ПРОБЛЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ
(Вычислительные системы)
1983 год Выпуск 100

УДК 681.322

ОДНОРОДНЫЕ СТРУКТУРЫ И ПАРАЛЛЕЛЬНОЕ МИКРОПРОГРАММИРОВАНИЕ

О.Л.Бандман

§ I. Однородные вычислительные структуры. Реальные устройства и абстрактные модели

Однородными структурами в вычислительной технике называют устройства, состоящие из множества одинаковых процессорных элементов. Иногда это понятие трактуют в широком смысле, не принимая во внимание никаких других свойств (назначение, производительность, сложность), и причисляют к однородным структурам широкий спектр разнообразных устройств. Наиболее типичными из них являются: программируемые логические матрицы [I], схемы параллельной арифметики [2], ассоциативные процессоры [3], устройства обработки изображений [4], систолические структуры для решения задач линейной алгебры [5], системы из микропроцессоров [6], однородные вычислительные системы [7].

Существует более узкая трактовка термина "однородные структуры". Она утвердилась за устройствами специализированного типа, у которых сложность процессорного элемента не превышает сложность микро-ЭВМ. Такие однородные структуры заняли промежуточное место между вычислительными средами [8] и однородными вычислительными системами, хотя границы, определяющие это место, расплываются и в значительной степени условны, особенно по отношению к вычислительным средам.

Понятие вычислительной среды, выдвинутое в I962 г. [9], основывалось на представлении о том, что быстрое развитие микроэлектроники предоставит в распоряжение разработчиков ЭВМ дешевый и удобный строительный материал в виде сверхбольших интегральных схем однородной структуры, способных настраиваться программным пу-

тем на реализацию любого алгоритма. Хотя такой прогноз не оправдался, концепция вычислительной среды оказала большое влияние на развитие идей и методов проектирования цифровых устройств на основе больших интегральных схем. Более того, с каждым годом появляются реализации новых однородных структур, которые могут классифицироваться как специализированные вычислительные среды. Так, например, широко используемые программируемые логические матрицы есть не что иное, как вычислительная среда, ориентированная на реализацию систем логических функций. Систолические структуры обладают основными свойствами вычислительной среды для реализации задач линейной алгебры. Ассоциативные процессоры - для информационно-поисковых задач [IO]. Можно считать, что воплощение идей вычислительной среды происходит путем создания проблемно-ориентированных устройств параллельной обработки, т.е. как раз таких, которые мы называем однородными структурами (в узком смысле).

Класс однородных вычислительных систем (или, что то же самое, вычислительных систем с программируемой структурой) соприкасается с классом однородных структур по линии мультимикропроцессорных систем. Однако поскольку однородные вычислительные системы существенно универсальны, пересечение этих классов можно считать пустым.

Следующие свойства характеризуют однородные структуры и присутствуют в них частично или полностью:

I) параллельность обработки информации;

2) локальность взаимодействий;

3) однородность;

4) гибкость (перестраиваемость) функциональная и структурная.

Проявление этих свойств в специализированных однородных структурах имеет свои особенности, обусловленные следующим требованием: структура устройства и функции процессорных элементов должны соответствовать структуре информационных связей и операторам реализуемого алгоритма. Это требование в дальнейшем считается основным для синтеза логической структуры устройства и называется п р и н ц и п о м с о о т в е т с т в и я.

П а р а л л е л ь н о с т ь обработки информации в однородных структурах проявляется в двух аспектах: пространственном и временно́м. Пространственная параллельность (повсеместность) означает, что все процессорные элементы могут работать одновременно.

В чистом виде повсеместность имеет место, когда все процессоры на каждом шаге выполняют одни и те же команды, последовательность которых заранее определена программой. Временная параллельность означает, что на каждом шаге много (все или часть) команд воспринимаются каждым процессорным элементом и каждый выполняет ту, которая ему в данный момент подходит. Предельная параллельность имеет место тогда, когда на каждом шаге все команды воспринимаются всеми процессорными элементами. Естественно, что принцип соответствия структур алгоритма и устройства требует компромиссных сочетаний обоих аспектов.

Локальность взаимодействий в однородной структуре определяется структурой связей. Если каждый процессорный элемент непосредственно связан с ограниченным числом соседей и число соседей намного меньше, чем общее число элементов структуры, то возможны только локальные взаимодействия. Именно такие взаимодействия характеризуют параллельную обработку массивов. Однако реализуемый алгоритм может содержать и "глобальные" операторы (использующие данные всего массива). Реализация их в структуре с локальными взаимодействиями требует итеративного выполнения операций всех участвующих процессорных элементов, что уменьшает эффективность обработки. Поэтому глобальные связи во многих случаях дополняют однородную структуру, увеличивая ее соответствие со структурой реализуемого алгоритма.

Однородность - свойство, от которого устройства получили название. Оно присутствует обязательно, поскольку обусловлено назначением устройства обрабатывать массив однотипных данных. Нарушение однородности связано обычно с необходимостью введения вспомогательных или управляющих операторов, а также в случаях, когда реализуется композиция алгоритмов.

Перестраиваемость - свойство, которое не является обязательной характеристикой однородной структуры. Оно делает ее более гибкой, расширяя класс реализуемых алгоритмов. Существуют два вида перестраиваемости: структурный и функциональный. Структурный требует наличия коммутационных элементов, функциональный - настроечной памяти в процессорных элементах. Для функциональной настройки самый распространенный способ - микропрограммный, причем микропрограммная настройка может выполняться как на этапе производства микросхем, так и в условиях эксплуата-

ции. При этом носителями настроечной информации могут быть программируемые логические матрицы, постоянные ЗУ, ЗУ с произвольной выборкой, регистры и отдельные триггеры.

Общей теории однородных структур, которая бы в комплексе учитывала перечисленные свойства и была бы доступна проектировщикам, до сих пор не существует. Для каждого класса известных устройств делались попытки создать собственные методы синтеза. Так, например, хорошо разработаны способы проектирования параллельных арифметических блоков [I2], цифровых интегрирующих структур [II], коммутационных структур [I2], ассоциативных процессоров [I3] и др. Во всех случаях формальные методы проектирования не выходят за рамки классической теории конечных автоматов. Поэтому их применение ограничено синтезом процессорного элемента. Главной задачей является обеспечить соблюдение принципа соответствия структуры проектируемого устройства реализуемому алгоритму. Для параллельной обработки массивов данных этот принцип не может быть выражен в терминах классической теории вычислительных машин, поскольку две математические модели, на которых стоит эта теория - алгоритм и автомат, - отображают основные черты машин первых поколений: последовательный характер преобразования информации и четкое разделение функций памяти и логики.

Известные абстрактные модели преобразования массивов информации, оказавшиеся чрезвычайно плодотворными в теоретическом плане, к сожалению, не получили развития в направлении практического проектирования. Так, модель клеточного автомата [I4], созданная для исследования процессов самовоспроизведения, используется до сих пор для исследования возможностей существования различных конфигураций [I5]. Один из клеточных автоматов стал служить игровым полем для состязаний в построении конфигураций (игра "Жизнь") [I6]. Однако, к сожалению, даже более практические модификации клеточных автоматов [I7,I8] не нашли применения при разработке однородных структур, поскольку не разработан способ их представления, удобный для задания нужного алгоритма.

Другая интересная модель - итеративная сеть [I9] - принесла много пользы для изучения параллельных арифметических устройств, параллельно-последовательных (или,как их еще называют,пространственно-временных) преобразований вычислительных процессов, а также вычислительных способностей однородных структур (в смысле

классов распознаваемых языков) [20].Эта модель оказала влияние на развитие идей, однако практического распространения не получила.

Модель вычислительной среды [8] в отличие от предыдущих была предложена со специальной целью стать основой для методов проектирования однородных структур. В соответствии с этим назначением - быть универсальным конструктором - вычислительные среды исследовались на предмет реализации в них вычислительных устройств и конечных автоматов [21]. Результаты этих исследований частично трансформировались в методы проектирования БИС и СБИС типовых структур (ПЛМ, базовые кристаллы), однако не помогли развитию теории однородных структур параллельного типа, поскольку сама модель параллельности преобразований не отражает.

Интенсивные поиски способов реализации параллельных алгоритмов в вычислительных средах привели к появлению еще одной модели - алгоритма параллельных подстановок [22].Этот алгоритм представляет собой,с одной стороны,параллельную версию алгоритма Маркова и Колмогорова-Успенского, с другой - обобщение клеточного автомата. Для практических целей важно, что алгоритм параллельных подстановок сочетает в себе три следующих качества: I) способность выражения алгоритма преобразования массивов данных, 2) параллель-

Таблица

Модель ОС	Вычисление		Взаимодействие		Тип структуры		Перестраиваемость	
	Параллельное	Последовательное	Локальное	Глобальное	Однородная	Произвольная	Структурная	Функциональная
Клеточный автомат	+	-	+	-	+	-	-	+
Итеративная сеть	+	-	+	+	+	-	-	+
Вычислительная среда	-	+	+	+	+	-	+	-
Алгоритм параллельных подстановок	+	+	+	+	+	+	-	+

Способность отображать соответствующее свойство обозначается "+", неспособность "-".

ность преобразований и 3) возможность формального перехода от алгоритма к "конструкции" однородной структуры.

Выразительные способности перечисленных моделей сведены в таблицу, анализ которой убеждает в целесообразности выбора алгоритма параллельных подстановок в качестве основной модели для построения методов синтеза однородных структур.

Результаты исследования свойств алгоритмов параллельных подстановок [23] и их использование в конкретном проектировании [24] привели к появлению методов описания, преобразования, композиции, моделирования и построения однородной структуры, реализующей заданный алгоритм. Все это в совокупности составляет предмет параллельного микропрограммирования, которое может рассматриваться как обобщение традиционного микропрограммного способа проектирования.

§ 2. Микропрограммные описания параллельных вычислений

Параллельная микропрограмма отличается от традиционной (последовательной) тем, что порядок применения микрокоманд в ней не определен. Вместо этого каждая микрокоманда содержит условие ее применимости и выполняется там и тогда, где и когда это условие удовлетворяется. Если много микрокоманд оказывается применимыми, то они выполняются одновременно или в произвольном порядке. Выполнение микрокоманды состоит в изменении состояний указанных в ней элементов памяти.

Для формального определения микрокоманды каждый процессорный элемент структуры характеризуется парой (a_i, m_i), где a_i – символ алфавита состояний A, а m_i – имя элемента из множества M. Множество таких пар (клеток) $W = \{(a_i, m_i)\}(i = 1,\dots, n)$, в котором нет двух пар с одинаковыми именами, называется клеточным множеством. Конечные клеточные множества часто называют "словами". Параллельная микропрограмма применяется к слову и представляет собой неупорядоченную совокупность микрокоманд $\Phi = \{\theta_1,\dots,\theta_v\}$. Микрокоманда, в свою очередь, определяет множество микроопераций $\theta = \{\rho_1,\dots,\rho_m\}$, причем

$$\rho: W' \cup W'' \to W''', \qquad (1)$$

где $W' = \{(a_i, m_i)\}$; $W'' = \{(b_j, n_j)\}$; $W''' = \{(c_i, m_i)\}$ – клеточные множества $(a_i, b_j, c_i \in A,\ m_i, n_j \in M;\ i = 1,2,\dots,p,\ j = 1,2,\dots,q)$.

Микрокоманда, состоящая из одной микрооперации, называется э л е м е н т а р н о й. Элементарная микрокоманда применима к клеточному множеству W, если

$$W' \cup W'' \subseteq W. \qquad (2)$$

Применение элементарной микрокоманды означает, что W преобразуется в W_1 следующим образом:

$$W_1 = W \backslash W' \cup W'''. \qquad (3)$$

Иными словами, применение микрооперации происходит путем смены состояний $a_i \to c_i$ в клетках с именами m_i $(i = 1,2,\ldots,p)$.

В общем случае алфавит состояний A содержит подмножество переменных символов $A_X = \{x,y,z,u,v\ldots\}$ и функциональных $A_F = \{f_1,\ldots,f_k\}$, где $f_i(x,y,z\ldots)$ имеет область определения и значений, равную множеству (или подмножеству) конкретных символов алфавита $A' = \{a,b,c,\ldots\}$, причем $A = A' \cup A_X \cup A_F$. Слово, которое содержит клетки с переменными символами, называется п е р е м е н н ы м словом. Оно представляет множество слов, получающихся заменой каждого переменного символа на любой символ из A_X. Так, например, слово, содержащее p клеток с переменными символами, представляет множество слов мощностью $|A^p|$.

Микрокоманды, в которых используются переменные и функциональные символы, называются ф у н к ц и о н а л ь н ы м и и выглядят следующим образом:

$$\theta : \{(x_i,m_i)\} \cup \{(y_j,n_j)\} \to \{(f_i(x_1,\ldots,x_p,y_1,\ldots,y_q)),m_i)\}. \qquad (4)$$

Функциональная микрокоманда представляет множество микроопераций, соответствующих словам, входящим в переменное слово, равное левой части θ. Функциональная микрокоманда применима, если для одного из этих слов выполняется (2). Результат применения выражается в виде (3) с той лишь разницей, что состояния клеток из W''' равны значениям функций правой части. Процедура вычисления этих функций в микропрограмму Ф не входит.

Выражения (I) и (4) определяют изменения состояний на конкретном подмножестве клеток с именами, перечисленными в W'. С помощью таких выражений отобразить повсеместность обработки массива чрезвычайно сложно, так как их надо записать для подмножеств клеток, покрывающих все клеточное множество. Чтобы избежать повторений и выразить в одной микрокоманде возможность ее применения в любой области клеточного множества, вводится понятие конфигурации.

Пусть $\varphi_i(m)$ $(i = 1,2,\ldots,p)$ – функции, областью определения и значений которой служит множество имен клеток M. Для каждого $m \in M$ этот набор функций определяет некоторое подмножество клеток-соседей, называемое окрестностью m. Каждая пара: набор функций $\varphi_1(m),\ldots,\varphi_p(m)$ и набор символов $a_1,\ldots,a_p$ образуют множество слов вида

$$S(m) = \{(a_1,\varphi_1(m)),\ldots,(a_p,\varphi_p(m))\},\ \forall m \in M\ , \qquad (5)$$

называемое к о н ф и г у р а ц и е й. Слово $S(m_i)$, полученное подстановкой $m = m_i$ в (5), называется элементом конфигурации.

Поскольку реальные массивы данных имеют чаще всего структуры k-мерных конечных массивов, множеством имен считают k-компонентные векторы координат, а функции $\varphi_i(m)$ выбирают из класса сдвигов по осям координат.

Микрокоманды, в которых вместо конкретных клеточных множеств W', W'' и W''' используются конфигурации S', S'' и S''', называются п а р а л л е л ь н ы м и и имеют вид

$$\theta: S' * S'' \to S'''\ , \qquad (6)$$

где $S' = \{(a_1,\varphi_1(m))\ldots(a_p,\varphi_p(m))\}$, $S''=\{(b_1,\ \phi_1(m)\ldots(b_q,\phi_q(m))\}$, $S''' = \{(c_1,\varphi_1(m))\ldots(c_q,\varphi_q(m))\}$, а операция "$*$" называется произведением конфигураций и обозначает, что для любого $m \in M$ имеем $S'(m) * S''(m) = S'(m) \cup S''(m)$. Операция "$*$" определена, если $\{\varphi_1(m),\ldots,\varphi_p(m)\} \cap \{\phi_1(m),\ldots,\phi_q(m)\}=\emptyset$ для любого $m \in M$.

Параллельная микрокоманда содержит $|M|$ микроопераций $\rho = \theta(m)$: $S'(m) * S''(m) \to S'''(m)$.

Она применима к W, если для какого-либо $m \in M$

$$S'(m) \cup S''(m) \le W. \qquad (7)$$

Применение параллельной микрокоманды происходит путем выполнения всех содержащихся в ней микроопераций. При этом результат применения θ к W

$$\theta(W) = W \setminus \bigcup_{m \in M'} S'(m) \bigcup_{m \in M'} S'''(m)\ , \qquad (8)$$

где M' – подмножество имен клеток, для которых выполнено (7). Параллельная микропрограмма Φ применима к W тогда, когда применима хотя бы одна из ее микрокоманд $\theta_i \in \Phi$. Применение Φ к W есть итерационная процедура, на каждом k-м шаге которой

получается новое клеточное множество W^k, и так до тех пор, пока к полученному W микропрограмма применима.

Существуют два способа выполнения этой процедуры, или два способа интерпретации параллельной микропрограммы: синхронный и асинхронный.

С и н х р о н н а я и н т е р п р е т а ц и я использует правила применения подстановок в алгоритмах параллельных подстановок, т.е. на каждом шаге выполняются все применимые микрокоманды. Так, если $\Phi' \subset \Phi$ - множество применимых к W' микрокоманд, то

$$W^{k+1} = \bigcup_{\theta_i \in \Phi'} \theta_i(W^k). \qquad (9)$$

А с и н х р о н н а я и н т е р п р е т а ц и я [25] предполагает, что на каждом шаге применяется только одна микрооперация, т.е. если $\Phi' \subset \Phi$ - множество применимых к W^k микрокоманд, объединяющих микрооперации из $R = \bigcup_{m \in M'} \bigcup_{\theta \in \Phi'} \theta(m)$ (M' - множество имен клеток таких, что $m \in M'$, если $\theta(m)$ применима к W^k), то $W^{k+1} = \rho(W^k)$, где ρ - любая микрооперация из R.

Каждой параллельной микрокоманде соответствует интерпретирующая ее сеть конечных автоматов. При этом множеству M соответствует множество имен автоматов, алфавиту A - множество их внутренних состояний. Связи между автоматами и их функционирование однозначно задаются микропрограммой Φ следующим образом. Пусть $X(m) = \{x_0(m),\ldots,x_p(m)\}$ и $z(m) = \{z_0(m),\ldots,z_q(m)\}$ - множество входов и выходов автомата с именем m, причем $x_0(m)$ и $z_0(m)$ - вход и выход внутренней памяти. Тогда из условий применения микрокоманд непосредственно вытекают два правила построения связей и логического функционирования автомата с именем $m \in M$.

1. Выход $z_0(m)$ связан с $x_k(m')$, если существует $\theta \in \Phi$, в которой $m = \varphi_i(m')$ или $m = \phi_i(m')$; $z_i(m)$ связан с $x_0(m')$, если существует $\theta \in \Phi$, в которой $\varphi_i(m) = m'$ ($i = 1,\ldots,p$; $l = 1,\ldots,q$).

2. Каждая микрокоманда $\theta_i \in \Phi$, которая содержит клетку с именем m_i в базовой части, определяет переход автомата из состояния a_i в c_i, когда на соответствующих входах m_i имеются состояния клеток из левой части θ_i. При этом на выходах появляются состояния клеток правой части.

Построенная по приведенным правилам сеть автоматов с именами из области определения именующих функций имеет однородную структуру. Это значит, что все автоматы одинаковы и связи между ними однотипны. Неоднородности обусловлены обычно тем, что в каких-либо микрокомандах упоминаются клетки с конкретными именами. Такие клетки обычно играют управляющую роль.

Каждому виду интерпретации соответствует свой режим работы интерпретирующей сети. Синхронная сеть предполагает наличие общего тактового сигнала, управляющего всеми автоматами так, чтобы все применимые микрооперации могли выполниться одновременно. Асинхронные сети автоматов интерпретируют асинхронный способ выполнения микропрограмм и никаких синхронизирующих сигналов не имеют. Изменение состояний автоматов становится возможным, как только в сети оказывается ситуация, соответствующая применимости какой-либо микрооперации. Смена состояний, интерпретирующая выполнение микрооперации, происходит через какое-то время, которое не регламентируется (требуется только, чтобы оно было конечным). При этом предполагается, что никакие две микрооперации не выполняются одновременно. Такое предположение, хотя и идеализирует асинхронную работу, вполне правомерно для исследования свойств микропрограмм, поскольку все другие режимы можно рассматривать как сочетание синхронных и асинхронных интерпретаций.

> Параллельная микропрограмма содержит полную информацию о структуре и логическом функционировании реализующего ее устройства. Процедуры структурно-логического синтеза носят формальный характер и могут быть автоматизированы.

Параллельные микропрограммы, как и всякое математическое обеспечение, должны иметь формы выражения, понятные обычной ЭВМ, для отладки, коррекции, моделирования работы в различных режимах и т.д. В связи с этим разработан язык параллельного микропрограммирования, отличающийся от приведенных форм записи точным описанием синтаксиса и возможностью перевода на стандартный машинный язык [26]. Основные конструкции языка подразделяются на две группы: 1) конструкции, предназначенные для описания структуры устройства (алфавит, клеточный массив, композиции массивов, процедуры вычисления имен), 2) конструкции, определяющие алгоритм функционирования (микрокоманды, циклический блок, микропрограммы). В резуль-

тате использования языка для моделирования работы ряда типовых структур (вычислительная среда, ассоциативный процессор, мультимикропроцессорное устройство) выработались приемы построения моделей для однородных структур разных типов: синхронных и асинхронных; управляющих и вычислительных; комбинационных, автоматных и микропроцессорных.

Язык параллельного микропрограммирования является средством отладки параллельных микропрограмм и моделирования работы однородных структур на ЭВМ.

§ 3. Теория параллельного микропрограммирования, проблемы и результаты

В отличие от параллельного программирования, в котором предполагается, что процессы, протекающие в системе, взаимодействуют редко, в параллельном микропрограммировании взаимодействия не отделяются от основных действий. Поэтому проблемы, присущие параллелизму, хотя и имеют ту же природу (синхронизация, детерминизм, сложность), проявляются иным образом и требуют самостоятельного решения.

Три группы проблем требуют решения на теоретическом уровне: I) проблемы, связанные с детерминированностью вычислений; 2) проблемы композиции параллельных алгоритмов (корректность и синхронизация) и 3) проблемы пространственно-временной оптимизации.

I. Д е т е р м и н и р о в а н н о с т ь п а р а л л е л ь н ы х м и к р о п р о г р а м м. Все клеточные множества, которые могут получиться в процессе применения Φ к W, образуют множество достижимых слов $\hat{W}$. Это множество вместе с определенным на нем отношением r ($W_i \, r \, W_j$, если существует последовательность шагов, в которой W_i предшествует W_k) называется в ы ч и с л е н и е м и обозначается Φ_W. Вычисление может быть синхронным $\bar{\Phi}_W$ и асинхронным $\bar{\bar{\Phi}}_W$, в зависимости от того, какая интерпретация микропрограммы имеется в виду. Вычисление называется детерминированным, если в $\hat{W}$ существует только одно слово $W_i \subset \hat{W}$, которое не имеет последователя, т.е. если вычисление имеет только один результат. Микропрограмма д е т е р м и н и р о в а н а, если для любого $W \in A \times M$ вычисление Φ_W детерминировано.

Параллельная микропрограмма отвечает своему назначению только, если она детерминирована. Необходимые и достаточные условия де-

терминированности, методы проверки микропрограмм на детерминированность, синтез детерминированных микропрограмм - главные проблемы параллельного микропрограммирования. Эти проблемы еще далеки от полного решения, хотя некоторые результаты уже получены и пути решения намечены [25,27]. Известно, что для синхронных и асинхронных интерпретаций условия детерминированности разные.

Для синхронных интерпретаций необходимым и достаточным условием детерминированности является непротиворечивость. Свойство непротиворечивости заключается в том, что при применении микропрограммы к любому $W \in A \times M$ в ней не могут оказаться применимыми сразу 2 микрооперации, изменяющие состояния одной и той же клетки по-разному. Это свойство достаточно подробно исследовано [23], однако удалось получить необходимые и достаточные условия только для класса параллельных микропрограмм с именующими функциями вида $\varphi(x) = x \pm c$ $(c \in M)$ [28]. Для асинхронных вычислений свойство непротиворечивости не является признаком детерминированности. В [27] показано, что существуют противоречивые микропрограммы, для которых асинхронные интерпретации детерминированы, и наоборот.

Проблема обеспечения детерминированности в асинхронном случае более сложна и менее разработана. Нам известны пока только некоторые достаточные условия детерминированности, выраженные через свойства однозначности и устойчивости, которые, в свою очередь, определяются через поведенческие свойства моделирующих сетей Петри [29].

В асинхронных вычислениях порядок выполнения примененных на каждом шаге микроопераций не определен, и поэтому возможно много разных последовательностей $\sigma_k \in \Sigma$ вида $\sigma_k = W^0 W_k^1 \ldots W_k^n$, преобразующих исходное W^0 в конечное W_k^n. Эти последовательности называются реализациями. Если в реализации σ_k выделить клетку с именем $m \in M$ и затем в последовательности ее состояний исключить из каждой группы рядом стоящих одинаковых символов все, кроме одного, то получится последовательность $h(m)$, называемая историей клетки. История клетки характеризует смену ее состояний в процессе вычисления. Если для всех реализаций $\sigma_k \in \Sigma$ вычисления Φ_W история клетки одна и та же и если это справедливо для всех $m \in M$, то вычисление Φ_W называется о д н о з н а ч н ы м. Если для всех $W \in A \times M$ вычисления Φ_W однозначны, то микропрограмма Φ однозначна.

Об однозначных микропрограммах мы уже имеем немало сведений. Известно, что противоречивая микропрограмма не может быть однозначной. Построен общий метод написания однозначной параллельной микропрограммы, эквивалентной заданной непротиворечивой. Показано, что при этом необходимо не менее чем втрое увеличить число микрокоманд и так же растет число итераций в вычислениях.

Класс однозначных микропрограмм включает в себя устойчивые микропрограммы, порождающие вычисления, характеризующиеся следующим. Если какая-либо микрооперация оказалась применимой, то ее применимость не может быть разрушена до тех пор, пока она не выполнится. Устойчивые вычисления моделируются устойчивыми сетями Петри [27] (т.е. такими сетями, в которых возбуждение перехода может быть снято только путем его собственного срабатывания).

Методы распознавания детерминированности основываются на анализе вычислений, порожденных применением микропрограммы к о п а ц е н ы м клеточным множествам. Такие клеточные множества конструируются путем объединения левых частей микрокоманд и их всевозможных сдвигов. В [27] показано, что достаточно исследовать вычисления на клеточных множествах, полученных для пар пересекающихся микрокоманд (пересекающиеся микрокоманды имеют одинаковые клетки в левых частях или каких-либо их сдвигах). Если эти вычисления детерминированы, то микропрограмма детерминирована. Если эти вычисления устойчивы, то микропрограмма устойчива. Анализ детерминированности (устойчивости) можно проводить непосредственным построением вычислений, можно путем построения моделирующих сетей Петри.

Сеть Петри, моделирующая асинхронное вычисление, строится следующим образом. Множеству клеток $(a_i, m_j) \in A \times M$ ставится в соответствие множество позиций $\Pi = \{p_1, \ldots, p_q\}$ сети Петри, множеству микроопераций $\{\theta_i(m)\}$ - множество переходов с указанием для каждого из них входных и выходных позиций. Исходному клеточному множеству W^0 соответствует множество маркированных позиций при начальном маркировании M^0. Выполнению микрооперации соответствует срабатывание перехода. Граф вычисления Φ_{W^0} изоморфен графу достижимых маркирований $G(M^0)$. Асинхронные вычисления моделируются ординарными консервативными и безопасными сетями Петри, притом устойчивые вычисления - устойчивыми сетями Петри.

Для распознавания детерминированности важна следующая доказанная в [27] теорема: асинхронное вычисление детерминировано тогда и только тогда, когда моделирующая его сеть Петри жива.

> Условиями детерминированности параллельной микропрограммы с именующими функциями вида $\varphi(x) = x \pm c$ являются непротиворечивость (для синхронных интерпретаций) и живость сети Петри, моделирующей вычисления на опасных клеточных множествах (для асинхронных интерпретаций). Для микропрограммы с именующими функциями произвольного вида проблема детерминированности открыта

П. К о м п о з и ц и я параллельных микропрограмм. Композиция понимается как способ составления сложной микропрограммы Φ из более простых $\Phi_1, \ldots, \Phi_n$ путем определения отношений между ними. Используется стандартный набор отношений: безусловный переход, условный переход, условное слияние, параллельное ветвление, объединение параллельных ветвей. Структура композиции наглядно изображается в виде параллельной граф-схемы алгоритмов [23] и выражается в виде параллельной микропрограммы, называемой управляющей. Поскольку время выполнения компонентных микропрограмм не определено, предполагается асинхронная интерпретация управляющей микропрограммы. Написание управляющей микропрограммы сводится к формальной процедуре замены вершин граф-схемы фрагментами сети Петри, которые моделируют организацию взаимодействий компонентных микропрограмм. Имеют смысл только такие композиции, структура отношений в которых обеспечивает а) детерминированность вычислений, б) бесступиковость (конечность времени протекания процесса до заданного в композиции оператора "конец"). В совокупности эти свойства называются к о р р е к т н о с т ь ю композиции. Условия корректности выражаются через свойства сетей Петри в виде следующей теоремы [23]: композиция корректна тогда и только тогда, когда моделирующая управляющую программу сеть Петри жива и безопасна.

Свойства живости и безопасности сетей Петри находятся в стадии исследования. Полезные для практики результаты получены только для ограниченного класса сетей, называемого сетями свободного выбора [30]. Однако этих результатов оказалось достаточно для разработки алгоритмов и программ анализа параллельных композиций на

корректность [31]. Стремление уменьшить сложность алгоритма привело к постановке и решению задачи р е д у к ц и и сетей: поиска способов их сокращения, сохраняющих свойства живости и безопасности. В результате найден набор правил редукции, применение которых имеет полиномиальную сложность и поэтому значительно расширяет возможности алгоритма [32].

Применение редукции породило новую проблему, названную проблемой редуцируемости сетей Петри. В [33] было высказано предположение, что большинство сетей редуцируются в очень простые и даже в тривиальные (имеющие не более двух вершин). Подробное исследование редуцируемости для разных классов сетей Петри проведено в [32], где выявлен ряд отношений между классами сетей Петри по редуцируемости. В частности,показано, что класс хорошо сформированных сетей (соответствующих структурированным граф-схемам) более узкий,чем класс сетей,редуцируемых в тривиальную типа N^0 (состоящую из одного перехода). Класс сетей N^1 (редуцируемых к контуру из двух вершин) является подклассом сетей, для которых существует живое и безопасное маркирование. К сожалению, автору не удалось доказать или опровергнуть гипотезу о совпадении этих двух классов, что было бы очень полезно для практики.

Для полного анализа свойств сетей Петри разработаны алгоритм и программа построения множества достижимых маркирований для сетей произвольного типа [34]. Этот алгоритм трудоемкий (имеет экспоненциальную сложность), но дает некоторую информацию не только о некорректности, но и о причинах ее появления.

> Проблема корректности параллельных композиций удовлетворительно решена в части проверки факта корректности (разработан алгоритм полиномиальной сложности). В части диагностики некорректности удовлетворительного решения не найдено.

Включение микропрограммы в композицию требует дополнения ее микрокомандами, обеспечивающими композиционную синхронизацию,т.е. восприятие сигнала о разрешении начать работу ("пуск") и выработку сигнала о получении результата ("завершение"). Композиционная синхронизация представляет собой сложную проблему, поскольку операции "пуск-завершение" являются по своей сути глобальными, что не согласуется с локальным характером взаимодействий в однородной

структуре. Сигнал "пуск" должен дать разрешение выполняться в с е м применимым микрооперациям, а "завершение" должно удостоверить, что н и о д н а микрооперация неприменима. Отсюда ясно, что решение задачи композиционной синхронизации существенно зависит от степени локальности микропрограммы. Поэтому задача рассматривалась отдельно для случаев, когда в микрокомандах допускаются конфигурации трех типов: I) г л о б а л ь н ы е (в интерпретирующей сети существуют автоматы с числом соседей, зависящем от мощности клеточного множества); 2) о г р а н и ч е н н ы е (каждый автомат связан не более чем с k ($k \ll |W|$) соседями); 3) л о к а л ь н ы е (связи каждого автомата не длиннее d ($d \ll L$ - линейный размер ограничивающего сеть пространства)).

Пусть Φ - микропрограмма, включаемая в композицию, Φ' - ее композиционное расширение, $\nu = |\Phi|$, $N = |\Phi'|$, τ и T - числа итераций вычислений Φ_W и Φ'_W соответственно, $W' \subset A \times M$. Получены следующие оценки сложности композиционных расширений*).

I. При использовании глобальных конфигураций, если требуется минимизировать сложность автомата в интерпретирующей сети, то

$$N = \nu + 3, \quad T = 2\tau,$$

если требуется минимизировать время, то

$$N = 2\nu + 3, \quad T = \tau + 3.$$

2. При использовании ограниченных конфигураций

$$N = 2\nu + 15, \quad T = \tau + 3 \log_k |W| + c .$$

3. При использовании локальных конфигураций (для двумерной структуры размерами $m \times n$, $d = 1$)

$$N = \nu + 10, \quad T = \tau + 4(m+n) + c,$$

где $c \leq 10$.

Показана возможность, и определена сложность построения композиционной синхронизации с использованием конфигураций, определяющих любую степень локальности взаимодействий.

---

*) Исследование композиционной синхронизации выполнялось С.В.Пиекуновым.

Ш. Пространственно-временная оптимизация параллельных микропрограмм. В отличие от первых двух групп проблем, решение которых определяет, как составить работоспособную параллельную микропрограмму, третья группа ставит вопрос, как сделать микропрограмму хорошей (в каком-то смысле). Требования к качеству микропрограммы обычно содержат, с одной стороны, производительность (минимум времени), с другой стороны, технические ограничения на реализацию, которые выражаются через параметры интерпретирующей сети автоматов: длину микропрограммы, мощность окрестности, мощность алфавита. Эти параметры определяют пространственную организацию интерпретирующей сети автоматов и обычно находятся в противоречии с требованием по производительности, поскольку уменьшение времени вычислений всегда дается ценой увеличения пространственных параметров. Отсюда вытекает очень важная задача исследования пространственно-временных эквивалентных преобразований параллельных микропрограмм. К сожалению, общих подходов к решению этой задачи еще не найдено. Можно предложить только перебор вариантов с привлечением моделирования [23].

Проблема пространственно-временной оптимизации параллельных микропрограмм открыта.

К проблеме пространственно-временной оптимизации примыкают также вопросы построения универсальных параллельных микропрограмм, моделирующих работу других микропрограмм. Этот вопрос рассмотрен в [23]. Там показан способ построения моделирующей микропрограммы для класса алгоритмов, задаваемых набором именующих функций.

В общей постановке проблема построения универсальной микропрограммы открыта.

§ 4. Структурное проектирование однородных параллельных вычислений

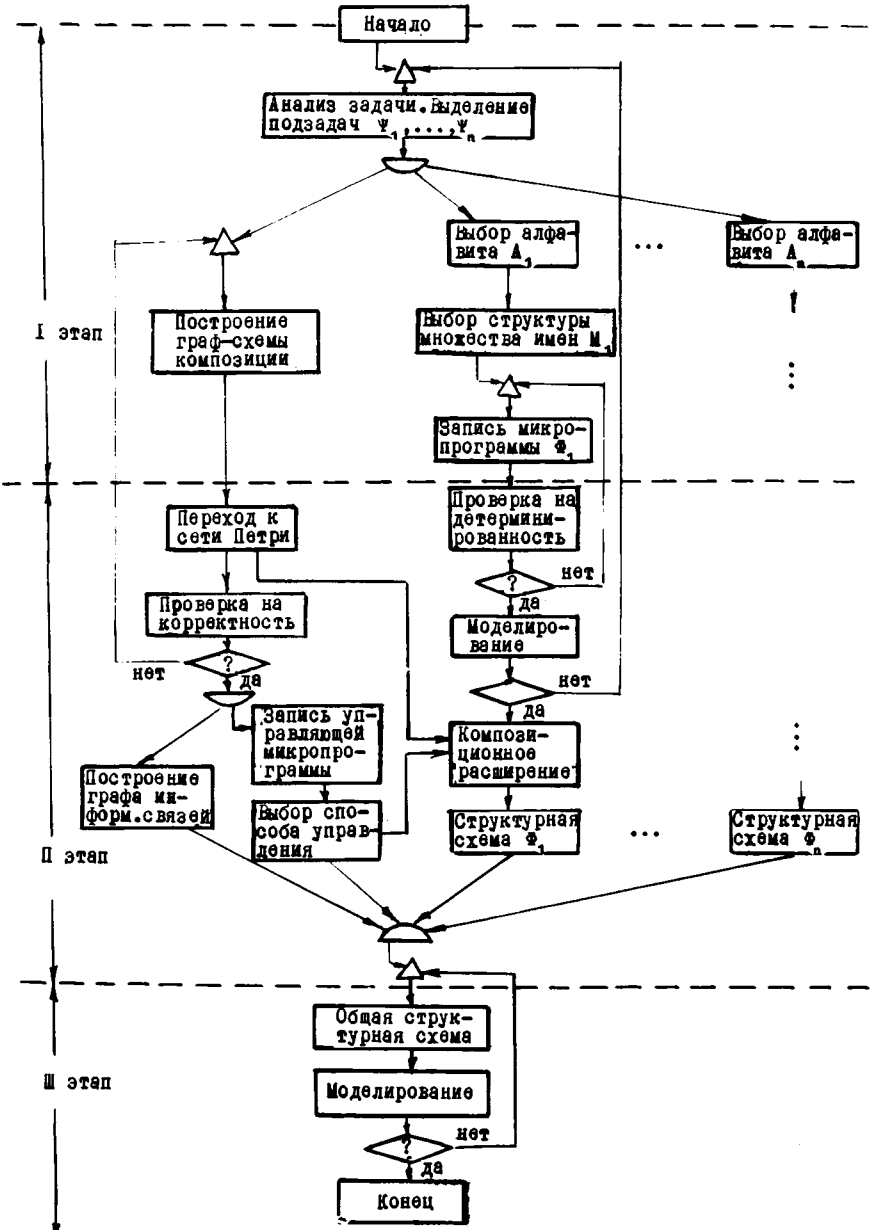
Основное назначение параллельного микропрограммирования - формализация процесса структурно-логического проектирования проблемно-ориентированных вычислительных устройств. Под этим понимается систематический переход от алгоритма решения заданного класса

задач к структурной схеме устройства, реализующего этот алгоритм. Структурная схема представляется множеством связанных между собой ячеек. Каждой ячейке поставлены в соответствие функция и некоторое количество входов и выходов. Связи определены как отношение между входами и выходами ячеек. Построение структурной схемы (структурный синтез) в системах проектирования обычно выполняется вручную. Методы параллельного программирования позволяют формализовать большую часть процедур структурного синтеза и тем самым поднять уровень автоматизации проектирования устройств параллельного типа на более высокую ступень, чем для последовательных машин.

Разработанная методика [35] состоит из трех этапов. Первый, неформализуемый этап содержит анализ заданного класса задач, разработку алгоритма решения и представления его в виде композиции параллельных микропрограмм. Второй этап состоит в выполнении формальных процедур построения структурных схем блоков, т.е. сетей автоматов, интерпретирующих микропрограммы, включенные в композицию. Наконец, на третьем этапе выполняется (тоже формальная) процедура определения связей между блоками (см.рисунок).

Э т а п I. Алгоритм ϕ решения заданного класса задач представляется в виде композиции более простых алгоритмов $\phi_1, \ldots, \phi_n$, каждый из которых реализует преобразование над своим массивом данных. Композиция этих алгоритмов изображается в виде параллельной граф-схемы алгоритма.*) Каждому компонентному алгоритму ставится в соответствие операторная вершина. Возможности распараллеливания используются наиболее полно, если граф-схема составляется с учетом следующего правила: если $W_j = \phi_i(W_i)$, то в граф-схеме есть путь из ϕ_i в ϕ_j, не проходящий ни через одну $\phi_k \in \phi$ ($i \neq k$, $j \neq k$). Параллельная граф-схема полностью определяет макроструктуру устройства: функции блоков, реализующих $\phi_1, \ldots, \phi_n$, информационные связи между ними, а также алгоритм управления. Первый этап включает также составление параллельных микропрограмм Φ_i для всех $\phi_i \in \phi$. Составление микропрограммы начинается с определения структуры множества имен, которую естественно выбирать идентичной

*) На рисунке параллельная граф-схема алгоритма изображает композицию процедур структурного синтеза параллельных спецвычислителей.



Граф-схема композиции процедур структурного синтеза спецвычислителей.

структуре массива перерабатываемой информации. Так, например, если производится преобразование матрицы размерами m x n, то удобно иметь $M = N_1 \times N_2$, где $N_1 = \{1,\ldots,n\}$, $N_2 = \{1,\ldots,m\}$. Если в вычислении участвуют вектор и матрица, то множество имен может быть составным $M = M_1 \cup M_2$. Алфавит состояний тоже часто бывает составным. В начале процесса составления микропрограммы бывает известна о с н о в н а я часть алфавита A_0, представляющая числовую информацию задачи. В процессе составления микропрограммы к ней добавляются символы, играющие роль управляющих, указывающих, например, на готовность промежуточных результатов. Такие дополнительные символы удается часто использовать в качестве признака завершения вычисления для упрощения композиционного расширения Φ' микропрограммы. Микропрограмма Φ' считается написанной, если она проверена на детерминированность и отлажена путем моделирования.

Э т а п II заключается в построении структурных схем для всех блоков, реализующих компонентные микропрограммы Φ'_j (j = = I,...,), а также для блока, реализующего управляющую программу. Структура блоков, реализующих Φ'_j, определяется путем формального анализа микропрограммы Φ'_j, вычисления для каждого $m \in M$ именующих функций и выявления связей в соответствии с правилами построения интерпретирующей сети автоматов (с.I9).

Синтез структуры управляющего блока также имеет формальный характер. Однако формализуемые процедуры здесь начинают применяться на стадии составления управляющей микропрограммы по граф-схеме алгоритма композиции. От граф-схемы осуществляется переход к сети Петри управления. Для сетей Петри управления разработан ряд эквивалентных преобразований, сохраняющих алгоритм управления и минимизирующих число вершин сети [36]. По этой минимизированной сети и соответствующей ей микропрограмме Φ_c строится структура блока управления. Хотя она может быть построена с помощью той же процедуры, что и структура для блоков Φ'_j, ее проще получить непосредственно по сети Петри. Другой способ синтеза управления сводится к автоматному представлению с последующей реализацией в программируемых логических матрицах [37].

Э т а п III. Определяются информационные и управляющие связи между блоками. Эта процедура выполняется формально. Для проведения информационных связей исходя из граф-схемы строится граф информационных связей. Он представляет собой ориентированный граф с помеченными дугами. Множеству вершин соответствует множество ϕ, а

дуги между ними проводятся следующим образом. Если в граф-схеме есть путь от ϕ_k к ϕ_l, не проходящий ни через одну $\phi_i \in \phi$ ($i \neq k$, $i \neq l$) и $W_l = \phi_k(W_l)$, то между Φ_k и Φ_l существует связь по данным, т.е. между соответствующими вершинами есть дуга, помеченная W_l. Управляющие связи проводятся между блоком управления Φ_c и вычислительными блоками Φ_j ($j = 1, \ldots, n$) для соединения ячеек, соответствующих клеткам "пуск", "завершение".

Создана основа для автоматизации процесса структурного проектирования специализированных параллельных вычислителей. Дальнейшие усилия должны быть направлены на создание машинных программ.

Заключение

Известно, что темпы развития технологии электронных схем опережают создание математического обеспечения их проектирования. Это в полной мере относится к однородным структурам, реализующим параллельные вычисления над массивами данных. Для них отсутствие методов автоматизированного синтеза является одной из причин, сдерживающих их применение. Исследование путей преодоления отставания математического обеспечения однородных структур привело сначала к разработке алгоритмов параллельных подстановок, а затем на их основе к методу параллельного программирования. Как видно из предыдущего изложения, разработка параллельного программирования потребует еще много усилий. Однако первые опыты применения метода в проектировании реальных устройств обнадеживают.

Дальнейшее развитие параллельного микропрограммирования должно пойти как по пути теоретических исследований свойств параллельных вычислений, так и по пути разработки программного обеспечения автоматизированного проектирования. Особенно важно на данном этапе накопление опыта составления, проверки и моделирования параллельных микропрограмм.

Литература

I. KINNIMENT D.J. Regular Programmable Control Stwebures.-In: VLSI-81 (ed.John P.Gray). London, Academic Press, 1981, p. 193 - 205.

2. КАРЦЕВ М.А., БРИК В.А. Вычислительные системы и синхронная арифметика.- М.: Радио и связь,1981. - 359 с.

3. Однородные микроэлектронные ассоциативные процессоры./Под ред. И.В.Прангишвили. - М.: Сов.радио, 1973. - 280 с.

4. СМИТ Кевин. Матричный процессор, содержащий 96x96 ячеек для обработки изображений в реальном времени.- Электроника, 1980, № 10, с.14-15.

5. МОЛДОВАН Д.И. О разработке алгоритмов для систолических матриц СБИС. - ТИИЭР,1983,т.71, № 1, с.140-149.

6. ASPINALL D. Multi-micro Systems.- In: Future Systems. Pt.2. Infortech.International,1977,p.45-62.

7. ЕВРЕИНОВ Э.В., ХОРОШЕВСКИЙ В.Г. Однородные вычислительные системы. - Новосибирск: Наука, 1978. - 320 с.

8. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные вычислительные системы высокой производительности. - Новосибирск: Наука,1966.

9. ЕВРЕИНОВ Э.В. О макроструктуре элементарных машин вычислительной системы.- В кн.: Вычислительные системы, вып.4. Новосибирск,1962,с.3-28.

10. ЕВРЕИНОВ Э.В., ПРАНГИШВИЛИ И.В. Цифровые автоматы с настраиваемой структурой (однородные среды). -М.:Энергия,1974.-240 с.

11. КАЛЯЕВ А.В. Теория цифровых интегрирующих машин и структур. - М.: Сов.радио,1970. - 471 с.

12. КАЛЯЕВ А.В. Однородные коммутирующие регистровые структуры. - М.:Сов.радио,1978. - 334 с.

13. ФЕТ Я.И. Параллельные процессоры в управляющих системах. - М.:Энергоиздат,1981. - 160 с.

14. Дж.фон НЕЙМАН. Теория самовоспроизводящихся автоматов.- М.: Мир,1971. - 384 с.

15. ALADYEV V.Z. Mathematical Theory of Homogeneous Structures and Their Applications. - Tallin,1980.

16. ЭЙГЕН М., ВАНКЛЕР Р. Игра "Жизнь". - М.: Наука,1979.-92 с.

17. YAMADA H., AMOROSO S.A. Tesrelation Automata.- Information and Control,1969,v.14,N 3,p.299-317.

18. CODD E.F. Cellular Automata. - New York-London: Acad.Press, 1968.- 122 p.

19. HENNIE F.C.Iterative Arrays of Logical Civcuits.- New York: Acad.Press,1961.- 242 p.

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

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

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

23. Методы параллельного микропрограммирования. Под ред. О.Л.Бандман. - Новосибирск: Наука, 1981. - 180 с.

24. СЕРГЕЕВ С.Н. Однородный процессор для обработки сигналов. - В кн.: Вопросы теории и построения вычислительных систем (Вычислительные системы, вып.70). Новосибирск,1977, с.144-154.

25. БАНДМАН О.Л. Асинхронная интерпретация параллельных микропрограмм. - Новосибирск,1981. - 36 с. (Препринт / Институт математики СО АН СССР, ОВС-14).

26. ПИСКУНОВ С.В. Язык микропрограммного описания моделей однородных параллельных вычислительных устройств. - В кн.: Архитектура вычислительных систем с программируемой структурой (Вычислительные системы, вып.82). Новосибирск,1980, с.26-40.

27. АЧАСОВА С.М. Анализ асинхронной интерпретации параллельных микропрограмм. - В кн.: Однородные вычислительные системы из микро-ЭВМ (Вычислительные системы,вып.97). Новосибирск,1983,с.28-53.

28. СЕРГЕЕВ С.Н. Распознавание непротиворечивости алгоритмов стационарных подстановок. - В кн.: Архитектура вычислительных систем с программируемой структурой (Вычислительные системы,вып.82). Новосибирск,1980,с.18-25.

29. PETERSON J.L. Petri Nets and the Modeling of Systems.- London: Prentice-Hall International,1980.- 241 p.

30. HACK M. Analysis for Production Schemata by Petri Nets. - Computer Structure Group, TR-94, Project MAC. MIT, 1972.- 119 p.

31. АНИШЕВ П.А. Один способ анализа корректности граф-схем алгоритмов. - Программирование, 1981, № 1, с.20-28.

32. АНИШЕВ П.А. Редуцируемость сетей Петри. - Программирование, 1982, № 4,с.36-43.

33. BERTHELOT G., ROUCAIROL G. Reduction of Petri-nets. - In: Lecture Notes in Computer Sci.,1976,N 45,p.202-209.

34. ЕСИКОВА Т.Н. Алгоритмы построения множеств достижимых маркирований для анализа свойств сетей Петри. - В кн.: Однородные вычислительные системы из микро-ЭВМ (Вычислительные системы,вып.97). с.53-68.

35. БАНДМАН О.Л., ПИСКУНОВ С.В., СЕРГЕЕВ С.Н. Применение методов параллельного микропрограммирования для синтеза структуры специализированных вычислителей. - Новосибирск,1983. - 31 с. (Препринт / Институт математики СО АН СССР, № 35 (ОВС-18)).

36. БАНДМАН О.Л. Минимизация сетей Петри при синтезе асинхронного параллельного управления. - В кн.: Однородные вычислительные системы (Вычислительные системы,вып.90). Новосибирск,1982,с.3-21.

37. БАНДМАН О.Л. Синтез асинхронного микропрограммного управления параллельными процессами. - Кибернетика,1980, № 1,с. 47.

Поступила в ред.-и
14 ноября 198