АРХИТЕКТУРА И МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ
(Вычислительные системы)
1984 год Выпуск 104

УДК 681.322.02

АСИНХРОННАЯ КОМПОЗИЦИЯ ПАРАЛЛЕЛЬНЫХ МИКРОПРОГРАММ

С.В. Пискунов

Введение

В статье продолжается разработка приемов асинхронной композиции параллельных микропрограмм, начатая в [1], и используются введенные там понятия параллельного микропрограммирования.

Объектом преобразования в параллельном микропрограммировании служит клеточное множество W – конечная совокупность клеток, поименованных символами из алфавита M. В клетки вписаны символы из некоторого конечного алфавита состояний A. Элементарным преобразованием клеточного множества W является микрокоманда подстановки θ, имеющая левую и правую части (θ: $S_1(m) * S_2(m) \to S_3(m)$; S_1, S_2, S_3 называются конфигурациями, $m \in M$). Выполнение микрокоманды производится одновременно над всеми клетками множества W. Для каждой клетки из W (обозначим ее имя m^0) в соответствии с левой частью микрокоманды вычисляется множество клеток $S_1(m^0) \cup S_2(m^0)$. Если полученное множество включено в W, его часть $S_1(m^0)$ заменяется множеством $S_3(m^0)$. Совокупность микрокоманд $\Phi = \{\theta_i : S_{i1}(m) * S_{i2}(m) \to S_{i3}(m),\ i = 1, \ldots, \nu\}$, записанных в произвольном порядке, называется микропрограммой. Выполнение всех микрокоманд микропрограммы производится одновременно над всеми клетками множества W. Один шаг выполнения называется итерацией. По каждой параллельной микропрограмме Φ и совокупности множеств W, каждое из которых имеет мощность не более заданной, может быть построена сеть автоматов, интерпретирующая Φ.

Операция композиции позволяет строить сложную микропрограмму из заданных более простых $\Phi_1, \ldots, \Phi_q$. Эта операция записывается на языке параллельных граф-схем. В [1] был предложен вариант вклю-

чения любой параллельной микропрограммы Φ в асинхронную композицию. Будем называть его далее переводом Φ в каноническую форму. Микропрограмма Φ перестраивается так, что она начинает взаимодействовать с другими микропрограммами по принципу "пуск-завершение". Для этого множество M расширяется: к нему добавляются имена m_1, m_2, m_3, m_4, а в алфавите A выделяются символы a_0; a_1. По микропрограмме $\Phi = \{\theta_i: S_{i1} * S_{i2} \to S_{i3},\ i = 1,\dots,\nu\}$ строится микропрограмма

$$\Phi' = \begin{cases} \theta_i^{\text{Я}}: \{(S_{i1})(a_0,m_2)\} * \{(S_{i2})(a_1,m_1)\} \to \\ \qquad \to \{(S_{i3})(a_1,m_2)\}, \quad i = 1,\dots,\nu; \\ \theta_{\nu+1}: \{(a_0,m_3)\} * \{(a_1,m_1)\} \to \{(a_1,m_3)\}; \\ \theta_{\nu+2}: \{(a_1,m_2)(a_1,m_3)\} * \{(a_1,m_1)\} \to \{(a_0,m_2)(a_0,m_3)\}; \\ \theta_{\nu+3}: \{(a_1,m_3)(a_0,m_4)(a_1,m_1)\} * \{(a_0,m_2)\} \to \\ \qquad \to \{(a_0,m_3)(a_1,m_4)(a_0,m_1)\}. \end{cases}$$

По сравнению с исходной Φ микропрограмма Φ' содержит на три микрокоманды больше, и, если микропрограммой Φ выполняется τ итераций, то Φ' - $2\tau + 2$ итерации, т.е. "накладные расходы", практически при том же числе микрокоманд, сводятся к двукратному увеличению времени выполнения. Кроме того, можно убедиться на примерах, что при переходе от Φ к Φ' у сети, интерпретирующей Φ', могут исчезнуть такие структурные свойства, как близкодействие [2] и ограниченность заданной величиной числа входов-выходов каждого из элементарных автоматов, составляющих сеть.

Поэтому представляется целесообразным разработать несколько вариантов канонических форм, позволяющих варьировать такими параметрами, как число микрокоманд в микропрограмме (программная сложность Q), число итераций (временная сложность T), и обеспечивающих, если необходимо, сохранность структуры сети исходной Φ.

I. Канонические формы параллельных микропрограмм с ограничениями на временную сложность

Удвоение числа итераций может быть устранено модификацией Φ':

$$\Phi^{*} = \begin{cases} \theta_i^{Я}, \ i = 1,\ldots,\nu \ ; \\ \theta_i'^{Я} : S_{i1} * \{(S_{i2})(a_1, m_1)\} \to S_{i3}, \ i = 1,\ldots,\nu \ ; \\ \theta_{\nu+1} ; \\ \theta_{\nu+2} ; \\ \theta_{\nu+3} ; \end{cases}$$

Действительно, в этом случае последовательность итераций имеет вид:

№ итераций	1	2	3	...	j	j+1	...	τ+1	τ+2	τ+3
Имена микрокоманд	$\theta_i^{Я}$		$\theta_i^{Я}$		$\theta_i^{Я}$					
	$\theta_i'^{Я}$	$\theta_i'^{Я}$	$\theta_i'^{Я}$	...	$\theta_i'^{Я}$	$\theta_i'^{Я}$	...	$\theta_{\nu+1}$	$\theta_{\nu+3}$	$\emptyset$
	$\theta_{\nu+1}$	$\theta_{\nu+2}$	$\theta_{\nu+1}$		$\theta_{\nu+1}$	$\theta_{\nu+2}$				

Общее число итераций равно $\tau + 2$. "Накладные расходы" сводятся к двукратному увеличению числа микрокоманд.

Удваивая не все микрокоманды исходной Φ, а только некоторую их часть, можно построить целый спектр канонических форм с различными Q и T, занимающий весь промежуток между Φ' и Φ^{*}.

2. Канонические формы параллельных микропрограмм с ограничениями на тип используемых конфигураций

В этом разделе пойдет речь о канонических формах, сохраняющих структуру сети для Φ. Введем классификацию конфигураций, отражающую структурные свойства сети автоматов. Определим отношение соседства на клетках множества W для любой конфигурации S с именующими функциями $m, \varphi_2(m), \ldots, \varphi_k(m)$. Возьмем любую клетку из W, пусть ее имя m^0. Клетки с именами $\varphi_2(m^0), \ldots, \varphi_k(m^0)$ являются соседями m^0. Отношение соседства будем изображать графически, связывая ребрами клетку m^0 с соседями. Будем называть о к р е с т н о с т ь ю любой клетки из W множество смежных с ней клеток. Клеточное множество W вместе с графическим изображением отношения соседства будем называть с е т ь ю к л е т о к W.

Очевидно, окрестность клетки m^0 взаимно-однозначно соответствует множеству автоматов, с которыми автомат m^0 сети имеет ин-

формационные связи в соответствии с именующими функциями конфигурации S .

Конфигурация S называется г л о б а л ь н о й, если для любого целого положительного n найдется W , в котором есть хотя бы одна клетка, окрестность которой, построенная по S , содержит больше n клеток.

Наличие хотя бы одной глобальной конфигурации в записи параллельной микропрограммы означает, что интерпретирующая ее сеть содержит автоматы, у которых число связей с другими автоматами растет с ростом числа автоматов в сети.

Конфигурация S , не являющаяся глобальной, называется о г р а н и ч е н н о й.

Если все конфигурации в записи параллельной микропрограммы ограничены, то в сети, независимо от числа клеток в W , каждый автомат имеет число связей, не превосходящее некоторое фиксированное число.

При реальном построении сети, интерпретирующей параллельную микропрограмму, нельзя не учитывать, что составляющие ее конечные автоматы превращаются в устройства и имеют определенные геометрические размеры. Это означает, что даже при использовании ограниченных конфигураций в записи микропрограммы длина информационных связей, соединяющих автоматы, может расти с ростом мощности W . В связи с этим для тех случаев,когда клеточное множество может быть представлено точками в некотором метрическом пространстве, представляет интерес говорить о локальных конфигурациях.

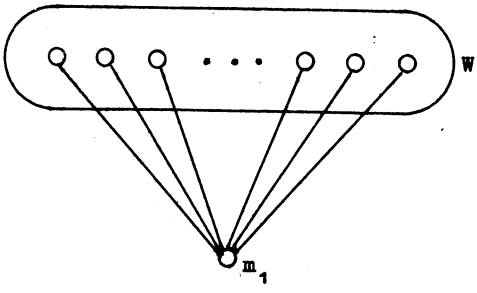

Рис.I

Конфигурация S называется л о к а л ь н о й, если для любого W найдется шар с фиксированным радиусом R , содержащий окрестность любой клетки из W.

Операция "пуск-завершение", которая фактически вводится в Ф при переходе к канонической форме, по своей сути является глобальной: сигнал "пуск" (переход клетки m_1 в состояние

a_1) дает разрешение выполняться всем микрокомандам во в с е х клетках множества W ,а "завершение" (переход клетки m_4 в состояние a_1) фиксирует,что ни одна микрокоманда неприменима н и к о д н о й клетке W . Это приводит к появлению глобальных конфигураций в каноническом представлении. Действительно, если хотя бы одна конфигурация, использованная в записи микрокоманд микропрограммы Ф , содержит бесконечное множество элементов,то с клеткой m_1 (m_2) смежны все клетки множества W (рис.I).

Приемы построения канонических форм, сохраняющих тип конфигураций, используемых в Ф , основаны:

а) в случае ограниченных конфигураций на переходе от сети клеток, представленной деревом (рис.I),к сети клеток, представленной деревом с заданной степенью вершин (рис.2);

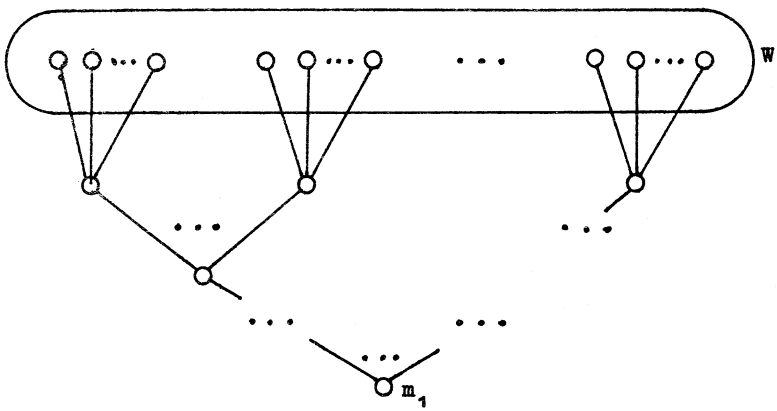

Рис.2

б) в случае локальных конфигураций на вводе вспомогательного клеточного множества, изоморфного W , с выделенной в нем пусковой клеткой (связанной с m_1 (m_2)), и осуществлении синхронизации состояний клеток вспомогательного множества по типу синхронизации цепи стрелков [3] с целью пуска (останова) микропрограммы.

I. Асинхронная композиция с использованием ограниченных конфигураций. С практической точки зрения наиболее интересен случай построения бинарных деревьев, потому что для них проведены иссле-

дования по оптимальной укладке на плоскости [4,5], что позволяет использовать их при реальном построении сетей автоматов в виде БИС, кроме того, для таких деревьев более простым получается канонический представление.

Обозначим буквой μ число клеток в W, буквой d – минимальную степень двойки, большую μ, буквой $\tilde{d}$ – число $(d-1)$. Из множества вспомогательных клеток m_1, m_2, m_3, m_4 исключим клетку m_2, добавим клетку m_Φ и два множества клеток: одно с именами $\{(m_1,m)\}$, другое с именами $\{(m_2,m)\}$, где $1 \le m \le \tilde{d}$. Клетки обоих множеств имеют состояния из множества $\{a_0, a_1, a_2\}$.

Если в записи конфигураций микрокоманд, преобразующих состояния клеток $\{(m_1,m)\}$, используются именующие функции (m_1, m), $(m_1, 2m)$, $(m_1, 2m+1)$, то сеть клеток с именами $\{(m_1, m)\}$ является бинарным деревом. В качестве корневой клетки выбирается клетка с именем $(m_1, 1)$, на втором ярусе – клетки с именами $(m_1,2),(m_1,3)$, на p-м ярусе – клетки с именами $(m_1,2^{p-1}),(m_1,2^{p-1}+1),\ldots$. Последний ярус заполняют клетки с именами (m_1, d), $(m_1, d+1)$, ..., $(m_1, d+\mu-1)$. Все клетки дерева, не лежащие на каком-то пути от корня дерева к вершине последнего яруса, отбрасываются.

Обозначим буквой Ψ_1 взаимно-однозначное соответствие имени каждой клетки последнего яруса и имени клетки множества W.

ПРИМЕР. Пусть имена клеток в W – числа $1,2,\ldots,\mu$; Ψ_1 может быть задана выражением $m \leftrightarrow (m_1, m+\tilde{d})$.

Аналогично можно построить дерево для клеток с именами $\{(m_2, m)\}$ и выбрать Ψ_2.

На рис.3 изображена сеть клеток W (с числом клеток равным 20), вспомогательные клетки и ребра, за исключением ребер, порожденных конфигурациями исходного алгоритма.

Построение канонического представления алгоритма Φ (обозначим Φ^0) проведем для клеточных множеств такого вида, как на рис.3. В исходном состоянии все вспомогательные клетки, кроме клеток с именами m_1 и m_Φ, находятся в состоянии a_0. Клетки m_1 и m_Φ находятся в состоянии a_1. В качестве прототипа для Φ^0 выберем Φ^*, полученную в п.I и имеющую наименьшую временную сложность. В общих чертах Φ^0 работает следующим образом. Сигнал "пуск" (клетка m_1 в состоянии a_1) взводит клетку $(m_1, 1)$, переводя ее в состояние a_1, распространяется ко всем клеткам последнего яруса $\{(m_1, m+\tilde{d})\}$ и включает в работу $\theta_i^{Я}$ и $\theta_i'^{Я}$, $i = 1,\ldots,\nu$. При этом клетка $(m_1, 1)$ переходит в состояние a_0, а клетка m_1 –

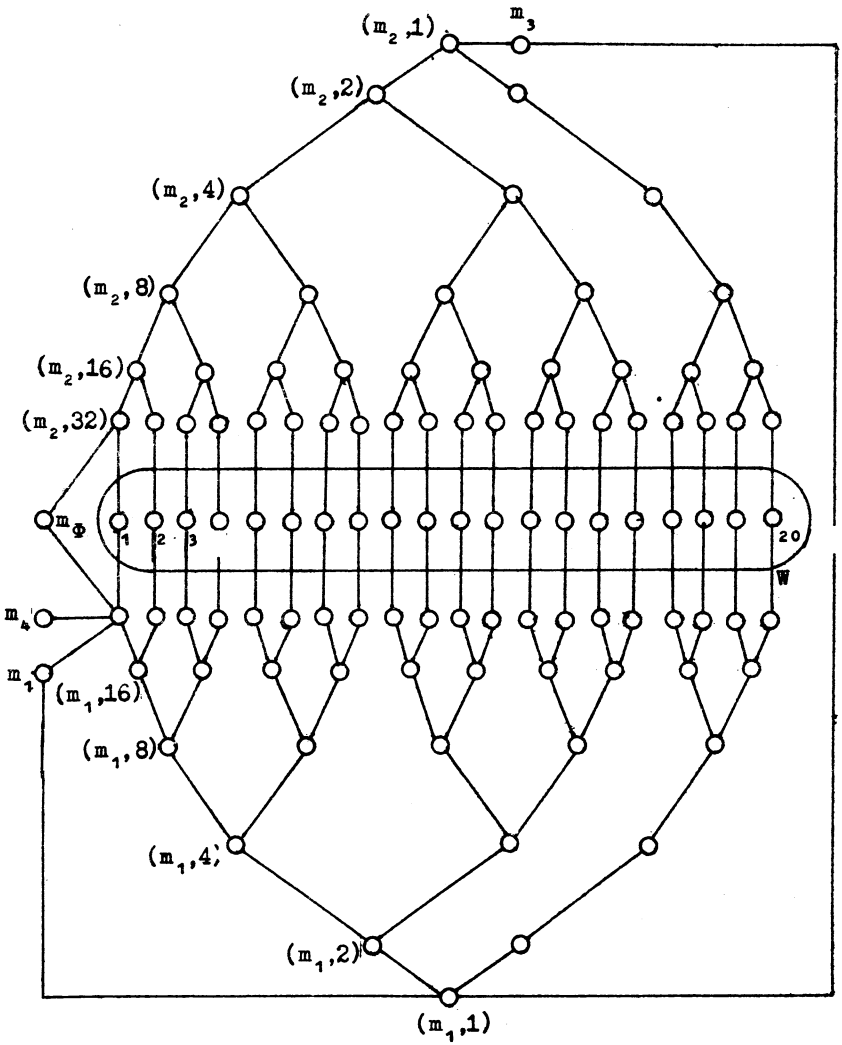

Рис.3. Сеть клеток.

в состояние a_2. Признаки применимости по дереву, построенному на клетках с именами $\{(m_2,m)\}$, конвейерно поступают в клетку m_3. Как только поступление признака применимости прекращается, клетка m_3 переводит клетку $(m_1, 1)$ в состояние a_1. Это состояние распространяется ко всем клеткам предпоследнего яруса клеток с именами $\{(m_1, m)\}$, при этом m_1 переходит в состояние a_0, m_4 – в состояние a_1 (сигнал "завершение"), клетки яруса $\{(m_1, m+\tilde{d})\}$ в состояние a_0 и алгоритм Φ^0 останавливается.

Выпишем систему подстановок:

$\theta_i^{\text{я}}$: $\{(S_{i1}(m))(a_0,(m_2,m+\tilde{d}))\} * \{(S_{i2}(m))(a_1,(m_1,m+\tilde{d}))\} \to$
$\to \{(S_{i3}(m))(a_1,(m_2,m+\tilde{d}))\}$, $i = 1,\ldots,\nu$;

$\theta_i'^{\text{я}}$: $S_{i1}(m) * \{(S_{i2}(m))(a_1,(m_1,m+\tilde{d}))\} \to S_{i3}(m)$, $i = 1,\ldots,\nu$;

$\theta_{2\nu+1}$: $\{(a_1, m_\Phi)(a_0,(m_2,1+\tilde{d}))\} * \{(a_1,(m_1,1+d))\} \to$
$\to \{(a_0,m_\Phi)(a_1,(m_2,1+d))\}$;

$\theta_{2\nu+2}$: $\{(a_1,m_1)(a_0,(m_1,1))\} * \to \{(a_2,m_1)(a_1,(m_1,1))\}$;

$\theta_{2\nu+3}$: $\{(a_1,(m_1,m))(a_0,(m_1,2m))\} * \to \{(a_0,(m_1,m))(a_1,(m_1,2m))\}$;

$\theta_{2\nu+4}$: $\{(a_1,(m_1,m))(a_0,(m_1,2m+1))\} * \to \{(a_0,(m_1,m))(a_1,(m_1,2m+1))\}$;

$\theta_{2\nu+5}$: $\{(a_0,(m_2,m))(a_1,(m_2,2m))\} * \to \{(a_1,(m_2,m))(a_0,(m_2,2m))\}$;

$\theta_{2\nu+6}$: $\{(a_0,(m_2,m))(a_1,(m_2,2m+1))\} * \to \{(a_1,(m_2,m))(a_0,(m_2,2m+1))\}$;

$\theta_{2\nu+7}$: $\{(a_0,m_3)(a_1,(m_2,1))\} * \to \{a_1,m_3)(a_0,(m_2,1))\}$;

$\theta_{2\nu+8}$: $\{(a_1,m_3)\} * \to \{(a_2,m_3)\}$;

$\theta_{2\nu+9}$: $\{(a_2,m_3)(a_1,(m_2,1))\} * \to \{(a_1,m_3)(a_0,(m_2,1))\}$;

$\theta_{2\nu+10}$: $\{(a_2,m_3)(a_0,(m_1,1))\} * \{(a_0,(m_2,1))\} \to \{(a_0,m_3)(a_1,(m_1,1))\}$;

$\theta_{2\nu+11}$: $\{(a_1,(m_1,m))(a_1,(m_1,2m))\} * \to \{(a_0,(m_1,m))(a_2,(m_1,2m))\}$;

$\theta_{2\nu+12}$: $\{(a_1,(m_1,m))(a_1,(m_1,2m+1))\} * \to \{(a_0,(m_1,m))(a_2,(m_1,2m+1))\}$;

$\theta_{2\nu+13}$: $\{(a_0,m_\Phi)\} * \{(a_2,(m_1,1+\tilde{d}))\} \to \{(a_1,m_\Phi)\}$;

$\theta_{2\nu+14}$: $\{(a_0,m_4)(a_2,m_1)\} * \{(a_2,(m_1,1+\tilde{d}))\} \to \{(a_1,m_4)(a_0,m_1)\}$;

$\theta_{2\nu+15}$: $\{(a_2,(m_1,m))\} * \to \{(a_0,(m_1,m))\}$.

Непосредственная проверка показывает, что эта система задает искомую каноническую форму Φ^0 .

ЗАМЕЧАНИЕ I. Пусть число итераций при переработке микропрограммой Φ клеточного множества W равно τ , числа ярусов в бинарном дереве равно h $(h =]\log_2 |W|[)$. Детальное построение последовательности итераций (мы его здесь не приводим) позволяет заключить, что при применении Φ^0 к W число итераций T равно $\tau + 3h + 3$.

ЗАМЕЧАНИЕ 2. При переходе от бинарного дерева к 1-арному можно уменьшить h , а значит, и T. Для этого достаточно микрокоманды, использующие в записи конфигураций именующие функции m , 2m, 2m+1, заменить микрокомандами, использующими функции m , 1(m-1)+2, 1(m-1)+3,..., 1m, 1m+1 . При этом общее число микрокоманд в микропрограмме Φ^0 возрастает.

2. Асинхронная композиция с использованием локальных конфигураций. Построение композиции проведем для типичного случая (если иметь в виду последующую реализацию сети автоматов в виде БИС), когда имена клеток можно интерпретировать как координаты узлов целочисленной решетки на плоскости; выберем M , равным N x N . Введем на плоскости координаты x,y . Тогда W – это некоторое конечное множество узлов на плоскости (x, y) . В качестве основы для построения канонической формы микропрограммы Φ (обозначим Φ^Λ) с использованием только локальных конфигураций выберем Φ^* . Введем ось координат v , перпендикулярную плоскости (x, y). Будем считать, что исходное W имеет вид прямоугольника. Все дополнительные клеточные множества и отдельные клетки будем располагать в плоскостях, параллельных (x, y), вдоль v . Клетки с одинаковыми x,y будем располагать одна под другой. Введем вспомогательные клетки m_1, m_2, m_3, m_4, m_B и вспомогательные клеточные множества W_1, W_2, W_C, W_Z, W_4 , изоморфные W . Вдоль оси v они отстоят друг от друга на единицу и упорядочены следующим образом: $W, W_1, W_C, m_B, W_Z, W_2, W_4, m_1, m_2, m_3, m_4$. В основном состояния вспомогательных клеток обозначаются символами a_0, a_1, a_2, дополнительные обозначения вводятся по ходу изложения. В исходном состоянии клетка m_1 находится в состоянии a_1, все остальные вспомогательные клетки в состоянии a_0. Введем именующие функции:

$$\Psi_1(x,y) = (m_1, x, y),$$

$$\Psi_2(x,y) = (m_2, x, y);$$

вместо $\theta_i^{Я}$ и $\theta_i'^{Я}$, $i = 1,\ldots, \nu$, запишем:

$$\tilde{\theta}_i^{Я} : \{(S_{i1}(x,y))(a_0,(m_2,x,y))\} * \{(S_{i2}(x,y))(a_1,(m_1,x,y))\} \to$$
$$\to \{S_{i3}(x,y))(a_1,(m_2,x,y))\}, \quad i = 1,\ldots,\nu ;$$

$$\tilde{\theta}_i'^{Я} : S_{i1}(x,y) * \{(S_{i2}(x,y))(a_1,(m_1,x,y))\} \to$$
$$\to S_{i3}(x,y), \qquad i = 1,\ldots,\nu ;$$

Эти две группы микрокоманд содержат только локальные конфигурации, если исходная Φ содержала локальные конфигурации. Объединение $\tilde{\theta}_i^{Я}$ и $\tilde{\theta}_i'^{Я}$, $i = 1,\ldots, \nu$, образует ядро микропрограммы Φ^{Λ} . Обозначим его $\Phi_{Я}^{\Lambda}$. Кроме ядра, Φ^{Λ} содержит еще две части: Φ_s^{Λ} – микропрограмму синхронизации и Φ_Z^{Λ} – микропрограмму зондирования.

Для обеспечения одинаковой работы Φ^{Λ} и Φ^{*} необходимо, чтобы взаимодействие частей Φ^{Λ} происходило по следующей схеме.

Клетка m_1 в состоянии 1 ("пуск") запускает микропрограмму Φ_s^{Λ} . Микропрограмма Φ_s^{Λ} использует клеточные множества W_C и W_1 и на некоторой итерации переводит сразу все клетки W_1 в состояние a_1. Это запускает в работу ядро $\Phi_{Я}^{\Lambda}$. Для индикации применимости подстановок ядра служат клетки множества W_2 . Микропрограмма Φ_Z^{Λ} , включаемая на той же итерации, что и Φ_s^{Λ} , использует множества W_Z, W_C и осуществляет конвейерный просмотр клеток множества W_2 . Если хотя бы одна клетка из W_2 оказывается в состоянии a_1 , ее состояние заменяется на a_0 ; состояние клетки $(m_Z,1,1)$ остается неизменным. Если при очередном просмотре все клетки из W_2 оказываются в состоянии a_0 (исходная Φ неприменима), клетка $(m_Z,1,1)$ переходит в состояние a_1 . Это служит сигналом для повторного включения W_s^{Λ} , которая переводит на некоторой итерации сразу все клетки W_1 в состояние a_0 . Это в свою очередь служит сигналом для перевода клетки m_4 в состояние a_1 ("ответ"), клеток m_3, m_2, m_1 в состояние a_0. Все описанные взаимодействия частей Φ^{Λ} осуществляются с помощью m_3, m_2.

Проведем детальное построение микропрограммы Φ^{Λ} , оценим ее сложность и число итераций, затраченное на исполнение.

Ядро содержит 2ν микрокоманд и выполняется τ итераций. Микропрограмма Φ_s^{Λ} с локальными конфигурациями легко строится на основе алгоритмов синхронизации в сети автоматов с локальными связями [3,6]. Покажем это. В качестве пусковой клетки выберем клетку

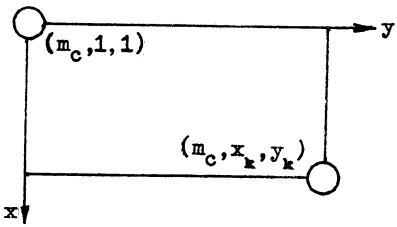

Рис.4

$(m_c,1,1)$ множества W_c (см. рис.4). Если двигаться из этой клетки вдоль границы прямоугольника в двух направлениях: I) параллельно оси x вниз и параллельно оси y вправо, 2) параллельно оси y вправо и параллельно оси x вниз, мы придем в одну и ту же клетку (m_c,x_k,y_k) . Фигуры на плоскости с таким свойством в [6] названы фигурами с фиксированной ориентацией. Все дальнейшие построения справедливы не только для клеточных множеств, имеющих вид прямоугольников, но и для клеточных множеств, являющихся любыми фигурами с фиксированной ориентацией. Если исходное W не является фигурой с фиксированной ориентацией, его всегда можно заменить минимальным по мощности клеточным множеством W', содержащим W и являющимся фигурой с фиксированной ориентацией (рис.5, ● - клетки исходного W , ○ - дополнительные клетки, принадлежащие W'), и все вспомогательные множества выбрать изоморфными W'. Штриховые линии на этой фигуре будем называть фронтами, их общее число обозначим f . В [6] показано, что если сеть автоматов образует фигуру с фиксированной ориентацией и каждый автомат имеет четырех соседей таких, как на рис.6, то синхронизация сети выполняется за (2f-2) такта. Постро-

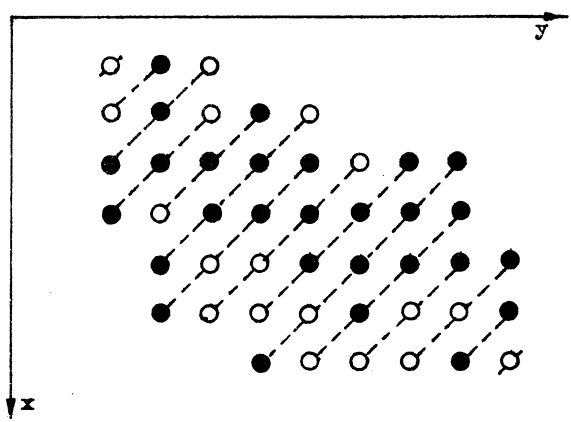

Рис.5

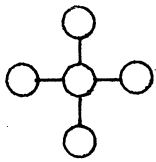

Рис.6

им микропрограмму, выполняющую синхронизацию W_c за (2f-2) итерации. Построение начнем с одномерного случая. Каждому правилу смены состояний автомата, имеющему вид [3,с.97]:

i-1	i	i+1
α	β	γ
	δ	

сопоставим микрокоманду $\{(\beta,i)\} * \{(\alpha,i-1)(\gamma,i+1)\} \to \{(\delta,i)\}$. Чтобы перейти к двумерному случаю, достаточно каждую такую микрокоманду заменить двумя:

$$\{(\beta,(m_c,x,y))\}*\{(\alpha,(m_c,x,y-1))(\gamma,(m_c,x,y+1))\}\to\{(\delta,(m_c,x,y))\};$$

$$\{(\beta,(m_c,x,y))\}*\{(\alpha,(m_c,x-1,y))(\gamma,(m_c,x+1,y))\}\to\{(\delta,(m_c,x,y))\}.$$

Такая процедура проделывается для всех правил смены состояний, кроме последнего. К полученному списку микрокоманд добавляются микрокоманды

$$\{(a_0,m_3)(a_0,(m_c,1,1))\}*\{(a_1,m_1)\}\to\{(a_1,m_3)(R,(m_c,1,1))\};$$

$$\{(F,(m_c,x,y))(a_0,(m_1,x,y))\}*\to\{(a_0,(m_c,x,y))(a_0,(m_1,x,y))\};$$

$$\{(F,(m_c,x,y))(a_1,(m_1,x,y))\}*\to\{(a_0,(m_c,x,y))(a_0,(m_1,x,y))\};$$

$$\{(a_1,m_1)(a_1,m_2)(a_1,m_3)(a_0,m_4)\}*\{(a_0,(m_1,1,1))\}\to$$
$$\to\{(a_0,m_1)(a_0,m_2)(a_0,m_3)(a_1,m_4))\}.$$

Обозначения состояний клеток взяты из [3,с.96].

Если обозначить мощность множества правил смены состояний автомата буквой ω, то общее число микрокоманд в Φ_a^A равно $2\omega + 3$.

Прежде чем построить Φ_z^A, более детально определим выполняемые ею действия. Они таковы.

I. Порождение фронта в клетке с именем (m_z,x_k,y_k) с периодичностью, зависящей от максимального размера окрестности клетки в W. Фронт в данном случае образуют клетки множества W_z, находящиеся в одном и том же состоянии, отличном от a_0, и расположенные вдоль штриховой линии.

2. Движение фронта.

3. Трансформация фронта, состоящего из клеток в состоянии a_1, во фронт, состоящий из клеток в состоянии a_2, при встрече с признаком применимости (клеткой из W_2, находящейся в состоянии a_1).

4. Регистрация неприменимости микрокоманд из $\Phi^{\Lambda}_{\text{я}}$, осуществляемая при помощи перевода клетки $(m_z,1,1)$ в состояние a_1 при переходе фронта, состоящего из клеток в состоянии a_1.

Прежде чем перейти к построению микрокоманд, порождающих фронт, отметим следующее. Так как все конфигурации, используемые в записи Φ локальны, существуют такие l_1, l_2, l_3, l_4 $(l_i \geq 0)$, что окрестность любой клетки множества W содержится в прямоугольнике со сторонами $(x+l_2)-(x-l_1)$, $(y+l_4)-(y-l_3)$. Обозначим $r = (l_2 + l_4)+1$. Чтобы какой-то фронт не "проскочил" признак применимости, необходимо чтобы между порождением очередных фронтов была задержка, равная r итерациям. Ниже выписаны микрокоманды, порождающие фронт:

$$\{(a_0,(m_z,x_k,y_k))(a_0,m_B)\}*\{(a_1,(m_1,x_k,y_k))\}\to$$
$$\to \{(a_1,(m_z,x_k,y_k))(a_1,m_B)\};$$

$$\{(a_i,m_B)\}*\{(a_1,(m_1,x_k,y_k))\}\to \{(a_{i+1},m_B)\},$$
$$i = 1,\ldots,r-1;$$

$$\{(a_r,m_B)\}*\{(a_1,(m_1,x_k,y_k))\}\to \{(a_0,m_B)\}.$$

Выпишем микрокоманды, выполняющие действия 2–4, при этом совокупность микрокоманд разделим на группы, связанные общим назначением.

Движение фронта:

$$\varepsilon_1\colon \{(a_1,(m_z,x,y))(a_0,(m_z,x-1,y))\}*\{(a_0,(m_4,x,y))(a_0,(m_2,x,y))\}\to$$
$$\to \{(a_0,(m_z,x,y))(a_1,(m_z,x-1,y))\};$$

$$\varepsilon_2\colon \{(a_1,(m_z,x,y))(a_0,(m_z,x,y-1))\}*\{(a_0,(m_4,x,y))(a_0,(m_2,x,y))\}\to$$
$$\to \{(a_0,(m_z,x,y))(a_1,(m_z,x,y-1))\}.$$

Встреча фронта с признаком применимости:

$$\varepsilon_3\colon \{(a_1,(m_z,x,y))(a_0,(m_z,x-1,y))(a_1,(m_2,x,y))\}*\{(a_1,(m_4,x,y))\}\to$$
$$\to \{(a_0,(m_z,x,y))(a_2,(m_z,x-1,y))(a_0,(m_2,x,y))\};$$

$$\varepsilon_4\colon \{(a_1,(m_z,x,y))(a_0,(m_z,x,y-1))(a_1,(m_2,x,y))\}*\{(a_1,(m_4,x,y))\}\to$$
$$\to \{(a_0,(m_z,x,y))(a_2,(m_z,x,y-1))(a_0,(m_2,x,y))\}.$$

Движение фронта:

$$g_5:\{(a_2,(m_Z,x,y))(a_0,(m_Z,x-1,y))\}*\{(a_1,(m_4,x,y))(a_0,(m_2,x,y))\}\to$$
$$\to\{(a_0,(m_Z,x,y))(a_2,(m_Z,x-1,y))\};$$

$$g_6:\{(a_2,(m_Z,x,y))(a_0,(m_Z,x,y-1))\}*\{(a_1,(m_4,x,y))(a_0,(m_2,x,y))\}\to$$
$$\to\{(a_0,(m_Z,x,y))(a_2,(m_Z,x,y-1))\};$$

$$g_7:\{(a_2,(m_Z,x,y))(a_0,(m_Z,x-1,y))(a_1,(m_2,x,y))\}*\{(a_1,(m_4,x,y))\}\to$$
$$\to\{(a_0,(m_Z,x,y))(a_2,(m_Z,x-1,y))(a_0,(m_2,x,y))\};$$

$$g_8:\{(a_2,(m_Z,x,y))(a_0,(m_Z,x,y-1))(a_1,(m_2,x,y))\}*\{(a_1,(m_4,x,y))\}\to$$
$$\to\{(a_0,(m_Z,x,y))(a_2,(m_Z,x,y-1))(a_0,(m_2,x,y))\};$$

$$g_9:\{(a_1,(m_Z,x,y))\}*\{(a_0,(m_4,x,y))(a_2,(m_Z,x,y-1))\}\to$$
$$\to\{(a_0,(m_Z,x,y))\};$$

$$g_{10}:\{(a_1,(m_Z,x,y))\}*\{(a_0,(m_4,x,y))\ (a_2,(m_Z,x,y-1))\}\to$$
$$\to\{(a_0,(m_Z,x,y))\}.$$

Регистрация неприменимости:

$$g_{11}:\{(a_2,(m_Z,1,1))\}*\{(a_0,(m_2,1,1))\}\to\{(a_0,(m_Z,1,1))\};$$

$$g_{12}:\{(a_2,(m_Z,1,1))(a_1,(m_2,1,1))\}*\to\{(a_0,(m_Z,1,1))(a_0,(m_2,1,1))\};$$

$$g_{13}:\{(a_1,(m_Z,1,1))(a_1,(m_2,1,1))\}*\to\{(a_0,(m_Z,1,1))(a_0,(m_2,1,1))\};$$

$$g_{14}:\{(a_1,(m_Z,1,1))(a_1,m_3)(a_0,m_2)\}*\{(a_0,(m_2,1,1))\}\to$$
$$\to\{(a_0,(m_Z,1,1))(a_0,m_3)(a_1,m_2)\}.$$

Система микрокоманд g_1–g_{14} сделана непротиворечивой при помощи клеток с именами (m_4,x,y). Состояние этих клеток изменяют микрокоманды

$$\{(a_0,(m_4,x,y))\}*\{(a_1,(m_1,x,y))\}\to\{(a_1,(m_4,x,y))\};$$
$$\{(a_1,(m_4,x,y))\}*\{(a_1,(m_1,x,y))\}\to\{(a_0,(m_4,x,y))\ \}.$$

И, наконец, после прихода всех клеток множества W_1 в состояние a_0, необходимо вернуть в исходное состояние клетки множеств W_4, W_Z и клетку m_B. Это делают микрокоманды:

$$\{(a_1,(m_4,x,y))\}*\{(a_0,(m_1,x,y))\}\to\{(a_0,(m_4,x,y))\};$$

$$\{(a_i,m_B)\}*\{(a_0,(m_1,x_k,y_k))\}\to\{(a_0,m_B)\},\quad i=1,\ldots,r;$$

$$\{(a_1,(m_Z,x,y))\}*\{(a_0,(m_1,x,y))\}\to\{(a_0,(m_Z,x,y))\}.$$

Теперь можно подсчитать как общее число микрокоманд в микропрограмме, так и число итераций, которое выполняет микропрограмма. Они соответственно равны $2(\nu+\omega+r+11)$ и $6t+\tau+2$.

Последнее, что нам следует сделать, это убедиться в том, что при построении Φ^A использованы только локальные конфигурации. Расположим клетки m_1, m_2, m_3, m_4 в одном столбце с клетками, имеющими координаты $x=1$, $y=1$, а клетку m_B – с клетками, имеющими координаты $x=x_k$, $y=y_k$. Непосредственная проверка показывает, что все возможные связи между клетками вдоль оси v, порожденные конфигурациями, которые использованы при построении Φ^A, исчерпываются показанными на рис.7, т.е. действительно конфигурации локальны.

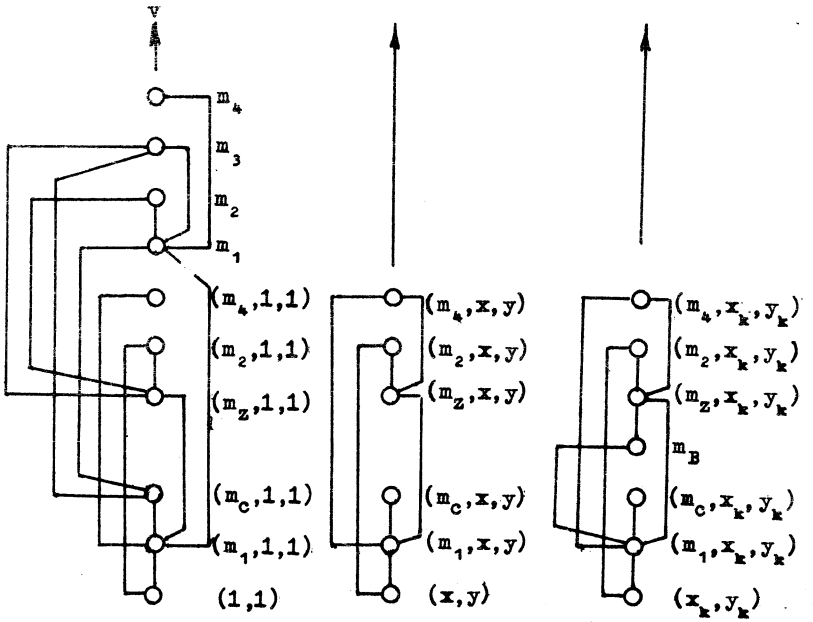

Рис.7

3. Сложность канонических форм. Сведем в единую таблицу характеристики сложности канонических форм микропрограмм, полученные выше. Причем в качестве одного из вариантов канонической формы будем рассматривать также Φ'. Обозначения в таблице те же, что и ранее: ν – число микрокоманд в исходной микропрограмме; τ – число итераций, затраченное на исполнение исходной микропрограммы; h – число ярусов в бинарном дереве, ω – число правил смены состояний автомата в алгоритме синхронизации цепи стрелков; r – радиус окрестности клетки в исходном W; f – линейный размер, характеризующий W.

Таблица

Тип	Сложность	
	микропрограммная (Q)	временная (T)
Φ	$Q = \nu$	$T = \tau$
Φ'	$Q = \nu + 3$	$T = 2\tau + 2$
Φ^{*}	$Q = 2\nu + 3$	$T = \tau + 2$
Φ^{0}	$Q = 2\nu + 15$	$T = \tau + 3h + 3$
Φ^{A}	$Q = 2(\nu + \omega + r + 11)$	$T = \tau + 6f + 2$

Заключение

Оценки сложности канонических форм получены при самых общих предположениях о механизмах пуска–останова микропрограмм, включаемых в композицию. При практической разработке микропрограмм часто можно не прибегать к общим каноническим формам, а, используя содержательный смысл микропрограмм, получать частные варианты канонических форм. В таких случаях полученные здесь оценки могут играть роль ориентиров, позволяющих определить целесообразность перехода к частному варианту и оценить его качество.

Литература

1. БАНДМАН О.Л., ПИСКУНОВ С.В., СЕРГЕЕВ С.Н. Синтез параллельных микропрограммных структур. – Кибернетика, 1981, № 5, с. 48–54.

2. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные универсальные системы высокой производительности. – Новосибирск: Наука, 1966. – 308 с.

3. Однородные структуры. Анализ. Синтез. Поведение / Варшавский В.И., Мараховский В.Б., Песчанский В.А., Розенблюм Л.Я. – М.: Энергия, 1973. – 150 с.

4. CORDON D., KOREN I., SILBERMAN G.M. Embeding tree structures in VLSI hexagonal arrays. - IEEE Transaction on Computers, 1984,v.C-33,N 1,p.104-107.

5. HOROWITZ, ZORAT A. The binary tree as an interconnection network: Application to multiprocessor systems and VLSI. - IEEE Transaction on Computers,1981,v.C-30,N 4,p.247-253.

6. GRASSELLI A. Synchronization of cellular array: The firing squad problem in two dimentions.- Information and Control, 1975, v.28,p.113-124.

Поступила в ред.-изд.отд.
28 сентября 1984 года