ISSN 0136-3549
0134-3823

TALLINNA
POLÜTEHNILISE INSTITUUDI
TOIMETISED

530

ТРУДЫ ТАЛЛИНСКОГО
ПОЛИТЕХНИЧЕСКОГО
ИНСТИТУТА

TPI '82

СИНТЕЗ И ДИАГНОСТИКА ЦИФРОВЫХ УСТРОЙСТВ И СИСТЕМ

530

TPI '82

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.32

СИНТЕЗ И ДИАГНОСТИКА ЦИФРОВЫХ УСТРОЙСТВ И СИСТЕМ

Электротехника и автоматика XIX

Таллин 1982

Содержание

ТАЛЛИНСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
Труды ТПИ № 530
СИНТЕЗ И ДИАГНОСТИКА ЦИФРОВЫХ УСТРОЙСТВ И СИСТЕМ
Электротехника и автоматика XIX
Редактор В. Кукк. Техн. редактор Е. Зорина

Сборник утвержден коллегией Трудов ТПИ 10 марта 1982 года
Подписано к печати 29 октября 1982 года. Формат 60X90/16.
Печ. л. 7,0. + 0,25 приложение. Уч.-изд. л. 6,0. Тираж 300. МВ-03785
Ротапринт ТПИ, Таллин, ул. Коскла, 2/9. Зак. № 502 . Цена 90 коп.

№ 530

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.32

А.А. Воолайне, А.Р. Йыги,

М.А. Пальль

ПРОЕКТИРОВАНИЕ КОНТРОЛЬНЫХ ЭКСПЕРИМЕНТОВ В АВТОМАТИЗИРОВАННОЙ СИСТЕМЕ КОНТРОЛЯ ЦИФРОВЫХ АВТОМАТОВ "НАКС"

I. Введение

Основными компонентами автоматизированной системы контроля цифровых автоматов "НАКС" являются программно-управляемый тестер, программное обеспечение тестера и программное обеспечение проектирования контрольного эксперимента. Тестер работает под управлением малой ЭВМ (СМ-3), где реализовано программное обеспечение тестера. Подсистема проектирования (ПП) системы "НАКС" работает на ЕС ЭВМ с ОС.

Данная статья посвящена вопросам проектирования контрольного эксперимента. ПП системы "НАКС" предназначена для выполнения следующих работ:

- проверка описания автомата;
- составление модели автомата;
- составление тест-процедуры автомата;
- учет полноты тест-процедуры автомата.

Модель и тест-процедура используются для управления тестером при проведении контрольного эксперимента с автоматом. Они являются основными выходными данными ПП.

2. Рассматриваемые объекты и неисправности

Данная система предназначена для генерации тестов для цифровых автоматов, реализованных на цифровых интегральных ТТЛ-микросхемах малой и средней степени интеграции. Алгоритмы программ ориентированы в основном на синхронные схемы, но позволяют обработать также асинхронные схемы, хотя с меньшей эффективностью. Допускаются контуры в схеме. Количество микросхем в одном объекте - до 150. Не могут обрабатываться автоматы, принцип работы которых зависит от конкретных величин задержек элементов. Исключением являются случаи, когда соответствующие подсхемы полностью содержатся в одном элементе (микросхеме), например, триггера в интегральном исполнении. В последнем случае используются готовые функциональные модели. Генерирующие подсхемы не допускаются. Допускаются расширители из указанных типов микросхем, а также элементы с открытым коллектором. В случае соединения выходов таких элементов программы добавляют в модель фиктивные элементы.

Круг рассматриваемых неисправностей в данной системе ограничивается классом логических неисправностей, под которыми понимают дефекты элементов автомата, а также дефекты связей, сводящиеся к изменению логических функций, реализуемых элементами.

Рассматриваемые неисправности делятся на две группы:

- простые (логические константы на межэлементных связях автомата);

- сложные (функциональные неисправности внутри элементов и монтажные неисправности на уровне межэлементных связей типа короткого замыкания, перепутывания проводов и т.д.).

Первая группа неисправностей обрабатывается автоматически. При этом гарантируется учет всех константных неисправностей на входах и выходах микросхем, а также некоторая часть константных неисправностей на сигнальных путях внутри элементов (полнота охватываемых внутриэлементных неисправно-

стей определяется точностью описания структуры элемента в модели этого элемента).

При обработке второй группы неисправностей требуется задание их в виде перечисления дополнительных условий [I]. Так, например, роль дополнительных условий для функциональных неисправностей элементов играют их стандартные тесты.

3. Модель объекта контроля

Данная система базируется на альтернативной граф-модели автомата [2]. Модель автомата представляет собой систему графов, компонентами которых являются модели физических компонентов схемы. В некоторых случаях целесообразно сжатие системы графов [4].

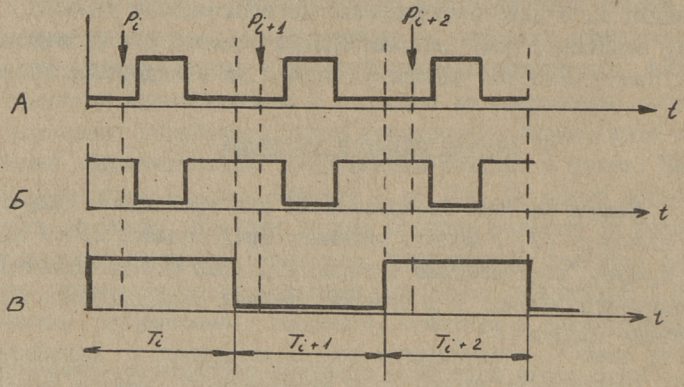
Рассмотрим подробнее проблему составления граф-моделей для синхронизируемых элементов-триггеров, счетчиков и др. Модель таких элементов будет зависеть от типа сигнала на входе элемента (потенциальный или импульсный), а также от временной диаграммы синхросигнала. Можно разделить модели элементов на две группы:

- модели, ориентированные на использование импульсных и потенциальных сигналов;

- модели, ориентированные на использование только потенциальных сигналов.

Модели первой группы условно будем называть импульсными, а модели второй группы - потенциальными моделями. Для определенности будем ниже считать, что временная диаграмма входных сигналов автомата имеет вид, показанный на фиг.I.

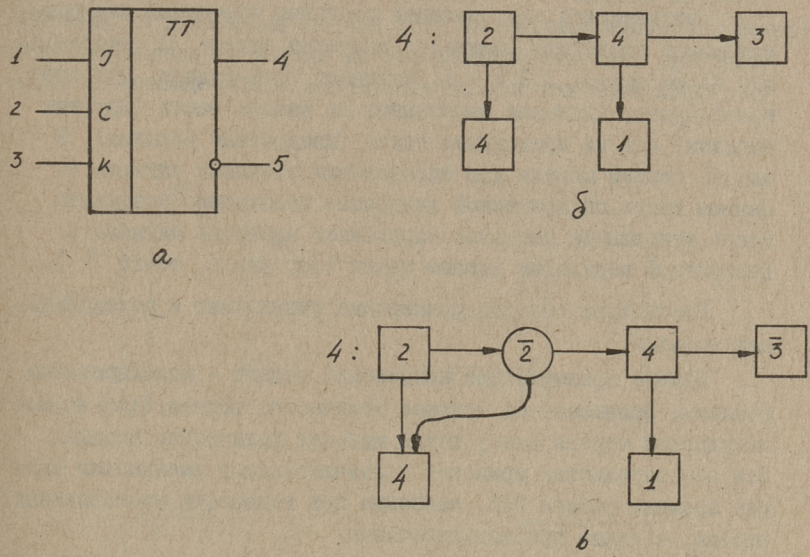
Диаграммы А и Б представляют импульсные сигналы положительной и отрицательной полярности, а диаграмма В - потенциальный сигнал. T_i, T_{i+1} и т.д. означают номера тактов. Стрелками P_i, P_{i+1} и т.д. на диаграмме показаны моменты, когда на выходах автомата определяются значения сигналов для соответствующего такта. Эта диаграмма определяет соответствие реального процесса и дискретной модели. В дискретной модели для положительных импульсных сигналов "I" означает наличие импульса на данном такте, а "0" - его отсутствие.



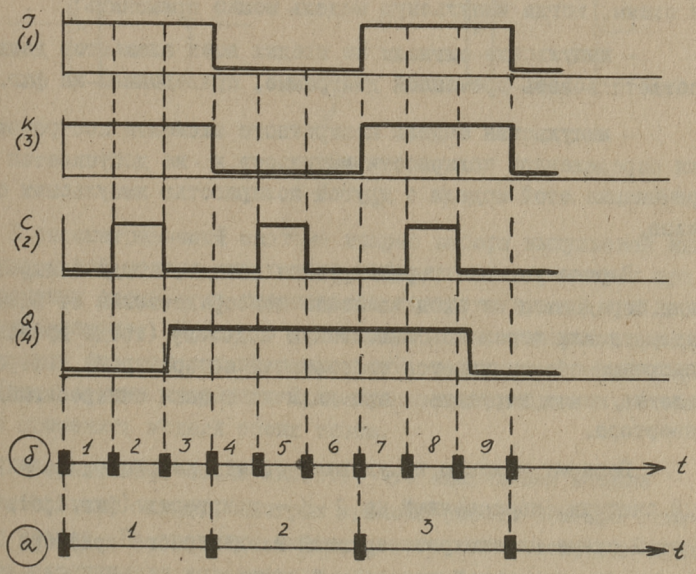
Фиг. 1.

Для отрицательного импульса "0" означает его наличие, а "I"- отсутствие.

Работу любого конкретного синхронизируемого элемента из рассматриваемого класса (цифровые ТТЛ-микросхемы) можно описывать как импульсной, так и потенциальной моделью. Покажем это на примере J-K - триггера (фиг.2,а). На фиг.2,б показана импульсная граф-модель, а на фиг.2,в - потенциальная граф-модель для выхода 4 этого триггера. Работу триггера и его двух моделей поясняем на примере конкретной последовательности входных сигналов. Соответствующий процесс представлен на фиг.3, где имеются две временные оси - ось "а" для импульсной модели, ось "б" для потенциальной модели. В двух разных моделях один и тот же физический процесс разбивается на разное количество тактов: в первом случае процесс занимает три такта, а во втором - девять. Это связано с тем, что во второй модели (потенциальной) любое изменение сигналов на входе автомата связано с переходом на следующий такт, а в первой модели последовательность значений 0 - I - 0 на импульсном входе рассматривается как один сигнал.


ТТ
1
2
3
J
C
K
4
5
а
4 :
2
4
3
4
1
б
4 :
2
2̄
4
3̄
4
1
в

Фиг. 2.


J
(1)
K
(3)
C
(2)
Q
(4)
б
1
2
3
4
5
6
7
8
9
t
а
1
2
3
t

Фиг. 3.

Соответствие двух моделей реальным временным диаграммам можно проверить, вычисляя значение выхода Q (переменная 4) на тактах 2,3 и т.д. согласно граф-модели. При этом используются значения переменных на данном такте (круглые вершины) или на предыдущем такте (квадратные вершины). В случае первой модели при определении значения сигнала на данном такте по временной диаграмме необходимо учитывать, какое мгновенное значение определяет значение сигнала в дискретной модели на данном такте (см. фиг.I, точки P_i).

Рассмотрим области применения импульсных и потенциальных моделей.

Важное преимущество импульсной модели – моделирование реальных процессов за меньшее количество тактов. Граф-модель импульсной модели имеет также меньшее количество вершин. Эти обстоятельства приводят к значительному уменьшению затрат времени работы ЭВМ, особенно при генерации многотактных тестов, а также при моделировании.

С другой стороны, импульсные модели элементов можно применять не для любых схем. Имеется два типа ограничений на схемы, когда импульсную модель можно применять:

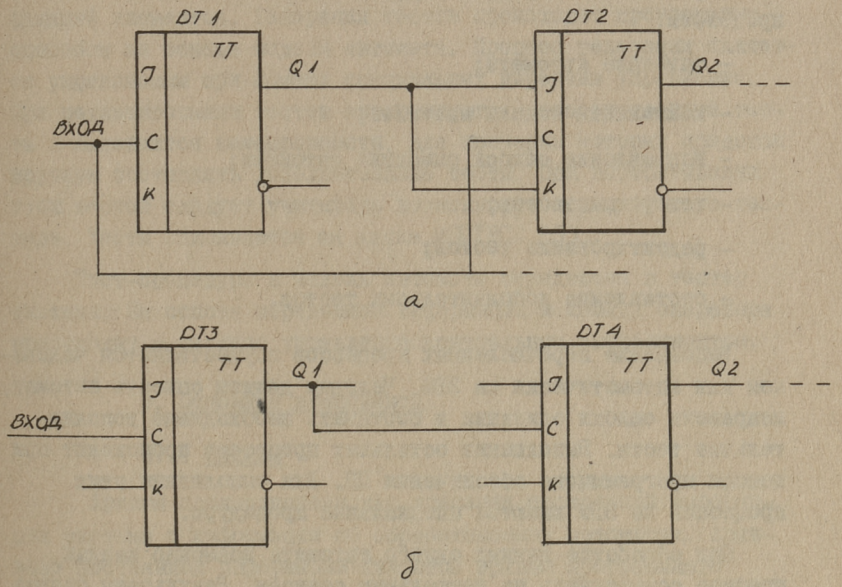
– импульсные сигналы на входах всех элементов должны соответствовать временной диаграмме, приведенной на фиг.I;

– импульсная модель конкретного элемента построена для определенной полярности импульсов, и не допускается применение этой модели с другой полярностью импульсных сигналов.

Первое условие гарантируется всегда, когда синхросигналы передаются на вход элемента непосредственно от входа автомата или через комбинационную подсхему (тогда временная диаграмма обеспечивается тестирующей аппаратурой). Второе условие может нарушаться при наличии в цепи синхросигнала инвертора.

Первому условию не удовлетворяет, например, асинхронный счетчик, построенный на J–K – триггерах (фиг.4,б). В этой схеме на синхронизирующий вход первого триггера DT1 поступает корректный импульсный сигнал, а на вход второго

триггера DT2 – потенциальный сигнал Q_1 с выхода первого триггера. В данном случае для DT2 необходимо применить потенциальную модель. В случае же синхронного счетчика (фиг.4,а) можно применить только импульсные модели для всех каскадов.



Фиг. 4.

В рассматриваемой системе вопрос выбора импульсной или потенциальной модели для какого-либо элемента решается при кодировании схемы, где указывается соответствующий тип для каждого элемента. В архиве для каждого типа синхронизируемой микросхемы имеется как потенциальный, так и импульсный вариант (под разными именами). В пределах модели одного автомата можно применить модели обоих типов.

В заключение можно сказать, что импульсные модели элементов дают большой выигрыш времени работы ЭВМ при генерации тестов. На практике имеется много автоматов, где приведенные ограничения выполняются. Там целесообразно применение

импульсных моделей. Такими автоматами являются чисто синхронные схемы на базе микросхем средней степени интеграции.

4. Основные процессы

Обработка данных в ПП подразделяется на шесть основных процессов:

- описание автомата;
- компиляция модели автомата;
- исправление ошибок описания автомата;
- генерация тестов;
- редактирование тестов;
- составление дополнительных тестов.

Выполнение перечисленных процессов осуществляется человеком или автоматически на ЭВМ. Человек должен описать автомат, исправить ошибки описания и составить необходимые дополнительные тесты. Выполнение остальных процессов происходит при помощи программного обеспечения ПП. Для выполнения этих процессов на ЭВМ имеются специальные процедуры.

При обработке данных одного автомата возможно каждый процесс осуществить по нескольким сеансам. Результаты сеанса можно прибавить к результатам предыдущих сеансов, но новыми результатами можно также заменить старые результаты. Характер сеанса (дополняющий или заменяющий) необходимо указать пользователем системы. Соединение или перезапись данных в архивах системы осуществляется автоматически процедурами ПП.

Автомат описывается на основе принципиальной электрической схемы. Описание составляется в входном языке системы "НАКС". Оно содержит списки элементов и соединений контактов элементов. При помощи специального элемента (разъема) описываются входы и выходы автомата. Для локализации неисправностей можно указать координаты каждого элемента на печатной плате. Синхронизирующие входы автомата необходимо перечислить отдельно. Многолистовые схемы автоматов можно описать по отдельным листам. При описании автомата необходимо назначить автомату символьное имя.

При компиляции модели осуществляются проверка, преобразование и дополнение описания автомата. Если в описании обнаруживаются ошибки, то следует исправить ошибки и повторить компиляцию. Проверенное описание преобразуется из текстового вида в табличный вид и дополняется описаниями логических функций элементов. Генерация тестов происходит программным способом на основе модели автомата. Процесс генерации является управляемым при помощи специальных директив управления. При редактировании тестов анализируются сгенерированные тесты и выясняются неисправности, для проверки которых придется вручную составлять дополнительные тесты. При вырабатывании этих тестов следует учитывать возможности используемого тестера. Тесты описываются на языке МЭМТЕСТ [3] .

Тест-процедура и модель автомата передаются в тестер системы. На основе переданной информации и команд оператора происходит в тестере управление контрольным экспериментом.

5. Основные компоненты

5.1. Архивы

Архивы предназначены для хранения постоянной информации системы и информации об обрабатываемых автоматах. Архивами постоянной информации являются архивы программного обеспечения и архивы элементов.

Архивы программного обеспечения содержат загрузочные модули программ и процедуры для выполнения основных процессов программной обработки данных.

Архивы элементов содержат разные данные об элементах автоматов. Элементом может служить микросхема или какой-то другой типичный модуль автоматов. Архивы содержат описания всех контактов каждого элемента и описания логических функций, которые выполняются данным элементом. Описания функций представлены в виде системы альтернативных графов [4] . Эти графы называются в дальнейшем стандартными графами (СГ) элемента. Посредством СГ элемента описывается взаимная зависимость логических значений контактов элемента. В СГ сложных элементов используются кроме действительных контак-

тов этого элемента и фиктивные (внутренние) контакты.

Архивы элементов могут одновременно содержать СГ микросхем многих серий. Для функционально тождественных элементов содержатся в архиве общие СГ.

Архивы элементов могут также содержать стандартные тесты для некоторых типичных модулей автоматов.

Архивы автомата содержат описания, модели и тесты автоматов. Одновременно могут в архивах содержаться данные многих автоматов. Всем автоматам необходимо назначать уникальное имя. Вопрос удаления из архивов больше ненужных автоматов решается пользователем системы.

Использование всех архивов автоматизировано. Чтение и запись данных, участвующих в процессах программной обработки данных, осуществляется процедурами этих процессов. При вызове процедур должен пользователь системы обязательно задавать имя обрабатываемого автомата. На основе заданного имени происходит поиск необходимых данных из архивов автоматов.

5.2. Компилятор модели

Компилятор модели предназначен для проверки правильности описания автомата и составления модели автомата. Модель автомата является в ПП информационной основой генерации и редактирования тестов автомата. Кроме этого, модель автомата используется в тестере при локализации неисправностей в схеме автомата.

Компиляция модели начинается анализом и проверкой описания автомата. При проверке сравнивается описание с данными из архива элементов. Программная проверка описания автомата не гарантирует правильность описания, но все-таки обнаруживает большинство ошибок.

Компилятор составляет таблицу элементов и таблицу переменных автомата. Для каждого элемента выбираются на основе типа элемента из архива нужные СГ. Все графы элементов собираются в таблицу графов автомата.

Всем основным понятиям назначаются цифровые идентификаторы. Обычно они представляют собой просто номер строки

таблицы, где находятся характеристики рассматриваемого понятия, т.е. идентификаторы являются указателями. Такими указателями соединяются между собой названные таблицы модели автомата. Система указателей в модели соответствует соединениям элементов автомата. Эти указатели используются только в программной обработке данных. В сообщениях системы используются символьные имена объектов, которые содержатся в описании автомата.

Для описания входов и выходов автомата составляется таблица разъемов. Общие данные об автомате собираются в паспорт автомата. Названные пять таблиц являются основными компонентами модели. Работа компилятора заканчивается записью модели в архив автоматов.

5.3. Генератор тестов

Генератор предназначен для автоматического составления контрольных тестов автомата. Генератор является управляемым - пользователь может задавать генератору предельное время работы, режимы использования разных методов генерации и перечень подсхем (выходов) автомата, которые подлежат обрабатыванию на данном сеансе. Генератору можно задавать также лимиты работы. Задание лимитов влияет на полноту получаемых тестов и на длительность генерации. Если пользователь не управляет генератором, то в работе применяются стандартные значения управляющих воздействий.

Основной операцией генератора является многократное решение разных систем задач фиксирования логических значений переменных автомата. Задача фиксирования (ЗФ) формулируется следующим образом: обеспечить переменной автомата y на такте t теста логическое значение $l \in \{0,1\}$.

Если переменная является функцией от аргумента x так, что $y^t = f(x^t, x^{t-1}, \ldots, x^{t-s})$, то формулированная задача приводит к решению уравнения $f(x^t\, x^{t-1}, \ldots, x^{t-s}) = l$. Фиксирование значения независимой переменной y решается просто присвоением переменной y на такте t требуемого значения l. Решение уравнения $f(x^t, x^{t-1}, \ldots, x^{t-s}) = l$ требует в общем случае выполнения следующих трех операций: моделирование, синтез вторичных ЗФ и коррекция дерева ЗФ.

Моделирование представляет собой процесс вычисления значения функции y^t. Оно возможно, если значения аргументов заранее фиксированы. При обнаружении в процессе вычисления аргументов, имеющих пустое значение, синтезируются для этих аргументов вторичные ЗФ. Переменная имеет пустое значение, если ее действительное значение еще не определено.

Синтез вторичных ЗФ требуется для обеспечения определенных значений аргументам функции, имеющим пустое значение. При синтезе решаются два вопроса:

- какое значение должен иметь аргумент (0 или I);

- является ли ЗФ этого значения критической.

ЗФ называется критической в том случае, если фиксирование аргумента рассматриваемого значения является необходимым условием для достижения нужного значения функции. Следует отметить, что часто оба вопроса в ходе синтеза нецелесообразно одинаково решать. В этом случае выбирается значение аргумента случайно. Если не будет решен вопрос о критичности, то следует соответствующую вторичную ЗФ в дальнейшем учитывать как некритическую задачу. Исходную ЗФ (первичную задачу) считаем обычно критической.

Новые вторичные ЗФ может порождать первичная или уже существующая вторичная задача. Система ЗФ расширяется до тех пор, пока возникают ЗФ для независимых переменных автомата. Полученная система ЗФ имеет древовидную структуру. Корневой вершиной дерева задач фиксирования (ДЗФ) является первичная задача. Вторичные ЗФ служат остальными вершинами дерева. Каждая вершина соединена дугой с вершиной, на основе которой она порождена.

Коррекция дерева ЗФ предназначена для изменения и удаления вторичных ЗФ. При изменении ЗФ инвертируется требуемое значение переменной и определяется снова критичность задачи. Инвертирование значения применяется только для некритических ЗФ. Обычно эти задачи становятся после инвертирования критическими задачами.

Суть коррекции ДЗФ заключается в изменении решений, принятых при синтезе ЗФ. Коррекция нужна в основном после неудачного решения критической ЗФ для исправления ошибок

синтеза. Описанный подход к решению ЗФ основывается на идее метода проб и ошибок. Другой причиной использования коррекции ДЗФ является необходимость просмотра нескольких вариантов решения первичной ЗФ. В таком случае корректируется ДЗФ такие после удачного решения первичной ЗФ.

Коррекция ДЗФ должна обеспечивать единственность процесса коррекции,т.е.при многократном применении коррекции нельзя порождать ни одну заранее уже просмотренного варианта ДЗФ. Несоблюдение этого условия может привести к бесконечному зацикливанию решения ЗФ. Другим необходимым условием коррекции является требование полноты перебора. Под полнотой перебора следует понимать свойство коррекции обеспечивать непосредственный или косвенный просмотр всех вариантов решения поставленной первичной ЗФ.Следует отметить,что при достаточно большом автомате строгое соблюдение условия полноты перебора не имеет смысла, так как практически оказывается неосуществимым просмотр всех вариантов решения первичной ЗФ. Заданием лимитов генерации можно выбрать компромисс между полнотой перебора и временных затрат на генерацию тестов.

При коррекции ДЗФ решаются три вопроса:

- какие задачи изменить;
- какие задачи удалить;
- какие задачи решить снова.

Все эти вопросы решаются в ходе просмотра существующего ДЗФ. При решении двух первых вопросов учитываются необходимые условия коррекции. Критерием решения третьего вопроса является то обстоятельство, что при изменении (удалении) ЗФ значений аргументов следует снова решить ЗФ значения функции.

Методы генерации тестов, описываемые в п.6, сводят проблемы вырабатывания теста к проблеме решения системы ЗФ. Для решения систем ЗФ предназначено ядро генератора (ЯГ). Предложенная методом генерации система ЗФ является для ЯГ системой первичных задач. Так как каждая первичная задача порождает одну ДЗФ, то работа ЯГ заключается в выполнении операции моделирования, синтеза и коррекции на множестве ДЗФ. Названные операции выполняются рекурсивной программой. Управляющим магазином программы является ДЗФ.

Исходная информация для ЯГ содержится в таблице первичных задач и в таблицах модели автомата. Из таблиц модели в ходе работы выбираются только необходимые для решения ЗФ переменные, т.е. ЯГ работает на модели автомата селективно. Результаты работы ЯГ записываются в таблицу значений переменных. В этой таблице содержатся значения всех переменных автомата по тактам генерируемого теста. Если какое-то значение не участвует в тесте, то на этом месте содержится в таблице пустое значение.

5.4. Редактор тестов

Редактор тестов предназначен для анализа и дополнения тестов, а также для окончательного оформления тест-процедуры автомата и соответствующей документации.

Сгенерированные тесты упаковываются, т.е. при возможности разные тесты объединяются. Упаковывание тестов увеличивает число одновременно проверяемых неисправностей и значительно сокращает длину тест-процедуры автомата.

В сгенерированных тестах некоторые входы и выходы автомата остаются часто неиспользованными даже после упаковывания тестов. Такие тесты доопределяются: неиспользованным входам присваиваются определенные значения (0 или 1), а значения неиспользованных выходов вычисляются моделированием теста. Моделирование происходит по многозначному алфавиту на альтернативной граф-модели автомата [5] . При этом осуществляется окончательная проверка корректности теста. Проверка выходов с неопределенным значением блокируется. Сгенерированные тесты переводятся на язык МЭМТЕСТ.

Полнота тестов оценивается окончательно методом реверсивного анализа [6] . Результаты анализа оформляются в виде списка необнаруживаемых неисправностей. Полноту полученных тестов можно увеличить проведением нового сеанса генерации и введением необходимых дополнительных тестов вручную. Новый сеанс генерации следует провести с увеличенными лимитами генерации, что приводит к увеличению времени работы ЭВМ. Часто оказывается более целесообразным написать трудно генерируемые тесты вручную. Дополнительные тесты можно также анализировать редактором.

Окончательная тест-процедура автомата составляется объединением сгенерированных и дополнительных тестов.

6. Методы генерации тестов

6.1. Этапы генерации

Разные методы генерации тестов реализованы в генераторе по одному общему плану, который предусматривает выполнение четырех этапов работы:

- планирование теста;
- составление базового решения;
- дополнение базового решения;
- формирование теста.

На этапе планирования теста выбираются действия на модели, которые необходимо выполнить для получения теста. Совокупность выбранных действий и условий называется планом теста. При планировании теста оцениваются надобность и реализуемость этого теста. Планирование теста заканчивается переводом плана в систему ЗФ, которая помещается в таблицу первичных задач.

В результате решения системы ЗФ, заданной планом теста, получается базовое решение (БР). Система ЗФ решается при помощи ЯГ. Полученное БР анализируется. При необходимости улучшения БР выбираются дополнительные действия, которые также переводятся в систему ЗФ. Анализ и дополнение БР можно осуществить повторно, т.е. тест можно выработать итеративным способом. Этот этап заканчивается запоминанием неисправностей, проверяемых полученным тестом.

На этапе формирования теста на основе таблицы значений переменных составляются последовательности входных и выходных наборов автомата. Полученные наборы (тесты) записываются в архив автомата.

6.2. Генерация тестов моделированием неисправностей

При планировании теста выбирается выход автомата и составляется для соответствующей выходной переменной ЗФ.

Какое значение следует фиксировать, решается на основе признаков проверенности неисправностей. Если никаких признаков не имеется, то значение выбирается произвольно.

Таблица первичных задач содержит после планирования только одну ЗФ. При решении этой задачи ЯГ синтезирует ДЗФ и на основе каждой вершины дерева фиксирует одно значение некоторого переменного автомата. Фиксированные значения называем существенными значениями. Совокупность этих значений является базовым решением.

БР является тестом определенной константной неисправности выхода автомата. Целью последующего дополнения БР является получение новых тестов на базе существенных значений. Новые тесты строятся для неисправностей, которые приводят к инвертированию рассматриваемого существенного значения. Такими неисправностями могут быть константные значения контактов элементов, а также короткие замыкания и другие. Неисправность, для которой следует строить тест, вводится в модель автомата. В результате ввода неисправности некоторые существенные значения становятся недействительными. Такие значения находятся просмотром ДЗФ базового решения. Недействительные значения заменяются в таблице логических значений переменных пустым значением. Для выхода автомата поставляется дополнительная ЗФ: фиксировать на выходе автомата значение, которое отличается от значения выхода в БР. Дополненное БР является искомым новым тестом.

Перед исследованием дальнейших возможностей дополнения БР восстанавливаются БР и модель в исходном виде. На основе результатов дополнения формируется один или несколько тестов. Новое БР планируется коррекцией ДЗФ старого БР выбором новой первичной ЗФ.

Описанный метод по сути дела представляет собой обратный процесс моделирования автомата с неисправностями.

6.3. Генерация тестов активизацией пути

На этапе планирования теста на модели планируется путь от одного выхода до некоторого входа автомата. Этому пути соответствует упорядоченное множество альтернативных графов,

называемых базовыми графами. Переменные пути назовем базовыми переменными.

Целью метода является генерация тестов для неисправностей на пути посредством его активизации. При этом возможно получение парных тестов [4] . Задача генерации тестов сводится на решение ЗФ, которые составляются на основе базовых графов. В ходе составления БР регистрируются все случаи, когда обнаруживается зависимость БР от базовых переменных. Для таких зависимостей составляются задачи нейтрализации, которые предусмотрены для блокировки ветвей активного пути. Решением задач нейтрализации снижается размерность пути и увеличивается число обнаруживаемых неисправностей. Следует отметить, что метод не использует ввода неисправностей в модель. Решение задач нейтрализации сводится также к решению системы ЗФ.

6.4. Модульный подход к генерации тестов

Метод используется для проверки модулей автомата заранее подготовленными для них стандартными тестами. Эти модули должны быть определены в описании автомата. Стандартный тест для проверки модуля выбирается из архива на основе типа этого модуля, следовательно, для всех однотипных модулей используется один и тот же стандартный тест.

При планировании теста модуля выбирается из архива требуемый стандартный тест и конструируется путь от выхода автомата до выхода модуля (при необходимости следует конструировать несколько путей). Этот путь активизируется аналогично предыдущему методу генераций тестов. Устранение многомерности пути не происходит. Активный путь должен обеспечить наблюдаемость значения выхода модуля на выходе автомата. Входным переменным модуля фиксируются предусмотренные стандартным тестом значения. На основе эталонного значения выхода модуля определяется эталонное значение выхода автомата. Перечисленные действия выполняются для каждого такта стандартного теста.

Если в автомате используются функциональные возможности модуля частично (например, некоторые контакты модуля оставлены свободными), то стандартный тест можно реализовать

также частично. Эффективность метода существенно зависит от структуры автомата и от способа выделения модулей. Оказывается, что в малоэффективных случаях среда модуля действует как фильтр, который не пропускает многие наборы значений переменных, которые предусмотрены стандартным тестом модуля. Метод оказывается эффективным для автоматов, имеющих магистральную структуру (в частности, для микропроцессорных схем).

Литература

1. Убар Р.Р. Описание неисправностей цифровых устройств. - Тр. Таллинск. политехн. ин-та, 1980, № 497, с. 3–9.

2. Плакк М.П., Убар Р.Р. Построение тестов цифровых схем при помощи модели альтернативных графов. - Автоматика и телемеханика, 1980, № 5, с. 152–163.

3. Гуляев В.А., Макаров С.М., Новиков В.С. Диагностика вычислительных машин. Киев, Техника, 1981. 168 с.

4. Убар Р.Р. Тестовая диагностика цифровых устройств, ч. I и II. Таллинский политехнический институт. 1981, с. 226.

5. Воолайне А.А., Паллъ М.А., Убар Р.Р. Обобщенный подход к многозначному моделированию цифровых схем на модели альтернативных графов. - Тр. Таллинск. политехн. ин-та, 1982, № 530, с. 23.

6. Китсник П.А., Убар Р.Р. Формулы для дедуктивного анализа тестов в синхронных последовательностных схемах. - Тр. Таллинск. политехн. ин-та, 1977, № 432, с. 15–23.

A. Voolaine, A. Jõgi, M. Pall

Automatic Test Generating System for Digital Devices

Summary

The paper represents an automatic test generating system for digital units under test. The system is meant to verify the description, to build a model, to build a test procedure, and to check the completeness of the procedure for the unit under test. The system operates on EC computers.

№ 530

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.32

А.А. Воолайне, М.А. Палль,
Р.Р. Убар

ОБОБЩЕННЫЙ ПОДХОД К МНОГОЗНАЧНОМУ МОДЕЛИРОВАНИЮ ЦИФРОВЫХ СХЕМ НА МОДЕЛИ АЛЬТЕРНАТИВНЫХ ГРАФОВ

I. Введение

При анализе цифровых схем часто используются различные виды моделирования, позволяющие воспроизводить поведение схемы при заданном наборе входных воздействий. Логическое моделирование сводится к решению системы булевых уравнений, каждое из которых моделирует логику переключения одного элемента. Для выявления состязаний и риска сбоя применяется многозначное моделирование, в частности, троичное моделирование [1] , где сигналы принимают значения из множества $A_3 = \{0, 1, x\}$, или пятеричное моделирование [2] , где сигналы принимают значения из множества $A_5 = \{0, 1, \varepsilon, h, x\}$. Здесь x – обозначает неопределенное (или неизвестное) значение сигнала, ε – обозначает монотонное изменение сигнала в направлении $0 \rightarrow 1$ и h – обозначает монотонное изменение сигнала в направлении $1 \rightarrow 0$.

Недостатками известных методов многозначного моделирования являются высокая размерность используемой модели объекта (требуется описание схемы на уровне логических элементов) и необходимость применения различных алгоритмов обработки для каждого элемента в отдельности. В данной статье предлагается модель описания цифровых схем в виде системы альтернативных графов [3] , позволяющая снизить размерность модели объекта, а также применить универсальный алгоритм не только для различных элементов схемы, но и для разных множеств A_i .

2. Математические положения предлагаемого подхода

Зададим цифровую схему в виде множества булевых функций F на множестве переменных Z, причем любая функция $z_k = f_k(Z_k)$, где $z_k \in Z$, $Z_k \subset Z$ и $f_k \in F$ соответствует некоторому элементу схемы (компоненту, подсхеме, интегральной схеме и т.д.).

Представим каждую функцию $z_k = f_k(Z_k)$ альтернативным графом $G_k = \{M_k, \Gamma_k\}$, где M_k – множество вершин графа, а Γ_k – отображение множества M_k в множество M_k. С каждой вершиной графа $m \in M_k$ сопоставлена некоторая переменная $z(m) \in Z_k$. Согласно определению модели АГ [3] вершины графа, а следовательно, весовые переменные $z(m)$ представляют определенные пути в исходной схеме (строго определенные одномерные пути в частном случае структурных АГ (САГ) и нестрого определенные многомерные пути в общем случае функциональных АГ (ФАГ)).

Введем обобщенный алфавит A для значений сигналов при многозначном моделировании. Пусть $A_2 = \{0,1\} \subset A$ двоичный алфавит статических сигналов для двоичного моделирования, а $A_g = A \setminus A_2$ – алфавит динамических сигналов, которые отображают ситуацию, когда напряжение в рассматриваемой точке или изменяется по какому-то закону, или его значение неизвестно. Так, например, при троичном моделировании $A_g = \{x\}$, при пятеричном моделировании $A_g = \{\varepsilon, h, x\}$.

Значение функции может быть определено двоичным моделированием, представляющим собой процесс движения по вершинам графа G_k, так чтобы направление выхода из каждой вершины m определялось значением весовой переменной вершины $z(m)$ [3]. В случае, когда значение переменной $z(m)$ относится к множеству A_2, направление движения на графе определено однозначно (направо при $z(m) = 1$ и вниз при $z(m) = 0$). В случае, когда значение переменной $z(m)$ статически не определено, т.е. относится к множеству $A_g \subset A$, направление выхода из вершины m не определено. Для исследования данной ситуации может быть применен аппарат булевого дифференцирования.

Известно, что значение функции $z_k = f_k(Z_k)$ зависимо от значения переменной $z(m) \in Z_k$, если выполнено равенство

$$\partial f_k / \partial z(m) = 1 . \qquad (I)$$

Применительно к модели АГ, выполнению условия (I) соответствует существование на графе при заданном входном наборе, т.е. при заданных значениях переменных $z(m)$, следующих путей:

- путь из начальной вершины $m^н$ графа G_k в вершину m;

- путь из m при значении $z(m) = 1$ к одному из выходов АГ $l \in \{0, 1\}$;

- путь из m при значении $z(m) = 0$ к другому из выходов АГ $\bar{l} \in \{0, 1\}$.

Рассмотрим некоторый входной набор Z_k^t логической схемы с функцией $z_k = f_k(Z_k)$, где переменные $z_i \in Z_k$ принимают, соответственно, значения $z_i^t \in A$. Пусть $Z_{k,g} \subseteq Z_k$ -подмножество переменных, имеющих на наборе Z_k^t значения из алфавита A_g.

Предположим, что $z(m) \in Z_{k,g}$, а для всех остальных вершин $m_i \in M_k \setminus m$ значения весовых переменных статические, т.е. $z(m_i) \notin Z_{k,g}$. Тогда легко доказать следующие леммы.

Лемма I. Чтобы значение функции $z_k = f_k(Z_k)$ при наборе Z_k^t, где $z(m) \in Z_{k,g}$, оказалось статическим $f_k^t \in A_2$, необходимо и достаточно выполнение условия

$$\partial f_k / \partial z(m) = 0 .$$

Лемма 2. Значение функции $z_k = f_k(Z_k)$ при наборе Z_k^t, где $z(m) \in Z_{k,g}$ будет динамическим $f_k^t \in A_g$, если

$$\partial f_k / \partial z(m) = 1 .$$

Лемма I определяет условия, когда процесс движения на графе можно считать однозначно-определенным, несмотря на неопределенности в вершине m (из вершины можно выйти в произвольном направлении). Лемма 2 определяет условия, когда многозначный сигнал $\alpha \in A_g$ на входе $z(m)$ передается на выход схемы в некотором виде $\alpha' \in A_g$ (лемма 2 не конкретизирует вид α' ; возможно как $\alpha' = \alpha$, так и $\alpha' \neq \alpha$).

Рассмотрим элементы множества А в следующем отношении

$$0 < \alpha < 1 , \qquad (2)$$

где $\alpha \in A_g$.

Тогда может быть введена функция

$$\max_{z(m_i)} \{ \partial f_k / \partial z(m) \} , \qquad (3)$$

где $z(m)$, $z(m_i) \in Z_{k,g}$ и $m \neq m_i$.

Заметим, что в производной $\partial f_k / \partial z(m)$ анализируется изменение переменной $z(m)$ только при вершине m (аналогично тому, как в эквивалентных нормах булевых функций рассматриваются изменения букв [4]). Изменения $z(m_i)$ для всех $m_i \in M_k$ в функции максимума и $z(m)$ в производной независимы даже тогда, когда $z(m_i)$ и $z(m)$ совпадают.

Функция (3) представляет собой обобщение булевого производного для случая многозначных сигналов. Наиболее важным свойством функции (3) является ее равенство I, если динамический сигнал $z^t(m) \in A_g$ на входе может каким-то образом в течение рассматриваемого такта t влиять на значение сигнала на выходе z_k данной схемы, превращая этот сигнал также в динамический.

Используя функцию максимума булевого производного и базируясь на лемме I можно доказать следующее утверждение.

Лемма 3. Чтобы значение функции $z_k = f_k(Z_k)$ при наборе Z_k^t не зависело от значения $z(m) \in Z_{k,g}$, необходимо и достаточно выполнение условия

$$\max_{z(m_i)\in Z_{k,g}} \partial f_k/\partial z(m)=0.$$

Из леммы 3 непосредственно вытекает следующая теорема.

Теорема I. Чтобы значение функции $z_k=f_k(Z_k)$ при наборе Z_k^t и при $|Z_{k,g}|\neq 0$ оказалось статическим $f_k^t\in A_2$, необходимо и достаточно выполнение условия

$$\bigvee_{z(m)}\ (\max_{z(m_i)}\ \partial f_k/\partial z(m))=0,$$

где $z(m),\ z(m_i)\in Z_{k,g}$.

Используя функцию максимума булевого производного и базируясь на лемме 2, легко доказать следующую теорему.

Теорема 2. Значение функции $z_k=f_k(Z_k)$ при наборе Z_k^t и при $|Z_{k,g}|\neq 0$ будет динамическим $f_k^t\in A_g$, если

$$\bigvee_{z(m)}\ (\max_{z(m_i)}\ \partial f_k/\partial z(m))=1,$$

где $z(m),\ z(m_i)\in Z_{k,g}$.

На базе теорем I и 2 может быть построен алгоритм многозначного моделирования, где аргументы принимают значения из произвольного алфавита A, $|A|\geq 3$, а функции принимают значения из $A_3=\{0,1,X\}$. При этом значение X присваивается функции в случае, когда выполняются условия теоремы 2.

Функция максимума булевого производного является простым средством интегрального подхода к анализу поведения схемы при динамических сигналах. Для более точного анализа требуется разложение тактов, где действуют динамические сигналы на части, с целью анализа поведения схемы на уровне составляющих этих сигналов.

3. Многозначное моделирование на модели альтернативных графов

При определении модели АГ [3] введены понятия весовых переменных и весовых функций для вершин графа. Обобщим понятие весовой функции в следующем виде:

$$S(m) = F_S(a(m), z(m)), \qquad (4)$$

где $a(m)$ – атрибут вершины, определяющий закон, как нужно интерпретировать в вершине m значение переменной $z(m)$. Некоторая система атрибутов иллюстрирована в табл. I (система может быть расширена в зависимости от функциональных особенностей объектов).

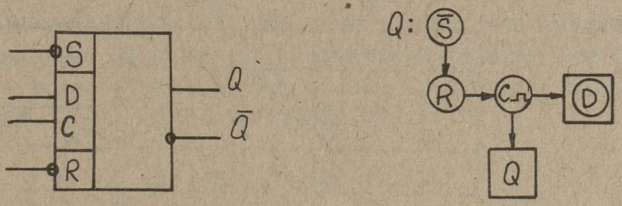
Таблица I

$a(m)$	Графическое обозн. вершины	Закон интерпретации $z(m)$
0	(z)	$S(m) = z(m)$
I	($\bar{z}$)	$S(m) = \bar{z}(m)$
2	[z]	$S(m) = z^{t-1}(m)$
3	[$\bar{z}$]	$S(m) = \bar{z}^{t-1}(m)$
4	($z_л$)	$S = \begin{cases} 1, \text{ если } z(m) = \delta \\ 0, \text{ если } z(m) \neq \delta \end{cases}$
5	[(z)]	$S(m) = z(m), z(m)$ должна быть синхронизована
6	[($\bar{z}$)]	$S(m) = \bar{z}(m), z(m)$ должна быть синхронизована

Атрибуты $a(m)$ устанавливаются в модель АГ в зависимости от функциональных особенностей объекта и могут учитывать такие моменты, как необходимость инвертирования сигнала (когда на пути схемы, представляемом переменной $z(m)$, нечетное количество инверторов, например, при значениях $a(m) = I, 3, 6$ в табл. I), необходимость задержки сигнала (когда $z(m)$ предоставляет собой обратную связь или выход элемента памяти, например, при $a(m) = 2, 3$), реагирование соответствующего входа элемента $z(m)$ на определенный вид сигнала

(на определенные фронты, импульсы, импульсные последовательности, например, при $a(m) = 4$), необходимость синхронизации сигнала $z(m)$ (например, при $a(m) = 5, 6$) и т.д.

Пример использования приведенной системы атрибутов при представлении функции D-триггера моделью АГ рассматривается на фиг. I.



Фиг. 1.

Возьмем в качестве примера 6-значный алфавит $A_6=\{0,1,\varepsilon,h,\delta,x\}$, динамика сигналов которых иллюстрирована в табл. 2. При этом такт разложен на три части и для указания сигнала на этих частях используется алфавит $A_3 = \{ 0, I, x \}$.

Таблица 2

Интегральный сигнал в такте	Разложенный сигнал		
t	t_1	t_2	t_3
0	0	0	0
I	I	I	I
ε	0	0 v I	I
h	I	I v 0	0
δ	0	I	0
X	X	X	X

Значения весовой функции $S(m)$ для всех комбинаций значений $a(m)$ и $z(m)$ приведены в табл. 3.

Таблица 3

$a(m)$ \ $S(m)$	$z(m)$: 0	I	ε	h	δ	X
0	0	I	ε	h	δ	X
I	I	0	h	ε	X	X
2	0	I	I	0	X	X
3	I	0	0	I	X	X
4	0	0	0	X	ε	X
5	0	I	X	X	X	X
6	I	0	X	X	X	X

При составлении таблицы 3 учтены как функциональные особенности объекта (физическая суть атрибутов $a(m)$), так и физика самих сигналов, подаваемых на входы $z(m)$. Так, например, при $a(m) = 5, 6$ все сигналы $\alpha \in A_g$ приведут к неопределенному результату, так как могут быть некорректно синхронизируемы. Сигнал h при $a(m) = 2$ будет принят как 0, так как из-за задержки принимается установившееся значение процесса $I \longrightarrow I \vee 0 \longrightarrow 0$.

Рассмотрим некоторую вершину $m \in M_K$ в графе G_K. Каждой такой вершине m соответствует некоторый подграф $G_K(m) \subseteq G_K$, так чтобы m являлось начальной вершиной для $G_K(m)$. Функцию, представляемую графом $G_K(m)$ обозначим через $f_K(m)$.

Определим далее для вершины m <u>правый подграф</u> $G_K(m,1)$ с соответствующей функцией $f_K(m,1)$ и <u>нижний подграф</u> $G_K(m,0)$ с соответствующей функцией $f_K(m,0)$. В частном случае, когда отсутствуют правый или нижний последователь вершины m, предположим, что $f_K(m,1) \equiv 1$ и $f_K(m,0) \equiv 0$.

Многозначное моделирование цифровой схемы по произвольному алфавиту A может быть проведено на модели АГ по следующему рекуррентному выражению:

$$f_K(m) = F_D(S(m), f_K(m,1), f_K(m,0)). \qquad (5)$$

Для вычисления значения функции $f_K(m)$ требуется анализ поведения объекта отдельно по частям такта, на которые сигналы разложены, т.е. по микротактам t_i $(i=1,2,3)$. Все значения $S(m)$, согласно табл.2, приводимы во временную последовательность сигналов из алфавита A_3. Физически последовательность $S(m)=\{S_i(m)|i=1,2,3\}$ указывает в каком направлении продолжать движение на графе при моделировании функции для определенного микротакта t_i. Отсюда следует алгоритм вычисления значений $f_{K,i}(m)$ как составляющих $f_K(m)$:

$$f_{K,i}(m) = \begin{cases} f_{K,i}(m,1), & \text{если } S_i(m)=1; \\ f_{K,i}(m,0), & \text{если } S_i(m)=0; \\ \alpha, & \text{если } S_i(m)=X; \end{cases} \qquad (6)$$

$$i=1,2,3,$$

где

$$\alpha = \begin{cases} f_{K,i}(m,1)=f_{K,i}(m,0), & \text{если } \partial f_{K,i}(m)/\partial z(m) = 0 \\ X, & \text{если } \partial f_{K,i}(m)/\partial z(m) = 1. \end{cases} \qquad (7)$$

Заметим, что

$$\partial f_{K,i}(m)/\partial z(m) = f_{K,i}(m,1) \oplus f_{K,i}(m,0). \qquad (8)$$

В частном случае, если $S(m) \in \{0,1\}$, имеем

$$f_K(m) = \begin{cases} f_K(m,1), & \text{если } S(m)=1 \\ f_K(m,0), & \text{если } S(m)=0. \end{cases} \qquad (9)$$

Если значения $f_{K,i}(m) \in A_3$ известны, то легко, согласно табл.2, перейти к интегральным значениям $f_K(m) \in A_6$.

$S(m)=\varepsilon$	$f_K(m,1)$					
$f_K(m,0)$	0	1	ε	h	δ	X
0	0	ε	ε	X	X	X
1	h	1	X	h	X	X
ε	X	ε	X	X	X	X
h	h	X	X	X	X	X
δ	X	X	X	X	X	X
X	X	X	X	X	X	X

$S(m)=h$	$f_K(m,1)$					
$f_K(m,0)$	0	1	ε	h	δ	X
0	0	h	X	h	X	X
1	ε	1	ε	X	X	X
ε	ε	X	X	X	X	X
h	X	h	X	X	X	X
δ	X	X	X	X	X	X
X	X	X	X	X	X	X

$S(m)=\delta$	$f_K(m,1)$					
$f_K(m,0)$	0	1	ε	h	δ	X
0	0	δ	X	X	X	X
1	X	1	X	X	X	X
ε	X	X	X	X	X	X
h	X	X	X	X	X	X
δ	X	X	X	X	X	X
X	X	X	X	X	X	X

$S(m)=X$	$f_K(m,1)$					
$f_K(m,0)$	0	1	ε	h	δ	X
0	0	X	X	X	X	X
1	X	1	X	X	X	X
ε	X	X	X	X	X	X
h	X	X	X	X	X	X
δ	X	X	X	X	X	X
X	X	X	X	X	X	X

Фиг. 2.

Вычисления могут быть проведены на уровне интегральных сигналов алфавита A_6, т.е. без перехода на микротакты, если значения функции $f_K(m)$ заранее подготовить таблично.

Примеры. Таблицы вычисления $f_K(m)$ при значениях $S(m) \in \{\varepsilon, h, \delta, X\}$ приведены на фиг.2.

4. Организация общего процесса моделирования

Рассмотрим общий процесс многозначного моделирования на альтернативном графе G_K. Процесс основывается на использовании (4) и (5) и можно представить следующим рекурсивным алгоритмом.

Алгоритм I. (Входной параметр m – вершина графа).

I. Определим значение $S(m)$ по формуле (4).

2. Если отсутствует правый последователь вершины m, то $f_K(m,1) := 1$; перейти к шагу 5.

3. Если значение $f_K(m,1)$ уже определено, то перейти к п.5.

4. Определим значение $f_K(m,1) = f_K(m^1)$ по алгоритму I. Здесь m^1 – правый последователь вершины m.

5. Если отсутствует нижний последователь вершины m, то $f_K(m,0) := 0$; перейти к шагу 8.

6. Если значение функции $f_K(m,0)$ уже определено, то перейти к шагу 8.

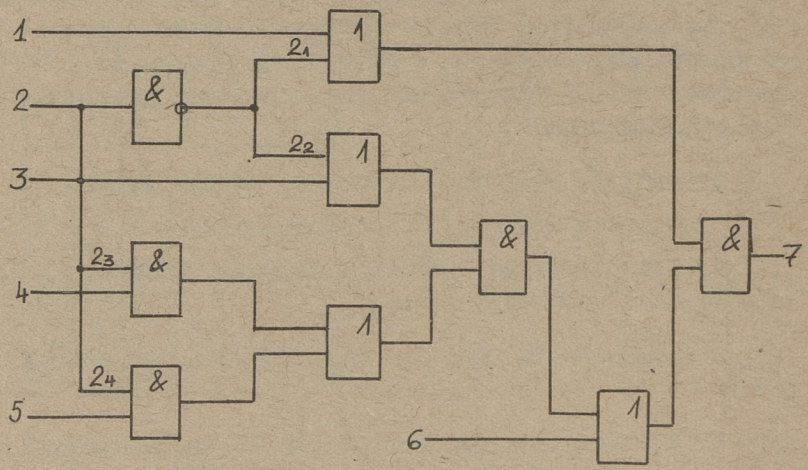
7. Определим значение функции $f_K(m,0) = f_K(m^0)$ по алгоритму I. Здесь m^0 – нижний последователь вершины m.

8. Определим значение функции $f_K(m)$ по формуле (5).

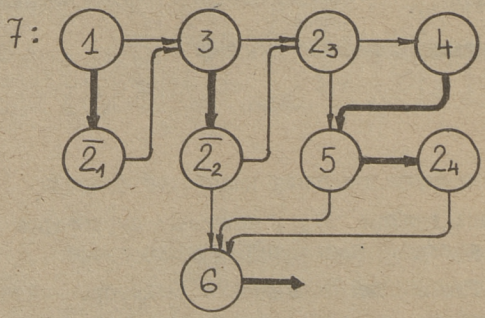
Конец.

При вычислении значения некоторой функции $z_K = f_K(Z_K)$, представленной графом G_K, исходным параметром m для алгоритма I является начальная вершина графа.

Пример. Рассмотрим поведение комбинационной схемы, представленной на фиг. 3 в виде логической сети и на фиг.4 в виде модели структурного АГ при подаче входного набора (0 ε 0 0 1 1). Процесс моделирования иллюстрирован в табл. 4.



Фиг. 3.



Фиг. 4.

Таблица 4

Шаг	m	z(m)	S(m)	$f_k(m,0)$	$f_k(m,1)$	Замечания
I	I	0	0			вычислить $z_7 = f_7(1)$
2	$\bar{2}_1$	ε	h			вычислить $f_7(1) = f_7(\bar{2}_1)$, записать задачу $f_7(\bar{2}_1)$ в магазин
3	3	0	0			вычислить $f_7(\bar{2}_1,1) = f_7(3)$
4	$\bar{2}_2$	ε	h			вычислить $f_7(3) = f_7(\bar{2}_2)$, записать задачу $f_7(\bar{2}_2)$ в магазин
5	2_3	ε	ε			вычислить $f_7(\bar{2}_2,1) = f_7(2_3)$, записать задачу $f_7(2_3)$ в магазин
6	4	0	0			вычислить $f_7(2_3,1) = f_7(4)$
7	5	I	I			вычислить $f_7(4) = f_7(5)$
8	2_4	ε	ε		I	вычислить $f_7(5) = f_7(2_4)$, записать задачу $f_7(2_4)$ в магазин, $f_7(2_4,1) := 1$
9	6	I	I		I	вычислить $f_7(2_4,0) = f_7(6) = 1$, читать задачу $f_7(2_4)$ из магазина
I0	2_4	ε	ε	I	I	$f_7(2_4) := 1$, $f_7(2_3,1) := 1$
II	5	I	I			вычислить $f_7(2_3,0) = f_7(5)$
I2	2_4	ε	ε	I	I	вычислить $f_7(5) = f_7(2_4) = 1$, $f_7(2_3,0) := 1$ читать задачу $f_7(2_3)$ из магазина

I3	2_3	ε	ε	I	I	$f_7(2_3):=1$, $f_7(\bar{2}_2,1):=1$
I4	6	I	I		I	вычислить $f_7(\bar{2}_2,0)=$ $=f_7(6)=1$, читать задачу $f_7(\bar{2}_2)$ из магазина
I5	$\bar{2}_2$	ε	ε	I	I	$f_7(\bar{2}_2):=1$, $f_7(\bar{2}_1,1):=1$
I6	$\bar{2}_1$	ε	ε	0		вычислить $f_7(\bar{2}_1,0)$, $f_7(\bar{2}_1,0):=0$, читать задачу $f_7(\bar{2}_1)$ из магазина
I7	$\bar{2}_1$	ε	ε	0	I	$f_7(\bar{2}_1):=h$, $z_7:=h$

5. Заключение

Преимуществом предложенного метода многозначного моделирования является его универсальность как относительно элементной базы моделируемых объектов, так и относительно выбранного алфавита моделирования. Каждый элемент естественно должен быть представлен своим графом, но алгоритм моделирования одинаковый для всех графов. При введении в реализованный алфавит A новых сигналов должно быть определено (или доопределено при введении новых микротактов) кодирование этих сигналов в табл. 2, а также введен новый столбец в табл. 3 для каждого нового сигнала. Алгоритм самого моделирования не изменяется.

Заметим,что при традиционных подходах к многозначному моделированию [I, 2], каждый элемент требует свой алгоритм обработки, изменяющийся при изменении алфавита моделирования.

Литература

I. Eichelberg E.B. Hazard detection in combinational and sequential switching circuits. - IBM J. Res. Dev., 1965, v. 9, Mar.

2. Muth P. Ein Verfahren zur Erkennung statisches und dynamisches Hazards in Schaltnetzen. - Elektron. Rechenanlagen, 1974, Nr. 5.

3. Убар Р. Тестовая диагностика цифровых устройств. Таллинский политехнический институт, 1980. 114 с.

4. Основы технической диагностики / Под ред. П.П. Пахоменко. М., Энергия, 1976. 464 с.

A. Voolaine, M. Pall, R. Ubar

Verallgemeinertes Verfahren zur mehrwertigen Modellierung von digitalen Schaltungen auf dem Modell von alternativen Graphen

Zusammenfassung

Es wird eine Methode der mehrwertigen Modellierung von digitalen Schaltungen auf dem Modell von alternativen Graphen vorgeschlagen. Die Methode ist universal sowohl für die beliebige Elementbasis als auch für das beliebige Alphabet von Signalen. Für die Begründung der Methode wird der Kalkül der Booleschen Differenzen verwendet.

№ 530

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.32

П.А. Китсник

АНАЛИЗ ПОЛНОТЫ ТЕСТОВ ДЛЯ ЦИФРОВЫХ СХЕМ НА МОДЕЛИ АЛЬТЕРНАТИВНЫХ ГРАФОВ

Рассматривается задача определения множества неисправностей цифровой схемы (ЦС), обнаруживаемых заданными тестами. К решению этой задачи существует два подхода: обработка тестов путем моделирования неисправностей и аналитическая обработка тестов. При этом более эффективной следует считать аналитическую обработку тестов, в частности, для больших ЦС [1].

Наиболее известным аналитическим методом является дедуктивное моделирование тестов, описанное в работе [2]. В [3, 4] представлен реверсивный (обратный дедуктивный) метод для анализа полноты тестов, обладающий следующими преимуществами по сравнению с дедуктивным методом [2]:

- возможность прямого расширения класса рассматриваемых логических неисправностей;

- проведение анализа тестов только в активизированных ими частях ЦС;

- возможность проведения параллельного анализа тестов.

Это достигается, во-первых, реверсивным прослеживанием схемы при анализе (от выходов к входам), во-вторых – определением множества активизированных тестами узлов ЦС и доопределением по этому множеству конкретных неисправностей из заданного класса.

Эффективность процесса анализа тестов зависит также

от используемой модели объекта диагностирования. Перспективной можно считать модель ЦС в виде системы альтернативных графов (АГ) [5,6] . Основными преимуществами этой модели, с точки зрения решения задач анализа тестов, являются:

- компактность по сравнению с традиционными методами описания ЦС;

- сохранена возможность применения методов анализа тестов, базирующихся на прослеживании активизированных путей в ЦС;

- регулярность модели позволяет заметно упростить разработку методов и алгоритмов для системы АГ по сравнению с традиционными моделями ЦС.

Согласно положениям работы [5] модель АГ задается в виде тройки

$$\Phi = (G, Z, V),$$

где $G = \{G_k\}$ – множество АГ;

Z – множество булевых переменных ;

V – отображение G в Z .

Здесь

$$G_k = (M_k, \Gamma_k),$$

где M_k – непустое множество вершин ;

Γ_k – отображение M_k в M_k (при этом $\forall m \in M_k$: $|\Gamma_k m| \leq 2$).

Через V_G (V_G^{-1}) обозначается отображение, сопоставляющее каждому графу G_k переменную $z_k = V_G(G_k)$ ($G_k = V_G^{-1}(z_k)$) , а через V_M (V_M^{-1}) – отображение, сопоставляющее каждой вершине $m \in M$ весовую переменную $z(m) = V_M(m)$ ($m = V_M^{-1}(z(m))$). Множества входных и выходных переменных ЦС обозначаются соответственно через $Z_{вх}$ и $Z_{вых}$.

Моделирование тестов на АГ

Решение задач анализа тестов базируется на моделировании. Для моделирования тестов на модели АГ каждому G_k определяется оператор моделирования $G_k(m_{k0}, t)$, вычисляющий значение переменной $z_k = V_G(G_k)$ для такта времени t. Моделирование начинается с начальной вершины $m_{k0}: \Gamma_k^{-1} m_{k0} = \phi$. В каждой вершине $m \in M_k$ вычисляется весовая функция [5] в виде

$$e(m) = \beta(m) \oplus z(m)^{t-\tau(m)}, \qquad (I)$$

где $\beta(m)$ – признак инвертирования весовой переменной $z(m)$;

$\tau(m)$ – признак запаздывания на один такт.

По значению $e(m) \in \{0,1\}$ выбирается очередная вершина из множества $\Gamma_k m$. Моделирование завершается выходом из G_k, при этом значение z_k определяется направлением выхода из АГ.

Моделирование тестов на АГ имеет следующие преимущества по сравнению с моделированием на традиционных моделях ЦС:

– оператор моделирования проходит только определенное подмножество вершин;

– за счет компактности модели объем моделирования сокращается;

– за счет регулярности модели каждый шаг моделирования является стандартным (заключается в вычислении по (I)).

Однако согласно специфике модели АГ моделирование необходимо провести для каждого выхода ЦС в отдельности. Другими словами, моделирование теста осуществляется путем последовательного выполнения операторов $G_k(m_{k0}, t)$, где $V_G(G_k) \in Z_{вых}$.

Учитывая вышеизложенное, подход к анализу тестов путем моделирования неисправностей можно считать перспективным. Основной недостаток подхода – большая затрата машинного времени – на модели АГ выявляется в меньшей степени. Дополнительное повышение эффективности процесса анализа достигается тем, что количество моделируемых неисправностей заметно сокращается. Действительно, имеет смысл моделировать неисправности только в тех вершинах АГ, которые оказываются на пути операторов моделирования.

Таким образом, преимуществами метода моделирования неисправностей на модели АГ по сравнению с соответствующим методом на традиционных моделях ЦС, являются:

– сокращение общего объема моделирования;

– упрощение процесса моделирования, а также ввода неисправностей в модель;

– значительное сокращение количества моделируемых неисправностей.

Недостаток по сравнению с реализациями на традиционных моделях заключается в том, что параллельное моделирование на АГ не достаточно эффективно. Причиной является потеря возможности селективного прохода АГ в случае параллельной обработки либо неисправностей, либо тестов.

Анализ тестов на АГ

Ниже предлагается реализация реверсивного подхода к анализу полноты тестов на модели АГ. Согласно положениям работ [3,4] реверсивный анализ проводится с целью определения множества активизированных тестами узлов ЦС. При задании класса логических неисправностей, определяются также конкретные неисправности, обнаруживаемые проанализированными тестами.

Определение I. Вершина $m \in M$ является для заданного теста существенной, если изменение значения весовой функции $e(m)$ приводит к изменению значения некоторой выходной переменной $z_k \in Z_{вых}$.

Обозначим множество существенных вершин через M_c. Поскольку рассматривается конкретный тест, то его номер в обозначении не указан.

Легко убедиться, что тестом могут быть обнаружены неисправности только в тех узлах ЦС, которым в модели АГ сопоставлены весовые переменные $z(m) = V_M(m), m \in M_c$. Таким образом, задачу анализа полноты тестов на модели АГ можно сформулировать в виде задачи определения множеств существенных вершин M_c.

Рассмотрим сначала определение множества локально существенных вершин M_c^k для G_k, т.е. определение вершин, существенных относительно переменной $z_k = V_G(G_k)$. Согласно изложенному, вершина $m \in M_c^k$, если выполнено два условия:

- $m \in M_k$ находится на пути оператора моделирования $G_k(m_{k0}, t)$;

- изменение $e(m)$ приводит к изменению результата $G_k(m_{k0}, t)$.

Первое условие легко проверяется для всех вершин графа в ходе моделирования. Второе условие выполнено, если моделирование в G_k, начиная с вершины m, по альтернативным направлениям дает различные результаты (выход из G_k происходит в различных направлениях). Это можно представить в следующем аналитическом виде:

$$G_k(m^0, t) \oplus G_k(m^1, t) = 1, \qquad (2)$$

где m^0 и m^1 – последователи вершины m в альтернативных направлениях, т.е. $m^0, m^1 \in \Gamma_k m$.

Легко показать, что компоненты в выражении (2) можно найти по следующему выражению:

$$G_k(m,t) = G_k(m^0,t) \cdot \overline{e(m)} \vee G_k(m^1,t) \cdot e(m). \qquad (3)$$

Таким образом, $G_k(m,t)$ определяется по значениям $G_k(m',t)$, где $m' \in \Gamma_k m$. Следовательно, имеется возможность вычислить значения $G_k(m,t)$ по выражению (3) путем последовательного прохода вершин АГ от некоторого m, $\Gamma_k m = \phi$ в направлении к вершине m_{k0}, $\Gamma_k^{-1} m_{k0} = \phi$.

Так как в выражениях (2) и (3) отсутствуют булевы операции, требующие переноса значений переменных, то возможно проведение параллельного анализа для нескольких тестов. Более подробно этот вопрос рассмотрен в работе [7].

Анализ тестов в системе АГ

Рассмотрим теперь определение вершин $m \in M$, существенных относительно выходных переменных $z_k \in Z_{вых}$. Разбиваем множество графов G в два подмножества соответственно выходных графов $G_{вых} = \{G_k\}$, $V_G(G_k) \in Z_{вых}$ и внутренних графов

$G_{вн} = \{G_l\}$, $V_G(G_l) \notin Z_{вых}$.

Определение 2. Граф $G_l \in G_{вн}$ является для заданного теста существенным, если изменение значения переменной $z_l = V_G(G_l)$ приводит к изменению значения некоторой выходной переменной $z_k \in Z_{вых}$.

Обозначим существенность G_l относительно $z_k \in Z_{вых}$ через c_{lk}.

Сперва рассмотрим анализ тестов для комбинационных ЦС. Так как вершины $m \in M_c^k \cap M_k$, $G_k \in G_{вых}$ являются существенными и для системы АГ, т.е. для заданной ЦС, внимание уделяется организации анализа тестов во внутренних графах $G_l \in G_{вн}$.

Нетрудно заметить, что задача определения существенности графов $G_l \in G_{вн}$ соответствует задаче анализа многомерных путей (сходящихся разветвлений) в схеме от узлов l: $z_l = V_G(G_l)$.

При реализации реверсивного метода анализа тестов [3,4] предлагался анализ многомерных путей при помощи дополнения к структурной модели ЦС. При решении этой же задачи на модели АГ многомерные пути предлагается анализировать путем повторного моделирования (вводится измененное значение

переменной $z_l = V_G(G_l)$ в виде $dz_l = 1$ и вычисляется значение переменной $z_k \in Z_{вых}$). Сравнением полученного результата с эталонным определяется c_{lk}.

Такой подход обосновывается по следующим причинам:

– модель АГ имеет большую компактность по сравнению со структурной моделью;

– моделирование на АГ достаточно эффективно.

Согласно изложенному граф $G_l \in G_{вн}$ является существенным ($c_{lk} = 1$), если выполнено два условия:

– $m \in V_M^{-1}(V_G(G_l))$, $G_l \in G_{вн}$ находится на пути оператора $G_k(m_{k0}, t)$, $G_k \in G_{вых}$;

– изменение значения z_l приводит к изменению результата $G_k(m_{k0}, t)$.

Второе условие проверяется повторным моделированием при $dz_l = 1$.

Следовательно, повторное моделирование требуется, если выполнено первое условие. Обозначим через $L_k(m,t)$ признак нахождения вершины m на пути $G_k(m_{k0}, t)$. Тогда это условие выражается в виде

$$L_k(m,t) = 1, \quad m \in V_M^{-1}(V_G(G_l)). \qquad (4)$$

Для определения c_{lk} получим

$$c_{lk} = L_k(m,t) \cdot (G_k(m_{k0}, t) \oplus G_k(m_{k0}, t)\big|_{dz_l=1}), \qquad (5)$$

где $m \in V_M^{-1}(V_G(G_l))$.

Очевидно, существует несколько соответствующих вершин m. Но легко убедиться, что для вычисления по (5) достаточно выполнения условия (4) хотя бы для одной из них. Кроме того, в общем случае оператором $G_k(m_{k0}, t)$ определяются различные графы $G_h = V_G^{-1}(V_M(m))$, $L_k(m,t) = 1$, которые требуется зафиксировать для вычисления c_{hk} по (5) в случае

$dz_h = 1$. Тогда повторное моделирование оператором $G_k(m_{k0}, t)$ проводится для всех $dz_h = 1$.

В общем случае многомерный путь, начинающийся от некоторого z_{l_1} , может пройти через несколько графов $G_{l_1}, G_{l_2}, \ldots, G_{l_n}$, где $G_{l_j} \in G_{вн}$, $j = \overline{1,n}$. Тогда выражение (5) получает следующий обобщенный вид:

$$c_{l_1 l_n} = \prod_{j=1}^{n} L_{l_j}(m_j, t) \cdot (G_{l_j}(m_{l_j 0}, t) \oplus G_{l_j}(m_{l_j 0}, t)\Big|_{dz_{l_1}=1}), \quad (6)$$

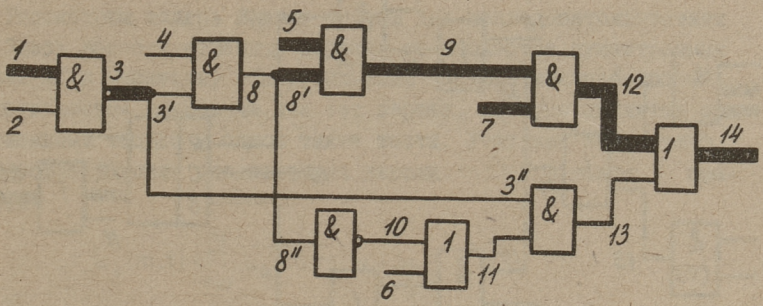
где $m_j \in M_{l_j} \cap V_M^{-1}(V_G(G_{l_{j-1}}))$.

Вычисление по (6) можно провести путем последовательного определения компонентов $c_{l_j l_{j+1}}$, начиная от $c_{l_1 l_2}$. Из (6) вытекает, что вычисления можно прекратить при получении первого $c_{l_j l_{j+1}} = 0$.

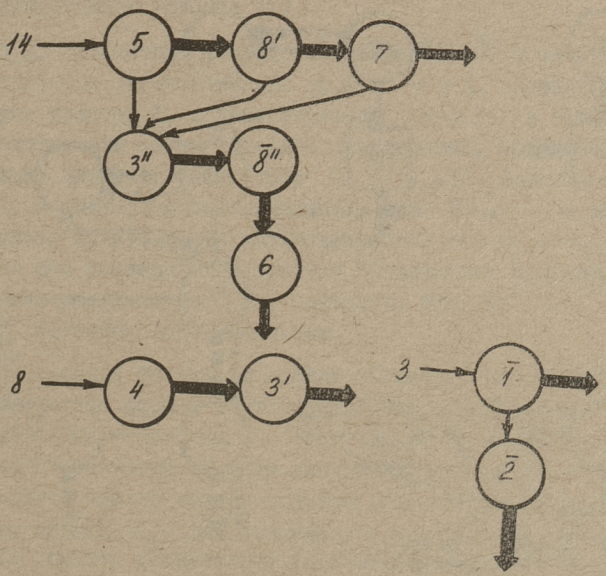
В работах [3,4] рассмотрен случай прекращения активизированности на ветвях многомерных путей. Чтобы учесть это на модели АГ, моделирование необходимо провести также в тех графах $G_h = V_G^{-1}(V_M(m))$, $L_k(m,t) = 1$, для которых по (5) получен $c_{hk} = 0$. Следовательно, применение операторов $G_{l_j}(m_{l_j 0}, t)$ требуется для решения двух задач:

- для фиксирования графов $G_{l_{j-1}} = V_G^{-1}(V_M(m)), L_{l_j}(m,t) = 1$;
- для определения вершин $m \in M_{l_j} \cap M_c$ в случае $c_{l_j k} = 1$.

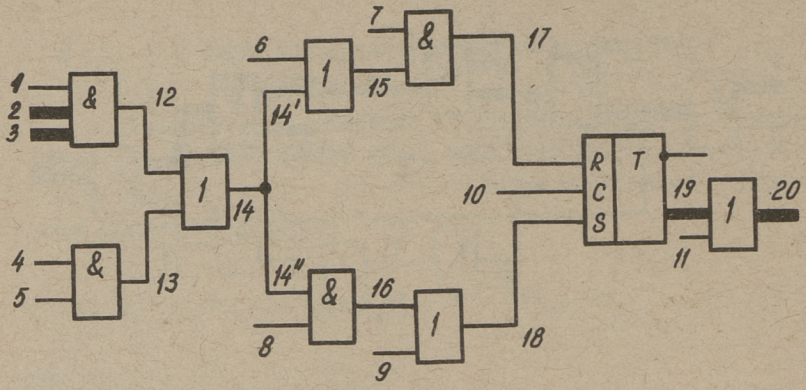
Пример I. Рассмотрим комбинационную ЦС и ее модель АГ, представленные соответственно на фиг. I и 2. Для простоты индексы переменных совпадают с номерами вершин АГ. Значения весовых функций $e(m)$ при заданном тесте представлены жирными стрелками на фиг.2 (направление направо соответствует $e(m) = 1$, а вниз – $e(m) = 0$), а активизированные пути, определенные в результате анализа, представлены жирными линиями на фиг. I. В графе $G_{14} \in G_{вых}$ фиксируется существенность для m_5, m_8 и m_7, находящихся на пути $G_{14}(m_{14,0})$. Повторное моделирование при $dz_8 = 1$ дает $c_{8,14} = 0$, однако применением $G_8(m_{8,0})$ фиксируется G_3 . Повторное моделирование при $dz_3 = 1$ дает результат $c_{3,14} = 1$, а в итоге анализа графа G_3 устанавливается $m_1 \in M_c$.



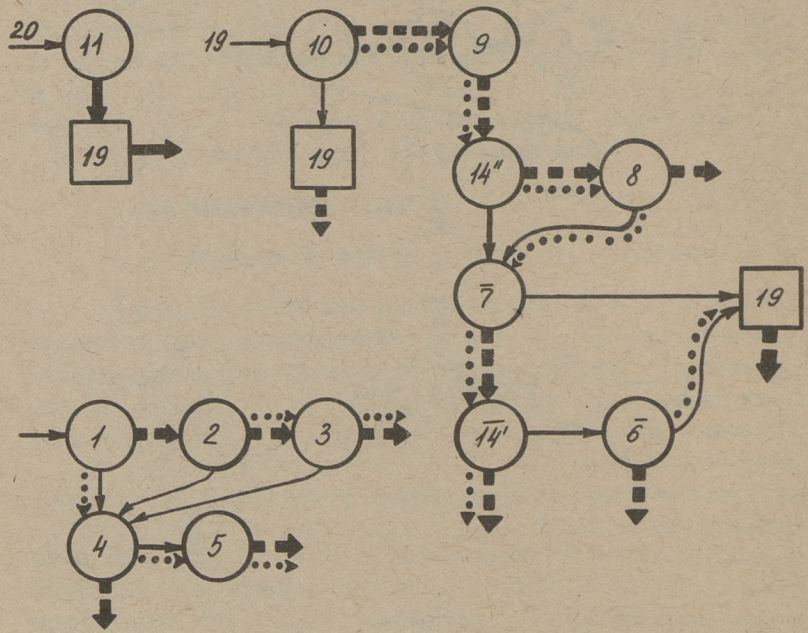
Фиг. 1.



Фиг. 2.


1
2
3
&
12
1
14
4
5
&
13
6
1
15
14'
7
&
17
14"
&
16
8
1
18
9
10
R
C
S
T
19
1
20
11

Фиг. 3.


20
11
19
19
10
19
9
14"
8
7
19
1
2
3
14'
6
4
5

Фиг. 4.

Рассмотрим теперь анализ тестов для последовательностных ЦС. Здесь непосредственный переход от множества существенных вершин АГ к множеству обнаруживаемых тестами неисправностей представляется возможным для тех вершин, которые являются существенными только в одном такте теста. Учитывая это, можно в качестве условия рассмотрения вершин $m \in M$ в такте t принимать

$$L_k(m,t) \cdot \prod_{r=1}^{n_t} \overline{L_k(m,r)} = 1, \qquad r \neq t, \tag{7}$$

где n_t – длина теста.

Анализ тестов в системе АГ, описывающей последовательностную ЦС, управляется операторами $G_k(m_{k0},t)$, $G_k \in G_{вых}$. При этом в случае $\tau(m) \cdot L_h(m,t) = 1$, $t = \overline{1,n_t}$ анализ продолжается в такте $t-1$. Процесс анализа аналогичен обработке тестов комбинационных ЦС, за исключением использования дополнительного условия (7).

Пример 2. Рассмотрим последовательностную ЦС и ее модель АГ, представленные соответственно на фиг. 3 и 4. Обозначения соответствуют примеру I, причем значения $e(m)$ в тактах I,2 и 3 представлены соответственно пунктирными, прерывистыми и сплошными линиями. Оператором $G_{19}(m_{19,0},2)_{(2)}$ фиксируется граф G_{14}, а повторным моделированием при $dz_{14}=1$ и последующим анализом в этом графе устанавливается $m_2 \in M_c$ и $m_3 \in M_c$. Заметно, что при последовательностных ЦС возможно получение неопределенных результатов. Так, например, при $dz_{14}^{(1)}=1$ значение $z_{19}^{(2)}$ остается неопределенным.

Процесс анализа тестов на модели АГ может быть организован в три этапа. На первом этапе проводится моделирование тестов операторами $G_k(m_{k0},t)$, $G_k \in G_{вых}$ с целью получения эталонного состояния объекта диагностирования. На втором этапе проводится анализ тестов с целью определения множества существенных вершин M_c. Для повышения эффективности анализа предлагается заранее осуществить локальный анализ для всех АГ в отдельности, а затем анализ относительно выходных переменных $z_k \in Z_{вых}$. При этом анализ в системе АГ представляет собой реверсивное прослеживание АГ в направлении от графов $G_k \in G_{вых}$ к графам $G_h \in G_{вн}$, $\forall m \in M_h : z(m) \in Z_{вх}$.

Согласно вышеизложенному этот процесс сопровождается повторным моделированием с целью определения существенных графов. На третьем этапе множество существенных вершин M_c превращается во множество обнаруживаемых тестами неисправностей из заданного класса.

Литература

I. Chang H., Chappell S.G., Elmendorf C.H., Schmidt L.D. Comparison of parallel and deductive fault simulation methods. - IEEE Trans. on Comp., 1974, vol. C-23, N 11, p. 1132-1138.

2. Armstrong D.B. A deductive method for simulating faults in logic circuits. - IEEE Trans. on Comp., 1972, vol. C-21, N 5, p. 464-472.

3. Убар Р.Р. Анализ диагностических тестов для комбинационных цифровых схем методом обратного прослеживания неисправностей. - Автоматика и телемеханика, 1977, № 8, с. 168-176.

4. Китсник П.А., Убар Р.Р. Формулы для дедуктивного анализа тестов в синхронных последовательностных схемах. - Тр. Таллинск. политехн. ин-та, 1977, № 432, с. 15-23.

5. Убар Р.Р. Описание цифровых устройств моделью альтернативных графов. - Тр. Таллинск. политехн. ин-та, 1979, № 474, с. 11-33.

6. Убар Р.Р. Тестовая диагностика цифровых устройств, ч. I. Таллинский политехнический институт, 1980.

7. Убар Р.Р. Тестовая диагностика цифровых устройств, ч. II. Таллинский политехнический институт, 1981.

P. Kitsnik

Test Completeness Analysis for Digital Circuits Using Alternative Graph Model

Summary

In this paper a method for finding the set of detectable faults of a digital circuit for a given set of tests is described. This method is based on test reversive analysis and uses the alternative graph model of the circuit. The main properties of this method are discussed.

№ 530

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

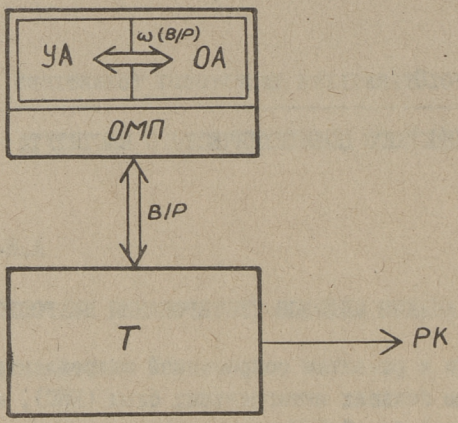
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.32

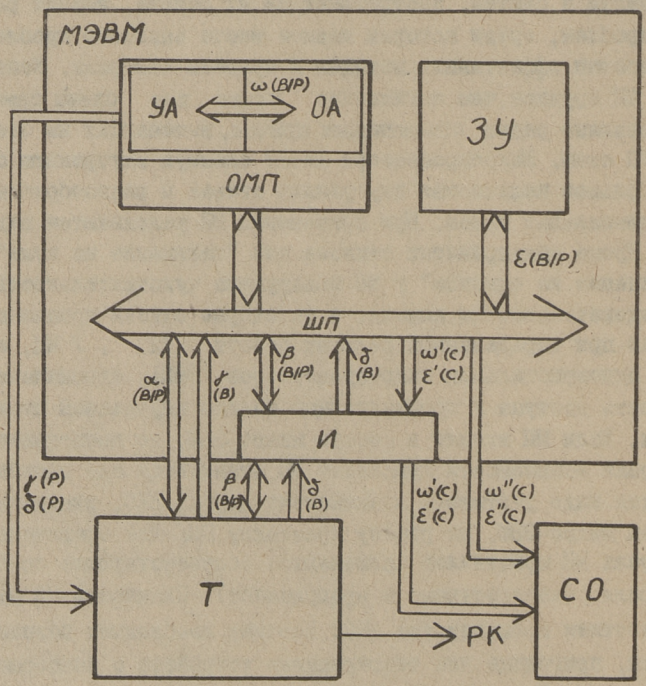
А.К-Ф. Тоомсалу

АНАЛИЗ МЕТОДОВ ТЕСТИРОВАНИЯ МИКРОПРОЦЕССОРОВ

Успехи в развитии современной микроэлектроники привели к появлению больших интегральных схем (БИС), которые являются элементной базой ЭВМ четвертого поколения. Центральное место среди БИС имеют микропроцессорные (МП) схемы и схемы памяти. В связи с широким применением микропроцессорных устройств и систем, построенных на их основе, возник ряд сложных проблем, среди которых важное место занимает проблема разработки эффективных методов и средств контроля. Контролировать МП труднее, чем логические платы или схемы памяти. Это вызвано рядом объективных причин, вытекающих из особенностей МП схем. Микропроцессоры имеют сложную внутреннюю структуру, большое количество внутренних связей и разнообразных функциональных узлов. При применении МП усложняются виды отказов. Кроме общепринятых отказов как "залипание на нуле" или "залипание на единице" у МП обнаружена чувствительность к последовательностям команд, а некоторые отказы проявляются только при определенных режимах работы [I2] . У МП, особенно у однокристальных микропроцессоров (ОМП), ограничена возможность доступа к промежуточным точкам внутренней логической схемы. Если МП входит в состав микро-ЭВМ, то проведение контроля исправности кристалла МП становится еще сложнее, так как надо учитывать воздействия других БИС, входящих в состав микро-ЭВМ, на работу кристалла МП. Для проведения контроля МП кристаллов применяются соответствующие средства контроля (аппаратурные и программные). Сложность МП растет с развитием производства БИС. Поэтому необходимо разработать методы, пригодные для тестирования устройств с возросшей сложностью. Процесс контроля МП состоит из нескольких этапов, на каждом из которых применяется соответствующая методика и



Фиг. 1. Контроль ООМП.



Фиг. 2. Контроль ОМПП.

средства. При выборе метода контроля МП необходимо знать объект контроля, отрицательные и положительные стороны и сферу применения каждого метода.

Объект контроля

В качестве объекта контроля рассматривается отдельная однокристальная микропроцессорная интегральная схема (ООМП) и однокристальная микропроцессорная интегральная микросхема, входящая в состав одноплатной микро-ЭВМ (ОМПП).

Средства контроля

Для контроля исправности ОМП можно использовать средства самоконтроля ОМП или средства внешнего контроля. В качестве средств внешнего контроля могут быть использованы различные тестеры.

Доступ к объекту контроля

В зависимости от объекта контроля возможен или непосредственный доступ к выводам ОМП, или косвенный через разъем печатной платы микро-ЭВМ. На фиг. I и 2 приведены варианты систем контроля исправности ООМП и ОМПП. Приняты следующие условные обозначения:

ОМП - однокристальная микропроцессорная интегральная схема;

УА - управляющий автомат ОМП;

ОА - обрабатывающий автомат ОМП;

Т - тестер;

МЭВМ - одноплатная микро-ЭВМ;

ЗУ - запоминающее устройство;

И - интерфейс МЭВМ;

ШП - шина ОМП (в состав ШП входят линии для передачи информации, адресов и сигналов управления);

СО – средства отображения результатов самоконтроля;

В – входные тестовые наборы;

Р – выходные тестовые наборы;

РК – результаты контроля;

С – сигналы отображения результатов самоконтроля;

α – непосредственный доступ к шинам ОМП;

β – доступ к шинам ОМП через интерфейс МЭВМ;

γ – непосредственный доступ к шинам ОМП с использованием управляемого пробника;

δ – доступ к шинам ОМП через интерфейс МЭВМ с использованием управляемого пробника;

ω – самоконтроль ОМП (микропрограммный или с помощью внутрисхемных аппаратурных средств самоконтроля);

ε – самоконтроль ОМП (программный).

Методы получения входных тестовых наборов

Наиболее распространенные методы получения входных тестовых наборов можно разделить на две группы:

– генерация входных воздействий;

– применение хранимых входных тестовых наборов [3,4,7].

Генерация входных воздействий может быть осуществлена двумя способами: алгоритмической генерацией тестов [1,2,3,4,7,8] или генерацией псевдослучайных тестовых кодов [3].

При алгоритмическом методе генерации входных наборов, контролируемому ОМП, передаются последовательности входных наборов по специальным формулам или алгоритмам. Для формирования этих наборов применяются быстродействующие генераторы, которые формируют последовательности наборов в соответствии с заданным алгоритмом. Сам генератор реализуется схемно и управляется микропрограммно. Метод позволяет уменьшить объем

памяти системы контроля, обеспечивает гибкость при генерации различных последовательностей команд, требует меньшего объема ручного программирования, чем аналогичная программа, составленная вручную [3] . Метод особенно эффективен для тестирования памяти [9] . Недостатками являются сложность программирования и сфера узкого применения [9] .

При генерации псевдослучайных входных тестовых наборов базируются на предпосылке, что большое количество повторяемых последовательностей испытательных кодов, которые генерируются по статически случайному закону, способны вызывать на выходах ОМП сигналы, характерные неисправностям. Естественно, псевдослучайным тестовым наборам должны предшествовать специальные тесты для запуска и синхронизации ОМП. Аппаратура для данного метода не сложна и не дорога. Затраты на математическое обеспечение не велики. Недостатками метода является возможность подать "запрещенные" входные наборы или последовательности, которые приводят контролируемое устройство в такое состояние, что оно блокируется от последующего тестового кода. Для выявления неисправности потребуется проведение большего количества испытаний связи, что чрезмерно увеличивает время контроля [3] . При контроле с помощью хранимых входных наборов фиксируются значения входных наборов, а также ожидаемые выходные наборы в запоминающем устройстве. При проведении контроля из запоминающего устройства извлекаются эти наборы и подаются на испытываемый прибор. Метод эффективен для проверки постоянных запоминающих устройств малой и средней степени интеграции [9]. Контроль БИС и ОМП требует больших объемов памяти для хранения входных наборов. Для уменьшения объема памяти применяются сложные программные и аппаратные средства, связанные с организацией циклов, полуалгоритмическими подпрограммами и т.д. [9] .

Методы получения выходных тестовых наборов

Для получения выходных тестовых наборов применяются следующие методы:

- генерация выходных тестовых наборов. При генерации наборов может быть применена эмуляция [1,2,6] или алгоритмическая генерация [1,6,9] ;

- применение хранимой характеристики [3,4,7,9] ;

- сравнение с эталонным прибором [I,2,7,9] .

При эмуляции используется модель ОМП, которая представляется в виде программы. Для эмуляции применяется быстродействующая ЭВМ, в которой работает эмулирующая программа. Данная программа моделирует процессы, происходящие в МП при выполнении тестовых последовательностей. В результате работы программы получаются значения эталонных выходных тестовых наборов. Обычно данный метод используется совместно с алгоритмическим методом генерации входных воздействий [3] . Качество проверки зависит в большой мере от полноты и точности моделирования процессов, происходящих в микропроцессоре. Так как скорость эмуляции должна быть не меньше быстродействия испытываемого МП, то требуется быстродействующая ЭВМ, на которой работает эмулирующая программа. Недостатками метода являются невозможность программного моделирования временных особенностей обработки информации, присущих физическим образцам, и низкое быстродействие [6] .

Алгоритмическая генерация выходных тестовых кодов применяется при блочном подходе к построению тестов. Для получения выходных тестовых наборов реализуется алгоритм функционирования контролируемого блока ММП с помощью схемных средств. При данном способе увеличиваются затраты на создание тестового оборудования, но отпадает необходимость в хранении записанных характеристик и использования быстродействующей и объемной памяти [6] .

При применении хранимых характеристик, в качестве выходных тестовых кодов, значения ожидаемых реакций ОМП для заданных входных тестовых кодов записываются в память тестера. Значения этих кодов могут быть получены от заведомо исправного МП или определяются ручным анализом, или с помощью математического моделирования.Метод имеет те положительные и отрицательные стороны, которые отмечались при соответствующем способе получения входных тестовых наборов. Если для получения выходных тестовых кодов применяется метод сравнения с эталонным прибором, то на эталонный (заведомо исправный) МП и испытываемый МП подаются одни и те же

входные тестовые коды и сравниваются значения выходных сигналов обоих МП. Расхождение выходных сигналов указывает на наличие неисправности. По сравнению с другими методами данный метод относительно прост для реализации, но требует выбора эталонного МП и гарантии его исправной работы. Данный метод используется часто совместно с методом псевдослучайных входных воздействий и с ручным методом формирования входных тестовых наборов [3] .

Отображение выходных тестовых наборов

Наборы логических состояний, фиксируемых на выходных контактах испытываемого МП, в течение определенного периода времени могут быть представлены в виде машинных слов или векторами двоичных сигналов, число элементов которых соответствует числу контролируемых точек схемы. Кроме указанных способов представления выходных тестовых наборов применяется сигнатурный способ отображения [5,II] . Сигнатура является сжатым представлением логических сигналов, присутствующих в некотором узле испытываемой схемы, или она формируется на основе выходного тестового вектора по определенному закону. Для формирования сигнатур применяется соответствующая аппаратура.

Виды контроля

Для проведения контрольных испытаний в настоящее время наиболее распространены следующие виды контроля:

- статический контроль МП;
- динамический контроль МП;
- параметрический контроль МП.

Для повышения эффективности контроля ОМП желательно провести испытания всесторонне, для чего не ограничиваться только одним видом контроля, а применять различные виды контроля на одном объекте контроля. Последние исследования в области контроля МП показывают, что для контроля ОМПП перспективным является применение возможностей самоконтроля [I3] . Для проведения самоконтроля используются как аппара-

турные, так и программные средства [3,7,10,11] . Из программных средств используются тест-программы, которые находятся в самом ОМП (микропрограммный контроль) или в запоминающем устройстве микро-ЭВМ (программный контроль). Из аппаратурных средств применяются внутри ОМП специальные схемы комплементарной логики [10] , самопроверяющие узлы и дополнительные микросхемы контроля, которые соединяются с проверяемым ОМП. При применении самоконтроля может часть неисправностей остаться необнаруженной из-за недостаточности внутренней тест-программы или аппаратуры.

Важным направлением контроля ОМП является создание машинных методов и программ синтеза тестов, позволяющих частично или полностью автоматизировать этот процесс. Усложнение архитектуры МП, появление специальных схем ввода-вывода в составе микропроцессорного кристалла, например, каналы, цифроаналоговые и аналого-цифровые преобразователи информации, ставят новые требования к методике и аппаратуре применяемой для тестирования.

Литература

I. Chiang A.C.L. Test schemes for microprocessor chips. - Computer Design, April 1975, p. 87-92.

2. Hackmeister D., Chiang A.C.L. Microprocessor test technique reveals instruction pattern sensitivity. - Computer Design, December 1975, p. 81-85.

3. Андерсон. Методы контроля микропроцессорных устройств. - Электроника, 1976, № 8, с. 64-70.

4. Чанг, Маккасил. Два новых способа упрощения тестовой проверки микропроцессоров. - Электроника, 1976, № 2, с. 45-52.

5. Nadig H.J. Signature analysis - concepts, examples and guidelines. - Hewlett-Packard Journal, May 1977, p. 15-21.

6. Гобземис А.Ю., Удалов В.И. Методы тестового контроля микропроцессорных устройств. - Автоматика и вычислительная техника, 1978, № 6, с. 18-27.

7. Jefferies K. Microprocessor testing techniques. - Electronic Engineering, January 1978, p. 61-62.

8. Crichton G. Testing microprocessors. - IEEE Journal of Solid-State Circuits, June 1979, vol. SC-14, N 3, p. 603-613.

9. Bluestone A. Logical environment comparison testing handles complex LSI devices. - Computer Design, April 1979, p. 95-102.

10. Bennets R.G. Fault tolerance and digital systems. - Microprocessors and Microsystems, October 1979, vol. 3, N 8, p. 365-373.

II. L o w e L. Designing for testability. - Microprocessors and microsystems, Jan./Feb. 1979, vol. 3, N 1, p. 3-6.

I2. B r a d l e y R.S. A three stage approach to LSI board testing. Part 1. - Electronic Engineering, April 1981, p. 83-91.

I3. G a l l a c h e r J. The test engineer's nightmare. - Microprocessors and Microsystems, September 1981, vol. 5, N 7, p. 291-293.

A. Toomsalu

Analysis of Test Methods for Microprocessors

Summary

The test methods for one-chip microprocessors and for one-chip microprocessors in a singleboard microcomputers are analysed in this paper. The techniques for input and output pattern generation are described. Advantages and disadvantages of these methods are observed.

№ 530

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.32

А.К.-Ф. Тоомсалу,
Р.Р. Убар

ГЕНЕРИРОВАНИЕ ОПЕРАНДОВ ПРИ СИНТЕЗЕ ТЕСТОВ ДЛЯ МИКРОПРОЦЕССОРОВ

Рассмотрим некоторые возможности формализации процесса синтеза тестов для проверки работоспособности микропроцессоров, пользуясь при этом моделью векторных альтернативных графов.

Одним из подходов к тестовой проверке микропроцессоров является модульное тестирование. Любой микропроцессор можно рассматривать как набор независимых функционально законченных подсистем, или модулей, каждый из которых можно проверять отдельно. Шины адресов и данных связывают такие модули друг с другом и обеспечивают доступ извне к любому из них.

Первым подготовительным этапом тестовой проверки микропроцессора является декомпозиция его на функциональные модули. Такими могут быть все программно доступные операционные элементы (регистры, триггеры и т.д.), функции которых определяются системой команд микропроцессора и могут быть описаны моделью ВАГ [1-2] . К каждому операционному элементу должен быть обеспечен доступ через шину ввода-вывода путём выполнения соответствующей группы команд микропроцессора, определяемых формально на модели ВАГ. Другими словами, каждый операционный элемент должен иметь возможность прямо или косвенно выдавать свои выходные данные на шину, чтобы их можно было принимать и оценивать извне путём выполнения определённой группы последовательных команд собственного языка микропроцессора.

Задачу контроля микропроцессора можно разбить на две части. В первую часть входит контроль функционирования

операционных элементов (шин,регистров), во вторую - контроль устройства управления операциями.

При решении первой части контроля можно базироваться на традиционной модели структурных неисправностей (логические константы на шинах и в регистрах, короткие замыкания и т.д.). Для проверки этих неисправностей могут быть использованы команды типа передачи информации между регистрами и вводом-выводом. Такие команды можно формально определить на модели ВАГ.

На втором этапе, при проверке устройства управления, предположим, что операционная часть микропроцессора проверена и функционирует правильно. Такое предположение позволяет базироваться на поразрядной эталонной информации при контроле выполнения команд, а следовательно, относительно просто определить необходимые операнды при генерировании тестов. Синтез таких операндов может быть проведен формально на модели ВАГ.

Рассмотрим функции любой команды I_l микропроцессора в следующем общем виде:

$$S_k = f_{l,k}(S_i, S_j), \quad S_k \in S(I_l) \subseteq S, \qquad (I)$$

где S - множество всех программно доступных элементов микропроцессора;

$S(I_l)$ - подмножество элементов, состояние которых может быть изменено в результате выполнения команды I_l;

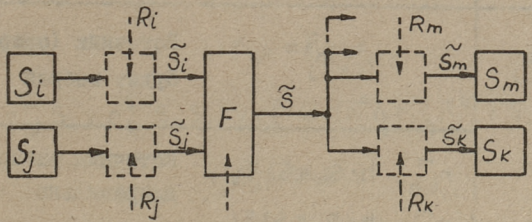
S_k - элемент, в котором фиксируется результат (один из результатов) выполнения команды I_l;

$f_{l,k}$ - функция, которая реализуется при выполнении команды I_l в элементе S_k;

S_i, S_j - элементы, в которых хранятся операнды для операции $f_{l,k}$ до выполнения команды I_l.

В частных случаях число аргументов функций $f_{l,k}$, а также число самих функций $f_{l,k}$ при выполнении команды I_l могут быть равными единице.

Пусть $|S(I_l)| = I$. Тогда множество неисправностей, влияющих на выполнение команды I_l, определим в следующем виде (см. фиг. I):



Фиг. 1.

$$R = R_i \cup R_j \cup R_f \cup R_k \cup R_m ,$$

где R_i, R_j – подмножества неисправностей, действующих на операнды операции f_l (ошибки управления мультиплексором);

R_f – подмножество неисправностей, действующих на операцию f_l (ошибки управления АЛУ);

R_k, R_m – подмножества неисправностей, связанных с ложным фиксированием результата операции f_l (ошибки управления демультиплексором).

Подробная интерпретация рассматриваемых неисправностей приведена в табл. I.

Базируясь на приведенной в табл. I модели неисправностей управляющего устройства микропроцессора, получим вместо функции (I) следующие обобщенные функции:

Таблица I

Класс неисправности	Обозначение неисправности	Физическое содержание неисправности
R_i, R_j	r_i^o, r_j^o	Источник (операнд) не будет выбран
	$r_{i,h}^m$, $h: S_h \in S \setminus S_i$ $r_{j,h}^m$, $h: S_h \in S \setminus S_j$	Источник выбирается неправильно
	$r_{i,h}^b$, $h: S_h \in S \setminus S_i$ $r_{j,h}^b$, $h: S_h \in S \setminus S_j$	Выбирается одновременно два источника
R_f	$r_{l,h}$	Вместо f_l выполняется неправильная операция f_h
R_k	r_k	Результат не будет зафиксирован (состояние элемента $S_k \in S(I_l)$ не изменяется)
R_m	$r_m, m: S_m \in S \setminus S_k$	Результат передаётся по неправильному адресу

$$\tilde{S}_k = r_k S_k \vee \bar{r}_k \hat{S} ;$$

$$\tilde{S}_m = r_m \tilde{S} \vee \bar{r}_m S_m , \; m : S_m \in S \setminus S_k ;$$

$$\tilde{S} = \bigvee_h r_{l,h} f_h(\tilde{S}_i, \tilde{S}_j) \vee f_l(\tilde{S}_i, \tilde{S}_j) \bigwedge_h \bar{r}_{l,h} ,$$

$$h : r_{l,h} \in R_p ; \qquad (2)$$

$$\tilde{S}_i = \bigvee_h (r^m_{i,h} S_h \vee r^b_{i,h} S_h S_i) \vee r^o_i \cdot 0 \vee (\bar{r}^o_i \bigwedge_h \bar{r}^m_{i,h} \bar{r}^b_{i,h}) S_i ,$$

$$h : S_h \in S \setminus S_i ;$$

$$\tilde{S}_j = \bigvee_h (r^m_{j,h} S_h \vee r^b_{j,h} S_h S_j) \vee r^o_j \cdot 0 \vee (r^o_j \bigwedge_h \bar{r}^m_{j,h} \bar{r}^b_{j,h}) S_j ,$$

$$h : S_h \in S \setminus S_j .$$

Дифференцированием обобщенных функций (2) по переменным $r \in R$ получим дополнительные условия $W(r)$, выполнение которых необходимо для обнаружения соответствующих неисправностей $r \in R$.

В таблице 2 приведены соответствующие условия для случая одиночных неисправностей. При этом f_l обозначает результат операции $f_l(S_i, S_j)$ при выполнении команды I_l.

Таблица 2

r	$W(r)$
r_k	$f_l \oplus S_k = 1$
r_m	$f_l \oplus S_m = 1$
$r_{l,h}$	$f_l \oplus f_h = 1$
r_i^o, r_j^o	$S_i \frac{\partial f_l}{\partial S_i} = 1$, $S_j \frac{\partial f_l}{\partial S_j} = 1$
$r_{i,h}^m$, $r_{j,h}^m$	$(S_i \oplus S_h) \frac{\partial f_l}{\partial S_i} = 1$, $(S_j \oplus S_h) \frac{\partial f_l}{\partial S_j} = 1$
$r_{i,h}^b$, $r_{j,h}^b$	$S_i \bar{S}_h \frac{\partial f_l}{\partial S_i} = 1$, $S_j \bar{S}_h \frac{\partial f_l}{\partial S_j} = 1$

Исходя из данных таблицы 2 получим следующую систему условий :

$$\bar{S}_h S_i [f_l(S_i=1) \oplus f_l(S_i=0)] = 1, \quad h: S_h \in S \setminus S_i; \qquad (3)$$

$$\bar{S}_h S_j [f_l(S_j=1) \oplus f_l(S_j=0)] = 1, \quad h: S_h \in S \setminus S_j; \qquad (4)$$

$$f_l \oplus f_h = 1,\ h : r_{l,h} \in R_f ; \qquad (5)$$

$$f_l \oplus S_m = 1,\ m : S_m \in S \setminus S_k ; \qquad (6)$$

$$f_l \oplus S_k = 1, \qquad (7)$$

выполнение которой требуется при проверке команды I_l. Операнды, используемые при тестировании команды I_l, должны быть выбраны так, чтобы условия (3) – (7) оказались выполненными. При этом для каждого условия (3) – (7) достаточно выбрать лишь один разряд, где оно будет выполнено.

Генерирование операндов начинается с младших разрядов. Значения операндов в выбранном разряде находятся путем активизации соответствующих путей на модели АГ. Направление выхода из графа для этих путей задано рассматриваемым условием. При переходе к следующему условию проверяется его выполнение для ранее зафиксированных разрядов. Если оно не выполняется ни в одном из этих разрядов, выбирается следующий свободный старший разряд и генерируются значения выбранного разряда, гарантирующие выполнение рассматриваемого условия.

Пример I. Рассмотрим фрагмент системы команд некоторого гипотетического микропроцессора с форматом слова $S(1:n)$, где каждая команда вызывает изменение состояния элемента S_1 (в частности, аккумулятора):

I_1 : MOV S_1, A	$S_1 \leftarrow IN(A)$,
I_2 : MOV S_1, S_2	$S_1 \leftarrow S_2$,
I_3 : MOV S_1, S_3	$S_1 \leftarrow S_3$,
I_4 : ADD S_2	$S_1 \leftarrow S_1 + S_2$,
I_5 : SUB S_2	$S_1 \leftarrow S_1 - S_2$,
I_6 : ANA S_2	$S_1 \leftarrow S_1 \wedge S_2$,
I_7 : ORA S_2	$S_1 \leftarrow S_1 \vee S_2$,

I_8 : XRA S_2 $\qquad$ $S_1 \leftarrow S_1 \oplus S_2$,

I_9 : CMA $\qquad$ $S_1 \leftarrow \neg S_1$,

I_{10} : POP $\qquad$ $S_1 \leftarrow$ IN(SP).

Примененные здесь мнемокоды взяты из системы команд микропроцессора INTEL 8080. При этом использованы следующие обозначения:

S_1 – аккумулятор;

$S_2 S_3$ – регистры общего назначения;

SP – регистр-указатель стека;

IN(A) – ввод информации по адресу A, указанному в команде;

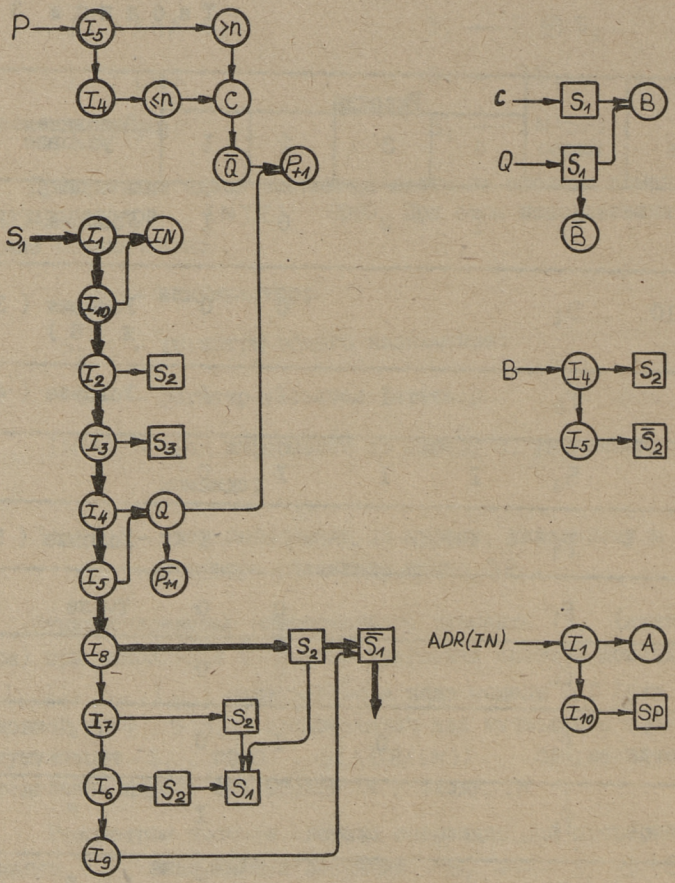
IN(SP) – ввод информации по адресу, указанному в регистре-указателе стека SP .

Функционирование операционного элемента S_1 (аккумулятора) при выполнении заданного фрагмента системы команд ($I_1, I_2, \ldots, I_{10}$) описывается в виде модели ВАГ (фиг. 2). Основной граф $G(S_1)$ предназначен для вычисления состояния аккумулятора S_1 , граф G(ADR(IN)) предназначен для определения адреса при обращении к входу IN .

Рассмотрим процесс синтеза операндов для проверки команды I_8 , выполняющей операцию $S_1 \leftarrow S_1 \oplus S_2$. Шаги процесса при обработке условий иллюстрированы в табл.3.

Таблица 3

№ шага	Слово	Разряды				Выполняемое условие
		4	3	2	I	
I	S_1	I	0	0	I	Условие (3)
I, I0	S_2			0	0	Условия (3) и (6)
2	f_8	0	I	I	I	Условие (4)
	S_3	I	I	I	0	
3	f_2				I	Условие (5)
4	f_3			0	0	То же
5	f_4			0	0	"
6	f_5		0	I	0	"
7	f_6				I	"
8	f_7				I	"
9	f_9	0	I	I	0	"
II	S_1				I	Условие (7)



Фиг. 2.

На первом и втором шагах определяются значения $S_1(n)$, $S_2(n)$ и $S_3(n)$, обеспечивая выполнение условий (3) и (4). На третьем шаге устанавливается, что значения разряда n удовлетворяют условию (5) для операции f_2. На четвертом, шестом и девятом шагах генерируются значения

соответственно разрядов $(n-1)$, $(n-2)$ и $(n-3)$ с целью удовлетворения условия (5) соответственно для операций f_3, f_5 и f_9. На остальных шагах введение новых разрядов не требуется.

Литература

I. Убар Р.Р. Описание ЦВМ моделью векторных альтернативных графов с целью синтеза диагностических микропрограмм. – Тр. Таллинск. политехн. ин-та, 1980, № 497, с. 11-20.

2. Ubar R.R. Vektorielle alternative Graphen und Fehlerdiagnose für digitale Systeme. – Nachrichtentechnik/ Elektronik, 1981, 31, H. 1, S. 25-29.

A. Toomsalu, R. Ubar

Erstellung von Operanden bei der Testsynthese für Mikroprozessoren

Zusammenfassung

Es werden Fragen zur Formalisierung der Testsynthese für Mikroprozessoren betrachtet. Als mathematisches Modell des Objekts werden die vektoriellen alternativen Graphen verwendet. Zwecks der Beschreibung des Objekts mit Fehlern werden verallgemeinerte Funktionen eingeführt. Die Boole-schen Ableitungen dieser Funktionen geben die Bedingungen, die bei der Testsynthese berücksichtigt werden müssen.

№ 530

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.32

К.В. Григорьева, Т.В. Лохуару,
Т.А. Эвартсон

МЕТОД АЛГОРИТМИЧЕСКОГО ГЕНЕРИРОВАНИЯ ТЕСТОВ ПРИ КОНТРОЛЕ ЦИФРОВЫХ СХЕМ

Быстрое развитие технологии производства интегральных схем очень резко изменило ситуацию в области контроля этих схем. Появились сложные элементы: микропроцессоры, блоки памяти с большим объемом, большие интегральные схемы (БИС) специального назначения и т.д. Все это привело к тому, что существующие аппаратные и программные средства контроля оказались морально устаревшими. Поэтому ведутся поиски новых подходов к генерированию тестов и проведению контрольных экспериментов.

Тесты для проверки БИС обычно очень длинны и поэтому возникают серьезные проблемы при их хранении (в случае, когда тесты будут заранее подготовлены). С другой стороны, учитывая факт, что такие тесты для контроля БИС часто генерируются по простым алгоритмам и выполняются при небольшой модификации многократно (циклически), оказывается возможным автоматическое генерирование тестов непосредственно во время тестового эксперимента. Так, например, при проверке БИС памяти, достаточно проверить все ячейки циклически на одни и те же функции - на запись и чтение нескольких операндов. При проверке микропроцессоров могут быть проведены циклически все команды с небольшим количеством разных операндов.

Алгоритмы генерирования таких тестов могут быть реализованы в тестере как программно, так и аппаратно.

Аппаратной реализации подвергаются обычно такие этапы генерирования тестов, которые не требуют сложных вычислений, но нуждаются в большой скорости. Сюда относятся, на-

пример, псевдослучайное генерирование кодов, алгоритмическое генерирование кодов по простому закону изменения (например, при сканировании всех состояний счетчиков), автономное считывание из ЗУ в быстром темпе заранее подготовленных последовательностей.

К программной реализации необходимо отнести вопросы алгоритмического генерирования тестов (или частей тестов) по алгоритмам, которые настолько специальны, что их аппаратурная реализация теряет смысл (сюда входят обычно заранее вручную подготовленные тестовые программы, имеющие циклический характер).

Недостатком алгоритмического подхода является его слишком тесная зависимость от проверяемого объекта. Для каждой проверяемой функции необходимо специально создавать или специальное аппаратурное дополнение в тестере или специальную программу на каком-то языке, используемом при данном тестере. Аппаратурные дополнения реализуются, очевидно, лишь только тогда, когда они или достаточно универсальны или их использование предсказано требуемым быстродействием тестера. Программные алгоритмы необходимо всегда программировать вручную заранее.

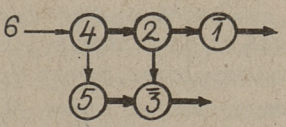
В данной статье предлагается новый подход к алгоритмическому генерированию тестов, где не только тесты, а также алгоритмы их генерирования формируются автоматически, базируясь на описании проверяемого [1] объекта моделью альтернативных графов (АГ). При этом подходе отпадает необходимость ручной подготовки тестов (алгоритмов их генерирования) и хранения их в архиве. Архив будет состоять из моделей АГ для интегральных схем (в том числе для СИС и БИС), которые описывают их функциональное поведение. Построение АГ может быть автоматизировано. Программы генерирования тестов (алгоритмов формирования входных воздействий на контролируемый объект), входящие в программное обеспечение тестера, универсальны для любого типа БИС, заданного в виде модели АГ.

Рассмотрим более подробно возможности использования модели АГ для управления диагностическими экспериментами.

Каждому пути на АГ, описывающем некоторую цифровую схе-

му, можно сопоставить определенный набор значений входных переменных схемы.

Если рассматриваемым путем определены однозначно значения не всех переменных, то ему соответствует не один, а некоторое множество наборов. Так, например, пути (4, 2, $\bar{1} \rightarrow$ I) на графе фиг. I, проходящей через вершины 4, 2 и $\bar{1}$ и выходящей из графа направо (т.е. со значением I), соответствует неполностью определенный входной набор OI x I x (I,2,3,4,5), где $X \in \{0,I\}$, который, в свою очередь, можно рассматривать как множество из четырех полностью определенных входных наборов: {OIOIO, OIOII, OIIIO, OIIII}. Набор OIXIX можно также рассматривать как один возможный куб в кубическом покрытии $C(z_6)$ функции



Фиг. 1.

$$z_6 = z_4(\bar{z}_1 z_2 \vee \bar{z}_2 \bar{z}_3) \vee z_3(z_1 z_4 \vee z_4 z_5), \qquad (I)$$

соответствующей графу на фиг. I [2]. В общем случае множество всех путей на АГ, представляющем некоторую функцию $z_k = f_k(Z_k)$, задает кубические покрытия $C(z_k)$ и $C(\bar{z}_k)$.

Итак, модель АГ можно рассматривать как сжатое представление кубических покрытий заданной функции.

Каждому пути графа можно сопоставить некоторый эксперимент над соответствующей схемой: значения весовых переменных для вершин на выбранном пути задают входной набор, а направление выхода из графа задает ожидаемый выходной сигнал объекта. Сканирование всех возможных путей вне графа соответствует генерированию некоторого множества экспериментов над объектом.

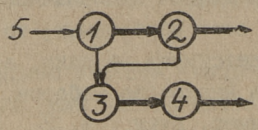
Учитывая вышеизложенное, процедуру сканирования путей на модели АГ можно рассматривать как алгоритм генерирования определенных экспериментов над объектом.

Целесообразно подчеркнуть следующие моменты:

- заданием модели АГ однозначно задано определенное множество экспериментов над объектом;

– алгоритм генерирования этих экспериментов, заключающийся в сканировании путей на модели АГ, достаточно прост и легко реализуем на ЦВМ.

Открытым, однако, остается пока вопрос, являются ли эксперименты, определяемые простым сканированием модели АГ, тестами, и, если да, то какой будет полнота этих тестов.



Фиг. 2.

Легко убедиться, что не каждый такой эксперимент является тестом. Так, например, эксперимент IIXX,I (z_1 z_2 z_3 z_4,z_5) соответствующий пути (I/I, 2/I)/I на графе фиг. 2 не обязательно является тестом. Если $z_3 = I$ и $z_4 = I$ (см. жирные стрелки на графе фиг. 2), то данный входной набор не чувствителен к неисправностям (точнее, одиночным неисправностям). С другой стороны, если зафиксировать дополнительно значение $z_3 = 0$, то данный эксперимент является тестом для неисправностей $z_1 = 0$ и $z_2 = 0$.

Каждый эксперимент является предпосылкой получения теста для некоторых вершин – для вершин, через которые проходит активизированный этим экспериментом путь на графе.

Пусть активизирован некоторый путь l^{α}, выходящий из графа в направлении $\alpha \in \{0,1\}$, и пусть $M(l^{\alpha})$ – множество вершин, через которые проходит путь l^{α}. Для того, чтобы проверялась вершина $m \in M(l^{\alpha})$, необходимо, чтобы был активизирован дополнительно некоторый путь $l^{\bar{\alpha}}(m^{\bar{\beta}})$, начинающийся из последователя вершины m (нижнего, если $\bar{\beta} = 0$, и правого, если $\bar{\beta} = I$) и выходящий из графа в направлении $\bar{\alpha}$. Здесь β – направление выхода из вершины m по пути l^{α}.

Выполнение указанного условия в общем случае трудоемко, так как может привести к длинным поискам непротиворечивого решения методом проб и ошибок. Другими словами, в этом и заключается традиционная задача синтеза тестов ЦУ.

Предполагая, что скорость сканирования модели АГ приемлема для проведения в реальном времени, должно и дополнение любого эксперимента до конкретного теста соответствовать требованиям быстродействия. Здесь возможны два решения:

I) случайное дополнение;

2) детерминированное полное дополнение.

При случайном дополнении эксперимента понимаем следующее. Пусть Z - множество всех входных переменных объекта, а $Z_1 \subseteq Z$ - подмножество переменных, значения которых зафиксированы текущим экспериментом. Тогда случайное дополнение эксперимента заключается в случайном генерировании различных наборов значений для переменных $z \in Z_2 = Z \setminus Z_1$. Чем больше таких наборов генерируется, тем больше вероятность, что текущий эксперимент превращается в тесты, предпосылкой которых он является.

Существенно подчеркнуть тот факт, что значение выходного сигнала объекта при случайном дополнении эксперимента остается неизменным, тем, что установлено самим экспериментом, т.е. направлением выхода из графа. Это существенно потому, что тогда не возникает проблемы получения эталонных реакций объекта, что типично для методов генерирования случайных тестов.

При детерминированном дополнении эксперимента значения переменных $z \in Z_2$ устанавливаются детерминированно так, чтобы обязательно все тесты, предпосылкой которых текущий эксперимент является, реализовались.

Разбиваем множество Z следующим образом:

$$Z = Z^* \cup Z^0 \cup Z^1, \qquad (2)$$

где Z^* - подмножество переменных, которые или взвешивают хотя бы одну немонотонную вершину [3], или встречаются в качестве весовых переменных как с инверсией, так и без инверсии;

$Z^0(Z^1)$ - подмножество переменных, которые встречаются в вершинах АГ только без инверсии (только с инверсией) и не входят в Z^*.

Текущий эксперимент, устанавливающий значения $z \in Z_1 \subseteq Z$, будет доопределен до всех возможных тестов, предпосылкой которых он является, если в результирующих наборах существуют следующие значения:

I) переменные $z \in Z^* \cap Z_2$ должны иметь оба значения;

2) переменные $z \in Z^0 \cap Z_2$ ($z \in Z^1 \cap Z_2$) должны получить

противоположное (то же самое) значение по сравнению с тем, которое имеет выход схемы.

Заметим, что доопределение значений переменных $z \in (Z^0 \cup Z^1) \cap Z_2$ очень простое. Для этого можно использовать следующий алгоритм (который может быть легко реализован в тестере аппаратно):

$$S := S' \vee (S^0 \wedge M)\bar{z} \vee (S^1 \wedge M)z, \qquad (3)$$

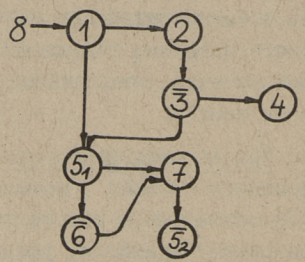
где S – окончательный вектор входного набора;

S' – вектор входного набора, установленный текущим экспериментом (зафиксированы значения $z \in Z_1$);

M – маска, выделяющая компоненты вектора $z \in Z^0 \cup Z^1$;

$S^0(S^1)$ – вектор значений переменных $z \in Z^0 \cup Z^1$, которые необходимо присвоить при значении выхода $z = 0$ ($z = 1$).

Доопределение значений переменных $z \in Z^* \cap Z_2$ сводится к сканированию всех возможных наборов для множества $Z^* \cap Z_2$.



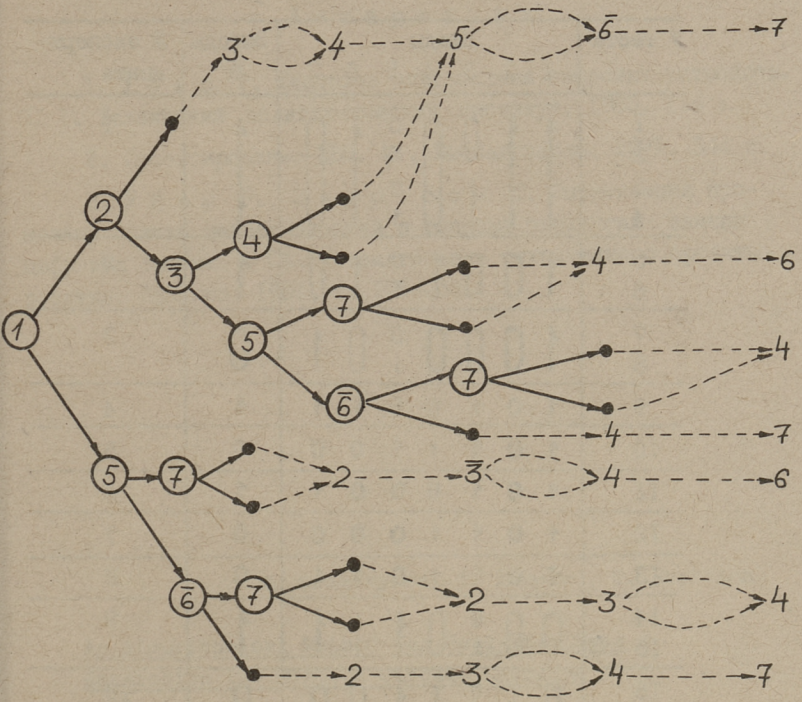
Фиг. 3.

Пример. Рассмотрим ФАГ на фиг. 3, представляющий функцию

$$z_3 = z_1(\bar{z}_2 + \bar{z}_3 z_4) + (\bar{z}_1 + z_3)(z_5 + \bar{z}_6)(\bar{z}_5 + z_7).$$

Сканирование путей графа иллюстрировано на фиг. 4 (пунктиром показано доопределение экспериментов). Здесь имеем $Z^* = \{3,5\}$, $Z^0 = \{1,2,4,7\}$, $Z^1 = \{6\}$. Результирующие наборы приведены в табл. I (жирно выделены сканированные значения). Число всех сканированных экспериментов – I3, число всех полученных тестов – 23. Заметим, что общее количество всех возможных наборов – $2^7 = 128$.

Преимуществом предложенного подхода является то, что не требуется предварительного генерирования тестов и хранения в памяти длинных тестовых массивов. Для управления



Фиг. 4.

экспериментом требуется всего лишь описание объекта в виде модели АГ.

Недостатками данного подхода являются:

- снижение быстродействия тестера, так как процесс сканирования путей на модели АГ медленнее, чем процесс чтения из памяти заранее подготовленных наборов;

- сканирование модели эффективно относительно только одного выхода схемы; в присутствии выходов требуется проведение экспериментов отдельно для каждого выхода;

- при сканировании модели АГ для асинхронных автоматов возникают дополнительные задачи генерирования установочных последовательностей, что, в свою очередь, замедляет метод.

Таблица I

№ теста	Входы 1 2 3 4 5 6 7	Выходы 8	№ эксперимента
1 2 3 4	1 1 0 0 0/1 1 0	1	I
5 6	1 0 0 1 0/1 1 0	1	2
7 8	1 0 0 0 0/1 0 1	0	3
9	1 0 1 0 1 1 1	1	4
I0	1 0 1 1 1 0 0	0	5
II	1 0 1 0 0 0 1	1	6
I2	1 0 1 1 0 0 0	0	7
I3	1 0 1 1 0 1 1	0	8
I4 I5	0 0 0/1 0 1 1 1	1	9
I6 I7	0 1 0/1 1 1 0 0	0	I0
I8 I9	0 0 0/1 0 0 0 1	1	II
20 2I	0 1 0/1 1 0 0 0	0	I2
22 23	0 1 0/1 1 0 1 1	0	I3

Учитывая вышеизложенное, метод генерирования тестов в реальном времени путем сканирования модели АГ представляется целесообразным в следующих случаях:

- когда модели объекта заметно проще, чем структурные модели, требуемые для генерирования структурных тестов;

- когда объект имеет регулярную структуру, тесты для которой очень длинны, но легко генерируемы (регистры, счетчики, запоминающие устройства, микропроцессоры и т.д.).

Литература

I. Убар Р. Тестовая диагностика цифровых устройств. Ч. I. Таллинский политехнический институт, 1980. 114 с.

2. Проектирование ЦВМ. М., Высшая школа, 1972. 344 с.

3. Убар Р., Эвартсон Т. Оптимизация процессов поиска неисправностей в типовых элементах замены цифровых систем. - Сб.: Автоматизация технического проектирования. Вильнюс, 1981, с. 175-184.

K. Grigoryeva, T. Lohuaru, T. Evartson

An Algorithmical Method for Real-Time Test Generation for Digital Circuit Checking

Summary

An algorithmical method for test generation by automatic tester during the checking process of digital circuit is presented. The object under checking is presented in the memory of tester by the alternative graph model. The tests are generated by scanning all the existing ways at the graph model and consequent complementing of these ways. A random method and a determinated one for complementing the ways at the graph are presented.

№ 530

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.3.001.63:621.3

В.И. Божич, Г.А. Галуев, А.В. Судницын

СИНТЕЗ КОММУТАТОРОВ НА ОСНОВЕ ПЛМ ДЛЯ МИКРОПРОЦЕССОРОВ, РЕАЛИЗУЮЩИХ НАБОРЫ КРУПНЫХ ОПЕРАЦИЙ

Особенностью микропроцессоров, реализующих наборы крупных операций, является наличие коммутатора в структуре его арифметико-логического устройства (АЛУ) [1]. Введение коммутатора позволяет организовать структурное программирование микропроцессора, увеличить его быстродействие [1, 2].

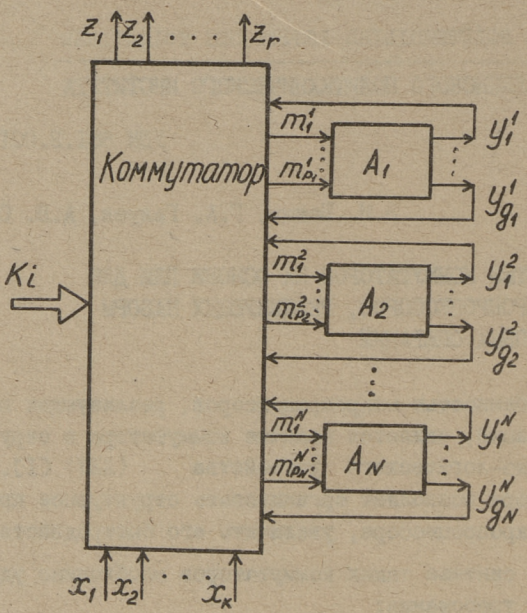
При синтезе таких коммутаторов необходимо удовлетворить два требования:

1) перестройка коммутатора должна производиться практически мгновенно;

2) базисом реализации коммутаторов должны быть перепрограммируемые БИС.

Удовлетворить эти требования можно использованием в качестве базиса программируемые логические матрицы (ПЛМ) [3], тем более, что ПЛМ нашли широкое применение в структурах, выпускаемых промышленностью микропроцессоров. Однако при этом необходимо представить коммутатор в виде логического преобразователя.

Как следует из источника [4], коммутатор должен осуществлять различные варианты межсоединений подавтоматов ($j = 1, 2, \ldots, N$), реализующих в общем случае различные функции (фиг. 1). Пример множества подавтоматов в структуре АЛУ такого микропроцессора приведен в источнике [2]. Причем, каждый вариант коммутации соответствует выполнению одной из крупных операций. Настройка на заданный вариант межсоединения подавтоматов (из числа n различных систем каналов связи) осуществляется под действием команды k_i

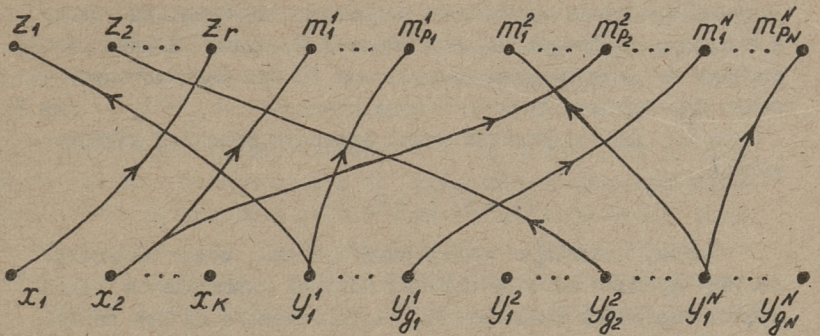


Фиг. 1.

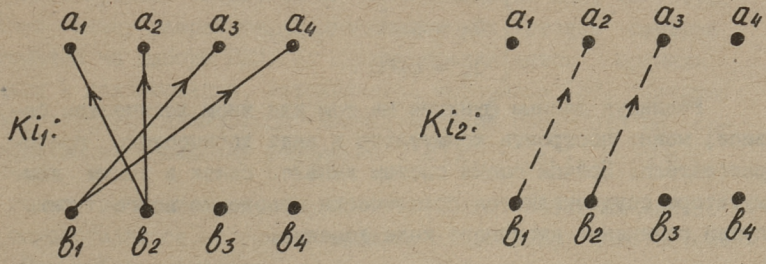
$(i = 1,2,\ldots,n)$, поступившей из микропрограммного управляющего автомата в виде двоичного кода $k_i = (\tilde{i}_1, \tilde{i}_2, \ldots, \tilde{i}_s)_i$, $(\tilde{i} \in \{0,1\})$, [I].

Рассмотрим особенности описания закона функционирования коммутатора системой функций алгебры логики. Каждой команде k_i поставим в соответствие двудольный ориентированный граф G, определяющий каналы связи в коммутаторе. Причем вершинам первого и второго подмножеств соответствуют выходные и входные шины коммутатора. Соединение в графе вершин ребрами, ориентированных от вершин второго подмножества с вершинами первого подмножества, производится в соответствии с заданной системой каналов связи. Общий пример такого графа приведен на фиг. 2.

С целью простоты дальнейших рассуждений рассмотрим два графа G_1 и G_2 (фиг. 3), соответствующих командам k_{i_1} и k_{i_2}. Вершины первого подмножества условно обозначены $a_{\alpha_1} \in \{z_{j_1}, m^i_1, m^i_2, \ldots, m^i_{p_i}\}$, а вершины второго подмножества

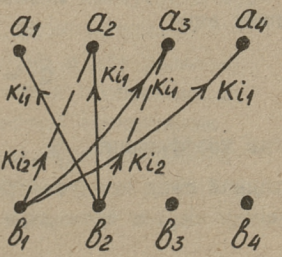


Фиг. 2.



Фиг. 3.

обозначены $b_{\alpha_2} \in \{x_{j_1}, y^i_1, y^i_2, \ldots, y^i_{g_i}\}$. Заменим эти два графа одним обобщенным графом (фиг. 4), полученным объединением графов G_1 и G_2. При этом ребра, соответствующие графу G_1, пометим меткой k_{i_1}, а ребра графа G_2 – меткой k_{i_2}. Построенный таким образом обобщенный граф позволяет проиллюстрировать принцип реализации коммутатора в виде логического преобразователя на основе ПЛМ. Действительно, из обобщенного графа видно, что



Фиг. 4.

условие появления единичного сигнала, например, на выходной шине коммутатора a_2 заключается в подаче единичного сигнала на входную шину коммутатора b_1 или b_2, соответственно для случаев двоичного кода команды $k_{i_2} = l_1$ или $k_{i_1} = \bar{l}_1$. Данное условие можно записать функцией алгебры логики:

$$a_{i_2} = b_1 \cdot l_1 \vee b_2 \cdot \bar{l}_1 .$$

Тогда, используя приведенный принцип, можно записать систему функций алгебры логики, которой описывается закон функционирования коммутатора, настраиваемого на две системы каналов связи в соответствии с графами G_1 и G_2

$$\begin{cases} a_1 = b_2 \cdot \bar{l}_1 \\ a_2 = b_2 \cdot \bar{l}_1 \vee b_1 \cdot l_1 , \\ a_3 = b_1 \cdot \bar{l}_1 \vee b_2 \cdot l_1 , \\ a_4 = b_1 \cdot \bar{l}_1 . \end{cases}$$

Реализуя данные функции на том или ином логическом базисе, можно построить коммутатор в виде логического преобразователя. Причем смена систем каналов связи в таком коммутаторе осуществляется практически мгновенно всего лишь путем изменения двоичного кода команды.

В общем случае при реализации коммутатора в виде логического преобразователя, настраиваемого на число n различных систем каналов связи (каждая из которых соответствует определенной команде k_i), необходимо:

- построить обобщенный граф $G = \bigcup_{i=1}^{n} G_i$, а затем
- по этому графу записать систему функций алгебры логики вида

$$a_{j_1} = \bigvee_{i=1}^{n} k_i \cdot b_{j_2} \cdot \beta_{i,j_1,j_2} , \qquad (I)$$

где коэффициент $\beta_{i,j_1,j_2} = 1$, если существует соединение выходной шины коммутатора a_{j_1} с входной шиной b_{j_2} для команды k_i, в противном случае $\beta_{i,j_1,j_2} = 0$.

Заметим, что различные системы каналов связи, каждая из которых соответствует команде k_i, могут иметь каналы связи, соединяющие одни и те же входные и выходные шины коммутато-

ра. Следовательно, различные функции алгебры логики (I), заданные в СДНФ, будут иметь одинаковые члены СДНФ, что позволяет удобно реализовать систему таких функций на основе ПЛМ.

Также заметим, что двоичный код k_i изменяется в процессе работы микропроцессора по определенной микропрограмме. Следовательно, алгоритм работы коммутатора и алгоритм смены систем каналов связи можно представить моделью конечного автомата, что приводит к необходимости синтеза такого автомата, наряду с микропрограммным автоматом, управляющим работой микропроцессора. При этом, как отмечалось выше, удобным базисом является программируемая логическая матрица. Однако, ввиду ограниченных "размеров" ПЛМ, такой управляющий автомат можно синтезировать только на основе нескольких ПЛМ, что остро ставит задачу разработки декомпозиционных методов синтеза автоматов на наборе ПЛМ. За основу предлагается принять подход к декомпозиции, изложенный в источнике [5]. Следуя этому методу, декомпозицию автомата можно построить по ортогональному множеству разбиений $P=\{\pi_1,\ldots,\pi_n\}$, $\prod_{i=1}^{n} \pi_i = 0$ на множестве состояний исходного автомата. Выбор множества P осуществляется на основе анализа декомпозируемого автомата. При выборе исходного множества разбиений можно использовать методику, изложенную в источнике [6].

В заключение рассмотрим вопрос использования изложенной идеи построения коммутатора с ускоренной настройкой в виде логического преобразователя на более высоком уровне – при создании однородных многопроцессорных вычислительных структур (ОМВС) [I]. Здесь также указанный способ коммутации можно использовать не только для соединения блоков отдельного процессора (в частности, микропроцессора), но и для реализации необходимой структуры соединений между самими процессами. Более того, такой способ коммутации оказывается единственно возможным при синтезе управляющих систем с перестраиваемой структурой (УСПС) [7], которые используются при управлении динамическими процессами или объектами, параметры которых быстро меняются во времени.

Таким образом, настройка ОМВС [4] УСПС осуществляется заданием вариантов коммутации как для процессоров, так и для межпроцессорных связей. Использование же способа построения

структуры коммутатора, предложенного выше, позволяет сделать процедуру настройки и перестройки всей структуры простой, ускоренной и однотипной. Более того, если ввести зависимость между появлением двоичных наборов, определяющих внутрипроцессорные и межпроцессорные связи, от внешней ситуации, то такая многопроцессорная система приобретает способности структурной адаптации к изменениям внешней среды. При этом алгоритмы работы коммутатора самого процесса и коммутатора, определяющего соединение процессов, а вместе с ними алгоритм изменения множества двоичных наборов k_i, еще более усложняется. Последнее еще раз подчеркивает необходимость разработки декомпозиционных методов синтеза цифровых автоматов применительно к проблеме создания микропроцессоров с набором крупных операций, а также ОМВС и УСПС на их основе.

Литература

I. Калляев А.В. Многопроцессорные перестраиваемые структуры. - В кн.: Труды пятого Международного семинара "Прикладные аспекты теории автоматов", т. 2. Варна, Болгария, изд-во БАН, 1979, с. 453-464.

2. Калляев А.В. Многопроцессорные системы с распределенной памятью. - Электронное моделирование, 1979, № 1, с. 31-42.

3. Якубайтис Э.А. Синтез структуры программируемой логической матрицы.- Автоматика и вычислительная техника, 1976, № 4, с. 1-10.

4. Калляев А.В. Автоматы с программируемой структурой и коммутацией. - Problems of Control and Information Theory. Budapest, N 1, 1975, p. 35-56.

5. Jacobson G., Keevallik A., Leis P. Some aspects of the construction of microprogram automata networks. - IFAC-symposium "Discrete systems", Dresden, 1977, vol. 1, p. 120-128.

6. Лейс П.Л., Судницын А.В. Один метод декомпозиционного синтеза микропрограммных автоматов на программируемых логических матрицах. - Труды Международного

семинара "Прикладные аспекты теории автоматов", т. I. Варна, 1979, с. 253-262.

7. Букатова И.Л., Елинсон М.И. Современные системы управления и задачи микроэлектроники. Микроэлектроника, т. 10, вып. I, 1981, с. 42-58.

V. Bozhich, A. Galuev, A. Sudnitsyn

The Synthesis of Switches Using PLA for Microprocessors Realizing the Set of Macrooperations

Summary

The method based on the decomposition of control automata is proposed for the synthesis of switches for microprocessors realizing the set of macrooperations proposed. It is shown that if we consider the switch a logical transducer, it is feasible to realize a high speed network on programmable logic arrays.

№ 530

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 62.507

Б.Е. Беркман

МЕТОД ВЫБОРА РАЗБИЕНИЙ ДЛЯ ДЕКОМПОЗИЦИИ МПА С РАЗДЕЛЕНИЕМ ВХОДНЫХ ПЕРЕМЕННЫХ

I. Введение и постановка задачи

Настоящая работа посвящена решению одной важной, с практической точки зрения, задачи декомпозиции микропрограммного автомата (МПА), при которой ограничено число входных каналов компонентных автоматов. Эта задача возникает, например, при синтезе МПА на базе БИС, которые ограничены по количеству входных полюсов.

Декомпозиция автоматов, разработанная Хартманисом и Стирнзом [1], была развита и распространена на микропрограммные автоматы [2]. Этот известный метод декомпозиции МПА по ортогональному множеству разбиений $P = \{\pi_1, \pi_2, \ldots, \pi_n\}$ на множество состояний декомпозируемого автомата может быть применен лишь после нахождения ортогонального множества разбиений таких, которые давали бы необходимые свойства сети из компонентных автоматов. Таким образом, несмотря на наличие общего конструктивного метода декомпозиции, многие важные для практики задачи декомпозиции не могут быть решены, так как не существует методики нахождения разбиений, обеспечивающих требуемые свойства получаемой сети автоматов.

Данная работа посвящена решению одной из таких задач – нахождению ортогонального множества разбиений для декомпозиции с ограничением на число входов компонентных автоматов сети и развивает идеи, изложенные в источниках [3, 4]. В этой работе уточняются некоторые необходимые понятия, что позволяет расширить класс декомпозируемых с учетом данных ограничений МПА. Приведенные алгоритмы ориентированы на решение задачи декомпозиции для автоматов больших размерностей (до сотен состояний).

2. Основные понятия

Определение I. Формально под МПА будем подразумевать пятерку

$$A = (S, X, Y, \delta, \lambda),$$

где $S = \{s_1, \ldots, s_m, \ldots, s_M\}$ – множество состояний;

$X = \{x_1, \ldots, x_l, \ldots, x_L\}$ – множество входных двоичных переменных;

$Y = \{y_1, \ldots, y_n, \ldots, y_N\}$ – множество выходных двоичных переменных;

$\delta : S \times \{x_l\} \to S$ – функция переходов;

$\lambda : S \to \{y_n\}$ или $S \times \{x_l\} \to \{y_n\}$ – функция выходов (модель Мура или модель Мили).

МПА может быть задан в виде списка. Одной из таких форм является прямая таблица переходов МПА, в которой (см. табл. I) в первом и втором столбцах записываются соответственно исходное состояние s_m и состояние перехода s_r, в третьем столбце записывается входной сигнал $x(s_m, s_r)$, под действием которого происходит переход из состояния s_m в состояние s_r. В четвертом столбце (который отсутствует в МПА без выхода) записывается выходной сигнал $Y(s_m, s_r)$ (в случае автомата Мили). В случае автомата Мура можно обойтись всего тремя столбцами, записывая выходной сигнал $Y(s_m)$ рядом с соответствующим ему исходным состоянием.

Каждому состоянию s_m соответствует система булевых функций $F_m = \{f_{m_1}, \ldots, f_{m_r}, \ldots, f_{m_R}\}$. Автомат из состояния s_m переходит в состояние s_r, если f_{m_r} является истинной на поступившем входном наборе $\Delta_i \in \{0,1\}^L$. Из однозначности функций переходов следует ортогональность этих функций относительно операции конъюнкции. Обычно функции из F_m заданы в ДНФ, и в каждой строке таблицы переходов записывается некоторая конъюнкция.

Для МПА, который полностью определен, система булевых функций F_m обладает свойством полноты, т.е. $\bigvee_r f_{m_r} = 1$. Свойство полноты может быть нарушено, если при переходе из состояния s_m заданы неиспользуемые входные наборы. Кроме того, структурное задание входных и выходных сигналов по-

зволяет описывать дискретные устройства самой различной сложности. Поэтому в данной работе рассматривается также задание алгоритма функционирования управляющего устройства таблицей, в которой системы булевых функций могут не обладать свойством полноты. Такую модель описания поведения автомата будем называть неполностью определенным или частичным микропрограммным автоматом.

При изучении различных проблем синтеза МПА часто можно ограничиться рассмотрением МПА без выходов, задаваемых тройками $A = (S, X, \delta)$, где S, X, δ имеют тот же смысл, что и прежде.

Определение 2. Некоторая пара элементарных конъюнкций (C_1, C_2), принадлежащая соответственно булевым функциям f_1 и f_2, заданным в ДНФ, находится в отношении смежности $(C_1 \, adj \, C_2)$ по компоненте i, если и только если все соответствующие компоненты этой пары конъюнкций неортогональны, кроме компоненты порядка i (т.е. в одну из конъюнкций компонента порядка i входит с отрицанием, а в другую - без отрицания). В этом случае будем говорить, что и сами функции f_1 и f_2 находятся в отношении смежности.

Таблица I

Исход. сост.	Сост. перех.	Входной сигнал
I	2	$x_1 x_2$
	3	$x_1 \bar{x}_2$
	5	$\bar{x}_1 x_2$
	IO	$\bar{x}_1 \bar{x}_2$
2	2	$\bar{x}_1$
	5	x_1
3	I	$\bar{x}_3 \bar{x}_4$
	4	$x_3 \bar{x}_4$
	6	$\bar{x}_3 x_4$
	7	$x_3 x_4$
4	2	x_2
	3	$\bar{x}_2$
5	3	x_6
	8	$\bar{x}_5 \bar{x}_6$
	9	$x_5 \bar{x}_6$
6	2	$\bar{x}_1 \bar{x}_2$
	3	$\bar{x}_1 x_2$
	5	$x_1 \bar{x}_2$
	IO	$x_1 x_2$
7	2	$\bar{x}_1$
	5	x_1
8	I	x_4
	6	$\bar{x}_4$
9	I	$x_3 x_4$
	4	$\bar{x}_3 x_4$
	6	$x_3 \bar{x}_4$
	7	$\bar{x}_3 \bar{x}_4$
IO	3	$\bar{x}_6$
	8	$\bar{x}_7$

Определение 3. Пусть b_d – некоторое подмножество множества S. Тогда разбиение $\pi_1=\{b_1,\ldots,b_n\}$ на множестве $S\setminus b_d$ будем называть неполностью определенным разбиением (НОР) на множестве S и записывать его как $\pi=\{b_1,\ldots,b_n,\ldots,b_d\}$, где b_d – специальный блок НОР.

НОР π будем рассматривать как компактную запись целого множества разбиений на S (которые в отличие от НОР будем писать как ПОР) $G(\pi)$, называемых генерируемыми (порождаемыми) НОР π. Причем, $\pi'\in G(\pi)$, если и только если для каждого $b'\in\pi'$ либо найдется $b\in\pi$ такой, что $b'\leq b\cup b_d$, либо $b'\leq b_d$.

Это определение НОР отличается от ранее данных в источниках [4, 5] тем, что разрешается (при генерации ПОР) создание блоков, являющихся подмножествами b_d, что увеличивает мощность множества $G(\pi)$ и, следовательно, расширяет класс декомпозируемых автоматов. Эта особенность определяет введение следующего отношения порядка.

Определение 4. НОР π_1 меньше или равно НОР π_2 ($\pi_1\leq\pi_2$), если и только если для всех неспециальных блоков $b_i\in\pi_1$ существует $b_j\in\pi_2$ такой, что $b_i\leq b_j\cup b_{d_2}$, где b_{d_2} – специальный блок π_2.

Кроме данного отношения порядка введем отношение порядка в смысле степени определенности НОР.

Определение 5. НОР π_1 меньше определено, чем НОР π_2 ($\pi_1\leq_G\pi_2$), если и только если:

1) каждый неспециальный блок π_1 содержится в некотором неспециальном блоке π_2;

2) каждый неспециальный блок или содержит в точности один неспециальный блок π_1, или сам содержится в специальном блоке π_1.

Ясно, что $\pi_1\leq_G\pi_2$, если и только если $G(\pi_2)\leq G(\pi_1)$.

Можно доказать, что $\pi_1\leq\pi_2$ (π_1,π_2 – НОР) в том и только в том случае, если существуют разбиения (ПОР) τ_1 и τ_2 такие, что $\tau_1\in G(\pi_1)$, $\tau_2\in G(\pi_2)$ и $\tau_1\leq\tau_2$. Очевидно, что $G(\pi)$ включает те и только те разбиения (ПОР) τ, которые не меньше определены, чем π, т.е. $\pi\leq_G\tau$.

Операции суммирования и произведения двух НОР π_1 и π_2 определяются аналогично [5].

Пусть дано множество неполностью определенных разбиений $P = \{\pi_1, \ldots, \pi_n\}$ на множестве S. Определим матрицу связей для P как двоичную матрицу бинарного отношения связности на множестве S. При этом выделим следующие два отношения связности.

Определение 6. Бинарное отношение минимальной связности - это отношение на множестве S такое, что некоторые два элемента S_i и S_j находятся в этом отношении (в матрице связей стоит единица в пересечении порядка i строки, порядка j столбца), если и только если в каждом разбиении из P π_i $(i = 1 \div n)$ существует неспециальный блок, в котором содержатся оба S_i и S_j.

Определение 7. Бинарное отношение максимальной связности - это отношение на множестве S такое, что некоторые два элемента S_i и S_j находятся в этом отношении (в матрице связей стоит единица в пересечении порядка i строки и порядка j столбца), если и только если не существует такого разбиения в P, в котором состояния S_i и S_j входили бы в различные неспециальные блоки.

Связь (S_i, S_j) считается разбитой, если S_i и S_j не находятся в отношении связности (в пересечении порядка i строки и порядка j столбца матрицы связей стоит нуль) и неразбитой - в противном случае. Связь (S_i, S_j), $i = 1 \div N$ для бинарных отношений связности двух видов считается разбитой по определению. Для разбиений (ПОР) бинарные отношения связности обоих видов (минимальной и максимальной) совпадают.

Определение 8. НОР π считается нулевым $(\pi = 0)$, если и только если все его блоки, кроме специального, состоят в точности из одного состояния (в этом случае существует нулевое ПОР в множестве ПОР, генерируемых данными НОР). НОР называется непосредственно нулевым (нулевым по сумме).

Множество НОР будем называть ортогональным, если произведение входящих в него НОР равно нулевому НОР $(\prod_{i=1}^{n} \pi_i = 0)$.

Очевидно, что в этом случае матрица отношения минимальной связности будет состоять из одних нулей.

3. Метод построения искомого ортогонального множества

Построение ортогонального множества разбиений для данного вида декомпозиции разбивается на три самостоятельных этапа, описание которых будет сопровождаться иллюстрацией декомпозиции на примере МПА, изображенного в таблице I.

3.I. Нахождение первичных α-разбиений

Определение 9. Первичные α-разбиения на множестве состояний S МПА $\alpha(i), i = 1 \div L$ (L - количество входов МПА) - это минимальные (в смысле отношения $\leq_G$) НОР, которые обладают следующим свойством:

два состояния S_p и S_q находятся в одном и том же неспециальном блоке разбиения $\alpha(i)$, если существует такое состояние S_t, что функции перехода f_{tp} и f_{tq} находятся в отношении смежности по компоненте i (опред. 2).

Из этого определения непосредственно следует алгоритм построения первичных α-разбиений.

При построении $\alpha(i)$ $(i = 1 \div L)$ перебираются все состояния $(t = 1 \div M)$.

Для каждого перехода (кроме последнего) в таблице переходов автомата из состояния S_t в состояние S_p $(p \in I_t$, I_t - множество индексов состояний, в которые автомат может переходить из состояния S_t), находятся все такие переходы из S_t в некоторое состояние S_q $(q \in I_t, q > p)$, входная функция которых $(f(S_t, S_q))$ смежна по компоненте i с $f(S_t, S_p)$. Все такие состояния S_q заключаются в один блок друг с другом и с состоянием S_p. после чего находится транзитивное замыкание нового блока (если он не пустой) с уже найденными блоками строимого разбиения, т.е. все блоки, с которыми новый блок пересекается, объединяются друг с другом и с ним в новый блок.

После описанного перебора к полученным неспециальным блокам разбиения $\alpha(i)$ добавляется специальный блок, в ко-

торый входят состояния, не вошедшие ни в один неспециальный блок.

Нетрудно видеть, что время, затрачиваемое на реализацию данного алгоритма, определяется количеством сравнений $f(S_t, S_p)$ и $f(S_t, S_q)$ на соседство. Это количество равно

$$N = L \cdot \sum_{t=1}^{M} C^2_{|I_t|} = L \cdot \sum_{t=1}^{M} \frac{|I_t| \cdot (|I_t| - 1)}{2},$$

где $|I_t|$ - мощность множества переходов из состояния S_t;

$C^2_{|I_t|} = \frac{|I_t|(|I_t|-1)}{2}$ - количество различных пар переходов (S_t, S_p) и (S_t, S_q) из состояния S_t;

N - ограничено сверху числом $\frac{L \cdot M^2}{2}$ для всех автоматов размерности (L - число входов;

M - число состояний.

Итак, имеем полиномиальный алгоритм с оценкой перебора - $O(L \cdot M^2)$.

В результате выполнения описанного алгоритма для автомата в таблице I найдены следующие первичные α- разбиения:

$\alpha(1) = (\overline{2,5};\ \overline{3,10};\ <1,4,6,7,8,9>)$

$\alpha(2) = (\overline{2,3};\ \overline{5,10};\ <1,4,6,7,8,9>)$

$\alpha(3) = (\overline{1,4};\ \overline{6,7};\ <2,3,5,8,9,10>)$

$\alpha(4) = (\overline{1,6};\ \overline{4,7};\ <2,3,5,8,9,10>)$

$\alpha(5) = (\overline{8,9};\ <1,2,3,4,5,6,7,10>)$

$\alpha(6) = (\overline{3,8,9};\ <1,2,4,5,6,7,10>)$.

3.2. Построение ортогонального множества α-разбиений

Определение I3: α-разбиением порядка r называется неполностью определенное разбиение

$$\alpha^r(L'_x) = \sum_{j \in L'_x} \alpha(j),$$

где $L'_x = \{i_1, \ldots, i_r\} \leq \{1, \ldots, l\} = l_x$.

Здесь l - количество входов МПА, т.е. α - разбиение порядка r равно сумме некоторых r-первичных разбиений. Ясно, что первичные α-разбиения - это α-разбиения первого порядка.

В источнике [4] доказана следующая теорема.

Поведение компонентного автомата не зависит существенно от множества внешних входных переменных L'_x, если и только если $\pi_i \geq \alpha(L'_x)$.

Отсюда ясно, что для того, чтобы получить некоторое ортогональное множество разбиений (ПОР) $\{\pi_i\}$ для декомпозиции с зависимостью каждого компонентного автомата от $(l-r)$ (или независимостью от r) внешних входов надо получить некоторое множество α- разбиений порядка r и затем преобразовать его в множество ПОР (с помощью замены каждого НОР на некоторое генерируемое им ПОР).

Можно доказать, что некоторое множество НОР $\{\alpha_i\}$ можно преобразовать в ортогональное множество ПОР $\{\pi_i\}$, если и только если это множество НОР ортогонально.

При построении ортогонального множества α-разбиений порядка r будем стремиться найти множество $\{\alpha_i\}$ минимальной мощности.

Для точного решения этой задачи необходимо было бы найти все возможные α-разбиения порядка r, (чье количество равно M_r) и затем перебирать возможные подмножества этих разбиений (чье общее число равно 2^{M_r}, т.е. $2^{C_l^r}$). Разумеется, что этот перебор неосуществим даже при значениях r и l порядка десяти (ведь этот перебор $(O(2^{C_l^r}))$ больше экспоненциального).

Поэтому вполне естественно, что в данной работе предлагается не точная, а приближенная минимизация количества α-разбиений в ортогональном множестве (гарантирующая, однако, что в случае принципиальной возможности построения ортогонального множества НОР оно будет построено).

Эвристика предлагаемого алгоритма построения ортогонального множества НОР состоит в том, что это множество (вначале пустое) на каждом шаге дополняется одним α-разбиением. Причем в качестве очередного пополняющего множества α-разбиения выбирается такая сумма из r первичных разбиений, которая (при пополнении ею множества НОР) оставляет минимум связей в матрице связей I-го рода (в двоичной матрице бинарного отношения минимальной связности).

Нетрудно показать, что этот алгоритм обязательно приводит к построению ортогонального множества α-разбиений. Это гарантируется точностью алгоритма выбора оптимального пополняющего разбиения, который приводит к постоянному уменьшению (пока это возможно) связей в матрице связей. Это постоянное уменьшение может закончиться одним из двух событий:

I) все связи уничтожены, следовательно, получено ортогональное множество α-разбиений;

2) на некотором шаге оказалось невозможным уничтожить ни одной связи. Поскольку алгоритм выбора оптимального разбиения точный, это означает, что ни одну из оставшихся связей уничтожить нельзя (т.е. любая сумма из r первичных разбиений не может их уничтожить), и что невозможно построить ортогональное множество α-разбиений порядка r.

Максимальный размер перебора для построения ортогонального множества НОР равен $|\{\alpha_i\}| \cdot C_l^r$.

Заметим, что если все r первичных α-разбиений ненулевые, то минимальный размер $|\{\alpha_i\}|$ равен $\left]\frac{l}{l-r}\right[$ (если $|\{\alpha_i\}| < \frac{l}{l-r}$, то оно не может быть ортогональным, так как некоторое первичное α-разбиение войдет в каждую из сумм).

Для того, чтобы сразу отбросить случаи заведомо неразрешимые, приведем следующее необходимое условие существования ортогонального множества α-разбиений: необходимо, чтобы любая пара состояний S_p, S_q входила в один и тот же неспециальный блок не более, чем в r НОР из множества всех первичных разбиений.

Необходимо отметить, что в алгоритм введена возможность изменения находимого ортогонального множества $\{\alpha_i\}$. Для реализации этой возможности в качестве матрицы отношения, связи в которой минимизируются, на каждом шаге берется матрица, получаемая из фактической матрицы отношения в результате оставления случайных связей.

В результате применения описанного алгоритма к первичным разбиениям $\{\alpha(i)\}$, найденным ранее, получается ортогональное множество из 3-х α-разбиений 4-го порядка:

$$\alpha_1 = \alpha(1)+\alpha(2)+\alpha(3)+\alpha(4)=(\overline{2,3,5,10};\ \overline{1,4,6,7};\ <8,9>)$$
$$\alpha_2 = \alpha(1)+\alpha(2)+\alpha(5)+\alpha(6)=(\overline{2,3,5,8,9,10};\ <1,4,6,7>)$$
$$\alpha_3 = \alpha(3)+\alpha(4)+\alpha(5)+\alpha(6)=(\overline{1,4,6,7};\ \overline{3,8,9};\ <2,5,10>)$$

3.3. Преобразование ортогонального множества НОР в ортогональное множество ПОР

Перейдем теперь к рассмотрению этапа преобразований найденного ортогонального множества НОР $\{\alpha_i\}$ в ортогональное множество ПОР $\{\pi_i\}$. При этом будем стремиться построить $\{\pi_i\}$ такое, в котором общее число блоков в разбиениях минимально.

Точный алгоритм с минимизацией общего числа блоков в разбиениях должен бы был перебрать все возможные варианты преобразований. Количество различных преобразований НОР в ПОР для каждого НОР α_i (здесь g_i - количество неспециальных блоков в нем, а h_i -количество состояний в его специальном блоке) может быть ограничено снизу следующим образом:

Если все h_i состояния распределять между g_i неспециальными блоками, то это будет $g_i^{h_i}$ вариантов. Тогда без учета остальных вариантов преобразования (когда некоторые элементы специального блока НОР α_i образуют новые блоки ПОР π_i), общее количество вариантов преобразования $N = \prod_i N_i \geq \prod_i g_i^{h_i}$. Очевидно, что даже при небольших размерах НОР, входящих в множество $\{\alpha_i\}$, полный перебор

вариантов невозможен. Поэтому необходимо приближенное решение.

Эвристический алгоритм преобразования ортогонального множества НОР в ортогональное множество ПОР с приближенной минимизацией общего числа блоков в множестве ПОР может быть следующим:

Строится матрица связей 2-го рода (бинарного отношения максимальной связности) для множества НОР. Затем, по одному преобразуются НОР α_i в ПОР π_i. Причем, каждое состояние s_j специального блока объединяется с таким неспециальным блоком b_p этого разбиения, все связи с элементами которого для s_j уже разорваны другими разбиениями (т.е. b_p берется такой, что в матрице связей в пересечении порядка j строки и всех таких порядка t столбцов, что $s_t \in b_p$, стоят нули). Если такого блока b_p нет, то s_j образует новый неспециальный блок из одного элемента. Если таких блоков несколько, то s_j вставляется в I-й (или последний) из них. В итоге специальный блок порядка i разбиения становится пустым, а неспециальные блоки его образуют ПОР, включаемые в строимое множество разбиений. В ходе этого процесса ликвидируются некоторые связи (связи элементов специальных блоков по мере вхождения их в блоки ПОР), что отражается в присваивании нулю всех элементов порядка j строк.

Можно показать, что описанный алгоритм всегда приводит к построению ортогонального множества ПОР. Сложность предлагаемого алгоритма можно оценить следующим образом. При каждом выборе неспециального блока, в который включается состояние s_j из специального блока НОР α_i делается перебор имеющихся в порядка i разбиении блоков (их максимальное число может быть $h_i + g_i - 1$)), где g_i - количество неспециальных блоков в α_i; h_i – количество состояний в специальном блоке НОР α_i. Тогда общее количество просматриваемых вариантов ограничено сверху числом $\sum_{i=1}^{n} h_i \cdot (h_i + g_i)$, где n - количество НОР в $\{\alpha_i\}$. Этот сравнительно небольшой перебор и определяет время реализации предлагаемого алгоритма.

В результате преобразования, полученного на стр ице 102 ортогонального множества НОР $\{\alpha_i\}$, получается следующее ортогональное множество ПОР $\{\pi_i\}$:

$$\pi_1 = (\overline{1,4,6,7,8};\ \overline{2,3,5,10};\ \overline{9})$$

$$\pi_2 = (\overline{1};\ \overline{2,3,5,7,8,9,10};\ \overline{4};\ \overline{6})$$

$$\pi_3 = (\overline{1,2,4,6,7};\ \overline{3,8,9};\ \overline{5};\ \overline{10})\ .$$

Если затем произвести декомпозицию рассматриваемого МПА общим методом по полученному множеству разбиений $\{\pi_i\}$, то получим сеть из 3 комп. МПА:

1 МПА: 3 состояния; 4 входа (2 внешних, 2 внутренних). Связан с 3-им МПА.

2 МПА: 4 состояния; 5 входов (2 внешних,3 внутренних). Связан с 1-ым и 3-им МПА.

3 МПА: 4 состояния; 4 входа (2 внешних, 2 внутренних). Связан со 2-ым МПА.

4. Заключение

Приведенный в данной работе алгоритм запрограммирован на языке ЛЯПАС-М, предназначенном для решения задач логического синтеза. Написанные программы отлажены на ЭВМ ЕС-1022 и их результаты проверены с помощью программы построения сети автоматов полученной в результате декомпозиции исходного автомата. Проведенные эксперименты показали высокую эффективность предложенных алгоритмов и программ, позволяющих решать задачи для автоматов размерности порядка сотни состояний за время менее одной минуты.

Литература

1. Hartmanis J., Stearns R.E. Algebraic structure theory of sequential machines. Prentice-Hall Inc., Englewood Cliffs, N.-Y., 1966, p. 209.

2. Jakobson G., Keevallik A., Leis P. Some aspects of the construction of microprogram automata networks. - IFAC-symposium "Discrete systems", Dresden, 1977, vol. 1, p. 120-128.

3. Лейс П.Л., Судницын А.В. Один метод декомпозиционного синтеза микропрограммных автоматов на программируемых логических матрицах. - Тр. Международного семинара "Прикладные аспекты теории автоматов". Варна, 1979, т. I, с. 253-262.

4. Лейс П.Л., Судницын А.В. Метод декомпозиции микропрограммных автоматов с учетом ограничений на число входных переменных. - Тр. Таллинск. политехн. ин-та, 1980, № 497, с. 41-47.

5. Wilkens E.J. Realizations of sequential machines using random access memory. - IEEE Trans. Comput., 1978, vol. C-27, N 5, p. 429-441.

6. Закревский А.Д., Торопов Н.Р. Система программирования ЛЯПАС-М. Минск, Наука и техника, 1978. 238 с.

B. Berkman

Partition Method for the Decomposition of the MPA with Distribution of Inputs

Summary

A method for the decomposition of the microprogram automata (MPA) into the network of component automata with restricted number of binary inputs is introduced. Algorithms to find the set of partitions for such a decomposition are developed and realized using the programming system LYPAS.

№ 530

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 621.382

Т.Х. Ранг, Э.Э. Велмре

МОДЕЛИРОВАНИЕ ТЕХНОЛОГИЧЕСКОГО ПРОЦЕССА КРЕМНИЕВЫХ СИЛОВЫХ ПОЛУПРОВОДНИКОВЫХ ПРИБОРОВ НА ЭВМ

Введение. Расчет статических и динамических характеристик кремниевых силовых полупроводниковых приборов (СПП), как известно, невозможен без данных о их структуре (распределения примесей, толщины слоев и т.д.). Для программ анализа кремниевых СПП названные данные могут быть подготовлены в пакетах перфокарт начальных данных, но в этом случае пострадает универсальность данной программы, а также будет затруднена возможность исследования влияния технологических параметров (например, температура диффузии примеси) на характеристики СПП.

При разработке программы анализа кремниевых СПП авторы данной статьи выработали модель технологического процесса, которая является входной модулей программы анализа тиристора и описывается в данной статье.

Цель статьи – описание модели технологического процесса, в которой аналитически решаются уравнения диффузии, вычисляется толщина слоев структуры, определяются градиенты концентрации на переходах прибора и т.д.

Таким образом, входными данными технологической модели (а также программы анализа тиристорной структуры) являются температура и время диффузии, удельное сопротивление кремния и углы фаски структуры.

В модели коэффициенты диффузии примеси (D_0) считаются постоянными и процессы охлаждения не учитываются. Эти упрощения весьма приемлемы, так как результаты расчетов показывают хорошее совпадение с измеренными данными.

Моделирование технологического процесса может быть решено также численными методами, как это сделано в работах [I-3], но в конкретном случае этот путь не оправдался, так как требовалась новая модель, которая использовала бы машинного времени мало по сравнению с основной программой.

Описание модели. Как известно из источника [4], температурная зависимость коэффициентов примеси выражается следующим отношением:

$$D = D_0 \exp\left(-\frac{C}{t^\circ + T_0}\right), \qquad (I)$$

где D_0, C – эмпирические коэффициенты диффузионного процесса и имеют разные значения для разных примесей;

T_0 – как правило, 273,I6 °C;

t° – температура диффузии.

В данной модели для алюминия, бора и фосфора взяты следующие значения D_0 и C, которые приведены в табл. I.

Таблица I

Примеси	Коэффициенты	
	D_0 [см²/s]	C [°K]
Al	2,45 x I0⁴	5,3 x I0⁴
B	5,75	4,I8 x I0⁴
P	I0⁻³	2,93 x I0⁴

Диффузионная глубина примесей описывается формулой

$$L = 2\sqrt{Dt^\circ}. \qquad (2)$$

С помощью формул (I) и (2) и аналитических решений распределения определяются места металлургических переходов и ширины слоев структуры

$$W_0 = L_B L_{B^+} \sqrt{\frac{\ln(N_{S0}^{B^+}/N_{S0}^{B})}{L_B^2 - L_{B^+}^2}}, \qquad (3)$$

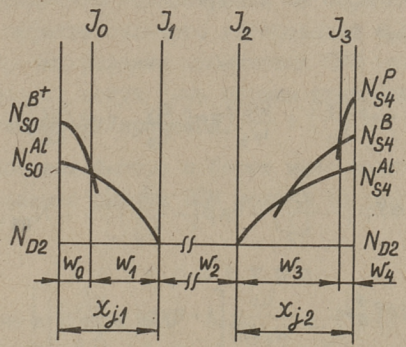
$$x_{j1} = L_{Al} \sqrt{\ln \frac{N_{S0}^{Al}}{N_{D2}}}, \qquad (4)$$

$$x_{j2} = L_{Al}\sqrt{\ln \frac{N_{S4}^{Al}}{N_{D2}}}\,. \qquad (5)$$

x_{j3} получается решением трансцендентного уравнения (6) методом хорд

$$N_{S4}^{P}\,\mathrm{erfc}\,\frac{x_{j3}}{L_P} = N_{S4}^{Al}\exp\left(-\frac{x_{j3}^2}{L_{Al}^2}\right) + N_{S4}^{B}\exp\left(-\frac{x_{j3}^2}{L_B^2}\right). \qquad (6)$$

Остальные ширины слоев и распределения примесей показаны на фиг. I.



Фиг. 1.

Концентрация N_{D2} в n-базе определяется из формулы (7), полученная с помощью кривых Ирвина [5, 6]:

$$N_{D2} = 10^{15{,}75}/\rho_2^{1{,}01}. \qquad (7)$$

В каждом слое структуры концентрация усредняется. Причиной данной процедуры является ускорение работы модуля. Расчеты показали, что машинное время при вычислении подвижностей и концентрации неосновных носителей, используя усредненные концентрации легирования, в три раза короче машинного времени, если вычислять те же величины отдельно, исходя из профиля примеси и затем усреднять их.

Усредненные концентрации вычисляются по формулам

$$N_{A0}=\frac{\sqrt{\pi}}{2w_0}\left(N_{s0}^{Al}L_{Al}\operatorname{erf}\frac{w_0}{L_{Al}}+N_{s0}^{B}L_{B}\operatorname{erf}\frac{w_0}{L_{B}}+N_{s0}^{B^+}L_{B^+}\operatorname{erf}\frac{w_0}{L_{B^+}}\right), \quad (8)$$

$$N_{A1}=\frac{\sqrt{\pi}}{2w_1}\left(N_{s0}^{Al}L_{Al}\operatorname{erf}\frac{x_{j1}}{L_{Al}}+N_{s0}^{B}\operatorname{erf}\frac{x_{j1}}{L_{B}}+N_{s0}^{B^+}L_{B^+}\operatorname{erf}\frac{x_{j1}}{L_{B^+}}\right)-\frac{w_0}{w_1}N_{A0}, \quad (9)$$

$$N_{A3}=\frac{\sqrt{\pi}}{2w_3}\left(N_{s4}^{Al}L_{Al}\operatorname{erf}\frac{x_{j2}}{L_{Al}}+N_{s4}^{B}L_{B}\operatorname{erf}\frac{x_{j2}}{L_{B}}\right)-\frac{w_4}{w_3}N_{A4}, \quad (10)$$

$$N_{A4}=\frac{\sqrt{\pi}}{2w_4}\left(N_{s4}^{Al}L_{Al}\operatorname{erf}\frac{x_{j3}}{L_{Al}}+N_{s4}^{B}L_{B}\operatorname{erf}\frac{x_{j3}}{L_{B}}\right), \quad (11)$$

$$N_{D0}=N_{D1}=N_{D2}, \quad (12)$$

$$N_{D3}=N_{s4}^{P}\left\{\left(1-\operatorname{erf}\frac{x_{j2}}{L_p}\right)+\frac{w_4}{w_3}\left(\operatorname{erf}\frac{x_{j3}}{L_p}-\operatorname{erf}\frac{x_{j2}}{L_p}\right)+\right.$$
$$\left.+\frac{L_p}{w_3\sqrt{\pi}}\left[\exp\left(-\frac{x_{j3}^2}{L_p^2}\right)-\exp\left(-\frac{x_{j2}^2}{L_p^2}\right)\right]\right\}+N_{D2}, \quad (13)$$

$$N_{D4}=N_{s4}^{P}\left\{1-\operatorname{erf}\frac{x_{j3}}{L_p}+\frac{L_p}{w_4\sqrt{\pi}}\left[1-\exp\left(-\frac{x_{j3}^2}{L_p^2}\right)\right]\right\}+N_{D2}. \quad (14)$$

Расчет подвижностей носителей проводится по формулам, приведенным в работе [7]. Коэффициенты диффузии носителей определяются, используя соотношение Эйнштейна $D_{n,p}=\varphi_T\mu_{n,p}$. Концентрации неосновных носителей в разных слоях структуры вычисляются из условия $np=n_i^2$, учитывая, что концентрация основных носителей равна концентрации примеси в данном слое.

В модели учитываются эффекты высокого легирования в эмиттерах тиристора по результатам работы [8]:

$$n_{ie}=n_{i0}\exp\left(\frac{\Delta V_{g0}}{\varphi_T}\right), \quad (15)$$

$$n_{i0}=3{,}88\cdot10^{20}T^{3/2}\exp(-7000/T), \quad (16)$$

$$\Delta V_{go} = 9 \cdot 10^{-3} \left[\ln(N/10^{17}) + \sqrt{[\ln(N/10^{17})]^2 + \frac{1}{2}} \right]. \qquad (17)$$

В конце модуля вычисляются коэффициенты электрического поля в p-базе и площади переходов, учитывая фаску структуры.

Выводы. Разработанная модель технологического процесса кремниевых СПП универсальна - она используется без основных преобразований структуры модуля для силовых диодов, транзисторов и тиристоров. В данной модели аналитически решаются уравнения диффузии, вычисляется толщина слоев структуры, определяются градиенты концентрации на переходах прибора и учитываются эффекты сильного легирования в эмиттерах тиристора. При проектировании кремниевых СПП результаты расчетов данной модели могут быть первым приближением определения технологического процесса.

Модель работает быстро и берет мало памяти ЭВМ. Время расчета и требование памяти ЭВМ модуля 15 секунд и 30 кбайт соответственно.

Литература

1. Antoniadis D.A., Dutton R.W. Models for computer simulation of complete IC fabrication process. - IEEE J. Sol. St. Circ., 1979, v. SC-14, N 2, p. 412-422.

2. Tarnay K., Masszi F., Mizsei J., Baji P., Rang T., Drozdy Gy., Kovacs B. Silicon planar technology process modelling. - Proc. of 3-rd Int. Spring Seminar on El. Techn. "Test methods in Electronics Technology", May 15-18, 1979, Balatonfüred, Hungary, p. 110-120.

3. Ранг Т.Х. Численное моделирование технологического процесса полупроводниковых приборов.-Тр. Таллинск. политехн. ин-та, 1982, № 538.

4. Зи С.М. Физика полупроводниковых приборов. М., Энергия, 1973. 656 с.

5. Irvin J.C. Resistivity of bulk silicon and of diffused layers in silicon. - Bell Syst. Techn. J., 1962, v. 41, p. 41-50.

6. S z e S.M.,,I r v i n J.C. Resistivity, mobility and impurity levels in GaAs, Ge and Si at 300 °K. - Sol. St. Electron, 1968, v. 11, p. 599-604.

7. Р а н г Т.Х., В е л м р е Э.Э. Зависимость подвижности электронов и дырок от температуры и концентрации примеси в кремнии . - Тр. Таллинск. политехн. ин-та, 1977, № 432, с. 115-120.

8. S l o t b o o m J.W. The pn product in silicon, - Sol. St. Electron, 1977, v. 20, p. 279-283.

T. Rang, E. Velmre

Computer Aided Modelling of Technological Process of Silicon Power Semiconductor Devices

Summary

Mathematical model of the technological process of silicon power semiconductor devices is presented. In the model the high concentration effects in emitters of the devices have been taken into account. The diffusion problems have been solved analytically. Also the thicknesses of layers, mobilities and concentrations of minority carriers have been found analytically. The model is a part of the model of analyse program of power thyristors.

Цена 90 коп.