ОДНОРОДНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ
(Вычислительные системы)
1981 год Выпуск 90

УДК 681.32

СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ СПЕЦИАЛИЗИРОВАННЫХ ПАРАЛЛЕЛЬНЫХ МИКРОПРОГРАММНЫХ УСТРОЙСТВ

С.Н.Сергеев

Рассмотрены вопросы использования методов параллельного микропрограммирования при проведении верхних этапов проектирования специализированных параллельных микропрограммных устройств, т.е. этапов, на которых осуществляется переход от алгоритма решения задачи к структуре устройства, выполняющего данный алгоритм.

В процессе определения структуры устройства выделим следующие четыре этапа:

- анализ задачи, сформулированной в традиционной математической форме;
- оценка требуемых характеристик устройства;
- формальное описание алгоритма;
- определение структуры устройства, синтез сети автоматов на основе формального описания.

Предлагается методика выполнения этих этапов применительно к задачам проектирования параллельных микропрограммных структур - класса вычислительных устройств, отличающихся конструктивной и функциональной однородностью и осуществляющих одновременное выполнение большого числа микроопераций.

Смысл и задачи первых двух этапов по существу не отличаются от случая синтеза устройств с последовательной обработкой. Усилия здесь направлены на выявление основных этапов вычислений и на определение общего объема вычислений в условно элементарных операциях (например, комплексное умножение, комплексное сложение и т.п.), на распределение этого объема по основным этапам вычислений и определение общей производительности и производительности на каждом этапе вычислений.

Выполнение следующих двух этапов предполагает принятие некоторой формальной системы для записи алгоритмов и соответствующей обобщенной структуры, в рамках которой будет осуществляться дальнейший синтез. В качестве формальной системы в данной работе используются алгоритмы параллельных подстановок, а способы их интерпретации дают представление об обобщенной структуре. Алгоритмы параллельных подстановок в достаточно строгом и полном изложении представлены в ряде публикаций [1,3], поэтому мы здесь приведем только основные понятия, сделав упор на их неформализованном изложении.

О п и с а н и е ф о р м а л ь н о й с и с т е м ы. Всякий алгоритм параллельных подстановок $\mathcal{A}$ задает процедуру преобразования клеточного множества W. Клеточное множество представляет собой совокупность клеток $W = \{(a_i, m_i)\}$, где m_i - имя клетки, a_i - ее состояние, причем в любом клеточном множестве все клетки имеют различные имена.

Понятие клетки имеет двойственную интерпретацию. С одной стороны, клетка представляет собой абстрактный элемент памяти, сопоставляемый элементу перерабатываемой информации при написании алгоритма. С другой,- клетка есть тот реально создаваемый элемент структуры проектируемого устройства, который должен осуществлять переработку и хранение информации в соответствии с заданным алгоритмом.

Для каждого алгоритма должны быть указаны множества, из которых берутся имена клеток (множество имен) и их состояния (множество состояний). Совокупность всех клеточных множеств, которые могут быть построены с использованием данных множеств имен и состояний, образуют область определения данного алгоритма.

Элементарное преобразование, осуществляемое над клеточным множеством, заключается в отыскании в нем некоторых подмножеств, удовлетворяющих определенным условиям (указанным в алгоритме) и в замене некоторых частей этих подмножеств другими подмножествами. Задается это преобразование с помощью оператора подстановки, имеющего вид:

$$\{(a_i, \varphi_i(m))\} \cdot \{(b_j, \psi_j(m))\} \to \{(c_i, \varphi_i(m))\}, \quad m \in M_W, \quad (1)$$

где m пробегает множество имен клеток (M_W) того клеточного множества, к которому мы хотим применить данный оператор. Выражение слева от стрелки задает те подмножества, которые нужно оты-

скать в исходном множестве, и называется левой частью, выражение слева от символа • задает те подмножества, которые необходимо удалить из исходного множества, а выражение, стоящее справа от стрелки, задает те подмножества, которые нужно добавить в исходное множество, и называется правой частью подстановки, выражение, стоящее справа от символа • в левой части, называется контекстом. В данной работе используется только один класс подстановок, для которых имена клеток в подмножествах, удаляемых и добавляемых, совпадают, т.е. в перерабатываемом множестве изменяется только состояние клеток, а имена остаются неизменными [I].

Например, при множестве состояний $A = \{0,1\}$ и множестве имен $M = N = \{1,2,3,\dots\}$ оператор подстановки: $(1,i) * (0,i+1) \to (0,i)$, примененный к исходному множеству $W = \{(I,I),(0,2),(I,3),(0,5),(I,6),(0,7)\}$, $M_W = \{I,2,3,4,5,6,7\}$, означает: проверить для всех $i \in M_W$ вхождение подмножеств $\{(1,i),(0,i+1)\}$ в W и для всех i, при которых вхождение обнаружено, осуществить замену состояния клетки $(1,i)$ на $(0,i)$. В результате получается множество $W' = \{(W \setminus \{(I,I),(I,6)\} \cup \{(0,I),(0,6)\} = \{(0,I),(0,2),(I,3),(0,5),(0,6),(0,7)\}$.

Алгоритм параллельных подстановок представляет собой список параллельных подстановок, применяемых к исходному клеточному множеству о д н о в р е м е н н о. В этом состоит одно из наиболее существенных отличий данной алгоритмической системы - любой алгоритм параллельных подстановок по определению задает вычислительный процесс в максимально параллельной форме. Однако это не означает, что в данной алгоритмической системе не может быть задана п о с л е д о в а т е л ь н о с т ь выполнения операторов. Эта возможность на формальном уровне реализуется средствами композиции алгоритмов и приемами задания сериализма (serial) в самих подстановках, основанными на выборе контекстной части и на структурировании алфавита состояний клетки. Кроме того, способы интерпретации алгоритмов представляют дополнительные возможности в этом плане. Некоторые из них будут рассмотрены в примерах.

Основу методов традиционного микропрограммирования составляют описания алгоритмов в терминах регистровых пересылок и способы их аппаратной интерпретации [2]. Простейший оператор пересылки имеет вид $R_1 \to R_2$ и означает пересылку содержимого элемента памяти (регистра) R_1 в элемент памяти R_2. Сопоставление основных понятий языков регистровых пересылок и алгоритмов параллельных

подстановок представлено в таолице, из которой видно, что основные понятия алгоритмов параллельных подстановок представ - ляют собой расширение соответствующих понятий языков регистровых пересылок на случай параллельного применения потенциально неограниченного числа операторов пересылок к потенциально неогра-

Таблица

Соответствие основных понятий языка регистровых пересылок и языка параллельных подстановок

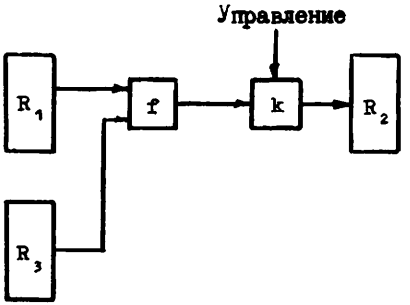
Регистровые пересылки	Параллельные подстановки
Регистр - элемент памяти с заданным множеством состояний и именем.	Клеточное множество - совокупность регистров (клеток).
Конечная совокупность регистров.	Конечная совокупность клеточных множеств.
Простая пересылка $R_1 \to R_2$: содержимое регистра R_1 переслать в регистр R_2.	Параллельная подстановка - $(a,i) * (c, i+1) \to (b, i), \quad \forall i \in M_W$ клетку (регистр) с именем i перевести в состояние "b", если клетка i находится в состоянии "a", а клетка i+1 - в состоянии "c". Причем это преобразование необходимо выполнить одновременно для всех клеток клеточного множества, т.е. i пробегает значения имен всех клеток исходного клеточного множества.
Функциональная пересылка $R_1 \overset{f}{\to} R_2$ над содержимым регистра R_1 выполнить преобразование f и результат переслать в регистр R_2.	Функциональная подстановка - $(s,i) * (b,i+1) \to (f(s),i)$ над содержимым клетки с именем i выполнить преобразование f и результат записать в эту же клетку, если клетка i+1 находится в состоянии "b". Символ s означает произвольное состояние клетки.
Микропрограмма - последовательность операторов пересылки, выполняемых в заданном порядке. Возможно одновременное выполнение некоторого, но всегда ограниченного числа операторов.	Параллельная микропрограмма - совокупность параллельных подстановок, выполняемых одновременно и повсеместно над любым клеточным множеством. Возможно последовательное выполнение подстановок, но при этом каждая подстановка выполняется над всем клеточным множеством одновременно.

ниченному массиву регистров. Продолжая это сопоставление, рассмотрим основные структурные элементы, из которых строятся вычислительные устройства, интерпретирующие алгоритмические описания в этих двух случаях.

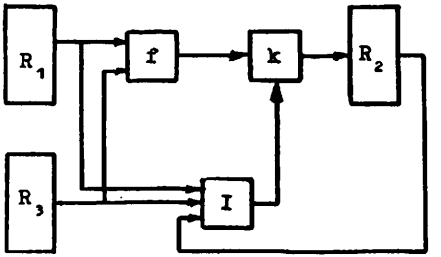
Операторы простой и функциональной регистровой пересылки интерпретируются на совокупности регистров, соединенных между собой управляемыми каналами обмена информации [2] (рис.I,а). В случае необходимости выполнить некоторый оператор пересылки (например, $f(R_1,R_3 \to R_2)$ на соответствующий ключ (ключ k на рис.I,а) должен быть подан управляющий сигнал от устройства управления. Все совокупности регистров, функциональных преобразователей, каналов передачи и ключей определяются полным набором операторов пересылки, выполняемых проектируемым устройством.

Рассмотрим интерпретацию параллельных подстановок на примере подстановки $(a,R_2)*(s,R_1)\ (r,R_3) \to (f(s,r),\ R_2)$. В соответствии с этой подстановкой необходимо выполнить преобразование f над содержимым регистров R_1,R_3 и результат записать в регистр R_2 и все это можно выполнить только в случае, если в регистре R_2 хранился код "а". Для выполнения данной подстановки необходимо в структуру, представленную на рис.I,а, ввести устройства, анализирующие содержимое регистров и управляющие каналами передачи данных между ними (рис.I,б). Полученная структура не отражает всех особенностей интерпретации параллельных подстановок, но одно существенное отличие здесь есть - это необходимость в л о к а л ь н о м управлении обменами между регистрами. Из приведенного примера этот вывод, однако, не следует с необходимостью, поскольку рассмотрен очень частный случай подстановки, где в записи конфигураций использованы конкретные имена регистров. Поэтому можно считать, что функции анализа состояний регистров возложены на устройство управления, подразумеваемое, но не показанное на рис.I,а.

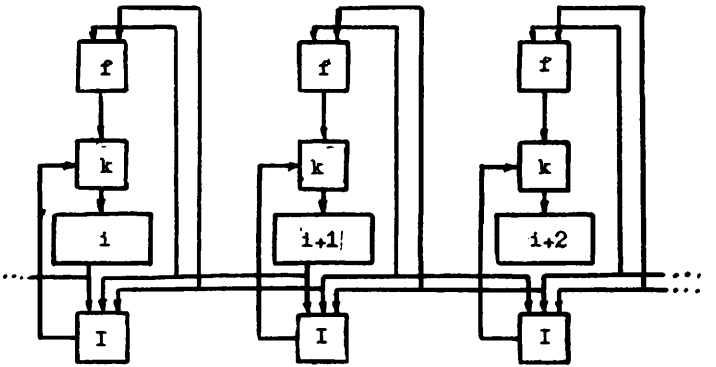
В общем случае для записи конфигураций в подстановках используются функции, определенные на потенциально бесконечном множестве имен регистров (клеток). Например, подстановка $(a,i)*\{(s,i+1)\times(r,i+2)\} \to (f(s,r),i)$, где i пробегает множество имен клеток заданного клеточного множества, интерпретируется структурой, представленной на рис.I,в. Для данной структуры возможно уже только локальное управление, в силу потенциально неограниченного числа регистров, состояние которых необходимо контролировать.



а)



б)



в)

Рис.I. Интерпретация операторов пересылки и подстановки.

Локальность управления, а также однородность и параллельная обработка информации являются наиболее существенными отличиями структур, интерпретирующих алгоритмы параллельных подстановок, от структур, интерпретирующих операторы регистровых пересылок.

Резюмируя проведенное сопоставление, можно отметить, что алгоритмы параллельных подстановок, являясь средством записи параллельных алгоритмов, включают в себя подмножество языков регистровых пересылок, а класс структур, интерпретирующих эти алгоритмы, включает в себя структуры, интерпретирующие регистровые пересылки. Таким образом, используя алгоритмы параллельных подстановок, мы можем записать любой алгоритм функционирования и сопоставить ему соответствующую структуру.

П о с т р о е н и е а л г о р и т м и ч е с к и х о п и с а н и й. Теперь мы рассмотрим основные вопросы, которые необходимо решать при построении алгоритмического описания проектируемого устройства. Есть два пути получения такого описания. Первый состоит в том, что вначале составляются алгоритмы решения задачи, на которые ориентирован спецвычислитель. Затем эти алгоритмы рассматриваются как описание работы спецвычислителя и по этому описанию строится соответствующая ему структура. Второй путь заключается в том, что для построения алгоритмов решения строится некоторый моделирующий алгоритм, который и является описанием проектируемого спецвычислителя. Однако в [1] показано, что оба эти пути приводят к одному классу устройств, названных параллельными машинами. Поэтому мы далее рассмотрим только первый подход.

При построении алгоритмических описаний необходимо прежде всего решить вопрос о выборе к л е т о ч н ы х множеств и а л ф а в и т а с о с т о я н и й клеток. При этом каждому элементу исходной, промежуточной и результирующей информации должна быть сопоставлена своя клетка, имеющая множеством состояний - множество значений, которые может принимать данная величина. Каждому массиву сопоставляется клеточное множество. Структура имен клеток обычно выбирается таким образом, чтобы учесть структуру массивов информации, фигурирующих в исходной задаче. Например, пусть объектом преобразования является матрица $A[m, n]$. В зависимости от алгоритма эту матрицу можно рассматривать либо как двумерный массив чисел, либо как одномерный массив мощностью $n \times m$. Соответственно выбранному представлению множество имен клеток будет либо $M = N \times N$, либо $M = N$ и исходной матрице A будет сопоставлено кле -

точное множество $W = \{(a_{i,j},(i,j)\}$ $(i = 1,\ldots,m;\ j = 1,\ldots,n)$ или клеточное множество $W = (a_{i,j}\ k)$. В этих множествах состояние клетки (i,j) или $k = f(i,j)$ соответствует значению элемента матрицы $a_{i,j}$. Множестве имен может иметь более сложную структуру, если в задаче используются массивы с различной организацией. Например, если наряду с матрицей в задаче используются вектор b и скалярная величина c^0, то множество имен естественно считать состоящим из трех подмножеств $M = M_1 \cup M_2 \cup M_3$, где $M_1 = N \times N$, $M_2 = N$, $M_3 = \{m_0\}$. Можно, конечно, было принять в качестве M только множество M_1 (или только M_2) и в нем представить все используемые в задаче величины и массивы, но это усложнило бы клеточные множества, сопоставленые этим массивам, усложнило бы в итоге алгоритм и его интерпретацию, и при этом было бы утеряно соответствие структуры клеточных множеств и структуры используемых в задаче данных, что существенно усложнило бы процесс написания алгоритма.

На выбор а л ф а в и т а с о с т о я н и й существенное влияние в равной мере оказывают два аспекта - информационный и управленческий. Это следует из самого понятия подстановки, поскольку единственным фактором, учитываемым при определении применимости подстановки, является состояние клетки. Поэтому, как правило, алфавит состояний заранее не фиксируется, а строится по мере необходимости в процессе написания алгоритма. Заранее определяется только часть алфавита состояний, служащая для представления числовой информации в исходной задаче. Эту часть мы будем называть основным алфавитом состояний и обозначать A_0. Множество вспомогательных символов A_1, которые вводятся при написании алгоритма с целью реализации необходимых управляющих функций, либо добавляется к основному алфавиту, и тогда имеем $A = A_0 \cup A_1$, либо используется как множество меток основных символов, тогда $A = A_0 \times A_1$.

Пусть, например, исходной информацией является некоторая величина α, которая поступает на проектируемый вычислитель от внешнего источника. Для того чтобы отметить факт поступления нового значения α, мы введем вспомогательный символ "-" (надчеркивание). Тогда если клетка, являющаяся хранителем величины α, находится в состоянии, помеченном символом "-", то этот факт означает поступление нового значения исходной величины, а отсутствие признака "-" означает, что данное значение исходной величины уже обрабатывалось алгоритмом. В этом случае алфавитом состояний будет множество $A = A_0 \times \{"-"\}$.

В процессе написания алгоритма могут возникать и более сложные структуры алфавита состояний, такие например, как $A = A_0 \times A_1 \cup A_2$.

Среди вопросов, которые приходится решать на этапе формального описания алгоритма, наиболее важным (кроме уже рассмотренных) являются:

- организация параллельной обработки и организация взаимодействия отдельных подалгоритмов, реализуемых проектируемым устройством.

Алгоритмы параллельных подстановок, как уже отмечалось, позволяют задать предельно параллельный вычислительный процесс, когда все операторы подстановок применяются одновременно и повсеместно на всем клеточном множестве. Поэтому при написании алгоритма основной целью является введение максимально допустимой для данной задачи степени параллелизма. Ограничения на степень параллелизма определяются, с одной стороны, взаимосвязью отдельных шагов вычислений по информации, с другой, - возможной противоречивостью полученного алгоритма, когда одновременное выполнение некоторых операторов подстановки может привести к неопределенности результата [I]. В этом случае приходится принимать специальные меры, чтобы исключить одновременное применение этих операторов. Это достигается выбором соответствующих структур алфавита, множества имен и специальным подбором контекстных частей в подстановках. Например, пусть подстановки $\Pi_1: S_1 \bullet S_2 \to S_3$ и $\Pi_2: S_1' \bullet S_2' \to S_3'$ противоречивы при применении к клеточным множествам с именами клеток из множества имен M и алфавитом состояний A_0. Тогда для того чтобы исключить их одновременное применение, можно, например, построить новое множество имен $M_1 = M \cup \{m_0\}$, $m_0 \notin M$, новый алфавит состояний $A = A_0 \cup \{s_0, s_1\}$, $s_0, s_1 \notin A_0$, добавить во все клеточные множества из области определения алгоритма клетку (s_0, m_0) или клетку (s_1, m_0); левую часть в Π_1 преобразовать к виду $s_1 \bullet s_2 \cup (s_0, m_0)$, левую часть в Π_2 – к виду $s_1' \bullet s_2' \cup (s_1, m_0)$ и добавить подстановки: $\Pi_3: (s_0, m_0) \to (s_1, m_0)$; $\Pi_4: (s_1, m_0) \to (s_0, m_0)$. Подстановки Π_3, Π_4, применяясь поочередно (одновременно они никогда не применяются), будут разрешать применение либо Π_1, либо Π_2, но только одной из них. Иногда способ интерпретации алгоритмов предоставляет дополнительные возможности в реализации нужной последовательности выполнения подстановок. Например, параллельные машины I-го типа по классификации, приведенной в [I], одновремен-

ное выполнение всех подстановок осуществляют путем имитации на основе циклического выполнения последовательного блока подстановок. Эту особенность машин I-го типа можно при необходимости использовать для выполнения операторов в нужной последовательности.

Пример алгоритмического описания. Рассмотрим теперь совместное выполнение всех этапов построения структуры на одном практическом примере.

Пусть требуется спроектировать устройство, решающее следующую задачу. Имеется входная последовательность цифровых отсчетов x_{ij} $(i = 1,\dots,n_1;\ j = 1,\dots,n_2)$, поступающих с интервалом τ. Необходимо для каждой подпоследовательности $x(i) = x_{i1}, x_{i2}, \dots, x_{in_2}, i = 1,\dots,n_1$, выполнить дискретное преобразование Фурье, т.е. получить подпоследовательность $\tilde{X}(i) = \tilde{X}_{i_1}\ \tilde{X}_{i_2} \dots \tilde{X}_{in_2}$, где

$$\tilde{x}_{i_k} = \sum_{r=1}^{n_2} b_k^r x_{i_r},$$

b_i^r - коэффициенты матрицы Фурье порядка n_2. Причем требуется получить результат за время $T = (n_1+1)n_1 \cdot \tau$ после поступления первого отсчета. Объем необходимых вычислений W, выраженный общим числом арифметических операций равен

$$W = \sum_{1}^{n_2} W(\tilde{X}(i)),$$

где $W(\tilde{X}(i)) = \sum_{1}^{n_1} W(\tilde{x}_{i_k})$, $W(\tilde{x}_{i_k}) = c_1 n_1 + c_2(n_2-1)$, c_1 - сложность операции умножения, выраженная числом условных тактов, c_2 - сложность операции сложения.

Чтобы определить производительность, необходимо выяснить, как распределяется данный объем вычислений по всему периоду T. В качестве основного расчетного алгоритма для вычисления $\tilde{X}(i)$ принимаем обычное матричное преобразование (не быстрое) исходной последовательности с помощью матрицы Фурье. Особенность этого алгоритма состоит в том, что все $\tilde{X}(i)$ можно вычислять одновременно и по мере поступления отсчетов x_{ij}. Поэтому весь объем вычислений можно равномерно распределить по всему периоду T. В этом случае имеем среднюю производительность $P = \frac{W}{T}$ опер./сек*).

*) Отметим, что выбор в качестве расчетного алгоритма быстрого преобразования Фурье для данной задачи привел бы к более высоким требованиям по производительности, несмотря на существенное уменьшение объема вычислений. Это связано с тем, что алгоритм БПФ можно применять имея всю последовательность отсчетов или хотя бы половину их.

А л г о р и т м ф у н к ц и о н и р о в а н и я. В результате выполнения заданного преобразования мы должны получить n_1 последовательностей длиной n_2 каждая. Следовательно, результирующую информацию можно представить двумерным массивом размером $n_1 \times n_2$, в котором i-й столбец соответствует i-й последовательности, а элемент с именем (i,j) - элементу $\tilde{x}_{ij}$. Кроме клеток для хранения результирующей информации, для каждого столбца отведем клетку для хранения очередного отсчета соответствующего данному столбцу (клетка с именем (i,0)) и клетку (с именем m_i) для хранения очередного коэффициента Фурье, на который умножается текущий отсчет. Кроме того, имеется клетка m_0, которая является источником входной последовательности.

С учетом сказанного будем иметь множество имен следующего вида: $M = M_1 \cup M_2$, $M_1 = \{0,1,2,\ldots,n_2\} \times \{1,2,\ldots,n_1\}$, $M_2 = \{m_0, m_1, m_2, \ldots, m_{n_1}\}$.

В качестве основного алфавита состояний считаем множество значений элементов входной последовательности и элементов матрицы Фурье с учетом заданной точности вычислений.

Поскольку вся последовательность отсчетов поступает от одного источника, то необходимо наладить распределение входных отсчетов так, чтобы отсчеты, соответствующие i-му столбцу воспринимались только клеткой (i,0). Это делается следующим алгоритмом: Φ_0: $(r,(i,0)) * (\bar{s},(i \ominus_{n_1} 1,0)), (t,m_0) \to (\bar{t},(i,0))$, применяемым к клеточному множеству вида $(a_1,(1,0)), (a_2(2,0)), \ldots$ $\ldots, (a_{n_2}, (n_2,0))(a_{n_2+1}, m_0)\}$.

Работа алгоритма Φ_0 происходит следующим образом. Если в перерабатываемом клеточном множестве клетка с именем $(i \ominus_{n_1} 1,0)$ $(i = 1,2,\ldots,n_1)$ находилась в некотором состоянии s с признаком "-" (надчеркивание), то клетка с именем (i,0) переходит в состояние, в котором находилась клетка с именем m_0, и переносит на него признак "-". Таким образом происходит распределение отсчетов по столбцам. Основным синхронизирующим элементом здесь является признак "-". Наличие его в соседней клетке означает, что следующий отсчет относится к данному столбцу. Кроме того, присутствие этого признака является также указателем того, что очередной отсчет для данного столбца в клетку (i,0) поступил и его можно использовать для выполнения очередной итерации.

Далее вычисление происходит следующим образом. Поступивший новый отсчет для i-го столбца должен быть использован для подсчета текущих значений сумм $\tilde{x}_{ij}$ для всех j $(1,\ldots, n_2)$. Это выполняется по следующему алгоритму:

$$\Phi_1\colon (\langle q, s\rangle,(i,j)),(\langle \bar{t}, \bar{p}\rangle,(i,j-1))\bullet(\langle r,j\rangle, m_i) \to$$

$$\to (\langle \bar{t}, \overline{s+t\cdot r}\rangle,(i,j))(\langle t,p\rangle,(i,j-1)).$$

Здесь алфавит состояний имеет вид $A = [A_0 x\{"-"\}]x[A_0 x\{"-"\}]$. Такое структурирование алфавита позволяет хранить в одной клетке пару символов основного алфавита с надчеркиванием или без него. Причем состояние клетки можно рассматривать как один символ, либо как пару символов и использовать компоненты пары независимо друг от друга, как в приведенном алгоритме. На этапе структурного проектирования, когда еще не определены способы выполнения элементарных операций, целесообразно считать, что эти операции как-то определены на алфавите состояний, и использовать их далее в записи подстановок. Это, во-первых, существенно сокращает запись и, во-вторых, дает возможность многоступенчатого описания алгоритма с постепенной детализацией выполняемых функций.

Работа алгоритма происходит следующим образом. Если клетка $(i,j-1)$ имеет обе компоненты состояния с признаком "-", а вторая компонента состояния клетки m_i совпадает со второй координатой имени клетки (i,j), то происходит следующее: клетка (i,j) в качестве первой компоненты состояния принимает первую компоненту состояния клетки $(i,j-1)$, а вторую компоненту вычисляет на основе старого значения второй компоненты своего состояния (предыдущее значение суммы $\tilde{x}_{i,j}$), первой компоненты состояния клетки $(i,j-1)$ (новый отсчет) и первой компоненты состояния клетки m_i ((k,j)-й коэффициент Фурье). Клетка $(i,j-1)$ при этом освобождается от признака готовности информации "-".

Построение сети автоматов. Алгоритмы Φ_0 и Φ_1 в совокупности составляют алгоритмическое описание процессора, по которому можно построить его структуру, т.е. сеть автоматов, интерпретирующую данные алгоритмы. Сеть, соответствующая алгоритму Φ_0, имеет вид, представленный на рис.2 (подсеть G_0). Функция автомата в этой сети состоит в проверке признака "-" у состояния соседнего автомата и приеме информации от автомата с именем m_0. Сеть, соответствующая алгоритму Φ_1, представлена

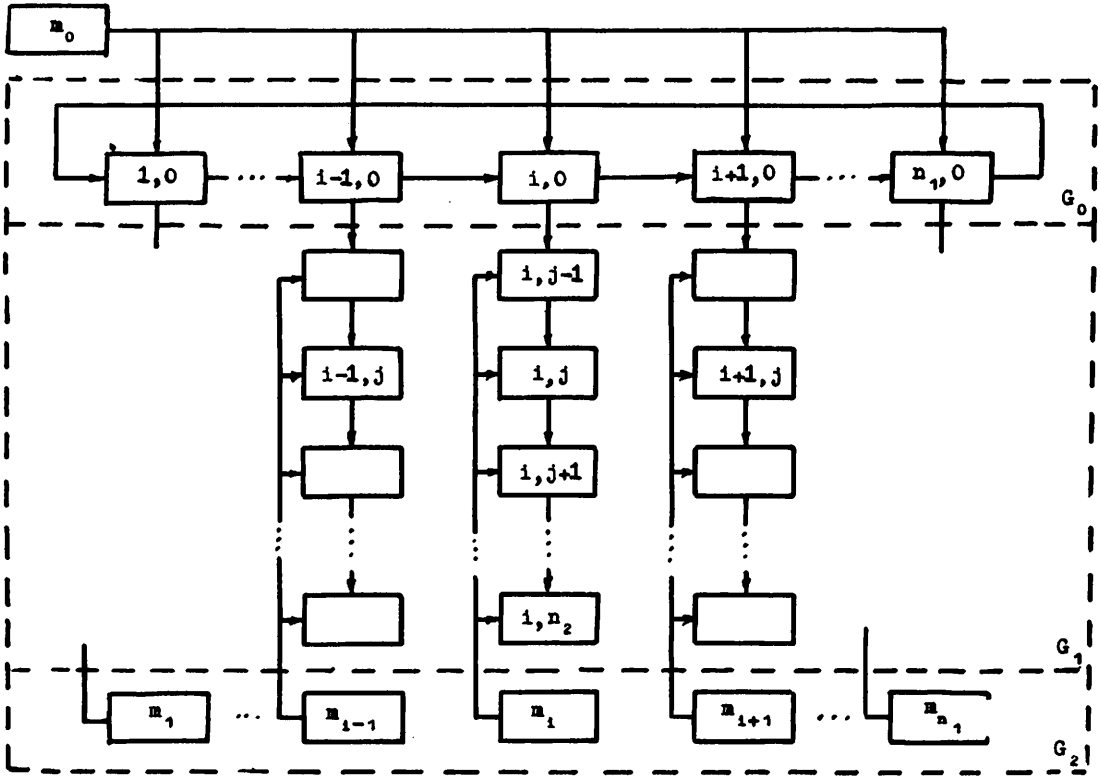


Рис.2. Сеть автоматов, интерпретирующая алгоритмы Φ_0, Φ_1.

подсеты G_1 на рис.2. Автомат с именем (i,j) в этой сети должен выполнять операции, определенные на алфавите состояний (сложение и умножение), контролировать наличие признака "-" у соседнего автомата и сверять содержимое второй компоненты автомата m_i со второй компонентой своего имени. Память автомата должна быть рассчитана на представление пары символов в алфавите $A_0 \times \{"-"\}$. Подсеть G_2 на рис.2 считается внешней по отношению к построенной и является источником коэффициентов Фурье для нее. На этом построение структуры заканчивается. Далее на основании полученных характеристик и функций элементарных автоматов сети, а также с учетом выбранной элементной базы и заданных ограничений на габариты, вес, энергопотребление можно проводить логический и технический синтез каждого автомата.

З а к л ю ч е н и е. В работе показано, что использование алгоритмов параллельных подстановок для построения алгоритмических описаний, а способов их интерпретации для определения структуры позволяет проводить формализованное построение структуры параллельных микропрограммных устройств. Рассмотренные приемы практического построения описаний могут послужить основой для созда - ния процедур автоматизированного проектирования.

Л и т е р а т у р а

I. Методы параллельного микропрограммирования /Анишев П.А., Ачасова С.М., Бандман О.Л., Пискунов С.В., Сергеев С.Н. Под ред. Бандман О.Л. - Новосибирск: Наука, 1981. - 182 с.

2. ХАССОН С. Микропрограммирование. Вып. I.-М.: Мир, 1973. - 240 с.

Поступила в ред.-изд.отд.
4 ноября 1981 года