ВВЕДЕНИЕ

Внедрение в практику проектирования вычислительных устройств средств автоматизации проектирования на базе персональных ЭВМ (ПЭВМ) позволяет повысить качество проектных решений, уменьшить стоимость, трудоемкость и сроки разработки. Системы автоматизации проектирования (САПР) позволяют осуществить цикл сквозного проектирования, включающий этапы системного, алгоритмического, логического и технического проектирования. При этом процесс проектирования носит итерационный характер и состоит из процедур синтеза и анализа. Анализ вычислительных устройств производится с помощью моделирующих систем, в основу которых положены алгоритмы, ориентированные на обработку той или иной модели устройства.

В учебном пособии рассматриваются теоретические и практические аспекты моделирования. Формулируются цели и задачи моделирования цифровых устройств. Базовым математическим аппаратом авторами выбран аппарат сетей Петри, обладающий мощными моделирующими возможностями аппаратно-программных средств. Разрабатываются методики моделирования устройств и программ на основе сетей Петри.

В практическом плане рассматриваются программы логического моделирования цифровых устройств Micro-Logic (разработка фирмы "Spectrum Software") и PC-LOGS (разработка фирмы "Personal CAD System, Inc."), входящая в состав САПР P-CAD.

Глава 1 написана Афанасьевым А.Н. и Шишкиным В.В. совместно, главы 2 и 4 написаны Афанасьевым А.Н., а главы 3 и 5 - Шишкиным В.В.

1. СИСТЕМЫ ПРОЕКТИРОВАНИЯ И МОДЕЛИРОВАНИЯ ЦИФРОВЫХ ВЫЧИСЛИТЕЛЬНЫХ УСТРОЙСТВ

1.1. Автоматизация проектирования средств вычислительной техники

Необходимость использования САПР в проектировании средств вычислительной техники (СВТ) определяется усложнением устройств, увеличением числа проектировщиков, возрастающей стоимостью разработки. Применение САПР СВТ позволяет:

- снизить трудоемкость проектных работ, производства и отладки, уменьшить затраты на эксплуатацию, модернизацию и замену;
- уменьшить вероятность ошибок за счет повышения точности расчетов, контроля за появлением ошибок и их оперативное исправление, системного согласования частных проектных решений, возможности эффективного анализа проектных решений;
- повысить техническую эффективность проектируемых устройств и систем за счет оперативного анализа вариантов проектных решений, управления эффективностью системы при проектировании, обобщения опыта проектирования и отладки, оперативной коррекции проектных решений;
- уменьшить возможность морального старения за счет сокращения сроков проектирования, унификации проектных решений.

В проектировании СВТ выделяются этапы системного, алгоритмического, логического и технического проектирования.

На этапе системного (структурного) проектирования решаются задачи:

- определяется состав и основные характеристики функциональных блоков;
- разрабатываются технические характеристики, способ взаимодействия функциональных блоков и их сопряжения между собой;

- определяется структура вычислительных процессов и их количественные характеристики.

На этапе алгоритмического проектирования решаются задачи:

- разрабатывается (или выбирается) система команд;
- разрабатываются алгоритмы микропрограмм (или временные диаграммы) функциональных блоков.

На этапе логического проектирования решаются задачи:

- разрабатываются логические структуры функциональных блоков;
- производится анализ логических схем;
- разрабатываются тестовые или функциональные средства контроля и диагностики.

На этапе технического (конструкторского) проектирования решаются задачи:

- компоновка логических элементов в подсхемы заданной сложности;
- размещение структурных компонент в пространстве и на плоскости;
- соединение компонент друг с другом (трассировка);
- изготовление производственной и эксплуатационной документации.

1.2. Анализ вычислительных устройств методом моделирования

Анализ вычислительных устройств базируется на характеристиках устройств, полученных в ходе проведения определенных экспериментов. Однако натурный эксперимент с вычислительным устройством часто невозможен или нецелесообразен. Причиной этому могут служить:

- отсутствие устройства в процессе проектирования;
- значительная стоимость натурного эксперимента;
- значительная модернизация устройства в процессе эксперимента;
- требование значительного времени для проведения эксперимента;
- непредсказуемость результатов эксперимента.

В этих случаях прибегают к модельным экспериментам или моделированию. По международному стандарту ISO 2382/1-84 **моделирование** (simulation) - это представление различных характеристик поведения физической или абстрактной системы с помощью другой системы. Таким образом, модель - это физическая или абстрактная (математическая) система, с помощью которой мы можем получить характеристики другой, в общем случае еще не существующей системы. Математические модели представляют собой функции, выражающие значения характеристик систем через их

параметры, и определяются целью анализа, например, модели функционирования, быстродействия и т.д. К математическим моделям предъявляются требования универсальности, адекватности, точности и экономичности.

Степень универсальности модели характеризует полноту отображения в модели свойств реального объекта.

Точность модели оценивается степенью совпадения характеристик реального объекта и значений характеристик, полученных с помощью модели.

Адекватность модели - способность отображать заданные свойства объекта не ниже заданной величины.

Экономичность модели характеризуется затратами вычислительных ресурсов: машинного времени и памяти на ее реализацию.

Требования высоких универсальности, точности и адекватности, с одной стороны, и высокой экономичности, с другой, являются противоречивыми.

Классификация математических моделей отражается в табл. 1.1.

Таблица 1.1

Признак классификации	Тип математической модели
Характер отображаемых свойств объекта	Функциональные, структурные
Принадлежность к иерархическому уровню	Микроуровни, макроуровни, метауровни
Способ представления свойств объекта	Аналитические, алгоритмические, имитационные
Способ получения модели	Теоретические, эмпирические

В процессе автоматизированного проектирования моделирование проектируемого устройства выполняется многократно и на различных уровнях детализации описания.

Для исследования СВТ применяются следующие методы:

- аналитические, заключающиеся в составлении математической модели вычислительного устройства (например, в виде системы уравнений);
- статистические, заключающиеся в составлении математико-статистической модели устройства или системы;

- описательные, обеспечивающие исследования на содержательном уровне;

- экспериментальные, основанные на натурных экспериментах с устройством или его физической моделью;

- имитационные, заключающиеся в проведении экспериментов с имитационной моделью системы или устройства в виде программы.

Формализация процесса обработки информации и определение аналитических зависимостей характеристик системы или вычислительного процесса от их параметров, т.е. построение математической модели функционирования системы или устройства, имеют ограничения с точки зрения полноты описания работы реального устройства.

Традиционно модели цифровых вычислительных устройств строились в форме алгоритмических описаний их функционирования на основе представлений об исследуемом объекте как о системе с дискретными событиями. Вне зависимости от языковых средств при создании алгоритмических моделей используется функциональный подход к описанию цифровых устройств, при котором модель представляется в виде совокупности развивающихся во времени взаимодействующих вычислительных процессов.

Метод имитационного моделирования выделяется своей эффективностью и универсальностью при решении задач синтеза и анализа цифровых вычислительных устройств. В широком смысле имитационное моделирование может быть определено как процесс представления динамической системы моделью для получения информации об этой системе путем проведения экспериментов над ее моделью. Применение этого метода полезно в том случае, когда исследуемое устройство не поддается изучению аналитическими методами, а прямое экспериментирование с ним нецелесообразно или невозможно.

Суть этого метода в том, что весь процесс функционирования вычислительного устройства рассматривается как совокупность процессов функционирования его составных частей.

Устройство разбивается на части, которые позволяют достаточно просто формально описать поведение каждой из них, и затем описываются взаимосвязи между отдельными частями устройства. Удобнее всего при этом разбивать вычислительное устройство на такие составные части, которые соответствовали бы отдельным узлам и блокам устройства, тогда модель будет состоять из частей, имитирующих поведение устройств и связей в реальном объекте, и в силу этого имитировать поведение самого объекта моделирования.

Используемые для построения имитационных моделей вычислительных устройств и систем алгоритмические языки благодаря гибкости и доступности позволяют отражать в моделях многие детали и

свойства структур и функций вычислительных устройств, которые опускаются или утрачиваются в математически строгих моделях. Свойственные имитационным моделям реалистичность, адекватность, простота основаны на использовании для их построения всех имеющихся представлений об объекте исследования как теоретического, так и эвристического характера.

1.3. Характеристика имитационного моделирования

Моделирование вычислительных устройств производится на следующих уровнях: системном, регистровом, логическом и электрическом.

Целью моделирования на системном уровне является определение эксплуатационных характеристик проектируемого устройства в целом (быстродействия, надежности и т.д.). Модель представляет собой описание структуры и порядка функционирования устройства в виде программы на одном из алгоритмических языков или языков имитационного моделирования. Использование в этом случае специальных языков имитационного моделирования предпочтительнее в силу наличия в них специальных лингвистических средств управления процессом моделирования, работы с псевдослучайными последовательностями, сбора статистических характеристик, организации параллельных процессов в системе и т.д. Кроме этого языки имитационного моделирования обеспечивают значительную автоматизацию при программировании имитационной модели.

На регистровом уровне моделирования исследуются следующие вопросы:

- определение временных параметров программ (скорость выполнения команд, цепочек команд, программ);

- определение частоты использования аппаратных средств;

- определение степени загрузки трактов передачи данных и распределение этой загрузки по времени.

Модель системы на уровне регистровых передач может быть использована как инструментальное средство для отладки математического обеспечения проектируемой системы.

На этапе моделирования логических элементов исследуется логико-временное поведение устройства, при этом решаются следующие основные задачи:

- проверка правильности логической структуры;
- сравнение характеристик различных вариантов логических схем;
- выявление состязаний и риска сбоя;
- оценка средств контроля и диагностики.

В моделировании на уровне анализа электронных схем предметом рассмотрения являются элементы типа транзисторов, резисторов и др. По характеристикам элементов и связям между ними вычисляются значения токов, и напряжений в схеме, параметры переходных процессов в элементах.

В моделирующих программах с помощью математических моделей имитируется процесс ввода, обработки и вывода данных проектируемого устройства. Процедуры получения математической модели объекта и имитации его функционирования на этой модели называются имитационным моделированием.

При имитационном моделировании прежде всего возникает необходимость в описании объекта проектирования. Это функционально-структурное описание обычно выполняется первоначально на естественном языке, включает в себя ряд формул и содержит значительную по объему графическую информацию. Подобное описание не формализовано, поэтому в практике имитационного моделирования особое место занимают языки моделирования, позволяющие формализовать запись описания произвольной структуры вычислительного устройства и алгоритма его функционирования.

Примерами языков имитационного моделирования на системном уровне являются GPSS, СИМУЛА, ПРОЛОГ, СЛЭНГ, на регистровом уровне - CDL, язык регистровых передач.

2. МОДЕЛИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ УСТРОЙСТВ НА БАЗЕ СЕТЕЙ ПЕТРИ

2.1. Элементы теории сетей Петри

Аппарат сетей Петри [2] является инструментом событийно-имитационного моделирования СВТ. С помощью сетей Петри можно получить качественные и количественные характеристики структуры и динамического поведения моделируемой системы.

Ниже рассматриваются три формы представления сетей Петри: графовая, структурная и матричная.

Графом сети Петри является двудольный ориентированный мультиграф $G=(V,A)$, где $V=\{v_1,v_2,...,v_n\}$ - множество вершин, а $A=\{a_1,a_2,...,a_m\}$ - множество дуг. Множество V состоит из двух подмножеств $V=P \cup T$ (P - подмножество позиций, T - подмножество переходов), причем $P \cap T=0$. Дуги соединяют позиции и переходы. Разрешены соединения вида:

В первом случае позиция P_i является входной для перехода T_j : $P_i \in I(t_j)$, а переход t_j считается выходным для позиции P_i: $t_j \in O(P_i)$. Во втором случае - наоборот: $P_i \in O(t_j)$ и $t_j \in I(P_i)$.

Дуга $a_i \in A$ связывает две вершины v_m, $v_n \in P$ $a_i = (v_m, v_n)$, причем $v_m \in P$, $v_n \in T$, либо $v_m \in T$, $v_n \in P$.

Пример сети Петри приведен на рис. 2.1.

Структурным (теоретико-множественным) представлением сети Петри является четверка множеств $C=(P, T, I, O)$, I и O - функции входов и выходов, заданные на отображениях вида: $I:T \rightarrow P$, $O:T \rightarrow P$. Функции входов и выходов представляют собой комплекты позиций. Комплект является обобщением множества, в которое включены многократно повторяющиеся элементы.

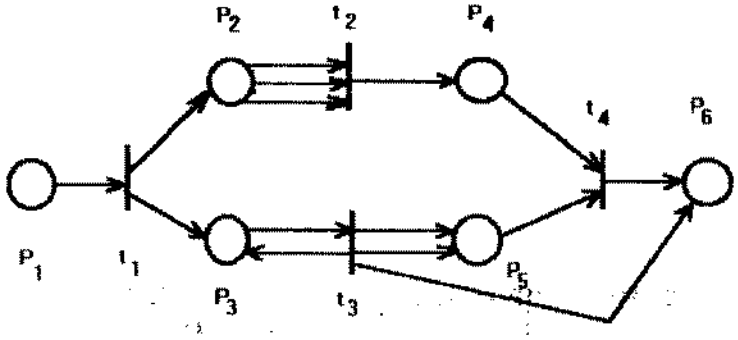


Рис. 2.1

Структурное представление сети Петри рис. 2.1. имеет вид:

$P = \{p_1, p_2, ..., p_6\}$, $T = \{t_1, t_2, t_3, t_4\}$,
$I(t_1) = \{P_1\}$, $O(t_1) = \{P_2, P_3\}$,
$I(t_2) = \{P_2, P_2, P_2\}$, $O(t_2) = \{P_4\}$,
$I(t_3) = \{P_3\}$, $O(t_3) = \{P_3, P_5, P_5, P_6\}$,
$I(t_4) = \{P_4, P_5\}$, $O(t_4) = \{P_6\}$.

В матричной форме сеть Петри представляется матрицами следования F и предшествования B. Элементы матриц F и B определяются по правилам:

$$f_{ij} = \begin{cases} n, \text{ если } t_i \in O(P_j) \text{ n раз}, \\ 0, \text{ если } t_i \notin O(P_j) ; \end{cases}$$

$$b_{ij} = \begin{cases} m, \text{ если } t_i \in I(P_j) \text{ m раз}, \\ 0, \text{ если } t_i \notin I(P_j) >. \end{cases}$$

Матрицы F и B для сети Петри рис. 2.1. имеют вид:

t\p	1	2	3	4	5	6
1	1					
2		3				
3			1			
4				1	2	

t\p	1	2	3	4	5	6
1		1	1			
2				1		
3			1		2	1
4						1

Для моделирования используются маркированные сети Петри.

Маркировка μ есть присвоение фишек позициям сети Петри. Фишка (аналогично позициям и переходам) есть примитивное понятие сетей Петри. Количество и положение фишек при выполнении сети может изменяться.

Маркировка μ сети Петри C= (P,T,I,O) есть функция, отображающая множество позиций P в множество неотрицательных целых чисел N: μ: P → N.

Маркировка определяется как вектор $\mu = (\mu_1, \mu_2, ..., \mu_n)$, где $n=|P|$, $\mu_i \in N$, $i \in \overline{(1, n)}$. Вектор μ определяет для каждой позиции P_i количество фишек в этой позиции.

Маркированная сеть M = (C, μ) есть совокупность структуры C и маркировки μ.

Таким образом, маркирование сети интерпретирует состояние системы, динамика изменения состояний моделируется движением меток в сети.

На графе фишки изображаются точками в позициях сети. Пример маркированной сети приведен на рис.2.2.

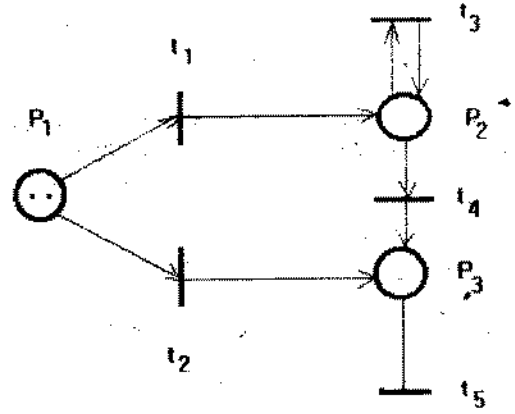


Рис. 2.2

Вектор маркирования для этой сети имеет вид:

$$\mu = (2,0,0).$$

Выполнением сети управляет количество и распределение фишек в сети. Сеть выполняется посредством запуска переходов. Переход может запускаться в том случае, когда он разрешен. Переход считает-

ся разрешенным, если каждая из его входных позиций имеет число фишек по крайней мере равное числу дуг из позиции в переход. Кратные фишки необходимы для кратных дуг. Переход запускается удалением всех разрешающих фишек из его входных позиций и последующим помещением в каждую из его выходных позиций фишек, число которых для одной выходной позиции определяется кратностью дуг, идущих из перехода в позицию. На рис. 2.3. показан переход до (слева) и после (справа) запуска.

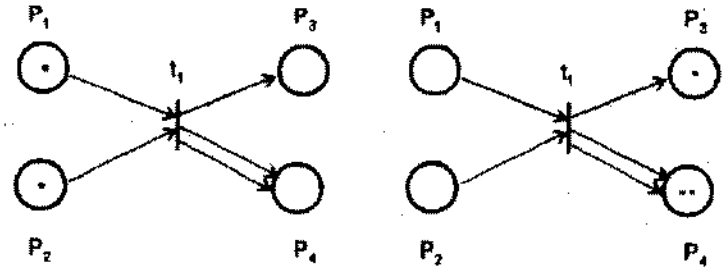


Рис.2.3

Состояние сети определяется ее маркировкой. Запуск перехода изменяет состояние сети и соответственно маркировку. Изменение состояния, вызванное запуском перехода, определяется функцией изменения δ. При применении функции δ к маркировке μ и переходу t_j образуется маркировка $\mu' : \mu' = (\mu, t_j)$.

Пусть дана сеть Петри C = (P, T, I, O, μ) с начальной маркировкой μ^0.

Эта сеть может быть выполнена посредством запусков последовательности переходов $t_{j1}, t_{j2}, \ldots, t_{jk}$, при этом образуется последовательность маркировок $\mu^0, \mu^1, \ldots, \mu^k$. Указанные последовательности связаны соотношением $\mu^{k+1} = \delta(\mu^k, t_{jk})$. Имея последовательность переходов и начальную маркировку, легко получить последовательность маркировок (и наоборот).

Пусть некоторый переход t_j в маркировке μ разрешен и запущен. Результат запуска есть новая маркировка μ'. μ' является непосредственно достижимой из μ. Это понятие распространяется на множество достижимых маркировок сети Петри.

Множество достижимости R(C, μ) является множеством всех маркировок, достижимых из μ'. Для сети Петри рис. 2.4 $R(C, \mu) = \{(1, 0, n), (0, 1, n), n \geq 0\}$.

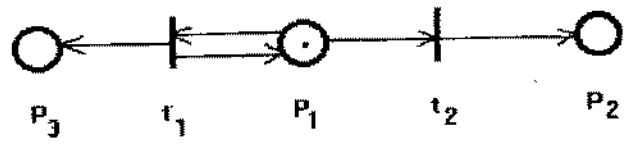


Рис. 2.4

Наглядное изображение возможных состояний и изменений сети дает граф достижимых маркирований (дерево достижимости). Вершинам графа соответствуют маркировки (в корне находится начальная маркировка), а дугам - разрешенные запускаемые переходы. Каждая вершина в процессе построения дерева относится к одному из 4 типов: граничная, терминальная, дублирующая, внутренняя. Граничными являются вершины, еще не обработанные алгоритмом, затем они превращаются в одни из трех других типов вершин. Терминальной является вершина, соответствующая тупиковой маркировке, в которой ни один из переходов не разрешен. Дублирующей является вершина, маркировка которой уже встречалась ранее в дереве достижимости. Внутренней является вершина с маркировкой, в которой разрешен хотя бы один переход. Бесконечное число фишек в какой-либо позиции обозначается через w.

Содержательное описание алгоритма построения дерева достижимости состоит в следующем.

Шаг 1. Корню дерева ставится в соответствие начальная маркировка, которая объявляется текущей.

Шаг 2. Определяются разрешенные в текущей маркировке переходы, их именами помечаются дуги, идущие из текущей вершины.

Шаг 3. Определяются непосредственно достижимые маркировки, полученные при срабатывании в текущей маркировке. Им ставятся в соответствие вершины дерева, в которые идут дуги, нагруженные соответствующими именами переходов.

Шаг 4. Определяется тип вершины. Если все вершины терминальные, дублирующие или внутренние, то построение дерева достижимости заканчивается. В противном случае шаги 2-4 алгоритма повторяются.

На рис. 2.5 изображены сеть Петри и ее дерево достижимости. Заметим, что для сети с другой начальной маркировкой дерево достижимости будет также другим.

Рассмотрим свойства маркированных сетей Петри, с их помощью производится анализ моделирующих систем.

1. Безопасность. Позиция $p_i \in P$ сети Петри C = (P, T, I, O, μ) с начальной маркировкой μ^0 является безопасной, если $\mu'(p_i) <= 1$ $\forall \mu' \in R(C, \mu^0)$. Т.е. позиция является безопасной, если число фишек

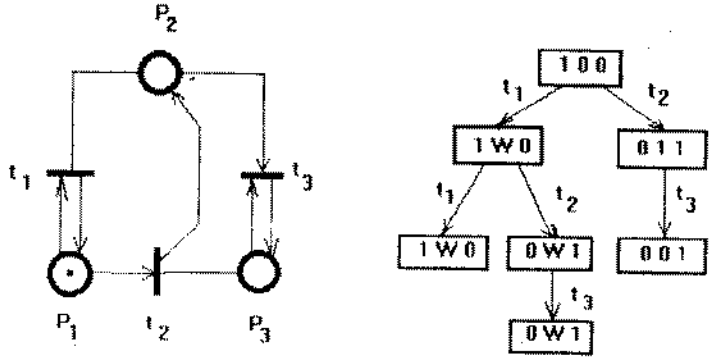


Рис. 2.5

в ней в любой маркировке из множества достижимых маркировок не превышает 1.

Сеть является безопасной, если все ее позиции безопасны.

2. Ограниченность. Является более общим свойством, чем безопасность. Позиция $p_i \in P$ сети Петри $C = (P, T, I, O, \mu)$ с начальной маркировкой μ^0 является ограниченной, если $\mu'(p_i) <= k$ $\forall \mu' \in R(C, \mu^0)$, где k - целое число. Т.е. позиция является k-ограниченной, если число фишек в ней в любой маркировке из множества достижимых маркировок не превышает некоторого целого числа k.

Сеть является k-ограниченной, если все ее позиции k-ограниченны.

3. Сохранение. Сеть Петри $C=(P,T,I,O)$ является строго сохраняющей, если $\sum_{p_i \in p} \mu'(p_i) = \sum_{p_i \in p} \mu^0(p)$. Т.е. сеть является строго сохраняющей, если суммарное число фишек во всех позициях во всех достижимых маркировках постоянно.

4. Живость (активность). Переход $t_j \in T$ сети Петри $C = (P,T,I,O,\mu)$ с начальной маркировкой μ^0 является живым, если $\exists \mu' \in R(C, \mu^0)$, в которой он разрешен и, следовательно, может быть запущен. Сеть жива, если все ее переходы живые.

Более строгое определение активности перехода связано с введением классификации уровней активности:

а) переход t_j обладает активностью уровня 0, если он никогда не может быть запущен;

б) переход t_j обладает активностью уровня 1, если он потенциально запустим, т.е. существует такая $\mu' \in R(C, \mu^0)$, что t_j разрешен в μ';

в) переход t_j обладает активностью уровня 2, если для всякого целого n существует бесконечная последовательность запусков, в которой t_j присутствует по крайней мере n раз;

г) переход t_j обладает активностью уровня 3, если существует бесконечная последовательность запусков , в которой t_j присутствует неограниченно часто;

д) переход t_j обладает активностью уровня 4, если для всякой $\mu' \in R(C, \mu^0)$ существует такая последовательность запусков $= t_{j1}$, t_{j2}, ..., t_{jk}, что t_j разрешен в $\delta(\mu', \sigma)$.

На рис. 2.6 показана сеть Петри, в которой переход t_0 пассивен, т.к. он никогда при данной начальной маркировке не может быть запущен (активность уровня 0);

- переход t_1 можно запустить только один раз (активность уровня 1);

- переход t_2 может быть запущен произвольное число раз, но это число зависит от числа запусков t_3 (активность уровня 2);

- переход t_3 можно запускать бесконечное число раз (активность уровня 3), но t_3 не обладает активностью уровня 4, т.к. после запуска t_1 t_3 запустить больше нельзя.

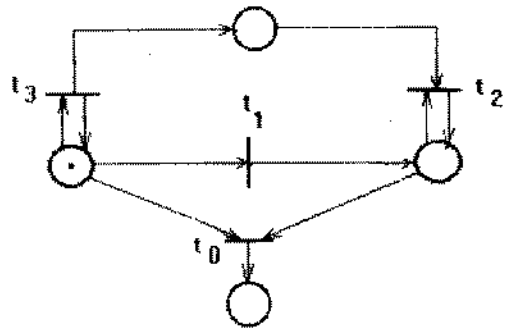


Рис. 2.6

В некоторых случаях удается оптимизировать сеть, разумеется, не изменяя ее поведения. Такое изменение сети заключается в удалении пассивных переходов, которые никогда нельзя запустить, или пассивных позиций, которые никогда не могут быть маркированы, или переопределении некоторых переходов.

Существуют два основных метода анализа сетей Пети. Первый связан с определением свойств сети по дереву достижимости, второй - с решением матричных уравнений.

Определение свойств безопасности, ограниченности и сохранения связано с анализом вершин дерева достижимости, живости - с анализом дуг. Из анализа дерева достижимости рис.2.5 можно сделать вывод о том, что сеть неограничена, строго несохраняющая и живая.

На практике часто рассматривается свойство нестрогого сохранения. В этом случае определяется взвешенная сумма фишек, которая должна быть постоянна для всех достижимых маркировок. Взвешенная сумма вычисляется путем умножения элемента вектора маркирования на соответствующий элемент вектора весов. Если сумма одинакова для каждой достижимой маркировки, то сеть является сохраняющей по отношению к данному весу.

Для того, чтобы сеть, имеющая маркировку с позицией w , была сохраняющей, необходимо иметь вес этой позиции равный нулю. Это означает, что число фишек в данной позиции не важно.

По дереву достижимости просто решается задача достижимости: определяется путь из некоторой вершины дерева с маркировкой в вершину с достижимой маркировкой. Если такого пути не существует, то данная маркировка является недостижимой для сети с данной начальной маркировкой. Возможно, для другой начальной маркировки такой путь существует, т.е. задача достижимости решается положительно.

Матричный анализ сетей Петри основан на использовании матриц следования F и предшествования B. Каждая матрица имеет m строк (в соответствии с количеством переходов) и n столбцов (в соответствии с количеством позиций). $F[i,j] = (p_j, I(t_i))$ определяет входы в переходы, а $B[i,j] = (p_j, O(t_i))$ - выходы из переходов.

Таким образом, определение маркированной сети в матричной форме имеет вид

$$C = (P, T, F, B, \mu^0).$$

Обозначим через e[j] m - вектор-строку, содержащий везде нули, кроме j- й компоненты. Будем представлять переход t_j вектором e[j]. Переход t_j в маркировке μ разрешен, если $\mu >= e[j]*F$. Результат запуска t_j в μ записывается в виде $\delta(\mu, t_j) = \mu - e[j]*F + e[j]*B = \mu + e[j]*(-F+B) = \mu + e[j]*D$, где D = B - F - составная матрица изменений.

Для последовательности запусков переходов $\sigma = t_{j1}, t_{j2}, \ldots, t_{jk}$ имеем $\delta(\mu, \sigma) = \mu + e[j_1]*D + e[j_2]*D + \ldots + e[j_k]*D = \mu + (e[j_1] + e[j_2] + \ldots + e[j_k])*D = \mu + f(\sigma)*D$. Вектор $f(\sigma) = e[j_1] + e[j_2] + \ldots + e[j_k]$ называется вектором последовательности σ. Элемент с номером i вектора $f(\sigma)$ определяет число запусков перехода t_i в последовательности σ.

Рассмотрим, как решаются задачи сохранения и достижимости с помощью матричных уравнений.

Задача сохранения. Для того, чтобы показать сохранение, необходимо найти ненулевой вектор взвешивания, для которого взвешенная сумма по всем достижимым маркировкам постоянна. Обозначим искомый вектор-столбец n-размерности через w .

По свойству сохранения необходимо, чтобы $\mu^0 * w = \mu' * w$, где μ' - произвольная маркировка, достижимая из μ^0. Так как μ' достижима из μ^0, то существует последовательность запусков переходов σ, переводящая сеть из μ^0 в μ'. Поэтому $\mu' = \delta(\mu, \sigma) = \mu + f(\sigma)*D$. Следовательно, $\mu' * w = \mu^0 * w = (\mu + f(\sigma)*D)*w = \mu*w + + f(\sigma)*D*w$, откуда $f(\sigma)*D*w = 0$.

Так как это должно быть верно для всех $f(\sigma)$, то $D*w = 0$. Таким образом, сеть Петри является сохраняющей тогда и только тогда, когда существует такой положительный вектор w, что $D*w = 0$. Другими словами, для анализа этого свойства необходимо решить уравнение $D*w = 0$. Если существует положительное решение, то сеть является сохраняющей, а найденный вектор-решение - вектором взвешивания w.

Задача достижимости. Пусть μ' достижима из μ^0, тогда существует последовательность запусков переходов σ, которая приводит из μ^0 к μ'. Это означает, что вектор $f(\sigma)$ является неотрицательным целым решением матричного уравнения вида $\mu' = \mu^0 + f(\sigma)*D$. Таким образом, если это уравнение имеет решение в неотрицательных целых, то μ' достижима из μ^0, в противном случае μ' недостижима из μ^0. На рис. 2.7 представлен пример сети Петри.

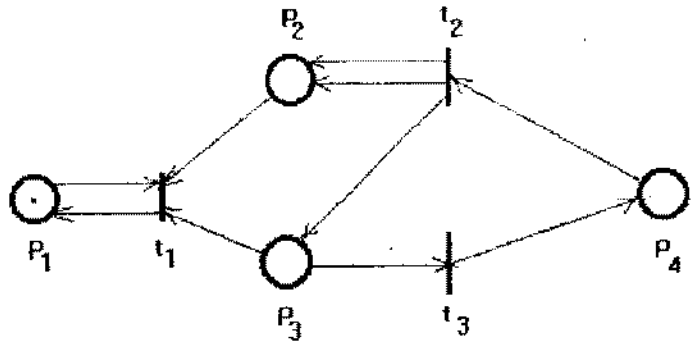


Рис. 2.7

Для этой сети матрицы F,B и D имеют вид

$$F = \begin{vmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{vmatrix}, \rightarrow \quad B = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}, \rightarrow D = \begin{vmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{vmatrix}.$$

В начальной маркировке $\mu^0 = (1,0,1,0)$ переход t_3 разрешен и приводит к маркировке μ' :

$$\mu' = (1,0,1,0) + (0,0,1) * \begin{vmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{vmatrix} = (1,0,1,0) + (0,0,-1,1) =$$

$$= (1,0,0,1).$$

Последовательность запусков $\sigma = t_3\ t_2\ t_3\ t_2\ t_1$ представляется вектором запусков $f(\sigma) = (1, 2, 2)$, который приводит к маркировке μ' вида

$$\mu' = (1,0,1,0) + (1,2,2) * \begin{vmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{vmatrix} = (1,0,1,0) + (0,3,-1,0) =$$

$$= (1,3,0,0).$$

Для определения того, является ли маркировка $\mu' = (1,8,0,1)$ достижимой из $\mu^0 = (1,0,1,0)$, получаем уравнение

$$(1,8,0,1) = (1,0,1,0) + X * \begin{vmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{vmatrix},$$

$$(0,8,-1,1) = X * \begin{vmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{vmatrix},$$

которое имеет решение $X = (0,4,5)$. Это соответствует последовательности $\sigma = t_3\ t_2\ t_3\ t_2\ t_3\ t_2\ t_3\ t_2\ t_3$.

Матричный подход к анализу сетей Петри обладает следующими недостатками.

1. Матрица D не полностью отражает структуру сети Петри. Так, переходы, имеющие как входы, так и выходы из одной позиции (петли), представляются элементами матриц F и B, а в матрице D взаимно уничтожаются (в примере рис.2.7 переход t_1 и позиция p_1).

2. Отсутствует информация о порядке запусков переходов в векторе $f(\sigma)$.

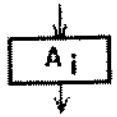
3. Решение уравнения $\mu' = \mu + X * D$ является необходимым условием для достижимости, но недостаточным.

2.2. Методика моделирования программных средств на базе сетей Петри

В качестве формального описания алгоритмов будем рассматривать класс параллельных операторных схем алгоритмов (ПОСА), включающий параллельные граф-схемы алгоритмов (ПГСА), параллельные логические схемы алгоритмов (ПЛСА) и параллельные матричные схемы алгоритмов (ПМСА). Очевидно, что излагаемый подход может быть легко интерпретирован применительно к конкретному языку программирования.

На содержательном уровне ПГСА представляет собой ориентированный граф, типы вершин которого представлены в табл. 2.1.

Таблица 2.1

Графическое изображение вершины	Буквенное обозначение вершины	Операция
	A	Выполнение действий под данными
	A_0	Начало алгоритма
	A_k	Конец алгоритма
	P	Проверка условия
	W	Соединение взаимоисключающих ветвей
	R	Переход к параллельному выполнению ветвей алгоритма
	L	Объединение параллельных ветвей алгоритма

На рис. 2.8 приведен пример ПГСА. ПГСА является корректной, если определяемый ею алгоритм удовлетворяет следующим условиям:

а) детерминирован (при одних и тех же исходных данных всегда получается одинаковый результат);

б) конечен (продолжительность работы алгоритма при применении его к любым исходным данным составляет конечное число шагов-тактов).

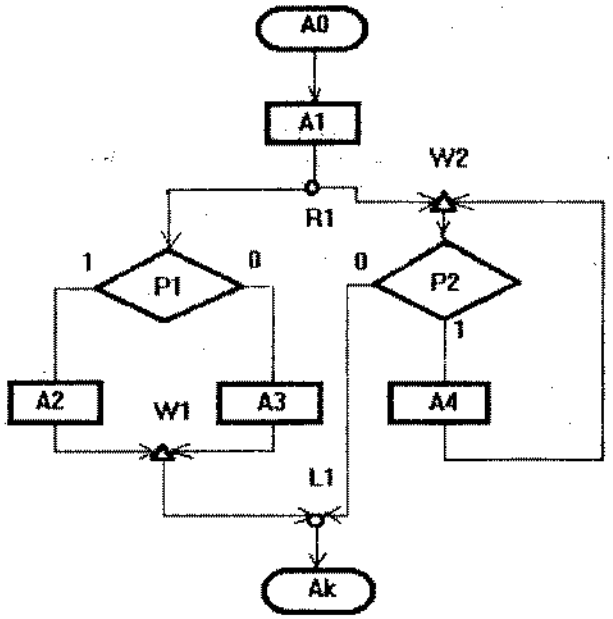


Рис. 2.8

К нарушению корректности ПГСА приводят следующие конфигурации:

а) дедлок (рис. 2.9,а);

б) неоднозначность (рис.2. 9,б);

в) зависание (рис. 2.9,в).

Кроме того, возможны ситуации ресурсных конфликтов, когда между двумя подалгоритмами ПГСА не существует причинно-следственной связи, но недопустимо, чтобы они выполнялись с пересечением во времени. Такие ситуации устраняются установлением фиксированного приоритета между подалгоритмами.

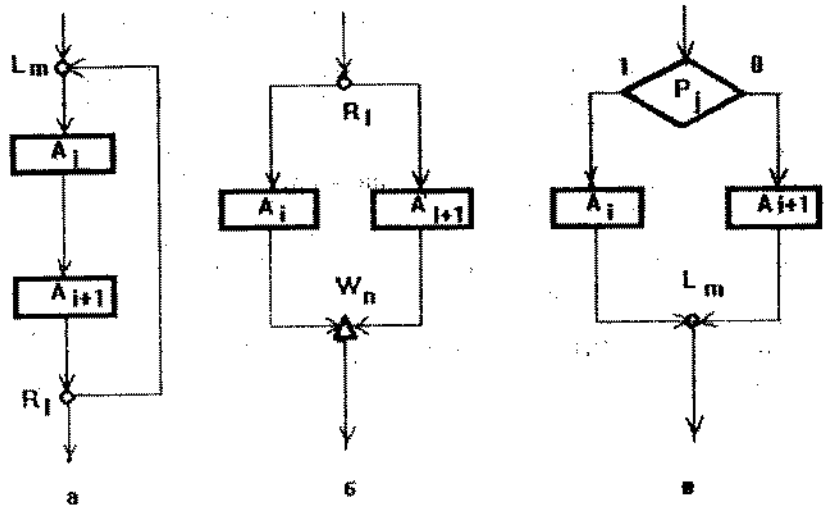


Рис. 2.9

Методика моделирования программного обеспечения в базисе ПОСА состоит в следующем.

1. Построение для ПОСА соответствующей сети Петри.
2. Проверка работоспособности сети Петри.
3. Определение и анализ свойств сети Петри.
4. Определение временных характеристик сети Петри.

Покажем на примере ПГСА построение соответствующей ей сети Петри. Интерпретируем элементы аппарата ПГСА следующим образом.

1. Множеству действий ПГСА соответствует множество переходов T. Срабатывание перехода соответствует выполнению действия.

2. Множеству условий, определяющих порядок действий в ПГСА, соответствует множество позиций P.

3. Условия, разрешающие выполнение действий, соответствуют условиям срабатывания переходов.

4. Наличие исходных данных для ПГСА соответствует начальному маркированию μ^0, а завершение работы ПГСА - конечному маркированию μ^*.

5. Каждой последовательности действий ПГСА соответствует последовательность срабатываний.

Сформулируем правила построения сети Петри по заданной ПГСА.

1. Каждая вершина ПГСА заменяется соответствующим фрагментом сети Петри (ФСП).

2. Фрагменты между собой соединяются так, чтобы сохранялись отношения следования и предшествования действий и условий, предписанных ПГСА.

3. Если две вершины ПГСА смежны, то выходная позиция одного ФСП отождествляется с входной позицией другого ФСП.

ФСП для вершин ПГСА показаны в табл. 2.2.

Сеть Петри для ПГСА рис. 2.8 приведена на рис. 2.10.

В табл. 2.2 для операторной вершины A_i ПГСА срабатывание перехода t_i в ФСП интерпретируется как "начало" выполнения оператора A_i, а срабатывание перехода t_i - как "конец" выполнения оператора A_i. В ФСП для условной вершины P_j на срабатывание переходов t^1_j и t^0_j накладываются дополнительные условия: для

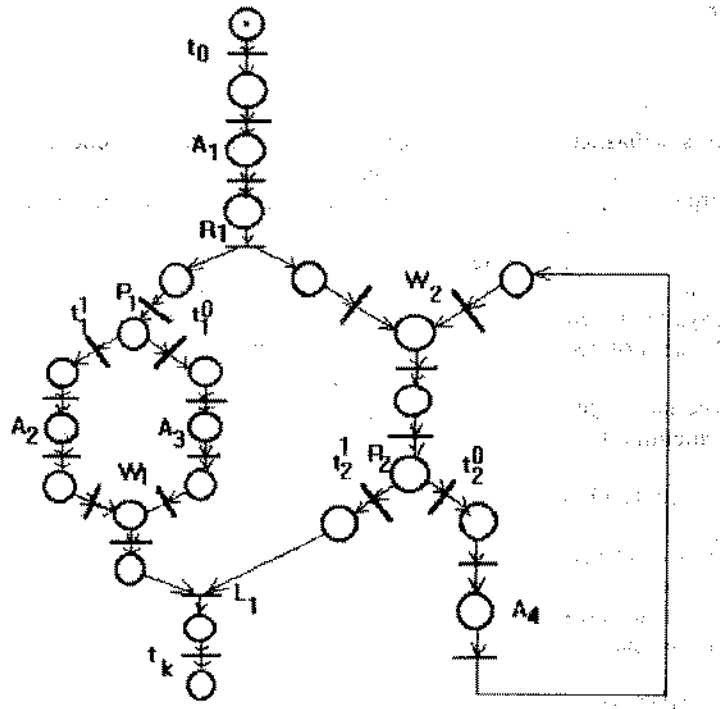


Рис. 2.10

Таблица 2.2

Вершины ПГСА	Фрагменты сети Петри
A0	t_0
Ak	t_k
Ai	t_i A_i t^1_i
1 Pj 0	t^1_j P_j t^0_j
Wn	Wn
R1	R1
Lm	Lm

перехода t^0_j - истинность предиката $Pr(P_j = true)$, для перехода t^1_j - истинность предиката $Pr(P_j = false)$. Такие сети Петри относятся к классу расширенных сетей и называются предикатными.

Проверка правильности работоспособности алгоритма по сети Петри состоит в анализе достижимых маркировок из начальной μ^0. В частности, может быть проверена достижимость конечной маркировки μ^k из μ^0.

Между свойствами сети Петри и условиями корректности ПОСА существует соответствие, доказанное в виде теоремы [3].

Теорема. *ПОСА корректна тогда и только тогда, когда соответствующая ей сеть Петри жива и безопасна.*

Таким образом, анализ критических свойств сети (живость, безопасность и конфликтность) дает основание судить о свойствах моделируемого алгоритма.

В [4] разработаны алгоритмы преобразования сетей Петри с целью устранения их критических свойств, там же предлагаются алгоритмы оптимизации сетей с целью их минимизации.

Вводя в рассмотрение временные характеристики переходов Δt_i, отражающие длительность выполнения операций, получают минимальное, максимальное и среднее времена выполнения алгоритма.

Таким образом, моделирование алгоритмов (программ) с помощью сетей Петри позволяет получить качественные (работоспособность, корректность) и количественные (временные) оценки алгоритмов и программ, как последовательных, так и параллельных.

Рассмотрим еще два примера использования сетей Петри для моделирования: моделирование вычислительных процессов и автоматов.

Вычислительная система обрабатывает задания, поступающие с устройства ввода, и выводит результаты на устройство вывода. Когда процессор свободен, и в устройстве ввода есть задание, процессор начинает обработку задания. Когда задание выполнено, оно посылается на устройства вывода. Процессор либо продолжает обрабатывать другое задание, если оно есть, либо ждет прихода нового задания. Сеть Петри, моделирующая работу такой вычислительной системы, представлена на рис. 2.11.

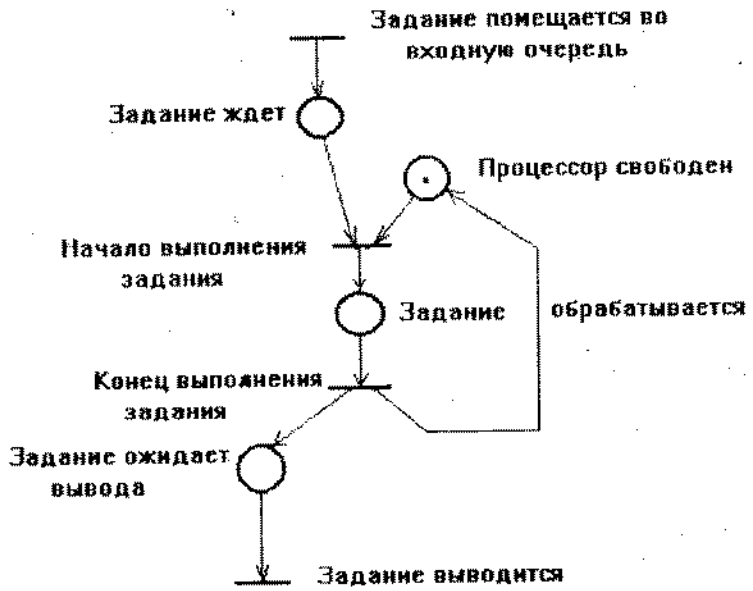


Рис. 2.11

Автомат подсчитывает количество единиц (четное или нечетное) в двоичном числе. Граф автомата Мили показан на рис. 2.12.

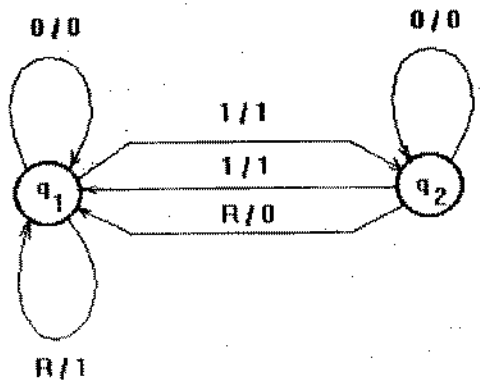


Рис. 2.12

Работа автомата начинается в состоянии q_1. Выход копирует вход до тех пор, пока входным символом не окажется сигнал сброса R. Выходом для R будет 0 в случае нечетного числа единиц и 1 - в случае четного. Сеть Петри строится для автомата следующим образом. Позициям сети ставятся в соответствие входной и выходной символы. В позицию, соответствующую входному символу, помещается фишка, а затем фишка, появившаяся в позиции, соответствующей выходному символу, удаляется оттуда. Общая схема имеет вид, показанный на рис. 2.13.

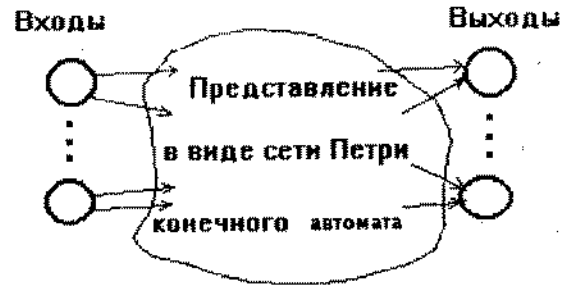


Рис. 2.13

Затем каждое состояние автомата представляется позицией. Текущее состояние отмечается фишкой, все остальные позиции пусты. Для каждой пары позиций (состояние и входной символ) определяется выходной переход, выходным позициям которого соответствуют следующее состояние и выходной символ. Сеть Петри, соответствующая автомату рис. 2.12, изображена на рис. 2.14.

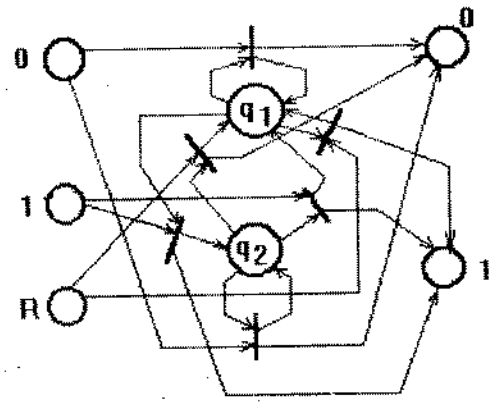


Рис. 2.14

Хотя описание сетью Петри более громоздко, чем с помощью автоматов, у сети есть преимущество при последовательной композиции (рис. 2.15), заключающееся в простом совмещении выходных позиций первой сети с входными позициями второй, и при параллельной композиции (рис. 2.16) - в дублировании фишек в позициях, соответствующих входным символам.

Моделирование и анализ сетей Петри выполняется с помощью аппаратных и программных средств.

В [4] предлагается специализированная аппаратная моделирующая система (рис. 2.17). Система содержит блок памяти (БП), блок сравнения и управления (БСУ), блок моделей вершин (БМВ), блок моделей входных сигналов управления (БМСУ), блок формирования последующей разметки (БФПР), индикаторную панель (ИП).

В БП записываются исходные данные, характеризующие моделируемую сеть:

1) множество всех входных разметочных векторов (входной разметочный вектор для перехода t_j характеризует наличие меток во

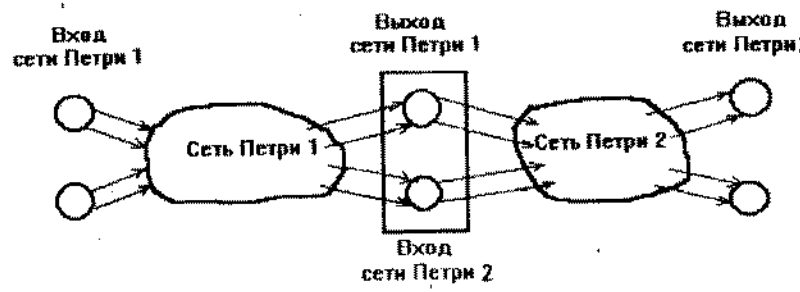


Рис. 2.15

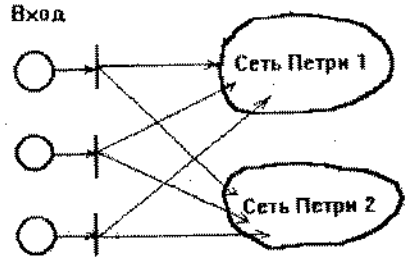


Рис. 2.16

входных позициях t_j, т.е. t_j является разрешенным и может сработать);

2) множество выходных разметочных векторов (выходной разметочный вектор для перехода t_j характеризует наличие меток в выходных позициях t_j после его срабатывания);

3) времена срабатывания переходов, моделирующие выполнение операций;

4) вектор начального маркирования μ^0.

Устройство предназначено для моделирования управляющих сетей [4], являющихся расширением классических сетей Петри. В управляющих сетях предусмотрена связь отрабатываемого параллельного алгоритма (программы) с объектами управления или другими алгоритмами (программами). Для этого опрашиваются входные сигналы управления, описываемые вектором X, и выдаются выходные сигналы (вектор Y).

В БСУ устройства осуществляется опрос выполнения условий возбуждения переходов в соответствии с множеством входных разметочных векторов. Для возбуждения переходов в БСУ дополнительно проверяется условие активизации подвектора входных

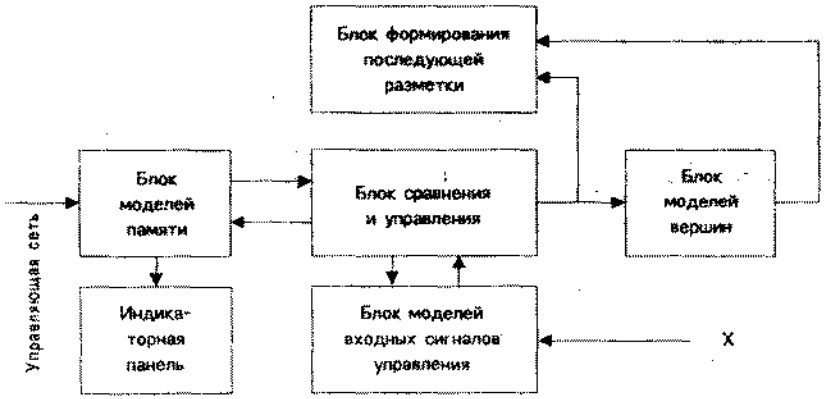


Рис. 2.17

сигналов, поступающее из БМСУ. Вектор входных сигналов управления X формируется по закону случайных последовательностей чисел и обеспечивает проверку моделируемого алгоритма в различных режимах эксплуатации. Для этого в БП вводятся интервалы изменения соответствующих времен срабатываний переходов Δt_j. БМВ предназначен для моделирования времен срабатывания Δt_j, соответствующих возбужденным и активизированным переходам t_j. БФПР предназначен для формирования последующей разметки, ИП служит для отображения вектора текущей разметки.

Устройство моделирования работает следующим образом. В БСУ определяются разрешенные переходы и формируется суммарный входной разметочный вектор, который в БФТР вычитается из вектора начальной разметки и записывается вместо него. Эта операция отражается на ИП, в которой вершине p_i соответствует индикатор. Если в вершине p_i есть метка, индикатор зажигается, нет - находится в потушенном состоянии. Отметим, что управляющие сети относятся к классу безопасных сетей. Одновременно с выполнением операции вычитания в БМВ запускаются соответствующие j-е модели вершин t_j. После моделирования операции a_j через время Δt_j в БМВ формируется сигнал окончания срабатывания перехода t_j, который разрешает выполнение операции сложения в БФПР выходного разметочного вектора с вектором текущей разметки и записи результата вместо него.

В [4] разрабатываются устройства для моделирования различных классов расширенных сетей Петри (о расширенных классах см. п. 2.3), предлагаются устройства анализа работоспособности и критических свойств. Там же отмечаются особенности и достоинства таких специализированных устройств:

1) одновременность опроса всех входных разметочных векторов на выполнения условия разрешенности переходов;

2) высокая надежность работы при значительно меньшем времени моделирования по отношению к интерпретаторам сетей Петри;

3) полная адекватность моделирования параллельных процессов и реального их протекания;

4) возможность проверки в реальном масштабе времени совместной работы программной и аппаратной частей мультипроцессорных систем управления.

Специализированные устройства моделирования сетей Петри могут применяться при проектировании и отладке конкретных специализированных мультипроцессорных систем управления. Для этого выбирается наиболее подходящее устройство моделирования, на нем реализуется параллельный алгоритм управления, который проверяется на работоспособность и некритичность. Затем удаляются все лишние связи, характерные для универсального моделирующего устройства, в результате получается готовая мультипроцессорная система управления.

Программные средства моделирования (интерпретаторы) и анализа сетей Петри просты и требуют для своей реализации 1-3 чел./мес. Входные данные программных систем задаются в матричном, теоретико-множественном видах или с помощью специализированных языковых средств (например, в [4] предлагается язык параллельных алгоритмов управляющих сетей ЯПАУС).

2.3. Методика моделирования аппаратных средств на базе сетей Петри

Для моделирования аппаратных средств используются расширенные сети Петри - аппаратные сети Петри. Рассмотрим отличительные особенности этого класса сетей.

Все переходы множества T разделяются на два класса: простые t_j и составные t^S_j. Простому переходу t_j соответствует элементарное действие a_j в моделируемом устройстве, а составному переходу t^S_j - совокупность элементарных действий $a_j = (a^1_j, a^2_j, ..., a^e_j)$. Каждый переход обладает временной характеристикой Δt_j (Δt^S_j), отражающей в единицах модельного времени длительность выполнения действия a_j (a^S_j). Величина Δt_j (Δt^S_j) может быть константой или задаваться по случайному закону распределения (нормальному, равномерному и др.). Каждому переходу поставлен также в соответствие элемент входного алфавита сигналов управления X^j. Условие $X^j = 1$ является достаточным для запуска перехода t_j (t_j). В аппаратные сети Петри введен механизм приоритетов переходов,

позволяющий определить очередность срабатывания переходов в случае возникновения конфликтных ситуаций, каждому переходу присваивается приоритет PR_j, $PR_j = 0$ отражает наиболее низкий уровень приоритетов. Уровень приоритета может изменяться в процессе моделирования.

Множество позиций P разделяется на подмножество простых позиций p_i и позиций аккумулирующего типа p^a_i. Позиции p_i изображаются двойными окружностями и отличаются от обычных правилами управления поступлением, учетом и удалением меток, определяемыми в зависимости от типа направленных к месту дуг. Учет меток в позициях p_i организуется следующими способами:

- хранение общего числа поступивших на текущий момент меток без учета их последовательности и характеристик;
- хранение в очереди (занесение и удаление меток может быть организовано по правилам FIFO или LIFO), в этом случае позиция характеризуется максимальной емкостью очереди $NQ_i = \{1, 2, ..., \infty\}$ и текущим числом элементов очереди (рис. 2.18 ,а);
- хранение в составе адресуемого списка, при этом поступившая в позицию p_i метка располагается по одному из адресов, которые определяются в зависимости от характеристики метки, номера поступившей метки и т.д. Место, содержащее адресуемый список, характеризуется максимально возможным адресом AS_i в списке (AS_i>=2) (рис. 2.18, б).

Позициям множества P может быть поставлено в соответствие время фиксации меток Δp_i. В классических сетях при срабатывании перехода t_j определенное число меток удаляется из его входных позиций. Если введено p_i, то из позиции p_i при запуске t_j метка не удаляется, а считается зафиксированной и, если в течение Δp_i будет попытка удаления фиксированной метки, устанавливается признак некорректных действий.

Множество дуг аппаратных сетей Петри классифицируется по следующим признакам:

- принципу определения веса дуги (количество передаваемых по дуге меток) - дуги с постоянным и переменным весами (w(k)=const, w(k)=var);
- влиянию на правила управления разметкой сети;
- типу условий срабатывания перехода.

Некоторые дуги могут относиться к запрещающим (ингибиторным) (рис. 2.19), для запуска перехода с ингибиторными дугами необходимо отсутствие меток в ингибиторных позициях.

По влиянию на управление разметкой дуги разделяются на простые, разрешающие и добавляющие. Разрешающие дуги позволяют удалять метки из позиций аккумулирующего типа, а добавляющие -

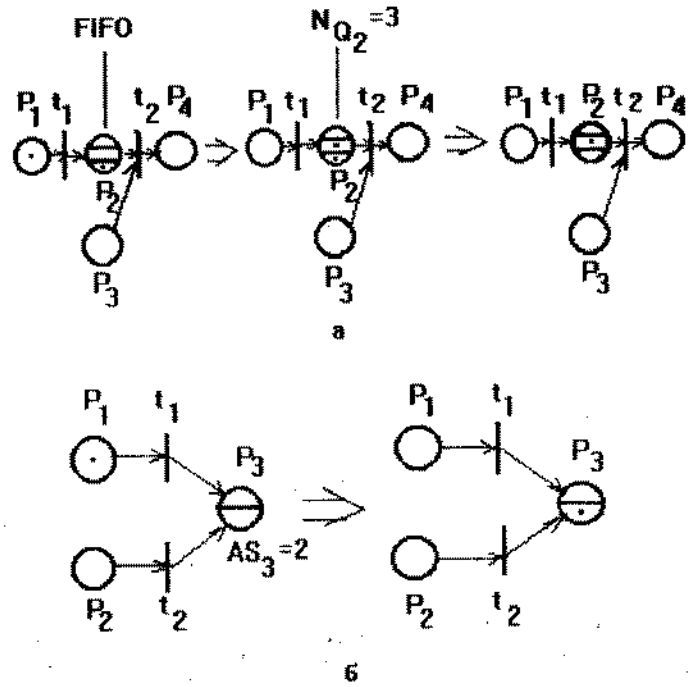


Рис. 2.18

заносить метки в такие места. Разрешающие и добавляющие дуги отмечаются пунктирной линией вдоль дуги.

По типу условий срабатывания перехода дуги разделяются на достаточные для срабатывания переходов и необходимые для правильного срабатывания переходов в смысле логики функционирования моделируемого устройства. Обычно дуги достаточных условий направлены от позиций, отображающих некоторые управляющие переменные, а дуги необходимых условий - от мест аккумулирующего типа, отражающих информационные переменные, наборы данных и т.п.

В аппаратных сетях метки (фишки) могут быть традиционного типа, а также раскрашенные в определенные числовые значения, например, для обозначения уровней сигналов (0 - сигнал низкого уровня, 1 - сигнал высокого уровня, 3 - переходной процесс).

Рассмотрим задачу оценки быстродействия вычислительных устройств, для которых время выполнения ими или их элементами предписанных функций является критерием успешной работы. Обыч-

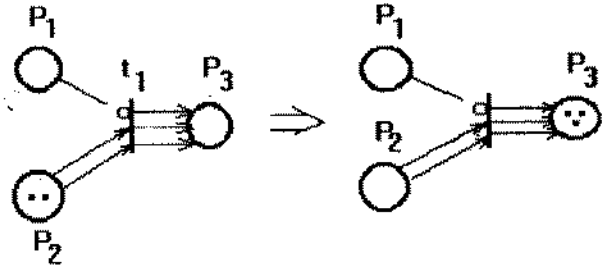


Рис. 2.19

но временные характеристики таких устройств определяются длительностью микрокомандного цикла управления. На рис. 2.20. показана сетевая модель цикла, разделенного на n микроциклов. Длительность микрокомандного цикла равна $\sum_{j=1}^{n} \Delta t_j$.

Управляющий цикл определяет распределение меток в множестве позиций P_{RMC}, имитирующих разряды регистра микрокоманды. Разряды имитируются при помощи аккумулирующих позиций p_i, причем каждому разряду соответствует одно место. Применение мест аккумулирующего типа вызвано возможностью использования одного и того же разряда микрокоманды для управления несколькими процессами. Такая конструкция управляющего цикла обеспечивает независимость модели блока управляющих сигналов от действий, выполняемых в любой части модели исполняющей части устройства. Каждому переходу присваивается время срабатывания.

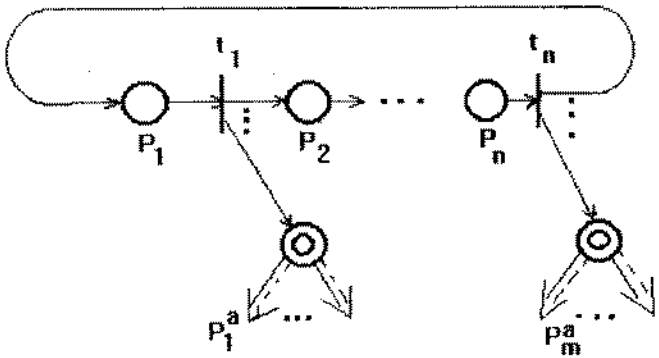


Рис. 2.20

В [4] определяются два временных параметра: максимальная частота срабатывания перехода t_j ($МЧС_{tj}$) и минимальный интервал между последовательным занесением меток в позицию p_i ($МИМ_{pi}$). $МЧС_{tj}$ - количество срабатываний перехода t_j за время между повторением разметки управляющего цикла. $МИМ_{pi}$ - наименьший возможный интервал между занесением двух последовательных меток в p_i.

Для того, чтобы позиция была безопасной, необходимо выполнение соотношения

$$МИМ_{pi} = D / МЧС_{tj},$$

где $D = \sum_{j=1}^{n} \Delta t_j$ - длительность выполнения микроцикла, t_j - входной переход позиции p_i.

Рассмотрим применение аппаратных сетей ПЕтри для моделирования цифрового процессора сигналов (ЦПС) [4].

Структура ЦПС представлена на рис. 2.21. В состав ЦПС входят два процессорных элемента (ПЭ1 и ПЭ2), блок усреднителя (БУС) и блок микропрограммного управления (БМУ). Данные для обработки поступают с выхода аналого-цифрового преобразователя (АЦП).

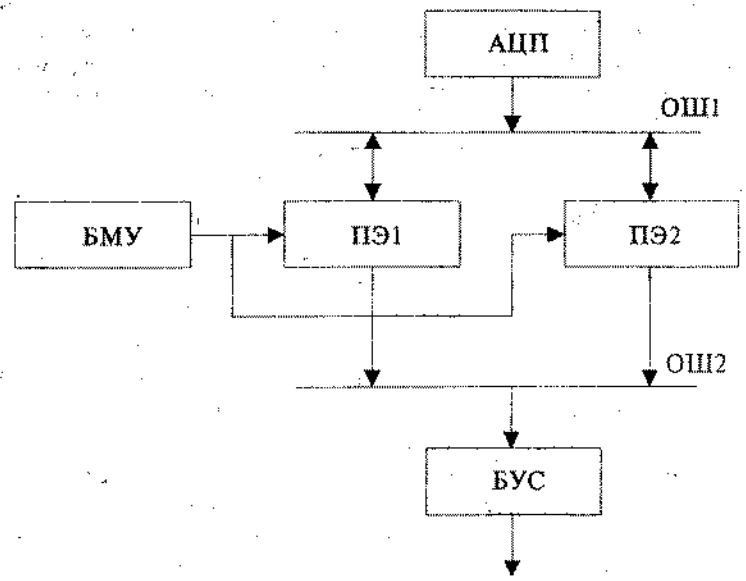


Рис. 2.21

В ПЭ1 и ПЭ2 могут выполняться два вида операций (ОП1 и ОП2). Каждая микрокоманда БМУ содержит восемь разрядов: П1 и П2 - разрешение функционирования ПЭ1 и ПЭ2 соответственно; ЧтОШ1 - занесение информации с общей шины (ОШ1) в регистры $RG_{вх1}$ и $RG_{вх2}$; РОП1 и РОП2 - разрешение выполнения соответствующих операций в ПЭ1 и ПЭ2; ЗпОШ21 и ЗпОШ22 - занесение данных с ОШ2 в БУС. Длительность микрокомандного цикла составляет 100 нс.

Модель ЦПС представлена на рис. 2.22. Переход t_1 моделирует длительность микрокомандного цикла и позволяет изменять разметку позиций p^a_1 - p^a_8, соответствующих разрядам микрокоманды, через определенные промежутки в установленные моменты модельного времени. Позиция p_9 моделирует последовательность микрокоманд, начальное число меток в ней соответствует длине микропрограммы. Хранение меток организовано в рамках очереди, метки окрашены в строки бит, равные значениям микрокоманд микропрограммы. Переход t_2 очищает позиции p^a_1 - p^a_8 в начале каждого микропрограммного цикла и обеспечивает занесение в них меток в соответствии со значениями разрядов строк бит и соответствующей раскраски поступающих из p_9 меток. Управление весами выходных дуг перехода t_2 производится с помощью специальной процедуры. Веса выходных из перехода t_2 дуг имеют значения 1 или 0, это позволяет изменять код микрокоманды при отработке новых микропрограммных циклов. От позиций p_i ($1 \le i \le 8$) к переходу t_2 направлены разрушающие дуги, которые убирают метки из аккумулирующих позиций p^a_i.

Позиция p_{12} моделирует возможные значения сигналов, поступающих с выхода АЦП. Переход t_3 моделирует передачу цифровых значений сигналов на ОШ1 в каждом микропрограммном цикле. Для включения ПЭ1 ($\mu(p^a_1) = 1$) или ПЭ2 ($\mu(p^a_2) = 1$) метка из p_{14} по сигналу считывания $\mu(p^a_1) = 1$ переходит в позицию p_{15}. Переходы t_4 и t_5 моделируют операции поступления значения сигнала на $RG_{вх1}$ и $RG_{вх2}$.

Для выполнения одной из двух операций (ОП1 или ОП2) включается переход t_6 (t_8) или t_7 (t_9). Поступление меток в позиции p_{17} или p_{18} отражает окончание выполнения ОП1 или ОП2 и поступление цифрового сигнала на $RG_{вых1}$ или $RG_{вых2}$. Переходы t10 и t11 моделируют выдачу значения сигнала на ОШ2, после чего по сигналу ЗпБУ ($\mu(p^a_8) = 1$) срабатывает переход t_{12} и данные поступают на $RG_{уср}$ в БУС.

Для получения численного результата прохождения сигнала от АЦП до БУС переходам t_4 - t_{12} устанавливаются временные за-

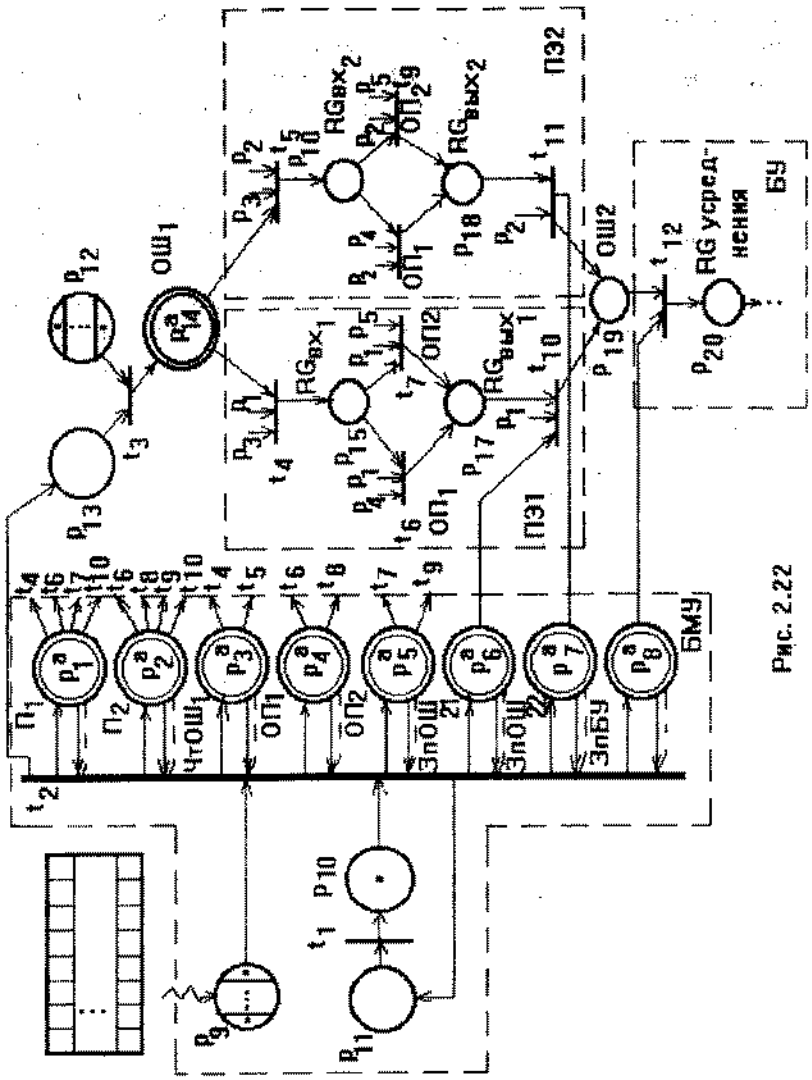


Рис. 2.22

держки t_4 - t_{12}, имитирующие соответствующие операции, а при позициях p_{14} - p_{19} организованы очереди.

В ходе моделирования кроме временных диаграмм и имитации численного результата может быть выявлено неправильное сочетание значений разрядов обрабатываемых команд, например, одновременное выполнение ОП1 и ОП2 в одном ПЭ (конфликтное свойство для t_6 и t_7 или для t_8 и t_9), попытка одновременного занесения данных на ОШ2 из ПЭ1 и ПЭ2 (небезопасное свойство, (μ $(p^a_{19}) > 1$).

Применение аппаратных сетей Петри позволяет проводить моделирование логики выполнения различных микропрограмм с имитацией совместной работы отдельных функциональных блоков. Использование таких сетей дает возможность моделировать параллельные процессы в вычислительных устройствах, исследовать их взаимодействие, оптимизировать структуру устройств. Моделирование может осуществляться с различным уровнем детализации: от уровня законченных функциональных модулей до уровня логических элементов.

3. ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

3.1. Модели массового обслуживания

Модели массового обслуживания нашли широкое применение при анализе вычислительных систем и оценке их эффективности [5-8]. Они применяются обычно для анализа характеристик производительности, но могут быть использованы и для оценки характеристик надежности. В моделях массового обслуживания рассматриваются системы, на вход которых в произвольные моменты времени поступают заявки, требующие обслуживания путем предоставления определенных ресурсов на некоторое время и покидающие систему после обслуживания.

Системы массового обслуживания (СМО) делятся на одноканальные - с одним обслуживающим прибором (ресурсом) и многоканальные, содержащие несколько обслуживающих приборов (рис 3.1). Для создания СМО необходимо задать:

- распределение длительности интервалов времени между заявками входного потока;
- число обслуживающих приборов;
- распределение длительности обслуживания заявок обслуживающими приборами;
- стратегию выбора заявок из очереди.

Для заданного множества параметров СМО можно получить такие характеристики системы, как:

- время обслуживания заявки в системе;
- время ожидания заявкой обслуживания (время нахождения в очереди);
- длина очереди;
- загрузка обслуживающих приборов;
- длительность простоя обслуживающих приборов и т.д.

Поскольку как минимум один из параметров модели имеет вероятностный характер, то и характеристики СМО также будут