Таблица 4.3.

Система команд матричного процессора

Тип команды	Код	Команда	Выполняемая операция
Арифметические операции над векторами и матрицами	VEM	Поэлементное умножение векторов	$y(i) =[y(i)]+u(i)x(i)$
	VES	Поэлементное сложение векторов	$y(i)= u(i) + x(i)$
	SMY	Умножение вектора на скаляр	$y(i) = [y(i)] + x(i)u$
	SSA	Возведение элементов вектора в квадрат	$y(i) = [y(i)] + x(i)[x(i)]$
	SSQ	Суммирование квадратов элементов вектора	$y=[y] + \sum^{n} x(i)[x(i)]$
	SVE	Суммирование элементов вектора	$y = [y] + \sum^{n} x(i)$
	VIP	Скалярное произведение векторов	$y = [y] + \sum^{n} x(i)u(i)$
	PMM	Частичное умножение матриц	Сложные операции
	CEM	Умножение комплексных векторов	
Специальные операции	CVM	Свернутое умножение	
	DEQ	Разностное уравнение	
	MAX	Сканирование вектора	
	FF2	БПФ по основанию 2	
	FF4	БПФ по основанию 4	
	FF4	Обратное БПФ по основанию 4	

Ассоциативная матрица обычно содержит до 256 слов достаточно большой разрядности, к которым можно обращаться традиционным адресным способом. Однако в дополнение к этому в АМ

все слова могут одновременно поразрядно сдвигаться вправо и влево. Именно это свойство и усиливает возможности ассоциативной обработки. В ассоциативном процессоре имеется ряд дополнительных регистров и устройств, обеспечивающих выполнение необходимых арифметико-логических операций сразу над одноименными разрядами всех слов.

Рассмотрим, как в ассоциативном процессоре выполняется операция поиска по образцу. Каждый разряд операнда-образца из ЦУУ одновременно передается на все АЛУ, куда синхронно с ним поступает разрядный срез из АМ. Каждое АЛУ производит сравнение своей пары разрядов и в случае их несовпадения устанавливает в единицу свой разряд регистра отклика (РО). После выполнения нужного числа сдвигов АМ в РО нулевыми будут обозначены те слова, содержимое которых совпадает с образцом.

Анализируя содержимое РО, можно установить номер первого совпавшего слова, номер всех совпавших слов, качество совпавших слов и т. д. Сравнение может производиться не по всей длине слова, а по части. С помощью регистра маски (РМ) отдельные слова могут маскироваться, а коммутатор К позволяет осуществлять обмен между АМ и массивом АЛУ с заданным сдвигом.

Таким образом, ассоциативную память можно определить как память, из которой находящиеся в ней данные могут быть выбраны на основании их содержания или части их содержания, а не по адресам данных.

Типичными операциями поиска и сравнения, выполняемыми ассоциативным процессором, являются следующие: “равно — не равно”, “меньше, чем — больше, чем”, “не больше, чем — не меньше, чем”, “максимальная величина — минимальная величина”, “между границами — вне границ”, “следующая величина больше — следующая величина меньше” и др. Перечисленные операции позволяют осуществлять с помощью ассоциативного процессора сложные виды числовой и нечисловой обработки.

Систолические массивы. Достижения микроэлектроники позволяют размещать на одном кристалле большое количество простых ПЭ. Скорости срабатывания ПЭ высоки, поэтому возрастает сложность создания коммутационных сетей, связывающих ПЭ. В некоторых специализированных ЭВМ можно отказаться от коммутаторов. Массивы ПЭ с непосредственными соединениями между

близлежащими ПЭ называются *систолическими*. Такие массивы исключительно эффективны, но каждый из них ориентирован на решение весьма узкого класса задач.

Рассмотрим, как можно построить систолический массив для решения некоторой задачи. Пусть, например, требуется создать устройство для вычисления матрицы $D=C+AB$, где

$$A = \begin{Vmatrix} a_{11} & a_{12} & & 0 \\ a_{21} & a_{22} & a_{23} & \\ a_{31} & a_{32} & \dots & \dots \\ & a_{42} & \dots & \dots \\ 0 & & \dots & \end{Vmatrix}, \quad B = \begin{Vmatrix} b_{11} & b_{12} & b_{13} & 0 \\ b_{21} & b_{22} & \dots & \dots \\ & b_{32} & \dots & \dots \\ 0 & & \dots & \end{Vmatrix},$$

$$C = \begin{Vmatrix} c_{12} & c_{12} & c_{13} & c_{14} & 0 \\ c_{21} & c_{22} & c_{23} & \dots & \dots \\ c_{31} & c_{32} & & \dots & \dots \\ c_{41} & \dots & & & \\ 0 & \dots & & & \end{Vmatrix}.$$

Здесь все матрицы — ленточные, порядка n. Матрица A имеет одну диагональ выше и две диагонали ниже главной; матрица B — одну диагональ ниже и две диагонали выше главной; матрица C по три диагонали выше и ниже главной.

Пусть каждый ПЭ может выполнять скалярную операцию $c+ab$ и одновременно осуществлять передачу данных. Каждый ПЭ, следовательно, должен иметь три входа: a, b, c и три выхода: a, b, c. Входные (*in*) и выходные (*out*) данные связаны соотношениями

$$a_{out} = a_{in}, \quad b_{out} = b_{in}, \quad c_{out} = c_{in} + a_{in}b_{in}.$$

Если в момент выполнения операции какие-то данные не поступили, то будем считать, что они доопределяются нулями. Предположим далее, что все ПЭ расположены на плоскости и каждый из них соединен с шестью соседними (рис. 4.14). Если распо

ложить данные, как показано на рисунке, то схема будет вычислять матрицу *D*.

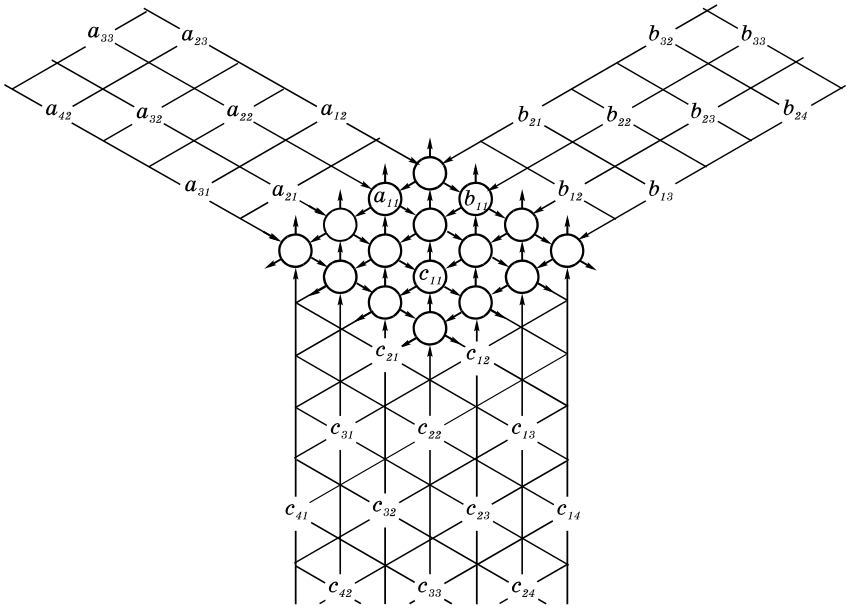


Рис.4.14. Систолический массив, реализующий операцию *D=C+AB*

Поясним работу массива. Массив работает по тактам. За каждый такт все данные перемещаются в соседние узлы по направлениям, указанным стрелками.

На рисунке показано состояние систолического массива в некоторый момент времени. В следующий такт все данные переместятся на один узел и элементы a_{11}, b_{11}, c_{11} окажутся в одном ПЭ, находящемся на пересечении штриховых линий. Следовательно, будет вычислено выражение $c_{11} + a_{11}b_{11}$. В этот же такт данные a_{12} и b_{21} вплотную приблизятся в ПЭ, находящемся в вершине систолического массива. В следующий такт все данные снова переместятся на один узел в направлении стрелок и в верхнем ПЭ окажутся a_{12} и b_{21} и результат предыдущего срабатывания ПЭ, находящегося снизу, т.е. $c_{11} + a_{11}b_{11}$. Следовательно, будет вычислено выражение $c_{11} + a_{11}b_{11} + a_{12}b_{21}$. Это есть элемент d_{11} матрицы *D*.

Продолжая потактное рассмотрение процесса, можно убедиться, что на выходах ПЭ, соответствующих верхней границе систолического массива, периодически через три такта выдаются элементы матрицы D, при этом на каждом выходе появляются элементы одной и той же диагонали. Примерно через $3n$ тактов будет закончено вычисление всей матрицы D. При этом загруженность каждой систолической ячейки асимптотически равна 1/3.

Систолический массив имеет черты как процессорных матриц (совокупность связанных ПЭ, выполняющих единую команду), так и явные признаки конвейерного вычислителя (потактное получение результата).

Многопроцессорные системы с программируемой архитектурой. В этих случаях в отличие от систолических матриц можно программно устанавливать связи между процессорными элементами и функции, выполняемые данными ПЭ. Это значительно расширяет возможности массива процессорных элементов.

Важнейшей составной частью многопроцессорной системы с *программируемой архитектурой* является универсальная коммутационная среда (УКС) [8], которая состоит из однотипных, соединенных друг с другом регулярным образом автоматических коммутационных ячеек, характеризующихся коллективным поведением.

Структура многопроцессорной системы, использующей универсальную коммутацию, приведена на рис. 4.15. Настройка системы на выполнение заданной задачи производится в два этапа.

Первый этап заключается в распределении крупных операций между ПЭ и в настройке данных ПЭ на эти операции. Примерами крупных операций являются: интегрирование, дифференцирование, матричные операции, быстрое преобразование Фурье. Использование таких макроопераций значительно упрощает программирование.

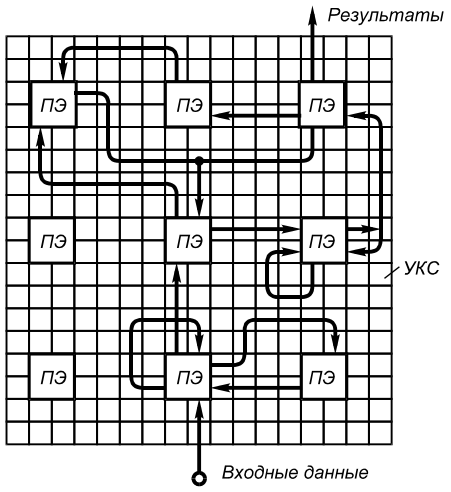
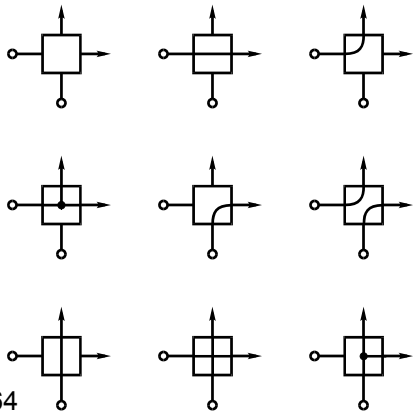


Рис. 4.15. Структура многопроцессорной системы с программируемой архитектурой

Второй этап заключается в настройке необходимых каналов связи между ПЭ в УКС. После этого осуществляется автоматический режим работы многопроцессорной системы, в течение которого на вход подается входная информация, а с выхода снимаются результаты.

Как указывалось ранее, УКС принципиально является наиболее важной частью систем с программируемой структурой. Ознакомимся с некоторыми способами построения и настройки такой среды. Для простоты будем рассматривать плоскую УКС, хотя могут быть использованы и многомерные УКС.



Простейший КЭ, необходимый для построения плоской УКС, содержит два входа и два выхода и связан с четырьмя соседними КЭ. В таком элементе возможны девять вариантов внутренней коммутации (рис. 4.16), что и обеспечивает возможность установления. Произвольных свя

Рис.4.16. Варианты внутренней коммутации простейшего КЭ

зей между ПЭ, в том числе и таких, которые изображены на рис. 4.15.

Рассмотрим настройку и функционирование плоской УКС.

Для образования каналов передачи информации намечается непрерывная цепочка свободных КЭ, соединяющих входы и выходы нужных ПЭ. При этом цепочка должна обходить как неисправные КЭ, так и элементы, уже вошедшие в другие каналы связи. После выбора канала связи во все выбранные КЭ вводится код настройки, затем канал связи становится подобным сдвиговому регистру, в котором информация потактно передается от одного КЭ к другому.

Настройка КЭ может производиться как вручную, так и автоматически. Для внешней настройки необходимо специальное внутреннее устройство управления, которое будет громоздким из-за сравнительной сложности алгоритма настройки. Поэтому более предпочтительной является самонастройка УКС.

Для автоматического образования одного канала связи между двумя ПЭ можно воспользоваться волновым принципом. Процедура распространения волны заключается в следующем. Вначале в УКС указываются входной и выходной КЭ. Затем входной элемент соединяется со всеми исправными и незанятыми соседними элементами, образуя волну подсоединенных КЭ. Если в ней не оказалось нужного выходного элемента, то каждый элемент волны соединяется с соседними, образуя новый фронт. Продвижение волны продолжается до тех пор, пока ее фронт не достигнет нужного выходного КЭ, после чего в элементы, составляющие кратчайший путь между двумя ПЭ, заносятся коды настройки, а остальные установленные связи разрываются.

Таким образом, последовательно устанавливаются все связи, необходимые для решения некоторой задачи.

Эффективность систем с программируемой архитектурой зависит от величины коэффициента $K = t_з / t_н$, где $t_з$ — время решения задачи; $t_н$ — время настройки системы. Чем больше K, тем эффективнее система. Время $t_н$ для систем с программируемой архитектурой велико, поэтому повысить K можно только увеличением $t_з$.

Один из путей увеличения $t_з$ состоит в том, что после настройки система длительное время используется для решения одной и той же задачи, при этом в каждый такт на вход системы по

ступает новый набор исходных данных, а с выхода системы в каждый такт снимаются новые результаты. Этим и объясняется очень высокое быстродействие при решении крупных задач.

Следовательно, наиболее выгодным для систем с программируемой архитектурой является режим конвейерной обработки. В то же время они принадлежат к многопроцессорным системам типа МКМД.

Контрольные вопросы

1. Объясните сущность скалярного параллелизма. Что такое длинная и сверхдлинная команды?
2. Сопоставьте классы команд CISC, RISC и MISC.
3. Назовите состав и определите функции ступеней скалярного конвейера.
4. Какое влияние на быстродействие конвейера оказывает зависимость по данным между командами?
5. Опишите влияние условных переходов на эффективность конвейера. Как ослабить это влияние?
6. Какова эффективность конвейера при выполнении многотактных команд?
7. Определите принципы построения универсальной КЭШ-памяти.
8. Каковы особенности конвейера на базе системы команд типа RISC?
9. Охарактеризуйте структуру суперскалярного микропроцессора с двумя конвейерами.
10. В чем заключается статический способ генерации программ для суперскалярных процессоров?
11. Опишите динамические способы генерации программ для суперскалярных процессоров.
12. Что такое матричные процессоры?
13. В чем сущность систолических структур?
14. Каковы отличительные признаки многопроцессорных систем с программируемой архитектурой?