$\dot{Q}_{18}$ — обобщенный p-оператор, сравнивающий $g_{i,\,m+2}^{(s)} = x_{ik}^{(s)}$ с нулем. При $x_{ik}^{(s)} > 0$ вычисляется $h_{ik}^{(s)}$, при $x_{ik}^{(s)} \leqslant 0$ в ячейку $\langle h_{ik}^{(s)} \rangle$ засылается наибольшее из возможных чисел (например, число 111 . . . 11);

$\dot{Q}_{19}$ — обобщенный p-оператор, находящий $\theta_0^{(s)} = \min\limits_{i} h_{ik}^{(s)}$. Аналогичен p-оператору $\dot{Q}_{11}$;

$\dot{O}_{20}$ — p-оператор обмена, передающий $\theta_0^{(s)}$ из ЭМ m_l во все ЭМ;

$\dot{A}_{21}$ — арифметический p-оператор, вычисляющий $p_{ik}^{(s)}$ во всех ЭМ, кроме m_l, в которой выполняется пустая операция;

$\dot{A}_{22}$ — арифметический p-оператор, вычисляющий $x_i^{(s+1)}$ во всех ЭМ, кроме m_l, в которой вместо $x_k^{(s+1)}$ подставляется θ_0;

$\dot{F}_{23}^{r}$ — p-оператор, формирующий изменение цикла по параметру $r\left(r = 1, 2, \ldots, \frac{n}{m+2}\right)$;

$\dot{A}_{24}^{r}$ — арифметический p-оператор, вычисляющий $q_{ij}^{(s)}$ во всех ЭМ;

$\dot{A}_{25}^{r}$ — арифметический p-оператор, вычисляющий $i_{ir}^{(s)}$ во всех ЭМ;

$\dot{A}_{26}^{r}$ — арифметический p-оператор, вычисляющий $u_{ir}^{(s+1)}$ во всех ЭМ, кроме m_l, в которой в ячейку $\langle u_{lj}^{(s+1)} \rangle$ засылается $q_{lj}^{(s)}$;

$\dot{P}_{27}(r)$ — p-оператор обобщенного условного перехода, осуществляющий при $r \leqslant n/(m+2)$-переход к p-оператору $\dot{F}_{23}^{r}$, при $r > n/(m+2)$ — к следующему p-оператору;

$\dot{P}_{28}(s)$ — p-оператор обобщенного условного перехода при $x_{m+n+2} < 0$, выполняющий переход к p-оператору $\dot{F}_{2}^{(s)}$, а при $x_{m+n+2} = 0$ — переход к p-оператору $\dot{P}_{29}$;

$\dot{P}_{29}$ — p-оператор обобщенного условного перехода. Если выполняется этап I, то он указывает на переход к следующему p-оператору, если — этап II, то — к $\dot{Я}_{31}$;

$\dot{Q}_{30}$ — p-оператор, формирующий изменения в программах, связанных с переходом на этап II (во всех ЭМ адрес $m+n+2$ заменяется на $m+n+1$ и изменяется признак p-оператора $\dot{Я}_{32}$, после чего передается управление p-оператору $\dot{F}_{2}^{(s)}$);

$\dot{Я}_{31}$ — p-оператор останова при окончании решения задачи;

$\dot{Я}_{32}$ — p-оператор останова с признаком. Признак указывает, на каком этапе произошел останов.

Как можно видеть из схемы p-алгоритма и соответствующих формул, потеря времени из-за простоя ЭМ при решении данной задачи на УВС будет незначительна, так как основное время затрачивается на вычисление p-операторов $\dot{A}_6$, $\dot{A}_7$, при выполнении которых работают все $m+2$ ЭМ.

Таким образом, УВС из $m+2$ элементарных машин при решении данной задачи дает выигрыш во времени по сравнению с одной ЭМ в $m+2$ раз (без учета в различии объемов оперативных памятей). Увеличение числа ЭМ до n может дать существенный выигрыш во времени, однако ЭМ при этом будут использоваться менее эффективно.

Решение на УВС общей транспортной задачи. Математическая формулировка транспортной задачи состоит в следующем. Определить значения переменных x_{ij}, которые минимизируют стоимость перевозок

$$\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{8.67}$$

при условиях:

$$\sum_{j=1}^{n} x_{ij} = a_i \quad (i = 1, 2, \ldots, m); \tag{8.68}$$

$$\sum_{i=1}^{m} x_{ij} = b_j \quad (j = 1, 2, \ldots, n); \tag{8.69}$$

$$x_{ij} \geqslant 0.$$

Для совместности уравнений необходимо, чтобы

$$\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} = \sum_{j=1}^{n} \sum_{i=1}^{m} x_{ij} = \sum_{i} a_i = \sum_{j} b_j = A. \tag{8.70}$$

Транспортная задача — это задача линейного программирования с $m + n$ уравнениями и mn переменными. Поэтому для ее решения пригоден любой метод линейного программирования. Однако при больших значениях m и n это приводит к огромному количеству вычислений. Для транспортной задачи разработаны специальные методы, которые можно разделить на две группы.

В первой группе решение начинается с отыскания допустимого решения. Если оно не оптимально, то дальнейший ход решения заключается в постепенном приведении его к искомому оптимуму. В эту группу входят метод потенциалов Л. В. Канторовича [9], распределительный метод и ряд его модификаций [10].

Методы второй группы в известном смысле противоположны методам первой группы. Здесь сначала описывается распределение, которое не обязательно удовлетворяет требованию допустимости, но строго соответствует требованию оптимальности. Процесс решения состоит в постепенном введении распределения в границы допустимости при соблюдении условия оптимальности. К этой группе относятся метод разрешающих слагаемых, метод дифференциальных рент, венгерский метод [11, 12].

Рассмотрение показывает, что вычислительная система для обеих групп методов оказывается эффективной. Для примера рассмотрим реализацию на вычислительной системе одной из модификаций распределительного метода [10].

Условия транспортной задачи можно представить в виде следующей таблицы

i \ j		b_1	b_2		b_j		b_n
	v_j	v_1	v_2		v_j		v_n
a_1	u_1	c_{11} / x_{11}	c_{12} / x_{12}				
a_2	u_2	c_{21} / x_{21}	c_{22} / x_{22}				
a_i	u_i				c_{ij} / x_{ij}		
a_m	u_m						

в которой u_i и v_j — числа, удовлетворяющие условию

$$u_i + v_j = c_{ij}. \tag{8.72}$$

Алгоритм решения транспортной задачи описывается следующим образом.

1. Определяем первый выбор (исходный план). Находим элементы первой строки матрицы $X = (x_{ij})$. В первой строке матрицы $C = (c_{ij})$ отыскиваем наименьший элемент. Пусть это будет элемент c_{1j_1}. Тогда $x_{1j_1} = \min(a_1, b_{j_1})$ (x-выбранный элемент). Если $a_1 > b_{j_1}$, то отыскиваем в той же строке второй наименьший элемент c_{1j_2}, удовлетворяющий условию $c_{1j_1} \geqslant c_{1j_2}$; тогда $x_{1j_2} = \min(a_1 - x_{1j_1}, b_{j_2})$. Шаги продолжаем до тех пор, пока не удовлетворим первому уравнению $a_1 = \sum_{j=1}^{n} x_{1j}$. Если на некотором шаге k окажется, что остаток от a_1 точно равен соответствующему b_{j_k}, то полагаем $x_{1j_k} = b_{j_k}$, а все остальные $x_{1j} = 0$ $(j \neq j_1, j_2, \ldots, j_k)$. Если $b_{j_1} > a_1$, то полагаем $x_{1j_1} = a_1$, а все остальные $x_{1j} = 0$.

После этого переходим ко второй строке, третьей и т. д. Полученное первое решение будет исходным планом и содержит $m + n - 1$ элементов.

2. Проверка опорного плана на оптимальность:

а) для x-выбранных элементов составляем систему уравнений $u_i + v_j = c_{ij}$ и находим значения u_i и v_j;

б) преобразуем для всех не x-выбранных элементов таблицы по формуле

$$\bar{c}_{ij} = c_{ij} - u_i - v_j; \tag{8.73}$$

в) если для всех не x-выбранных элементов таблицы элементы $\bar{c}_{ij}$ окажутся неотрицательными, то первый выбор оптимален и процесс решения заканчиваем. Если нет, то переходим к шагу 3.

3. Находим в таблице наименьшее из $\bar{c}_{ij}$.

4. Образуем замкнутую цепь наибольшего отрицательного элемента с обращенными в нуль x-выбранными элементами. Для этого вычеркиваем строки и столбцы, содержащие по одному отмеченному нулю (так как для образования цепи надо иметь в столбце или строке не менее двух нулей). После первого вычеркивания во всей таблице делаем второе и т. д., пока не придем к положению, из которого сразу видна замкнутая цепь.

5. Строим новый выбор, передвигая по цепи число x_{ij}, равное наименьшему в четной полуцепи (элементу x_{ij}, для которого $\bar{c}_{ij}$ был наибольшим по абсолютной величине отрицательным элементом, присваиваем первый номер, далее нумерация идет по часовой стрелке). Наименьшее x_{ij} прибавляем к нечетным элементам и вычитаем из четных элементов цепи. Наименьшее в четной полуцепи x_{ij} исключаем из числа x-выбранных элементов. Так как наибольшему отрицательному элементу соответствует x-выбранный, то обращаем в нуль соответствующий элемент $\bar{c}_{ij}$ в таблице, но так, чтобы остальные x-выбранные элементы, соответствующие новому выбору, оставались равными нулю. Для этого переходим к шагу 2.

Наиболее трудоемки в смысле выполняемого числа операций шаги 2б, 2в, 3. Для их выполнения требуется соответственно $2mn$, mn, $2mn$ операций. Для выполнения шагов 2а, 4, 5 требуется не более $2(m+n)$ операций на каждый шаг. Шаг 1 выполняется в ходе решения задачи однократно. Следовательно, время при решении задачи в основном затрачивается на выполнение шагов 2б, 2в, 3. При достаточно больших значениях m и n можно ограничиться параллельным счетом только при выполнении этих шагов. Другими словами, при решении транспортной задачи шаги 1, 2а, 4, 5 выполняются на одной из машин вычислительной системы, а шаги 2б, 2в, 3 реализуются при одновременной работе всех машин ВС.

Операции шага 2б сводятся к операциям сложения по формуле (8.73). Так как общее число чисел c_{ij} равно mn, то за один такт можно выполнять mn сложений, т. е. весь шаг вычислений можно реализовать за два такта. При этом потребуется mn машин. При числе машин в ВС, равном l, где $l = m/k$, общее число тактов на выполнение операции шага 2б составит $2kn$.

Операции шага 2в сводятся к операциям сравнения с нулем. Здесь также в принципе допускается одновременное выполнение операций над всеми числами c_{ij}, т. е. mn операций могут быть выполнены за один такт. В случае $l=m/k$ потребуется kn тактов.

Операции шага 3 состоят в отыскании наименьшего числа c_{ij} среди mn чисел матрицы. Сперва в каждой из l машин выполняются mn/l сравнений, затем для сравнения результатов во всех машинах потребуется еще l операций сравнения. Если одновременно вести попарные сравнения, то это число может быть уменьшено до s, равного округленному до большего целого $\log_2 l$. Таким образом, общее число операций на выполнение шага 3 составит $mn/l + s$. В случае необходимости шаги 1, 2а, 4, 5 могут быть также ускорены путем параллельного выполнения

операций, однако это ускорение не будет эффективным. При достаточно больших значениях m и n по сравнению с l доля затрат времени на выполнение шагов 1, 2а, 4, 5 составит

$$\frac{a_1(m+n)}{a_2\frac{mn}{l}} \approx \frac{l}{m} + \frac{l}{n},$$

где a_1 и a_2 — суммарный вес операции при выполнении шагов 1, 2а, 4, 5 и 2б, 2в, 3 соответственно. Если $l = 0{,}01m$, то затраты на эти шаги составят около 1%.

Операторная схема p-алгоритма может быть построена по аналогии с предыдущими задачами, и ее приводить не будем.

Таким образом, и для транспортной задачи применение УВС из l ЭМ дает выигрыш во времени по сравнению с одной ЭМ примерно в l раз.

Итак, экстремальные задачи линейного программирования могут эффективно решаться на УВС. При этом выигрыш во времени в широких пределах пропорционален числу элементарных машин в системе.

8.3. Обыкновенные дифференциальные уравнения

Обыкновенные дифференциальные уравнения встречаются довольно часто в различных прикладных задачах. Они играют важную роль в теории колебаний, теории устойчивости, в исследовании переходных процессов и т. д.

Решение систем обыкновенных дифференциальных уравнений при достаточно высоком порядке системы и сложных нелинейных зависимостях требует, как правило, выполнения большого числа операций. В данном параграфе будет рассмотрено решение на УВС задачи Коши для систем обыкновенных дифференциальных уравнений и граничных задач для систем линейных и нелинейных дифференциальных уравнений [13, 14].

Решение задачи Коши для системы дифференциальных уравнений методом Рунге-Кутта. Система дифференциальных уравнений может быть приведена к нормальной форме Коши:

$$\begin{aligned} y_1' &= f_1(t, y_1, y_2, \ldots, y_m); \\ y_2' &= f_2(t, y_1, y_2, \ldots, y_m); \\ &\ldots\ldots\ldots\ldots \\ y_m' &= f_m(t, y_1, y_2, \ldots, y_m) \end{aligned} \qquad (8.74)$$

при начальных условиях

$$t = t_0,\ y_1 = y_{10},\ldots,\ y_m = y_{m0},$$

где $f_1, f_2,\ldots, f_m$ — известные в общем случае нелинейные функции $t, y_1,\ldots, y_m$. Если t рассматривать как зависимую переменную $y_{m+1} = y_n = t$ и добавить к системе уравнений (8.74) уравнение $y_n' = 1$, то система примет вид:

$$\begin{aligned} y_1' &= f_1(y_1,\ldots, y_n);\\ y_2' &= f_2(y_1,\ldots, y_n);\\ &\cdots\cdots\cdots\\ y_n' &= f_n(y_1,\ldots, y_n), \end{aligned} \qquad (8.75)$$

где $n = m + 1$, $y_n = t$, $f_n(y_1,\ldots, y_n) = 1$.

Согласно методу Рунге-Кутта, решение системы дифференциальных уравнений (8.75) выполняется последовательными шагами. Новые значения функций на $(s+1)$-м шаге при $t = t_{s+1}$ вычисляются по следующим формулам:

$$k_{1i}^s = hf_i(y_1^s, y_2^s, \ldots, y_n^s); \qquad (8.76)$$

$$k_{ji}^s = hf_i(y_1^s + k_{j-1,1}^s,\ y_2^s + k_{j-1,2}^s,\ \ldots,\ y_n^s + k_{j-1,n}^s); \qquad (8.77)$$

$$j = 2, 3, 4;\ i = 1, 2,\ldots, n,$$

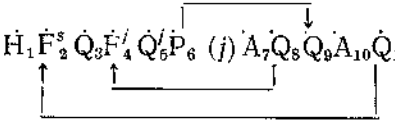
где h — константа ($^1/_2$ длины шага интегрирования);

$$\Delta y_i^s = \frac{1}{3}(\kappa_{1i}^s + 2\kappa_{2i}^s + 2\kappa_{3i}^s + \kappa_{4i}^s)\quad i = 1, 2, \ldots, n; \qquad (8.78)$$

$$y_i^{s+1} = y_i^s + \Delta y_i^s. \qquad (8.79)$$

Нетрудно видеть, что каждая последующая формула использует результаты предыдущей и счет по каждой из них ведется n раз. Поэтому данную задачу естественно решать на УВС из n ЭМ, причем в каждой машине m_i вычислять и хранить переменные, соответствующие значению параметра i.

Схема p-алгоритма в этом случае может быть записана в виде следующей строчки:



$$\dot{H}_1\dot{F}_2^s\dot{Q}_3\dot{F}_4^j\dot{Q}_5^j\dot{P}_6\ (j)\ \dot{A}_7\dot{Q}_8\dot{Q}_9\dot{A}_{10}\dot{Q}_{11}\ (s)\ \dot{Я}_{12}, \qquad (8.80)$$

где $\dot{H}_1$ — p-оператор настройки, соединяющий входы и выходы всех ЭМ с общим каналом связи;

$\dot{F}_2^s$ — p-оператор, формирующий переход к новому шагу;

$\dot{Q}_3$ — обобщенный p-оператор обмена, передающий коды y_i^s из ЭМ m_i во все ЭМ ($i = 1, 2,\ldots, n$);

$\dot{F}_4^j$ — p-оператор, формирующий переход к вычислению следующих k_{ji}^s;

$\dot{Q}_5^j$ — обобщенный p-оператор, вычисляющий функции $hf_i(z_{j1}^s, z_{j_2}^s, \ldots, z_{jn}^s)$;

$\dot{P}_6(j)$ — p-оператор обобщенного условного перехода. Если все k_{ji}^s вычислены, то он осуществляет переход к p-оператору $\dot{Q}_9$, если не все, то — к p-оператору $\dot{A}_7$;

$\dot{A}_7$ — арифметический p-оператор, вычисляющий в каждой ЭМ m_i суммы $z_{ji}^s = y_i^s + k_{j-1,i}^s$;

$\dot{Q}_8$ — обобщенный p-оператор обмена, пересылающий переменные z_{ji}^s из ЭМ m_i во все ЭМ. Он выполняется за n тактов и может совмещаться с оператором $\dot{Q}_5$;

$\dot{Q}_9$ — обобщенный p-оператор, вычисляющий во всех ЭМ $\Delta_i^s y$;

$\dot{A}_{10}$ — арифметический p-оператор, вычисляющий во всех ЭМ y_i^{s+1};

$\dot{Q}_{11}(s)$ — обобщенный p-оператор, устанавливающий конец вычислений;

$\dot{Я}_{12}$ — p-оператор конца.

Как видно из схемы p-алгоритма, процесс вычислений на УВС состоит из параллельного счета всех машин и обмена информацией между ними. По сравнению с обычной ЭВМ получается выигрыш во времени решения примерно в n раз. Потери времени на обмен информацией между ЭМ малы.

Решение граничных задач для системы линейных дифференциальных уравнений методом сопряженных уравнений. Пусть имеется система n линейных дифференциальных уравнений пер-

вого порядка

$$Y'(t) = A(t)Y(t) + F(t), \tag{8.81}$$

где $Y'(t)$ — матрица-столбец $\begin{bmatrix} y_1'(t) \\ \cdot \\ \cdot \\ \cdot \\ y_n'(t) \end{bmatrix}$;

$Y(t)$ — матрица-столбец из неизвестных $\begin{bmatrix} y_1(t) \\ \cdot \\ \cdot \\ \cdot \\ y_n(t) \end{bmatrix}$;

$A(t)$ — квадратная матрица $(n \times n)$ из коэффициентов, которые могут быть функциями независимой переменной t, но не зависят от y_i;

$F(t)$ — матрица-столбец свободных членов $\begin{bmatrix} f_1(t) \\ \cdot \\ \cdot \\ \cdot \\ f_n(t) \end{bmatrix}$.

Предположим, что первые r из y_i заданы при $t = 0$ и что $n - r$ из y_i заданы при $t = T$. Обозначим их y_{i_k} (T), где $k = 1, 2, \ldots, n - r$. Так как условия заданы для различных значений t, то граничная задача не может быть непосредственно решена методом Рунге-Кутта. Поэтому ее сперва сводят к задаче Коши. Применим для этого метод сопряженных уравнений, идея которого заключается в следующем.

Сначала определяется система дифференциальных уравнений, сопряженная с системой (8.81),

$$-X'(t) = \overline{A}(t)X(t), \tag{8.82}$$

где $\overline{A}(t)$ — транспонированная матрица $A(t)$;

$X(t)$, $X'(t)$ — новые матрицы-столбцы из неизвестных $\begin{bmatrix} x_1(t) \\ \cdot \\ \cdot \\ \cdot \\ x_n(t) \end{bmatrix}$

и их производных $\begin{bmatrix} x_1'(t) \\ \cdot \\ \cdot \\ \cdot \\ x_n'(t) \end{bmatrix}$.

Можно показать, что

$$\frac{d}{dt}(x_i y_i) = x_i f_i. \tag{8.83}$$

Интегрируя (8.83) от 0 до T, получаем

$$x_i(T)\,y_i(T) - x_i(0)\,y_i(0) = \int_0^T x_i(t)\,f_i(t)\,dt. \tag{8.84}$$

Чтобы свести задачу к начальной, надо при $t = 0$ найти значения переменных $y_{r+1}(0), \ldots, y_n(0)$. Для их вычисления нужны $n - r$ уравнений (8.84). Сопряженная система уравнений (8.82) интегрируется $n - r$ раз от T до 0. При этом используются специально подобранные значения $x_i(T)$ такие, что при m-м интегрировании $(1 \leqslant m \leqslant n - r)$

$$x_i(T) = 0 \text{ для } i \neq i_m; \tag{8.85}$$

$$x_{i_m}(T) = 1,$$

где $x_{i_m}(T)$ — коэффициент в формуле (8.84) при одном из $n - r$ заданных $y_{i_m}(T)$.

В результате каждого интегрирования системы (8.82) получаем значения соответствующих ${}_m x_i(t)$, где параметр m указывает номер интегрирования. Это может быть записано в виде матрицы-столбца ${}_m X(t)$.

Из условий (8.85) вытекает, что все $x_i(T)$ равны нулю, кроме одного $x_{i_m}(T)$, которое для удобства выбрано равным единице. Уравнение (8.84) принимает вид

$${}_m x_i(0)\,y_i(0) - y_{i_m}(T) = \int_T^0 {}_m x_i(t)\,f_i(t)\,dt. \tag{8.86}$$

Полагая m равным последовательно $1, 2, \ldots, n - r$, получаем систему $n - r$ линейных алгебраических уравнений для неизвестных $y_{r+1}(0), \ldots, y_n(0)$. После решения этой системы становятся известными значения всех переменных при $t = 0$ и исходная задача может рассматриваться как начальная.

Решение этой задачи на вычислительной системе из n машин можно представить следующим образом.

Этап 1. Вычисление $y_{r+1}(0), \ldots, y_n(0)$.

Шаг 1. Интегрирование сопряженной системы (8.82) $n - r$ раз от T до 0 при условиях (8.85).

В общем виде, если коэффициенты матрицы $\bar{A}(t)$ являются функциями времени t, заданными таблично, решение сопряженной системы сводится к решению системы конечно-разностных уравнений при данных начальных условиях. Для общности будем предполагать также, что коэффициенты матрицы-столбца $F(t)$ также заданы таблично.

Разместим в каждой машине m_i i-ю строку матрицы $\bar{A}(t)$, а также $x_i(t)$ и $f_i(t)$ и пусть она вычисляет $x_i(t)$ $(i = 1, 2, \ldots, n)$ при m-м интегрировании (8.82). Система дифференциальных уравнений (8.82) при начальных значениях (8.85) может быть преобразована по следующей схеме. На $(k+1)$-м шаге при $t = t_{k+1}$ сначала вычисляются приращения

$$\Delta x_i^{(k)} = (a_{i1}^{(k)} x_1^{(k)} + a_{i2}^{(k)} x_2^{(k)} + \cdots + a_{in}^{(k)} x_n^{(k)})\,\Delta t \quad i = 1, 2, \ldots, n, \quad (8.87)$$

где $a_{i1}^{(k)}$ — элементы i-й строки матрицы $\bar{A}$ при $t = t_k$.

Затем определяются значения переменных для следующего приближения

$$x_i^{(k+1)} = x_i^{(k)} + \Delta x_i^{(k)}. \quad (8.88)$$

Первый шаг повторяется $n - r$ раз. Вычисленные значения $x_{i_m}^{(k)}$ запоминаются в машинах с номерами i_m.

Шаг 2. Вычисление правых частей выражения (8.86) по формуле $b_{i_m} = \int_T^0 x_i(t) f_i(t) dt$. В машинах с номерами, соответствующими индексам i_m $(1 \leqslant m \leqslant n - r)$, одновременно вычисляются интегралы по формулам численного интегрирования.

Шаг 3. Определение $y_{r+1}(0), \ldots, y_n(0)$. В машинах m_{i_m} $(1 \leqslant m \leqslant n - r)$ определяются $y_{r+1}(0), \ldots, y_n(0)$ по формуле

$$_m x_i(0)\, y_i(0) - y_{i_m}(T) = b_{i_m}. \quad (8.89)$$

Этап II. Решение системы уравнений (8.81), например, методом Рунге-Кутта.

Основное время решения данной задачи приходится на первый шаг этапа I и этап II. При их выполнении коэффициент эффективности системы δ близок к единице. На остальных шагах этапа I использование ЭМ несколько хуже, однако на них приходятся ничтожные затраты в общем балансе времени. Кроме того, существуют дополнительные возможности распараллеливания вычислений на этих шагах. Поэтому можно считать, что УВС из n машин дает выигрыш во времени по сравнению с одной ЭМ примерно в n раз.

Решение граничных задач для нелинейных систем. Общая нелинейная система имеет вид

$$y_i'(t) = g_i(y_1, \ldots, y_n, t);\ i = 1, \ldots, n. \quad (8.90)$$

Предполагается, что все функции g_i дифференцируемы по каждому из y_i и что граничные условия определяют

$$y_1(0), \ldots, y_r(0) \text{ и } y_{i_k}(T), \text{ где } k = 1, \ldots, n - r.$$

Применяемый для нелинейных уравнений метод состоит в нахождении приближенных значений $y_{r+1}(0), \ldots, y_n(0)$. Зададимся исходными значениями $y_{r+1}^*(0), \ldots, y_n^*(0)$. После этого задача может быть решена как начальная для системы уравнений (8.90). Компоненты решения, получаемые с помощью приближенных начальных условий, обозначим $y_i^*(t)$. Если вычисленные таким образом значения $y_{i_k}(T)$ не совпадают с заданными, то сделаем такие поправки начальных приближенных значений, чтобы все величины $|y_{i_k}(T) - y_{i_k}^*(T)|$ стали возможно меньше. Определим $\delta y_i(t)$ посредством равенства

$$\delta y_i(t) = y_i(t) - y_i^*(t);\ i = 1, 2, \ldots, n. \quad (8.91)$$

Если подставить найденные из этих равенств $y_i(t)$ в систему уравнений (8.90), учесть, что $y_i^*(t)$ есть решение этой же системы при некоторых граничных условиях и пренебречь членами порядка выше первого, то получим

$$\delta y_i'(t) = \left(\frac{\partial g_i}{\partial y_j}\right) \delta_{y_i}, \quad (8.92)$$

где

$$\delta y_i'(t) = \frac{d}{dt}[\delta y_i(t)],\ i = 1, 2, \ldots, n.$$

Уравнения (8.92) образуют однородную систему линейных дифференциальных уравнений, сопряженная система для которой имеет вид

$$-x_i' = \left(\frac{\partial g_j}{\partial y_i}\right) x_j,\ i = 1, 2, \ldots, n. \quad (8.93)$$

При этом формула (8.84) имеет вид

$$x_i(T)\delta y_i(T) - x_i(0)\delta y_i(0) = 0. \quad (8.94)$$

В этих уравнениях $\delta y_i(0) \equiv 0$ для $i = 1,\ldots, r$, а остальные δy_i представляют собой $n - r$ неизвестных величин. Чтобы получить $n - r$ уравнений для этих неизвестных, необходимо решить для уравнений (8.93) $n - r$ начальных задач. Решение m-й задачи есть вектор-столбец ${}_mX(t)$ с компонентами ${}_mx_i(t)$. При отыскании m-го решения все значения $x_i(T)$ принимаем равными нулю, кроме коэффициентов при $y_{i_m}(T)$, который выбираем равным единице.

Таким образом, уравнения (8.94) приводятся к виду

$$\delta y_{i\,m}(T) = {}_mx_{r+1}(0)\delta y_{r-i}(0). \qquad (8.95)$$

Решая уравнение (8.95), находим поправки $\delta y_{r+1}(0)$, которые позволяют вычислить уточненные приближения. Процесс повторяем до тех пор, пока начальные условия не обеспечат достаточно малых значений погрешностей

$$|y_{i_k}(T) - \overset{*}{y}_{i_k}(T)| < \varepsilon. \qquad (8.96)$$

Решение этой задачи на вычислительной системе из n машин будем вести следующим образом.

Этап I. По полученным приближенным значениям $\overset{*}{y}_{r+1}(0),\ldots, \overset{*}{y}_n(0)$ вычисляем $\overset{*}{y}_i(t)$.

Шаг 1. Решаем задачу Коши для системы из n дифференциальных уравнений методом Рунге-Кутта, как это было описано выше.

Шаг 2. По найденным значениям проверяем условия (8.96). Если для всех $k = 1,\ldots, n - r$ оно выполнено, то вычисления окончены, в противном случае переходим на этап II.

Этап II. Вычисление поправок δy_{i_m} $m = 1,\ldots, n - r$.

Шаг 1. Решается $n - r$ начальных задач для сопряженной системы однородных линейных уравнений (8.93), например, описанным перед этим методом.

Шаг 2. По найденным значениям x_j из уравнения (8.95) определяем $\delta y_{r+i}(0)$; $i = 1,\ldots, n - r$. В каждой из $n - r$ вычислительных машин находим $\delta y_{r+i}(0)$.

Шаг 3. По приближенным значениям $\overset{*}{y}_{r+i}(0)$ и $\delta y_{r+i}(0)$ вычисляем в $n - r$ машинах по формулам $y_{r+i}(0) = \overset{*}{y}_{r+i}(0) + \delta y_{r+i}(0)$ уточненные приближения $y_{r+i}(0)$. После чего переходим к этапу I.

В данной задаче наиболее трудоемкие вычисления (на этапе I и шаге 1 этапа II) могут одновременно выполняться на n ЭМ при ничтожном простое ЭМ. На шагах 2 и 3 этапа II $n - r$ ЭМ используются полностью. Для загрузки остальных ЭМ надо применить более сложную схему распараллеливания. Но даже их простой мало повлияет на значение коэффициента эффективности из-за небольшого числа вычислений на этих шагах.

Таким образом, в этой задаче как и в других, связанных с решением систем обыкновенных дифференциальных уравнений, выигрыш во времени при применении УВС из n ЭМ будет примерно равен n по сравнению с одной ЭМ (без учета различия в объеме оперативной памяти).

8.4. Дифференциальные уравнения в частных производных

Задачи данного типа охватывают широкий круг разнообразных научных и инженерных проблем. Рациональное проектирование сооружений и машин, рациональная разработка многих природных полезных ископаемых, а также многие другие проблемы требуют решения дифференциальных уравнений в частных производных с различными краевыми условиями. Приведем примеры таких задач. При проектировании тепловых устройств нужно знать распределение температуры в различных частях этих устройств и объектах обогрева при стационарных и нестационарных режимах. Проектирование строительных сооружений требует знания возникающих в них нормальных и касательных напряжений, деформаций, осадок. Для проектирования турбомашин требуется знание законов распределения давлений и скоростей в проточной части и напряжений в рабочих лопатках. При проектировании кораблей и самолетов требуется знать распределение скоростей среды. При проектировании рациональной системы разработки нефтяных месторождений, схем подземной газификации, организации поисковых работ по рудным ископаемым нужно знать распределение пластовых давлений, дебит жидкости или газа, распределение потенциалов. Составление прогноза погоды требует знания распределения температур, давлений, скоростей в большом числе точек. К решению уравнений в частных производных сводится определение траекторий электрона в электронном микроскопе.

В качестве примеров рассмотрим решение на УВС линейных дифференциальных уравнений в частных производных второго порядка с двумя независимыми переменными. Общее уравнение такого рода имеет вид

$$H\frac{\partial^2 u}{\partial x^2} + 2K\frac{\partial^2 u}{\partial x\,\partial y} + L\frac{\partial^2 u}{\partial y^2} + M = 0, \qquad (8.97)$$

где H, K, L — функции x, y, u; M — функция x, y, u, $\frac{\partial u}{\partial x}$, $\frac{\partial u}{\partial y}$.

Различают три типа таких уравнений: эллиптическое ($K^2 - HL < 0$), параболическое ($K^2 = HL$) и гиперболическое ($K^2 - HL > 0$). При решении на УВС уравнений всех трех типов, ограничимся такими случаями, когда тип уравнения не изменяется при переходе от одной точки области к другой.

Решение задачи Дирихле для эллиптических дифференциальных уравнений итеративным методом (см., например, [13, 14]).

Рассмотрим в качестве примера решение уравнения Пуассона

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = g(x, y) \qquad (8.98)$$

для прямоугольной области $0 \leqslant x \leqslant a$, $0 \leqslant y \leqslant b$. На границе заданы значения функции u. Разобъем область на квадраты с помощью сетки с шагом h. Обозначим через u^s_{jk} и u^{s+1}_{jk} значения функций в узловой внутренней точке (j, k) на s-м и $(s+1)$-м последовательных приближениях. Для вычисления $u^{s+1}_{j,k}$ воспользуемся формулой

$$u^{s+1}_{jk} = \frac{1}{4}(u^s_{j-1,k} + u^s_{j+1,k} + u^s_{j,k-1} + u^s_{j,k+1} - h^2 g_{jk}). \qquad (8.99)$$

Процесс решения данным методом весьма прост. Выбираем исходное приближение. Последовательные приближения вычисляем по формуле (8.99). Затем для всех точек сравниваем новое и предыдущее значения функции.

Если хотя бы в одной точке не удовлетворяется условие

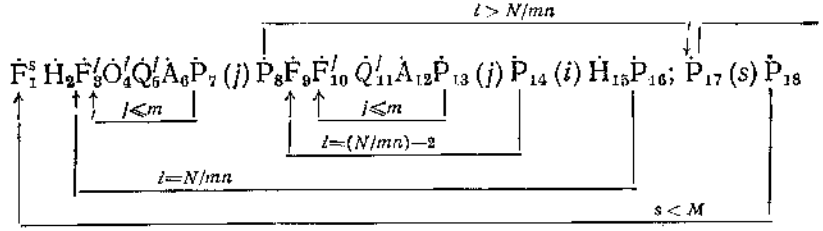
$$|u^{s+1}_{jk} - u^s_{jk}| \leqslant \varepsilon, \qquad (8.100)$$

то делаем следующую итерацию и т. д., пока не будет удовлетворено условие (8.100), либо число итераций не превысит заданное.

Вычисления по формулам (8.99) и (8.100) можно выполнять сразу для всех N точек области. Поэтому базовая точка на характеристической функции p-алгоритма данной задачи будет соответствовать количеству ЭМ, равному N, которое обычно велико, т. е. данная задача практически не накладывает ограничений на число ЭМ, используемых в системе.

Пусть число ЭМ $n < N$, тогда имеет смысл разбить область на n полос (вертикальных и горизонтальных) и значения функций в узловых точках каждой i-й полосы вычислять на ЭМ m_i $(i = 1, 2, \ldots, n)$. При этом в памяти каждой ЭМ m_i будут храниться значения функции для N/n узловых точек, т. е. данные, необходимые для вычисления функции во всех внутренних точках, будут находиться в памяти той же ЭМ. Для точек, расположенных на границе с соседней полосой, один из аргументов будет находиться в соседней ЭМ. Пусть в каждой полосе будет $2m$ пограничных точек, кроме 1-й и n-й, где их m, тогда каждая ЭМ должна на данном шаге итерации обменяться со своими соседями $2m$ кодами. Чтобы крайние ЭМ не составляли исключения из этого правила и имели ту же программу работы, можно правые граничные точки области поместить в ЭМ m_1, а левые — в ЭМ m_n. Будем вычислять значения функций в каждой полосе по колонкам: сначала 1-й, состоящей из пограничных точек, затем $(N/nm) - 2$ колонок с внутренними точками, и, наконец, (N/nm)-ю колонку, состоящую из пограничных точек.

Схема p-алгоритма может быть записана в виде, где все p-операторы состоят из n одинаковых составляющих:



$$\dot{Я}_{19}\dot{Я}_{20}, \qquad (8.101)$$

где $\dot{F}^s_1$ — p-оператор, формирующий переход к новой итерации;

$\dot{H}_2$, $\dot{H}_{15}$ — p-операторы настройки, соединяющие вход ЭМ m_i с выходом ЭМ m_{i-1} и m_{i+1} соответственно $(i = 1, 2, \ldots, n)$;

$\dot{F}^j_3$, $\dot{F}^j_{10}$ — p-операторы, формирующие переход от j-й точки колонки к $(j+1)$-й;

$\dot{O}^j_4$ — p-оператор обмена информацией, посылающий код в канал связи и принимающий код из канала связи;

$\dot{Q}^j_5$, $\dot{Q}^j_{11}$ — обобщенные арифметические p-операторы, ведущие счет по формуле (8.99);

$\dot{A}_6$, $\dot{A}_{12}$ — арифметические p-операторы, ведущие счет по формуле (8.100);

$\dot{P}_7(j)\,\dot{P}_{13}(j)$ — p-операторы обобщенного условного перехода, определяющие цикл счета точек в одной колонке;

$\dot{P}_8$ — p-оператор обобщенного условного перехода, определяющий, все ли колонки просчитаны;

$\dot{F}_9$ — p-оператор, формирующий переход от одной колонки к другой;

$\dot{P}_{14}(i)$ — p-оператор обобщенного условного перехода, определяющий цикл счета колонок;

$\dot{P}_{16}$ — p-оператор обобщенного безусловного перехода;

$\dot{P}_{17}(s)$ — p-оператор обобщенного условного перехода, определяющий, удовлетворено ли неравенство (8.100) во всех ЭМ, и в зависимости от этого передающий управление p-оператору $\dot{Я}_{20}$ либо $\dot{P}_{18}$;

$\dot{P}_{18}$ — p-оператор обобщенного условного перехода, проверяющий, не привысило ли число итераций заданного числа M;

$\dot{Я}_{19}$ и $\dot{Я}_{20}$ — p-операторы конца работы. Первому соответствует прекращение вычислений из-за большого числа итераций, второму — из-за получения решения.

Решение краевой задачи для параболических уравнений. Рассмотрим решение типичного параболического уравнения — уравнения теплопроводности

$$\frac{\partial^2 u}{\partial x^2} = c\frac{\partial u}{\partial t}, \qquad (8.102)$$

удовлетворяющее условиям: u принимает заданные значения при $t = 0$ для всех x на отрезке $0 \leqslant x \leqslant L$ и при любых t в точках $x = 0$ и $x = L$.

Рассмотрим метод решения уравнений (8.102), состоящий в замене производной второго порядка конечно разностным приближением (см., например, [13, 14]).

Обозначим $u(jh, t)$ через $u_j(t)$, где $j = 0, 1, 2, \ldots, N + 1$; $h = L/(N + 1)$ — длина шага в направлении оси x; $u_j(t)$ — функция только одной переменной t, функции $u_0(t)$ и $u_{N+1}(t)$ известны из граничных условий.

Уравнение (8.102) можно приближенно заменить уравнениями

$$\frac{du_j}{dt} = \frac{1}{ch^2}(u_{j+1} + 2u_j + u_{j-1}), \quad j = 1, 2, 3, \ldots, N, \qquad (8.103)$$

где u_j — функции только одной переменной t. Уравнения (8.103) образуют систему дифференциальных уравнений первого порядка с заданными значениями неизвестных при $t = 0$. Решая эту систему, например, методом Рунге-Кутта, как было описано в 8.3, можно определить все N неизвестных функций $u_j(t)$.

Данная система проще приведенной в 8.3, так как производная u_j' зависит только от трех переменных, причем одна переменная имеет тот же индекс j, а две другие — индекс, отличающийся на единицу. Это существенно упрощает вычисление правых частей, а также снижает требования к обмену информацией между ЭМ.

Как и в предыдущем случае, процесс решения данной задачи может быть представлен в виде N одинаковых (или почти одинаковых) параллельных ветвей вычислений. Наиболее просто процесс решения будет выглядеть на УВС из N машин. Если число ЭМ $n < N$, то область, как и в предыдущем случае, можно разделить на n частей и для пограничных точек осуществлять обмен кодами между соседними ЭМ.

Таким образом, и при решении дифференциальных уравнений параболического типа применение УВС дает выигрыш во времени решения, пропорциональный числу ЭМ.

Решение задачи Коши для линейных дифференциальных уравнений гиперболического типа. В качестве примера рассмотрим решение на УВС задачи Коши методом сеток (см. [13, 14]) для простейшего дифференциального уравнения гиперболического типа:

$$\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = 0 \qquad (8.104)$$

с начальными условиями

$$u\Big|_{y=0} = \varphi(x), \quad \frac{\partial u}{\partial y}\Big|_{y=0} = \psi(x) \ (a \leqslant x \leqslant b). \qquad (8.105)$$

Будем использовать прямоугольную сетку с шагами h по x и k по y

$$\alpha = k/h \geqslant 1.$$

Разностное уравнение имеет вид:

$$u_{i,\,j+1} = 2\,u_{ij} - u_{ij-1} + \alpha^2\,(u_{i+1,j} - 2u_{ij} + u_{i-1,\,j}). \qquad (8.106)$$

Зная решения в узлах первых двух горизонтальных рядов (определяемые начальными условиями), будем последовательно вычислять значения решения на втором, третьем и т. д. рядах (рис. 89).

В каждом последующем ряду число вычисляемых узлов уменьшается на 2. При числе машин в УВС, равном $n < N$,

отрезок $[a, b]$ можно разбить на n частей (для простоты примем, что $l = N/n$ — целое). При этом в ЭМ m_1 будут находиться узлы $1,2,\ldots, l$ и $1^{(1)}, 2^{(1)}, \ldots, (l-1)^{(1)}$, в ЭМ $m_2 — l+1, l+2,\ldots, 2l$ и $l^{(1)}, (l+1)^{(1)}, \ldots, (2l-1)^{(1)}$ и т. д., и, наконец, в ЭМ m_n — $N-l+1,\ldots, N$ и $(N-l-1)^{(1)},\ldots, (N-2)^{(1)}$.

В каждой ЭМ по формуле (8.106) вычисляются узлы последующего ряда. При вычислении узлов на границе между ЭМ производится двусторонний обмен информацией между соседними ЭМ.

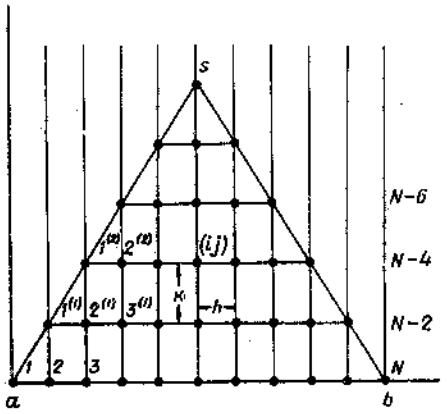


Рис. 89. Вид области при решении задачи Коши для гиперболического дифференциального уравнения

Число узлов в каждом последующем ряду уменьшается на 2. В связи с этим загрузка ЭМ становится неравномерной и коэффициент эффективности $\delta \approx 0{,}5$. Величина δ может быть увеличена путем перераспределения узлов между ЭМ системы. Для этого перед вычислением 3-го ряда узлов из ЭМ m_2 в m_1 пересылаются узлы $l+1$, $l^{(1)}$, а из ЭМ m_{n-1} в m_n — узлы $N-l$, $(N-l-2)^{(1)}$. Перед вычислением 4-го ряда соответствующие пары узлов пересылаются из ЭМ m_3 в m_2, из m_2 — в m_1, из m_{n-2} — в m_{n-1}, из m_{n-1} — в m_n и т. д. Указанный процесс заканчивается, когда пересылка узлов сделана из ЭМ $m_{n/2}$ в $m_{n/2+1}$ при четном числе ЭМ в УВС и из ЭМ $m_{(n+1)/2}$ в $m_{(n+1)/2+1}$ — при нечетном. Вблизи точки s (см. рис. 89) вычисление можно продолжать на нескольких или даже на одной ЭМ.

Из примера видно, что при $l \gg n$ простаивание ЭМ может быть сделано незначительным, а выигрыш во времени пропорционален числу ЭМ в системе.

Рассмотренные примеры дифференциальных уравнений в частных производных эллиптического, параболического и гиперболического типов и методы их решения типичны. Поэтому решение подобных задач на УВС по сравнению с одной ЭВМ, эквивалентной по параметрам ЭМ системы, дает выигрыш во времени, пропорциональный числу ЭМ в системе при изменении числа ЭМ в широких пределах.

8.5. Численное интегрирование и дифференцирование

Рассмотрим решение на УВС весьма распространенных задач математического анализа — интегрирования и дифференцирования функций.

Численное интегрирование. По определению интеграла

$$\int_c^d f(x)\,dx = \lim_{\substack{n\to\infty \\ \Delta x_i \to 0}} \left(\sum_{i=1}^{n-1} f(x_i)\,\Delta x_i \right), \qquad (8.107)$$

откуда следует, что с помощью интегральной суммы $S_n = \sum_{i=1}^{n} f(x_i)\,\Delta x_i$ можно найти интеграл с любой степенью точности, хотя сходимость S_n к $\int_c^d f(x)dx$ может быть очень медленной.

Для получения высокой точности приходится брать n весьма больших порядков, особенно для сложных функций.

При численном интегрировании $f(x)$ заменяется интерполяционной функцией $\varphi(x)$.

Рассмотрим, например, вычисление интегралов $\int_c^d f(x)dx$ по формулам Ньютона-Котеса (см., например, [14]), которые получаются путем замены $f(x)$ интерполяционным полиномом Лагранжа с узлами, разбивающими интервал интегрирования на равные части длиною h.

Для случая, когда узлы интегрирования содержат точки c и d (формулы замкнутого типа),

$$x_i = a + ih, \quad (i = 1, 2, \ldots, n); \qquad (8.108)$$
$$c = a + h, \quad d = a + nh.$$

Обозначим $F(y) = f(a + hy)$.

Тогда

$$\int_c^d f(x)\,dx = h\int_1^n F(y)\,dy = (d-c)\sum_{i=1}^{n} I_i^{(n)} f(a+ih) +$$
$$+ h\int_1^n (y-1)(y-2)\cdots(y-n)\cdot F(y, 1, 2, \ldots, n)dy, \qquad (8.109)$$

где

$$I_i^{(n)} = \frac{(-1)^{n-i}}{(n-1)(i-1)!\,(n-i)!}\int_1^n \frac{(y-1)(y-2)\cdots(y-n)}{y-i}\,dy. \qquad (8.110)$$

Значения $I_i^{(n)}$ не зависят от интервала интегрирования и могут быть вычислены раз и навсегда.

Отрезок интегрирования можно разделить на части и для каждой из них применить формулу (8.109), после чего результаты по всем частям суммируются. Этим можно воспользоваться для параллельного выполнения численного интегрирования с помощью УВС. Если число ЭМ в УВС равно l, то отрезок интегрирования может быть разделен на l частей. В этом случае на решение задач потребуется число тактов:

$$\frac{N-l}{l}+s, \tag{8.111}$$

где N — число тактов, требующихся на решение задачи на одной ЭМ;

s — округленный до большого целого $\log_2 l$.

Первое слагаемое соответствует интегрированию каждой из частей, а второе — суммированию результатов этого интегрирования.

Если $N \gg s$, то общий выигрыш во времени при применении системы можно полагать пропорциональным числу машин в системе. В принципе число ЭМ можно увеличивать до n, однако при l близких к N коэффициент эффективности δ может заметно отличаться от оптимального из-за увеличения доли второго слагаемого в (8.111).

То же самое будет и при применении других формул численного интегрирования, например, обобщенной формулы трапеций, обобщенной формулы Стирлинга, (которые получаются из (8.109) при $n=2$, и 3), а также для случая, когда узлы интегрирования не содержат точек c и d (формулы открытого типа). Особенно важную роль УВС могут играть при вычислении кратных интегралов, которое требует выполнения огромного числа операций, часто превышающего возможности современных ЭВМ. Существующие методы вычисления кратных интегралов, например метод кубатурных формул [15], позволяют легко распараллелить процесс вычислений на большое число однородных ветвей.

Численное дифференцирование. Пусть $f(x)$ — функция, заданная таблично либо с помощью сложного аналитического выражения, тогда для нахождения ее производной можно заменить функцию $f(x)$ интерполирующей функцией $\varphi(x)$, производная от которой может быть легко вычислена

$$f(x)=\varphi(x)+R(x), \tag{8.112}$$

где $R(x)$ — остаточный член интерполяционной формулы. В том случае, когда $f(x)$ и $\varphi(x)$ имеют производные k-го порядка, дифференцируя (8.112) k раз, получим

$$f^{(k)}(x)=\varphi^{(k)}(x)+R^{(k)}(x). \tag{8.113}$$

Рассмотрим в качестве примеров численное дифференцирование для неравноотстоящих узлов интерполирования и для равноотстоящих узлов [13]. Для неравноотстоящих узлов воспользуемся интерполяционной формулой Ньютона для неравных промежутков:

$$\begin{aligned} f(x)=f(x_0)+(x-x_0)f(x_0,\ x_1)+(x-x_0)\ (x-x_1)\times \\ \times f(x_0,\ x_1,\ x_2)+\ldots+(x-x_0)\ (x-x_1)\ldots \\ \ldots(x-x_{n-1})f(x_0,\ x_1,\ldots,\ x_n)+(x-x_0)\ (x-x_1)\ldots \\ \ldots(x-x_n)\ f(x,\ x_0,\ x_1,\ldots,\ x_n). \end{aligned} \tag{8.114}$$

Дифференцируя (8.114) k раз ($k<n$), получаем приближенное значение для k-й производной:

$$\begin{aligned} f^k(x)=k!\ [f(x_0,\ x_1,\ \ldots,\ x_k)+(\alpha_0+\alpha_1+\ldots+\alpha_k)\cdot f(x_0,\ x_1,\ldots \\ \ldots,x_{k+1})+(\alpha_0\alpha_1+\alpha_0\alpha_2+\ldots+\alpha_k\alpha_{k+1})\ f(x_0,\ x_1,\ldots,\ x_{k+2})+\ldots \\ \ldots+(\alpha_0\alpha_1\ldots\alpha_{n-k}+\ldots+\alpha_{k+1}\alpha_{k+2}\ldots\alpha_{n-1})\times \\ \times f(x_0,\ x_1,\ldots,\ x_n)], \end{aligned} \tag{8.115}$$

где $\alpha_i=x-x_i$.

Остаточный член

$$R(x)=\sum_{i=0}^{n}\frac{k!}{(k-i)\,(n+i+1)!}f^{(n+i+1)}(\xi_i)\,\omega_n^{(k-i)}, \tag{8.116}$$

где ξ_i — некоторые точки, заключенные в интервале между наибольшим и меньшим из чисел $x,\ x_0,\ x_1,\ldots,\ x_n$.

Для равноотстоящих узлов воспользуемся интерполяционной формулой Ньютона для интерполирования вперед

$$\begin{aligned} f(x)=f(x_0+ht)=f_0+tf_{1/2}^1+\frac{t(t-1)}{2!}f_1^2+ \\ +\frac{t(t-1)(t-2)}{3!}f_{3/2}^3+\ldots+\frac{t(t-1)\ldots(t-n+1)}{n!}f_{n/2}^n, \end{aligned} \tag{8.117}$$

где $t=\frac{x-x_0}{h}$, $h=x_i-x_{i-1}$ $(i=1,\ 2,\ldots,\ n)$, а f с верхним и

нижним индексом — конечные разности:

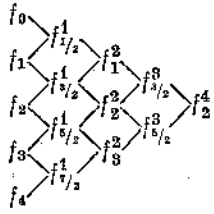
$$f^0_{1/2} = \frac{f_0 + f_1}{2}\,;\ f^1_1 = \frac{f^1_{1/2} + f^1_{3/2}}{2}\,;\ f^2_{3/2} = \frac{f^2_1 + f^2_2}{2}\,;\ \ldots$$

$$f^0_{3/2} = \frac{f_1 + f_2}{2}\,;\ f^1_2 = \frac{f^1_{3/2} + f^1_{5/2}}{2}\,;\ f^2_{5/2} = \frac{f^2_2 + f^2_3}{2}\,;\ \ldots \qquad (8.118)$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$f^0_{i+1/2} = \frac{f^1_i + f^1_{i+1}}{2}\,;\ f^1_i = \frac{f^1_{i-1/2} + f^1_{i+1/2}}{2}\,;\ f^2_{i+1/2} = \frac{f^2_i + f^2_{i+1}}{2}\,;\ \ldots$$

Вычисление конечных разностей можно вести по следующей схеме:



(8.119)

Формула для вычисления k-й производной весьма сложна. Для ее запоминания обычно пользуются операторно-символической формулой

$$\left(h\,\frac{d}{dx}\right)^k f(x_0) = \{\ln(1+\Delta)\}^k f(x_0). \qquad (8.120)$$

Формальное разложение (8.120) $\ln(1+\Delta) = \Delta - \frac{\Delta^2}{2} + \frac{\Delta^3}{3} - \ldots$, доведенное до постоянных разностей, формально возводится в степень как многочлен. Под Δ понимается оператор вычисления разности первого порядка, под Δ^2 — второго и т. д.

Применять вычислительную систему для численного дифференцирования выгодно, если нужно вычислить значение производной во многих точках. Тогда можно использовать число ЭМ, равное числу точек, и вычислять каждую точку на своей ЭМ. Если число точек больше числа ЭМ, то они распределяются равномерно между ЭМ. В этом случае выигрыш во времени будет пропорционален числу ЭМ.

Применение УВС для нахождения значения производной в отдельной точке может быть оправдано для очень сложных функций. В этом случае, по-видимому, трудно добиться полной загрузки всех ЭМ. При вычислениях по формуле (8.117) основное время тратится на отыскание конечных разностей, так как коэффициенты при них могут быть вычислены раз и навсегда. Как видно из (8.119), число разностей i-го порядка равно $n - i + 1$ ($i = 1, 2, \ldots, n$). Таким образом, если при вычислении разностей использовать n машин, то коэффициент эффективности δ будет равен примерно 0,5 и будет улучшаться при уменьшении числа ЭМ.

При использовании формулы (8.115) процесс вычислений также может быть распараллелен за счет одновременного вычисления как функций, так и коэффициентов при них. В этом случае коэффициент эффективности будет не очень высок. Однако, по-видимому, имеются возможности улучшения этого коэффициента путем усложнения программирования.

8.6. Некоторые задачи теории вероятностей и математической статистики

Рассмотрим реализацию на УВС метода статистических испытаний (метод Монте-Карло) и статистических решений.

Метод статистических испытаний [1]. Идея метода статистических испытаний состоит в том, что искомым величинам задачи ставятся в соответствие параметры некоторого случайного процесса, моделируемого на ЭВМ. Данный метод весьма эффективен, а иногда практически единственный для решения некоторых сложных задач. Благодаря своей простоте метод статистических испытаний широко применяется. С его помощью решаются системы линейных уравнений, краевые задачи для эллиптических уравнений, задача Коши с граничными условиями для параболических уравнений, задачи нахождения собственных значений для эллиптических дифференциальных операторов, вычисления многомерных интегралов, взаимодействия частиц с веществом (в частности, расчет атомных реакторов), массового обслуживания, исследования систем управления со случайными воздействиями на входах, моделирования различных самообучающихся систем и многие другие.

Общая математическая схема метода статистических испытаний описывается с помощью цепей Маркова.

Цепью Маркова называется система S с конечным множеством состояний $\{s_1, s_2, \ldots, s_l\}$. В каждый момент $t = 0, 1, 2, \ldots, n$ система S находится в одном определенном состоянии s_i, которому соответствует набор условных вероятностей $p_{i_1}, p_{i_2}, \ldots, p_{il}$, где p_{ij} — вероятность того, что система, находящаяся в момент t в состоянии s_i, в момент $t+1$ перейдет в состояние s_j. Особенность цепей Маркова состоит в том, что вероятность

[1] См., например, [16].

перехода определяется лишь исходным состоянием s_i и не зависит от предыстории системы. Совокупность всех условных вероятностей p_{ij} образует матрицу $P = (p_{ij})$, полностью определяющую свойства данной цепи.

Состояние s_i, для которого $p_{ij} = 1$ при $i = j$, и $p_{ij} = 0$ при $i \neq j$, называется *особым*. Однажды попав в это состояние, система пребывает в нем сколь угодно долго.

Для практических приложений важную роль играют останавливающиеся цепи Маркова, у которых для каждого состояния s_i существует отличная от нуля вероятность перехода в особое состояние. Останавливающиеся цепи Маркова через конечное число шагов с вероятностью единица переходят в особое состояние.

Для большинства задач схема ее решения методом статистических испытаний выглядит следующим образом. Данной задаче ставится в соответствие останавливающаяся цепь Маркова, которая затем моделируется. Моделирование процесса последовательных переходов этой цепи протекает по схеме

$$x \sim s_{i_0} \to s_{i_1} \to s_{i_2} \to \cdots \to s_{i_l}, \qquad (8.121)$$

где s_{i_0} — начальное состояние;

s_{i_l} — одно из особых состояний.

При этом определяется значение некоторой функции $F(x)$, зависящей от последовательности переходов (8.121). Функция $F(x)$ является случайной величиной, математическое ожидание которой требуется найти. После того, как зафиксировано состояние $F(x)$, система S возвращается в состояние s_{i_0} и процесс переходов повторяется. После достаточно большого числа испытаний N получается сумма

$$\frac{1}{N}\sum_{x} F(x) \approx \mathbf{M}F(x), \qquad (8.122)$$

взятая по всем реализованным последовательностям переходов (8.121).

Полное время решения задачи на одной ЭВМ составляет

$$T \approx N(\mathbf{M}\tau)T_0, \qquad (8.123)$$

где $\mathbf{M}\tau$ — математическое ожидание числа переходов в последовательности (8.121);

T_0 — среднее время реализации переходов на ЭВМ.

Рассмотрим в качестве примера решение задачи прохождения однородного потока нейтронов через плоскую пластину H.

Пусть пластина H однородна и не содержит делящихся веществ. Нейтроны при взаимодействии с атомами пластинки могут либо рассеиваться ими, либо поглощаться. Количественно эти процессы принято характеризовать сечениями Σ_s и Σ_c соответственно. Их сумма (при отсутствии деления) равна полному сечению Σ_t. Отношения Σ_s/Σ_t и Σ_c/Σ_t являются вероятностями соответственно рассеяния и поглощения.

Рассмотрим теперь моделирование физических траекторий. Пусть поверхности пластины H, через которую поступают нейтроны, соответствует ордината $z = 0$, а другой поверхности — $z = h$. Состояние нейтрона при данных условиях характеризуется тремя величинами: координатой z, энергией E и направлением полета $\mu = \cos\theta$ (рис. 90). Начальное значение координаты для траекторий всегда постоянно ($z_0 = 0$), а значение энергии E_0 и направление μ_0 зависят от свойств падающего потока. На практике часто приходится иметь дело с четырьмя случаями:

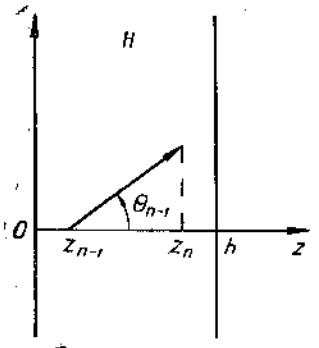


Рис. 90. Прохождение нейтронов через пластину.

а) моноэнергетический поток с заданным значением E_0;

б) поток с заданным энергетическим спектром $n(E)$. Тогда значение E_0 разыгрывается по формуле

$$\int_{E_{\min}}^{E_0} n(E)\,dE = \gamma \int_{E_{\min}}^{E_{\max}} n(E)\,dE, \qquad (8.124)$$

где $0 \leqslant \gamma \leqslant 1$ — случайное число;

в) направленный поток с заданным значением μ_0;

г) пространственный изотропный процесс. Значение μ_0 разыгрывается по формуле $\mu_0 = \gamma$. Дальше траектория рассчитывается так:

1) разыгрываем координату n-го столкновения ($n = 1,2,\ldots$)

$$z_n = z_{n-1} - \frac{\mu_{n-1}\ln\gamma}{\Sigma_t(E_{n-1})}; \qquad (8.125)$$

2) проверяем выполнение условия

$$0 \leqslant z_n \leqslant h. \tag{8.126}$$

Если условие (8.126) не выполнено, то нейтрон покинул пластину. Это фиксируется, и разыгрывается следующая траектория;

3) если $0 \leqslant z_n \leqslant h$, то находим очередное γ, и, если $\gamma \leqslant \Sigma_c/\Sigma_t$, то нейтрон считается поглощенным и его траектория на этом заканчивается, о чем делается запись;

4) если $\gamma > \Sigma_c/\Sigma_t$, то считаем, что нейтрон испытывает рассеяние и нужно разыграть его направление и энергию.

В частности, если рассеяние упругое, то изменение энергии определяется направлением рассеяния. Пусть рассеяние нейтрона происходит с ядром, массовое число которого равно A. Такое рассеяние определяется двумя случайными величинами, в качестве которых обычно берут угол рассеяния θ в системе координат, связанной с центром масс пары нейтрон — ядро (рис. 91), и азимутальный угол рассеяния χ:

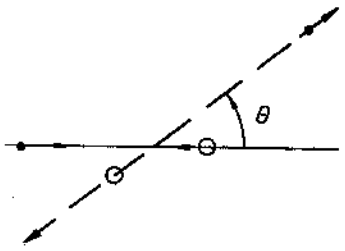


Рис. 91. Рассеяние нейтрона в системе центра масс.

$$0 \leqslant \theta \leqslant \pi;\ 0 \leqslant \chi \leqslant 2\pi.$$

На основании законов сохранения импульса и энергии можно вычислить угол ψ, на который отклонился нейтрон от своего первоначального движения в системе координат, связанной с пластиной,

$$\cos\psi = \frac{A\cos\theta + 1}{\sqrt{A^2 + 2A\cos\theta + 1}}, \tag{8.127}$$

а затем определить энергию, которую он сохраняет, и новое направление полета:

$$E_n = E_{n-1}\frac{A^2 + 2A\cos\theta + 1}{(A+1)^2}; \tag{8.128}$$

$$\mu_n = \mu_{n-1}\cos\psi + \cos\chi\sqrt{(1-\mu_{n-1}^2)(1-\cos^2\psi)}. \tag{8.129}$$

Обычно рассеяние предполагается изотропным в системе центра масс: $\cos\theta$ распределен равномерно в интервале $(-1, +1)$, а угол χ распределен равномерно в интервале $(0, 2\pi)$.

Правило розыгрыша упругого рассеяния, изотропного в системе центра масс, состоит в следующем: находим два случайных числа γ и γ_1 $(0 \leqslant \gamma, \gamma_1 \leqslant 1)$ и полагаем

$$\cos\theta = 2\gamma - 1,\ \chi = 2\pi\gamma_1.$$

Затем по формулам (8.127) — (8.129) находим $\cos\psi$, E_n и μ_n.

Таким образом, нейтрон из состояния $(z_{n-1}, \mu_{n-1}, E_{n-1})$ перешел в состояние (z_n, μ_n, E_n). Расчет траектории продолжаем до тех пор, пока нейтрон не покинет пластинку, либо не поглотится. Затем выбираем (или разыгрываем) начальные значения z_0, μ_0, E_0 для расчета следующей траектории. И так до тех пор, пока не будет сделано обусловленное число испытаний N.

Из рассмотренного примера видны следующие особенности данного вычислительного процесса: траектории могут вычисляться независимо друг от друга; время счета каждой траектории — переменная величина; информация, установленная при каждом испытании, должна суммироваться; в ходе вычислений нужно проверять выполнение условия, определяющего завершение испытаний.

Из этих особенностей вытекает и схема работы УВС. При числе ЭМ, равном n, одновременно вычисляют n траекторий. После окончания вычисления траектории каждая ЭМ посылает в одну из ЭМ информацию, на основе которой устанавливается окончание процесса решения. ЭМ, воспринимающая эту информацию, определяет момент завершения вычислений. Одновременно эта ЭМ может суммировать результаты, полученные всеми ЭМ.

Нетрудно видеть, что при решении задач методом статистических испытаний загрузка ЭМ системы будет полной и выигрыш во времени будет пропорционален числу ЭМ.

Решение задач теории статистических решений[1]. Теория статистических решений представляет большой интерес для многих отраслей науки и техники, так как она позволяет вырабатывать правила поведения в условиях, когда человек или автомат, выбирающий тот или иной образ действия, не располагает полной информацией о всех факторах, учет которых существенно влияет на этот выбор. Задачи такого рода встречаются, например, в радиолокации и радиоастрономии (обнаружение слабых сигналов в шумах), при решении вопросов планирования

[1] См., например, [17].

производства, при оценке результатов экспериментальных исследований и т. п.

Большое значение имеет приложение теории статистических решений к проблеме распознавания образов, которая сводится к отождествлению некоторой реализации (совокупности признаков) образа с эталоном. К проблеме распознавания образов относятся задачи распознавания звуковых, зрительных, радиолокационных и других образов, поиска полезных ископаемых, классификации биологических объектов, диагноза, поиска новых физических явлений, метеорологии, исследования неизвестных письменностей и т. д.

Теория статистических решений тесно связана с теорией игр. Раздел теории игр — «игры против природы» — по существу относится к теории статистических решений, так как в этом случае одному из партнеров игры противостоит некоторая не полностью известная ему обстановка — «состояние природы». Как уже упоминалось в 8.2, многие задачи теории игр (а следовательно, и теории статистических решений) можно успешно решать методами линейного программирования, эффективность реализации которых на УВС была показана выше.

Рассмотрим в качестве примера приложение теории статистических решений к проблеме распознавания образов. На языке теории статистических решений эта проблема носит наименование «множественной классификации, или дискриминантного анализа». Теория множественной классификации рассматривает задачу отнесения данного объекта к одной из нескольких возможных групп, к которым он может быть отнесен на основании множества наблюдений его измеримых характеристик.

С теоретической точки зрения задача дискриминантного анализа представляет собой отыскание решений при фиксированном объеме выборки и конечном числе возможных состояний природы Ω. Эта задача имеет следующие отличительные особенности: а) решение обычно принимается на основании единичного наблюдения нескольких коррелированных характеристик и б) допускается существование априорных вероятностей.

Пусть число возможных состояний природы $\Omega = 1, 2, \ldots, h$, число возможных решений $A = 1, 2, \ldots, h$ и потери равны постоянному числу, например, ω, если состояние природы есть j, а принимается решение, что оно (состояние) есть $i \neq j$. Если $i = j$, то потери равны нулю.

Распределение вероятностей для заданного $j \in \Omega$ будем обозначать через $p_i(z)$, где z — координата точки в пространстве выборок. Пусть любое априорное распределение вероятностей для состояний будет $\xi = \xi(1), \ldots, \xi(h)$.

Тогда для любого решения i апостериорный риск τ_z будет

$$\tau_z(i) = \frac{\omega \sum\limits_{(j \neq i)=1}^{h} p_j(z)\,\xi(j)}{\sum\limits_{j=1}^{h} p_j(z)\,\xi(j)}. \qquad (8.130)$$

Байесово правило заключается в том, что нужно принимать решение i $(i = 1, 2, \ldots, h)$, если

$$\sum_{(j \neq i)=1}^{h} p_j(z)\,\xi(j) \leqslant \sum_{(j \neq k)=1}^{h} p_j(z)\,\xi(j) \qquad (8.131)$$

или, что равнозначно,

$$\frac{p_k(z)}{p_i(z)} = \frac{\xi(i)}{\xi(k)}. \qquad (8.132)$$

При решении этой задачи на УВС из $n \leqslant h$ машин в каждой ЭМ можно одновременно производить вычисления по формулам (8.130) и (8.131) для $r = \frac{h}{n}$ решений. При этом на заключительной стадии для принятия окончательного решения i выполняется операция сравнения результатов отдельных ЭМ.

Таким образом, при решении задач теории статистических решений также можно получить выигрыш во времени решения, пропорциональный числу ЭМ, как в том случае, когда задача сводится к соответствующей задаче линейного программирования, так и при непосредственном решении методами математической статистики.

8.7. Примеры невычислительных задач

В предыдущих параграфах были рассмотрены основные классы задач вычислительной математики. Анализ задач невычислительного характера выходит за рамки данной книги, поэтому ограничимся только иллюстративными примерами.

Информационно-логические задачи. К информационно-логическим обычно относят задачи обработки результатов научных и инженерных исследований и разработок (исходной информацией при этом служат отчеты, статьи, книги, экспериментальные данные); поиск информации в мировом фонде литературы; обработка статистических сведений, накапливающихся в промышленности,

сельском хозяйстве, транспорте (исходными данными в этом случае служат сводки, таблицы, графики); обработка результатов наблюдений за больными в клиниках, больницах, амбулаториях; обработка результатов наблюдений за явлениями природы (обрабатываются данные, сообщаемые метеорологическими и сейсмологическими станциями, обсерваториями, спутниками Земли, автоматическими станциями космических ракет и т. п.); обработка результатов переписи населения, статистической, плановой информации; решение задач классификации в гуманитарных, биологических и других науках; перевод с одного языка на другой; исследование неизвестных письменностей и т. п.

Для информационно-логических задач характерна обработка больших массивов информации по одним и тем же алгоритмам. Хотя эти алгоритмы довольно сложны, они допускают, как правило, расчленение общего массива информации на независимые части, к каждой из которых применяется один и тот же алгоритм. Следовательно, информационно-логические задачи могут эффективно решаться на вычислительной системе. Действительно, расчленяя массив информации на n частей по числу машин в вычислительной системе, можно получить выигрыш по времени при обработке всего массива на УВС в n раз по сравнению с решением на одной ЭМ.

Рассмотрим примеры. Одной из важных задач, возникающих при решении информационно-логических проблем, является поиск слова в словаре при неупорядоченных массивах. При разбиении общего массива словаря на n частей можно ускорить поиск слова в n раз, если в каждой машине разместить соответствующую часть словаря и производить поиск слова сразу всеми ЭМ. В этом случае при решении задачи на УВС между ЭМ не происходит обмена информацией и все машины работают независимо по своей программе до того, как одной или несколькими ЭМ выдается результат.

Весьма распространенную задачу представляет сравнение больших массивов информации. Пусть, например, требуется сравнить массивы M и N, состоящие соответственно из частей M_1, M_2,..., M_n и N_1, N_2,..., N_n. Пусть это сравнение производится по некоторому алгоритму A, который требуется выполнить над каждой парой слов $b_k \in M_i$ и $c_l \in N_j$ ($i, j = 1, 2, \ldots, n$).

Разместим в каждой машине m_i части массивов M_i, N_i и начнем в каждой из них выполнять алгоритм A над словами частей массивов N_1 и M_i. Для этого одновременно с обработкой информации по алгоритму A будем передавать из машины m_1 во все остальные слова c_l массива N_1. После завершения операций над словами части массива N_1 включается на передачу вторая машина и сравниваются слова части массива N_2 со словами массива M и т. д. На каждом шаге работают все элементарные машины вычислительной системы. Следовательно, общий выигрыш во времени при решении этой задачи на УВС будет пропорционален числу ЭМ.

Задачи управления. К этому обширному классу задач относятся управление производственными процессами (станками, домнами, сложными химическими установками, поточными линиями, предприятиями и т. п.); управление транспортом (отдельными самолетами, судами, поездами, транспортными узлами и сетями и т. п.); управление хозяйственной деятельностью предприятий, учреждений, торговых и других организаций и их объединениями; управление научными экспериментами (автоматизация сложных физических, химических, биологических, технологических исследований и т. п.).

Задачи управления сводятся, собственно, к переработке информации, получаемой от управляемого объекта, к ее анализу и к принятию решения об изменении воздействий на объект. Эти задачи обычно состоят из управления отдельными процессами, характеризующимися большим числом параметров.

Это свойство уже говорит о возможности расчленения процесса решения задач управления на параллельные ветви, подобно рассмотренным выше вычислительным задачам, т. е. и в этом случае вычислительные системы оказываются применимыми.

Высокая надежность УВС дает им определенные преимущества перед другими системами управления, особенно при решении таких задач, как управление транспортными средствами, взрывоопасными процессами и т. п.

Высокая надежность УВС обусловлена однородностью их структуры. Кроме простоты и высоких эксплуатационных качеств, однородные УВС обладают большой живучестью.

Действительно, пусть задача управления каким-либо объектом выполняется УВС из n машин.

Если предположить, что в каждый данный момент может выйти из строя не более одной ЭМ, то процесс работы такой УВС можно представить следующим образом.

Вся информация, содержащаяся в каждой ЭМ, дублируется в памятях других ЭМ. В каждой из n ЭМ, кроме действующей программы, хранятся запасная программа и необходимая информация для работы при числе машин $n - 1$.

После того как установлена неисправность какой-либо ЭМ m_k, происходит перенастройка системы на работу по запасной программе. На место запасной программы и резервной информации помещаются новая запасная программа и новая резервная

информация. Нередко запасная программа будет мало отличаться от основной (в частности, это будет, когда процесс, подобно рассмотренным вычислительным задачам, разделяется на большое число одинаковых ветвей). В этом случае можно обойтись лишь внесением небольших изменений в программу.

Таким образом, система сможет работать, пока в ней останется хотя бы одна исправная машина. По мере выхода из строя машин производительность системы будет падать (пропорционально относительному уменьшению числа ЭМ), поэтому резервные программы могут предусматривать полное или частичное отключение контроля и регулировки некоторых второстепенных параметров для сохранения наиболее важных функций управления, подобно тому как это наблюдается в биологических управляющих системах.

Чтобы сделать столь же надежную (или лучше сказать „живучую") управляющую систему из разнородных машин, потребуются неизмеримо бо́льшие затраты оборудования.

* * *

В главах 7 и 8 показано, что существует широкий круг задач, процесс решения которых может быть представлен в виде большого числа параллельно выполняемых ветвей; разработана методика решения задач; найдены эффективные схемы параллельных алгоритмов для основных классов задач вычислительной математики; предложен язык описания схем параллельных алгоритмов; показана возможность построения универсальных программ, автоматически настраивающихся на заданное число машин и параметры задачи.

Опыт решения задач на системе «Минск-222» полностью подтвердил указанные теоретические выводы. Время решения задач на системе из M машин оказалось в kM раз меньше, чем на одной машине ($k \geqslant 1$). Программирование оказалось того же порядка сложности, что и для одной ЭВМ, причем составление универсальных программ не вызывает сколько-нибудь заметных трудностей. Показана возможность автоматизации программирования с помощью существующих трансляторов и установлены принципы составления трансляторов, учитывающих особенности параллельных алгоритмов.

ЗАКЛЮЧЕНИЕ

Резкое повышение производительности вычислительной техники — необходимое условие дальнейшего научно-технического прогресса.

В связи с этим большое значение приобретает выбор правильного направления в развитии средств вычислительной техники. Как показал анализ возможных путей повышения производительности электронных вычислительных машин, наиболее реален переход к одновременному выполнению большого числа операций, что в техническом аспекте означает переход от отдельных ЭВМ к вычислительным системам.

Анализ задач и существующих технических возможностей позволяет сделать вывод, что основным направлением в области создания вычислительных систем высокой производительности должна быть разработка *универсальных* вычислительных систем с переменной структурой, которые могут эффективно решать задачи различных классов.

На основе требований максимального упрощения технологии изготовления вычислительных систем и высокой надежности их работы был сделан вывод о том, что необходимо соблюдать принцип однородности как на уровне элементарных машин вычислительной системы (макроуровень), так и на уровне элементов, из которых строятся элементарные машины (микроуровень).

Соблюдение принципа однородности на макроуровне позволяет уже сейчас создавать однородные универсальные системы с переменной структурой, успешно конкурирующие с лучшими образцами существующих ЭВМ по всем основным параметрам (производительности, надежности, удобству эксплуатации и т. п.). В частности, подобные вычислительные системы могут быть построены на основе некоторых существующих серийных ЭВМ путем добавления к ним небольших дополнительных устройств.

Принцип однородности на микроуровне может быть реализован на основе вычислительной среды, которая представляет со-

бой n-мерную решетку, образованную многократным повторением одного универсального элемента, одинаковым образом соединенного со своими соседями, программно настраивающегося как на выполнение функции элементарного автомата, так и функции коммутации. В вычислительной среде с указанными свойствами программно могут быть реализованы схемы любых специализированных и универсальных вычислительных машин и систем. Однородность вычислительной среды делает ее исключительно надежной и, что особенно важно, максимально упрощает требования к технологии изготовления. Это играет исключительно важную роль при создании микроминиатюрных конструкций. Следует отметить, что во многих технических приложениях может оказаться рациональным применение вычислительной среды из обычных элементов современной вычислительной техники.

На основе анализа задач было установлено, что для широкого круга задач существуют параллельные алгоритмы, эффективно реализуемые на вычислительных системах. Во всех рассмотренных случаях время решения задач на системе из n машин оказалось практически в n раз меньше времени решения на ЭВМ, эквивалентной по своим параметрам элементарной машине системы, а если учесть различия в объеме оперативной памяти, то выигрыш во времени будет еще значительней.

Логические схемы параллельных алгоритмов могут быть сравнительно просто описаны на матричном p-языке.

Рассмотрение схем алгоритмов решения задач различных классов, записанных на p-языке, показывает, что программирование для вычислительных систем не сложнее программирования для обычных ЭВМ.

ЛИТЕРАТУРА

К главе 1

1. *Р. С. Ледли, Л. Б. Ластед.* Объективные основания диагноза.— Кибернетический сб., 1961, № 2, 5—40.
2. *R. N. Freed.* Legal implications of computer use.— Comm. ACM, 1962, N 12, 607—612.
3. *Э. В. Евреинов, Ю. Г. Косарев, В. А. Устинов.* Исследование рукописей древних майя с помощью электронновычислительной машины: а) методы, б) алгоритмы и программы.— Докл. на конф. по обработке информации, машинному переводу и автоматическому чтению текста. М., 1961.
4. *Э. В. Евреинов, Ю. Г. Косарев, В. А. Устинов.* Применение электронновычислительных машин в исследованиях письменности древних майя, т. I—III. Новосибирск, Изд-во СО АН СССР, 1961.
5. *В. А. Устинов.* Применение электронных математических машин в исторической науке.— Вопр. истории, 1962, № 8, 97—117.
6. *Н. Ю. Брайчевский.* Машинный библиографический поиск в области археологии Украины.— Мат-лы науч. семинаров по теоретическим и прикладным вопросам кибернетики. Изд-во Киевского дома науч.-техн. пропаганды, 1963.
7. Программированное обучение и кибернетические обучающие машины. Сборник статей. Под ред. А. И. Шестакова. М., Изд-во «Сов. радио», 1963.
8. Опыт применения обучающих машин в США. Зарубежная электроника, 1962, № 9.
9. *А. Н. Колмогоров.* К изучению ритмики Маяковского.— Вопр. языкознания, 1963, № 4, 64—71.
10. *А. Н. Колмогоров, А. Н. Прохоров.* О дольнике современной русской поэзии.— Вопр. языкознания, 1963, № 6, 84—95; 1964, № 1, 75—94.
11. *А. М. Кондратов.* Теория информации и поэтика.— В сб. «Проблемы кибернетики». М., Физматгиз, 1963, № 9, 279—286.
12. *Р. Х. Зарипов.* Об алгоритмичном описании процесса сочинения музыки.— Докл. АН СССР, 1960, т 132, № 6, 1283—1286.
13. *Р. Х. Зарипов.* Кибернетика и музыка. М., Изд-во «Знание», 1963.
14. *Ван Хао.* На пути к механической математике.— Кибернетический сб., 1962, № 5, 114—165.
15. *Т. Килберн, Р. Л. Гримсдейл, Ф. Г. Самнер.* Эксперименты с обучающейся и мыслящей машиной.— Кибернетический сб., 1962, № 4, 183—197.
16. *В. М. Глушков.* Введение в кибернетику. Киев, Изд-во АН УССР, 1964.
17. *В. М. Глушков.* Теория обучения одного класса дискретных персептронов.— Ж. вычисл. матем. и матем. физ., 1962, т. 2, № 2, 317—335.
18. *В. М. Глушков.* К вопросу о самоорганизации в персептроне.— Мат-лы науч. семинаров по теорет. и прикладным вопр. кибернетики. Изд-во Киевского дома науч.-техн. пропаганды, 1962.

19. *В. Н. Елкина, Н. Г. Загоруйко.* Современное состояние вычислительной техники за рубежом (обзор.) — Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1962, вып. 1, 3—33.
20. *A. L. Leiner* and oth. Using digital computers in the design and maintenance of new computers.— IRE Trans. on Electronic Computers, Dec. 1961, vol. EC — 10, 680—690.
21. *А. И. Берг, А. И. Китов, А. А. Ляпунов.* О возможности автоматизации управления народным хозяйством.— В сб. «Проблемы кибернетики», М., Физматгиз, 1961, № 6, 83—100.
22. *Е. Мануфарова, Л. Корнилов.* Работают живые цифры.— Известия, 23 июня 1964 г.
23. *В. Глушков, А. Дородницын, Н. Федоренко.* О некоторых проблемах кибернетики.— Известия, 6 сентября 1964 г.
24. *Л. И. Гутенмахер.* Электронные информационно-логические машины. М., Изд-во АН СССР, 1960.
25. *D. H. Kelley, R. I. Taylor.* Computers in the iron and steel industries.— Research, 1962, N 5. 191—194.
26. *А. В. Комаров, Н. С. Кормилицын.* Применение электронновычислительных машин в системах передачи информации.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1962, вып. 1, 34—62.
27. *А. П. Петров.* Методы решения некоторых транспортных задач на электронных цифровых вычислительных машинах.— Ежемес. бюлл. междунар. ассоц. ж.-д. конгрессов, 1962, № 2.
28. *Э. В. Евреинов, Ю. Г. Косарев.* О системах автоматизации научных экспериментов для разработки вычислительных систем.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1963, вып. 8, 3—10
29. *И. С. Лискер.* Использование электронной вычислительной машины для комплексного исследования характеристик полупроводниковых материалов и управления физическим экспериментом.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1963, вып. 8.
30. Monthly computer census.— Computers and automation, 1963, vol. XII, N 8, 7, 42—43.
31. *М. В. Келдыш, А. А. Ляпунов, М. Р. Шура-Бура.* Математические вопросы теории счетных машин.— Вестн. АН СССР, 1956, № 11, 16—37.
32. *А. А. Дородницын.* Электронные помощники человека.— Правда, 10 февраля 1961 г.
33. *В. М. Глушков.* Два универсальных критерия эффективности вычислительных машин.— Докл. АН УССР, 1960, № 4, 477—481.
34. *Н. Г. Загоруйко.* Об обмене устной информацией между человеком и вычислительной машиной.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1964, вып. 10, 3—12.
35. *B. H. McCormick.* The Illinois pattern recognition Computer — ILLIAC III.— IEEE Trans. on Electr. Comput., Dec. 1963, 791—813.
36. *Дж. Нейман.* Вероятностная логика и синтез надежных организмов из ненадежных компонент.— В сб. «Автоматы». М., Изд-во иностр. лит., 1956, 68—139.
37. *К. Шеннон.* Работы по теории информации и кибернетике. М., Изд-во иностр. лит., 1963, 114—153.
38. *Э. В. Евреинов, Ю. Г. Косарев.* О вычислительных системах высокой производительности.— Изв. АН СССР, серия техн. кибернетика, 1963, № 4, 3—25.
39. Project Lightning. High-speed data processor system research.— RCA Ind. Electr. Prod., June 1961, 123.

40. *J. Gregory* and *R. McReynolds.* The Solomon computers.— IEEE Trans. on Electr. Comput., 1963, N 6, 774—781.
41. Advances in Computers, vol. I—II. N. Y.— London, Acad. Press, 1961.
42. *Э. В. Евреинов, Ю. Г. Косарев.* О методике разработки вычислительных систем.— Сборник трудов Ин-та матем. СО АН СССР. Новосибирск, 1963, вып. 6, 3—20.
43. *В. Л. Дятлов, Л. С. Мещанинов.* Вычислительные системы и автоматизированные системы управления научными разработками.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1964, вып. 11, 7—25.
44. *Ю. А. Авдеев, А. П. Николаева.* Управление сложными разработками по методу критического пути.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1964, вып. 11, 26—52.
45. *Г. С. Поспелов, А. И. Тейман.* Метод логических диаграмм для планирования разработок сложных систем.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1964, вып. 11, 53—68.

К главе 2

1. *В. А. Зимин.* Надежность работы стандартных элементов БЭСМ.— В сб. «Надежность радиоэлектронной аппаратуры». М., Изд-во «Сов. радио», 1958, 3—26.
2. *P. Miram.* Zuverlässigkeit des Problems der Elektronik.— Bürotechn. und Automat., 1963, N 4, 99—105.
3. Project Tinkertoy.— NBS Technical News Bulletin, Nov. 1953, 161—170.
4. *Danko.* The micro-module: A logical approach to microminiaturization.— Proc. IRE, 1959, N 5, 894—903.
5. *Klass.* Space to spark avionics revolution.— Aviation Week, June 1958, 16, 248.
6. *E. Keonjian.* Microminiature electronics circuitry for space guidance.— IRE Wescon Con. Rec., 1959, vol. 3, P. 6, 92—99.
7. New molecular electronics.— Electronics, 1959, vol. 32, N 24, 43.
8. *L. S. Killby.* Semiconductor solid circuits.— Electronics, 1959, vol. 32, N 32, 110—111.
9. *H. Awender.* Mikrominiaturisierung und molekular elektronik.— Radio Mentor, 1960, vol. 26, N 2, 108—112.
10. *В. И. Фистуль.* Туннельные диоды. Сборник статей. Предисл. ред. Табл. 1. М., Изд-во иностр. лит., 1961.
11. *W. W. Bledsoe.* A basic limitation on the speed of digital computers.— IRE Trans. on Electr. Comput., 1961, N 3, 530.
12. *J. A. Swanson.* Physical versus logical coupling in memory systems.— IBM J. Res. and Develop., 1960, vol. 4, N 3, 305—310.
13. *R. Landauer.* Irreversibility and heat degeneration in the computing process.— IBM J. Res. and Develop., 1963, vol. 7, N 3, 183—191.
14. IBM announces transistorized 7070.— Res. and Engng., 1958, N 5, 38.
15. *M. Lehman.* High-speed digital multiplication.— IRE Trans. on Electr. Comput., 1957, EC-6, 204—205.
16. *I. M. Frankovich, H. P. Peterson.* A functional description of the Lincoln TX-2 Computers.— Proc. WJCC, Feb. 26—28, 1957, 146—155.
17. *L. R. Turner, A. Manos, N. Zandis.* Initial experience in multiprogramming on the Lewis Research Center 1103 Computer.— 15th Nat. Conf. ACM, Aug. 1960.
18. *I. P. Eckert* and oth. Design of Univac-Lark System.— Proc. EJCC, Dec. 1—3, 1959, 59—65.

19. *I. E. Thornton* and oth. The Univac M-460 Computer.— Proc. WJCC, 1959, 50—74.
20. *W. F. Schmidt, A. B. Tonik.* Sumpathetically programmed computers.— Infor. Processing, VI, 1959, 344—348.
21. *S. W. Dunwell.* Design objectives for the IBM Stretch Computer.— Proc. EJCC, 1957, 20—22.
22. *T. Kilburn* and oth. The Atlas supervisor.— Proc. EJCC, Dec. 1961.
23. *R. M. Beck, M. Palevsky.* The DDA.— Instrum. and Automat., 1958, N 2, 1836—1837.
24. *Г. С. Жданов, В. И. Власенко.* Счетные методы в рентгенографии и электронная счетная машина «Кристалл».— В сб. «Проблемы физ. химии», вып. 1. М., Госхимиздат, 1958, 129—138.
25. *Н. П. Брусенцов* и др. Малая автоматическая цифровая машина «Сетунь».— Вести. Моск. ун-та, серия матем. и мех., 1962, № 4, 3—12.
26. *S. Muroga, K. Takashima.* The parametron digital computer MUSASINO-1.— IRE Trans. on Electr. Comput., 1959, vol. 8, N 3, 308—316.
27. *E. N. Adams.* Applications of cryotrons to the high — speed computer.— Electr. Recht, 1962, N 5, 212—216.

К главе 3

1. *S. Unger.* Pattern recognition and detection.— Proc. IRE, Oct., 1959, 47, 1737—1752.
2. *D. L. Slotnick* and oth. The Solomon computer.— Proc. EJCC, 1962, 97—107.
3. *J. R. Ball* and oth. On the use of the Solomon parallel — processing computer.— Proc. EJCC, Dec. 1962, 137—146.
4. *J. Gregori* and *R. McReynolds.* The Solomon Computers.— IEEE Trans. on Electr. Comput., 1963, N 6, 774—781.
5. *J. H. Holland.* Universal computer capable of executing an arbitrary number of subprogramms simultaneosly.— Proc. EJCC, Dec. 1959, 108—113.
6. *J. H. Holland.* Iterative circuits computers.— Proc. EJCC, 1960, 259—265.
7. *C. Y. Lee, M. C. Paull.* A content addressable distributed logic memory with applications to information retrieval.— Proc. IRE, June 1963, 924—932.
8. *J. K. Hawkins* and *Munsey.* A parallel computer organization and mechanizations.— IEEE Trans. on Electr. Comput., June 1963, 251—262.

К главе 4

1. *A. L. Leiner, S. N. Alexander.* System organisation of the DYSEAC.— IRE Trans. on Electr. Comput., March 1954, 1—10.
2. Вычислительные машины (СЕАК и ДИСЕАК). М., Машгиз, 1958.
3. *J. P. Eckert* and oth. Design of Univac-Lark System.— Proc. EJCC, Dec. 1—3, 1959, 59—65.
4. *S. W. Dunwell.* Design objectives for the IBM Stretch Computer.— Proc. EJCC, 1957, 20—22.
5. *A. L. Leiner* and oth. Pilot- a new multiple computer system.— J. ACM, July 1959, 6, 313—335.
6. The CDC 3600.— Datamation, 1962, vol. 8, N 5, 37—40.
7. *Э. В. Евреинов.* О возможности построения вычислительных систем в условиях запаздывания сигналов.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1962, вып. 3, 3—17.

8. *J. M. Frankovich, H. P. Peterson.* A functional description of the Lincoln TX-2 Computers.— Proc. WJCC, Feb. 26—28, 1957, 146—155.
9. *G. Estrin.* Organisation of computer system: the fixes plus variable structure computer.— Proc. WJCC, May 3—5, 1960, 33—40.
10. *G. Estrin* and oth. Parallel processing in a structurable computer system.— IEEE Trans. on Electr. Comput., 1963, N 6, 747—755.
11. *G. Estrin* and *R. Turn.* Automatic assignment of computations in a variable structure computer system.— IEEE Trans. on Electr. Comput., 1963, N 6, 755—773.
12. *Э. В. Евреинов, Ю. Г. Косарев.* О возможности построения вычислительных систем высокой производительности. Новосибирск, Изд-во СО АН СССР, 1962.

К главе 5

1. *В. М. Глушков.*, Синтез цифровых автоматов. М., Физматгиз, 1962.
2. *Э. В. Евреинов.* Теоретические основы построения вычислительных сред.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1965, вып. 16, 3—72.
3. *S. Unger.* Pattern recognition and detection.— Proc. IRE, Oct. 1959, 1737—1752.
4. *J. H. Holland.* Universal computer capable of executing an arbitrary number of subprogramms simultaneously.— Proc. EJCC, Dec. 1959, 108—113.
5. *C. Y. Lee.* A content addressable distributed logic memory with applications to information retrieval.— Proc. IRE, June 1963, 924—932.
6. *D. L. Slotnick* and oth. The Solomon computer.— Proc. EJCC, 1962, 97—107.
7. *Ю. Г. Косарев.* О методике решения задач на универсальных вычислительных системах.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1965, вып. 17.
8. *Э. В. Евреинов, Ю. Г. Косарев.* О решении задач на универсальных вычислительных системах.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1965, вып. 17.
9. *В. И. Сифоров.* О методах расчета надежности работы систем, содержащих большое число элементов.— Изв. АН СССР, ОТН, 1954, № 6.
10. *М. А. Синица.* К вопросам резервирования радиоэлектронных устройств.— В сб. «Надежность радиоэлектронной аппаратуры». М., Изд-во «Сов. радио», 1958.
11. *Б. В. Гнеденко.* Курс теории вероятностей. М., Физматгиз, 1961.
12. *M. V. Wilkes.* Micro-programming.— Proc. EJCC, Dec. 1958, 18—20.
13. The CDC 3600.— Datamation, 1962, vol. 8, N 5, 37—40.
14. *Э. В. Евреинов.* О микроструктуре элементарных машин вычислительной системы.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1962, вып. 4, 3—28.
15. *В. М. Глушков.* Теория алгоритмов. Киев, 1961.
16. *А. И. Китов, Н. А. Криницкий.* Электронные цифровые машины и программирование. М., Физматгиз, 1959.
17. *А. М. Гильман.* К проекту вычислительной машины последовательного действия.— Мат-лы конф. «Пути развития советского математического машиностроения и приборостроения», ч. I, 1956, 82—91.
18. *Ю. Я. Базилевский.* Универсальная электронная вычислительная машина «Стрела».— Приборостроение, 1957, № 3.
19. R. *Gerwin.* Ein elektronischer Rechner ohne Elektronenröhren.— VDI-Nachr., 1959, N 9—10.

20. *Ф. П. Брукс, Д. А. Блоу, У. Бухгольц.* Переработка данных поразрядно и кусками.— Кибернетический сб. М., Изд-во иностр. лит., 1961, № 2, 169—186.
21. *W. Shooman.* Parallel computing with vertical data.— Proc. EJCC, 1960, 111—115.
22. *J. H. Holland.* Iterative circuits computers.— Proc. EJCC, 1960, 259—265.
23. *К. Э. Шеннон.* Универсальная машина Тьюринга с двумя внутренними состояниями.— В сб. «Автоматы». М., Изд-во иностр. лит., 1956, 213—225.
24. *A. Gill.* Cascaded finite-state machines.— IRE Trans., Sept. 1961, EC-10, 366—370.
25. *S. H. Unger.* A computer oriented toward spatial problems.— Proc. IRE, 1958, N 10, 1744—1750.
26. *Э. В. Евреинов, Ю. Г. Косарев.* О возможности построения вычислительных систем высокой производительности. Новосибирск, Изд-во СО АН СССР, 1962.
27. *Ю. Г. Решетняк.* О задаче соединения элементов вычислительной системы.— Сборник трудов Ин-та матем. СО АН СССР. «Вычислительные системы» Новосибирск, 1962, вып. 3, 17—30.
28. *Э. В. Евреинов.* О возможности построения вычислительных систем в условиях запаздывания сигналов.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1962, вып. 3, 3—17.

К главе 6

1. *В. М. Глушков.* Синтез цифровых автоматов. М., Физматгиз, 1962.
2. *Н. Е. Кобринский, Б. А. Трахтенброт.* Введение в теорию конечных автоматов. М., Физматгиз, 1961.
3. *М. А. Айзерман, Л. А. Гусев, Л. И. Розоноэр, И. М. Смирнова, А. А. Таль.* Логика, автоматы, алгоритмы. М., Физматгиз, 1963.
4. *F. C. Hennie.* Iterative arrays of logical circuits. N. Y., J. Willey, 1961.
5. Project Tinkertoy.— NBS Technical News Bulletin, Nov. 1953, 161—170.
6. *Danko.* The micro-module: a logical approach to microminiaturization.— Proc. IRE, 1959, N 5, 894—903.
7. *Klass.* Space to spark avionics revolution.— Aviation Week, June 1958, 16, 248.
8. *E. Keonjian.* Microminiature electronics circuitry for space guidance.— IRE Wescon Con. Rec., 1959, vol. 3, p. 6, 92—99.
9. New molecular electronics.— Electronics, 1959, vol. 32, N 24, 43.
10. *L. S. Killby.* Semiconductor solid circuits.— Electronics, 1959, vol. 32, N 32, 110—111.
11. *H. Awender.* Mikrominiaturisierung und molekular Elektronik.— Radio Mentor, 1960, vol. 26, N 2, 108—112.
12. *E. N. Adams.* Applications of cryotrons to the high-speed computer.— Elektr. Rech., 1962, N 5, 212—216.
13. *H. D. Crane.* Neuristor — a novel device and system concept.— Proc. IRE, 1962, N 10, 2048—2060.
14. *И. М. Тетельбаум.* Электрическое моделирование. М., Физматгиз, 1959.
15. *В. А. Веников.* Применение теории подобия и физического моделирования в электротехнике. М., Госэнергоиздат, 1949.
16. *У. Карплюс.* Моделирующие устройства для решения задач теории поля. М., Изд-во иностр. лит., 1962.
17. *Л. И. Гутенмахер.* Электрические модели. М., Изд-во АН СССР, 1949.
18. *Э. В. Евреинов.* О микроструктуре элементарных машин вычислительной системы.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1962, вып. 4, 3—16.

19. *И. М. Гельфанд, М. Л. Цетлин.* О континуальных моделях управляющих систем.— Докл. АН СССР, 1960, 131, № 6, 1242—1245.
20. *И. С. Балаховский.* О возможности моделирования простейших актов поведения дискретными однородными средами.— В сб. «Проблемы кибернетики», вып. 5. М., Физматгиз, 1961, 272—277.
21. *Н. Винер, А. Розенблют.* Проведение импульсов в сердечной мышце. Математическая формулировка проблемы проведения импульсов в сети связанных возбудимых элементов, в частности, в сердечной мышце.— Кибернетический сб. М., Изд-во иностр. лит., 1961, № 3, 7—56.
22. *И. Копи, К. Элот, Д. Райт.* Реализация событий логическими сетями.— Кибернетический сб. М., Изд-во иностр лит., 1961, № 3, 147—166.
23. *А. Беркс, Д. Райт.* Теория логических сетей.— Кибернетический сб. М., Изд-во иностр. лит., 1962, № 4, 33—57.
24. *С. В. Яблонский.* Основные понятия кибернетики.— В сб. «Проблемы кибернетики». М. Физматгиз, 1959, вып. 2, 7—38.
25. *Е. С. Федоров.* Курс кристаллографии. СПб., Риккер, 1901.
26. *А. В. Шубников.* Симметрия. М., Изд-во АН СССР, 1940.
27. *Л. А. Люстерник.* Выпуклые фигуры и многогранники. М., Гостехиздат, 1956.
28. *К. Берж.* Теория графов и ее применение. М., Изд-во иностр. лит., 1962.
29. *К. Шеннон.* Синтез двухполюсных переключательных схем.— В кн.: К. Шеннон. Работы по теории информации и кибернетике. М., Изд-во иностр. лит., 1963, 59—105.
30. *О. Б. Лупанов.* О синтезе некоторых классов управляющих систем.— В. сб. «Проблемы кибернетики». М., Физматгиз, 1963, вып. 10, 63—97.

К главе 7

1. *Э. В. Евреинов, Ю. Г. Косарев.* О возможности построения вычислительных систем высокой производительности. Новосибирск, Изд-во СО АН СССР, 1962.
2. *Ю. Г. Косарев.* О методике решения задач на универсальных вычислительных системах.— Сборник трудов Ин-та математики СО АН СССР. „Вычислительные системы". Новосибирск, 1965, вып. 17.
3. *Э. В. Евреинов, Ю. Г. Косарев.* О методике разработки вычислительных систем. Вычислительные системы.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1963, вып. 6, 3—20.
4. *Л. В. Канторович.* Математические методы организации и планирования производства. Изд-во ЛГУ, 1939.
5. *Р. Беллман.* Динамическое программирование. М., Изд-во иностр. лит. 1960.
6. *Л. В. Канторович, М. К. Гавурин.* Применение математических методов в вопросах анализа грузопотоков.— В сб. «Проблемы повышения эффективности работы транспорта». М., Изд-во АН СССР, 1949.
7. *Е. Г. Гольштейн, Д. Б. Юдин.* Об одном классе задач планирования народного хозяйства.— В сб. «Проблемы кибернетики». М., Физматгиз, 1961, вып. 5, 165—182.
8. *А. А. Ляпунов.* О логических схемах программ.— В сб. «Проблемы кибернетики». М., Физматгиз, 1958, вып. 1, 46—74.
9. *Ю. И. Янов.* О логических схемах алгоритмов.— В сб. «Проблемы кибернетики». М., Физматгиз, 1958, вып. 1, 75—127.
10. *J. Neumann.* The general and logical theory of automata.— «Cerebral Mechanisms in Behavior». The Hixon Symposium. L. A. Jeffress, N. Y.— London, 1951, 2070—2098. Русский перевод в кн. А. Тьюринг «Может ли машина мыслить?». М., Физматгиз, 1960, 59—101.

11. *А. А. Марков.* Теория алгорифмов.— Тр. матем. ин-та АН СССР, им. Стеклова, 1954, т. XLII.
12. *Л. А. Калужнин.* Об алгоритмизации задач.— В сб. «Проблемы кибернетики». М., Физматгиз, 1959, вып. 2, 51—67.
13. *В. Л. Дятлов, Л. С. Мещанинов.* Вычислительные системы и автоматизированные системы управления научными разработками.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1964, вып. 11, 7—25.
14. *Ю. А. Авдеев, А. П. Николаева.* Управление сложными разработками по методу критического пути.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1964, вып. 11, 26—52.
15. *Г. С. Поспелов, А. И. Тейман.* Метод логических диаграмм для планирования разработок сложных систем.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1964, вып. 11, 53—68.
16. *Э. В. Еврейнов, Ю. Г. Косарев.* Матричный *p*-язык для описания параллельных алгоритмов.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1965, вып. 17.
17. *C. Y. Lee.* An algorithm for Path Connections and its Applications.— IRE Trans. on Electr. Comput., 1961, EC — 10, N 3.
18. *В. Л. Харченко.* О машинном методе проектирования соединений. — Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1963, вып. 6, 32—40.
19. *В. А. Тюренков.* Алгоритм нахождения кратчайшего пути. — Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1963, вып. 6, 41—44.

К главе 8

1. *А. К. Фадеев, В. Н. Фадеева.* Вычислительные методы линейной алгебры. М., Физматгиз, 1960.
2. *Г. А. Бекишев.* О распараллеливании вычислительных алгоритмов. — Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1963, вып. 5, 22—30.
3. *Г. А. Бекишев.* Об алгоритмах, эффективно реализуемых на вычислительных системах.— Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1963, вып. 7, 24—37.
4. *Г. А. Бекишев.* К вопросу эффективности решения простейших задач алгебры матриц на вычислительной системе, состоящей из машин. — Сборник трудов Ин-та матем. СО АН СССР. „Вычислительные системы". Новосибирск, 1963, вып. 9, 3—29.
5. *Л. В. Канторович.* Математические методы организации и планирования производства. Л., Изд-во ЛГУ, 1939.
6. *G. B. Dantzig.* A proof of the equivalence of the programming problem and the game problem.— В сб. под ред. T. C. Koopmans. Activity analysis of production and allocation. N. Y., J. Willey Sons, 1951, 330—335.
7. *Л. В. Канторович.* Экономический расчет наилучшего использования ресурсов. М., Изд-во АН СССР, 1959.
8. *С. Гасс.* Линейное программирование. М., Физматгиз, 1961.
9. *Л. В. Канторович, М. К. Гавурин.* Применение математических методов в вопросах анализа грузопотоков.— В сб. «Проблемы повышения эффективности работы транспорта». М., Изд-во АН СССР, 1949.
10. *G. B. Dantzig.* Application of the simplex method to a transportation problem.— В сб. под ред. T. C. Koopmans. Activity analysis of production and allocation. N. Y., J. Willey Sons, 1951, 359—373.

11. *А. Л. Лурье.* Методы достижения наименьшего пробега грузов при составлении перевозочных схем.— В сб. «Применение математики в экономических исследованиях». М., Соцэкгиз, 1959, 354—389.
12. *H. W. Kuhn.* The Hungarian method for solving the assignment problem.— Naval Res. Logist. Quart. 2, 1955, 83—97.
13. *И. С. Жидков, Н. П. Березин.* Методы вычислений. М., Физматгиз, 1959, т. I и II.
14. *Д. Н. Ланс.* Численные методы для быстродействующих вычислительных машин. М., Изд-во иностр. лит., 1962.
15. *С. Л. Соболев.* Лекции по теории кубатурных формул. Изд. Новосибирского гос. ун-та, 1964.
16. *Н. П. Бусленко, Д. И. Голенко, И. М. Соболь, В. Г. Срагович, Ю. А. Шрейдер.* Метод статистических испытаний. М., Физматгиз, 1962.
17. *Д. Блекуелл, М. А. Гиршик.* Теория игр и статистических решений. М., Изд-во иностр. лит., 1958.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

Эдуард Владимирович Евреинов
Юрий Гаврилович Косарев

ОДНОРОДНЫЕ УНИВЕРСАЛЬНЫЕ
ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ
ВЫСОКОЙ ПРОИЗВОДИТЕЛЬНОСТИ

Редактор А. А. Сницаренко. Художественный редактор В. Г. Бурыкин.
Обложка худ. И. Е. Вяткина. Технический редактор А.М. Вялых.
Корректоры М. А. Лапшина, М. П. Оськина

Сдано в набор 8 декабря 1965 г. Подписано в печать 7 октября 1966 г. МН. 03110
Бумага 60×90¹/₁₆. 19,25 печ. л.+1 вкл. 16,9 уч.-изд. л. Тираж 4850 экз.

Издательство «Наука», Сибирское отделение.
Новосибирск-99, Советская, 20. Заказ № 187.
2-я типография издательства «Наука». Москва, Г-99,
Шубинский пер., 10.
Цена 1 р. 12 к.