ВОПРОСЫ ТЕОРИИ И ПОСТРОЕНИЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ
(Вычислительные системы)
1978 год Выпуск 73

УДК 681.325.5

РЕАЛИЗАЦИЯ АЛГОРИТМОВ ПАРАЛЛЕЛЬНЫХ ПОДСТАНОВОК В МИКРОПРОЦЕССОРНЫХ СИСТЕМАХ

С.Н.Сергеев

В работе рассматривается вопрос использования алгоритмов параллельных подстановок для организации массовых вычислений в параллельных микропроцессорных системах. Оперативные межмодульные взаимодействия в таких системах являются одной из серьезных проблем, не имеющей в настоящее время удовлетворительного решения. Причиной этого, на наш взгляд, являются два обстоятельства: во-первых, относительная новизна проблемы (первые сообщения о разработке больших однородных вычислительных систем на базе микропроцессоров появились в 1975 г. [1]); и во-вторых, то, что при решении этой проблемы обычно пытаются перенести на микропроцессорные системы традиционные методы межмашинных взаимодействий, разработанные для вычислительных систем на основе больших, средних или мини-машин, которые требуют для своей реализации достаточно развитых операционных систем, что для систем из микропроцессоров не всегда осуществимо.

Более близкой моделью для такого рода вычислительных структур является клеточный автомат - однородная совокупность элементарных автоматов (клеток), одинаковым образом связанных между собой и одинаково функционирующих. Важной особенностью клеточного автомата является локальность переработки информации: каждый автомат обменивается и обрабатывает информацию, получаемую от небольшого числа соседних. Принимая клеточный автомат в качестве модели интересующего нас класса структур, мы должны выбрать и подходящее алгоритмическое средство для описания поведения клеточного автомата. В данной работе для этой цели используются алгоритмы параллельных подстановок [2].

Если состояние сети автоматов в некоторый момент времени представить как совокупность состояний клеток, рассматриваемую в соответствии со структурой связей автоматов в сети, то все последующие состояния клеточного автомата можно представить как результат воздействия на эту совокупность некоторых операторов подстановки. Общее определение алгоритмов параллельных подстановок дано в работах [2,3]. Ниже дается определение одного класса параллельных подстановок и показывается, как с их помощью можно записывать алгоритмы переработки информации для систем из микропроцессоров.

I. А л г о р и т м ы п а р а л л е л ь н ы х п о д с т а н о в о к. В данной работе используется некоторая разновидность алгоритмов подстановок, использующая функциональные возможности микропроцессоров – алгоритмы функциональных подстановок. Так же, как и в [2,3], микрокоманда подстановки для данного класса алгоритмов есть выражение вида $S_1 * S_2 \to S_3$, где S_1, S_2, S_3 – конфигурация. Кроме того, на множестве внутренних состояний A задается некоторое множество операций $f_1, f_2, \ldots, f_k$, и в записи первых проекций правых частей подстановок допускается использование данных операций над операндами из первых проекций левых частей тех же подстановок. Например, если среди операций f_i есть операция "+" (сложение), то при построении алгоритмов можно использовать подстановки вида

$$(a,i) * (s,i+1) \to ((a+s),i). \qquad (I)$$

Ясно, что при этом первые проекции конфигураций будут в общем случае записываться в некотором расширении алфавита A – $\tilde{A}$, полученном добавлением к A вспомогательных символов для обозначения символов операций, символов переменных (s – в примере) и др. Внутренний алфавит (алфавит состояний) клеток клеточных множеств при этом не изменяется и остается A. Во всех остальных компонентах определение алгоритмов функциональных подстановок совпадает с определением, данным в [2,3].

Функциональная подстановка является естественным расширением определений, данных в работах [2,3], делающим запись алгоритмов более компактной, а реализацию – эффективной. Эффективность реализации достигается тем, что функции, используемые в подстановках, могут быть выполнены за один такт.

Прежде чем переходить к вопросам интерпретации, остановимся на одной проблеме, возникающей в связи с использованием функциональных подстановок – проблеме непротиворечивости. Для систем подстановок обычного вида этот вопрос подробно исследован в работах [2,4], где даны необходимые и достаточные условия непротиворечивости данной системы подстановок. Для систем с функциональными подстановками этот вопрос не исследован, хотя всегда есть путь прямого полного перебора всех возможных для данной функции вариантов правых и левых частей и отыскания возможных противоречий в получающихся системах подстановок. Но этот путь даже для несложных систем подстановок может привести к необозримо большому числу вариантов и потому практически не реализуем. Учитывая это обстоятельство, мы в данной работе будем использовать только такие функциональные подстановки и такие системы подстановок, непротиворечивость которых можно установить известными способами. Например, алгоритм, состоящий из одной подстановки (I), непротиворечив, так как каждая клетка меняет только собственное состояние и никакая другая клетка изменить его не сможет. Следовательно, ситуация, когда в некоторую клетку должны быть записаны одновременно два разных символа, не возникает.

2. И н т е р п р е т а ц и я о п е р а ц и и п о д с т а н о в к и м и к р о п р о ц е с с о р о м. Вопросы интерпретации алгоритмов параллельных подстановок сетями автоматов подробно рассмотрены в [2].

Говоря о микропроцессорной интерпретации подстановок, нужно сразу же отметить те ограничения, которые здесь возникают.

Во-первых, каждый микропроцессор – это вполне реальный, физически реализованный конечный автомат, с вполне определенными функциями переходов и множеством входных и выходных переменных; во-вторых, множества входных и выходных полюсов микропроцессора задают множество структур связей в сети; в-третьих, каждый микропроцессор является универсальным вычислителем последовательного действия и потому способен выполнить любой алгоритм подстановок, но с растяжкой во времени. Поэтому более целесообразно говорить о вложении алгоритмов подстановок в некоторый заданный класс однородных структур из микропроцессоров, а не о построении сети автоматов, интерпретирующей данный алгоритм.

Пусть задана некоторая однородная структура из микропроцессоров. Для определенности положим, что элемент этой структуры имеет n k-разрядных элементов памяти.

Варианты и размеры клеточных множеств, представляемых в данной структуре, определяются тем, какой элемент памяти микропроцессора мы принимаем в качестве аналога "клетки". Таким элементом может быть как один разряд регистра, так и совокупность из n k-разрядных регистров. Соответственно этому мощность α клеточных множеств, представленных в данной структуре, будет находиться в диапазоне $N \le \alpha \le knN$, где N – общее число микропроцессоров. В рамках данных ограничений на мощность мы можем в принципе представить клеточные множества любой структуры. Это может быть, напри-

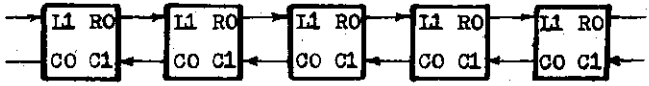

Рис. I

мер, одномерный массив клеток с именами $i \in [1,2,\ldots,\alpha]$ или двумерный массив клеток с именами (i,j), $i \in (1,\ldots,\alpha_1)$, $j \in (1,\ldots,\alpha_2)$, $\alpha_1 \cdot \alpha_2 = \alpha$ и т.д.

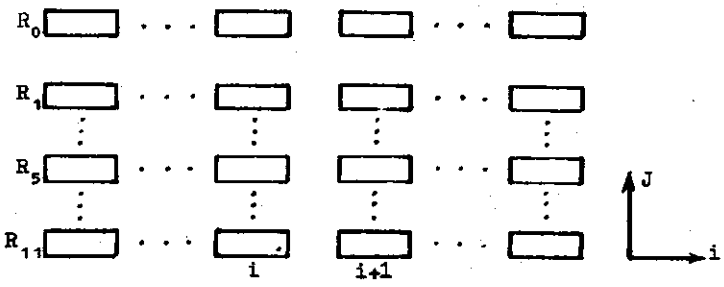

Рис. 2

Рассмотрим, например, одномерную цепочку процессорных элементов серии Intel-3000 *), представленную на рис.I. Примем, что каждый из регистров может представлять клетку клеточного множества.

*) Процессорный элемент этой серии представляет собой двухразрядную процессорную секцию, содержит I3 (двухразрядных) регистров и выполняет 256 различных микроопераций, имеет три группы (I, M, K) входов и две группы (A, D) выходов, входы и выходы для организации сдвиговых операций и передачи переносов в арифметических операциях [5].

Тогда в данной структуре можно представить, например, двухмерный массив размерностью $[\alpha_1,\alpha_2]$, $\alpha_1 \in (1,\ldots, N)$, $\alpha_2 \in [0,1,\ldots,11]$ (рис.2) или одномерный массив, составленный, скажем, из всех регистров R_0 всех микропроцессоров.

Выбор "аналога" клетки в микропроцессоре в некоторой степени определяет и класс допустимых конфигураций. "Аналог" однозначно определяет алфавит, в котором могут быть записаны конфигурации. Скажем, конфигурация {(0,(i,5)), (I,(i+I,5))} допустима для клеточного множества рис.2, а конфигурация (0,(i,I00)) или (5,(i, I)) недопустимы.

Второй вопрос, возникающий в связи с вложением алгоритмов функциональных подстановок в сеть из микропроцессоров, - это интерпретация подстановки. Очевидно, что конкретная реализация подстановки целиком определяется выбранным типом микропроцессора и его системой микроопераций. Тем не менее существуют общие обязательные этапы, которые всегда необходимо выполнить в каждой клетке:

Этап I. Сбор информации о состоянии соседних клеток. Соседство клеток определяется второй проекцией конфигурации левой части.

Этап 2. Определение совпадения информации о соседях со словом состояний, указанным в первой проекции.

Этап 3. Если этап 2 был удачным, то записать информацию, указанную в первой проекции конфигурации правой части, в клетки, определяемые второй проекцией. Например, для подстановки

Таблица I

Левая часть уравнения	Микропрограмма
ab	CDA CDA M ⊕ AC → AC M ⊕ T → T AC ∨ T → AC C1 ∨ AC → CO
(a,R_1)(b,R_2)	R_1 → AC CDL R_2 → AC CDL M ⊕ AC → AC M ⊕ T → T AC ∨ T → AC C1 ∨ AC → CO

Таблица 2

Правая часть уравнения	Микропрограмма
cd	M → AC M → T CDL AC CDL T
f(a,b)	Микрокоманда с именем f

$$\{(a,i)(b,i+1)\} \to \{c,(i-1)\}$$

каждая клетка i будет получать информацию о состоянии соседней справа (i+I)-й клетки, и в случае совпадения их совместного состояния со словом ab запишем в соседнюю слева (i-I)-ю клетку символ c . В таблицах I и 2 приведены микропрограммы выполнения этапов I-3 микропроцессором Intel 3000 для подстановок в сети рис.I.

При написании этих микропрограмм предполагалось, что первые проекции подстановок поступают на входы M каждого процессорного элемента. В качестве r(s,t) может быть указан код любой из бинарных операций из системы микроопераций микропроцессора либо имя микропрограммы, имеющейся в памяти микропроцессора.

3. Примеры организации массовых вычислений на однородных микропроцессорных структурах с использованием подстановок в качестве простейшего макрооператора.

ПРИМЕР I. Вычисление матричного преобразования c = Ab . Преобразование задается матрицей

$$A = \begin{vmatrix} a_{11} \cdots a_{1n} \\ \\ a_{n1} \cdots a_{nn} \end{vmatrix},$$

преобразуемая информация - вектором $b = (b_1,\ldots,b_n)$.

В качестве клеточного множества возьмем трехмерный массив $\hat{A}$ размером n x n x 3. Элементу матрицы a_{ij} сопоставим клетку $\hat{A}(i,j,0)$. Клетки A(i,j,1) и A(i,j,2) предназначены для хранения промежуточного значения i-го скалярного произведения и j-й компоненты вектора b соответственно. Алгоритм вычисления вектора b·A имеет вид:

$$\begin{aligned} &\{(t,(i,j,2))(\overline{s},(i-1,j,2))\} \to \{(\overline{s},(i,j,2))(s,(i-1,j,2))\} \\ &(s,(i,j,1))*\{(t,(i,j,0))(\overline{r},(i,j,2))\} \to (s\cdot r,(i,j,1)) \\ &\{(s,(i,j,1))(\overline{t},(i,j-1,1))\}*(\overline{r},(i,j,2)) \to \\ &\to \{(\overline{s+t},(i,j,1))(0,(i,j-1,1))\} \ . \end{aligned} \qquad (2)$$

Символы s,t,r обозначают произвольный символ алфавита состояний. Считается, что значением произвольного символа является содержимое памяти указанного в данной конфигурации элемента и за этим значением в момент применения данной подстановки закреплено имя указанного произвольного символа, т.е. содержимое клетки (i-1,j,2) в первой подстановке имеет имя s . Значение произвольного символа

сохраняется только в пределах данной подстановки. Каждый символ алфавита состояний может иметь или не иметь признак "~" (надчеркивание). Этот признак введен для синхронизации процесса вычислений. Наличие этого признака в клетке говорит о том, что информация в ней готова для использования. Для тех клеток, состояние которых в процессе не меняется, например, клетки с именами (i,j,0), использование признака не обязательно. На рис.3 представлена сеть

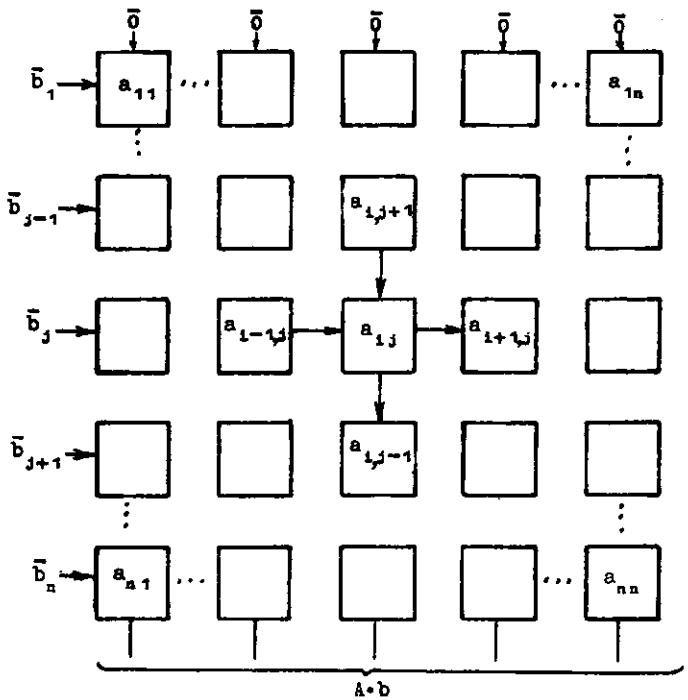

Рис. 3

из микропроцессоров, построенная в соответствии со вторыми проекциями конфигурации алгоритма (2). Микропроцессор с координатами (i,j) содержит элементы клеточного множества с именами (i,j,0), (i, j, 1),(i, j, 2).Отметим одну особенность микропроцессорной реализации алгоритмов подстановок. В связи с тем, что микропроцессор в каждый момент времени может выполнять только одну микрооперацию, подстановки алгоритма выполняются последовательно в том порядке,как они указаны в алгоритме.Таким образом,для алгоритма (2)

вначале будет выполнена первая подстановка, затем вторая, третья, затем опять первая и т.д. до неприменимости всех подстановок.

Для всех алгоритмов, приведенных далее, порядок выполнения такой же.

ПРИМЕР 2. Вычисление преобразования Адамара. Преобразование Адамара является частным случаем матричного преобразования, поэтому алгоритм из примера I может быть применен и для этого случая. Однако мы здесь приведем другой алгоритм, учитывающий свойства матрицы Адамара. Предположим, что требуется выполнить преобразование над двоичным вектором длины 2^n, тогда нетрудно получить выражение для i-го коэффициента разложения. Оно имеет вид:

$$a_i = (-1)^{(W_0, i)} + (-1)^{(W_1, i)} + \ldots + (-1)^{(W_k, i)}, \qquad (3)$$

где (α, β) – скалярное произведение векторов α, β; $W_0, \ldots, W_k$ – двоичные номера ненулевых координат исходного вектора, i – двоичная запись номера вычисляемого коэффициента. Используя выражение (3), можно отказаться от хранения матрицы преобразования и предложить следующую схему вычислений. Предположим, что мы имеем 2^n элементарных вычислителей, расположенных так, как показано на рис.4. Первый слева вычислитель является входным. На его вход поступает

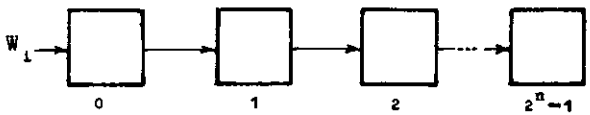

Рис. 4

последовательность номеров ненулевых компонент преобразуемого вектора. Для определенности будем считать, что вычислителями являются блоки центрального процессорного элемента из серии Intel 3000. Клеточное множество для данной задачи имеет вид рис.2. Соответствие элементов клеточного множества и хранимой в них информации для одного элемента цепочки рис.4 приведено на рис.5, где σ_k – значение суммы (3) после подачи i-го номера ненулевой компоненты; w_{k+1} – номер следующей ненулевой компоненты, i – двоичное представление номера процессорного элемента, совпадающего с номером вычисляемого коэффициента. Организация вычислений такова: каждый элементарный вычислитель (микропроцессор) вычисляет свой коэффи –

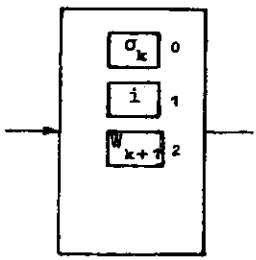

Рис. 5

циент преобразования, номер коэффициента совпадает с номером микропроцессора в цепочке. Поскольку номера ненулевых коэффициентов поступают на крайний слева вычислитель и распространяются затем по цепочке, каждый микропроцессор должен на очередном шаге вычислений, во-первых, получить от соседнего слева вычислителя очередной номер W_{k+1} и, во-вторых, – вычислить скалярное произведение (W_{k+1},i) и добавить +I или –I к сумме σ_k. В подстановках этот алгоритм запишется так:

$$(s,(i,2))*(t,(i-1,2)) \to (t,(i,2))$$

$$(s,(i,3))*\{(t,(i,1))(r,(i,2))\} \to (f_1(t,r),(i,3))$$

$$(s,(i,0))*(0,(i,3)) \to (s+1,(i,0))$$

$$(s,(i,0))*(1,(i,3)) \to (s-1,(i,0)) \ .$$

Функция $f_1(t,r)$ есть имя микропрограммы, вычисляющей скалярное произведение. В данном случае это произведение вычисляется логическим умножением векторов t и r и сложением по mod 2 компонент произведения.

Нетрудно видеть, что итеративная передача исходных данных по цепочке и учет только ненулевых компонент преобразуемого вектора дают оценку времени на выполнение данного преобразования, равную $k+2^n$, где k – число ненулевых компонент. Не усложняя алгоритма и не меняя общего числа микропроцессоров, можно простым соединением их в древовидную структуру с коэффициентом ветвления 2 получить время преобразования, равное $k+n$.

ПРИМЕР 3. Цифровая фильтрация. В работе [6] автором был рассмотрен вопрос построения специализированного однородного процессора, ориентированного на цифровую обработку сигналов. Было показано, что, используя алгоритмы параллельных подстановок и способы их аппаратной реализации, можно достаточно просто решить задачу проектирования специализированного устройства и обеспечения его соответствующим набором микропрограмм. Предложенный в работе [6] однородный процессор имеет структуру, представленную на рис.6. Основу процессора составляет однородный массив вычислительных блоков I. Элемент этого массива – вычислительный модуль – представ –

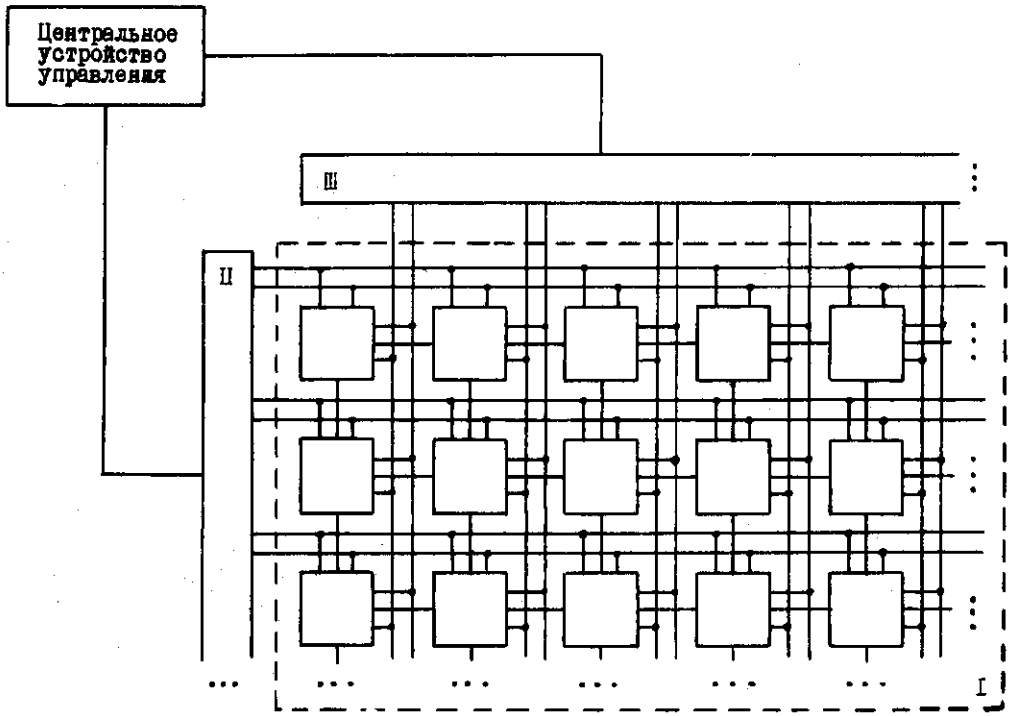

Рис. 6

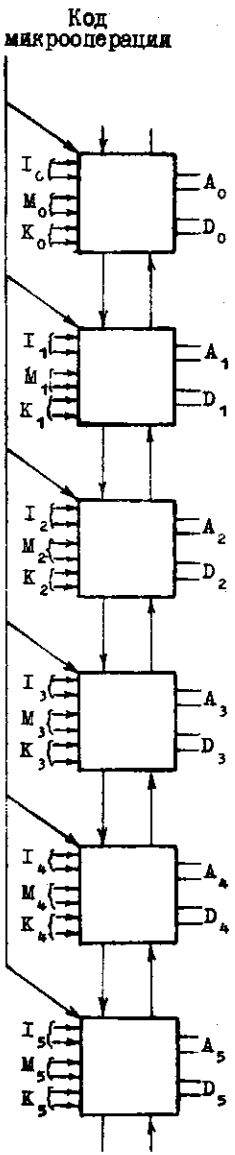

Рис. 7

ляет, в свою очередь, тоже однородную совокупность из ячеек, реализующих микрокоманды подстановок. Здесь мы рассмотрим реализацию вычислительного модуля на основе блоков центрального процессорного элемента серии Intel 3000. Структура процессора та же, что на рис.6, только вместо вычислительных модулей в массиве I находятся блоки, построенные из центральных процессорных элементов. Каждый блок построен из нескольких модулей элементов, соединенных по схеме рис.7. Число модулей в блоке определяется разрядностью чисел и величиной окрестности, принятой для данного класса задач. Зависимость числа модулей от величины окрестности блока объясняется тем, что каждый модуль имеет небольшое число связей, для получения нужного числа связей отдельные блоки элементов соединяются по схеме рис.7 и обмен информацией с соседями происходит по "слоям", с последующим сдвигом информации по "вертикали".

Типичной операцией обработки сигналов является фильтрация. В работе [6] она сформулирована так. Для каждой из $N=N_1 \times N_2$ точек нужно производить вычисления итеративных сумм $U_k^{i,j}$ и $V_k^{i,j}$ по формулам

$$U_k^{i,j} = U_{k-1}^{i,j} + a_{jk} X_i - b_{jk} y_i,$$

$$V_k^{i,j} = V_{k-1}^{i,j} + b_{jk} X_i + a_{jk} y_i,$$

$i \in N_1$, $j \in N_2$, k-номер итерации,

до тех пор пока выполняется неравенство

$$(U_k^{ij})^2 + (V_k^{ij})^2 - h^2 < 0 .$$

Величины $X_i, Y_i, a_{jk}, b_{jk}, -h^2$ для данного алгоритма являются входными. Каждой точке (i,j) сопоставлен свой вычислительный мо-

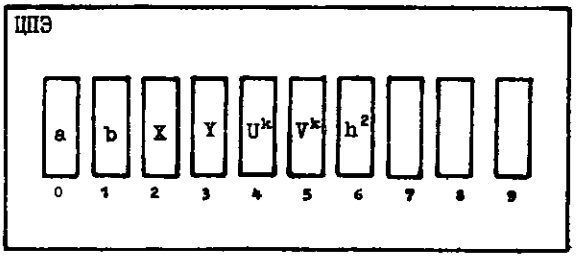

Рис. 8

дуль. Клеточное множество для данного алгоритма удобно представить как совокупность из $N_1 x N_2$ одномерных клеточных множеств - каждой точке (i,j) - свое клеточное множество. Распределение информации в каждом клеточном множестве перед очередной итерацией представлено на рис.8 (ЦПЭ - центральный процессорный элемент).Алгоритм вычисления выражений (5) и (6) имеет вид:

		Что вычисляется
1	$(s,7)*\{(t,0)(r,2)(0,9)\} \to (f_1(t,r),7)$	aX
2	$(s,8)*\{(t,0)(r,3)(0,9)\} \to (f_1(t,r),8)$	aY
3	$(s,4)*\{(t,7)(0,9)\} \to (f_2(s,t),4)$	$U^{k-1}+aX$
4	$(s,5)*\{(t,8)(0,9)\} \to (f_2(s,t),5)$	$V^{k-1}+aY$
5	$(s,7)*\{(t,1)(r,2)(0,9)\} \to (f_1(t,r),7)$	bX
6	$(s,8)*\{(t,1)(r,3)(0,9)\} \to (f_1(t,r),8)$	bY
7	$(s,4)*\{(t,7)(0,9)\} \to (f_2(s,t),4)$	U^k
8	$(s,5)*\{(t,8)(0,9)\} \to (f_2(s,t),5)$	V^k
9	$(s,7)*\{(t,4)(0,9)\} \to (f_1(t,t),7)$	$(U^k)^2$
10	$(s,8)*\{(t,5)(0,9)\} \to (f_1(t,t),8)$	$(V^k)^2$
11	$(s,7)*\{(t,8)(0,9)\} \to (f_2(s,t),7)$	$(U^k)^2 + (V^k)^2$
12	$(s,7)*\{(t,6)(0,9)\} \to (f_2(s,t),7)$	$(U^k)^2 + (V^k)^2 - h^2$
13	$(s,9)*(-t,7) \to (1,9)$	Проверка неравенства и останов

В алгоритме функции f_1 и f_2 – микропрограммы умножения и сложения двух чисел с фиксированной запятой. Как видно из алгоритма, подстановки будут применяться в данной точке (i,j) до тех пор пока клетка с именем 9 будет иметь состояние 0.

Таблица 3

	Время решения	
	Среднее	Максимальное
Сеть из Intel 3000	288 τ	792 τ
Спец-процессор	110 τ	110 τ

В табл. 3 приведены оценки средних и максимальных времен выполнения алгоритма фильтрации для микропроцессорного варианта и варианта с аппаратной реализацией подстановок [6]. Оценки даны для типичной в обработке сигналов разрядности чисел – 12. Заметим, что величина τ в одном случае есть время выполнения одной микрооперации блоком центрального процессорного элемента серии Intel 3000, в другом – время выполнения одной микрооперации подстановки при ее аппаратной интерпретации.

Для сложных систем цифровой обработки сигналов величины N_1 и N_2 имеют значения нескольких сотен и десятков соответственно, а время, отводимое на один цикл обработки, составляет единицы миллисекунд. Поэтому использование традиционной архитектуры приводит к очень жестким временным требованиям не достижимым в настоящее время [7]. В то же время использование принципов параллельной обработки информации и серийных микропроцессоров позволяет решить эту задачу достаточно простым путем.

Заключение

В статье рассмотрен вопрос использования алгоритмов параллельных подстановок для организации межмашинных взаимодействий в микропроцессорных системах. Введен класс функциональных подстановок, позволяющий полнее использовать функциональные возможности микропроцессоров.

Рассмотренные в статье примеры позволяют сделать вывод о том, что использование оператора подстановки, в качестве, простейшего микрооператора для организации массовых вычислений на микроуровне, с одной стороны позволяет достаточно легко и компактно записывать алгоритмы, с другой – не требует построения сложной системы интерпретации.

Эти достоинства параллельных подстановок могли бы проявиться еще полней, если бы структура микропроцессора позволяла непосредственно интерпретировать подстановку как микрооперацию. Как можно заметить, в приведенных микропрограммах оператора подстановки функциональные возможности микропроцессора используются только при выполнении правой части подстановки, левая же часть интерпретируется с потерями времени, поэтому введение в логику микропроцессора операции сравнения двух кодов с выработкой признака сравнения, используемого затем для управления информационными входами микропроцессора, позволит существенно упростить микропрограммы реализации оператора подстановки.

Из рассмотренных примеров микропрограмм подстановок можно вывести некоторые требования к структурным и логическим характеристикам микропроцессоров для более эффективной реализации подстановок,

I. Возможно большее число внешних полюсов.Лучше всего в этом отношении приспособлены для реализации подстановок так называемые "bit-slice" микропроцессорные наборы [5]. Они обладают возможностью наращивания разрядности и это их свойство можно использовать для увеличения окрестности.

2. Система микрокоманд должна включать в себя микрокоманду сравнения двух кодов.

3. Центральный процессорный элемент должен содержать возможно большее число внутренних регистров. Это позволит получить более быструю реализацию подстановки.

Из известных микропроцессорных серий наиболее полно этим требованиям отвечают модели Intel 3000, SBPO 400.

Литература

I. SCRUPSKI S.E. Coming: Cheap, powerful computers.-"Electronics",1975,v.48,N 25,p.67-68.

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

3. БАНДМАН О.Л., ПИСКУНОВ С.В., СЕРГЕЕВ С.Н. Задачи параллельного микропрограммирования. -Настоящий сборник, с.

4. КОРНЕВ Ю.Н., ПИСКУНОВ С.В.,СЕРГЕЕВ С.Н. Алгоритмы обобщенных подстановок и вопросы их интерпретации. -"Техническая кибернетика". Труды семинара, вып. 4. Киев, 1970.

5. МАКАРЕВИЧ О.Б., СПИРИДОНОВ Б.Г., КРАВЧЕНКО П.П. Микро-ЭВМ как средство обработки информации. Обзор. -"Зарубежная электронная техника", вып. 9, 1977, с.18-24.

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. ХВОСТАНЦЕВ М.А. Микропроцессоры и системы обработки данных. -"Зарубежная радиоэлектроника", 1975, № 9, с.31-56.

Поступила в ред.-изд.отд.
8 февраля 1978 года