АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ С ПРОГРАММИРУЕМОЙ СТРУКТУРОЙ
(Вычислительные системы)
1980 год Выпуск 82

УДК 681.326.3

# ЯЗЫК МИКРОПРОГРАММНОГО ОПИСАНИЯ МОДЕЛЕЙ ОДНОРОДНЫХ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ УСТРОЙСТВ

С.В. Пискунов

## В в е д е н и е

В настоящее время широкое распространение получило имитационное моделирование ЦВМ. Предложен целый ряд языков и систем моделирования. Языки отражают различные уровни детализации моделируемого устройства: от уровня логических вентилей до системного уровня [1,2,3]. Наряду с этим быстро развивается микропрограммирование - эффективное средство проектирования и программирования ЦВМ, и также строятся соответствующие языки моделирования [4,5].

Анализ языков описания оборудования и языков микропрограммирования показывает, что им присущи следующие ограничения:

1) ориентация на традиционную архитектуру ЦВМ в наборе операций;

2) необходимость указания в вычислительном устройстве всех элементов и связей между ними на соответствующем уровне описания структуры и функционирования;

3) невозможность задания параллельного выполнения микрокоманд во времени и повсеместного в пространстве при произвольном числе функциональных элементов в устройстве.

Все это затрудняет применение указанных языков для описания структуры и функционирования параллельных однородных вычислительных устройств, как-то: однородных машин [6], ассоциативных процессоров [7,8], вычислительных сред [9], систем из микропроцессоров [10], асинхронных однородных управляющих структур [11] и др. Требуются языки, которые отражали бы локальность взаимодействий ячеек, параллельность переработки информации в устройстве и обеспечивали моделирование на уровне детализации, соответствующем

уровню регистровых передач ЦВМ. Прототипом таких языков является комплекс средств описания функционирования параллельных однород - ных устройств, названный параллельным микропрограммированием [I2] и основанный на алгоритмах параллельных подстановок [I3].

Целью статьи является описание основных конструкций языка, базирующегося на понятиях параллельного микропрограммирования, и демонстрация его использования для построения микропрограммных моделей однородных параллельных устройств.

В качестве модели любого однородного параллельного вычислительного устройства используется совокупность взаимодействующих блоков. Отдельный блок - это массив однотипных ячеек. В каждую ячейку параллельно поступают все микрокоманды микропрограммы, выполняемой блоком. Отдельная ячейка в зависимости от собственного состояния и состояния соседних ячеек выбирает и выполняет ту или иную микрокоманду микропрограммы. Это отличает предлагаемый язык от языков регистровых передач, в которых основной структурной единицей является регистр и все преобразования информации осуществляются в совокупности регистров.

## §I. Краткая характеристика алгоритмов параллельных подстановок

Пусть $A$ - конечный алфавит, $M$ - множество имен с мощностью не более чем счетной. Клеткой $m$, которая находится в состоянии $a$, называется пара $(a,m) \in A \times M$. В алгоритмах параллельных подстановок основным объектом преобразования является клеточное множество $W$ - конечная совокупность клеток, в которой нет ни одной пары клеток с одинаковыми именами. Совокупность всех клеточных множеств в $(A,M)$ будем обозначать $K(A,M)$.

Для любого конечного $k$ запишем множество пар: $\{(a_1,\varphi_1(m)), \ldots, (a_k,\varphi_k(m))\}$, где $a_i \in A$, $\varphi_i : M \xrightarrow{b} M$ такие, что для $i \neq j$, $i,j = 1,2,\ldots,k$, и $\forall m \in M$ $\varphi_i(m) \neq \varphi_j(m)$. Упорядочим элементы $M$: $\{m_1,m_2,\ldots,m_p,\ldots\}$. Возьмем некоторое $m_1 \in M$ и, подставив вместо $m$ во все функции $\varphi_i$, $i = 1,2,\ldots,k$, получим клеточное множество $W_1 = \{(a_1,\varphi_1(m_1)),\ldots,(a_k,\varphi_k(m_1))\}$. Проделаем эту процедуру для всех $1 = 1,2,\ldots,|M|$. Полученное таким образом множество $\{W_1\}_{1=1,2,\ldots,|M|}$ будем называть **конфигурацией** $S$ в $K(A,M)$ и записывать как $S = \{(a_1,\varphi_1(m)),\ldots,(a_k,\varphi_k(m))\}$. Конкретный элемент конфигурации, например, $W_1$ будем часто обозначать $S(m_1)$.

Пусть даны две конфигурации: $S_1=\{(a_1,\varphi_1(m)),\dots,(a_{n_1},\varphi_{n_1}(m))\}$ и $S_2=\{(b_1,\phi_1(m)),\dots,(b_{n_2},\phi_{n_2}(m))\}$, такие что $\forall m \in M$, $\forall i,j$, $i=1,\dots,n_1$, $j=1,\dots,n_2$, имеет место $\varphi_i(m) \neq \phi_j(m)$. Тогда **произведением** $S_1 * S_2$ конфигураций $S_1$ и $S_2$ называется конфигурация вида

$$\{(a_1,\varphi_1(m)),\dots,(a_{n_1},\varphi_{n_1}(m)),\ (b_1,\phi_1(m)),\dots,(b_{n_2},\phi_{n_2}(m))\}.$$

Процесс переработки клеточного множества $W \in K(A,M)$ заключается в применении к нему операций подстановок. **Подстановкой** называется выражение вида $S_1 * S_2 \to S_3$, где $S_1, S_2, S_3$ – конфигурации; $S_1 * S_2 = S$ называется левой частью, $S_1$ – контекстом, $S_3$ – правой частью. Если существуют имена $m_1, m_2, \dots, m_p \in M$ такие, что $S(m_1), S(m_2), \dots, S(m_p) \subseteq W$, то подстановка считается применимой к $W$ и результат применения подстановки есть клеточное множество

$$\{W \setminus \bigcup_{i=1}^{p} S_2(m_i)\} \cup \{\bigcup_{i=1}^{p} S_3(m_i)\}.$$

Если таких имен не существует, то подстановка неприменима к $W$.

Конечное множество $\Phi$ подстановок, записанных в произвольном порядке, называется **системой параллельных подстановок**. Применение системы к $W \in K(A,M)$ задается итерационной процедурой. Пусть клеточное множество $W^{i-1}$ – результат $(i-1)$-й итерации. Если ни одна из подстановок неприменима к $W^{i-1}$, то $W^{i-1}$ является результатом применения $\Phi$ к $W$. Если же какие-то подстановки из $\Phi$ применимы (будем обозначать их символами $\Pi^1, \Pi^2, \dots, \Pi^k$), то результатом применения $\Phi$ к $W^{i-1}$ является клеточное множество

$$W^i = \{W^{i-1} \setminus \bigcup_{j=1}^{k} (\bigcup_{i=1}^{p^j} S_2^j(m_i^j))\} \cup \{\bigcup_{j=1}^{k} (\bigcup_{i=1}^{p^j} S_3^j(m_i^j))\}.$$

Оно служит исходным для следующей итерации. Система $\Phi$ вместе с итерационной процедурой применения подстановок называется **алгоритмом параллельных подстановок**.

## §2. Основные конструкции языка моделирования

2.1. О б щ а я  х а р а к т е р и с т и к а  я з ы к а. Средства языка обеспечивают описание как структуры моделируемого устройства, так и алгоритмов его функционирования. Структуру устройства характеризуют на макроуровне:

1) размерность блока,
2) размеры блока,
3) число блоков в устройстве,
4) взаимосвязи блоков в устройстве;

на микроуровне:

1) алфавит ячейки,
2) число входов ячейки,
3) число выходов ячейки,
4) топология связей ячейки по входам и выходам.

К характеристикам алгоритма функционирования относятся:

1) функциональное преобразование, реализуемое ячейкой,
2) параллельное и повсеместное выполнение функциональных преобразований во всех ячейках устройства.

Первую группу характеристик отражают конструкции языка, описанные в пп.2.3, 2.4, 2.5; вторую - конструкции, описанные в пп. 2.2 и 2.6.1; третью - конструкции, описанные в пп.2.6 и 2.7.

ЗАМЕЧАНИЕ. В языке микропрограммирования используются имена процедур определения координат соседей (п.2.4) и функциональных преобразований (п.2.6.1). Сами процедуры определения координат соседей и функциональные преобразования оформляются в виде процедур на языке реализации микропрограммных моделей, в качестве которого принят язык высокого уровня (ПЛ-1).

2.2. А л ф а в и т  я з ы к а. В качестве имен состояний ячеек устройства в языке используются символы (возможно, с индексами) шестидесятисимвольного алфавита, кроме символов $ , # , @ , машин серии ЕС. Этот алфавит образует группу основных символов, которые можно разбить на классы. Имя класса образует пара символов, первым в паре является знак $ , вторым - буквенный символ, возможно, с индексом. Имена классов образуют группу переменных символов языка, среди которых имеется символ @, обозначающий любой символ языка. Переменные символы используются только в записи микрокоманд. Основные символы могут быть представлены в виде символьных кодов в некотором базовом алфавите, в качестве которого

применяются различные подмножества шестидесятисимвольного алфавита, чаще всего {0,I}. Для представления переменных символов используются коды в расширении базового алфавита, т.е. при пополнении последнего символом ✱ , обозначающим любой символ базового алфавита.

2.3. К л е т о ч н ы й  м а с с и в является аналогом клеточного множества и ставится в соответствие блоку устройства. Элементами массива служат символьные коды, представляющие состояния соответствующих ячеек блока. Положение ячейки в блоке отражает набор координат соответствующего элемента клеточного массива. Для всех массивов, соответствующих блокам устройства, принята единая система координат. Клеточный массив обозначается служебным словом TS с положительным целым индексом.

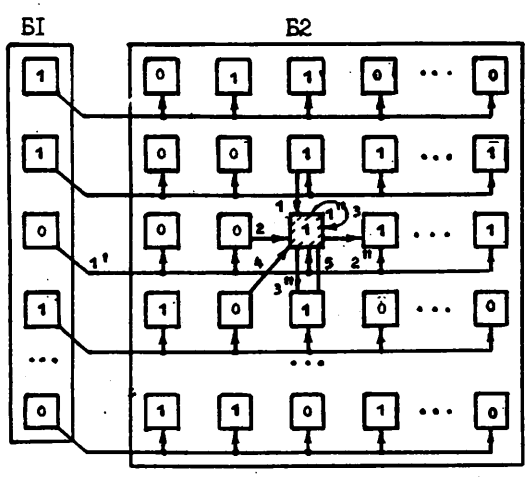



Рис. I

Устройство, изображенное на рис.I, состоит из двух блоков. Символами {0,I} внутри квадратов обозначены внутренние состояния ячеек. Перенумеруем входы и выходы выделенной ячейки так, как показано на рисунке. Каждая ячейка блока Б2 может выполнять любое из двух преобразований информации: а) если ячейка находится в состоянии I (вход 3 находится в состоянии I), вход I - в состоянии I, вход 2 - в состоянии 0, вход 4 - в состоянии 0 и вход I' от соответствующей ячейки в БI - в состоянии 0, то ячейка переходит в состояние 0 (по выходу I" записывает в себя 0) и переводит соседнюю справа ячейку по выходу 2" в состояние I; б) если вход 3 находится в состоянии I, вход I - в состоянии 0, вход 5 - в состоянии 0 и вход I' от соответствующей ячейки в БI в состоянии I, то ячейка переходит в со-

стояние 0 и переводит соседнюю снизу ячейку по входу 3" в состояние I. Блоку I поставим в соответствие массив TS1 ,блоку 2 - массив TS2 . Их запись приведена в табл. I.

Таблица I

| I \ J | TS1 | TS2 | | | | | |
|---|---|---|---|---|---|---|---|
| I | 'I' | '0' | 'I' | 'I' | '0' | . . . | '0' |
| | 'I' | '0' | '0' | 'I' | 'I' | . . . | 'I' |
| | '0' | '0' | '0' | 'I' | 'I' | . . . | 'I' |
| | 'I' | 'I' | '0' | 'I' | '0' | . . . | '0' |
| | ... | | | . . . | | | |
| | '0' | 'I' | 'I' | '0' | 'I' | . . . | '0' |

Размерность массива задается перечислением координат после имени массива: TS1(I) - одномерный, TS2(I,J) - двумерный, взаимное расположение - именами координат: TS1(I), TS2(I,J) .

Клеточный массив, состоящий из одного элемента, задается без указания координат. Отдельный элемент клеточного массива, рассматриваемый как самостоятельный клеточный массив, задается конкретным набором координат, TS2(2,3); строка - конкретным заданием первой координаты: TS 2(2,J ); столбец - второй: TS 2(I,3).

2.4. Процедуры определения координат соседей. В языке используются имена процедур,каждая из которых является аналогом набора функций, задающего элементы конфигурации. Эти процедуры позволяют вычислять координаты тех элементов, соответствующие ячейки которых являются соседними по входам или выходам. Полученные таким способом элементы будем называть соседями по входам и выходам. Процедуры вычисления сосе - дей могут быть двух типов. Процедуры первого типа вычисляют координаты соседей суммированием с заданным набором констант - шаблоном. Имя такой процедуры состоит из служебного слова PAT и целого положительного индекса. Например, для TS2 (рис.I) процедура вычисления соседей элемента (I,J) по входам состоит в вычислении следующих значений:

$$(I-1,J)(I,J-1)(I,J)(I+1,J-1)(I+1,J) .$$

Аналогично по выходам: $(I,J)(I,J+1)(I+1,J)$ .

Эти процедуры задаются с помощью шаблонов:

PAT1 = (-1,0) (0,-1) (0,0) (1,-1) (1,0);

PAT2 = ( 0,0) (0,1 ) (1,0).

И процедуру, и шаблон будем обозначать одинаково.

Процедуры второго типа определяют координаты соседей вычислением заданных функций, не сводящимся к суммированию с шаблоном. Отметим, что функции могут зависеть не только от координат, но и от других параметров, например, времени. Имя такой процедуры состоит из слова FUN и целого положительного индекса. Пример процедуры FUN приведен в следующем разделе вместе с примером композиции массивов.

2.5. Композиция клеточных массивов. Если устройство состоит из нескольких блоков, причем ячейки одного блока имеют соседей по входам в другом блоке, то соответствующие клеточные массивы образуют композицию. Например (рис.1), ячейки блока Б2 имеют соседей по входам в Б1. Пара TS1 , TS2 образует композицию. Словами TS2 и TS1 обозначаются соответственно основной и контекстный клеточные массивы. При выполнении микрокоманды подстановки, в записи которой используются имена таких массивов, индексы просмотра этих массивов меняются синхронно.

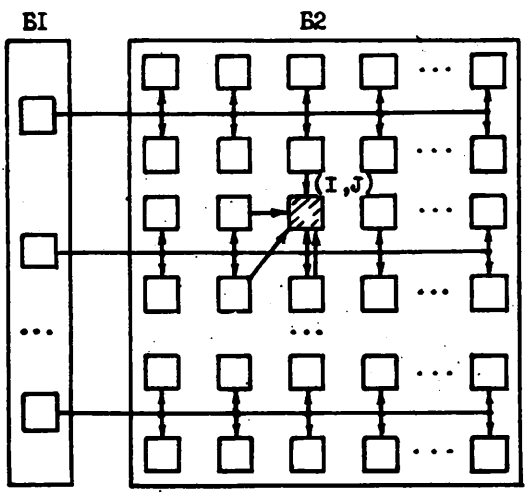



Рис. 2

Ясно, что в композиции с основным массивом может находиться несколько контекстных. Координаты соседей по входам в контекстном массиве по отношению к элементам основного массива задаются аналогично тому, как это сделано в п.3.3. Для примера (рис. 1) имеем PAT3 = (0).

Координаты соседей в разных клеточных массивах могут задаваться процедурами разного типа. Например, на рис. 2 соседи по входам в TS2 любого элемента (I,J) из TS2 , сопоставленного Б2, задаются шаблоном PAT1 (см. п.2.4). Но для

элемента (I,J) в TS2 сосед по входам из TS1 (БI) не может быть задан шаблоном и задается некоторой процедурой FUN I, состоящей в вычислении функции

$$y = \begin{cases} \left[\frac{I}{2}\right] + 1, & \text{если } \frac{I}{2} \text{ - дробь,} \\ \left[\frac{I}{2}\right], & \text{если } \frac{I}{2} \text{ - целое.} \end{cases}$$

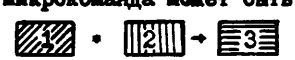
Нумерация входов и выходов ячеек в блоках (см. рис.I) порождает в соответствующих клеточных массивах нумерацию соседей, согласованную с порядком получения координат соседей в процедурах. Так, для элемента (I,J) из TS2 для процедуры PAT1 номера сосе - дей по входам в TS2 - I,2,3,4,5; для процедуры PAT2 номера соседей по выходам в TS2 - I,2,3; для процедуры PAT3 номер соседа по входу в TS1 - I.

Отметим, что процедуры вычисления соседей не связаны постоянно с клеточным массивом. Каждый массив может использоваться со всем набором имеющихся процедур. Информация о том, какие массивы образуют композицию и какие процедуры для них используются, задается в микрокоманде подстановки перечислением имен массивов и процедур.

2.6. М и к р о к о м а н д ы. В языке имеется два типа микрокоманд: микрокоманды подстановки и микрокоманды управления.

2.6.I. Микрокоманда подстановки определяет изменение состояний элементов основного клеточного массива и является аналогом подстановки (см. п.I).

Схематично микрокоманда может быть изображена как




Она состоит из трех зон: контекстной - I, базовой - 2 и операционной - 3. В зависимости от наличия зон микрокоманды подразделяются на следующие виды: полная (содержит все зоны), бесконтекстная (не содержит зоны I и • ), контекстная (не содержит зоны 2), с пустой левой частью (не содержит зон I и 2).

Кроме того, к операционной зоне микрокоманды любого вида может быть присоединен список признаков, который будет описан в п.2.6.2.

В записи микрокоманды может быть не одна зона вида  • , а несколько. Их максимальное число равно максимальному числу контекстных массивов для некоторого основного.

Если требуется указывать в записи зоны состояния не всех соседей, задаваемых процедурой, а только некоторой их части, в эту запись, наряду с именем процедуры (PAT или FUN), вводится перечень пар, каждая из которых содержит состояние элемента и его номер. Так, например, преобразованиям информации а) и б) в устройстве (рис.I) могут соответствовать микрокоманды:

а) TS1(I)PAT3(0,1)•TS2(I,J)PAT1(1,1)(0,2)(1,3)(0,4) →

TS2(I,J)PAT2(0,1)(1,2);

б) TS1(I)PAT3(1,1)•TS2(I,J)PAT1(0,1)(1,3)(0,5) →

TS2(I,J)PAT2(0,1)(1,3).

Для операционной зоны микрокоманды существует еще один способ записи, при котором вместо состояния элемента указывается имя функционального преобразования, состоящее из служебного слова MAP и целого положительного индекса. После имени преобразования в круглых скобках указываются имена клеточного массива и шаблона, а также номера элементов, состояния которых являются аргументами преобразования.

Для примера (рис.I) преобразуем работу ячейки однородного устройства так, что при тех же состояниях входов, что и в преобразовании а), она переходит в состояние 0 и записывает по входу 2 в соседнюю справа ячейку результат сложения по mod 2 состояний входов I,2. Обозначим через MAP1 функцию сложения по mod 2 двух двоичных переменных. Тогда микрокоманда, описывающая преобразование, получит вид:

TS1(I)PAT3(0,1) • TS2(I,J)PAT1(1,3)(0,4) →

TS2(I,J)PAT2(0,1)MAP1(TS2(I,J)PAT1(1,2)),2).

Если состояние элемента представляется символьным кодом в базовом алфавите, то в микрокоманде может указываться состояние отдельного разряда или группы разрядов кода. В этом случае элемент перечня пар может иметь, например, вид: (I,2(3)) – единица в третьем разряде второго соседа или (I0I,3(I+3)) – I0I в первых трех разрядах третьего соседа.

2.6.2. Для управления процессом применения микрокоманды служит с п и с о к п р и з н а к о в – заключительная часть микрокоманды.

А. Признак записи W. Параллельное применение микрокоманд микропрограммы к ячейкам клеточных массивов имитируется в модели-

лирующей ЦВМ при помощи клеточных массивов - двойников, которые вводятся на этапе перевода микрокоманд в процедуры языка реализации без ведома программиста. Каждому основному массиву TSi соответствует клеточный массив - двойник TS'i . Микрокоманда подстановки выполняется потактно. Такт состоит из двух шагов: сравнения и замены. Для каждого набора координат массива TSi производится сравнение состояний элементов из левой части команды с состояниями элементов - соседей по входам основного TSi и контекстных (если они есть) массивов. Если сравнение успешно, то выполняется шаг замены состояний элементов в TS'i : состояния соседей по выходам заменяются состояниями, извлекаемыми из правой части микрокоманды. Указание о том, что после применения данной микрокоманды ко всем элементам массива необходимо выполнить перепись информации из TS'i в TSi , дается признаком W .

Б. Признак применимости L . Основной способ остановки процесса моделирования состоит в проверке неприменимости выделенной группы микрокоманд на очередном шаге. Это проявляется в том, что в соответствующие массивы-двойники не производится ни одной записи символов. К микрокомандам, для которых производится проверка применимости, приписывается к правой части символ L , за которым в круглых скобках следуют имя клеточного массива, координаты элемента и его состояние, которое должно там возникнуть, если микрокоманда неприменима. Например, признак L(TS3(2,3), A) указывает, что в случае неприменимости микрокоманды в элемент (2,3) массива TS3 записывается символ A .

В. Признак повторения обозначается символом R и целым положительным числом. Число указывает, сколько раз нужно применить микрокоманду к клеточному массиву для получения результата моделирования. Это - второй способ остановки процесса моделирования.

2.6.3. Управляющие микрокоманды условного и безусловного перехода служат для более рациональной организации процесса моделирования, т.е. для обеспечения возможности имитации на каждом такте параллельного применения не всех микрокоманд микропрограммы, а только некоторой их части.

2.7. М и к р о п р о г р а м м а. Основными ее фрагментами являются циклический блок и одиночная микрокоманда.

Циклический блок состоит из имени (метки), заголовка, списка непомеченных микрокоманд подстановок и микрокоманды END с именем блока. Новая микрокоманда END играет роль правой скобки, роль ле-

вой скобки играет заголовок блока. Заголовок начинается служебным словом TITLE и содержит: перечисление клеточных множеств с указанием индексов пробега, граничные значения индексов пробега, имена процедур вычисления соседей, некоторые или все элементы списка признаков микрокоманды. Указание признака в заголовке означает, что этот признак один и тот же для всех микрокоманд блока. Признак R указывается только в заголовке. Кроме того, следует учитывать, что если клеточные множества указаны только в заголовке, то порядок их перечисления важен, так как в записи микрокоманд указаний на клеточные множества уже не будет, в каждой микрокоманде он будет таким, каким записан в заголовке.

Циклические блоки с заголовками вида:

M1: TITLE...;L(...); W;

M1: TITLE...;R200; W;

M1: TITLE...;R200;L(...); W

имитируют полностью параллельное выполнение всех преобразований, реализуемых в однородном параллельном вычислительном устройстве. Действительно, пусть основным клеточным массивом является TSi , его двойником - TS'i , тогда перепись информации из TS'i в TSi производится только после однократного преобразования массива в с е м и микрокомандами блока. Такое преобразование соответствует одной итерации в алгоритмах параллельных подстановок. Выход из цикла в блоке с заголовком вида "а" происходит по неприменимости, с заголовком вида "б" - после двухсот повторений всех микрокоманд, с заголовком вида "в" - после двухсот повторений, если оказалось, что микрокоманды применимы более двухсот раз.

Можно привести варианты циклических блоков, в которых признаки W и L не стоят в заголовке, а приписаны к отдельным микрокомандам. Например, пусть заголовок блока имеет вид M1:TITLE...;L(...); а i-я, j-я и последняя n-я микрокоманды в списке содержат W . Такой блок имитирует последовательную работу трех параллельных групп преобразований, представленных микрокомандами:

$$П_1;\ldots;П_i;$$

$$П_{i+1};\ldots;П_j;$$

$$П_{j+1};\ldots,П_n;\ .$$

Выход из цикла происходит по неприменимости в с е х микрокоманд. Любая микропрограмма состоит из заголовка, последовательно-

сти циклических блоков и отдельных микрокоманд. Заголовок записывается таким же образом, как и для циклического блока, но дополнительно содержит сведения о мощностях всех клеточных массивов. Микропрограмма заканчивается микрокомандой END .

## §3. Микропрограммная модель параллельного сумматора

В [I4] предложен параллельный сумматор, содержащий m рядов по n двоичных ячеек каждый. Этот сумматор может осуществлять параллельное сложение группы из m чисел, если в отдельный ряд сумматора записывается число. Кроме того, сумматор при помощи настроечных входов может быть разбит на несколько зон, в каждой из которых может осуществляться сложение группы чисел. Алгоритм сложения, реализуемый сумматором, состоит в преобразовании прямоугольной, бинарной таблицы, строки которой являются суммируемыми числами, по следующим правилам. В каждом такте преобразования о д н о в р е м е н н о во всех конфигурациях вида:

I)

| | |
|---|---|
| | I |
| 0 | I |
| 0 | 0 |

2)

| |
|---|
| 0 |
| I |
| 0 |

подконфигурация

| | |
|---|---|
| | I |
| 0 | I |

из конфигурации (I) заменяется на конфигурацию

| | |
|---|---|
| | 0 |
| I | 0 |

,

а подконфигурация .

| |
|---|
| 0 |
| I |

из конфигурации (2) заменяется на

| |
|---|
| I |
| 0 |

и т.д., пока такие преобразования возможны. Когда такие преобразования невозможны (т.е. в бинарной таблице нет ни одной конфигура-

ции вида (I) и (2)), это значит, что сумма вычислена. Константы, подаваемые на свободные входы крайних ячеек сумматора, связи между логическими элементами в каждой ячейке сумматора и связи между ячейками сумматора подобраны так, что сумматор выполняет алгоритм сложения, описанный выше. Некоторая ячейка (I,J) сумматора и ее соседи показаны на рис. 3.

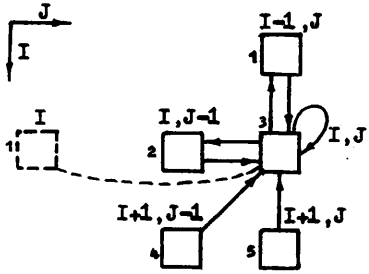



Рис. 3

Сумматору ставится в соответствие два клеточных массива: основной TS1 - двумерный массив с размерами (m+1)x n, элементами которого являются коды состояний памяти ячейки, и контекстный TS2 - одномерный массив, m+1 элементами которого являются коды настройки.

Согласно рис.3, в качестве шаблона для TS1 , отражающего соседство по входам и выходам разрядов сумматора, может быть выбран PAT1 = = (-1,0)(0,-1)(0,0)(1,-1)(1,0). Будем считать, что настройка сумматора производится при помощи настроечного регистра; причем у каждой ячейки некоторого ряда I есть сосед с координатами I в этом регистре (на рис.3 такой сосед показан штриховым контуром). Это соседство в TS2 отражается шаблоном PAT2 = (0).

Таблица 2

| J / I | TS2 | TS1 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 'I' | '0' | '0' | '0' | 'I' | '0' | 'I' | '0' |
| | 'I' | '0' | '0' | '0' | '0' | 'I' | '0' | 'I' |
| | 'I' | '0' | '0' | '0' | 'I' | 'I' | '0' | 'I' |
| | '0' | '0' | '0' | '0' | '0' | '0' | '0' | '0' |
| | '0' | '0' | '0' | '0' | '0' | 'I' | 'I' | '0' |
| | 'I' | '0' | '0' | '0' | 'I' | '0' | 'I' | 'I' |
| | 'I' | '0' | '0' | '0' | '0' | 'I' | 'I' | '0' |
| | 'I' | '0' | '0' | '0' | '0' | '0' | '0' | '0' |

В качестве основных символов алфавита массивов TS1 и TS2 примем символы {0,I}. В табл. 2 приведен пример записи исходного состояния массивов TS1 и TS2 для сумматора с размерами 7x7, разбитого на две зоны. Элементы с кодом '0' в TS1 отмечают последнюю строку первой зоны и первую - второй. Элементами нижней строки массива TS1 фиксируются состояния свободных входов ячеек сумматора.

Преобразованиям информации, реализуемым в алгоритме сложения, соответствуют две микрокоманды:

1) TS2(I)PAT2(1,1)•TS1(I,J)PATI(1,1)(0,2)(1,3)(0,4)(0,5) →
TS1(I,J)PATI(0,1)(1,2)(0,3);

2) TS2(I)PAT2(1,1)•TS1(I,J)PATI (0,1)(1,3)(0,5) →
TS1(I,J) PATI(1,1)(0,3).

Запись микропрограммы выглядит следующим образом:

```
TITLE CON TS2(8), I(2:7), PAT2; BAS TS1(8,7), I(2:7),
      J(I:7), PATI; W; L(TS3,I);
   (1,1)•(1,1)(0,2)(1,3)(0,4)(0,5) → (0,1)(1,2)(0,3);
   (1,1)•(0,1)(1,3)(0,5) → (1,1)(0,3);
END.
```

ПРИМЕЧАНИЕ. В заголовке CON означает контекстный массив, BAS - базовый. После имени клеточного массива в скобках указывается его размеры. Пара (2:7) задает границы индекса I и т.п., TS3 - вспомогательный массив, в исходном состоянии в его элемент вписан код '0'.

## Заключение

В работе показано, как на основе алгоритмов параллельных подстановок строится язык микропрограммирования, обеспечивающий имитацию параллельного во времени и повсеместного в пространстве применения микрокоманд микропрограммы. По сравнению с алгоритмами параллельных подстановок в языке конкретизировано понятие клеточного множества, расширено понятие контекста, введены средства, сокращающие запись микропрограмм, учтены особенности моделирования параллельных структур на последовательной машине, это позволяет строить достаточно лаконичные микропрограммные описания однородных параллельных вычислительных устройств.

## Литература


1. Автоматизация проектирования вычислительных систем. Языки, моделирование и базы данных. Под ред. М.Брейера. -М.: Мир, 1979.- 463 с.

2. Языки программирования. Под ред. Ф.Женюи. - М.: Мир, 1972.- 406 с.

3. ГОРБАТЕНКО Д.Д. Языки описания оборудования вычислительных систем. -Зарубежная радиоэлектроника, 1976, №10, с.15-22.

4. ЧУ Я. Организация ЭВМ и микропрограммирование. -М.: Мир.- 1975.- 592 с.

5. БАРАНОВ С.И., МАРИН А.В. Языки микропрограммирования. - Зарубежная радиоэлектроника, 1977, №6, с.85-102.

6. СЕРГЕЕВ С.Н. Однородный процессор для обработки сигналов. - В кн.: Вопросы теории и построения вычислительных систем (Вычислительные системы, вып. 70), Новосибирск, 1977, с.144-155.

7. YAU S.S., TUNG H.S. Associative Processor Architecture. A survey.- Computing Surveys,1977, v.9, N 1,p.3-28.

8. ФЕТ Я.И. Массовая обработки информации в специализированных однородных процессорах. - Новосибирск: Наука, 1976.-198 с.

9. ЕВРЕИНОВ Э.В., ПРАНГИШВИЛИ И.В. Цифровые автоматы с настраиваемой структурой. -М.: Энергия, 1974.- 239 с.

10. LIPOVSKI C.S. On a Varistructured Array of Microprocessors.- IEEE Trans.on Computers,1977,v.C-26,N 2,p.125-138.

11. JUMP J.R. Asynchronous Control Arrays.- IEEE Trans.on Computers, 1974,v.C-23, N 10,p.1020-1029.

12. БАНДМАН О.Л., ПИСКУНОВ С.В., СЕРГЕЕВ С.Н. Задачи параллельного микропрограммирования.-В кн.: Вопросы теории и построения вычислительных систем. (Вычислительные системы, вып. 73). Новосибирск, 1978, с.3-24.

13. КОРНЕВ Ю.Н., ПИСКУНОВ С.В., СЕРГЕЕВ С.Н. Алгоритмы обобщенных подстановок и их интерпретация сетями автоматов и однородными машинами. - Изв. АН СССР, Техническая кибернетика, 1971, № 6, с.131-142.

14. КОРНЕВ Ю.Н., ПИСКУНОВ С.В., СЕРГЕЕВ С.Н. Двоичный сумматор, Авт.св. № 436350, - Бюллетень изобретений, 1974, №26.

Поступила в ред.-изд.отд.
4 апреля 1980 года