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

УДК 681.322.01

О ДЕТЕРМИНИРОВАННОСТИ ПАРАЛЛЕЛЬНЫХ ГРАФ-СХЕМ

П.А. Анишев

Одной из задач, возникающих при проектировании параллельных асинхронных микропрограммных устройств управления, является задача проверки на детерминированность параллельных граф-схем алгоритмов. Эта задача относится к анализу поведения, поэтому при ее решении целесообразно использовать модель взаимодействий процесов в проектируемой системе.

Наиболее удобной моделью, применяемой в синтезе устройств управления, являются сети Петри. Переход от параллельных граф-схем к сети Петри позволяет свести проверку первых на детерминированность к анализу соответствующей сети Петри на живость и безопасность (LS) . Есть попытки решения этой задачи [3,4], хотя для общих сетей Петри она до сих пор не решена. Наиболее широким классом сетей Петри, для которого получено конструктивное решение, являются сети свободного выбора (FC). Опыт показывает, что в терминах FC-сетей можно описывать поведение достаточно широкого класса устройств управления (без арбитров).

В данной работе предлагается использовать LS-условия для FC-сетей, полученные ранее в [2], для построения алгоритма проверки параллельных граф-схем на детерминированность.

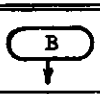
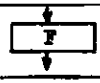
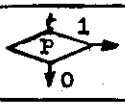
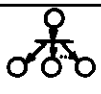
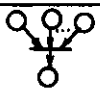
I. Основные определения

Дадим основные определения, связанные с параллельными граф-схемами алгоритмов и сетями Петри (PN), которые моделируют их работу.

I:I. П а р а л л е л ь н а я г р а ф - с х е м а алгоритма представляет собой ориентированный граф: вершины графа соответствуют действиям, а дуги - зависимостям между действиями. Парал-

лельные граф-схемы являются обобщением широко применяемых в программировании граф-схем алгоритмов. Для отображения распараллеливания и слияния процессов добавлены два типа вершин: разветвление и соединение. Таким образом, в параллельных граф-схемах используются следующие типы вершин (см. табл. I, колонка I): начальная (B); конечная (E); функциональная (F); предикатная (P); объединение (U); разветвление (W); соединение (C).

Таблица I

Граф-схемы	PN	Наименование
		Начальная
		Конечная
		Функциональная
		Предикатная
		Объединение
		Разветвление
		Соединение

Дуги задают отношение частичного порядка на множестве вершин [6]. Если две вершины связаны направленным путем, то соответствующие им действия должны выполняться только в предписанном порядке. Если две вершины не имеют направленного пути между собой и не лежат в разных ветвях предикатной вершины, то соответствующие действия могут быть выполнены в любом порядке или параллельно.

ОПРЕДЕЛЕНИЕ. Параллельная граф-схема называется детерминированной, если при одних и тех же входных данных любые допустимые последовательности действий дают один и тот же конечный результат.

Хорошо сформированная (WF)-граф-схема определяется рекурсивно следующим образом [I]:

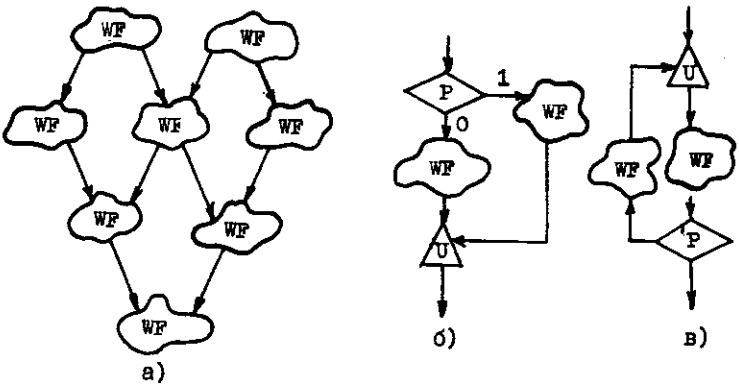


Рис. I

I) отдельная вершина типа F,W и C является WF;

2) граф без циклов, каждая вершина которого есть WF,является WF (рис.I,а);

3) условное ветвление (начинающееся в вершине P и заканчивающееся в вершине U), каждая ветвь которого WF, является WF ; причем ветви не имеют ни входящих, ни исходящих дуг, связывающих их с другими частями граф-схемы (рис.I,б);

4) контур (включающий вершины P и U),тело которого есть WF, является WF; причем тело контура не имеет ни входящих, ни исходящих дуг,связывающих его с другими частями граф-схемы (рис.I,в).

Нарушение вышеперечисленных требований может привести к возникновению таких условий ("запрещенных конфигураций"), при которых граф-схема становится недетерминированной или вычисление становится невыполнимым.

Хорошо сформированная граф-схема не содержит запрещенных конфигураций и является детерминированной [I].

Заметим, что условие WF является достаточным для детерминированности, но не является необходимым. Можно привести примеры детерминированных граф-схем, для которых условие WF не выполняется. Очевидно, что граф-схема на рис.2 детерминированная, хотя ветви, исходящие из предикатных вершин,связаны дугами с другими частями граф-схемы, т.е. нарушено требование 4 в определении WF.Отсюда возникает задача проверки параллельных граф-схем на детерминированность. Поскольку каждой граф-схеме может быть поставлена в

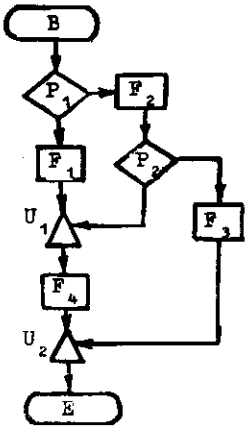


Рис. 2

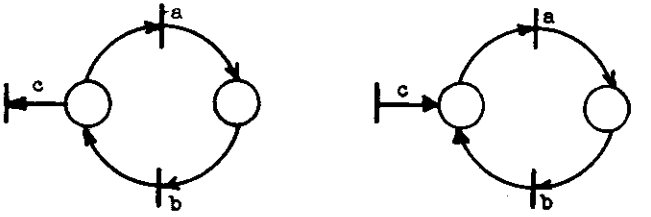
соответствие сеть Петри (см. п.2), на основе которой строится управление вычислением, то проверку ее на детерминированность можно заменить анализом соответствующей сети Петри. Кроме того, анализ сетей Петри имеет самостоятельную ценность, так как они находят широкое применение при исследовании различных вопросов, связанных с параллельными вычислениями.

I.2. С е т ь П е т р и – это двудольный ориентированный граф. Вершины первого типа (изображаемые кружками) соответствуют условиям в системе и называются п о з и ц и я м и; вершины второго типа – события – называются п е р е х о д а м и и обозначаются черточками. Переходы связаны с позициями, и, наоборот, позиции – с переходами. Обозначим: $N = (П, \Sigma, ')$ – сеть Петри; $П$ – множество позиций; Σ – множество переходов; $'$ – отношение, соответствующее дугам. Тогда $'p$ – множество входных переходов позиции p; $'t$ – множество входных позиций перехода t; p' – множество выходных переходов позиции p; t' – множество выходных позиций перехода t.

М а р к и р о в а н и е сети Петри это функция $M: П \to R$, где R – множество натуральных чисел и нуль. Нуль соответствует отсутствию м е т к и в позиции. Позиция, не имеющая метки, называется п у с т о й. Сеть Петри отображает не только статику (взаимосвязь событий и условий), но и динамику моделируемой системы. Динамика (изменение состояний) отображается движением меток по сети, вызываемым срабатыванием возбужденных переходов. Переход в о з б у ж д е н, если в с е входные позиции имеют метки. При с р а б а т ы в а н и и перехода от каждой входной позиции отнимается метка, а к каждой выходной позиции добавляется одна метка. Возбужденные переходы сети срабатывают независимо. Маркирование M' называется д о с т и ж и м ы м из M, если существует последовательность срабатываний, приводящая из M к M'.

Важными характеристиками сети Петри являются живость переходов и безопасность позиций. Переход (при заданном маркировании M) называется ж и в ы м, если существует M' (достижимое из M), при котором он срабатывает. Позиция p называется б е з о п а с н о й

(при заданном маркировании M), если для любого M', достижимого из M, имеем $M'(p) \leq 1$. Сеть называется *живой*, если все переходы ее живые, и *безопасной*, если все позиции безопасные. Живость переходов сети гарантирует то, что события, запланированные в системе (моделью которой является сеть Петри), обязательно наступят. Безопасность позиций обеспечивает согласованную работу взаимосвязанных событий. Обозначим через $A^{\bullet}$ множество всех выходных переходов для множества позиций A, а через ${}^{\bullet}A$ – множество входных переходов. Тогда *дедлок* (D) и *ловушка* (T) определяются как множества позиций сети Петри такие, что ${}^{\bullet}D \subset D^{\bullet}$ и $T^{\bullet} \subset {}^{\bullet}T$ соответственно (рис.3).



а) ${}^{\bullet}D = \{a,b\}$; $D^{\bullet} = \{a,b,c\}$;

б) $T^{\bullet} = \{a,b\}$; ${}^{\bullet}T = \{a,b,c\}$.

Рис. 3.

Можно выделить следующие классы сетей Петри (PN): маркированные графы (MG); автоматные графы (SM); сети свободного выбора (FC); простые сети (SN);

$MG \equiv \forall p(|p^{\bullet}| = |{}^{\bullet}p| = 1)$;

$SM \equiv \forall t(|t^{\bullet}| = |{}^{\bullet}t| = 1)$;

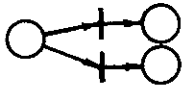
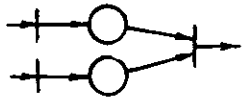
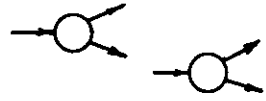
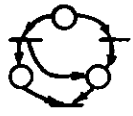
$FC \equiv \forall p(|p^{\bullet}| = 1) \vee (\forall t \in p^{\bullet}(|{}^{\bullet}t| = 1))$;

$SN \equiv \forall t \forall p \in {}^{\bullet}t(|p^{\bullet}| > 1 \Rightarrow$ p единственна);

PN ≡ нет ограничений.

В маркированных графах каждая позиция имеет один вход и один выход. В автоматных графах каждый переход имеет один вход и один выход. В сетях свободного выбора либо каждая позиция имеет один выход, либо каждая выходная дуга является единственным входом в переход. В простых сетях каждый переход имеет не более одной разделяющей входной позиции. Графическое представление органичений приведено в табл. 2.

Таблица 2

Классы	Допустимые фрагменты	Запрещенные фрагменты
SM		
MG		
FC		
SN		

Граф будем называть сильно связанным (SC), если (в отличие от общепринятого) каждая его компонента сильно связана.

2. Моделирование граф-схем сетями Петри

В этом разделе определяются правила перехода от граф-схемы к сети Петри, ставится задача определения детерминированности граф-схем, рассматриваются граф-схемы, моделируемые FC-сетями и обсуждаются способы устранения ресурсных конфликтов.

2.1. Правила перехода от граф-схемы к сетям Петри приведены в табл. I (колонки I и 2). Каждая вершина граф-схемы заменяется фрагментом сети Петри (рис.4). Если две вершины граф-схемы смежны, т.е. существует дуга (v_1, v_2), то выходная позиция фрагмента, соответствующего v_1, отождествляется с входной позицией фрагмента, соответствующего v_2. Каждая граф-схема имеет одну начальную и одну конечную вершины, которые (для целей последующего анализа) соединяются, т.е. в полученной сети Петри выходная позиция оператора E отождествляется с входной позицией оператора B.

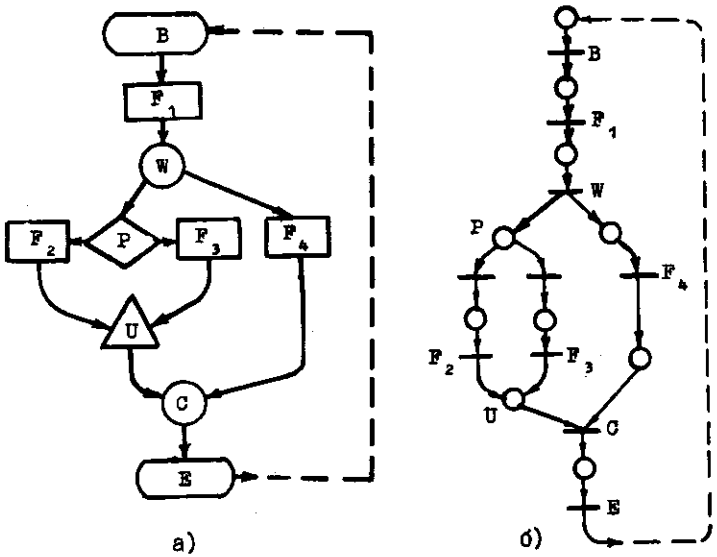


Рис. 4

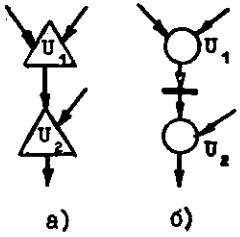


Рис. 5

Если смежны два оператора объединения, то при переходе к сети Петри между позициями, соответствующими этим операторам, нужно поставить переход (рис.5).

Правила перехода определены так, что сеть Петри, полученная из детерминированной граф-схемы, является живой и безопасной FC-сетью [2]. Это позволяет решать задачу определения детерминированности граф-схемы следующим образом: перейти к сети Петри; анализировать полученную FC-сеть на живость и безопасность.

2.2. Ресурсные конфликты в граф-схемах. Действия, выполняемые параллельно, могут вступать в конфликты при обращении к одному и тому же ресурсу (одновременная запись разных результатов в один регистр, обращение к одному

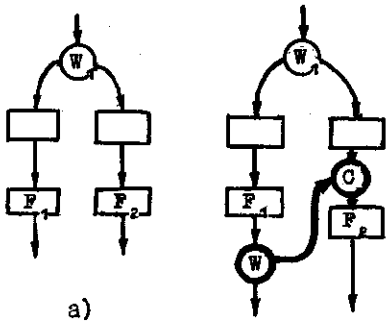


Рис. 6

вычислителю и т.п.). Для устранения ресурсных конфликтов существует два метода: установление фиксированных приоритетов и арбитраж. Для того чтобы для двух конфликтующих действий F_1 и F_2 установить приоритеты (например, F_1 F_2), выход оператора F_1 соединяется со входом F_2 . При этом используются дополнительно два оператора W и C и дуга (W,C) (рис.6). Очевидно, что введение в параллельные ветви граф-схемы таких дополнительных дуг не нарушает детерминированности.

Введение арбитра (A) между двумя конфликтующими действиями (переходами) заключается в следующем. В сеть Петри вводится дополнительная позиция с меткой (рис.7); любой из возбудившихся переходов "захватывает" ресурс, а после срабатывания отдает ресурс - возвращает метку в позицию, давая возможность сработать другому переходу.

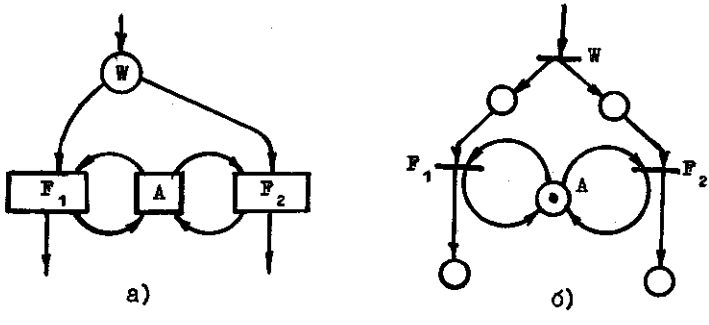


Рис. 7

Введение арбитров выводит сеть из класса FC-сетей: если дополнительные позиции вводятся между двумя переходами, то получаются простые сети (SN); а между тремя переходами - общие сети Петри (см. табл. 2). Задача определения правильности общих сетей Петри до сих пор не решена. Для FC-сетей (их во многих прак-

тических применениях оказывается достаточно для моделирования поведения различных систем) эта задача решена Хаком (см.[2]). FC-сети - это самый сложный класс сетей Петри, для которого получены условия живости и безопасности. Поэтому в дальнейшем мы будем рассматривать только такие сети. Однако если арбитры вводить в параллельные ветви граф-схемы между неупорядоченными (см. раздел I.I) операторами типа F, то условия живости и безопасности соответствующей сети Петри сохраняются. Действительно, если каждый из переходов F_1 и F_2 (см. рис.7) живой, то введение маркированной позиции A не нарушает живости этих переходов (на остальные переходы позиция A не влияет), позиция A безопасна, так как F_1 и F_2 не могут сработать одновременно.

Таким образом, анализ граф-схем с арбитрами проводится в три этапа:

- удаляются все арбитры из сети;
- анализируется оставшаяся сеть;
- проверяется правильность введения арбитров.

3. Анализ FC-сетей на живость и безопасность

В этом разделе рассматривается алгоритм анализа FC-сетей, приведена теорема, на которой основан этот алгоритм, и получена оценка его сложности.

Предварительно введем несколько определений.

ОПРЕДЕЛЕНИЕ. Размещение автоматных графов (State Machine) — SM-размещение на FC-сети $(\Pi, \Sigma, \cdot)$ это функция $f:\Sigma \to \Pi$ такая, что $\forall t \in \Sigma\ (f(t) \in {}^{\bullet}t)$.

ОПРЕДЕЛЕНИЕ. Размещение маркированных графов (Marked Graph) — MG-размещение на FC-сети $(\Pi, \Sigma, \cdot)$ это функция $g:\Pi \to \Sigma$ такая, что $\forall p \in \Pi (g(p) \in p^{\bullet})$.

При заданном SM-размещении определим SM-редукцию FC-сети по шагам:

1) удаляем неразмещенные позиции $(\Pi - f(\Sigma))$;

2) удаляем все переходы, у которых все выходные позиции удалены;

3) удаляем все позиции, у которых хотя бы один выходной переход удален;

4) повторяем шаги 2 и 3 до тех пор, пока применим хотя бы один из них.

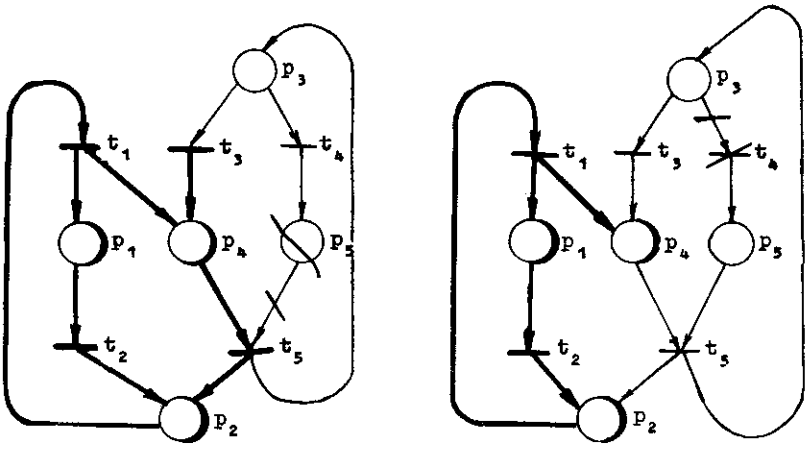


Рис. 8

Оставшаяся часть сети Петри называется SM - р е д у к - ц и е й (рис.8,а). SM-редукция выделена жирным, неразмещенные позиции зачеркнуты. Определение MG-редукции является двойствен - ным по отношению к SM-редукции: слово "выходной" заменяется на "входной", "позиция" - на "переход", "переход" - на "позиция". На рис.8,б показана MG-редукция. Неразмещенные переходы зачеркнуты. Проследим последовательность удаления переходов и позиций для SM-редукции на рис.8,а.

Т а б л и ц а 3

Номер шага	I	2	3	4
удаленные вершины	p_5	t_4	p_3	шаги 2 и 3 неприменимы

Для MG-редукции на рис. 8,б аналогично:

Т а б л и ц а 4

Номер шага	I	2	3	2	3	4
удаленные вершины	t_4	p_5	t_5	p_3	t_3	шаги 2 и 3 неприменимы

Алгоритм Хака служит для определения: имеет ли FC-сеть живое и безопасное маркирование? Он основан на следующей теореме:

ТЕОРЕМА [2]. Для FC-сети эквивалентны следующие утверждения:

1) сеть имеет живое и безопасное маркирование;

2) каждая SM-редукция есть SCSM, редукции покрывают сеть, нет пустых редукций;

3) каждая MG-редукция есть SCMG, редукции покрывают сеть, нет пустых редукций.

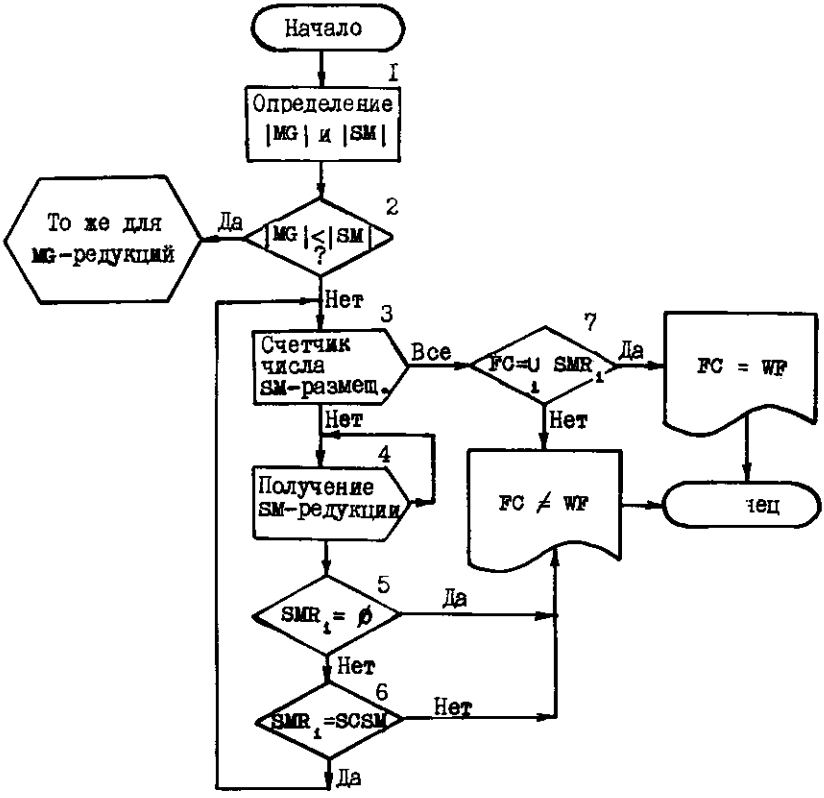


Рис. 9

Каждое из утверждений (2 или 3) этой теоремы может служить основой алгоритма, который заключается в получении всех SM-редукций (MG-редукций) и проверке, покрывают ли эти редукции сеть. Если какая-нибудь редукция пуста или не является сильно связанным автоматным графом SCSM (SCMG) , то сеть не имеет живого и безопасного маркирования (см.блок-схему на рис.9). Сложность алгоритма в основном определяется числом SM-размещений (MG-размещений); SM - размещение заключается в удалении входных дуг (кроме одной) у всех переходов. Поэтому число SM-размещений равно $|SM| = \prod_{t\in\Sigma} |{}^{\cdot}t|$. Число MG-размещений аналогично $|MG| = \prod_{p\in\Pi} |p^{\cdot}|$.Сравнивая числа $|SM|$ и $|MG|$, можно выбрать режим работы алгоритма. Сложность алгоритма определяется числом основных циклов (3,4,5,6,3) и равна произведению числа разветвлений ($\Pi(p^{\cdot})$) или числа соединений ($\Pi({}^{\cdot}t)$) . Если принять $|p^{\cdot}| = 2$, т.е. разветвление по двум направлениям, то общее число циклов равно 2^n , где n - число разветвлений. Сеть с числом разветвлений ~ 30 на ЭВМ с быстродействием порядка 10^6 опер/сек будут обрабатываться около часа.

Экспоненциальная сложность алгоритма Хака накладывает жесткое ограничение на число вершин анализируемой сети, поэтому представляется целесообразным ввести в алгоритм правила сокращения сетей Петри, сохраняющие свойства живости и безопасности, например, [3]. Так как введение арбитров и применение правил сокращения расширяет класс сетей (алгоритм Хака может применяться только к сетям свободного выбора), необходимо разработать алгоритм анализа общих сетей Петри с небольшим (~ 20) числом позиций.

В дальнейшем желательно ввести в алгоритм правила декомпозиции (выделения "хорошо сформированных" кусков) сети Петри. Замена выделенных подграфов на "макропереходы" или "макропозиции" [5] позволит сократить сеть и облегчить последующий анализ.

Литература

I. HUMBAUGH J.E. A Parallel Asynchronous Computer Architecture for Data Flow Programs. Project MAC, M.I.T., Cambridge, Massachusetts, 1975.

2. HACK M. Analysis of Production Schemata by Petri Nets. Project MAC, M.I.T., Cambridge, Massachusetts, 1972.

3. BERTHELOT G., ROUCAIROL G. Reduction of Petri-nets. Lecture Notes in Computer Sciences, N 45, p.202-209.

4. BEST E., SCHMID H.A. Systems of Open Paths in Petri Nets. Lecture Notes in Computer Sciences, N 32,p.186–193.

5. ANDRE C.,BOERI F., MARIN J. Synthèse et Realisation des Systèms Logiques a Evolutions Simultanées.–"Revue francaise d'Automatique Informatique Recherche Operationnelle",1976, vol.10, N 4, Avril, p.67–86.

6. БАНДМАН О.Л., ПИСКУНОВ С.В., СЕРГЕЕВ С.Н. Задачи параллельного микропрограммирования. –Настоящий сборник, с.3–24.

Поступила в ред.-изд.отд.
27 января 1978 года