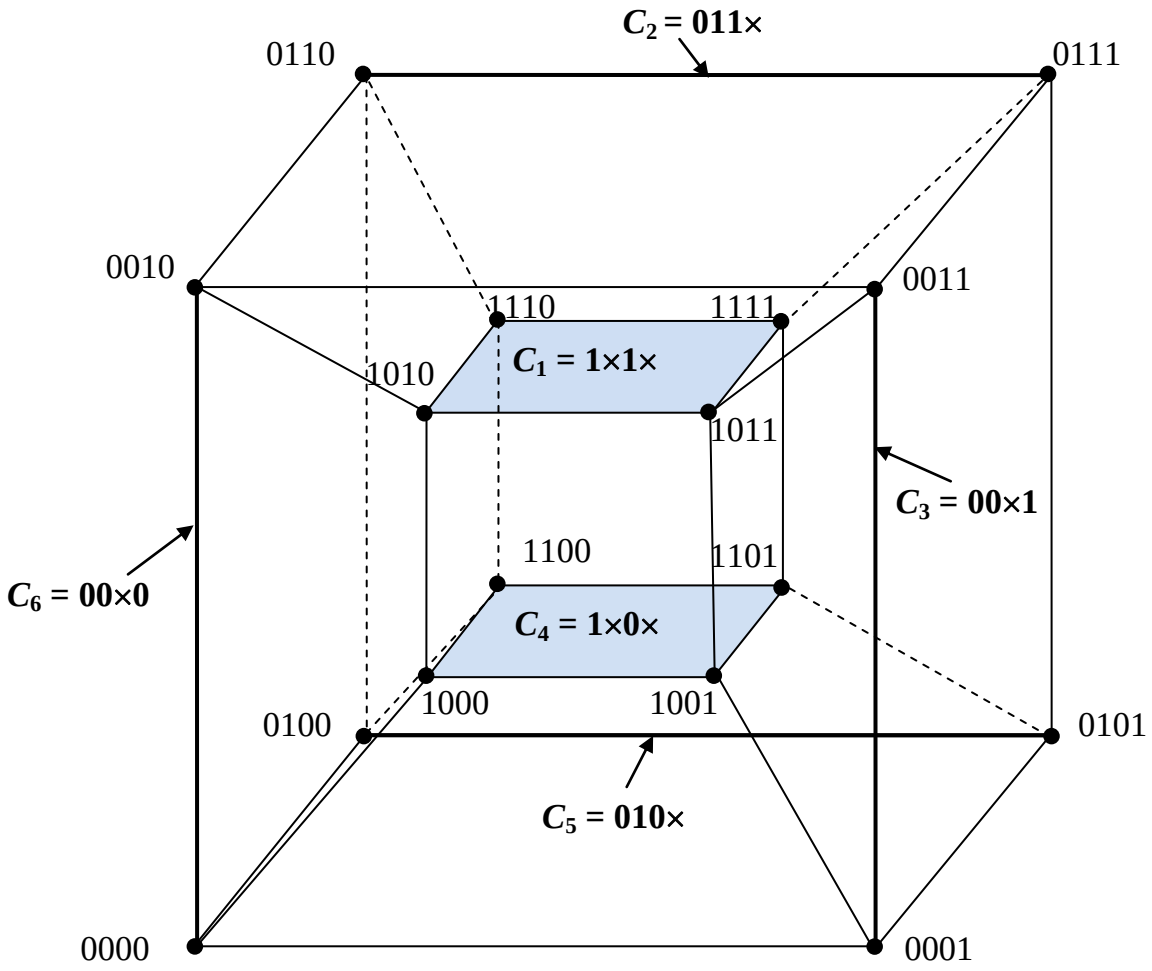
УНИВЕРСИТЕТ ИТМО

Ю.В. Донецкая, А.Г. Зыков, В.И. Поляков

МЕТОДЫ ВЕРИФИКАЦИИ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ



Санкт-Петербург

2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

Ю.В. Донецкая, А.Г. Зыков, В.И. Поляков

МЕТОДЫ ВЕРИФИКАЦИИ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ
РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ ИТМО
по направлениям подготовки 09.03.01 «Информатика и вычислительная техника», 09.03.04 «Программная инженерия», 10.03.01 «Информационная безопасность» в качестве учебно-методического пособия для реализации основных профессиональных образовательных программ высшего образования бакалавриата

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург

2019

Донецкая Ю.В., Зыков А.Г., Поляков В.И. Методы верификации вычислительных процессов. Часть 1. – СПб: Университет ИТМО, 2019. – 142 с.

Рецензент: к.т.н. Балакшин П.В., доцент факультета ПИ и КТ Университета ИТМО

Учебное пособие охватывает различные аспекты верификации вычислительных процессов, реализованных аппаратно, на уровне логических схем (часть 1) и программно (часть 2). В данной части приведено обоснование важности верификации проектов, даны определения формальной и функциональной верификаций вычислительных процессов, их валидации. Представлены теоретические основы методов верификации, таких, как основные положения теории множеств, булевой алгебры, теории графов и теории конечных автоматов. Подробно рассмотрены методы синтеза контролирующих тестов для функциональной верификации цифровых схем, включая формальные модели схем, построение комплексных покрытий и графов переходов, решение установочной задачи, синтез тестовых последовательностей и устранение статического риска сбоя. Рассмотрен вопрос формальной верификации моделей разного уровня представления.

Для закрепления теоретического материала предложены контрольные вопросы. Приведёны пример выполнения домашнего задания на синтез теста для последовательностной схемы и варианты заданий.

Работа выполнена при частичной поддержке РФФИ, грант 17-07-00700.

Учебное пособие предназначено для бакалавров широкого круга инженерных специальностей (проектирование вычислительных систем, программирование и др.), а также оно будет полезно для магистрантов, специализирующихся в области вычислительной техники.

УНИВЕРСИТЕТ ИТМО

Университет ИТМО – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

Содержание

Предисловие

В современных компьютерных технологиях и не только в этом направлении, обязательным элементом является верификация на каждом этапе проектирования, производства и эксплуатации соответствующего продукта.

В основе любого устройства преобразования информации лежит вычислительный процесс, инвариантный относительно способа физической реализации, в нашем случае в виде логической схемы или программы, подвидами которых могут выступать программируемая логика или микропрограммы.

Основная цель настоящего учебного пособия – помочь студенту приобрести теоретические знания и практические навыки в решении задач бездефектного проектирования и верификации вычислительных процессов (ВП). Рассматриваемый в пособии теоретический материал сопровождается большим количеством примеров, что делает излагаемый материал более понятным и упрощает выполнение домашнего задания.

Для работы с пособием необходимо изучение теоретического материала, приведенного во втором разделе.

В третьем разделе подробно рассмотрены методы синтеза контролирующих тестов для функциональной верификации цифровых схем, включая формальные модели схем, построение комплексных покрытий и графов переходов, решение установочной задачи, синтез тестовых последовательностей и устранение статического риска сбоя. В четвертом разделе приводятся варианты домашнего задания и подробный пример его выполнения. В пятом разделе рассмотрен вопрос формальной верификации моделей разного уровня представления.

В результате освоения материалов пособия обучающийся приобретает следующие умения и навыки:

- уметь формулировать основные понятия и этапы верификации вычислительного процесса;
- выбирать и использовать методы верификации для решения задач анализа ВП;
- оценивать проблемные ситуации, возникающие при синтезе тестовых последовательностей;
- уметь управлять жизненным циклом этапов верификации проекта;
- уметь верифицировать модели цифровых схем разного уровня иерархии;
- владеть базовыми методами верификации ВП;
- владеть навыками синтеза и анализа цифровых схем;
- владеть навыками разработки тестов для функциональной верификации цифровых схем.

Для закрепления пройденного материала и получения навыков, студенту следует проработать контрольные вопросы и предлагаемые задания для самостоятельной работы. Успешно выполнив задания, студент получает указанные выше навыки, знания и умения.

Большой вклад в развитие методов технической диагностики, алгебро-топологических методов анализа и синтеза логических структур, верификации проектов внёс доктор технических наук, профессор Олег Фомич Немолочнов, памяти которого и посвящено данное пособие.

ВВЕДЕНИЕ

Проблема анализа качества аппаратного и программного обеспечения становится сегодня все более острой, особенно по мере расширения использования нанотехнологий в приборостроении и информационных технологий при разработке программного обеспечения. Экспоненциальный рост сложности аппаратного и программного обеспечения вычислительных процессов порождает повышенные требования к бездефектному проектированию. Известны примеры, как дорого обходятся ошибки, допущенные на различных этапах проектирования, поэтому все современные системы автоматизированного проектирования (САПР) обязательно снабжаются методологическими, программными и инструментальными средствами анализа разрабатываемого изделия на всех этапах автоматизированного проектирования. Не менее актуальными являются проблемы, связанные с обеспечением проектирования надежных программ. Данное учебное пособие посвящено рассмотрению методов верификации вычислительных процессов, являющихся составной частью искусственного интеллекта. Основой любого вычислительного устройства (ВУ) – вычислительной машины (ЭВМ), реализующей дискретный способ преобразования информации, служит вычислительный процесс, порождаемый процессором при выполнении команд программы. С этой позиции ВУ – ЭВМ необходимо рассматривать как единый аппаратно-программный комплекс. Действительно, программа без процессора есть некоторый набор абстрактных высказываний и предложений, а процессор без программы – набор электронных компонентов. В тот момент, когда программа исполняется процессором и рождается вычислительный процесс. Вычислительный процесс и является целью проектирования вычислительного устройства любой степени сложности.

Д. Кнут дает следующее определение алгоритма:

«Алгоритм — это конечный набор правил, который определяет последовательность операций для решения конкретного множества задач и обладает пятью важными чертами: конечность, определённость, ввод, вывод, эффективность». (*Кнут Д. Э.* Искусство программирования. Том 1. Основные алгоритмы. 3 издание — Москва: Вильямс, 2017. — Т. 1. — 720 с.)

Свойства алгоритма:

- Конечность – алгоритм должен заканчиваться после выполнения конечного числа шагов.
- Определенность – каждый шаг алгоритма должен быть точно определен.
- Наличие входных данных. Алгоритм имеет некоторое число входных данных, задающихся до начала работы или определяющихся динамически во время его выполнения.

- Наличие выходных данных. Алгоритм имеет одно или несколько выходных данных, имеющих определенную связь с входными данными.
- Эффективность – алгоритм считается эффективным, если все его операторы достаточно просты для того, чтобы их можно было точно выполнить в течение конечного промежутка времени.

Вычислительный процесс может быть реализован программно или аппаратно. В случае ***аппаратной реализации*** преобразование входных данных (в виде булевых переменных) в выходные сигналы осуществляется логической схемой в соответствии с алгоритмом вычисления булевой функции.

В случае ***программной реализации*** преобразование входных данных в выходные осуществляется путем вычисления по программе, записанной в память.

Как аппаратная, так и программная реализации управления имеют преимущества и недостатки. Аппаратная реализация обладает более высоким быстродействием. Недостатком является отсутствие гибкости системы, так как переход к другому алгоритму потребует изменений в электрической схеме устройства управления.

Преимущество программной реализации – гибкость устройства управления, выражающаяся в возможности изменять алгоритм управления программным путем, не изменяя электрическую схему устройства. Недостаток – низкое быстродействие, поскольку время обработки информации зависит от времени выполнения команд программы.

1. ВЕРИФИКАЦИЯ

1.1. Место верификации при проектировании вычислительных систем

Согласно закона Э. Мерфи, если какая-то неприятность может случиться, то она случается. Человек может ошибиться, поэтому он обязательно ошибается. Ошибки случаются при проектировании аппаратуры, при написании программ, при изготовлении аппаратуры и ее эксплуатации. Иногда ошибки обходятся очень дорого.

В 1994 году профессор математики из Вирджинии Томас Найсли при вычислении обратных величин простых чисел обнаружил, что микропроцессор Pentium в некоторых случаях неправильно делит числа с плавающей запятой. Через месяц фирма Intel согласилась заменить микропроцессоры с неправильно спроектированным устройством деления, что обошлось ей в 300 миллионов долларов (по некоторым источникам, итоговые потери корпорации составили 500 миллионов долларов). Этот случай нанес большой ущерб репутации фирмы.

В 2003 году в северо-восточных штатах США произошла крупнейшая в истории Америки авария в системе энергоснабжения. Без электричества остались 50 миллионов человек, экономические потери составили 6 миллиардов долларов. Одной из главных причин оказалась ошибка в программе, которая не смогла правильно обработать стечение событий. Чтобы исправить эту ошибку, пришлось просмотреть несколько миллионов строк программы.

В военной области ошибки часто приводят к трагическим последствиям. В системе управления торпедой был предусмотрен ее автоматический подрыв при отклонении от курса более чем на 90°, для того чтобы торпеда в случае неисправности не поразила корабль, с которого она пущена. Однажды при пуске торпеды она застряла в торпедном аппарате, командир корабля решил вернуться в порт, чтобы устранить неисправность на базе. Он приказал повернуть на 180°… Понятно, к чему это привело.

В 1983 году мир был на грани ядерной войны. Из-за ошибки советской системы раннего предупреждения блики солнца на облаках воспринялись как пуск американских ракет. На пульт системы поступило сообщение о запуске пяти американских ракет. К счастью, дежурный офицер усомнился в правильности работы системы, он понимал, что первый ядерный удар не может наноситься таким небольшим количеством ракет. Похожая история произошла несколькими годами ранее с американской системой ПВО. Она восприняла Луну как приближающиеся советские ракеты.

В 1991 году во время Войны в заливе (США против Ирака) батарея американских зенитных ракет «Patriot» не смогла перехватить запущенную иракцами ракету «Scud» советского производства. Ракета попала в казарму американских солдат, при этом погибло 28 человек. Причиной этого была погрешность вычисления времени. При вычислении нужно было умножать время, задаваемое тактовым генератором компьютера (оно измерялось в десятых долях секунды), на 1/10, но это десятичное число невозможно точно

представить в двоичном виде. Для хранения этой константы использовался 24-разрядный регистр. Разница между точным значением 1/10 и ее неточным двоичным представлением составляет в двоичном виде 0,00000000000000000000000011001100…, или около 0,000000095 в десятичном виде. Компьютер был включен около 100 часов, за это время накопилась ошибка в измерении времени в 0,34 секунды. Скорость ракеты «Scud» составляла примерно 1700 м/сек, и за это время она прошла более 500 метров. Этого хватило для того, чтобы зенитные ракеты не смогли ее перехватить.

По некоторым данным, на 1000 строк обычной программы в среднем приходится 25 ошибок. В хорошей программе — 2 ошибки на 1000 строк. В программном обеспечении американского космического Шаттла не более одной ошибки на 10000 строк. Для космической отрасли вся аппаратура и программное обеспечение проверяются особенно тщательно, так как от их качества и надежности зависит человеческая жизнь. Тем не менее, во время 10-дневного полета американского космического корабля Аполло-14 было обнаружено 18 ошибок.

Эти примеры показывают важность верификации проектов вычислительных систем.

1.2. Основные термины и определения

Слово *верификация* произошло от латинских *verus* — истинный и *facere* — делать. Вообще, верификация означает подтверждение того, что описание проекта полностью соответствует спецификации (техническому заданию) проектируемой системы. Спецификация — документ, подробно перечисляющий условия, которым должна соответствовать проектируемая или изготовляемая система.

Верификация определяется как разновидность анализа, имеющая целью установление соответствия двух описаний одного и того же объекта. Различают верификацию структурную и функциональную. При структурной верификации устанавливается соответствие структур, отображаемых двумя описаниями, при функциональной (параметрической) - проверяется соответствие процессов функционирования и выходных параметров, отображаемых сравниваемыми описаниями. Структурная верификация связана с меньшими затратами вычислительных ресурсов, чем функциональная. Поэтому последняя часто выполняется не в полном объеме и после того, как проверено соответствие структурных свойств. Примером структурной верификации служит установление соответствия системы электрических межсоединений на печатной плате и в принципиальной электрической схеме, заданных своими топологическими моделями в виде графов. Верификация в этом случае сводится к установлению изоморфизма графов. Функциональная верификация в данном примере выполняется путем анализа переходных процессов с учетом перекрестных помех и задержек

сигналов в длинных линиях, определяемых конструктивным исполнением электронного блока.

Согласно Модели зрелости процесса разработки (Capability Maturity Model Integration) (*Денис М. Ахен, Арон Клауз, Ричард Тернер*. CMMI: Комплексный подход к совершенствованию процессов. Практическое введение в модель. — М: «МФК», 2005, 300 с.), "верификация" определяется следующим образом: Верификация обеспечивает соответствие набора результатов работы требованиям, которые для них заданы.

Верификация в самом общем виде предполагает установление соответствия между различными объектами. Сами объекты могут задаваться математическими моделями одного уровня либо моделями разного уровня представления.

На рис. 1.1 представлена общая схема верификации. Эта схема состоит из нескольких этапов. На первом этапе требования извлекаются из нормативных документов и систематизируются. В результате получается *каталог требований*, в котором требования сформулированы максимально однозначно, требования классифицированы, и, возможно, установлены связи между отдельными требованиями. Каталог требований используется на последующих этапах.

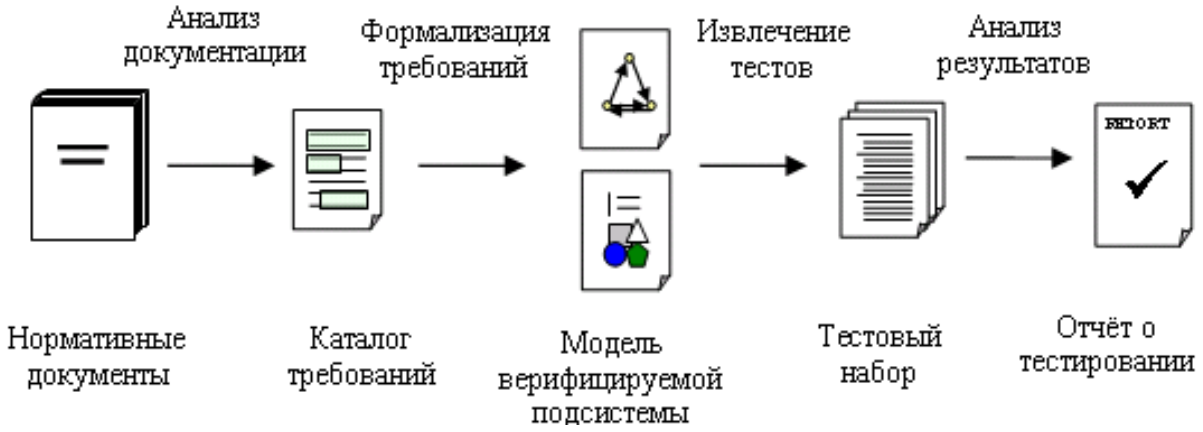


Рис. 1.1. Общая схема верификации

Второй этап нацелен на представление требований в формальном виде. Требования из каталога записываются с использованием того или иного математического формализма. Такая запись требований называется формальной спецификацией или формальной моделью.

На третьем этапе, на основе построенной модели, автоматизированным образом генерируются тесты. В зависимости от задачи, тесты могут либо быть просто тестовыми данными, либо дополнительно содержать оракул для автоматического вынесения вердикта о корректности наблюдаемого поведения объекта.

В результате исполнения тестов строятся отчёты о тестировании. В них содержится информация о том, насколько наблюдаемое поведение объекта соответствует поведению формальной модели.

Введем некоторые понятия и определения, связанные с процессом тестирования, как составной части верификации. Эти определения распространяются на тестирование как аппаратуры, так и программ. Гленфорд Майерс (Гленфорд Майерс, Том Баджетт, Кори Сандлер. Искусство тестирования программ. 3-е издание. Изд. Диалектика, 2012. 272 с.) дает следующие определения основных терминов:

Тестирование - процесс выполнения программы с целью обнаружения ошибки.

Тестовые данные – входы, которые используются для проверки системы.

Тестовая ситуация (*test case*) – входы для проверки системы и предполагаемые выходы в зависимости от входов, если система работает в соответствии с ее спецификацией требований.

Хорошая тестовая ситуация – та ситуация, которая обладает большой вероятностью обнаружения пока еще необнаруженной ошибки.

Удачный тест - тест, который обнаруживает пока еще необнаруженную ошибку.

Ошибка - действие программиста на этапе разработки, которое приводит к тому, что в программном обеспечении содержится внутренний дефект, который в процессе работы программы может привести к неправильному результату.

Отказ – непредсказуемое поведение системы, приводящее к неожидаемому результату, которое могло быть вызвано дефектами, содержащимися в ней.

Таким образом, в процессе тестирования системы, как правило, выполняют следующие проверки:

- проверка того, что исследуемая система соответствует требованиям на нее;
- проверка того, что в ситуациях, не отраженных в требованиях, система ведет себя адекватно, то есть не происходит отказ системы;
- проверка объекта тестирования на предмет типичных ошибок, которые делают разработчики.

На рис. 1.2. показана схема тестирования, верификации и валидации. Несмотря на кажущуюся схожесть, термины «тестирование», «верификация» и «валидация» означают разные уровни проверки корректности работы вычислительной системы. Чтобы избежать дальнейшей путаницы, четко определим эти понятия.

Тестирование – вид деятельности в процессе разработки, связанный с выполнением процедур, направленных на обнаружение (доказательство наличия) ошибок (несоответствий, неполноты, двусмысленностей и т.д.) в текущем определении разрабатываемой системы. Процесс тестирования относится в первую очередь к проверке корректности программной или аппаратной реализации системы, соответствия реализации требованиям, т.е.

тестирование – это управляемое функционирование объекта тестирования с целью обнаружения несоответствий его поведения и требований.

Верификация – более общее понятие, чем тестирование. Целью верификации является достижение гарантии того, что верифицируемый объект (требования или программный код) соответствует требованиям, реализован без непредусмотренных функций и удовлетворяет проектным спецификациям и стандартам. Процесс верификации включает в себя инспекции, тестирование, анализ результатов тестирования, формирование и анализ отчетов о проблемах. Таким образом, принято считать, что процесс тестирования является составной частью процесса верификации, такое же допущение сделано и в данной работе.

Валидация системы (от лат. validus «здоровый, крепкий; сильный») – процесс, целью которого является доказательство того, что в результате разработки системы мы достигли тех целей, которые планировали достичь благодаря ее использованию. Иными словами, валидация – это проверка соответствия системы ожиданиям заказчика. Вопросы, связанные с валидацией, выходят за рамки данного пособия и представляют собой отдельную интересную тему для изучения.

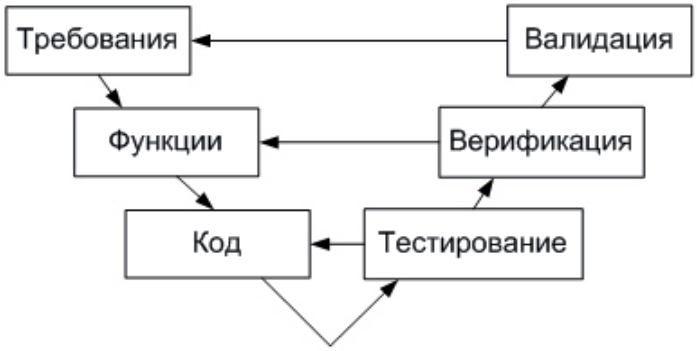


Рис. 1.2. Тестирование, верификация и валидация

Если посмотреть на эти три процесса с точки зрения вопроса, на который они дают ответ, то тестирование отвечает на вопрос «Как это сделано?» или «Соответствует ли поведение разработанной системы требованиям?», верификация – «Что сделано?» или «Соответствует ли разработанная система требованиям?», а валидация – «Сделано ли то, что нужно?» или «Соответствует ли разработанная система ожиданиям заказчика?».

На рис. 1.3. приведена схема взаимосвязи между верификацией и валидацией.

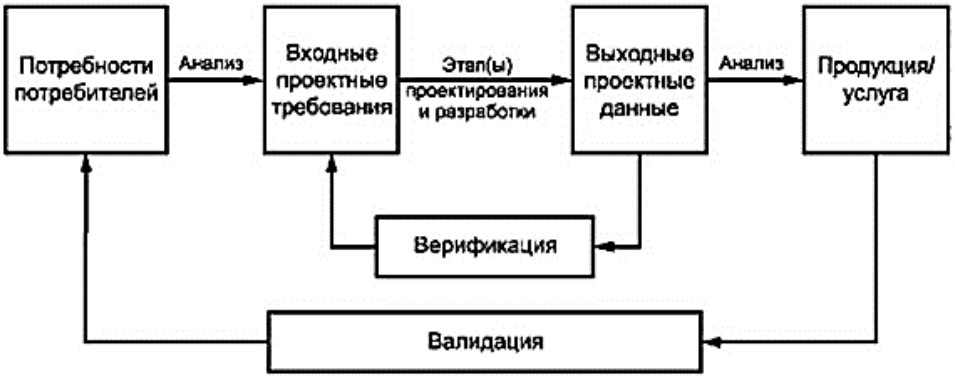


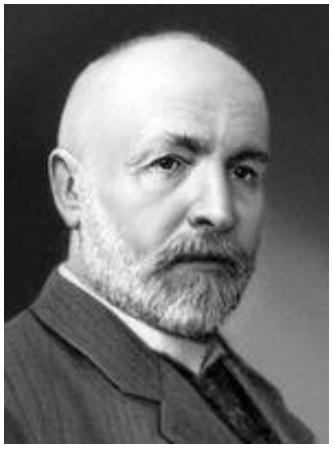
Рис. 1.3. Упрощенная схема взаимосвязи между верификацией и валидацией в процессе проектирования и разработки

2. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ВЕРИФИКАЦИИ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

2.1. Теория множеств. Основные понятия

Вряд ли можно назвать какую-либо возникшую в последней трети девятнадцатого века математическую дисциплину, которая оказала бы большее влияние на прогресс всей математики и, шире, на математическое мышление в целом, чем теория множеств. К идеям теории множеств в разное время подходили с разных сторон многие ученые, но оформление ее в самостоятельную науку, со своими особыми предметом и методом исследования, осуществил в своих работах 1872-1897 г.г. Георг Кантор. Среди современников Кантора правильно оценили значение этих работ только немногие, прежде всего Рихард Дедекинд, который внес собственный значительный вклад в новую теорию. Обнаруженные в конце Х1Х — начале ХХ вв. логические и методологические парадоксы теории множеств отпугнули некоторых выдающихся математиков, первоначально приветствовавших ее появление, как, например, Анри Пуанкаре. Однако плодотворные приложения теории множеств в различных областях анализа стимулировали ее дальнейшую разработку во многих направлениях, в том числе глубокое исследование самых ее основ средствами бурно расцветавшей математической логики. Какие-либо окончательные и общепризнанные решения всех трудностей не достигнуты, и все более и более тонкие изыскания здесь продолжаются; вместе с тем современная математика не может обойтись без основного аппарата понятий и приемов теории множеств.

Кантору принадлежит заслуга привнесения в математику самого понятия "множества" (или "совокупности"). Это понятие относится к категории изначальных и неопределяемых. Его можно лишь толковать и иллюстрировать на примерах. "Под множеством - писал Кантор - я понимаю вообще всякое многое, мыслимое нами как единое, т.е. всякую совокупность определенных элементов, которая может быть связана в одно целое с помощью некоторого закона..." (Кантор Г. Труды по теории множеств. - М.: Наука, 1985. - С. 101).



Другая формулировка Г. Кантора: "Под «множеством» мы понимаем соединение в некое целое *M* определённых хорошо различимых предметов *m* нашего созерцания или нашего мышления (которые будут называться «элементами» множества *M*)." Кантору принадлежит также следующая характеристика понятия «множество»: *Множество — это объединение определённых, различных объектов, называемых элементами множества, в единое целое.*

Кантор Георг (19.02.1845 - 06.01.1918) – немецкий математик.

В основе теории множеств лежат первичные понятия: ***множество*** и отношение ***быть элементом*** множества (обозначается как $x \in A$ — «x есть элемент множества A». В свою очередь непринадлежность некоторого элемента a множеству M принято обозначать: $a \notin M$ или $a \ni M$).

Среди производных понятий наиболее важны следующие:

- ***Пустое множество*. Пустым множеством** называется множество, не содержащее ни одного элемента. Обычно пустое множество обозначают символом $\varnothing$.

- ***Подмножество*** и **надмножество**. Множество A является ***подмножеством*** множества B, если любой элемент, принадлежащий A, также принадлежит B. Пишут: $A \subset B$ или $A \subseteq B$. Таким образом, $(A \subset B) \Leftrightarrow (x \in A \Rightarrow x \in B)$. Множество B в таком случае называется ***надмножеством*** множества A, и этот факт часто записывают: $B \supset A$ или $B \supseteq A$. По определению полагают, что пустое множество является подмножеством любого множества: $\forall A: \varnothing \subset A$.

- ***Универсальное множество*** (***универсум***) — множество, содержащее все мыслимые объекты. Универсальное множество единственно. Универсальное множество обычно обозначается U (от англ. *universe, universal set*), реже E.

- ***Абсолютным дополнением множества A*** называется множество всех тех элементов x универсального множества, которые не принадлежат множеству A. Абсолютное дополнение множества A обозначается $\overline{A}$.

- Пусть A — множество. Множество всех подмножеств множества A называется **б*улеаном A*** (также ***степенью множества***) и обозначается $P(A)$. Ясно, что $\varnothing \in P(A)$ и $A \in P(A)$. Справедливо следующее утверждение: число подмножеств конечного множества, состоящего из n элементов, равно 2^n.

- ***Конечное множество*** состоит из конечного числа элементов, например, множество страниц в книге, множество учеников в школе и т.д.

- ***Бесконечное множество*** состоит из бесконечного числа элементов, т.е. это множество, которое не является ни конечным, ни пустым. Примеры: множество действительных чисел, множество точек плоскости, множество атомов во Вселенной и т.д.

- ***Счётное множество*** – множество, элементы которого можно пронумеровать. Например, множества натуральных, чётных, нечётных чисел. Счётное множество может быть конечным (множество книг в библиотеке) или бесконечным (множество целых чисел).

- ***Несчётное множество*** – множество, элементы которого невозможно пронумеровать. Например, множество действительных чисел. Несчётное множество может быть только бесконечным.

2.1.1. Способы задания множеств

Множества могут быть заданы списком, порождающей процедурой, арифметическими операциями, описанием свойств элементов или графическим представлением.

1. Задание множеств ***списком*** предполагает перечисление элементов. Например, множество A состоит из букв a, b, c, d: $A = \{a, b, c, d\}$ или множество N включает цифры $0, 2, 3, 4$: $N = \{0, 2, 3, 4\}$.

2. Задание множеств ***порождающей процедурой*** или арифметическими операциями означает описание характеристических свойств элементов множества: $X = \{x \mid H(x)\}$, т. е. множество X содержит такие элементы x, которые обладают свойством $H(x)$.

Например:

$B = \{b \mid b = \pi/2 \pm k\,\pi, k \in N\}$, N - множество всех натуральных чисел;

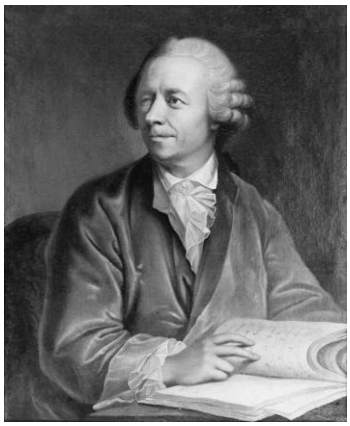
$C = A+B = \{x \mid x = a + b, a \in A, b \in B\}$.

3. Задание множества ***описанием свойств*** элементов. Например, M - это множество чисел, являющихся степенями двойки.

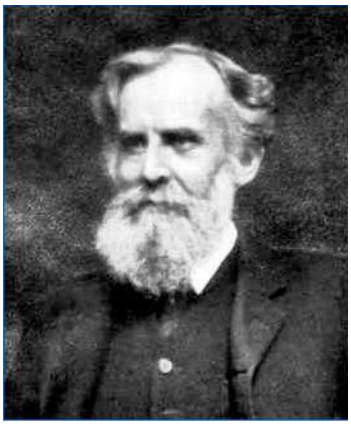
К описанию свойств естественно предъявить требования точности и недвусмысленности. Так, "множество всех хороших песен 2019 года" каждый составит по-разному. Надежным способом однозначного задания множества является использование разрешающей процедуры, которая для любого объекта устанавливает, обладает ли он данным свойством и соответственно является ли элементом рассматриваемого множества.

Например, S - множество успевающих студентов. Разрешающей процедурой включения во множество S является отсутствие неудовлетворительных оценок в последней сессии.

4. ***Графическое*** задание множеств проводят с помощью диаграмм Эйлера-Венна.



Леонард Эйлер (15.04.1707 - 07.09.1783) – швейцарский математик, механик, физик и астроном.



Джон Венн (04.08.1834 - 04.04.1923) – английский математик и логик.

Построение диаграммы заключается в изображении большого прямоугольника, представляющего универсальное множество U, а внутри его – кругов (или каких-нибудь других замкнутых фигур), представляющих

множества. Фигуры должны пересекаться в наиболее общем случае, требуемом в задаче, и должны быть соответствующим образом обозначены.

Точки, лежащие внутри различных областей диаграммы, могут рассматриваться как элементы соответствующих множеств. Имея построенную диаграмму, можно заштриховать определенные области для обозначения вновь образованных множеств. На рисунке 2.1 на диаграмме Эйлера-Венна показаны универсальное множество ***U***, множества ***A*** и ***B.***

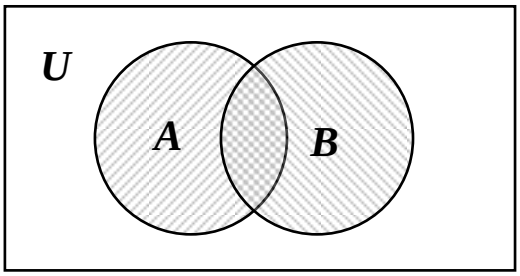
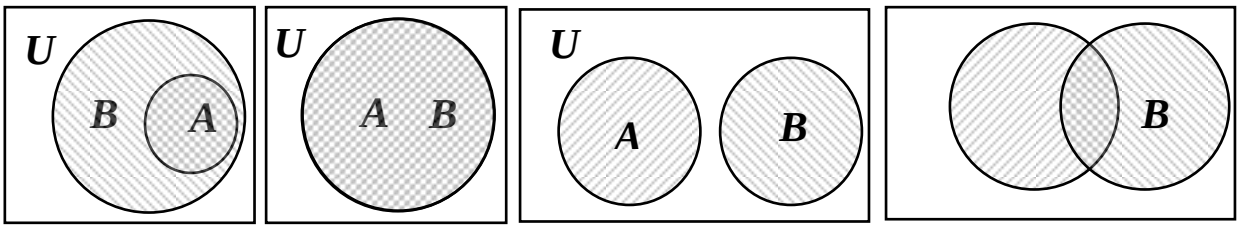


Рис. 2.1. Пример диаграммы Эйлера-Венна

Два множества ***A*** и ***B*** могут вступать друг с другом в следующие отношения:

- множество ***A включено*** в ***B***, если каждый элемент множества ***A*** принадлежит также и множеству ***B*** (рис. 2.2*а*): $A \subseteq B \Leftrightarrow \forall a \in A : a \in B$;
- множество ***A строго включено*** в ***B***, если ***A*** включено в ***B***, но не равно ему (рис. 2.2 *а*): $A \subset B \Leftrightarrow (A \subseteq B) \wedge (A \neq B)$.

В этом случае множество ***A*** называют ***собственным*** (строгим, истинным) подмножеством множества ***B***;



а *б* *в* *г*

Рис. 2.2. Возможные отношения множеств ***A*** и ***B***

- множество ***A равно*** множеству ***B***, если ***A*** и ***B*** включены друг в друга (рис. 2.2 *б*): $A = B \Leftrightarrow (A \subseteq B) \wedge (B \subseteq A)$.

Вторая часть определения указывает на наиболее типичный метод доказательства того, что данные множества равны, заключающийся в доказательстве сначала утверждения $A \subseteq B$ (“в одну сторону”), а затем $B \subseteq A$ (“в другую сторону”);

- множества ***A*** и ***B не пересекаются***, если у них нет общих элементов (рис. 2.2 *в*): ***A*** и ***B*** не пересекаются $\Leftrightarrow a \in A : a \notin B$;
- множества ***A*** и ***B находятся в общем положении***, если существуют: элемент, принадлежащий исключительно множеству ***A***; элемент, принадлежащий исключительно множеству ***B***; а также элемент,

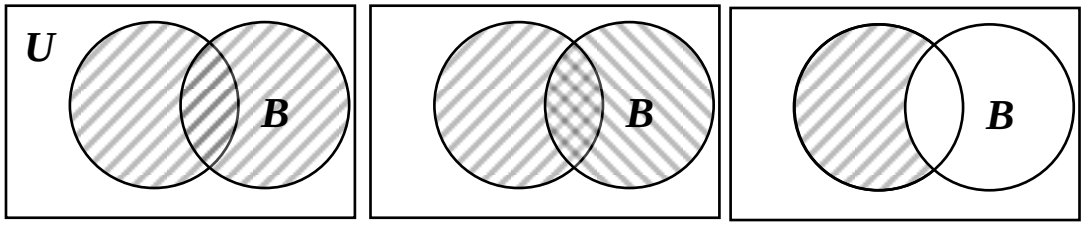
принадлежащий обоим множествам (рис. 2.2 *г*): A и B находятся в общем положении $\Leftrightarrow \exists a, b, c: (a \in A) \wedge (a \notin B) \wedge (b \in B) \wedge (b \notin A) \wedge (c \in A) \wedge (c \in B)$.

Отношение подмножества обладает целым рядом свойств:

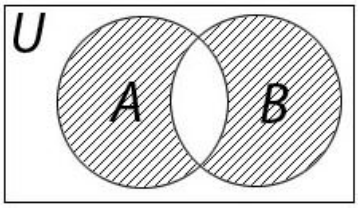
- Отношение подмножества ***рефлексивно***: $B \subset B$;
- Отношение подмножества ***антисимметрично***: $(A \subset B \wedge B \subset A) \Leftrightarrow (A=B)$;
- Отношение подмножества ***транзитивно***: $(A \subset B \wedge B \subset C) \Rightarrow (A \subset C)$.

Над множествами определены следующие *операции*:

Объединением множеств A и B называется множество, состоящее из всех тех элементов, которые принадлежат хотя бы одному из множеств A, B (рис. 2.3 *а*): $A \cup B = \{x \mid x \in A$ или $x \in B\}$.



а *б* *в*



г

Рис. 2.3. Операции над множествами (*а* – объединение, *б* – пересечение, *в* – разность, *г* – симметрическая разность).

Операцию объединения можно распространить на произвольное, в том числе и бесконечное количество множеств, например $M=A \cup B \cup C \cup D$.

В общем случае используется обозначение $\bigcup_{A \in S} A$, которое читается так: “объединение всех множеств A, принадлежащих совокупности S”.

Если же все множества совокупности индексированы (занумерованы с помощью индексов), то используются другие варианты обозначений:

1. $\bigcup_{i=1}^{k} A_i$, если $S=\{A_1, A_2, \ldots, A_k\}$;
2. $\bigcup_{i=1}^{\infty} A_i$, если S – бесконечная совокупность занумерованных множеств;
3. $\bigcup_{i=I} A_i$, если набор индексов множеств задан множеством I.

Примеры

$A=\{a, b, c\}, B=\{b, c, d\}, C=\{c, d, e\}$.

$A \cup B=\{a, b, c, d\}; A \cup C=\{a, b, c, d, e\}; B \cup C=\{b, c, d, e\}; A \cup B \cup C=\{a, b, c, d, e\}$.

Пересечением множеств ***A*** и ***B*** называется множество, состоящее из всех тех и только тех элементов, которые принадлежат одновременно как множеству ***A***, так и множеству ***B*** (рис. 2.3 *б*): $A \cap B = \{x \mid x \in A \text{ и } x \in B\}$.

Аналогично определяется пересечение произвольной (в том числе бесконечной) совокупности множеств. Обозначения для пересечения системы множеств аналогичны рассмотренным ранее обозначениям для объединения.

Пример (для множеств из предыдущего примера):
$A \cap B = \{b, c\}$; $A \cap C = \{c\}$; $B \cap C = \{c, d\}$; $A \cap B \cap C = \{c\}$.

Разностью множеств ***A*** и ***B*** называется множество всех тех и только тех элементов ***A***, которые не содержатся в ***B*** (рис. 2.3 *в*): $A \setminus B = \{x \mid x \in A \text{ и } x \notin B\}$.

Иначе разность множеств ***A*** и ***B*** называется дополнением множества ***B*** до множества ***A*** (относительным дополнением).

Пример (для тех же множеств)
$A \setminus B = \{a, b, c\} \setminus \{b, c, d\} = \{a\}$.

Симметрической разностью множеств ***A*** и ***B*** называется множество элементов этих множеств, которые принадлежат либо только множеству ***A***, либо только множеству ***B*** (рис. 2.3 *г*). Симметрическую разность обозначают как $A \Delta B$ или $A - B$: $A \Delta B = (A \setminus B) \cup (B \setminus A)$.

Пример
$A \Delta B = \{a\} \cup \{d\} = \{a, d\}$.

С учетом введенных операций ***абсолютное дополнение*** множества ***A*** записывается:

$\overline{A} = \{x \mid x \notin A\} = U \setminus A$, а разность множеств ***A*** и ***B***: $A \setminus B = A \cap \overline{B}$.

Старшинство операций (операции даны по убыванию приоритетов):

$\overline{A}$, $\setminus$, $\cap$, $\cup$, Δ.

Операции над множествами используются для получения новых множеств из уже существующих.

Обозначения наиболее часто используемых множеств следующие:

N – множество всех натуральных чисел (т. е. $N = \{1, 2, 3, \ldots\}$);

Z – множество всех целых чисел;

$Z+$ – множество целых неотрицательных чисел ($Z+ = N \cup \{0\}$, иногда обозначают N_0);

$Z-$ – множество целых неположительных чисел ($Z- = Z \setminus N$);

U – множество всех рациональных чисел;

V – множество всех иррациональных чисел;

R – множество всех действительных чисел;

$R+$ – множество неотрицательных действительных чисел;

$R-$ – множество неположительных действительных чисел.

2.1.2. Основные тождества (законы) алгебры множеств

Тождества алгебры множеств обладают свойством ***дуальности*** (двойственности).

1). Коммутативные: $A \cup B = B \cup A$; $A \cap B = B \cap A$.

2). Ассоциативные: $(A \cup B) \cup C = A \cup (B \cup C)$; $(A \cap B) \cap C = A \cap (B \cap C)$.

3). Дистрибутивные: $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$; $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

4). Законы тавтологии (идемпотентности): $A \cup A = A$; $A \cap A = A$.

5). Законы де Моргана (двойственности):

$\overline{A \cup B} = \overline{A} \cap \overline{B}, \quad \overline{A \cap B} = \overline{A} \cup \overline{B}.$

6). Законы поглощения: $A \cup (A \cap B) = A$; $A \cap (A \cup B) = A$.

7). Закон равенства:

$A = B \Leftrightarrow ((A \subset B) \wedge (B \subset A) \Leftrightarrow (A \cap B) \vee (\overline{A} \cap \overline{B}))$.

8) Закон противоречия: $A \cap \overline{A} = \varnothing$.

9) Закон «третьего не дано»: $A \cup \overline{A} = U$.

10) Закон двойного отрицания: $\overline{\overline{A}} = A$.

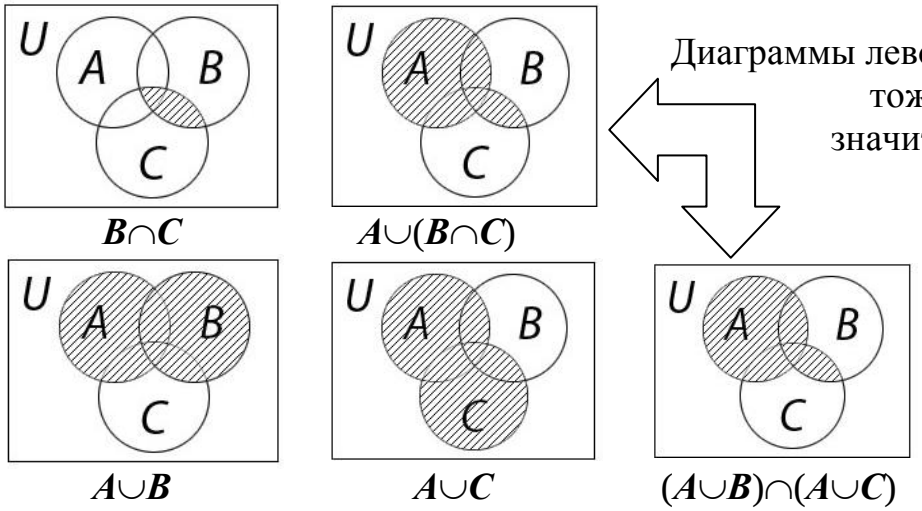
11) Свойства универсального множества: $A \cup U = U$; $A \cap U = A$.

12) Свойства пустого множества: $A \cup \varnothing = A$; $A \cap \varnothing = \varnothing$.

Огастес де Морган, (27.06.1806 – 18.03.1871) шотландский математик и логик.

Убедиться в справедливости тождеств можно с помощью диаграмм Эйлера-Венна. Для этого необходимо изобразить на диаграммах левую и правую части тождеств и сравнить их.

В качестве примера проверим первый дистрибутивный закон: $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ (рис. 2.4).



Диаграммы левой и правой частей тождества совпадают, значит оно справедливо.

Рис. 2.4. Проверка дистрибутивного закона на диаграммах Эйлера-Венна

Справедливость тождеств можно доказать двумя способами:

I. по определению равенства двух множеств: $A=B \Leftrightarrow A\subseteq B$ и $B\subseteq A$;

II. путем преобразования одной части тождества в другую с использованием других тождеств.

Примеры

I. Докажем первый дистрибутивный закон:
$A\cup(B\cap C)=(A\cup B)\cap(A\cup C)$.

Необходимо доказать два утверждения:

1. если элемент принадлежит левой части тождества, то он принадлежит и правой части, т.е. $A\cup(B\cap C)\subseteq(A\cup B)\cap(A\cup C)$ и

2. если элемент принадлежит правой части тождества, то он принадлежит и левой части, т.е. $(A\cup B)\cap(A\cup C)\subseteq A\cup(B\cap C)$.

Обозначим левую часть тождества $A\cup(B\cap C)$ через *Dl*, а правую – $(A\cup B)\cap(A\cup C)$ через *Dr*.

1.1. Если элемент $x\in A\cup(B\cap C)$, то по определению операции объединения: $x\in A$ или $x\in B\cap C$.

а) Если элемент $x\in A$, то, по определению операции объединения множеств, $x\in A\cup B$ и $x\in A\cup C$, следовательно $x\in (A\cup B)\cap(A\cup C)$.

б) Если элемент $x\in B\cap C$, то, по определению операции пересечения множеств, $x\in B$ и $x\in C$, отсюда, по определению операции объединения, $x\in A\cup B$ и $x\in A\cup C$, следовательно, $x\in(A\cup B)\cap(A\cup C)$.

Так как для любого $x\in$Dl следует, что $x\in Dr$, то, по определению отношения включения, $Dl\subseteq Dr$.

1.2. Если элемент $x\in Dr$, то $x\in A\cup B$ и $x\in A\cup C$, значит ($x\in A$ или $x\in B$) и ($x\in A$ или $x\in C$), следовательно, ($x\in B$ и $x\in C$), т.е. $x\in B\cap C$. Отсюда, $x\in A$ или $x\in B\cap C$, т.е. $x\in Dl$.

Таким образом, между множествами *Dl* и *Dr* существуют отношение взаимного включения, значит *Dl=Dr*, что и требовалось доказать.

2. Докажем первый закон двойственности: $\overline{A\cup B}=\overline{A}\cap\overline{B}$.

2.1. Пусть элемент $x\in\overline{A\cup B}$. Тогда $x\in U$ и $x\notin A\cup B$, значит $x\notin A$ и $x\notin B$ (тонкий момент в доказательстве: x не принадлежит **ни** A, **ни** B), следовательно, $x\in\overline{A}$ и $x\in\overline{B}$, т.е. $x\in\overline{A}\cap\overline{B}$. Значит $Dl\subseteq Dr$.

2.2. Пусть теперь элемент $x\in\overline{A}\cap\overline{B}$. Тогда $x\in\overline{A}$ и $x\in\overline{B}$, значит $x\in U$ и (x одновременно не принадлежит A и B, т.е. $x\notin A$ или $x\notin B$), следовательно, $x\notin A\cup B$, т.е. $x\in\overline{A\cup B}$. Из этого следует, что $Dr\subseteq Dl$. Тождество доказано. Проверим справедливость этого тождества на диаграммах (рис. 2.5).

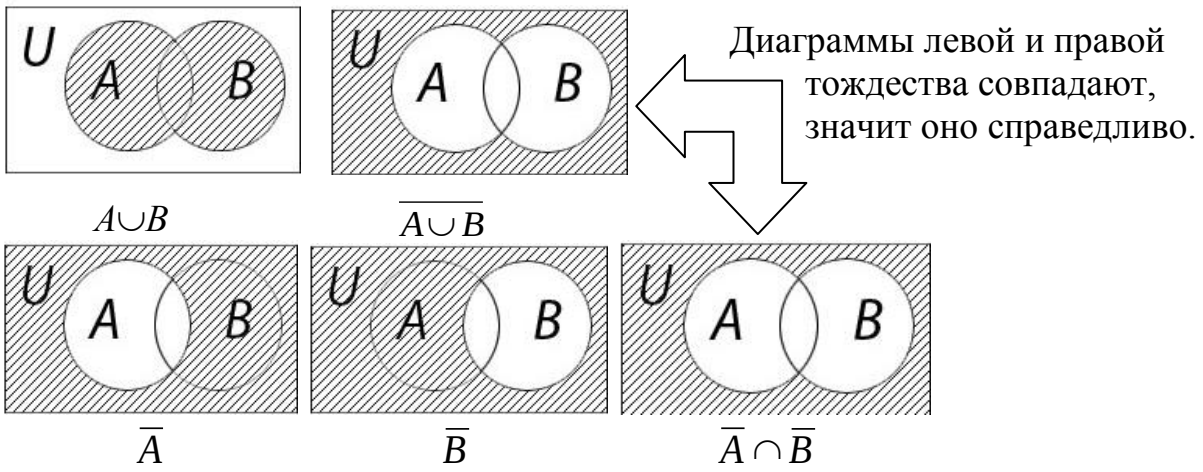


Рис. 2.5. Проверка правила де Моргана $\overline{A \cup B} = \overline{A} \cap \overline{B}$ на диаграммах Эйлера-Венна

II. Докажем второй закон поглощения: $A \cap (A \cup B) = A$ путем преобразования левой части тождества к правой с использованием других тождеств:

$$A \cap (A \cup B) = (A \cup \varnothing) \cap (A \cup B) = (A \cup (\varnothing \cap B)) = A \cup \varnothing = A.$$

свойство пустого множества — *по дистрибутивному закону* — *свойства пустого множества*

2.1.3. Прямое (декартово) произведение множеств

Понятие вектора

Под вектором понимается упорядоченный набор элементов. Определение является не строгим (интуитивным), так же, как и определение множества.

Элементы, образующие вектор, называются координатами или компонентами вектора. Число координат вектора называется его длиной или размерностью. Синонимом понятия «вектор» является «кортеж».

Для обозначения вектора обычно используются скобки, например, (1, 2, 1, 3). Иногда скобки и даже запятые в обозначении вектора опускаются.

Примером векторов могут служить целые числа, координаты – отдельные цифры числа.

Замечание

В отличие от элементов некоторого множества, некоторые координаты вектора могут совпадать.

Векторы длины 2 называются упорядоченными парами (или просто парами), длины 3 – тройками и т.д.

Два вектора равны, если они имеют одинаковую длину и соответствующие их координаты равны, т.е.

$(a_1, a_2, \ldots, a_m) = (b_1, b_2, \ldots, b_n)$, если $m=n$ и $a_1=b_1, a_2=b_2, \ldots, a_m=b_m$.

Прямым (декартовым) произведением множеств A и B называется множество всех пар (a, b), таких, что $a \in A$ и $b \in B$.

Обозначение: $A \times B = \{(a, b) \mid a \in A \text{ и } b \in B\}$.

Пример

$A = \{a, b, c\}$; $B=\{b, c\}$.

$A\times B = \{(a, b), (a, c), (b, b), (b, c), (c, b), (c, c)\}$;

$B\times A = \{(b, a\}, (b, b), (b, c), (c, a), (c, b), (c, c)\}$.

В общем случае $A\times B\neq B\times A$ – коммутативный закон не действует.

Пример

X – множество точек отрезка [0; 1];

Y – множество точек отрезка [1; 2].

Тогда $X\times Y$ – множество точек квадрата с вершинами в точках (0, 1), (0, 2), (1, 1), (1, 2).

Если $B = A$, то $A\times B = A\times A = A^2$ – декартова степень множества A.

Пример

R – множество всех действительных чисел,

R^2 – множество координат точек плоскости.

Прямым (декартовым) произведением множеств $A1, A2, \ldots, An$ называется совокупность всех упорядоченных n-ок (векторов длиной n) $(a_1, a_2, \ldots, a_n)$ таких, что

$$a_i\in Ai \ (i = 1, 2, \ldots, n).$$

В случае, если $A1 = A2 = \ldots = An = A$, то $A1\times A2\times\ldots\times An = A^n$.

Пример

X – множество точек отрезка [0;1];

Y – множество точек отрезка [1;2];

Z – множество точек отрезка [0;0,5].

$X\times Y\times Z$ – множество точек пространства, ограниченного параллелепипедом.

Пример

R^3 – множество координат точек пространства.

Пример

Изобразить на координатной плоскости следующее множество:

$$A = \{(x, y)\in R^2 \mid |y-1| < x < 3\}.$$

Решение (заштрихованная область) показано на рисунке 2.6.

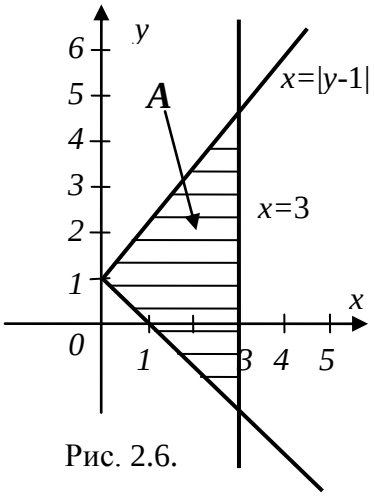


Рис. 2.6.

Теорема. Пусть $A1, A2, \ldots, An$ – конечные множества мощностью $m_1, m_2, \ldots, m_n$, соответственно, т.е. $|A1| = m_1, |A2| = m_2, \ldots, |An| = m_n$. Тогда мощность их прямого произведения равна произведению мощностей множеств - сомножителей, т.е.

$$|A1\times A2\times\ldots\times An| = m_1 * m_2 * \ldots * m_n.$$

Следствие. $|A^n| = |A|^n$.

Контрольные вопросы
по теме «Теория множеств»

1. Доказать тождества по определению равенства множеств:
   а) $A\cap(B\cup C)=(A\cap B)\cup(A\cap C)$
   б) $A\cup(\overline{A}\cap B)=A\cup B$
   в) $A\cap(B-C)=(A\cap B)-(A\cap C)$
2. Доказать тождества с использованием других тождеств:
   а) $A\cap(\overline{A}\cup B)=A\cap B$
   б) $(A\cup B)\cap(A\cup\overline{B})=A$
   в) $A\setminus(B\setminus C)=(A\setminus B)\cup(A\cap C)$
3. С помощью диаграмм Эйлера-Венна показать правомочность тождеств:
   а) $A\cap(B\cup C)=(A\cap B)\cup(A\cap C)$
   б) $A\cup(\overline{A}\cap B)=A\cup B$
   в) $\overline{A\cap B}=\overline{A}\cup\overline{B}$
   г) $A-(B-C)=(A-B)-C$
4. Убедится в правомочности тождества: $A\setminus(B\setminus C)=(A\setminus B)\cup(A\cap C)$ на примере числовых множеств:
   $A=\{x\in N|\ x$ − нечётное и $x\le 30\}$,
   $B=\{x\in N|\ x$ делится на 3 и $x\le 30\}$,
   $C=\{x\in N|\ x$ делится на 5 и $x\le 30\}$.
5. Записать заданные множества перечислением их элементов:
   а) $A=\{x\in Z|\ x^2+2x-3<0\}$
   б) $B=\{x\in N_0|\ |x+1|\le 5\}$
   в) $C=\{x\in Z|\ 0{,}1<3^{1-x}<100\}$
   г) $D=\{x\in Z|\ -1<\log_2|1-x|<2\}$
   д) $F=\{x\in R|\ \sin^2 2x=1$ и $0\le x\le 2\pi\}$
6. Изобразить на координатной плоскости следующие множества:
   а) $G=\{(x,y)\in R^2|\ |x-1|-1<y<3-|x+1|\}$
   б) $H=\{(x,y)\in R^2|\ y<x<\sqrt{4-y^2}\}$
   в) $K=\{(x,y)\in R^2|\ \log_{\frac{1}{3}}(x+1)<y<8+2x-x^2\}$
   г) $L=\{(x,y)\in R^2|\ \left(\frac{1}{2}\right)^{|x|}<y<2\cos x\}$
7. Действует ли коммутативный закон в отношении операции разности множеств? Утверждение обосновать.
8. Проверить правомочность дистрибутивных законов для операций пересечения и разности (относительного дополнения) множеств с использованием диаграмм Эйлера-Венна (3 балла за каждый) и/или с использованием тождественных преобразований (5 баллов за каждый).

9. Убедиться в правомочности тождества $A\times(B \setminus C)=(A\times B) \setminus (A\times C)$ на примере заданных множеств: A:{a,b,c}, B={c,d,e}, C={a,b,d}.

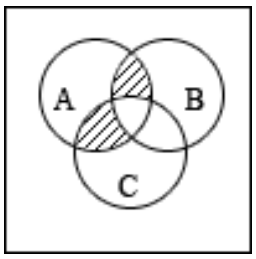
10. Упростить выражение: $((A\cup B)\cap C)\cup(\overline{A}\cap(\overline{B}\cup C))$

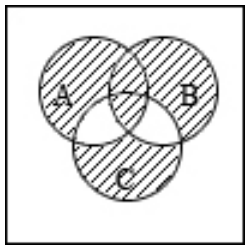
11. Доказать тождества:

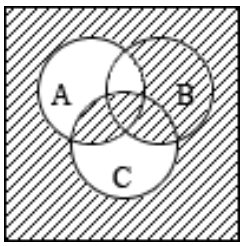
а) $(A \setminus B) - (B \setminus C) - (B \setminus A) - (C \setminus B)=A-C$

б) $(A \setminus B) \cup (B \setminus A)=(B \setminus A) - (A \setminus B)$

12. Записать множества, приведённые на диаграммах Эйлера-Венна:

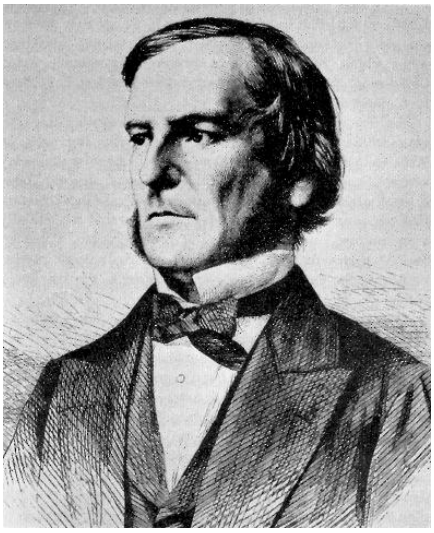
а) 

б) 

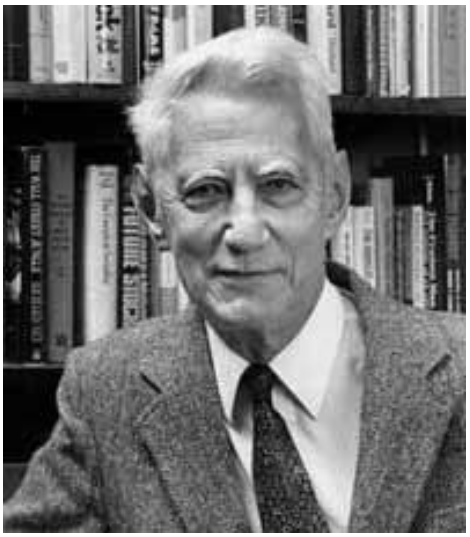
в) 

2.2. Булева алгебра. Основные понятия

Теоретическим фундаментом современных ЭВМ является алгебра логики, основы которой разработал Дж. Буль. В 1847 году вышла его работа с характерным названием – “Математический анализ логики, являющийся опытом исчисления дедуктивного рассуждения” (*Boole G.* The Mathematical Analysis of Logic: Being an Essay Towards a Calculus of Deductive Reasoning. Cambridge Library Collection — Mathematics. Cambridge University Press, 2009). Применяя алгебру (в дальнейшем она стала называться булевой алгеброй), можно было закодировать высказывания, истинность и ложность которых требовалось доказать, а потом оперировать ими, как в математике оперируют с числами. Дж. Буль ввел три основные операции: И, ИЛИ, НЕ, хотя алгебра допускает и другие операции – логические действия. Эти действия бинарны по своей сути, т.е. они оперируют с двумя состояниями: “истина” – “ложь”. Данное обстоятельство позволило в дальнейшем использовать булеву алгебру для описания переключательных схем. Добрых семьдесят лет после публикации его труд считался не более чем изящной, но чисто умозрительной конструкцией, пока Клод Шеннон не создал на основе булевской логики современную информатику. Клода Шеннона считают отцом теории информации и теории кодирования. Необходимо отметить, что окончательное оформление и завершение булева алгебра получила в работах последователей Дж. Буля: У.С. Джевонса и Дж. Венна (Англия), Э. Шредера (Германия), П.С. Порецкого (Россия).



Джордж Буль (02.11.1815 - 08.12.1864) – английский логик, математик и философ



Клод Шеннон (30.04.1916 - 24.02.2001) – американский инженер и математик

2.2.1. Элементы булевой алгебры

Основными элементами булевой алгебры являются:

- логические константы
- переменные;
- операции;

- выражения;
- функции;
- законы.

Логические константы

В булевой алгебре определены две логические константы: логический ноль (0) и логическая единица (1), которые отождествляются с понятиями “истина” и ”ложь” алгебры логики.

Переменные

Булевы (логические, двоичные) переменные - переменные, принимающие значения из множества {0, 1}.

Операции

Основными операциями булевой алгебры являются:

- отрицание (инверсия);
- конъюнкция (логическое умножение);
- дизъюнкция (логическое сложение).

Операция отрицания является унарной, а конъюнкция и дизъюнкция – n-арными.

Операции обозначаются следующим образом:

- Отрицание $\bar{a}$, $\neg a$;
- Конъюнкция $a\&b$, $a \cdot b$, $a*b$, ab, $a \wedge b$;
- Дизъюнкция $a \vee b$.

Выражения

Определение. ***Логическим (булевым) выражением*** называется совокупность булевых переменных, соединенных знаками булевых операций при возможном наличии скобок для изменения порядка выполнения операций.

При отсутствии скобок порядок выполнения операций определяется их приоритетом (значимостью). Для булевых операций порядок убывания приоритета следующий: $\neg$, &, $\vee$.

Примеры логических выражений: $a \vee b \cdot \bar{c}$, $(a \vee b) \cdot \bar{c}$,

$\overline{x_1 \bar{x}_3}$, $x_1 \bar{x}_1 x_3$.

Функции

Определение. ***Булевой (логической) функцией*** называется функция, аргументами которой являются булевы переменные, а сама функция принимает значение из множества {0, 1}.

Областью определения булевой функции является совокупность 2^n двоичных наборов ее аргументов. Набор аргументов можно рассматривать как n-компонентный двоичный вектор.

Булеву функцию можно задать с помощью следующих форм:

- аналитической;
- табличной;
- графической;
- таблично-графической;
- числовой;
- символической.

Аналитическая форма – булева функция задается логическим выражением, например:

$$y_1 = (\overline{x}_1 \vee x_2)x_3;$$

$$y_2 = \overline{x}_1\overline{x}_2\overline{x}_3 \vee \overline{x}_1 x_2\overline{x}_3 \vee x_1x_2\overline{x}_3.$$

Табличная форма – булева функция задается таблицей истинности.

Переход от аналитической формы к табличной однозначен. Обратный переход однозначным не является.

Пример: таблица истинности для функции y_1

x_1	x_2	x_3	$\overline{x}_1 \vee x_2$	y
0	0	0	1	0
0	0	1	1	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

Остальные формы задания булевой функции рассматриваются в последующих разделах.

Основные законы булевой алгебры

К основным законам (тождествам, правилам) булевой алгебры относятся:

1. Коммутативные (переместительные) законы:
   $$a \vee b = b \vee a;\ a \cdot b = b \cdot a.$$
2. Ассоциативные (сочетательные) законы:
   $$a \vee (b \vee c) = (a \vee b) \vee c;\ \ a \cdot (b \cdot c) = (a \cdot b) \cdot c.$$
3. Дистрибутивные (распределительные) законы:
   $$a \vee (b \cdot c) = (a \vee b) \cdot (a \vee c);\ \ a \cdot (b \vee c) = a \cdot b \vee a \cdot c.$$

4. Закон двойного отрицания: $\overline{\overline{a}} = a$.

5. Законы тавтологии (идемпотентности): $a \vee a = a$; $a \cdot a = a$.

6. Законы нулевого элемента: $a \vee 0 = a$; $a \cdot 0 = 0$.

7. Законы единичного элемента: $a \vee 1 = 1$; $a \cdot 1 = a$.

8. Законы дополнительного элемента. В булевой алгебре дополнительным элементом по отношению к a является отрицание $(\overline{a})$.

$$a \vee \overline{a} = 1; \quad a \cdot \overline{a} = 0.$$

9. Законы двойственности (де Моргана):

$$\overline{a \cdot b} = \overline{a} \vee \overline{b}; \quad \overline{a \vee b} = \overline{a} \cdot \overline{b}.$$

Следствия: $$a \cdot b = \overline{\overline{a} \vee \overline{b}}; \quad a \vee b = \overline{\overline{a} \cdot \overline{b}}.$$

10. Законы поглощения: $a \vee a \cdot b = a; \quad a \cdot (a \vee b) = a.$

11. Правила сокращения: $a \vee \overline{a} \cdot b = a \vee b; \quad a \cdot (\overline{a} \vee b) = a \cdot b.$

Следствия: $\overline{a} \vee a \cdot b = \overline{a} \vee b; \quad \overline{a} \cdot (a \vee b) = \overline{a} \cdot b.$

12. Правила склеивания:

$$a \cdot b \vee a \cdot \overline{b} = a; \quad (a \vee b) \cdot (a \vee \overline{b}) = a.$$

Комментарии к законам булевой алгебры

1. Большинство законов задается парой соотношений (дуальность законов булевой алгебры). При этом одно соотношение можно получить из другого заменив операции конъюнкции на дизъюнкцию или дизъюнкцию на конъюнкцию. В законах, в которых участвуют логические константы (0 и 1), они заменяются на противоположные значения.

2. Некоторые законы можно распространять на произвольное число элементов.

3. В любом законе любую букву можно заменить на произвольное логическое выражение.

4. Законы применяются для упрощения булевых функций.

2.2.2. Разнообразие булевых функций

Булевы функции от одной переменной приведены в табл.2.1.

Таблица 2.1.

Обозначение аргумента и функции	Значения аргумента и функции		Наименование функции
$\boldsymbol{x}$	0	1	
f_0^1	0	0	Логический ноль
f_1^1	0	1	Повторение $\boldsymbol{x}$
f_2^1	1	0	Инверсия $\boldsymbol{x}$
f_3^1	1	1	Логическая единица

Определение. Булева функция от n аргументов $f^n(X)$ называется ***вырожденной по аргументу*** $\boldsymbol{x_i}$, если ее значение не зависит от этого аргумента, то есть для всех наборов аргументов имеет место равенство:

$$f(x_1, x_2, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) = f(x_1, x_2, x_{i-1}, 1, x_{i+1}, \dots, x_n).$$

Среди функций от одной переменной содержатся две вырожденные: логический ноль и логическая единица.

Определение. Невырожденные функции от двух переменных с добавлением функции отрицания принято называть ***базовыми функциями*** булевой алгебры. С учетом обращаемости некоторых базовых функций по отношению к своим аргументам, их общее количество равно девяти.

Булевы функций от двух переменных приведены в табл. 2.2.

Среди функций от двух переменных шесть вырожденных, к которым относятся:

- логический ноль;
- логическая единица;
- функция повторения аргументов $\boldsymbol{x_1}$ и $\boldsymbol{x_2}$;
- отрицание аргументов $\boldsymbol{x_1}$ и $\boldsymbol{x_2}$.

Таблица 2.2.

Аргументы и функции (в символической форме)	Значения аргументов и функций				Обозначение функций	Наименование	Вырожденность	Представление функции в булевом базисе
x_1	0	0	1	1				
x_2	0	1	0	1				
f_0^2	0	0	0	0	0	Логический ноль	+	-
f_1^2	0	0	0	1	$x_1 \& x_2$	Конъюнкция	-	$x_1 \cdot x_2$
f_2^2	0	0	1	0	$x_1 \Delta x_2$	Запрет x_1 по x_2	-	$x_1 \cdot \bar{x}_2$
f_3^2	0	0	1	1	x_1	Повторение x_1	+	-
f_4^2	0	1	0	0	$x_2 \Delta x_1$	Запрет x_2 по x_1	-	$\bar{x}_1 \cdot x_2$
f_5^2	0	1	0	1	x_2	Повторение x_2	+	-
f_6^2	0	1	1	0	$x_1 \oplus x_2$	Сумма по модулю 2, неравнозначность, исключающее ИЛИ	-	$\bar{x}_1 \cdot x_2 \vee x_1 \cdot \bar{x}_2$
f_7^2	0	1	1	1	$x_1 \vee x_2$	Дизъюнкция	-	$x_1 \vee x_2$
f_8^2	1	0	0	0	$x_1 \downarrow x_2$	Функция Вебба, стрелка Пирса	-	$\overline{x_1 \vee x_2}$
f_9^2	1	0	0	1	$x_1 \sim x_2$ $(x_1 \equiv x_2)$	Равнозначность, эквивалентность	-	$x_1 \cdot x_2 \vee \bar{x}_1 \cdot \bar{x}_2$
f_{10}^2	1	0	1	0	$\bar{x}_2$	Отрицание x_2	+	$\bar{x}_2$
f_{11}^2	1	0	1	1	$x_2 \rightarrow x_1$	Импликация от x_2 к x_1	-	$x_1 \vee \bar{x}_2$
f_{12}^2	1	1	0	0	$\bar{x}_1$	Отрицание x_1	+	$\bar{x}_1$
f_{13}^2	1	1	0	1	$x_1 \rightarrow x_2$	Импликация от x_1 к x_2	-	$\bar{x}_1 \vee x_2$
f_{14}^2	1	1	1	0	$x_1 \mid x_2$	Штрих Шеффера	-	$\overline{x_1 \cdot x_2}$
f_{15}^2	1	1	1	1	1	Логическая единица	+	-

Некоторые функции от трех переменных представлены в табл. 2.3.

Таблица 2.3.

Значение аргументов			Значение функций		
			Сумма по модулю 2	Исключающее ИЛИ	Функция мажоритарности
x_1	x_2	x_3	$x_1 \oplus x_2 \oplus x_3$	$XOR\ (x_1, x_2, x_3)$	$x_1 \# x_2 \# x_3$
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	1	1	0
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	0	0	1
1	1	0	0	0	1
1	1	1	1	0	1

Замечание. Функции сумма по модулю 2 и исключающее ИЛИ для трех аргументов являются неэквивалентными.

Утверждение. Общее число разнообразных булевых функций, в том числе и вырожденных, от n аргументов равно 2^{2^n} .

2.2.3. Нормальные формы булевых функций

Нормальные формы – это особый класс аналитических выражений, используемых при решении задачи минимизации булевых функций и для перехода от табличной формы задания к аналитической. Нормальные формы строятся на основании операций конъюнкции, дизъюнкции и отрицания, причем отрицание только единственной переменной.

Определение. ***Элементарной конъюнкцией*** **(*дизъюнкцией*)** называется конъюнкция (дизъюнкция) конечного числа попарно различимых переменных или их отрицаний.

Элементарную конъюнкцию (дизъюнкцию) называют также ***конъюнктивным* (*дизъюнктивным*) *термом*.**

В частном случае терм, как конъюнктивный, так и дизъюнктивный, может состоять из единственной буквы (литерала). Под буквой будем понимать аргумент булевой функции или его отрицание.

Примерами термов являются: x_1, x_2, $x_1\bar{x}_3$, $x_2 \vee \bar{x}_4 \vee x_5$.

Выражения типа: $\overline{x_1\bar{x}_3}$, $x_1\bar{x}_1x_3$.термами не являются, так как в первом случае знак отрицания стоит над двумя переменными, а во втором случае переменная x_1 находится в выражении с отрицанием и без него.

Определение. ***Рангом терма*** называется количество букв, входящих в него.

Определение. ***Дизъюнктивной (конъюнктивной) нормальной формой*** булевой функции называется дизъюнкция (конъюнкция) конечного числа попарно различимых конъюнктивных (дизъюнктивных) термов.

Определение. ***Конституентой единицы (нуля)*** называется конъюнктивный (дизъюнктивный) терм максимального ранга, т.е. для булевой функции от *n* переменных конституента включает в себя *n* букв.

Свойство конституенты. Конституента единицы (нуля) принимает значение единицы (нуля) на одном и только одном наборе аргументов.

Пример: При $n = 4$ конъюнктивный терм $x_1\bar{x}_2x_3\bar{x}_4$ принимает значение, равное единице, на наборе 1010, а дизъюнктивный терм $\bar{x}_1 \vee \bar{x}_2 \vee x_3 \vee \bar{x}_4$ принимает значение, равное нулю, на наборе 1101.

Определение. Дизъюнктивная (конъюнктивная) нормальная форма называется ***канонической***, если все ее конъюнктивные (дизъюнктивные) термы представляют собой конституенты единицы (нуля). Канонические формы называют также ***совершенными.***

Замечания:

1. С помощью канонических форм наиболее просто осуществляется переход от табличной формы задания булевой функции к аналитической.
2. С помощью канонических форм любую булеву функцию можно представить в булевом базисе.
3. Любая булева функция, за исключением логического нуля и логической единицы, имеет единственные КДНФ и ККНФ. Логическую единицу можно представить в виде КДНФ, а логический ноль - в виде ККНФ.
4. Правило перехода от табличной формы задания булевой функции к аналитической:

а) в таблице истинности выделяются все наборы аргументов, при которых функция равна единице (нулю).

б) для каждого из этих наборов составляют конституенты единицы (нуля).

в) объединением конституенты единицы (нуля) знаками дизъюнкции (конъюнкции) получается аналитическая форма в виде КДНФ (ККНФ).

Пояснение. При составлении конституент единицы (нуля) используют следующее правило:

Если некоторый аргумент принимает на наборе значение, равное нулю, то в конституенту единицы он входит с отрицанием, а в конституенту нуля без него.

Пример: получим канонические формы для функции $\boldsymbol{y = x_1 \oplus x_2}$ (Таблица 2.4).

Таблица 2.4.

x_1	x_2	y	Конституенты единицы	Конституенты нуля
0	0	0	-	$x_1 \vee x_2$
0	1	1	$\bar{x}_1 \cdot x_2$	-
1	0	1	$x_1 \cdot \bar{x}_2$	-
1	1	0	-	$\bar{x}_1 \vee \bar{x}_2$

КДНФ – каноническая дизъюнктивная нормальная форма:

$$y = \bar{x}_1 \cdot x_2 \vee x_1 \cdot \bar{x}_2 ;$$

ККНФ – каноническая конъюнктивная нормальная форма:

$$y = (x_1 \vee x_2) \cdot (\bar{x}_1 \vee \bar{x}_2) .$$

КДНФ и ККНФ представляют собой две различные, но эквивалентные аналитические формы булевой функции. Это означает, что из одной формы можно получить другую, используя законы булевой алгебры.

$$y = (x_1 \vee x_2) \cdot (\bar{x}_1 \vee \bar{x}_2) = x_1 \cdot \bar{x}_1 \vee x_1 \cdot \bar{x}_2 \vee x_2 \cdot \bar{x}_1 \vee x_2 \cdot \bar{x}_2 =$$

$$= x_1 \cdot \bar{x}_2 \vee x_2 \cdot \bar{x}_1 = x_1 \cdot \bar{x}_2 \vee \bar{x}_1 \cdot x_2 \quad \text{(КДНФ)}$$

Существует другой способ получения ККНФ:

а) составляется КДНФ, но не для самой булевой функции, а для ее отрицания.

б) берется отрицание над полученной КДНФ, которое снимается с применением закона двойственности.

$$\bar{y} = \bar{x}_1 \cdot \bar{x}_2 \vee x_1 \cdot x_2$$

$$\bar{\bar{y}} = y = \overline{\bar{x}_1 \cdot \bar{x}_2 \vee x_1 \cdot x_2} = \overline{\bar{x}_1 \cdot \bar{x}_2} \cdot \overline{x_1 \cdot x_2} = (x_1 \vee x_2) \cdot (\bar{x}_1 \vee \bar{x}_2)$$

2.2.4. Числовая и символическая формы представления булевых функций

Для любой булевой функции можно предложить две ***числовые формы***, основанные на перечислении десятичных эквивалентов наборов аргументов, на которых функция принимает значение единицы (нуля).

Например, $f^3(X) = \underset{f=1}{\vee}(0,2,6,7).$

От числовой формы легко перейти к КДНФ путем замены каждого из наборов в перечислении конституентой единицы.

$$f^3(X) = \bar{x}_1 \bar{x}_2 \bar{x}_3 \vee \bar{x}_1 x_2 \bar{x}_3 \vee x_1 x_2 \bar{x}_3 \vee x_1 x_2 x_3 .$$

Аналогично можно перейти к ККНФ:

$$f^3(X) = \underset{f=0}{\&}(1,3,4,5).$$

$$f^3(X) = (x_1 \vee x_2 \vee \bar{x}_3) \cdot (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \cdot (\bar{x}_1 \vee x_2 \vee x_3) \cdot (\bar{x}_1 \vee x_2 \vee \bar{x}_3).$$

В самом компактном виде любую булеву функцию можно представить в следующей ***символической форме***: f_N^n, где n – количество аргументов, а N – десятичный эквивалент двоичного набора значений функции на упорядоченном множестве аргументов.

Пример: $f^3(X)=x_1 \oplus x_2 \oplus x_3$. N = (01101001) = 64+32+8+1 = 105,

f_{105}^3 – символическая форма булевой функции.

2.2.5. Задача минимизации булевых функций

Выражения булевых функций являются математическими моделями, на основании которых строятся комбинационные схемы. Проектируемые схемы должны быть оптимальными в смысле минимума используемого оборудования при выполнении ограничений на быстродействие схемы, которое определяется временем распространения сигналов от входов схемы к ее выходам. Количество оборудования, используемого в схеме, принято характеризовать ценой схемы. Если в схеме используются элементы k типов с ценами $s_1, s_2, ..., s_k$, и в количестве $n_1, n_2, ..., n_k$, то цена схемы определяется суммарной ценой элементов:

$$S = \sum_{i=1}^{k} s_i n_i.$$

В настоящее время отсутствует методика проектирования логических схем, оптимальных в смысле минимума цены S. В связи с этим используются методы проектирования, основанные на ряде допущений, которые позволяют синтезировать схемы, близкие к оптимальным.

Канонический метод проектирования комбинационных схем, обеспечивающий решение задачи синтеза для широкого класса схем, состоит в следующем. Закон функционирования проектируемой схемы, в общем случае, задается системой булевых функций, в частном случае одной булевой функцией. Аналитические выражения булевых функций путем эквивалентного преобразования приводятся к виду, позволяющему строить экономичные схемы.

Обычно задача упрощения булевых функций решается в два этапа. На первом этапе упрощается ДНФ или КНФ функций путем решения задачи минимизации. Нормальные формы, получаемые при этом, называются минимальными. На втором этапе производится дальнейшее упрощение минимальных нормальных форм функций путем преобразования их в скобочные формы (задача факторизации булевых функций). Окончательная форма функций используется для построения проектируемой схемы.

Считается, что схемы, характеризуемые малым значением S, являются минимальными по количеству используемого оборудования. Задача минимизации булевых функций по критерию минимальности числа букв, входящих в ДНФ функции, называется ***канонической задачей минимизации***. Схема, получаемая в результате ее решения, не является абсолютно минимальной. Поэтому, используя скобочные формы функций, можно

провести дальнейшее упрощение схемы, но абсолютный минимум оборудования в большинстве случаев так и не достигается.

2.2.6. Кубическое представление булевых функций

В кубическом представлении булевой функции от n переменных все множество из 2^n наборов ее аргументов рассматривается как множество координат вершин n-мерного куба с длиной ребра, равной 1. В соответствии с этим наборы аргументов, на которых булева функция принимает значение равное 1, принято называть существенными вершинами.

Существенные вершины образуют так называемые ноль-кубы (0-кубы). Между 0-кубами существует отношение соседства и определена операция склеивания. Два 0-куба называются соседними, если они отличаются только по одной координате.

Пример: n=4

$$\left.\begin{array}{cccc} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{array}\right\} \text{ – два соседних 0-куба}$$

$$\begin{array}{cccc} 0 & \times & 0 & 1 \end{array} \text{ – результат склеивания: (1)}$$

Склеивание двух соседних 0-кубов дает в результате 1-куб. Координата, отмечаемая символом ×, называется свободной (независимой, несвязанной) и может принимать любое значение из множества {0, 1}, а остальные (числовые) координаты называются зависимыми (связанными). Аналогичное отношение соседства существует между 1-кубами, в результате склеивания которых получается 2-куб.

$$\left.\begin{array}{cccc} 0 & \times & 0 & 1 \\ 0 & \times & 1 & 1 \end{array}\right\} \text{ – два соседних 1-куба}$$

$$\begin{array}{cccc} 0 & \times & \times & 1 \end{array} \text{ – результат склеивания: (2)}$$

В продолжение аналогии: два r-куба называются соседними, если они отличаются только по одной (естественно зависимой) координате. r-куб содержит r независимых и n-r зависимых координат. В результате склеивания двух соседних r-кубов образуется (r+1)-куб, содержащий r+1 независимую координату.

Операция склеивания над кубами соответствует применению закона склеивания к конъюнктивным термам, отождествляемым с этими кубами.

Пример: для склеивания (1)

$$\bar{x}_1 x_2 \bar{x}_3 x_4 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 x_4 = \bar{x}_1 \bar{x}_3 x_4$$

$$0\ 1\ 0\ 1 \qquad 0\ 0\ 0\ 1 \qquad 0 \times 0\ 1$$

для склеивания (2)

$$\bar{x}_1 \bar{x}_3 x_4 \vee \bar{x}_1 x_3 x_4 = \bar{x}_1 x_4$$

$$0 \times 0\ 1 \qquad 0 \times 1\ 1 \qquad 0 \times \times 1$$

Определение. Кубическим комплексом $K^0(f)$ булевой функции f называется множество 0-кубов этой функции. В общем случае, кубическим комплексом $K(f)$ булевой функции f называется объединение множеств кубов

всех размерностей этой функции $K(f)=\bigcup_{r=0}^{m}K^r(f)$, где m – максимальная размерность кубов функции *f*.

Пример получения кубического комплекса: $y = f^3(X) = \vee_{f=1}(1,2,3,6,7)$.

$$K^0(f)=\left\{\begin{matrix}001\\010\\011\\110\\111\end{matrix}\right\}\left(\begin{matrix}1\\2\\3\\4\\5\end{matrix}\right) \qquad K^1(f)=\left\{\begin{matrix}0\times1\\01\times\\\times10\\\times11\\11\times\end{matrix}\right\}\left(\begin{matrix}1-3 & 1\\2-3 & 2\\2-4 & 3\\3-5 & 4\\4-5 & 5\end{matrix}\right) \qquad K^2(f)=\left\{\begin{matrix}\times1\times\\\times1\times\end{matrix}\right\}\left(\begin{matrix}2-5\\3-4\end{matrix}\right)$$

$K^3(f)=\varnothing$ - пустое множество. $K(f)=K^0(f)\cup K^1(f)\cup K^2(f)$.

Для получения кубического комплекса $K(f)$ необходимо провести всевозможные операции склеивания над 0-кубами, 1-кубами и т.д. до тех пор, пока на очередном шаге не получится $K^{r+1}(f) = \varnothing$. При склеивании 1-кубов 2-кубы представлены в двух экземплярах, как результаты склеивания 2-х различных пар 1-кубов.

Распространяя этот принцип, можно утверждать, что r-кубы как результат склеивания $(r\text{-}1)$-кубов получаются в r-кратном количестве экземпляров.

Определение. Куб, входящий в состав кубического комплекса $K(f)$, называется максимальным, если он не вступает ни в одну операцию склеивания.

В приведенном примере максимальными кубами являются ×1× и 0×1.

2.2.7. Геометрическая интерпретация кубов малой размерности. Графическое представление булевых функций

Графическое представление булевых функций носит ограниченный характер и, как правило, является наглядным для булевых функций от двух и трех переменных.

Геометрическим местом 0-куба является точка, представляющая существенную вершину.

Два соседних 0-куба являются концами какого-либо ребра.

Геометрическим местом 1-куба является ребро, замыкаемое склеивающимися 0-кубами, образующими данный 1-куб.

Два параллельных ребра, образующих грань, являются образами склеивающихся 1-кубов. В соответствии с этим геометрической интерпретацией 2-куба является грань, образуемая парой параллельных ребер. Так как любую грань можно определить одной из пар параллельных ребер, 2-куб может быть получен как результат склеивания двух различных пар 1-кубов, то есть представляется в двух экземплярах.

Геометрическим образом 3-куба можно считать трехмерный куб. Так как он может быть образован тремя способами как пара параллельных граней, то при склеивании он получается в трех экземплярах.

По аналогии r-куб ($r \geq 2$) получается в r экземплярах как результат склеивания r различных пар (r-1) кубов.

Пример: На рисунке 2.7 показано графическое представление функции $f^3(X) = \vee_{f=1}(1,2,3,6,7)$.

Графический способ представления булевых функций является обобщением аналитического и табличного способов. Он объединяет в себе понятия алгебры логики, теории множеств и топологические свойства геометрии n-мерного куба. В силу этого данный способ представления булевых функций получил наименование ***алгебро-топологический метод***.

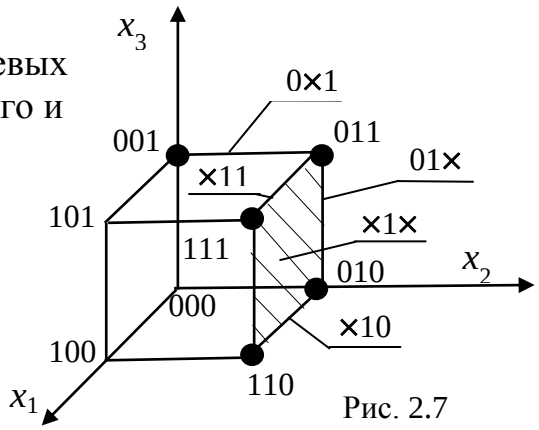


Рис. 2.7

2.2.8. Покрытия булевых функций

Между кубами различной размерности, входящими в кубический комплекс $K(f)$, существует отношение включения или покрытия. Принято говорить, что куб ***A*** меньшей размерности покрывается кубом ***B*** большей размерности. Куб ***A*** включается в куб ***B***, если при образовании куба ***B*** хотя бы в одном склеивании участвует куб ***A***.

Отношение включения (покрытия) между кубами принято обозначать ***A*** $\subset$ ***B***. В теории множеств отношение включения связывает между собой некоторое множество и его подмножества.

Для рассмотренного примера отношения включения имеют место между следующими кубами: 001$\subset$0×1; 011$\subset$ ×11$\subset$ ×1×. Любой 1-куб покрывает два 0-куба, 2-куб - четыре 0-куба и четыре 1-куба, 3-куб покрывает восемь 0-кубов, двенадцать 1-кубов и шесть 2-кубов (см. геометрическую интерпретацию).

Определение. ***Покрытием булевой функции*** f называется такое подмножество кубов из кубического комплекса $K(f)$, которое покрывает все существенные вершины функции.

В связи с тем, что любому кубу комплекса $K(f)$ можно поставить в соответствие конъюнктивный терм, для любого покрытия можно составить некоторую ДНФ булевой функции.

Частным случаем покрытия булевой функции является кубический комплекс $K^0(f)$, покрытие $C_0(f)=K^0(f)$. Этому покрытию соответствует КДНФ.

Для рассмотренного выше примера покрытием является также комплекс $K^1(f)$:

$$C_1(f) = K^1(f) = \left\{ \begin{matrix} 0 \times 1 \\ 0\ 1 \times \\ \times\ 1\ 0 \\ \times\ 1\ 1 \\ 1\ 1 \times \end{matrix} \right\}.$$

Этому покрытию соответствует ДНФ вида:

$$f(X) = \bar{x}_1 x_3 \vee \bar{x}_1 x_2 \vee x_2 \bar{x}_3 \vee x_2 x_3 \vee x_1 x_2.$$

Приведенная ДНФ не является минимальной.

В качестве еще одного варианта покрытия можно использовать множество максимальных кубов. Для рассмотренного выше примера:

$$C_2(f) = Z(f) = \left\{ \begin{matrix} 0 \times 1 \\ \times\ 1 \times \end{matrix} \right\}.$$

Действительно, кубы, входящие в $Z(f)$, покрывают все существенные вершины: $0\times1 \supset (001, 011)$, $\times1\times \supset (010, 011, 110, 111)$.

Замечание. Множество максимальных кубов булевой функции всегда является ее покрытием.

Покрытию $C_2(f)$ соответствует ДНФ вида:

$$f(X) = \bar{x}_1 x_3 \vee x_2.$$

Эта ДНФ является ***минимальной***.

Определение. Покрытие булевой функции, которое соответствует минимальной ДНФ, называется ***минимальным покрытием***.

Замечание. Минимальное покрытие должно состоять только из максимальных кубов.

В частном случае множество максимальных кубов может являться минимальным покрытием. Это справедливо для рассмотренного выше примера. В общем случае множество максимальных кубов является избыточным, и для получения минимального покрытия достаточно выделить некоторое его подмножество.

Пример:

$$K^0(f) = \left\{ \begin{matrix} 000 \\ 001 \\ 100 \\ 110 \\ 111 \end{matrix} \right\} \left(\begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \right); \quad K^1(f) = \left\{ \begin{matrix} 00\times \\ \times 00 \\ 1\times 0 \\ 11\times \end{matrix} \right\} \left(\begin{matrix} 1 - 2 \\ 1 - 3 \\ 3 - 4 \\ 4 - 5 \end{matrix} \right); \quad K^2(f) = \varnothing.$$

Для данного примера множество максимальных кубов совпадает с комплексом $K^1(f)$: $Z(f) = K^1(f)$.

Минимальными покрытиями являются

$$C_1(f) = \left\{\begin{matrix}00\times\\11\times\\\times00\end{matrix}\right\};\ C_2(f) = \left\{\begin{matrix}00\times\\11\times\\1\times0\end{matrix}\right\}.$$

Определение. ДНФ, соответствующая множеству максимальных кубов, называется ***сокращенной*** (СДНФ).

Для рассматриваемого примера СДНФ: $f(X) = \bar{x}_1\bar{x}_2 \vee \bar{x}_2\bar{x}_3 \vee x_1\bar{x}_3 \vee x_1x_2.$

Из анализа покрытия существенных вершин максимальными кубами из комплекса $K^1(f)$ следует:

1. Куб 00× должен обязательно включаться в покрытие, так как он и только он покрывает существенную вершину 001, аналогично только куб 11× покрывает существенную вершину 111.

Определение. Множество максимальных кубов, без которых не может быть образовано покрытие булевой функции, называется ***ядром покрытия*** и обозначается $T(f)$: $T(f) = \{00\times, 11\times\}$.

2. Так как ядром покрытия, кроме существенных вершин 001 и 111, покрываются также существенные вершины 000 и 110, то не покрытой ядром остается только существенная вершина 100. Для ее покрытия достаточно взять любой из оставшихся максимальных кубов: ×00 или 1×0.

Нулевое покрытие булевой функции и получение минимальной КНФ

Выше было рассмотрено покрытие булевой функции на наборах аргументов, для которых функция равна единице.

Такие покрытия можно назвать единичными. Наряду с единичными покрытиями существуют и нулевые, для которых покрываются наборы аргументов, на которых функция равна нулю, то есть покрытие реализуется для существенных вершин, но не самой функции, а ее отрицания (инверсии).

Нулевое покрытие строится так же, как и единичное, но только для отрицания исходной функции.

$$f^3(X) = \bigvee_{f=1}(0,1,4,6,7), \quad f^3(X) = \&_{f=0}(2,3,5).$$

$$K^0(\bar{f}) = \left\{\begin{matrix}010\\011\\101\end{matrix}\right\} \qquad S^a=9,\ S^b=12;$$

$$K^1(\bar{f}) = \{01\times\}$$

$$Z(\bar{f}) = C_{\min}(\bar{f}) = \left\{\begin{matrix}01\times\\101\end{matrix}\right\} \quad S^a=5,\ S^b=7.$$

Цена минимального нулевого покрытия оказалась меньше цены минимального единичного покрытия.

Так как заранее предсказать, какое из минимальных покрытий данной функции, единичное или нулевое, будет иметь меньшую цену, невозможно, то для построения схемы, обладающей минимальной ценой по Квайну, целесообразно решать задачу минимизации в отношении обоих покрытий.

2.2.9. Импликанты булевой функции. Системы импликант

Решение задачи минимизации булевой функции базируется на понятиях импликант и их систем.

Определение. Булева функция $g(X)$ называется ***импликантой*** булевой функции $f(X)$, если для любого набора аргументов, на которых $g(X)=1$, $f(X)$ также равна единице.

$g(\tilde{X}) = 1 \Rightarrow f(\tilde{X}) = 1$, где $\tilde{X}$ – некоторый набор аргументов.

Свойства импликант:

1) Между импликантой и самой функцией существует отношение включения $g(X) \subset f(X)$.

2) Можно утверждать, что для любого набора аргументов, на котором функция равна нулю, ее импликанта также равна нулю.

3) Если $g(X)$ и $\varphi(X)$ являются импликантами функции $f(X)$, то их дизъюнкция также является импликантой этой функции.

Простейшими примерами импликант могут служить конъюнктивные термы, входящие в произвольную ДНФ данной функции.

Пример: для $f^{3}(X) = \bigvee_{f=1}(0,1,4,6,7)$ (3) импликантами являются $\bar{x}_1\bar{x}_2x_3$, $x_1x_2\bar{x}_3$, x_1x_2.

Произвольная дизъюнкция этих термов также является импликантой функции.

Определение. ***Простой (первичной) импликантой*** булевой функции называется конъюнктивный терм, который сам является импликантой этой функции, но никакая его собственная часть уже не является импликантой этой функции.

Определение. Под ***собственной частью терма*** понимается новый терм, полученный из исходного, путем вычеркивания произвольного числа букв.

Для данного примера функции (3) простыми импликантами являются: $\bar{x}_1\bar{x}_2$, x_1x_2, $\bar{x}_2\bar{x}_3$, $x_1\bar{x}_3$.

Множеству простых импликант можно поставить в соответствие множество максимальных кубов.

Дизъюнкция всех простых импликант булевой функции представляет собой ДНФ этой функции, которая также является ***сокращенной*** - СДНФ.

Для функции (1) из приведенного примера СДНФ:

$$y = \bar{x}_1\bar{x}_2 \vee x_1x_2 \vee \bar{x}_2\bar{x}_3 \vee x_1\bar{x}_3.$$

Понятие «сокращенная» присвоено ДНФ в том смысле, что она, как правило, содержит меньшее количество букв и термов по сравнению с

КДНФ. Для нашего примера КДНФ содержит 15 букв и 5 термов, а СДНФ - 8 букв и 4 терма.

Аналогия между импликантами и кубическим представлением булевой функции

Любому кубу из $K(f)$ можно поставить в соответствие конъюнктивный терм, который можно рассматривать как импликанту булевой функции. Любой простой импликанте булевой функции соответствует максимальный куб, и, в свою очередь, множество всех простых импликант соответствует множеству $Z(f)$ всех максимальных кубов $K(f)$.

Таким образом, можно провести некоторую аналогию между сокращенной СДНФ и $Z(f)$.

В отношении импликант булевой функции так же, как и в отношении кубов, соответствующих им, существует отношение покрытия.

Принято считать, что импликанта булевой функции покрывает некоторую существенную вершину этой функции или, в общем случае, некоторый куб из $K(f)$, если значение импликанты на наборе аргументов, представляющем данную существенную вершину, равно 1, или, в общем случае, значение импликанты равно 1 для всех существенных вершин, покрываемых кубом из $K(f)$.

Пример: импликанта x_1x_2 функции $f^{3}(x)$ покрывает существенные вершины (110, 111) и в свою очередь покрывает куб 11×.

Определение. Множество импликант булевой функции образует ***полную систему импликант***, если любая существенная вершина булевой функции покрывается хотя бы одной импликантой этого множества.

Если считать, что в полную систему импликант включаются импликанты только в виде конъюнктивных термов и не включаются импликанты в виде дизъюнкции термов, то полной системе импликант можно поставить в соответствие некоторое множество кубов из $K(f)$ образующих покрытие булевой функции f.

Так, например, кубам из кубического комплекса $K^0(f)$ соответствует полная система импликант, представляющая собой множество конституент 1 данной функции f. В свою очередь, множеству максимальных кубов $Z(f)$, естественно образующих покрытие булевой функции, соответствует полная система простых импликант.

Определение. Система простых импликант называется ***приведенной***, если она является полной, а никакая ее собственная часть уже не образует полную систему импликант.

Для функции $y = x_1x_2 \vee \bar{x}_1\bar{x}_2 \vee \bar{x}_2\bar{x}_3 \vee x_1\bar{x}_3$ система простых импликант $\{x_1x_2,\ \bar{x}_1\bar{x}_2, \bar{x}_2\bar{x}_3, x_1\bar{x}_3\}$ является полной, но не является приведенной, т.к. из нее можно исключить одну из импликант $\bar{x}_2\bar{x}_3$ или $x_1\bar{x}_3$ не нарушая полноты системы.

Определение. Простая импликанта булевой функции называется ***существенной***, если она и только она покрывает некоторую существенную вершину этой функции.

Множество существенных импликант соответствует максимальным кубам, образующим ядро покрытия.

2.2.10. Операции над кубами

Операции над кубами являются аналогами операций над множествами. Единственным отличием является то, что кубы всегда являются гранями n-мерного куба, то есть несут в себе геометрические (топологические) свойства и являются замкнутыми в той или иной структуре n-мерного пространства.

Операция объединения кубов. Операция объединения кубов обозначается символом «$\cup$». В этом смысле любое покрытие и сам комплекс булевой функции есть объединение кубов различной размерности $K(f)=K^0(f) \cup K^1(f) \cup \ldots \cup K^r(f)$, где $r \le n$. Для удобства записи знак объединения часто заменяется на запятую, и покрытие задается перечислением кубов покрытия.

Операция пересечения кубов. Операция пересечения кубов является аналогом пересечения множеств и состоит в поиске их общих частей, если таковая существует. Следовательно, пересечение кубов может давать либо пустое множество, либо общую грань.

Пример. Пусть заданы три куба c_1=0××, c_2=×11 и c_3=1×1. Тогда у кубов c_1 и c_2 имеется общая вершина 011 и их пересечение не пусто, а $c_1 \cap c_2$=011. Кубы c_1 и c_3 общих вершин не имеют, и их пересечение $c_1 \cap c_3=\varnothing$. Кубы c_2 и c_3 имею общую вершину 111, и их пересечение $c_2 \cap c_3$=111.

Алгебраически операция пересечения кубов сводится к покоординатному пересечению.

Пусть заданы два куба: $A=(a_1, a_2, \ldots, a_i, \ldots, a_n)$ и $B=(b_1, b_2, \ldots, b_i, \ldots, b_n)$, где a_i, b_i могут принимать значения 0, 1 и ×. Пересечение координат $a_i \cap b_i$ можно представить в виде таблицы 2.5.

Таблица 2.5.

$a_i \setminus b_i$	0	1	×
0	0	*y*	0
1	*y*	1	1
×	0	1	×

Здесь пустое пересечение $a_i \cap b_i = 0 \cap 1 = 1 \cap 0 = \varnothing$ обозначено буквой *y*.

Заметим, что из топологии кубов следует пустое пересечение кубов $A \cap B=\varnothing$, если существует одна или более координат, по которым зафиксировано значение *y*.

Итак, пересечение кубов можно записать следующей формулой:

$$A \cap B = \begin{cases} \varnothing, & \text{если} \quad \exists (a_i \cap b_i) = y, \\ (a_1 \cap b_1), \ldots, (a_i \cap b_i), \ldots, (a_n \cap b_n). \end{cases}$$

Операция пересечения обладает следующими свойствами:

$A \cap B = B \cap A$ – коммутативность и

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ – дистрибутивность.

Из первого свойства следует, что порядок кубов, вступающих в пересечение, безразличен, а из второго – правило пересечения кубов с множеством кубов, образующих покрытие.

Операция вычитания кубов. Операция вычитания кубов является топологической операцией и обозначается специальным символом «#», чтобы отличать ее от разности множеств. Операция $A \# B$ состоит в том, что из куба A удаляется их общая часть – грань с кубом B, если таковая существует. В общем случае, в результате такого вычитания образуется множество остатков – граней куба A. Операцию вычитания в алгебро-топологической форме можно выразить через операцию пересечения кубов $A \cap B$.

Если $A \cap B = \varnothing$, то $A \# B = A$. Если $A \subseteq B$ и $A \cap B = A$, то результат вычитания кубов дает пустое множество, т.е. $A \# B = \varnothing$.

Если предыдущие условия не выполняются, то происходит объединение остатков куба A в множество по соотношению координат кубов A и B. Если заданы два куба $A=(a_1, a_2, \ldots, a_i, \ldots, a_n)$ и $B=(b_1, b_2, \ldots, b_i, \ldots, b_n)$ и, соответственно, зафиксированы значения $a_i=\times$ и $b_i=p$, где $p=0 \vee 1$, то размножение куба A происходит по всем таким соотношениям путем замены координаты $a_i=\times$ на значение $\overline{p}$ координаты b_i. Следовательно, такие грани-осколки будут зафиксированы для соотношений $a_i=\times$ и $b_i=p$. Эти грани объединяются между собой в единое множество граней-осколков граней куба A.

Операцию вычитания кубов $A \# B$ можно записать в виде следующей формулы:

$$A \# B = \begin{cases} A, \text{ если } A \cap B = \varnothing, \\ \varnothing, \text{ если } A \subseteq B, \\ \bigcup_{i=1\ldots n}(a_1, \ldots, a_{i-1}, \overline{p}, a_{i+1}, a_n), \text{ если } a_i=\times \text{ и } b_i=p. \end{cases}$$

Проиллюстрируем операцию вычитания на примере.

Пусть задано множество кубов: c_1, c_2, c_3 и c_4.

$$K(f) = \begin{bmatrix} 0 \times 0 \\ 0 \times \times \\ \times 1 1 \\ 1 \times 1 \end{bmatrix} \begin{matrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{matrix}$$

Видно, что у кубов c_2 и c_4 нет общих частей, поэтому $c_2 \# c_4 = c_2$.

Куб c_1 содержится в кубе c_2, и поэтому $c_1 \# c_2 = \varnothing$.

Кубы c_2 и c_3 покрывают общую вершину 011, которая равна пересечению $c_2 \cap c_3$. Куб $c_2 = 0\times\times$, поэтому вторую и третью координаты куба c_2 надо заменить на $\overline{p} = 0$. В результате $c_2 \# c_3 = \{00\times\} \cup \{0\times0\}$.

Укажем два свойства операции вычитания кубов:
$A\#B \neq B\#A$ и $A\#(B \cup C) = (A\#B)\#C$. Первое свойство свидетельствует о том, что операнды нельзя менять местами, а второе свойство показывает правило вычитания некоторого множества кубов из заданного путем их последовательного вычитания. Отметим также, что $(A \cup B)\#C = (A\#C) \cup (B\#C)$. Это правило распространяет операцию вычитания на множества кубов, заданных, например, в виде покрытий, между которыми следует установить наличие или отсутствие эквивалентности при их верификации между собой.

Контрольные вопросы по теме «Булева алгебра»

1 Записать правила склеивания. Доказать одно из них.

2 Что такое конституента единицы? Для какой цели они используются? Привести пример конституенты единицы для функции от четырех переменных. Какие значения принимает эта конституента на различных наборах аргументов булевой функции?

3 Действует ли дистрибутивный закон в отношении операции конъюнкции и сложения по модулю 2 $(a\cdot(b\oplus c)=a\cdot b\oplus a\cdot c)$? Ответ обосновать.

4 Привести аналитическое выражение булевой функции

5 $y=f^3(x)=(x_1\overline{x}_3)\downarrow\overline{(\overline{x}_1\vee\overline{x}_2)\cdot(\overline{x}_1\mid x_3}$ к нормальным формам: ДНФ и КНФ. Преобразовать полученные нормальные формы к каноническим (КДНФ и ККНФ), не пользуясь таблицей истинности.

6 Сколько 3-кубов накрывается одним 5-кубом? Ответ обосновать.

7 Найти существенные импликанты булевой функции от трех переменных, заданной в числовой форме $f^3(x)$=&(2,3,4).

8 Привести пример минимальной КНФ булевой функции от трех переменных, составленной по нулевому покрытию, для которого цена S^a=5, а цена S^b=8.

9 Привести пример булевой функции от четырех переменных, для которой минимальная КНФ совпадает с ККНФ, а минимальная ДНФ не совпадает с КДНФ и число существенных вершин равно 12. Функцию представить в числовой форме. Найти минимальную ДНФ этой функции.

10 Что понимается под монотонной булевой функцией? Является ли функция равнозначности $(y = x_1 \sim x_2)$ монотонной? Утверждение обосновать.

11 Что понимается под функцией, сохраняющей константу ноль? Перечислить все базовые функции булевой алгебры, не сохраняющие константу ноль.

12 Дополнить функцию импликации $(y = x_1 \rightarrow x_2)$ минимальным количеством других булевых функций так, чтобы полученная система была функционально полной, но не избыточной. Доказать функциональную полноту этой системы, используя теорему Поста – Яблонского и/или конструктивный подход.

13 Булева функция от четырех переменных принимает значение, равное 1, на наборах (1, 4, 13, 15) и безразличное значение – на наборах (6, 9, 10, 12). Найти минимальную КНФ этой функции.

2.3. Теория графов

Теория графов – один из самых распространенных математических инструментов, широко используемый в различных областях деятельности человека:

1. В микроэлектронике – при разработке топологии печатных плат и микросхем.
2. При разработке сложных электрических схем и схем их монтажа.
3. В химии – при разработке новых сложных молекулярных соединений.
4. В физике – при описании и анализе схем развития квантовых процессов.
5. При разработке коммуникационных систем различного назначения.
6. В транспортных системах – как для изучения самих систем, так и при составлении оптимальных маршрутов доставки грузов – логистика.
7. В биологии – ветвящийся процесс – размножение бактерий.
8. В психологии – для представления промежуточных и окончательных результатов теоретических и экспериментальных исследований.
9. В информатике и программирование – при разработке алгоритмов программ.
10. В экономике и планировании – в виде сетевых графиков

и т.д.

Такое широкое использование теории графов обусловлено тем, что с помощью графов можно описывать разнообразные реальные явления: организацию транспортных систем, сети передачи данных, человеческих взаимоотношений, структуру гена или молекулы. Возможность формального моделирования такого множества разных реальных структур позволяет программисту решать широкий круг прикладных задач.

Вместе с тем, проектирование эффективного алгоритма «с нуля» – очень трудная задача. Поэтому часто достаточно правильно построить модель задачи и применить уже известный, проверенный алгоритм, решающий задачу быстро и верно.

В рамках данного пособия рассматриваются основные понятия теории графов, необходимые решения задач верификации.

2.3.1. Основные понятия теории графов

Теория графов – дисциплина математическая, созданная усилиями математиков, поэтому ее изложение включает в себя и необходимые строгие определения.

Определение. Под ***графом G(X, U)*** понимают совокупность непустого множества X и подмножества U, представляющего собой множество всех упорядоченных пар (x_i, x_j), где $x_i, x_j \in X$. Элементы множеств X и U называют, соответственно, ***вершинами*** **и** ***дугами*** **(*ребрами*)** графа.

Определение. Вершины графа, которые не принадлежат ни одному ребру, называются ***изолированными*.**

Определение. Граф, состоящий только из изолированных вершин, называется ***нуль-графом*.**

Определение. Граф, в котором каждая пара вершин соединена ребром, называется ***полным*.**

K_m – полный граф с m вершинами.

Определение. Вершины, принадлежащие одному ребру, называются ***смежными*.** Аналогично, ***смежными*** называются ребра, принадлежащие одной вершине.

Определение. Вершина, являющаяся началом или концом некоторого ребра, называется ***инцидентной*** этому ребру. Аналогично, ребро ***инцидентно*** вершине, которая является его началом или концом.

Определение. ***Степенью вершины*** называется число ребер, инцидентных ей. $\rho(x_i)$ – степень вершины x_i. Степень изолированной вершины равна 0.

Определение. Граф, степени всех вершин которого одинаковы и равны k, называется ***однородным*** графом степени k**.**

Определение. ***Дополнением*** данного графа называется граф, состоящий из всех ребер и их концов, которые необходимо добавить к исходному графу, чтобы получить полный граф.

Определение. Граф, который можно представить на плоскости в таком виде, когда его ребра пересекаются только в вершинах, называется ***планарным***.

Определение. Многоугольник плоского графа, не содержащий внутри себя никаких вершин или ребер графа, называют его ***гранью*.**

Определение. Последовательность ребер $u_k \in U$, заданных парами вершин вида (x_0, x_1) (x_1, x_2) ... (x_{l-1}, x_l), в которой любые два соседних ребра смежные, называется ***маршрутом*.**

Определение. Число ребер в маршруте определяет его длину.

Определение. ***Циклом*** называют последовательность ребер $u_1=(x_1, x_i)$, ..., $u_k=(x_j, x_1)$, при которой в результате обхода вершин графа по этим ребрам возвращаются в исходную вершину x_1.

Каждое ребро графа встречается в цикле не более одного раза, в то время как вершины могут повторяться и несколько раз.

Определение. ***Простым циклом*** называется цикл, не проходящий ни через одну из вершин графа более одного раза.

Определение. Простой цикл, в котором содержатся все ребра графа, называется ***эйлеровым циклом***.

Определение. Цикл называют ***гамильтоновым,*** если он проходит через каждую вершину один раз.

Определение. Две вершины x_i и x_j в графе называются ***связными*** (***несвязными***), если в нем существует (не существует) путь, ведущий из x_i в x_j.

Определение. Граф называется ***связным,*** если каждые две его вершины связны; если же в графе найдется хотя бы одна пара несвязных вершин, то граф называется ***несвязным.***

Определение. ***Деревом*** называется связный граф, не содержащий циклов.

Определение. Несвязный граф, состоящий исключительно из деревьев, называется ***лесом***.

Определение. Два графа G и G' ***изоморфны,*** если они имеют одинаковое число вершин и если каждой паре вершин, соединенных ребром (дугой), в одном графе соответствует такая же пара вершин, соединенных ребром (дугой), в другом графе.

Определение. Если в графе $G(X, U)$ опущены некоторые ребра, а число вершин осталось прежним, то полученный граф $G(X, U')$ называют ***частичным графом G(X, U)*** или ***суграфом***.

Определение. Если в графе $G(X, U)$ опущены некоторые ребра и инцидентные им вершины, то полученный граф $G(X', U')$ называют ***подграфом G(X,U).***

Определение. Граф называется ***двудольным***, если существует такое разбиение множества его вершин на две части, что концы каждого ребра принадлежат разным частям.

Определение. ***Мультиграфом*** называется граф, в котором есть кратные ребра, т.е. пары вершин могут соединяться более чем одним ребром.

Определение. Граф, имеющий и кратные ребра, и петли, называется ***псевдографом***.

2.3.2. Способы представления графа

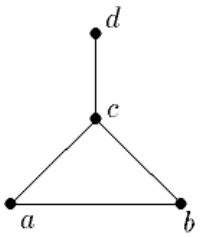
Выбор способа представления графа имеет важное значение при разработке эффективных алгоритмов. При решении задач используются следующие способы представления графа $G = (V, U)$:

1. Явное задание графа.
2. Геометрический.
3. Матрица смежности.
4. Матрица инцидентности.

1. Явное задание графа состоит в перечислении вершин, смежных каждой вершине.

$\Gamma x_i = \{x_a, x_b, \ldots, x_z\}$, где Γx – отображение множества вершин X на множество вершин X, а вершины $x_a, x_b, \ldots, x_z$ – вершины, смежные вершине x_i.

2. Геометрическое задание:



Для этого графа явное задание:

$\Gamma a = \{b, c\}$; $\Gamma b = \{a, c\}$; $\Gamma c = \{a, b, d\}$; $\Gamma d = \{c\}$.

3. Матрица смежности – квадратная матрица

$A = ||a_{ij}||_{m \times m}$, где $m = |X|$, число вершин.

$$a_{ij} = \begin{cases} 1, \text{ если вершина } x_i \text{ смежна вершине } x_j, \\ 0, \text{ в противном случае.} \end{cases}$$

4. Матрица инцидентности – прямоугольная матрица

$B = ||b_{ij}||_{m \times n}$, где $m = |X|$, число вершин, $n = |U|$, число ребер.

$$b_{ij} = \begin{cases} 1, \text{ если вершина } x_i \text{ инцидентна ребру } u_{ij}, \\ 0, \text{ в противном случае.} \end{cases}$$

Для рассматриваемого примера матрицы смежности и инцидентности:

$A =$

	a	b	c	d
a	0	1	1	0
b	1	0	1	0
c	1	1	0	1
d	0	0	1	0

$B =$

	u_{ab}	u_{ac}	u_{bc}	u_{cd}
a	1	1	0	0
b	1	0	1	0
c	0	1	1	1
d	0	0	0	1

Геометрическое представление графа мало приемлемо для использования ЭВМ. Для проведения формальных исследований удобны матричные формы представления графа.

Явное задание графа обеспечивает компактное описание разреженных графов, т.е. таких, для которых число ребер много меньше квадрата числа вершин.

Представление графа при помощи матрицы смежности удобнее для плотных графов, т.е. таких, у которых число ребер близко к квадрату числа вершин. Кроме того, достоинством представления с помощью матрицы смежности является возможность быстрого определения смежности любых пар вершин.

Явное представление требует объем памяти, равный $O(m + n)$.

Для представления графа в виде матрицы смежности требует объем памяти, равный m^2, независимо от количества ребер графа.

Матрица смежности легко адаптируется для представления мультиграфов и взвешенных графов, т. е. графов, с каждым ребром которых связан определенный вес, обычно определяемый весовой функцией *w*. Для мультиграфа w_{ij} – кратность ребра. В этом случае в каждый элемент матрицы смежности

$$a_{ij} = \begin{cases} w_{ij}, \text{ если вершина } x_i \text{ смежна вершине } x_j, \\ 0, \text{ в противном случае.} \end{cases}$$

2.3.3. Характеристические числа графов

Рассмотрим некоторые характеристические числа графа, не зависящие от изоморфных преобразований. Такие числа называют инвариантами графа, т.е. у изоморфных графов они равны.

Цикломатическое число. Цикломатическое число графа указывает то число ребер, которое нужно удалить из данного графа, чтобы получить дерево (для связного графа) или лес (для несвязного графа), т.е. добиться отсутствия у графа циклов.

Пусть связный граф $G(X, U)$ имеет $n = |X|$ вершин и $r = |U|$ ребер. Тогда дерево, построенное на этом графе, имеет $|W| = n - 1$ ребро. По определению цикломатическое число $\nu(G) = r - (n - 1) = r - n + 1$. Для несвязного графа с *p* компонентами связности цикломатическое число

$\nu(G) = r - n + p$.

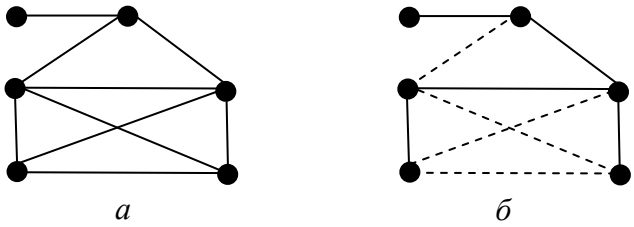
Цикломатическое число всегда неотрицательно.

Основное свойство цикломатического числа формулируется в виде теоремы:

Цикломатическое число графа равно максимальному числу независимых циклов.

Знание цикломатического числа оказывается полезным при анализе топологии электронных схем, а также для решения целого класса задач конструкторского проектирования ЭВМ.

Пусть дан граф (рис. 2.8 *а*):



а *б*

Рис. 2.8.

У графа $n = 6$ вершин и $r = 9$. Для заданного графа цикломатическое число $\nu(G) = r - n + 1 = 9 - 6 + 1 = 4$ (рис. 2.8. *б,* пунктиром обозначены удаленные ребра).

Хроматическое число. Пусть задан граф $G(X, U)$ без петель. Разобьем множество его вершин на k непересекающихся подмножеств $X_1, X_2, ..., X_k$, $X = \bigcup_{i=1}^{k} X_i, \forall X_i, X_j \subset X \ [X_i \cap X_j = \varnothing]$ так, чтобы любые две смежные вершины x_i и $x_j \in X$ принадлежали разным подмножествам, т.е. чтобы ребра графа $G(X, U)$ соединяли вершины из разных подмножеств ($\forall X_s \subset X \ [\Gamma X_s \cap X_s = \varnothing]$, где ΓX_s множество вершин, смежных вершинам множества X_s).

Задача раскраски вершин графа формулируется следующим образом. Необходимо раскрасить вершины графа таким образом, чтобы смежные вершины были окрашены в разные цвета. Минимальное число красок, в которые можно раскрасить граф, называется *хроматическим числом* графа и обозначается $\chi(G)$, а граф $G(X, U)$ называют χ-*хроматическим*.

Особое значение имеет частный вид χ-хроматического графа – *бихроматический граф*, для которого множество вершин X можно разбить на два непересекающихся подмножества X_1 и X_2 так, чтобы ребра соединяли вершины разных подмножеств ($X = X_1 \cup X_2$, $X_1 \cap X_2 = \varnothing$, $\forall x_i \in X_1 \ [\Gamma x_i \cap X_1 = \varnothing]$, $\forall x_i \in X_2 \ [\Gamma x_i \cap X_2 = \varnothing]$.

Такие графы называют бихроматическими, двудольными графами или графами Кенига и обозначают $G(X_1, X_2, U)$.

В отличие от цикломатического числа, определение хроматического числа осуществляется с помощью сравнительно сложных алгоритмов, в основу большинства которых положены методы целочисленного линейного программирования.

Хроматическое число графа на рисунке 2.8 *а* $\chi(G) = 4$.

Число планарности. Граф $G(X, U)$ называют ***плоским*** тогда и только тогда, когда он геометрически реализован на плоскости так, что все его ребра пересекаются только в вершинах X графа. Граф $G(X, U)$ называют ***планарным***, если он может быть геометрически реализован на плоскости без пересечения ребер. Т.е. плоскостность – это свойство его геометрической реализации на плоскости, а планарность – свойство графа быть реализованным на плоскости.

Минимальное число ребер, которое нужно удалить из графа, чтобы он стал планарным, называется ***числом планарности*** и обозначается $\Theta(G)$. Для полного графа K_n с $n \geq 4$ $\Theta(G) = (n - 3)(n - 4)/2$.

Минимальное число плоских суграфов, на которые разбивается граф $G(X, U)$ называется толщиной графа $t(G)$.

Число планарности графа на рисунке 2.8 *а* $\Theta(G) = 1$. Толщина графа $t(G)=2$.

Число внутренней устойчивости. Множество вершин X_s графа $G(X, U)$ называется ***внутренне устойчивым*** (***независимым***), если никакие две вершины из этого множества не смежны, $X_s \subset X \ [\Gamma X_s \cap X_s = \varnothing]$. Внутренне устойчивое множество называется ***максимальным***, если оно не является собственным подмножеством некоторого другого независимого множества.

Максимальное по мощности независимое множество называется ***наибольшим***. Число вершин в наибольшем независимом множестве графа $G(X, U)$ называется ***числом внутренней устойчивости*** этого графа и обозначается $\alpha(G)$, $\alpha(G)=\max \left| X_s \right|$.

Заметим, что при раскраске графа множество вершин, окрашенных в один цвет – внутренне устойчивое.

Число внешней устойчивости. Множество вершин X_s графа $G(X, U)$ называется ***внешне устойчивым*** (***доминирующим***), если каждая вершина из $X \backslash X_s$ смежна с некоторой вершиной из X_s. Иначе говоря, каждая вершина графа $x_j \in X_s$ или $x_j \in \Gamma X_s$, т.е. находится на расстоянии не более 1 от внешне устойчивого множества, $\Gamma X_s \cup X_s = X$.

Внешне устойчивое множество X_s называется ***минимальным***, если при удалении из него любой вершины получается множество, не являющееся внешне устойчивым. Внешне устойчивое множество, состоящее из минимального числа вершин, называется ***наименьшим***. Число вершин в наименьшем независимом множестве графа $G(X, U)$ называется ***числом внешней устойчивости*** этого графа и обозначается $\beta(G)$, $\beta(G) = \min \left| X_s \right|$.

Для графа на рисунке 2.8 *а* числа внутренней и внешней устойчивости равны $\alpha(G) = \beta(G) = 2$. Подмножество вершин графа, являющееся как внутренне устойчивым, так и внешне устойчивым, называется ***ядром.***

Плотность графа. Максимальный полный подграф графа $G(X, U)$ называется кликой графа G; другими словами, клика графа G есть подмножество его вершин, такое, что между каждой парой вершин этого подмножества существует ребро и, кроме того, это подмножество не принадлежит никакому большому подмножеству с тем же свойством. Число вершин клики графа называется ***плотностью графа*** и обозначается $f(G)$.

Плотность графа на рисунке 2.8 *а* $f(G)=4$.

Число Хадвигера. Операцией сжатия графа (стягивание графа) называется замена двух смежных вершин x_i и x_j одной x_s с удалением ребра u_{ij}. ***Числом Хадвигера*** $\eta(G)$ графа $G(X, U)$ называется такое наибольшее число η, что граф G стягиваем к полному графу K_η.

Число Хадвигера графа на рисунке 2.8 *а* $\eta(G)=4$.

Контрольные вопросы
по теме «Теория графов»

1. Определение графа. Различные типы графов.
2. Матричные способы задания графов.
3. Изоморфизм графов.
4. Маршруты, циклы в графе.
5. Эйлеров цикл.
6. Гамильтонов цикл.
7. Связность графа. Матрица связности.
8. . Деревья. Остовные деревья.
9. Минимальное остовное дерево.
10. Цикломатическое число графа.
11. Хроматическое число графа.
12. Число планарности.
13. Число внутренней устойчивости.
14. Число внешней устойчивости.
15. Плотность графа.
16. Число Хадвигера.

2.4. Элементы теории конечных автоматов

Предметом теории автоматов является изучение математических моделей преобразователей дискретной информации. В данной теории решаются следующие основные задачи: анализ и синтез автоматов, минимизация и эквивалентные преобразования автоматов. Дадим краткую формулировку каждой из перечисленных задач.

Задача анализа. По заданному автомату описать его поведение. Вариант постановки: по неполному описанию автомата установить некоторые его свойства.

Задача синтеза. Построить автомат с наперед заданным поведением (алгоритмом функционирования).

Задачу синтеза принято рассматривать двояко: абстрактный синтез как построение математической модели автомата и структурный синтез как разработку функциональной логической схемы автомата.

Задача минимизации. Построить автомат, минимальный заданному. Минимальный автомат обладает наименьшим числом компонентов модели (в частности, минимальной мощностью множества так называемых состояний) и при этом функционально эквивалентен заданному автомату.

Задача эквивалентных преобразований. Определить полную систему правил, позволяющую преобразовывать произвольный автомат в любой эквивалентный ему автомат. Частным случаем данной задачи является переход от одной модели автомата к другой. Два автомата функционально эквивалентны, если их поведение одинаково при воздействии одних и тех же последовательностей входных сигналов. В таком случае говорят о совпадении моделей поведения двух автоматов.

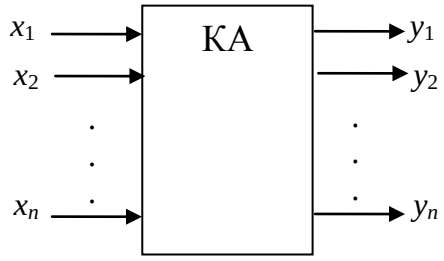
2.4.1. Понятие конечного автомата

Поведение многих технических объектов можно описать так называемой конечно-автоматной моделью. Согласно этой модели, объект может находиться в одном из конечного множества состояний. Под воздействием входных сигналов (команд, событий) он переходит в другое состояние. При этом он вырабатывает некий выходной сигнал. Иначе говоря, конечный автомат — математическая модель поведения устройств с конечной памятью. Конечные автоматы применяются, например, в системах синтаксического анализа и перевода текста, компьютерных играх, системах управления сложными техническими устройствами и др.

Конечный автомат (finite-state machine) — абстрактный автомат, число возможных внутренних состояний которого конечно.

Общую схему конечного автомата можно представить в виде какого-либо «черного ящика», преобразующего множество входных сигналов X во множество выходных сигналов Y (рис. 2.9).

x_2



2.9. Общая схема конечного автомата

Алгоритм функционирования конечного автомата можно задать различными способами. Конечный автомат может быть задан в виде кортежа пятерки элементов некоторых множеств:

$M = \{Q, X, Y, \Psi, \Phi\}$,

где Q — *множество внутренних состояний*;

X — *входной алфавит* (конечное множество *входных символов*), из которого формируются *входные слова*, воспринимаемые конечным автоматом;

Y — *выходной алфавит* (конечное множество *выходных символов*), из которого формируются *выходные слова*;

Ψ — *функция переходов*, определенная как отображение $Q \times X \to Q$, то есть каждой паре «состояние – входной сигнал» ставит в однозначное соответствие определенное состояние из множества Q;

Φ — *функция выходов*, определенная как отображение $Q \times X \to Y$, то есть каждой паре «состояние – входной сигнал» ставит в однозначное соответствие конкретный выходной сигнал – элемент множества Y

При этом среди множества $Q = \{q_0, q_1, \ldots, q_n\}$ необходимо выделить начальное состояния q_0, в котором автомат находится в момент времени $t = 0$.

Принято полагать, что конечный автомат начинает работу в состоянии q_0, последовательно считывая по одному символу входного слова (цепочки входных символов). Считанный символ переводит автомат в новое состояние в соответствии с функцией переходов. Читая входную цепочку символов и делая переходы из состояния в состояние, автомат после прочтения последнего символа входного слова окажется в некотором состоянии q_n.

Если это состояние является заключительным, то говорят, что автомат допустил слово x.

Конечные автоматы широко используются на практике, например, в синтаксических и лексических анализаторах, тестировании программного обеспечения на основе моделей в играх и др.

2.4.2. Способы задания конечного автомата

Автомат можно представить одним из трех эквивалентных способов: ***табличный***, ***геометрический*** и ***функциональный***.

Табличный способ. Для задания автомата составляется таблица состояний автомата для функции перехода – Ψ и функции выхода – Φ:

- столбцы таблицы соответствуют элементам входного алфавита – X,
- ·строки таблицы соответствуют состояниям (элементам конечного множества Q).

Пересечению i-й строки и j-го столбца соответствует клетка (i, j), которая является аргументом функций Ψ и Φ автомата. В ней записывается состояние автомата, в которое он переходит в момент, когда он находится в состоянии q_i на его входе – слово x_j, и значение выходного сигнала из множества Y. Таким образом, вся таблица переходов соответствует множеству $Q \times X$.

При заполнении таблицы переходов каждая клетка однозначно определяется парой символов: символом следующего состояния и символом выходного сигнала.

Рассмотрим конечный автомат $M = \{Q, X, Y, \Psi, \Phi\}$, где
$Q = \{1, 2, 3\}$; $X = \{a, b\}$; $Y = \{a, b, c\}$, а команды автомата следующие:
1) $1a \to 3b$, 2) $1b \to 2c$, 3) $2a \to 3a$, 4) $2b \to 3c$, 5) $3a \to 1b$, 6) $1b \to 3c$.

Таблица переходов данного автомата приведена в табл. 2.6.

Таблица 2.6.

$\Psi(q, x) / \Phi(q, x)$

q	x	
	a	b
1	3 / b	2 / c
2	3 / a	3 / c
3	3 / b	3 / c

Графический способ. При задании автомата удобно пользоваться элементами теории графов. Автомат задается в виде ориентированного мультиграфа.

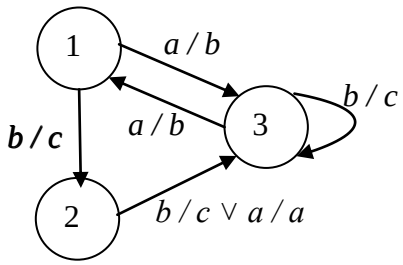
Граф автомата содержит:

- Вершины, которые соответствуют состоянию $q_i \in Q$,
- Дуги, соединяющие вершины – переходы автомата из одного состояния в другое. На дугах принято указывать пары входных и выходных сигналов – сигналов переходов ($x_i \in X$ и $y_i \in Y$).

Если автомат переходит из состояния q_1 в состояние q_2 под воздействием нескольких входных сигналов, то на соответствующей дуге графа этот вариант будет представлен через дизъюнкцию.

На рис. 2.10 показан граф автомата, заданного табл. 2.6.

Пусть в некоторый момент времени t автомат находится в состоянии 1 и на его вход подается последовательность входных сигналов baa, автомат перерабатывает ее в выходную последовательность cab, переходя в моменты времени $t + 1$ в состояние 2, $t + 2$ в состояние3 и $t + 3$ в состояние 1.



2.10. Граф автомата

Аналитический способ – автомат задается системой уравнений. Из такой системы следует, что при конечном числе возможных внутренних состояний количество возможных значений автоматных функций также оказывается конечным.

Функционирование автомата в дискретные моменты времени t может быть описано системой рекуррентных соотношений:

$$\begin{cases} q[t+1] = x[t]); \\ y[t] = x[t]. \end{cases}$$

2.4.3. Классификация абстрактных автоматов

Классификацию абстрактных автоматов принято проводить по трем признакам, а именно:

- определенность функции переходов и функции выходов;
- однозначность заданных функций;
- устойчивость состояний.

I. **По определенности характеристических функций.** В полностью определенных автоматах областью определения функций Ψ и Φ является множество всех пар $(q_i, x_j) \in Q \times X$, где $q_i \in Q$, $x_j \in X$. В частично определенных автоматах либо обе характеристические функции, либо одна из них имеют областью определения строгое подмножество декартова произведения $Q \times X$. Таким образом, характеристические функции подобных автоматов определены не для всех пар (q_i, x_j).

II. **По однозначности функции переходов**. В детерминированных автоматах выполняется условие однозначности переходов: если абстрактный автомат находится в некотором состоянии $q_i \in Q$, то под воздействием произвольного входного сигнала $x_j \in X$ автомат может перейти в одно и только одно состояние $q_k \in Q$, причем ситуация $q_i = q_k$ не исключается. В автоматах недетерминированных при воздействии одного и того же входного сигнала возможны переходы из состояния q_i в различные состояния из множества Q с заданной вероятностью.

Т.е., детерминированный автомат реализует конструкцию: если, a, то b. А недетерминированный – если a, то b или c, или …

III. **По устойчивости состояний.** В устойчивых автоматах выполняется условие устойчивости: если автомат под воздействием входного сигнала $x_j \in X$ оказался в состоянии $q_i \in Q$, то выход из него и переход в иное состояние возможен только при поступлении на вход автомата другого сигнала $x_k \in X$, $x_j \neq x_k$. Если условие устойчивости не выполняется хотя бы для одного состояния $q_i \in Q$, то такой автомат называют неустойчивым.

На рис. 2.11 приведена классификация автоматов по этим признакам.

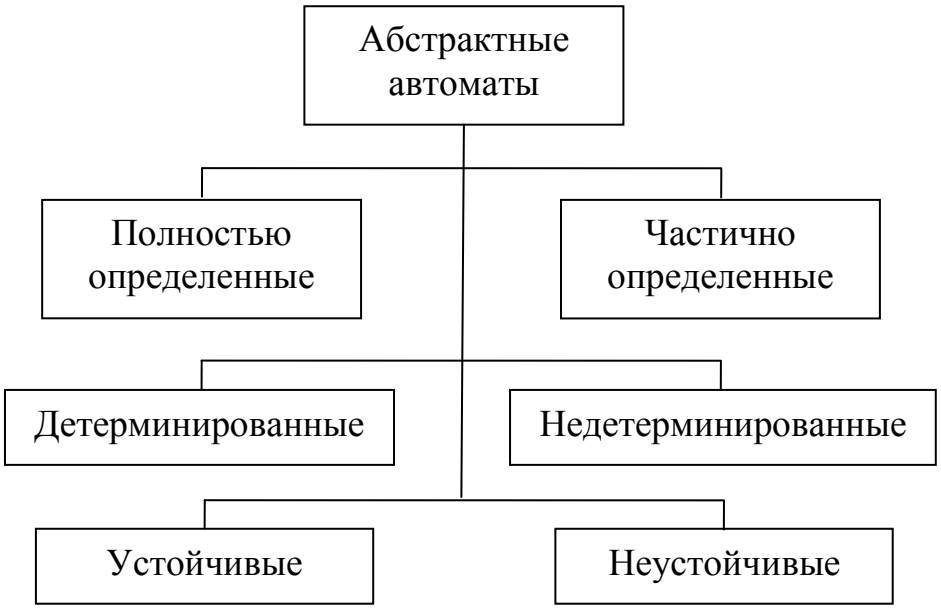


Рис. 2.11. Классификация абстрактных автоматов

2.4.5. Автоматы Мили и Мура

Различный характер функций переходов и выходов для различных автоматов позволяет выделить отдельные типы автоматов в классе конечных детерминированных автоматов.

Основными являются две модели: автоматы Мили и Мура.

В автомате Мили функцию выходов двухаргументная, а символ в выходном канале находится лишь при наличии символа во входном канале. Схема функциональная не имеет отличий от схемы абстрактного автомата (рис. 2.12).

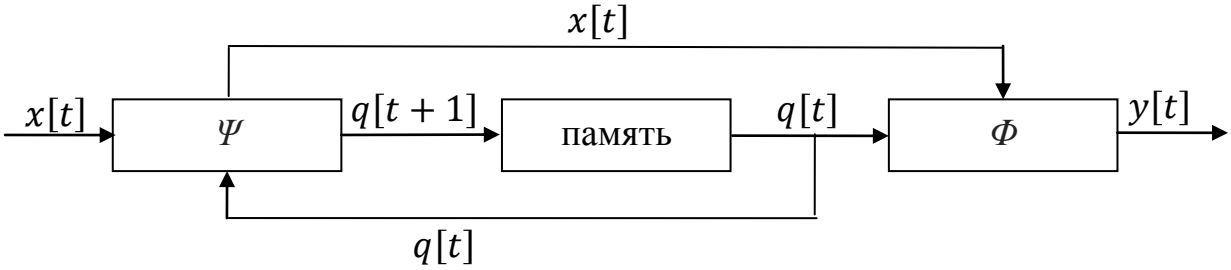


Рис. 2.12. Функциональная схема автомата Мили

Автомат Мура обладает функцией выходов, определяющей значение выходного символа лишь по одному аргументу - состоянию автомата. Таким образом в автомате Мура, в отличие от автомата Мили, представлена зависимость выходного сигнала только от состояния (рис. 2.13).

Рис. 2.13. Функциональная схема автомата Мура

Контрольные вопросы
по теме «Теория конечных автоматов»

1. Какой круг задач решается в теории автоматов?
2. Классифицируйте абстрактные автоматы по трем признакам.
3. Что такое конечный автомат?
4. Каковы характеристические функции конечного автомата?
5. Какие существуют способы задания конечного автомата?
6. В чем сходство и различие базовых моделей конечных автоматов?
7. Какой автомат называется детерминированным?
8. Чем различаются автоматы Мили и Мура?

3. ТЕСТИРОВАНИЕ ЦИФРОВЫХ СХЕМ

В процессе изготовления и эксплуатации схем цифровых устройств в них могут возникать различные дефекты: обрывы, короткие замыкания, пробои диодов и т. п. Физические дефекты аппаратуры приводят к неисправностям, которые изменяют логику работы элементов и схем. Логические неисправности проявляются при контроле или в процессе эксплуатации в виде возникновения неверных логических значений на внешних выходах устройства. Синтез контролирующих и диагностирующих тестов и тестирование являются составной частью функциональной верификации вычислительных процессов.

Поиск неисправностей в цифровых устройствах осуществляется подачей на входы специально подобранных входных воздействий и наблюдением за реакцией схемы. Эти воздействия, сопровождаемые эталонной реакцией схемы, называются тестами устройства. Задача контроля и диагностики цифровой аппаратуры заключается в определении состояния устройства по его реакции на тестовые воздействия.

Устранение физических дефектов, вызвавших зафиксированные и найденные неисправности, составляет задачу ремонта.

Представляется целесообразной такая последовательность работ при тестировании схем:

- контроль, фиксирующий наличие или отсутствие неисправностей в схеме устройства;
- диагностика, состоящая в поиске места неисправности и ей соответствующего дефекта;
- ремонт, сводящийся к устранению зафиксированного дефекта цифровой аппаратуры.

Под неисправностью будем понимать любое искажение логики работы элементов схемы. Например, дефект в виде обрыва проводника в схеме может приводить к появлению постоянного значения 0 или 1 на входах или выходах элементов. Дефект в виде короткого замыкания проводников может вызывать появление ложных монтажных элементов И или ИЛИ. Основным классом неисправностей являются неисправности константы 0 или 1 на входах или выходах элементов.

Некоторая неисправность m^0– константа 0, а m^1– константа 1, из множества возможных неисправностей схемы является существенной неисправностью, если существует входное воздействие или последовательность входных воздействий, на которых эта неисправность проявляется в виде ошибки на внешних выходах схемы. Часть неисправностей в схемах может быть несущественной из-за различного рода избыточности, специально заложенной в аппаратуре.

Поиск тестов, т. е. входных воздействий, на которых неисправность схемы проявляется в виде искажений реакции схемы, имеет смысл только для существенных неисправностей.

В качестве теста последовательностной схемы N будем рассматривать некоторую сегментированную последовательность наборов $T = \{T^j\}$, в которой для любой существенной неисправности m^p найдётся сегмент T^j, на котором эта неисправность проявляется в виде ошибки. Любой сегмент теста имеет следующую структуру: $T^j = P^j L^j$.

Здесь P^j – проверяющая подпоследовательность, осуществляющая «транспортировку» сигнала–ошибки от места неисправности (внутри схемы) до одного из внешних выходов схемы в соответствии с ее тактами работы. Подпоследовательность L^j является установкой, т. е. она вырабатывает требуемое для P^j внутреннее состояние схемы путем рекурсивного перевычисления состояний схемы из некоторого исходного состояния схемы, для которого значение всех сигналов обратной связи являются неопределёнными, т. е. равными ×.

Задача синтеза тестов является составной частью систем автоматизации проектирования. В подсистеме проектирования тестов можно выделить следующие этапы:

- выделение сигналов обратной связи и разбиение схемы на подсхемы;
- построение покрытий для подсхем;
- активизация отрезков путей в подсхемах по покрытиям;
- синтез проверяющих последовательностей для активизаций путей из активных отрезков в соответствии с тактами работы схемы;
- решение установочной задачи путем движения по графу переходов схемы в обратном направлении.

3.1. Основные операции и алгоритмы

3.1.1. Построение покрытий логических схем по π – алгоритму

π – алгоритм относится к прямым методам анализа логических схем. Он представляет собой процедуру построения покрытия значений (множество входных наборов) на входах схемы (независимых переменных), порождающих на выходе схемы заданное значение булевой функции $f = 1$ – покрытие $C(f)$ или значение $f = 0$ – покрытие $C(\bar{f})$. Иначе говоря, он позволяет определить множество входных наборов, порождающих на выходе схемы заданное логическое значение.

Логическая схема задаётся в виде элементов, описываемых вырожденными покрытиями в соответствии с их типами и координатами по входам и связей между ними. Процедура носит переборный характер и состоит в подстановке примитивов кубов из покрытия элементов путём движения от выхода схемы к её входам. В процессе движения выполняются операции пересечения, объединения и удаления пустых и поглощаемых кубов. Процедура заканчивается, когда все возможные подстановки выполнены, т.е. перебраны все пути сигналов в схеме и их композиции. Для упорядочения перебора все независимые входы, выход и элементы схемы

идентифицируются (обозначаются), например, целыми числами или переменными с индексами, образующими систему координат покрытия.

Отображение структуры схемы в покрытиях осуществляется с помощью эквивалентных координат. Т.о. каждому пути распространения сигнала в схеме будет соответствовать своя координата в эквивалентном покрытии,

Результат построения покрытия оформляется либо в виде таблицы решений или списков элементов, вступающих в операции объединения и/или пересечения кубов. Рассмотрим процесс построения покрытия на конкретном примере. Пусть задана одновыходная комбинационная схема (рис. 3.1), где x_1, x_2, x_3 и x_4 – независимые входы, z_1, z_2 и z_3 – промежуточные переменные (выходы элементов схемы), z_4 – выход схемы и 1, 2, 3 и 4 – элементы схемы.

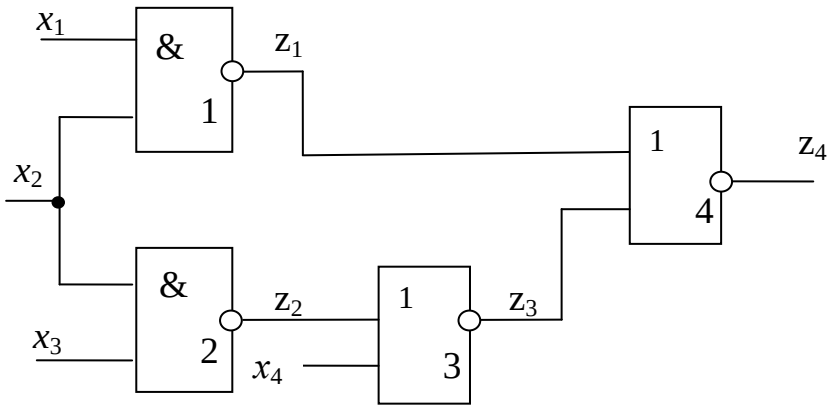


Рис. 3.1 Комбинационная схема

Для каждого элемента в соответствии с его типом (логической функцией) предварительно необходимо построить вырожденное покрытие (или использовать наработанные варианты) по их входам и выходу. В данной схеме использованы две двухвходовые схемы И-НЕ и две ИЛИ-НЕ. Их вырожденные покрытия выглядят так:

$$C(z_1)=\begin{array}{ccc|c} x_1 & x_2 & z_1 & \\ \hline 1 & 1 & 0 & c_1 \\ 0 & \times & 1 & c_2 \\ \times & 0 & 1 & c_3 \end{array} \qquad C(z_2)=\begin{array}{ccc|c} x_2 & x_3 & z_2 & \\ \hline 1 & 1 & 0 & c_1 \\ 0 & \times & 1 & c_2 \\ \times & 0 & 1 & c_3 \end{array}$$

$$C(z_3)=\begin{array}{ccc|c} z_1 & z_2 & z_3 & \\ \hline 1 & \times & 0 & c_1 \\ \times & 1 & 0 & c_2 \\ 0 & 0 & 1 & c_3 \end{array} \qquad C(z_4)=\begin{array}{ccc|c} z_1 & z_3 & z_4 & \\ \hline 1 & \times & 0 & c_1 \\ \times & 1 & 0 & c_2 \\ 0 & 0 & 1 & c_3 \end{array}$$

Построение покрытия начинается от выхода схемы z_4, задавая значение $z_4 = 1$ для построения единичного покрытия $C^1(z_4)$ и $z_4 = 0$ для построения нулевого покрытия схемы $C^0(z_4)$. Значение $z_4 = 1$ обеспечивается объединением кубов из покрытий: $C(z_1) \cup C(z_3)$, в которых необходимо обеспечить $z_1 = 0$ и $z_3 = 0$. Безразличное значение × не раскрывается. Значение $z_1 = 0$ обеспечивается кубом $c_1(z_1)$ и значение $z_3 = 0$ – кубами $c_1(z_3)$ и $c_2(z_3)$. Произведя соответствующие подстановки и пересечения с

исходными кубами, где все нераскрытые координаты полагаются равными ×, получим покрытие схемы в едином формате.

Этапы построения эквивалентного покрытия сведены в таблицу 3.1.

Таблица 3.1

x_1	x_{21}	x_{22}	x_3	x_4	z_1	z_2	z_3	z_4	Примечание
								1	$C^1(z_4)$
					0		0		$\pi^1 z_4$: $\overline{z_1 \vee z_3}$
				1	0	×			$\pi^0 z_3$: $\overline{z_2 \vee x_4}$
				×	0	1			
				1	0	×			$\pi^1 z_2$: $\overline{x_{22} \& x_3}$
			0	×	0				
		0	×	×	0				
1	1	×	×	1					$\pi^0 z_1$: $\overline{x_{21} \& x_1}$
1	1	×	0	×					
1	1	0	×	×					

Итак, применяя операции подстановки, пересечения и объединения, мы построили эквивалентное покрытие. Две координаты k_i и k_j считаются эквивалентными, если пути, им соответствующие, начинаются от одного и того же входа схемы. Число координат в эквивалентном покрытии схемы равняется числу её путей. Заметим, что в последнем кубе эквивалентного покрытия по эквивалентным координатам x_{21} и x_{22} получены противоположные значения (0 и 1) и при переходе к обычным координатам данный куб удаляется. Таким образом, построенное покрытие будет:

$$C^1(z_4) = \begin{array}{cccc|c} x_1 & x_2 & x_3 & x_4 & z_4 \\ \hline 1 & 1 & \times & 1 & 1 \\ 1 & 1 & 0 & \times & 1 \end{array} \begin{array}{l} \} c_1 \\ \} c_2 \end{array}$$

Аналогично, задавая значение $z_4 = 0$, построим нулевое покрытие схемы $C^0(z_4)$. Значение $z_3 = 0$ обеспечивается подстановкой кубов $C^1(z_1)$ или $C^1(z_3)$, иначе говоря, значениями $z_1 = 1$ или $z_3 = 1$. Значение $z_1 = 1$ обеспечивается объединением кубов $\{c_2(z_1) \cup c_3(z_1)\}$, а $z_3 = 1$ объединением кубов $\{c_1(z_2) \cup \overline{x}_4\}$.

Сведём этапы построения нулевого эквивалентного покрытия схемы в таблицу 3.2.

Объединив эквивалентные координаты, получим нулевое покрытие схемы в едином формате:

$$C^0(z_4) = \begin{array}{cccc|c} x_1 & x_2 & x_3 & x_4 & z_4 \\ \hline 0 & \times & \times & \times & 0 \\ \times & 0 & \times & \times & 0 \\ \times & 1 & 1 & 0 & 0 \end{array} \begin{array}{l} \} c_3 \\ \} c_4 \\ \} c_5 \end{array}$$

Таблица 3.2

x_1	x_{21}	x_{22}	x_3	x_4	z_1	z_2	z_3	z_4	Примечание
								0	$C^0(z_4)$
					1		×		$\pi^0 z_4$: $\overline{z_1 \vee z_3}$
					×		1		
					1				$\pi^1 z_3$: $\overline{z_2 \vee x_4}$
				0		0			
					1				$\pi^0 z_2$: $\overline{x_{22} \& x_3}$
		1	1	0					
0	×	×	×	×					$\pi^1 z_1$: $\overline{x_1 \& x_{21}}$
×	0	×	×	×					
×	×	1	1	0					

Для контроля вычислений отобразим построенные кубы на карте Карно, которая задаёт все вершины куба E_2^4, удалив из покрытий промежуточные переменные z_1, z_2 и z_3. При этом получим для выхода схемы z_4 следующую карту Карно:

x_1x_2 \ x_3x_4	00	01	11	10
00				
01				
11	1	1	1	
10				

При проверке по карте Карно видно, что $C^1(z_4) \cap C^0(z_4) = \varnothing$ и $C^1(z_4) \cup C^0(z_4) = E_2^4$. Указанную проверку можно выполнить и непосредственно с использованием операций пересечения ($\cap$) и объединения ($\cup$) кубов.

Таким образом, построены эквивалентные и вырожденные покрытия рассмотренной схемы.

3.1.2. Машинная модель схемы

Построение машинной модели последовательностной схемы сводится к заданию структуры схемы и закона ее функционирования. Структура задается указанием входов, выходов, сигналов обратной связи, элементов и связей между ними.

Идентификация входов и выходов схемы не представляет труда, так как они обычно задаются в явном виде. Выделение сигналов обратной связи сводится к поиску и разрыву циклов или замкнутых контуров в схеме. Решение этой задачи в общем виде является весьма трудоемкой и сложной процедурой.

Наибольшее распространение на практике получили схемы, в которых сигналы обратной связи реализуются с помощью ***RS***-триггеров. Поэтому можно сопоставить каждому ***RS***-триггеру один сигнал обратной связи,

снимаемый с его единичного плеча, на выходе которого до его разветвления и будет вноситься фиктивная точка разрыва. Сигнал, снимаемый с нулевого плеча, будем считать в модели инверсией сигнала единичного плеча. Работа триггера в такой машинной модели описывается парафазным кодом.

Модель схемы в самом общем виде можно представить так, как это показано на рис. 3.2. Здесь $X = \{x_1, \ldots, x_n\}$ — множество независимых входов схемы, $Z = \{z_1, \ldots, z_m\}$ — множество выходов и $Y = \{y_1, \ldots, y_k\}$ — множество сигналов обратной связи или псевдовходов схемы.

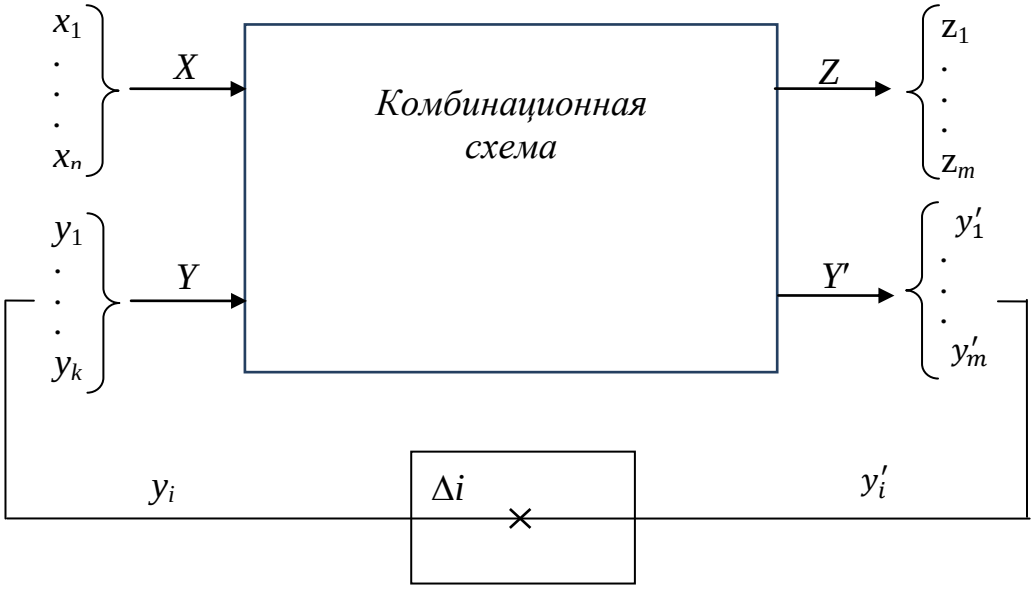


Рис. 3.2. Асинхронная модель последовательностной схемы

Любой i-сигнал обратной связи до точки фиктивного разрыва, изображаемой на схеме знаком ×, обозначается через y_i' а после нее — через y_i. Множество Y характеризует текущее состояние схемы, а $Y' = \{y_1', \ldots, y_k'\}$ — ее последующее состояние. Текущее и последующее состояния схемы могут различаться в переходном периоде работы схемы, а после его окончания они совпадают.

В машинной модели в точке фиктивного разрыва вводится асинхронная задержка Δi, которая соответствует задержке сигнала в комбинационной части схемы. Эта задержка вызывается задержками сигналов на элементах схемы и в линиях связи между элементами. Так как сигналы проходят различные пути в схеме, то задержка в общем случае является переменной как для различных сигналов, так и для одного и того же сигнала в разных тактах работы схемы. Однако после окончания переходного процесса, время которого равняется Δi, $y_i(t+\Delta i) = y_i'(t)$. Это соответствует основной модели асинхронной схемы, в которой, как бы ни была велика максимальная задержка Δi, переходный процесс всегда заканчивается.

Закон функционирования схемы можно задать различными способами, например, в виде таблиц выходов и переходов схемы, систем булевых уравнений для выходов и сигналов обратной связи или двоичных, кубических покрытий.

В системе машинного проектирования законы функционирования схемы удобно задавать в виде двоичных покрытий. Будем считать также, что в модели любой сигнал может принимать три значения 0, 1 или ×.

Покрытия строятся по π-алгоритму автономно для каждой из подсхем, на которые предварительно разбивается исходная схема.

Разбиение на подсхемы осуществляется следующим образом. С каждым внешним выходом или сигналом обратной связи отождествляется выход подсхемы, соответственно комбинационной или триггерной. Все подсхемы должны быть одновыходными. В триггерной подсхеме может быть только один цикл, задающий собственную или локальную обратную связь.

Число координат в покрытиях ограничивается некоторым числом, что связано с возможностью представления кубов одним машинным словом и ограничением на объем покрытий.

Объем покрытий находится в зависимости от числа координат кубов.

Ограничение на число координат вызывает появление комбинационных подсхем, не имеющих непосредственных связей с внешними выходами схемы.

Отображение структуры схемы в покрытиях будем осуществлять эквивалентными координатами и называть такие покрытия эквивалентными покрытиями. Каждому пути сигнала в подсхеме будет соответствовать в покрытии своя координата.

Две координаты *i*-я и *j*-я считаются эквивалентными, если пути, им соответствующие, начинаются от одного и того же входа подсхемы. Таким образом, в эквивалентных покрытиях существует два вида эквивалентности: эквивалентность между координатами и путями и эквивалентность некоторых координат между собой.

Число координат в эквивалентном покрытии подсхемы равняется числу ее путей. Ограничением числа координат в покрытиях добиваемся ограничения числа путей в подсхемах.

Построение эквивалентных покрытий, которые будем обозначать как $C^э$, осуществляется обычным образом, но без совмещения эквивалентных координат. Вследствие этого при построении $C^э$ не будет возникать как пустых пересечений кубов, так и поглощений последних.

На рис. 3.3 приведен пример комбинационной подсхемы. В подсхеме существуют четыре пути: l_1=146, l_2=246, l_3=256 и l_4= 356. Соответственно, в эквивалентных покрытиях $C^э$(z) и $C^э(\bar{z})$ будет по четыре координаты, из которых вторая и третья эквивалентны между собой, так как пути l_2 и l_3, им соответствующие, начинаются от одного и того же входа 2.

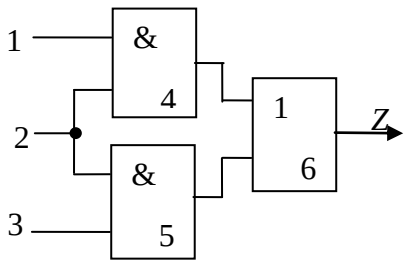


Рис. 3.3. Комбинационная подсхема

Построим эквивалентные покрытия, указав в формате Ф=(1223) для них только начала путей, соответствующих координатам кубов. Итак,

$$C^{э}(z) = \begin{array}{|cccc|}\hline 1 & 2_4 & 2_5 & 3 \\ \hline 1 & 1 & \times & \times \\ \times & \times & 1 & 1 \\ \hline \end{array} \quad \text{и} \quad C^{э}(\bar{z}) = \begin{array}{|cccc|}\hline 1 & 2_4 & 2_5 & 3 \\ \hline 0 & \times & 0 & \times \\ \times & 0 & 0 & \times \\ 0 & \times & \times & 0 \\ \times & 0 & \times & 0 \\ \hline \end{array}$$

Объединив эквивалентные координаты 2_4 и 2_5 путем пересечения значений в каждом из кубов покрытий $C'(z)$ и $C'(\bar{z})$ координат 2_4 и 2_5, получим:

$$C'(z) = \begin{array}{|ccc|}\hline 1 & 2 & 3 \\ \hline 1 & 1 & \times \\ \times & 1 & 1 \\ \hline \end{array} \quad \text{и} \quad C'(\bar{z}) = \begin{array}{|ccc|}\hline 1 & 2 & 3 \\ \hline 0 & 0 & \times \\ \times & 0 & \times \\ 0 & \times & 0 \\ \times & 0 & 0 \\ \hline \end{array}$$

Удалив из $C'(z)$ поглощаемые кубы, получим обычные покрытия для схемы:

$$C'(z) = \begin{array}{|ccc|}\hline 1 & 2 & 3 \\ \hline 1 & 1 & \times \\ \times & 1 & 1 \\ \hline \end{array} \quad \text{и} \quad C'(\bar{z}) = \begin{array}{|ccc|}\hline 1 & 2 & 3 \\ \hline \times & 0 & \times \\ 0 & \times & 0 \\ \hline \end{array}$$

Пример триггерной подсхемы приведен на рис. 3.4. Для построения машинной модели необходимо разорвать цикл, образуемый элементами 6 и 7. С этой целью введем фиктивную точку разрыва на выходе 7-го элемента, который непосредственно связан с внешним выходом схемы до точки разветвления сигнала.

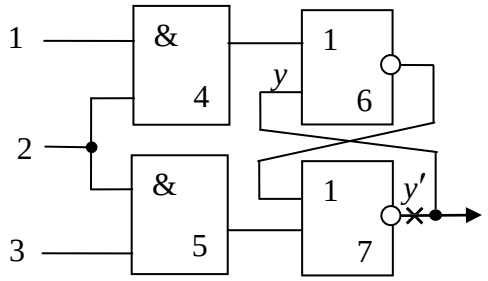


Рис. 3.4. Триггерная подсхема

Обозначим значение сигнала непосредственно на выходе 7-го элемента до точки фиктивного разрыва через y', а значение того же сигнала на входе 6-го элемента после точки фиктивного разрыва через y.

Таким образом, сигнал y на входе 6-го элемента становится псевдовходом схемы. Построим пути сигналов $l_1 = 1467$, $l_2 = 2467$, $l_3 = 257$, $l_4 = 357$ и
$l_5 = y67$ и эквивалентные покрытия при Ф= $(12_4 2_5 3y)$:

$$C^{э}(y') = \begin{array}{|cccc|c|}\hline 1 & 2_4 & 2_5 & 3 & y \\ \hline 1 & 1 & 0 & \times & \times \\ 1 & 1 & \times & 0 & \times \\ \times & \times & 0 & \times & 1 \\ \times & \times & \times & 0 & 1 \\ \hline \end{array} \quad \text{и} \quad C^{э}(\bar{y}') = \begin{array}{|cccc|c|}\hline 1 & 2_4 & 2_5 & 3 & y \\ \hline \times & \times & 1 & 1 & \times \\ 0 & \times & \times & \times & 0 \\ \times & 0 & \times & \times & 0 \\ \hline \end{array} \quad (1)$$

Объединив эквивалентные координаты, будем иметь:

$$C^э(y') = \left\{\begin{array}{ccc|c} 1 & y & \times & \times \\ 1 & 1 & 0 & \times \\ \times & 0 & \times & 1 \\ \times & \times & 0 & 1 \end{array}\right. \text{ и } C^э(\bar{y}') = \left\{\begin{array}{ccc|c} \times & 1 & 1 & \times \\ 0 & \times & \times & 0 \\ \times & 0 & \times & 0 \end{array}\right.$$

Удалив из $C'(y')$ куб $(1y\times\times) = \varnothing$, получим:

$$C'(y') = \left\{\begin{array}{ccc|c} 1 & 1 & 0 & \times \\ \times & 0 & \times & 1 \\ \times & \times & 0 & 1 \end{array}\right. \begin{array}{l} c_1 \\ c_2 \\ c_3 \end{array} \quad \text{и} \quad C'(\bar{y}') = \left\{\begin{array}{ccc|c} \times & 1 & 1 & \times \\ 0 & \times & \times & 0 \\ \times & 0 & \times & 0 \end{array}\right. \begin{array}{l} c_4 \\ c_5 \\ c_6 \end{array}$$

3.1.3. Граф переходов схемы

Граф переходов схемы служит для наглядного изображения закона функционирования последовательностных схем.

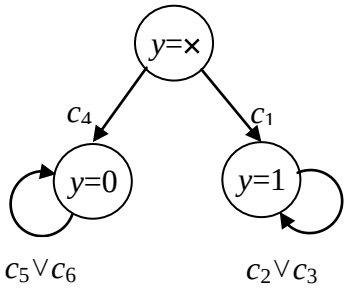


Рис. 3.5. Частично определенный граф переходов для схемы на рис. 3.4

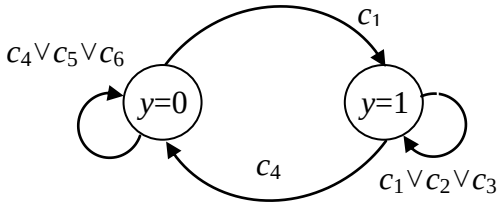


Рис. 3.6. Полностью определенный граф переходов для схемы на рис. 3.4

В узлах графа, обозначенных на рис. 3.5 и 3.6 кругами, помещаются состояния схемы с указанием непосредственно значений сигналов обратной связи $y \in Y$. Дуги графа со стрелками указывают переходы и их направления. Они помечаются входными наборами или кубами, вызывающими данный переход.

В покрытиях, являющихся математической моделью схемы, сигналы могут принимать три значения — 0, 1 и ×, где × есть неопределенное значение, которое можно полагать равным как 0, так и 1. Поэтому граф переходов схемы может быть частично или полностью определенным.

Если при отметке состояний схемы на графе некоторые сигналы обратной связи $y = \times$, то граф переходов является частично определенным и полностью определенным в противном случае. Отметка дуг графа всегда будет проводиться кубами из покрытий, в которых некоторые значения входов схемы или подсхемы могут быть неопределенными, т. е. равными ×.

Для триггерной подсхемы (рис. 3.4) интерпретация покрытий $C(y')$ и $C(\bar{y}')$ в виде частично определенного графа переходов показана на рис. 3.5. Дуги графа помечены идентификаторами кубов из покрытий.

Куб c_1 переводит схему из неопределенного состояния $y = \times$ в определенное состояние $y = 1$, так как куб $c_1 \in C(y')$ и, следовательно,

вычисляет значение $y'=1$, а после окончания переходного процесса в схеме $y=y'=1$.

Куб $c_1 \in C(\bar{y}')$ переводит схему из определенного состояния $y=0$ в то же самое состояние $y'=y=0$, т. е. не изменяет внутреннего состояния схемы.

Доопределив значение $y=\times$ в кубах $c_1 \in C(y')$ и $c_4 \in C(\bar{y}')$ как в значение 0, так и в значение 1, получим интерпретацию покрытий в виде полностью определенного графа переходов, показанного на рис. 3.6.

Различие между частично и полностью определенными графами переходов наблюдается только на модели, так как в реально работающей физической схеме значение любого сигнала обратной связи y является, как правило, определенным и равным 0 или 1. Исключение составляет случай, когда схема находится в режиме генерации для какого-то сигнала y, периодически изменяющего свое значение с 0 на 1 и обратно. Этот сигнал можно в данном случае условно считать равным ×.

Тестовый эксперимент состоит в подаче на схему последовательности входных воздействий и наблюдении реакции схемы. Исходное состояние схемы при этом не определено, поскольку в ходе физического эксперимента после включения питания схемы любой триггер может установиться как в состояние 0, так и в состояние 1. Это следует интерпретировать значением × соответствующего сигнала обратной связи $y \in Y$.

Частично определенный граф переходов схемы как раз и отображает эту неопределенность значений сигналов обратной связи.

По графу переходов схемы можно строить упорядоченные последовательности входных наборов. Например, для графа, приведенного на рис. 3.5, в качестве первых наборов любых последовательностей могут быть только кубы c_1 или c_4, так как они опираются на неопределенное предыдущее значение $y=\times$.

Куб c_2 не может быть первым кубом последовательности, так как он опирается на значение $y=1$, а схема могла находиться первоначально и в состоянии $y=0$. Примеры корректных последовательностей: $c_4c_1c_2c_3$, $c_4c_5c_1c_3$, $c_1c_2c_4c_6$. Примеры некорректных последовательностей: $c_2c_3c_4c_5$, $c_4c_2c_3$, $c_1c_2c_5$.

3.1.4. Активизация путей

Пусть путь l в схеме N начинается с входа схемы x_i, а заканчивается выходом z_j. Путь l считается активным на наборе, если изменение логического значения x_i вызывает изменение логических значений элементов схемы, входящих в путь l, а также изменение логического значения z_j.

Идея Элдриджа состоит в том, что неисправность, вызывающая появление ложного значения в любой точке активизированного пути, приводит к появлению на выходе этого пути логического значения, отличного от реакции исправной схемы.

Если на активном пути l в наборе t некоторый вход или выход элемента активизирован в значении $p = 0 \vee 1$, что будет обозначаться как p', то

проверяется неисправность m^p. Другими словами, если имеем активный ноль 0', то проверяется неисправность m^1, если активную единицу 1', то — m^0.

Активизация путей в подсхемах может быть формализована различными способами: с помощью эквивалентных нормальных форм, *d*-алгоритма или булевых разностей.

В системе проектирования тестов активизацию путей будем проводить по эквивалентным покрытиям, включив в процедуру все наиболее существенное из указанных выше методов.

Структура схем отображается, как и в эквивалентных нормальных формах, с помощью эквивалентных координат. Нахождению булевой разности в значении 1 будет соответствовать операция пересечения кубов, взятых из единичного и нулевого покрытий. Распространение активности от подсхемы к подсхеме, если это требуется, будем проводить аналогично *d*-алгоритму.

Заметим следующие отличия от *d*-алгоритма. Активизация будет проводиться в конкретных значениях 0' или 1', и активизироваться будет только один путь как внутри подсхем, так и при переходе от подсхемы к подсхеме. Последнее, скорее, является ограничением метода по сравнению с *d*-алгоритмом, который допускает активизацию сразу нескольких путей от точек разветвления сигналов, которые объединяются названием сложный путь.

Это ограничение связано с требованием устойчивости проверок неисправностей. При активизации сложного пути от места неисправности до внешнего выхода схемы принципиально существуют состязания сигналов, проходящих различные пути.

Если же для некоторого входа подсхемы существует только сложный путь, то такую подсхему будем подвергать дальнейшему разбиению, добиваясь того, чтобы в каждой подсхеме существовал бы для любого входа активный путь. Противоречие со сложным путем должно преодолеваться при распространении активностей от подсхемы к подсхеме.

Итак, пусть для схемы, выход которой обозначен через z, заданы эквивалентные покрытия $C^э(z)$ и $C^э(\bar{z})$. Требуется найти пару наборов, активизирующих некоторый заданный путь *l*, т. е. обеспечить переключение логического значения на выходе схемы в зависимости от переключения сигналов вдоль активного пути l при переходе от набора к набору.

Так как на любом отрезке, являющемся входом или выходом какого-либо элемента, может быть только одна неисправность m^p или $m^{\bar{p}}$, то любая одиночная константная неисправность 0 или 1 будет либо вызывать предварительное переключение на выходе схемы на первом наборе, либо препятствовать переключению при переходе ко второму набору. И в том и в другом случае на выходе схемы будет фиксироваться ошибка, порождаемая неисправностью m^p или $m^{\bar{p}}$. Таким образом, чтобы синтезировать тест, обеспечивающий контроль всех одиночных константных неисправностей, достаточно включить в него пары наборов, активизирующих все пути схемы.

В соответствии с вышеизложенным можно следующим образом сформулировать конструктивное определение теста.

Некоторое частично упорядоченное множество *T* является тестом схемы *N,* содержащей $L = \{l\}$ путей, если для любого $l \in L$ найдется пара наборов $(t/\tilde{t})$ из *T*, активизирующая данный путь *l*.

Путь l является активным как на наборе *t*, так и на наборе $\tilde{t}$, поэтому такой путь *l* можно считать переключающим путем на паре наборов $(t/\tilde{t})$.

Синтез пары переключающих наборов $t/\tilde{t}$ будем проводить с использованием операции пересечения кубов $a \in C^э(z)$ и $b \in C^э(\bar{z})$, обозначаемой как $c_y = a \cap b$, и операции обратного пересечения куба c_y с исходными кубами a и b, в результате которого $t = c_y \cap a$ и $\tilde{t} = c_y \cap b$. Действительно, так как кубы a и b принадлежат противоположным покрытиям, то при переходе от куба a к кубу b сигнал на выходе схемы должен изменяться. Это изменение будет происходить из-за переключения тех сигналов, по которым различаются кубы a и b, что и обозначается значением *y* в кубе $c_y = a \cap b$.

Далее, так как необходимо обеспечить один активный путь, то значение y должно быть только по одной эквивалентной входной координате, соответствующей активизируемому пути. Формирование пары наборов $t/\tilde{t}$ из куба c_y осуществляется на основе операции $\overset{\cap}{y}$. Эта операция отличается от обычного пересечения только тем, что $y \overset{\cap}{y} p = p'$, и состоит в синтезе активных значений p' или $\overline{p'}$, которые, естественно, и должны различаться в наборах t и $\tilde{t}$. Поэтому для того, чтобы $\overset{\cap}{y}$ приводило к непустым обратным пересечениям c_y с исходными кубами a и b, необходимо обеспечить безразличные значения × по всем координатам, эквивалентным координате, равной *y* в кубе c_y.

Рассмотрим правило активизации путей по эквивалентным покрытиям на примере комбинационной подсхемы, представленной на рис. 3.3. Предварительно запишем покрытия в вырожденном виде, т. е. зададим в покрытиях значение в явном виде. Получим вырожденные покрытия:

$C^э(1) =$

1	2_4	2_5	3	z	
1	1	×	×	1	c_1
×	×	1	1	1	c_2

и

$C^э(0) =$

1	2_4	2_5	3	z	
0	×	0	×	0	c_3
×	0	0	×	0	c_4
0	×	×	0	0	c_5
×	0	×	0	0	c_6

$C^э(1)$ обозначает единичное покрытие, а $C^э(0)$ – нулевое покрытие.

Проведем активизацию последовательно для всех путей подсхемы.

1. Путь $l_1 = 146$ соответствует первой координате и, следовательно, необходимо путем перебора пар кубов из покрытий $C^э(1)$ и $C^э(0)$ найти такие пары, которые дают в результате пересечения значение y по первой координате. Таких пар в покрытиях две: (c_1, c_3) и (c_1, c_5). Положив $a = c_1$ и $b = c_3$, получим

$a =$ 11××1
$b =$ 0×0×0 $\cap$
$c_y =$ y10×y

$c_y = \varnothing$, так как по эквивалентным координатам $2_4 = 1$ и $2_5 = 0$ содержатся противоположные значения, которые при совмещении образуют противоречивый куб c_y и, следовательно, дадут противоречивые или пустые наборы t и $\tilde{t}$.

Рассмотрим следующую пару, положив $a = c_1$ и $b = c_5$, получим

$a =$ 11××1
$b =$ 0××00 $\cap$;
$c_y =$ y1×0y

$c_y =$ y 1×0 y $\cap$
$a =$ 1 1××1 y
$t_1 =$ 1′1×01′

и

$c_y =$ y 1×0 y $\cap$
$b =$ 0 ××00 y
$\tilde{t}_1 =$ 0′1×00′

В данном случае в кубе c_y эквивалентные координаты $2_4 = 1$ и $2_5 = \times$ совмещаются. Значения y по первой координате и на выходе подсхемы (координата z) указывают, что изменение значения по первой координате, т. е. изменение значений вдоль пути в подсхеме вызывает изменение значения на выходе подсхемы. Итак, пара $t_1/\tilde{t_1} = (1'1\times01'/0'1\times00')$ или, после совмещения эквивалентных координат, $(1'101'/0'100')$, активизирует путь $l_1 = 146$.

На этой паре наборов проверяются все константные неисправности пути l_1. Множество проверяемых неисправностей $M(l_1) = \{1^0, 1^1, 4^0, 4^1, 6^0, 6^1\}$ для пары $t_1/\tilde{t_1}$, а для каждого из наборов $M(t_1) = \{1^0, 4^0, 6^0\}$ и $M(\tilde{t_1}) = \{1^1, 4^1, 6^1\}$. Любая из указанных неисправностей либо вызывает предварительное переключение на наборе t_1, либо задерживает переключение на наборе $\tilde{t_1}$.

2. Путь $l_2 = 246$ и ему соответствует вторая координата 2_4. Пары кубов для l_2 будут: (c_1, c_4) и (c_1, c_6). Положив $a = c_1$ и $b = c_4$, получим

$a =$ 11××1
$b =$ ×00×0 $\cap$;
$c_y =$ 1y0×y

в кубе c_y координата $2_4 = y$, а $2_5 = 0$, т. е. не ×, и, следовательно, $c_y = \varnothing$, так как нарушено требование о том, что если одна из эквивалентных координат равна $\bar{y}$, то все остальные, ей эквивалентные координаты, должны быть безразличными, т. е. равняться ×.

Действительно, если из данного $c_y = 1y0\times y$ сформировать $t = c_y \overset{\cap}{}_y a = 11'0\times1'$ и $\tilde{t} = c_y \overset{\cap}{}_y b = 10'0\times0'$, то $t = \varnothing$, а в кубе $\tilde{t}$ одна из эквивалентных координат $(2_4 = 0)$ является активной, т. е. может изменяться при наличии неисправности, а другая $(2_5 = 0)$ должна сохранять постоянное значение. Это противоречие может приводить к компенсации неисправности, т. е. к отсутствию ее проверки на наборе $\tilde{t}$, если в подсхеме есть еще один путь от данной неисправности, содержащий нечетное число инверсий до выхода схемы.

Для другой пары кубов, положив $a = c_1$ и $b = c_6$, получим:

$a =$ 11××1 ∩; $c_y =$ 1y ×0 y и $c_y =$ 1y ×0y ∩
$b =$ ×0×00 $a =$ 11 ××1 $b =$ ×0 ×00 y
$c_y =$ 1y×0y $t_2 =$ 11′×01′ $\tilde{t}_2 =$ 10′×00′

Активизация пути $l_2 = 246$ для пары наборов $t_2/\tilde{t}_2$ показана на рис. 3.7.

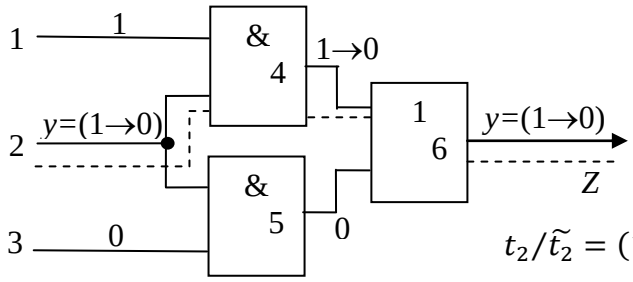


Рис. 3.7. Активизация пути $l_2 = 246$ на паре наборов $t_2/\tilde{t}_2 = (11'\times01'/10'\times00')$ и $c_y = 1y\times0y3$

Для $l_3 = 256$ существуют две пары (c_2, c_3) и (c_2, c_4), из которых вторая пара приводит к противоречию.

Выбрав $a = c_2$ и $b = c_3$, получим

$a =$ ××111 ∩; $c_y =$ 0×y 1 y ∩ и $c_y =$ 0×y 1y ∩
$b =$ 0×0×0 $a =$ ××1 11 y $b =$ 0×0 ×0 y
$c_y =$ 0×y1 y $t_3 =$ 0×1′11′ $\tilde{t}_3 =$ 0×0′10′

4. Для $l_4 = 356$ существуют пары (c_2, c_5) и (c_2, c_6), из которых подходит только первая. Положив $a = c_2$ и $b = c_5$, получим:

$a =$ ××111 ∩; $c_y =$ 0×1 y y ∩ и $c_y =$ 0×1 y y ∩
$b =$ 0××00 $a =$ ××11 1 y $b =$ 0××0 0 y
$c_y =$ 0×1yy $t_4 =$ 0×11′1′ $\tilde{t}_4 =$ 0×10′0′

Итак, для всех путей схемы на рис. 3.2 вычислены пары наборов, объединение которых дает тест, обладающий полнотой контроля любой одиночной константной неисправности. Запишем тест в явном виде:

$$T' = \left\{\begin{array}{cccc|c} 1 & 2_4 & 2_5 & 3 & z \\ \hline 1' & 1 & \times & 0 & 1' \\ 0' & 1 & \times & 0 & 0' \\ 1 & 1' & \times & 0 & 1' \\ 1 & 0' & \times & 0 & 0' \end{array}\right\}\begin{array}{c} t_1 \\ \tilde{t}_1 \\ t_2 \\ \tilde{t}_2 \end{array} \cup \left\{\begin{array}{cccc|c} 1 & 2_4 & 2_5 & 3 & z \\ \hline 0 & \times & 1' & 1 & 1' \\ 0' & \times & 0' & 1 & 0' \\ 0 & \times & 1 & 1' & 1' \\ 0 & \times & 1 & 0' & 0' \end{array}\right\}\begin{array}{c} t_3 \\ \tilde{t}_3 \\ t_4 \\ \tilde{t}_4 \end{array}.$$

Объединив эквивалентные координаты 2_4 и 2_5, получим тест:

$$T' = \left\{\begin{array}{ccc|c} 1 & 2 & 3 & z \\ \hline 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{array}\right\}\begin{array}{c} t_1 \\ \tilde{t}_1 \\ t_2 \\ \tilde{t}_2 \end{array} \cup \left\{\begin{array}{ccc|c} 1 & 2 & 3 & z \\ \hline 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{array}\right\}\begin{array}{c} t_3 \\ \tilde{t}_3 \\ t_4 \\ \tilde{t}_4 \end{array},$$

в котором $t_1 = t_2$ и $t_3 = t_4$. С целью минимизации длины теста эти наборы можно объединить, так как для комбинационной схемы порядок подачи наборов при тестировании безразличен.

В заключение кратко сформулируем правило активизации путей:

1) для активизации некоторого пути l_i необходимо найти пересечение двух кубов $a \in C^3(1)$ и $b \in C^3(0)$ так, чтобы только по i-й координате в кубе $c_y = a \cap b$ было бы значение y;

2) по всем остальным координатам, эквивалентным i-й, в кубе c_y должны быть безразличные значения x;

3) двойственная пара наборов $t/\tilde{t}$ образуется как $t = c_y \overset{\cap}{}_y a$, $\tilde{t} = c_y \overset{\cap}{}_y b$ и $p \cap y = p'$. Наборы различаются между собой только по i-координате, которая является активной и вызывает переключение сигнала на выходе схемы при переходе от набора t к набору $\tilde{t}$ по пути l_i.

3.1.5. Синтез проверяющих последовательностей

В соответствии с математической моделью схемы, представленной на рис. 3.1, любой набор при синтезе тестов будем представлять в формате $t = X_t Y_t Y_{t'}$, где Y_t есть исходное состояние схемы и $Y_{t'}$ – вырабатываемое состояние. Так как после окончания переходного процесса состояние схемы есть $Y_{t'}$, то при реальном тестировании схем набор t вырождается в формат $X_t Y_{t'}$.

Любой сигнал в наборе t при синтезе теста может принимать три значения – 0, 1 или ×. Этому неопределенному значению × в реальной схеме на наборе t может соответствовать как определенное значение 0 или 1, так и неопределенное значение $1/2$, например, при генерации сигнала.

В обоих случаях как определенное значение 0 или 1, нам неизвестное, так и действительно неопределенное значение $1/2$, т. е. ни 0, ни 1, не влияют на контролирующие свойства набора t.

Активизация путей в последовательностных одновыходных подсхемах осуществляются точно так же, как и в комбинационных – по эквивалентным покрытиям с учетом формата XYY'. Пара наборов $t/\tilde{t}$, активизирующая некоторый путь l, должна быть упорядочена по преемственности состояний схемы. Если наборы t и $\tilde{t}$ различаются по активному независимому входу подсхемы, то их исходные состояния должны быть одинаковы, т. е. $Y_t = Y_{\tilde{t}}$. Далее, так как наборы на схему подаются в последовательности от t к $\tilde{t}$, то $Y_{t'} = Y_{\tilde{t}}$ и, следовательно, $Y_t = Y_{t'}$. Поэтому на наборе t состояние схемы не должно изменяться, оно изменяется на наборе $\tilde{t}$, так как при переходе от набора t к набору $\tilde{t}$ выход схемы, в данном случае y', должен переключаться. В соответствии с вышеизложенным, набор t, на котором состояние схемы не изменяется, будем называть нейтральным набором, а набор $\tilde{t}$, изменяющий состояние схемы, – переключающим набором.

Рассмотрим активизацию путей на примере схемы, показанной на рис. 3.4. Предварительно перепишем покрытия (1) в вырожденном виде, где $Y = \{y\}$ и $Y' = \{y'\}$:

$$C^э(1) = \begin{array}{|cccccc|c|} \hline 1 & 1 & \times & 0 & \times & 1 & c_1 \\ \times & \times & 0 & \times & 1 & 0 & c_2 \\ \times & \times & \times & 0 & 1 & 1 & c_3 \\ \hline \end{array} \quad \text{и} \quad C^э(0) = \begin{array}{|cccccc|c|} \hline \times & \times & 1 & 1 & \times & 0 & c_4 \\ 0 & \times & \times & \times & 0 & 0 & c_5 \\ \times & 0 & \times & \times & 0 & 0 & c_6 \\ \hline \end{array},$$

(столбцы: 1, 2_4, 2_5, 3, y | y')

При переходе от $C^э(y')$ к покрытию $C^э(1)$ первый куб удален, так как он дает пустое пересечение при совмещении эквивалентных координат 2_4 и 2_5.

Проведем последовательную активизацию всех путей схемы.

1. Путь $l_1 = 1467$ и для него существует пара (c_1, c_5).

Положив $a = c_1$ и $b = c_5$, получим:

$$\begin{array}{ll} a = & 11\times0\times1 \\ b = & 0\times\times\times00 \\ \hline c_y = & y1\times00y \end{array} \cap; \quad \begin{array}{ll} c_y = & y\,1\times00\,y \\ a = & 1\,1\times0\times1 \\ \hline \tilde{t}_1 = & 1'1\times001' \end{array} \cap_y \quad \text{и} \quad \begin{array}{ll} c_y = & y\,1\times\,00y \\ b = & 0\,\times\times\times00 \\ \hline t_1 = & 0'1\times\,000' \end{array} \cap_y ,$$

где $\tilde{t_1} = c_y \cap_y a$, так как $Y_{\tilde{t}} \neq Y'_{\tilde{t}}$ и $Y'_{\tilde{t}} \neq Y_t$, а $t_1 = c_y \cap_y b$ и дает $Y'_t = Y_{\tilde{t}}$ и $Y_t = Y_{t'}$. Итак, упорядоченная пара наборов будет

$t_1/\tilde{t_1} = (0'1\times000'/1'1\times001')$. Заметим, что новое значение $y' = 1$ в наборе $\tilde{t_1}$ является устойчивым, так как куб $c_1 \supset \tilde{t_1}$ из $C^э{}_1$ вырабатывает значение $y' = 1$ из безразличного значения $y = \times$.

2. Путь $l_2 = 2467$ и для него $a = c_1$ и $b = c_6$, поэтому

$$\begin{array}{ll} a = & 11\times0\times1 \\ b = & \times0\times\times00 \\ \hline c_y = & 1y\times00y \end{array} \cap; \quad \begin{array}{ll} c_y = & 1y\times00y \\ a = & 11\times0\times1 \\ \hline \tilde{t}_2 = & 11'\times001' \end{array} \cap_y \quad \text{и} \quad \begin{array}{ll} c_y = & 1y\times00\,y \\ b = & \times0\times\times00 \\ \hline t_2 = & 10'\times000' \end{array} \cap_y .$$

Для активизации l_2 имеем $t_2/\tilde{t_2} = (10'\times000'/11'\times001')$.

3. Путь $l_3 = 257$ и для него $a = c_2$ и $b = c_4$, поэтому

$$\begin{array}{ll} a = & \times\times01\times1 \\ b = & \times\times11\times0 \\ \hline c_y = & \times\times y11y \end{array} \cap; \quad \begin{array}{ll} c_y = & \times\times y\;11y \\ a = & \times\times0\;1\times1 \\ \hline \tilde{t}_3 = & \times\times0'111' \end{array} \cap_y \quad \text{и} \quad \begin{array}{ll} c_y = & \times\times y\;11y \\ b = & \times\times1\;1\times0 \\ \hline t_3 = & \times\times1'110' \end{array} \cap_y .$$

Итак, для l_3 имеем $t_3/\tilde{t_3} = (\times\times0'111'/\times\times1'110')$.

4. Путь $l_4 = 357$ и для него существуют две пары кубов активизации (c_1, c_4) и (c_3, c_4).

Пусть $a = c_1$ и $b = c_4$, тогда:

$$\begin{array}{ll} a = & 11\times0\times1 \\ b = & \times\times11\times0 \\ \hline c_y = & 111y\times y \end{array} \cap; \quad \begin{array}{ll} c_y = & 111y\times y \\ a = & 11\times0\;\times1 \\ \hline t'_4 = & 1110'\times1' \end{array} \cap_y \quad \text{и} \quad \begin{array}{ll} c_y = & 11\;1y\times y \\ b = & \times\times11\;\times0 \\ \hline t''_4 = & 1\;111'\times0' \end{array} \cap_y .$$

Здесь в t'_4 и в t''_4 значение $y = \times$, поэтому любой из наборов можно принять и за нейтральный и за переключающий. Значение $y = x$ в наборах t'_4 и t''_4 задает так называемый псевдоконтурный режим работы схемы, т. е. работа схемы на наборах t'_4 и t''_4 не зависит от значения сигнала обратной связи $y = \times$ и тестирование происходит как в обычной комбинационной схеме. Можно составить как пару (t'_4/t''_4), так и пару (t''_4/t'_4). В обоих случаях переключение значения y', снимаемого с выхода 7-го элемента, будет происходить как при переходе от t'_4 к набору t''_4, так и наоборот.

Положив $a = c_3$ и $b = c_4$, получим:

$$\begin{array}{ll} a = & \times\times\times011 \\ b = & \times\times11\times0 \\ \hline c_y = & \times\times1y1y \end{array} \cap; \quad \begin{array}{ll} c_y = & \times\times1y\;1\;y \\ a = & \times\times\times0\;11 \\ \hline t_4 = & \times\times10'11' \end{array} \cap_y \quad \text{и} \quad \begin{array}{ll} c_y = & \times\times1y\;1y \\ b = & \times\times11\;\times0 \\ \hline \tilde{t}_4 = & \times\times11'10' \end{array} \cap_y ,$$

т.е. типичную ситуацию активизации пути в последовательностной схеме. Здесь t_4 есть нейтральный набор, а $\tilde{t}_4$ – переключающий. В тест для активации пути t_4 можно включать как пару (t_4'/t_4''), так и пару $(t_4/\tilde{t}_4)$. Выберем пару $(t_4/\tilde{t}_4)$ как типичную и содержащую больше неопределенностей по входам подсхемы, что может иметь решающее значение при объединении наборов, синтезированных для других подсхем.

5. Путь $t_5 = y67$ является особым путем в подсхеме, так как он начинается с собственного сигнала обратной связи. Для t_5 существует четыре непротиворечивых пары: (c_2, c_5), (c_2, c_6), (c_3, c_5) и (c_3, c_6). Выбрав первую по порядку, положим $a = c_2$ и $b = c_5$ и получим:

$$\begin{array}{lll} a = & \times\times 0\times 11 & \cap; \\ b = & 0\times\times\times 00 & \\ \hline c_y = & 0\times 0\times yy & \end{array} \qquad \begin{array}{lll} c_y = & 0\times 0\times y\,y & \cap \\ a = & \times\times 0\times 1\;1 & y \\ \hline t_5 = & 0\times 0\times 1'1' & \end{array} \quad \text{и} \quad \begin{array}{lll} c_y = & 0\times 0\times y\,y & \cap \\ b = & 0\times\times\times 0\;0 & y \\ \hline t_5' = & 0\times 0\times 0'0' & \end{array},$$

Здесь оба набора t_5 и t_5' являются нейтральными и не образуют упорядоченную пару, так как ни $Y'_{t_5} \neq Y_{t_5'}$, ни $Y'_{t_5'} \neq Y_{t_5}$.

Кроме того, переключение сигнала обратной связи при переходе от t_5 к t_5' или от t_5' к t_5 необходимо обеспечить специальными промежуточными наборами.

В отличие от объединения пар наборов в тест комбинационной схемы, порядок наборов в котором безразличен, наборы теста триггерной подсхемы образуют упорядоченную последовательность. При переходе от набора к набору необходимо обеспечить преемственность по внутренним состояниям схемы.

С этой целью построим граф переходов схемы на тестовых наборах, приведенный на рис. 3.8, дополнив его установками из безразличного состояния $y = \times$, взятыми из частично определенного графа переходов схемы, показанного на рис. 3.5.

По данному графу переходов можно составить эквивалентные между собой по контролирующим свойствам последовательности, что определяется различными способами обхода дуг графа переходов. В качестве первых элементов последовательностей могут выступать кубы c_1 или c_4, так как они опираются на безразличное значение $y = \times$.

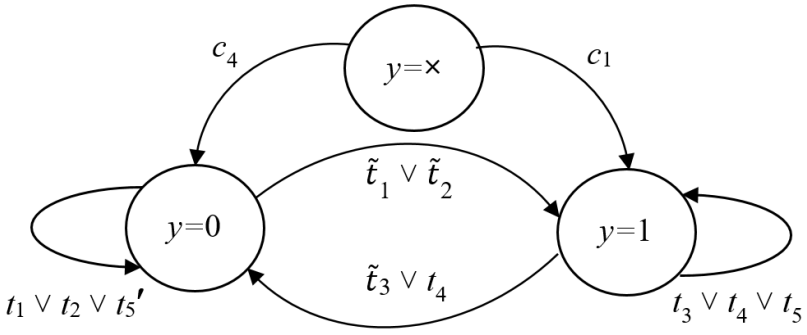


Рис. 3.8. Граф переходов

Для примера определим две такие тестовые последовательности: $T_1 = \{c_1, t_3, \tilde{t}_3, t_1, \tilde{t}_1, t_4, \tilde{t}_4, t_2, \tilde{t}_5, \tilde{t}_2, t_5\}$ и $T_2 = \{c_4, t_1, \tilde{t}_1, t_3, \tilde{t}_3, t_2, \tilde{t}_2, t_4, t_5, \tilde{t}_4, t'\}$. Тестом можно взять любую из них, например, последовательность T_1.

Запишем тестовые наборы в явном виде:

$T=$

1	2_4	2_5	3	y	y'	
1	1	×	0	×	1	c_1
×	×	0′	1	1	1′	t_3
×	×	1′	1	1	0′	$\tilde{t}_3$
0′	1	×	0	0	0′	t_1
1′	1	×	0	0	1′	$\tilde{t}_1$
×	×	1	0′	1	1′	t_4

$\cup$

1	2_4	2_5	3	y	y'	
×	×	1	1′	1	0′	$\tilde{t}_4$
1	0′	×	0	0	0′	t_2
0	×	0	×	0′	0′	t_5'
1	1′	×	0	0	1′	$\tilde{t}_2$
0	×	0	×	1′	1′	t_5

Объединяя эквивалентные координаты 2_4, 2_5 и удалив внутреннее значение y, которое необходимо только в математической модели, получим реальный или производственный тест, в котором должны быть указаны только входы $X = \{1, 2, 3\}$ и выходы $Z = \{y'\}$ схемы,

$T =$

1	2	3	y'	
1	1	0	1	c_1
×	0	1	1	t_3
×	1	1	0	$\tilde{t}_3$
0	1	0	0	t_1
1	1	0	1	$\tilde{t}_1$
×	1	0	1	t_4

$\cup$

1	2	3	y'	
×	1	1	0	$\tilde{t}_4$
1	0	0	0	t_2
0	×	0	0	t_5'
1	1	0	1	$\tilde{t}_2$
0	0	×	1	t_5

В общем виде при разбиении схемы на подсхемы часть подсхем комбинационных или триггерных могут не иметь непосредственной связи с внешними выходами схемы. Для этих невыведенных подсхем необходимо решать задачу последовательного распространения активизации от подсхемы к подсхеме. Если некоторая подсхема z_i соединена с внешним выходом через подсхему z_j, то все активизации путей из z_i должны быть проведены через подсхему z_j путем активизации путей из z_j, начинающихся с z_i.

На рис. 3.9 приведен пример разбиения схемы на две подсхемы, где комбинационная подсхема z связана с внешним выходом через подсхему y'.

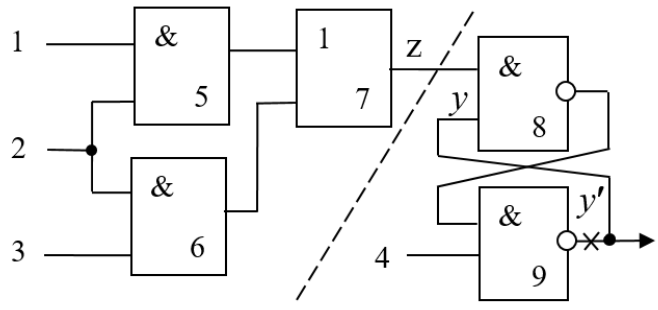


Рис. 3.9. Последовательностная схема, разбитая на две подсхемы

Построим эквивалентные покрытия для подсхем:

$$C^э(z)=\left\{\begin{array}{cccc|c|c} 1 & 2_5 & 2_6 & 3 & z & \\ \hline 1 & 1 & \times & \times & 1 & c_1 \\ \times & \times & 0 & 1 & 1 & c_2 \end{array}\right. \qquad C^э(\bar{z})=\left\{\begin{array}{cccc|c|c} 1 & 2_5 & 2_6 & 3 & z & \\ \hline 0 & \times & 1 & \times & 0 & c_4 \\ 0 & \times & \times & 0 & 0 & c_5 \\ \times & 0 & \times & 0 & 0 & c_6 \end{array}\right. .$$

$$C^э(y')=\left\{\begin{array}{ccc|c|c} z & 4 & y & y' & \\ \hline \times & 0 & \times & 1 & c_7 \\ 1 & \times & 1 & 1 & c_8 \end{array}\right. \qquad C^э(\bar{y}')=\left\{\begin{array}{ccc|c|c} z & 4 & y & y' & \\ \hline 0 & 1 & \times & 0 & c_9 \\ \times & 1 & 0 & 0 & c_{10} \end{array}\right. .$$

Проведем активизацию в подсхеме z. Для l_1 = 157 получим:

$$\begin{array}{c} 12_52_63z \\ \hline 1\,1\times\times 1 \\ 0\times 1\times 0 \\ \hline y\,1\,1\times y \end{array} \cap; \quad \begin{array}{c} y\,11\times y \\ 1\,1\times\times 1 \\ \hline 1'11\times 1' \end{array} \quad \text{и} \quad \begin{array}{c} y\,1\,1\times y \\ 0\times 1\times 0 \\ \hline 0'11\times 0' \end{array}$$

Так как пути из z должны быть активизированы через подсхему y', то в y' необходимо активизировать путь l = z89.

Итак,

$$\begin{array}{c} z4yy' \\ \hline 1\times 11 \\ 01\times 0 \\ \hline y1\,1y \end{array} \cap; \quad \begin{array}{c} y\,11y \\ 1\times 11 \\ \hline 1'111' \end{array} \quad \text{и} \quad \begin{array}{c} y\,11\,y \\ 0\,1\times 0 \\ \hline 0'110' \end{array}$$

Теперь необходимо так совместить наборы для подсхем z и y', чтобы активные значения по координате z у них совпадали. Объединение проведем в едином формате:

$$\begin{array}{c} 12_52_63z\,4yy' \\ \hline 1'1\,1\times 1'\times\times\times \\ \times\times\times\times 1'111' \\ \hline 1'1\,1\times 1'111' \end{array} \cap; = t_1 \quad \text{и} \quad \begin{array}{c} 12_52_63z\,4yy' \\ \hline 0'1\,1\times 0'\times\times\times \\ \times\times\times\times 0'110' \\ \hline 0'1\,1\times 0'110' \end{array} \cap = \tilde{t}_1$$

Для пары наборов $t/\tilde{t}_1$ необходимо предварительно выработать значение $Y_{t_1} = y = 1$. Это значение можно получить из $y = \times$ с помощью куба $c_7 \in C^э(y')$, который и является установкой. Объединив координаты 2_5 и 2_6 и удалив внутреннюю координату z, получим сегмент теста

$$T_1=\left\{\begin{array}{cccccc|c|c} 1 & 2 & 3 & 4 & y & y' & & \\ \hline \times & \times & \times & 0 & \times & 1 & c_7 & \\ 1 & 1 & \times & 1 & 1 & 1 & t_1, & \\ 0 & 1 & \times & 1 & 1 & 0 & \tilde{t}_1 & \end{array}\right.$$

на котором пара $t_1/\tilde{t}_1$ обеспечивает активизацию пути l=15798, а куб c_7 обеспечивает установку схемы в требуемое для проверяющей последовательности $\{t_1, \tilde{t}_1\}$ внутреннее состояние.

Таким способом можно построить весь тест из сегментов, содержащих проверяющую последовательность, которая активизирует заданный путь, и установочную последовательность. Так как каждый сегмент имеет свою собственную установку, то порядок сегментов в тесте является безразличным, в отличие от порядка наборов в каждом сегменте, который является строго определенным.

Алгоритм активизации в общем случае состоит в следующем:

1. Намечается некоторый путь l, состоящий из отрезков l_i, частных путей в подсхемах, от некоторого входа схемы до одного из ее выходов.

2. По эквивалентным покрытиям проводится активизация по подсхемам.

3. При активизации путей для любой пары $t=X_tY_tY_t'$ и $\tilde{t}=X_{\tilde{t}}Y_{\tilde{t}}Y_{\tilde{t}}'$, активизирующей некоторый сигнал y_i, т. е. $y_i = y_i'$ в наборе t и $y_i \neq y_i'$ в наборе $\tilde{t}$, должна быть обеспечена устойчивостью наборов t и $\tilde{t}$. Для всех $y_j \neq \times$, где $j = 1, 2, \ldots, k$ и $j \neq i$, должно быть обеспечено $y_j = y_j'$, т. е. вычисление y_j' кубами из покрытий $C_p(y_j')$ для наборов t и $\tilde{t}$.

4. Наборы, активизирующие отрезки, объединяются между собой так, чтобы активные значения, p' на выходе i-го и входе $(i+1)$-го отрезка пути совпадали.

5. Объединенные наборы упорядочиваются в пару наборов $t/\tilde{t} = X_tY_tY_t'/X_{\tilde{t}}Y_{\tilde{t}}Y_{\tilde{t}}'$, где $Y_t \supseteq Y_t'$, $Y_{\tilde{t}} \supseteq Y_{\tilde{t}}'$ и $Y_{\tilde{t}} \supseteq Y_t'$.

6. Если для некоторого пути l не удается построить объединенные наборы, то путь l разбивается на две части l_1 и l_2 так, чтобы существовали пары $t_1/\tilde{t}_1$ и $t_2/\tilde{t}_2$, объединение которых дает проверяющую последовательность $\{t_1,\ t_2/\tilde{t}_1,\ \tilde{t}_2\}$. Здесь $\{t_1,\ t_2\}$ есть пара нейтральных наборов, активизирующих путь l по частям l_1 и l_2 за два такта работы схемы. Аналогично $\{\tilde{t}_1, \tilde{t}_2\}$ есть переключающие наборы для активизации l также за два такта.

7. Если нейтральная $\{\tilde{t}_1, \tilde{t}_2\}$ и переключающая $\{t_1,\ t_2\}$ пары не стыкуются между собой по классам состояний, т. е. $Y_{\tilde{t}_1} \notin Y'_{t_2}$, то активизация проводится раздельно, и для каждой пары $\{t_1,\ t_2\}$ и $\{\tilde{t}_1, \tilde{t}_2\}$ строятся свои установки.

Проиллюстрируем алгоритм на примере схемы, показанной на рис. 3.9.

1. Наметим путь l_3=26789, проходящий через две подсхемы z и y'. Отрезки пути в подсхемах будут 267z и z89.

2. Проведем активизацию отрезков:

$$\begin{array}{l} 12_5 2_6 3z \\ \times\times 011 \\ 0\times 1\times 0 \\ \hline 0\times y1y \end{array} \cap; \quad \begin{array}{l} 0\times y\,1y \\ \times\times 0\,11 \\ \hline 0\times 0'11' \end{array} \quad \text{и} \quad \begin{array}{l} 0\times y\,1y \\ 0\times 1\,\times 0 \\ \hline 0\times 1'10' \end{array} \quad \text{для } l_3^1 = 267z.$$

$$\begin{array}{l} z4yy' \\ \hline 1\times11 \\ 01\times0 \\ \hline y11y \end{array} \cap; \quad \begin{array}{l} y\,11y \\ 1\times11 \\ \hline 1'111' \end{array} \quad \text{и} \quad \begin{array}{l} y\,11y \\ 0\,1\times0 \\ \hline 0'110' \end{array} \quad \text{для } l_3^2 = z89.$$

3. Объединим наборы по принципу совпадения активностей по координате z:

$$\begin{array}{l} 12_5 2_6 3z\,4yy' \\ \hline 0\times0'11'\times\times\times \\ \times\times\times1'111' \\ \hline 1\times0'11'111' = t_3 \end{array} \cap; \quad \text{и} \quad \begin{array}{l} 12_5 2_6 3z\,4yy' \\ \hline 0\times1'10'\times\times\times \\ \times\times\times0'110' \\ \hline 0\times1'10'110' = \tilde{t}_3 \end{array} \cap$$

4. Упорядочив объединённые наборы так, чтобы $Y_{\tilde{t}_3}=Y'_{t_3}$ и $Y'_{t_3}=Y_{t_3}$, получим пару $t_3 / \tilde{t}_3 = (1\times0'11'111' / 0\times1'10'110')$. После удаления промежуточного значения по координате z и объединения 2_5, 2_6 получим

$$P_3 = \left\{\begin{array}{cccccc|c} 1 & 2 & 3 & 4 & y & y' & \\ \hline 0 & 0' & 1 & 1 & 1 & 1' & t_3 \\ 0 & 1' & 1 & 1 & 1 & 0' & \tilde{t}_3, \end{array}\right.$$

Установку для $Y_{t_3} = y = 1$ вычислим кубом $c_7 = \times0\times1$ из $C^э(y')$. Итак, сегмент теста для проверки неисправностей пути $l_3 = 26789$:

$$T_3 = \left\{\begin{array}{cccccc|c} 1 & 2 & 3 & 4 & y & y' & \\ \hline \times & \times & \times & 0 & \times & 1 & L_3 \\ \hline 0 & 0 & 1 & 1 & 1 & 1 & \\ 0 & 1 & \times & 1 & 1 & 0 & P_3. \end{array}\right.$$

Построение теста заканчивается, если все отрезки путей по подсхемам активизированы на каком-либо сегменте T^j из теста T схемы.

3.2. Решение установочной задачи

Необходимость поиска установочных последовательностей возникает в двух случаях. В первом случае необходимо вычислить логические значения во внутренних точках схемы, которые были потребованы из условий активизации некоторого пути l набором t. Во втором случае в последовательностной схеме условия активизации на наборе $t = X_t Y_t Y'_t$ требуют предварительной установки схемы в состояние Y_t, которое невозможно вычислить в том же самом наборе t. Вычисление Y_t проводится с помощью специальной серии наборов $L=\{t_0, \ldots, t_r\}$, где $Y_{t_0} = (\times\times\times \ldots \times)$ есть класс безразличных состояний сигналов обратной связи и $Y'_{t_r} = Y_t$ для набора t.

Первая задача носит комбинаторный характер. Ее решение может быть получено с помощью π-алгоритма со следующим ограничением.

Любой куб, используемый для вычисления внутренних значений, должен давать непустое пересечение с набором t и, естественно, с накопленным результатом. Далее, при вычислении значений необходимо запретить опору на активные координаты p' на наборе t, т. е. разрешается использовать в π-алгоритме только те кубы, которые содержат неопределенное значение × по координатам, которые являются активными в наборе t.

Дело в том, что условия активизации, вычисляемые по π-алгоритму, должны существовать как в исправной схеме, так и в схеме с неисправностью. Эта неисправность изменяет активные значения p' на противоположные $\overline{p'}$. Если в π-алгоритме был использован куб, имеющий некоторую координату, равную p', то в схеме с неисправностью этот куб будет недействительным и может вычислять совсем не то значение, которое требовалось. Это может привести к компенсации логического значения при неисправности на выходе схемы и, следовательно, к отсутствию проверки на наборе t искомой неисправности.

Рассмотрим решение первой задачи на примере. Пусть задана схема, показанная на рисунке 3.10. Для простоты разобьем схему на подсхемы так, чтобы каждый ее элемент образовал подсхему. В этом случае алгоритм активизации вырождается в поэлементную процедуру, а решение установки опирается на обычные вырожденные покрытия элементов схемы. Составим единый формат схемы, пронумеровав все ее отрезки, Ф=(1235$_7$5$_6$64789).

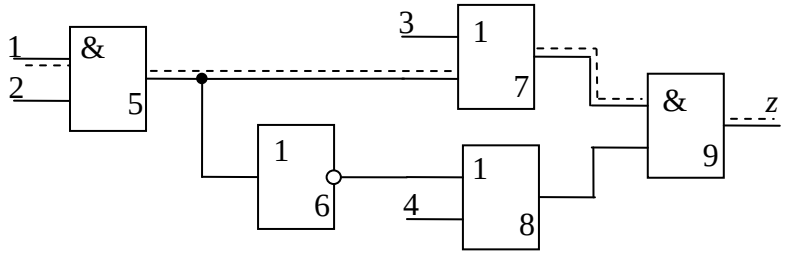


Рис. 3.10. Комбинационная схема с активным путем l_2=1579

Запишем вырожденные покрытия в едином формате Ф (табл. 3.3).

Таблица 3.3

1 2 5	3 5_7	5_6	6	4	7	8	9	Примечания
0 × 0 × 0 0								*C*(5) – покрытие элемента 5
		0 1	1 0					*C*(6) – покрытие элемента 6
	1 × × 1 0 0				1 1 0			*C*(7) – покрытие элемента 7
			1 × × 1 0 0					*C*(8) – покрытие элемента 8
					0 × × 0 1 1		0 0 1	*C*(9) – покрытие элемента 9, выход схемы *z*

Проведем активизацию пути l_2=155_779. Вычисления сведем в табл. 3.4 и выполним только для одного, например, нейтрального, набора t_2. Второй переключающий набор $\tilde{t}_2^{\prime}$ получается при указанных выше ограничениях на установку простой инверсией значений по активным координатам.

Таблица 3.4

1 2 5	3 5_7	5_6 6 4	7	8	9	Примечания
0 × 0 1 1 1 *y* 1 *y* 1 1 1 1′1 1′	∩ ∩ *y*					Активизация входа 1 через выход 5-го элемента в 1′
	× 1 0 0	∩	1 0			Активизация входа 5 через выход 7-го элемента при условии 5=5_7 =1′
	0 *y* × 1	∩ *y*	*y* 1			
	0 1′		1			
		∩	0 × 0 1 1 1		0 1	Активизация входа 7 через выход 9-го элемента при условии 7=7_9 =1′
		∩ *y*	*y* 1 1 1 1 1		*y* 1	
			1′ 1		1′	
1′ 1 1′	0 1′	× × ×	1′ 1		1′	t_2′ для l_2=155_779

Из табл. 4.2 видно, что условия активизации пути l_2=155_779 выполняются, если в схеме будут вычислены значения: 2=1, 3=0 и 8=1. Так как 2 и 3 являются внешними входами, то их значения устанавливаются непосредственно на входах схемы. Значение 8=1 является внутренним и его необходимо вычислить через внешние входы схемы. Вычисления сведены в таблицу 3.5.

Таблица 3.5

1 2 5	3 5_7	5_6	6	4	7	8	9	Примечания
1′ 1 1′	0 1′	×	×	×	1′	1	1′	t_2' для l_2=155_779
			1 ×	× 1		1 1	c_1 c_2	Покрытие $C(8)$ для 8=1
		0	1					Покрытие $C(6)$ для 6=1, $a_1 \cap C(6) = \varnothing$
1 1 1	0 1	0	1	×	1	1	1	
1 1 1	0 1	×	×	1	1	1	1	$a_2 = t_2$, $p' \leftarrow \overline{p'}$ в $a_2 = \tilde{t}_2$
0 1 0	0 0	×	×	1	0	1	0	

В покрытии $C(8)$ содержатся два куба c_1 и c_2, поэтому в результате пересечения с набором t_2' получаем также два набора a_1 и a_2. Рассмотрим первоначально набор a_1, в котором появлялась необходимость в вычислении внутреннего значения 6=1. В результате пересечения a_1 с покрытием $C(6)$ получаем пусто, так как эквивалентные по выходу 5-го элемента координаты 5_7=1 и 5_6=0 требуют противоположных значений.

Вычисления в наборе a_2 закончены, так как потребованное значение 4=1 есть значение на независимом входе схемы.

Итак, положив нейтральный набор $t_2 = a_2$, получим из него переключающий набор $\tilde{t}_2'$ простой инверсией активных координат, т.е. простой заменой p' на $\overline{p'}$. Такая замена возможна, так как все внутренние значения в t_2 вычислены без опоры на активные координаты и, следовательно, не изменят своих значений от смены значений по активным координатам.

Выбор активных значений p' или $\overline{p'}$ был произволен на первом шаге активизации пути l_2=155_779 (см. табл. 3.4) и не влиял в дальнейшем на заказы при установке внутренних значений, которые, следовательно, являются одинаковыми как для нейтрального, так и для переключающего наборов. Таким образом, переходя к формату Ф=(1234z), получим пару наборов, активизирующих путь l_2=155_779:

1	2	3	4	z	
1′	1	0	1	1′	t_2
0′	1	0	1	0′	$\tilde{t}_2'$.

В общем случае решение установочной задачи сводится к синтезу последовательности, обеспечивающей переход схемы из класса безразличных состояний в искомое состояние схемы. Решение этой задачи для сложных схем состоит в построении фрагментов частично определенного графа переходов, а именно путем поиска переходов на один такт работы схемы.

Например, если искомое состояние схемы, которое будем называть исходным заказом на синтез установки, есть $\tilde{Y}_1 = p_1 p_2$, то все переходы на один такт в состояние $Y' = \tilde{Y}_1$ определяются покрытием

$$C(\tilde{Y}_1) = C(y_1^{p_1}) \cap C(y_2^{p_2}).$$

Пересечение выполняется в формате Ф = (XYY'), в котором классы исходных состояний схемы определяются значениями координат из Y, а им соответствующие входные воздействия – значениями из X.

После того, как найдены исходные классы состояний для $Y' = \tilde{Y}_1$, выбирается новый заказ $\tilde{Y}_2$ из найденных исходных классов, и так до тех пор, пока на некотором i-м шаге рекурсии не будет получено $\tilde{Y}_r$ = (×××…×), т.е. класс безразличных состояний сигналов обратной связи.

Вычисление любого заказа $\tilde{Y}$ должно выполняться с соблюдением следующих правил:

1. Вычисления проводятся в формате Ф = (XYY') и для каждого набора c из пересечения покрытий $C(\tilde{Y})$ необходимо, чтобы $Y_c \supseteq Y'_c$ и $Y'_c \subseteq \tilde{Y}$. Условие $Y_c \supseteq Y'_c$ должно исключить состязания сигналов во время переходного процесса из состояния Y_c в состояние Y'_c под действием входного воздействия X_c. Действительно, для всех сигналов $y'_i = p$ из Y'_c им соответствующие сигналы $y_i = p \vee \times$. Если $y_i = p$ и $y'_i = p$, то сигнал $y_i = y'_i$ на данном переходе не изменяется. Если же $y_i = \times$ и $y'_i = p$, то значение $y'_i = p$ вырабатывается на данном переходе из безразличного значения $y_i = \times$. Таким образом, все сигналы $y'_i = p$ будут либо храниться неизменными на данном переходе, либо вырабатываться вне зависимости от безразличных сигналов $y_i = \times$ в исходном состоянии Y_c, т.е. состояние Y'_c будет вычисляться без критических состязаний сигналов.

Условие $Y'_c \subseteq \tilde{Y}$ обеспечивает достижение класса $\tilde{Y}$ при достижении любой его грани. При вычислении класса $\tilde{Y}$ в действительности приходится вычислять его грани Y'_c, что является достаточным условием достижения класса $\tilde{Y}$, так как все сигналы $\tilde{y}_i = p$ будут вычислены в Y'_c, в котором также будут вычислены некоторые $y'_i = p$, для которых $\tilde{y}_i = \times$ являются безразличными в исходном заказе.

2. Для сокращения лишних действий из покрытия $C(\tilde{Y})$ должны быть удалены все кубы, которые являются гранями других кубов. Итак, после выполнения каждой итерации вычисления: $C(\tilde{Y}) = C(\tilde{Y}) - \{a/a \subseteq b$ и $a, b \in C(\tilde{Y})\}$.

3. Для устранения зацикливания при синтезе $L=\{t_0, \ldots t_{i-1}, t_i, t_{i+1}, \ldots, t_r\}$ на основе $C(\tilde{Y}_i)$ для любого набора t_i из L должны выполняться условия: $\tilde{Y}_i = Y_{t_{i+1}}, Y_{t_i} \subseteq Y_{t_{i+1}} \cup \ldots \cup Y_{t_r})$ и $Y_{t_i} = \tilde{Y}_{t_{i-1}}$, для i =0,1, …, r-1.

4. Первоначально переход от формата XY в формат XYY' осуществляется путем приравнивания значений $y'_j = y_j$ и $y'_i = p$ для $j \neq i$ и i = 0, 1, …, k в любом кубе $c \in C(y_i'^p)$, так как между значениями y'_j и y_j лежит всего лишь фиктивная точка разрыва и $y'_i = p$ обеспечено по определению покрытия $C(y_i'^p)$.

Рассмотрим пример синтеза установочных последовательностей для схемы, показанной на рис. 3.11. Введем две точки фиктивного разрыва на входе 7-го и 12-го элементов и обозначим сигналы соответственно через y'_1 и

y_2'. При построении модели примем также, что сигнал с выхода 6-го элемента является инверсией сигнала y_1', т.е. *RS*-триггер, выполненный на элементах 6 и 7, работает в парафазном коде.

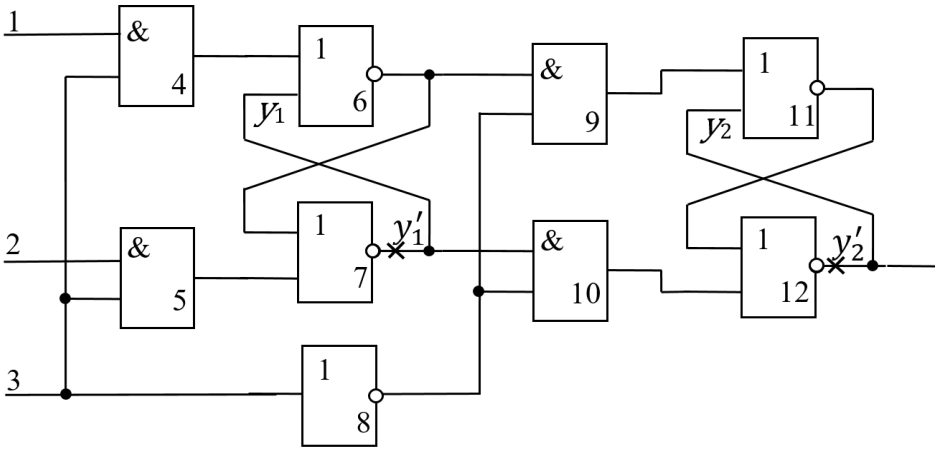


Рис. 3.11. Схема для синтеза установочной последовательности

Постоим эквивалентные покрытия для сигналов обратной связи y_1' и y_2':

$C^э(y_1')$:

1	3	2	3	y_1	y_1'	
1	1	0	×	×	1	c_1
×	×	×	0	1	1	c_2
×	×	0	×	1	1	c_3
1	1	×	0	×	1	∅

$C^э(y_2')$:

y_1	3	y_1'	3	y_2	y_2'	
0	0	0	×	×	1	c_7
×	×	×	1	1	1	c_8
×	×	0	×	1	1	c_9
0	0	×	1	×	1	∅

$C^э(\overline{y_1'})$:

×	×	1	1	×	0	
0	×	×	×	0	0	c_4
×	0	×	×	0	0	c_5
						c_6

$C^э(\overline{y_2'})$:

×	×	1	0	×	0	
1	×	×	×	0	0	c_{10}
×	1	×	×	0	0	c_{11}
						c_{12}

Объединив эквивалентные координаты и перейдя в формат
Ф = $(123y_1y_2y_1'y_2')$, получим покрытия для решения установочной задачи:

$C(y_1')$:

1	2	3	y_1	y_2	y_1'	y_2'	
1	0	1	×	×	1	×	c_1
×	×	0	1	×	1	×	c_2
×	0	×	1	×	1	×	c_3

$C(\overline{y_1'})$:

×	1	1	×	×	0	×	c_4
0	×	×	0	×	0	×	c_5
×	×	×	0	×	0	×	c_6

$C(y_2')$:

1	2	3	y_1	y_2	y_1'	y_2'	
×	×	0	×	×	0	1	c_7
×	×	1	×	1	×	1	c_8
×	×	×	×	1	0	1	c_9

$C(\overline{y_2'})$:

×	×	0	×	×	1	0	c_{10}
1	×	×	0	0	1	0	c_{11}
×	1	×	0	0	×	0	c_{12}

Поскольку исходным заказом на установку служит исходное состояние схемы для первого набора проверяющей последовательности, то первоначально проведем синтез какой-либо проверяющей последовательности. Выберем некоторый путь *l*= 3-8-9-11-12, для которого

y_1'	3	y_1'	3	y_2	y_2'
0	0	0	×	×	1
×	1	×	×	0	0
0	*y*	0	×	0	*y*

$\cap$

y_1'	3	y_1'	3	y_2	y_2'	
0	*y*	0	×	0	*y*	$\cap$ *y*
0	0	0	×	×	1	
0	0′	0	×	0	1′	$= \tilde{t}$

y_1'	3	y_1'	3	y_2	y_2'	
0	*y*	0	×	0	*y*	$\cap$ *y*
×	1	×	×	0	0	
0	1′	0	×	0	0′	$= t$.

Объединяя эквивалентные координаты, получим наборы *t* и $\tilde{t}$ в формате Ф:

1	2	3	y_1	y_2	y_1'	y_2'	
×	×	1′	×	0	0	0′	*t*
×	×	0′	×	0	0	1′	$\tilde{t}$.

Обеспечим значение y_1'= 0 в наборах *t* и $\tilde{t}$ без опоры на активные координаты путем пересечения *t* и $\tilde{t}$ с покрытием $C(\overline{y_1'})$. Пересечение с кубом c_4 буде пусто, так как он опирается на значение 1 по третьей координате, которая является активной в *t* и $\tilde{t}$; пересечение с кубом c_6 пусто. Таким образом, подходит только куб c_5, который дает:

1	2	3	y_1	y_2	y_1'	y_2'	
0	×	1	0	0	0	0	*t*
0	×	0	×	0	0	1	$\tilde{t}$.

$P=$

Итак, $\tilde{Y}$ = Y_t и $\tilde{Y}$ = (00) равняется исходному заказу на синтез установки для *P*. Вычисления $\tilde{Y}$ сведем в табл. 3.6.

На первом шаге исходный заказ на пересечение $\tilde{Y}_1$=00, и само пересечение содержит два куба, из которых второй куб определяет переход в то же самое состояние *Y*=00. Первый куб выводит за исходное состояние, и поэтому $\tilde{Y}_2$=×0. Иллюстрация кубов в виде фрагмента частично определенного графа переходов (рис. 3.12) наглядно показывает выбор класса состояний (×0) в качестве исходного заказа $\tilde{Y}_2$ для второго шага вычислений.

Вычисления заканчиваются на третьем шаге, так как достигнут класс безразличных состояний (××).

Таблица 3.6

Номер шага	X	Y	Y'	Примечание
	123	$y_1 y_2$	$y_1' y_2'$	
1	×××	× ×	0 0	$\tilde{Y}_1$=00
	×11	× 0	0 0	$C(\tilde{Y}_1) = C(\bar{y}_1') \cap C(\bar{y}_2')$
	0×1	0 0	0 0	
2	×××	× ×	× 0	$\tilde{Y}_2$=×0
	×0×	1 ×	1 0	$c_{10} \cap C(y_1')$
	××0	1 0	1 0	$c_{11} \cap C(y_1')$
	××1	× 0	× 0	c_{12}
3	×××	××	1 0	$\tilde{Y}_3$=1×
	101	× ×	1 ×	$C(y_1')$
	××0	1 ×	1 ×	
	×0×	1 ×	1 ×	

На рис. 3.12 дано объединение фрагментов из табл. 3.6 в виде частично определенного графа переходов, по которому легко осуществить синтез установочной последовательности.

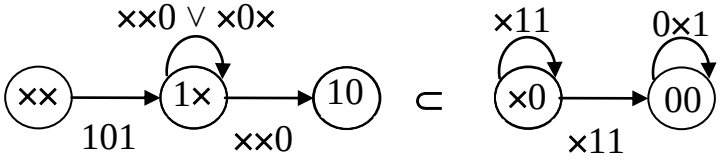


Рис. 3.12. Фрагмент частично определенного графа переходов схемы

Итак, сегмент теста T_1 для проверки пути l = 3 – 8 – 9 – 11 – 12 в формате Φ=(XYY')

$$T_1 = \left\{\begin{array}{c|ccccc|cc} & 1 & 2 & 3 & y_1 & y_2 & y_1' & y_2' \\ \hline & 1 & 0 & 1 & \times & \times & 1 & \times \\ & \times & \times & 0 & 1 & \times & 1 & 0 \\ & \times & 1 & 1 & \times & 0 & 0 & 0 \\ \hline & 0 & \times & 1 & 0 & 0 & 0 & 0 \\ & 0 & \times & 0 & 0 & 0 & 0 & 1 \end{array}\right. \begin{array}{l} \\ \\ L \\ \\ P \end{array}$$

Так как состязания сигналов в самих наборах сегмента устранены по построению, то для окончательного устранения критических состязаний сигналов в схеме на сегменте T осталось устранить критический риск статического сбоя сигналов обратной связи при переходе от набора к набору.

3.3. Устранение риска статического сбоя

Состязание сигналов в схемах возникают из-за задержек на элементах и в линиях связи между элементами. Различные сигналы проходят разные пути в схемах. Вследствие этого значения сигналов обратной связи в асинхронных схемах изменяются в произвольные моменты времени.

Рассогласование сигналов во времени может проявляться в работе схемы в виде:

- генерации сигналов, когда схема вследствие этого не приходит ни в одно из устойчивых состояний;
- критических состязаний сигналов, в результате которых схема после окончания переходного процесса может оказаться в состоянии, отличающемся от состояния, которое определено законом ее функционирования;
- риска статического сбоя для тех сигналов обратной связи, которые должны сохранять свои значения на некотором переходе, но из-за сбоев, т. е. кратковременных ложных значений, принимают противоположные значения и эти измененные значения являются устойчивыми после окончания переходного процесса.

Состязания первых двух типов устраняются при синтезе наборов. Необходимо, чтобы для любого набора теста $A = X_A Y_A Y'_A$ выполнялось условие

$$Y_A \supseteq Y'_A.$$

Риск статического сбоя проявляется при переходе от набора к набору, и для его устранения необходимо синтезировать специальные промежуточные наборы. Например, для некоторого перехода $A/B = X_A Y_A Y'_A \,/\, X_B Y_B Y'_B$, при $Y'_A \subseteq Y_B$, все сигналы обратной связи, вырабатываемые на наборе A и необходимые для Y_B, должны быть переданы при переходе от A к B без риска сбоя. В частном случае, можно положить, что $Y_B = Y'_A \mu Y'_B$, т. е. равняется обычному μ-произведению кубов, которые задают классы состояний Y'_A и Y'_B. Это хорошо иллюстрирует тот факт, что на переходе A/B нужно хранить без риска только те сигналы, которые не изменяются и равны $p = 0, 1$ в Y'_A и Y'_B.

Рассмотрим явление статического сбоя на примере схемы, показанной на рис. 3.13.

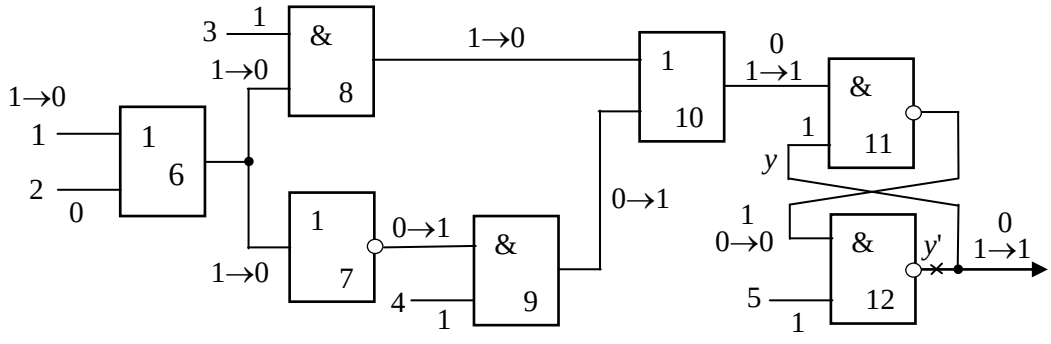


Рис. 3.13. Явление риска статического сбоя на переходе *A/B*

Для заданного на рисунке перехода изменяется только один сигнал по первому входу схемы, который является входом элемента 6. Это изменение вызывает переключение сигнала по первому входу 10-го элемента с 1 на 0 (показано как 1→0), которое компенсируется изменением 0→1 по второму входу этого же элемента.

Итак, без учета задержек сигнал на выходе 10-го элемента не должен изменяться на данном переходе и быть равным 1. Однако вследствие того, что компенсирующий сигнал по второму входу 10-го элемента проходит путь на один 7-й элемент больше, то он будет изменять свое значение с некоторым запаздыванием (предполагается равенство задержек на всех элементах схемы) по отношению к изменению сигнала на первом входе. Поэтому на входах 10-го элемента могут существовать сигналы, одновременно равные 0, и, следовательно, на выходе элемента 6, в соответствии с его логической функцией ИЛИ, может появиться ложный сигнал 0, что и обозначено на рис. 5.1 как $\overset{0}{1\to1}$.

Далее, значение 0 на выходе 10-го элемента имплицирует 1 на 11-м и 0 на 12-м элементах. Это ложное значение 0 на выходе 12-го элемента является устойчивым и сохраняется после окончания переходного процесса в схеме.

Анализ и устранение статического сбоя будем проводить по покрытиям.

Определение. В схеме *N*, выход которой идентифицирован как *f*, существует риск статического сбоя на переходе *A/B*, если на наборах *A* и *B* значение *f* не изменяется и равно *p*, а во время переходного процесса на выходе схемы может появиться ложное значение $f = \bar{p}$.

Покрытия, построенные по π-алгоритму, сохраняют состязательные свойства схемы. Если построено покрытие $C(f^p)$, то значение $f = p$ на выходе схемы имплицируется любым кубом $c \in C(f^p)$ с помощью своих границ $c_i \neq \times$.

Это свойство обеспечивается алгоритмом построения кубов покрытия $C(f^p)$. Действительно, если некоторая координата $c_i = \times$ в кубе *c* из $C(f^p)$, то любое значение 0 или 1, которое может принять соответствующий сигнал на входе схемы, и, следовательно, любой переход 0→1 или 1→0 не будет влиять на выходное значение $f = p$. Следовательно, если границы $c_i \neq \times$ не изменяются, то $f = p$ хранится кубом *c* без риска сбоя.

Итак, если для некоторого перехода *A/B* в покрытии $C(f^p)$ существует куб *c* такой, что $c \supset A$ и $c \supset B$, то на данном переходе $f = p$ сохраняется без риска статического сбоя. Другими словами, переход между любыми двумя гранями одного и того же куба осуществляется без риска статического сбоя.

Устранение состязаний, являющихся следствием статического сбоя, сводится к синтезу промежуточных наборов таким образом, чтобы любой переход лежал в границах какого-либо куба из покрытия, т. е. любые два соседних набора должны быть гранями одного и того же куба.

Построим вырожденное покрытие для схемы, показанной на рис. 3.13, для которой на переходе *A/B*=(1011111/0011111), при Ф=(12345*yy'*), существует риск статического сбоя:

$$C(12)=\left\{\begin{matrix}\times & \times & \times & \times & 0 & \times & 1\\ 0 & 0 & \times & 1 & \times & 1 & 1\\ \times & 1 & 1 & \times & \times & 0 & 0\\ 1 & \times & 1 & \times & \times & 1 & 1\end{matrix}\right.\begin{matrix}c_D\\ c_B\\ \\ c_A\end{matrix}\cup\left\{\begin{matrix}0 & 0 & \times & 0 & 1 & \times & 0\\ \times & \times & 0 & 0 & 1 & \times & 0\\ \times & 1 & 0 & \times & 1 & \times & 0\\ 1 & \times & 0 & \times & 1 & \times & 0\\ \times & \times & \times & \times & 1 & 0 & 0\end{matrix}\right\}$$

Действительно, в покрытии *C*(12) не существует куба, содержащего оба набора *A* и *B*, и, следовательно, в схеме на переходе *A/B* существует риск статического сбоя.

Для устранения состязаний найдем в покрытии *C*(12) кубы $c_A \supseteq A$, $c_B \supseteq B$ и куб c_Dтакой, что $c_D \cap c_A \neq \varnothing$ и $c_D \cap c_B \neq \varnothing$. Тогда переходы от *A* к D_1, от D_1 к D_2 и от D_2 к *B*, проиллюстрированные на рис. 3.14, где $D_1 = c_A \cap c_D$ и $D_2 = c_B \cap c_D$, осуществляются без риска статического сбоя.

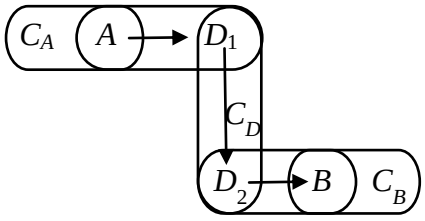


Рис. 3.14. Синтез переходов для устранения риска статического сбоя от набора *A* к набору *B*

Переход A/D_1 содержится в кубе c_A, переход D_1/D_2 – в кубе c_D и переход D_2/B – в кубе c_B, а весь переход $A/D_1/D_2/B$ заменяет переход *A/B*, на котором в схеме возможен риск статического сбоя.

В общем случае синтез переходов в формате схемы Ф=(*XYY'*) осуществляется следующим образом. Так как проверяющая последовательность является частным случаем установочной последовательности, то для любого перехода теста $A/B = X_A Y_A Y'_A / X_B Y_B Y'_B$ должны выполняться соотношения

$$Y_B \supseteq Y'_A, \quad Y_A \supseteq Y'_A \text{ и } Y_B \supseteq Y'_B,$$

$$C(Y_B) = \{c / Y'_C \subseteq Y_B \text{ и } Y_C \supseteq Y'_C\} - \{a / a \subseteq b \text{ и } a, b \subseteq C(Y_B)\},$$

и вычисление набора *A* проводится через вычисление покрытия

$$C(Y_B) = \{c / Y'_C \subseteq Y_B \text{ и } Y_C \supseteq Y'_C\} - \{a / a \subseteq b \text{ и } a, b \subseteq C(Y_B)\},$$

которое задает все переходы в класс состояний Y_B кубами максимальной размерности, не содержащими критических состязаний сигналов. Такие кубы из покрытия $C(Y_B)$ будем называть простыми кубами.

Понятие простого куба как куба, содержащего максимально возможное число безразличных значений *x*, является чрезвычайно актуальным для целей сокращения перебора при решении установочной задачи.

Пусть заданы два набора $A = X_A Y_A Y_A'$ и $B = X_B Y_B Y_B'$, и требуется устранить состязания при переходе от *A* к *B*, т. е. передать сигналы из $Y_B = Y'_A \mu Y'_B$ без риска статического сбоя. С этой целью вычислим в покрытии $C(Y_B)$ три куба, обладающих следующими свойствами:

1. Куб $c_A \supseteq A$ и $c_A = X_{c_A} Y_{c_A} Y'_A$, т. е. куб c_A является гранью куба A и вычисляет то же самое значение Y'_A. Так как $Y_{c_A} \supseteq Y'_A$ по построению, то $X_{c_A} Y'_A Y'_A$является устойчивым кубом.

2. Куб $c_D = X_{c_D} Y'_A Y'_A$, т. е. куб c_Dопределяет устойчивый переход из Y'_A в то же самое состояние Y'_A, и $c_D \cap c_A \neq \varnothing$. Непустое пересечение кубов c_D и c_A позволяет переходить от устойчивого вычисления состояния Y'_A кубом c_A или его гранью, набором A, к устойчивому хранению его кубом c_Dили какой-либо его гранью.

3. Куб $c_B = X_{c_B} Y_B Y_B$ такой, что $c_B \supseteq B$, и $c_B \cap c_D \neq \varnothing$. Условие не пустоты пересечения кубов c_B и c_D позволяет переходить от устойчивого хранения Y'_A, а следовательно и $Y_B \supseteq Y'_A$, гранью куба c_D к устойчивому хранению состояния Y_B гранью куба c_B. Условие включения $B \subset C_B$ позволяет переходить к грани B, которая вычисляет Y'_B на основе состояния Y_B, устойчивое хранение которого обеспечивается кубом c_B, а следовательно и любой его гранью (рис. 3.15, *а*).

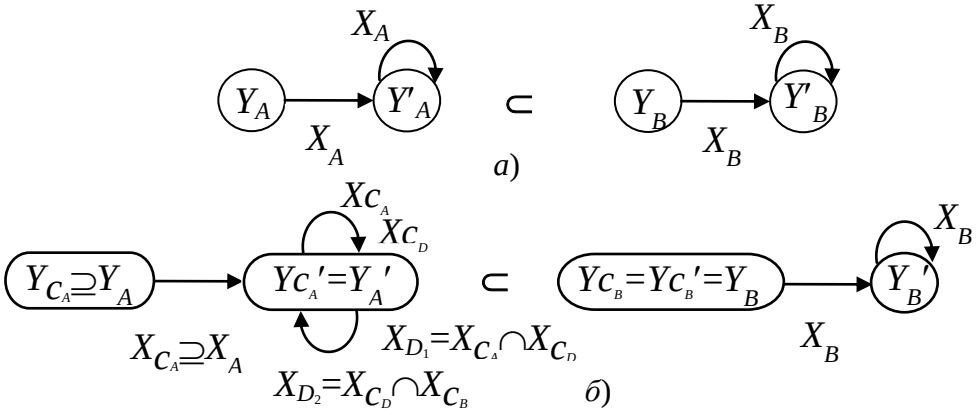


Рис. 3.15. Частично определенный граф переходов схемы для трех кубов: c_A, c_B и c_D.

Если в покрытии $C(Y_B)$ найдены три куба – c_A, c_B и c_D, то замена исходного перехода A/B на серию переходов $A/D_1/D_2/B$, где $D_1 = c_A \cap c_D$ и $D_2 = c_D \cap c_B$, устраняет риск статического сбоя. Заметим, что при $c_A = A$ исходный набор A заменяется на $A = X_{D_1} Y_A Y'_A$, так как D_1 в этом случае является гранью набора A, и $c_B \supset B$.

Покажем, что переходы от A к D_1, от D_1 к D_2 и от D_2 к B не содержат риска статического сбоя для состояния Y_B. Действительно, переход от A к D_1 происходит в пределах куба c_A, вычисляющего исходное состояние $Y'_A \subset Y_B$, так как $c_A \supseteq A$ и $c_A \supseteq D_1$. Переход D_1/D_2 лежит в пределах куба c_D, так как $c_D \supseteq D_1$ и $c_D \supseteq D_2$, и куб c_D обеспечивают по построению устойчивое хранение Y'_A, а следовательно, и $Y_B \supset Y'_A$. И, наконец, переход D_2/B лежит в пределах куба c_B, так как $c_B \supset D_2$ и $c_B \supset B$. Куб c_B по построению обеспечивает устойчивое хранение Y_B, а его грань – набор B – осуществляет переход из Y_B в Y'_B (рис. 3.15, *б*).

Рассмотрим алгоритм устранения состязаний для схемы, показанной на рис. 3.13, на примере построенного для нее сегмента теста в предыдущем

разделе 4. Для удобства перепишем сегмент теста заново без явного выделения установочной и проверяющей последовательностей, так как методика должна быть применима к любой последовательности. Итак,

$$T_1 = \begin{array}{|ccccc|cc|l} 1 & 2 & 3 & y_1 & y_2 & y_1' & y_2' & \\ \hline 1 & 0 & 1 & \times & \times & 1 & \times & t_1 \\ \times & \times & 0 & 1 & \times & 1 & 0 & t_2 \\ \times & 1 & 1 & \times & 0 & 0 & 0 & t_3, \\ 0 & \times & 1 & 0 & 0 & 0 & 0 & t_4 \\ 0 & \times & 0 & 0 & 0 & 0 & 1 & t_5 \end{array}$$

где наборы теста пронумерованы в порядке их подачи на схему в процессе тестового эксперимента.

1. Положим $A = t_1 = 101{\times}{\times}1{\times}$ и $B = t_2 = {\times}{\times}01{\times}10$, для которых $Y_B = Y_A'\mu Y_B' = (1{\times})\mu(10) = 1{\times}$. Вычислим по покрытиям (2) для схемы три куба c_A, c_B и c_D из покрытия переходов в класс состояний ($1x$) и результаты представим в табл. 3.7

Таблица 3.7

Номер шага	123	y_1y_2	$y_1'y_2'$	Примечание
1	×××	× ×	1 ×	$Y_B = Y_A'\mu Y_B'$
	101	× ×	1 ×	A
	××0	1 ×	1 0	B
2	101	× ×	1 ×	$c_A = A, c_B \supset B$
	××*0*	1 ×	1 ×	$c_D;\ c_D \cap c_A \neq \emptyset;$
	×0×	1 ×	1 ×	$c_D \cap c_B \neq \emptyset$
3	101	1 ×	1 ×	$D_1 = c_A \cap c_D$
	×00	1 ×	1 ×	$D_2 = c_D \cap c_B$
4	101	× ×	1 ×	$A = X_{D_1}Y_AY_A'$
	×00	1 ×	1 ×	D_2
	××0	1 ×	1 0	B

На первом шаге формируется исходный заказ на поиск переходов в Y_B=1× на основе исходных классов Y_A' и Y_B'.

На втором шаге вычисляется покрытие C(1×) с поиском в нем трех кубов c_A, c_B и c_D.

На третьем шаге формируются промежуточные наборы $D_1 = c_A \cap c_D$ и $D_2 = c_D \cap c_B$.

На четвертом шаге дается замена перехода A/B на серию $A/D_2/B$, где $A = c_{D_1}Y_AY_A'$, так как $c_A = A$, и поэтому D_1 является просто гранью набора A, сохраняющей свойство перехода из Y_A в Y_A'.

2. Положив $A = t_2$ и $B = t_3$, проведем устранение состязаний между наборами t_2 и t_3. Результаты вычислений показаны в табл. 3.8.

Таблица 3.8

Номер шага	123	y_1y_2	$y_1'y_2'$	Примечание
1	××× ××0 ×11	× × 1 × × 0	× 0 1 0 0 0	$Y_B = Y_A'\mu Y_B'$ A B
2	××0 ×0× ××1	1 × 1 0 × 0	1 0 1 0 × 0	$c_A = X_{c_A}Y_{c_A}Y_A'$ $c_D = X_{c_D}Y_A'Y_A'$ $c_B = X_{c_B}Y_BY_B \supset B$ $C(Y_B = x0)$
3	×00 ×01	1 0 1 0	1 0 1 0	$D_1 = c_A \cap c_D$ $D_2 = c_D \cap c_B$
4	×00 ×01 ×11	1 × 1 0 × 0	1 0 1 0 0 0	$A = X_{D_1}Y_AY_A'$ D_2 B

3. Положив $A = t_3$ и B = t_4, получим устранение состязаний при переходе от t_3 к .t_4 (табл. 3.9)

Таблица 3.9

Номер шага	123	y_1y_2	$y_1'y_2'$	Примечание
1	×11 0×1	× 0 0 0	0 0 0 0	A B
	×××	× ×	0 0	$Y_B = Y_A'\mu Y_B'$
2	×11 0×1	× 0 0 0	0 0 0 0	$c_A = A$ $c_B = B$ $C(Y_B = 00)$
3	011	0 0	0 0	$D_1 = D_2 = D = c_A \cap c_B$
4	011 011	× 0 0 0	0 0 0 0	$A = X_DY_AY_A'$ $B = X_DY_BY_B'$

На втором шаге вычислений в табл.3.9 получено, что $c_A = A$, $c_B = B$ и $c_B \cap c_A \neq \emptyset$, т. е. куб c_D произвольно можно положить как $c_D = c_A$, так и $c_D = c_B$. В соответствии с этим наборы D_1 и D_2 вырождаются в один набор $D = c_A \cap c_B$.

Так как $c_A = A$ и $c_B = B$, в тесте можно оставить один набор D, который является гранью как набора A, так и набора B с сохранением всех свойств наборов A и B одновременно.

На четвертом шаге даны две различные интерпретации набора *D*. Набор $A = X_D Y_A Y'_A$ отображает переходный процесс в схеме, а набор $B = X_D Y_B Y'_B$ характеризует состояние схемы после окончания переходного процесса в ней.

3. Так как набор t_4 изменился на предыдущем шаге вычислений, то, положив $A = t_4$ в новом значении и $B = t_5$, вычислим устранение состязания при переходе от t_4 к t_5 Полученная проверяющая последовательность представлена в табл. 3.10.

Таблица 3.10

Номер шага	123	$y_1 y_2$	$y'_1 y'_2$	Примечание
1	011 0×0 ×××	0 0 0 0 × ×	0 0 0 1 0 ×	A B $Y_B = Y'_A \mu Y'_B$
2	×11 0×	× × 0 ×	0 × 0 ×	$c_A \supset A$ $c_B \supset B$ $\}$ $C(Y_B = 0\times$
3	011	0 ×	0 ×	$D = D_1 = D_2 = c_A \cap c_B$
4	011 0×0	0 0 0 0	0 0 0 1	$A \subset D$ B

На втором шаге вычислений получаем, что $c_B \cap c_A \neq \varnothing$ и, следовательно, надобность в переходном кубе c_D исчезает. Найдя $D = c_A \cap c_B$, убеждаемся, что $D \supset A$, и надобности в промежуточном наборе *D* также нет, так как набор *A* является гранью *D* и сам выполняет его функции по устранению состязаний.

Запишем сегмент теста с устраненными состязаниями из табл. 3.10:

$T_2 =$

1	2	3	y_1	y_2	y'_1	y'_2	
1	0	1	×	×	1	×	t_1
×	0	0	1	×	1	×	
×	0	0	1	×	1	0	$\subset t_2$
×	0	1	1	0	1	0	
0	1	1	×	0	0	0	$\subset t_3$
0	1	1	0	0	0	0	$\subset t_4$
0	×	0	0	0	0	1	t_5

Исключив из теста переходный процесс, характеризуемый состоянием схемы *Y*, получим:

$T_3 =$

1	2	3	y'_1	y'_2
1	0	1	1	×
×	0	0	1	×
×	0	0	1	0
×	0	1	1	0
0	1	1	0	0
0	1	1	0	0
0	×	0	0	1

Объединение пересекающихся между собой наборов дает окончательный сегмент теста:

$$T_4 = \left\{\begin{array}{ccc|cc} 1 & 2 & 3 & y_1' & y_2' \\ \hline 1 & 0 & 1 & 1 & \times \\ \times & 0 & 0 & 1 & 0 \\ \times & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & \times & 0 & 0 & 1 \end{array}\right\}$$

Так как в схеме (рис. 3.13) имеется всего один внешний выход $z = y_2'$, снимаемый с 12 – го элемента, то производственный тест, который должен содержать значения только по внешним выходам схемы, окончательно примет вид

$$T = \left\{\begin{array}{ccc|c} 1 & 2 & 3 & z \\ \hline 1 & 0 & 1 & \times \\ \times & 0 & 0 & 0 \\ \times & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & \times & 0 & 1 \end{array}\right\}$$

Более подробно устранение статического риска сбоя при синтезе схем рассмотрено в Приложении 1.

Выводы по разделу

Вопросы построения тестов, рассмотренные в данном разделе, носят машинно-ориентированный характер, однако реализация методов в виде алгоритмов и создание на их основе систем программ, автоматизирующих процесс построения тестов для схем реальной сложности, т.е. схем, конструктивно оформленных в виде единого блока и содержащих десятки и сотни тысяч логических элементов, требует рассмотрения специальных задач.

Основная задача, которую необходимо решать при синтезе тестов схем реальной сложности, состоит в сокращении перебора, так как основными методами синтеза тестов являются методы направленного перебора. Если не применять специальных мер, т. е. вести перебор в той или иной мере случайно, то решение задачи синтеза тестовых последовательностей можно не получить за технически приемлемое время даже с применением самых современных вычислительных машин.

Вторая задача – синтез тестов для схем, критические состязания в которых устранены специальным подбором соотношений задержек между состязающимися сигналами в схемах. Это является характерным для схем, в которых используется *D*-, *JK*- и *T*- триггеры. В *D*- и *JK*-триггерах статический сбой устранен тем, что сигнал, распространяющийся по проводнику, обгоняет распространение этого же сигнала через логический элемент. Этот обгон и необходимо учитывать при устранении риска статического сбоя.

Работа *T*- триггера основана на явлении «защелки», когда во время переходного процесса происходит строго определенное последовательное изменение сигналов обратной связи.

Пусть некоторый сигнал $y_i = p_i$ вызывает изменение сигнала $y_i = x$ в определенное значение p_j, которое является устойчивым и не зависит от y_i, т. е. сигнал y_i после того, как $y_j = p_i$, может принимать безразличное значение ×. В этом и состоит механизм «защелки» значения $y_j = p_j$.

В свою очередь значение $y_j = p_j$ имплицирует новое значение $y_i = \bar{p}_i$. Все эти переключения происходят при одном и том же входном воздействии на схему.

Явление «защелки» приводит при синтезе набора $A = X_A Y_A Y'_A$ к тому, что $Y_A \subset Y'_A$, так как $y_i = p_i$ в Y_A и $y_i = \bar{p}_i$ в Y'_A. Учет этого явления приводит к расщеплению набора *A* на три составляющие A_1, A_2 A_3, которые, положив для простоты $Y = y_i y_j$, можно записать следующим образом:

	X	Y	Y'
A_1=	X_A	$p_i x_j$	$p_i p_j$;
A_2=	X_A	$x_i p_j$	$x_i p_j$;
A_3=	X_A	$x_i p_j$	$\bar{p}_i\, x_j$.

Здесь набор A_1 является неустойчивым, так как y_i изменяет свое значение с p_i на $\bar{p}_i$.

Третья задача состоит в обеспечении устойчивой, без явления риска сбоя, проверки неисправностей тестовыми последовательностями. Дело в том, что если некоторая неисправность *m* проверяется тестом с риском сбоя, то при проведении тестового эксперимента над физической схемой эта неисправность может быть, как зафиксированной, так и не зафиксированной. Следовательно, задача устранения состязаний должна решаться как для исправной схемы, так и для схем с неисправностями.

Синтез тестовых последовательностей, осуществлённый для условий одиночных константных неисправностей, не может гарантировать, в общем случае, контроль любого сочетания неисправностей и сложных неисправностей типа короткое замыкание. Распространение методов синтеза на эти неисправности и составляет четвёртую задачу.

И, наконец, при синтезе тестов предварительно необходимо все возможные неисправности разделить на три класса:

1) несущественных неисправностей, которые являются следствием избыточности в схеме и не могут быть проверены никакой допустимой входной последовательностью;
2) существенных неисправностей, которые через заданные внешние выходы схемы могут быть проверены только с риском сбоя;
3) существенных неисправностей.

Первый класс неисправностей при синтезе тестов должен быть просто исключен из рассмотрения. Для второго класса неисправностей в схеме необходимо либо предусмотреть дополнительные выходы, либо обеспечить доступ к некоторым внутренним точкам схемы с помощью специального щупа или других средств, чтобы обеспечить их контроль устойчиво, без явления риска сбоя. Поиск (синтез) тестов проводится только для существенных неисправностей.

4. Содержание домашней работы

1. Построение асинхронной модели Хаффмана.
2. Разбиение схемы на подсхемы.
3. Построение эквивалентных покрытий.
4. Переход от эквивалентных покрытий к общим.
5. Построение графа переходов схемы.
6. Активизация путей по эквивалентным покрытиям.
7. Синтез теста на схему:
   1) решить проверяющую задачу;
   2) решить установочную задачу;
   3) устранить состязания сигналов.
8. Анализ теста на полноту.

4.1. Пример синтеза теста

Рассмотрим все этапы домашнего задания на схеме, показанной на рис. 4.1.

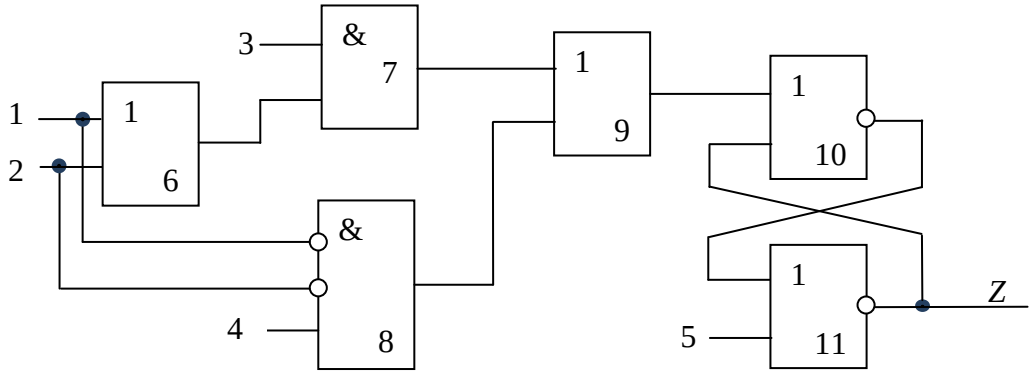


Рис. 4.1. Последовательностная схема

1. Построение асинхронной модели Хаффмана. В модели Хаффмана все сигналы обратной связи фиктивно разорваны. Введем в схему (рис. 4.1) фиктивную точку разрыва A таким образом, что $y\ (t + \Delta\tau) = y'(t)$, где $\Delta\tau$ – задержка сигнала. Построим асинхронную модель Хаффмана (рис. 4.2).

2. Разбиение схемы на подсхемы. Для упрощения работы с большими схемами проводим разбиение схемы на подсхемы. В данном примере удобно разбить схему так, как показано на рис. 4.2, выделив комбинационную и триггерную подсхемы.

3. Построение эквивалентных покрытий. Построение покрытий производим по $\boldsymbol{\pi}$-алгоритму.

В схеме восемь путей:

$l_1 = 3 – 7 – 9 – 10 – 11$, $l_2 = 1 – 6 – 7 – 9 – 10 – 11$, $l_3 = 2 – 6 – 7 – 9 – 10 – 11$, $l_4 = 4 – 8 – 9 – 10 – 11$, $l_5 = 1 – 8 – 9 – 10 – 11$, $l_6 = 2 – 8 – 9 – 10 – 11$, $l_7 = y – 10 – 11$ и $l_8 = 5 – 11$.

Формат эквивалентного покрытия $\Phi=(31_6 2_6 41_8 2_8 y5y')$.

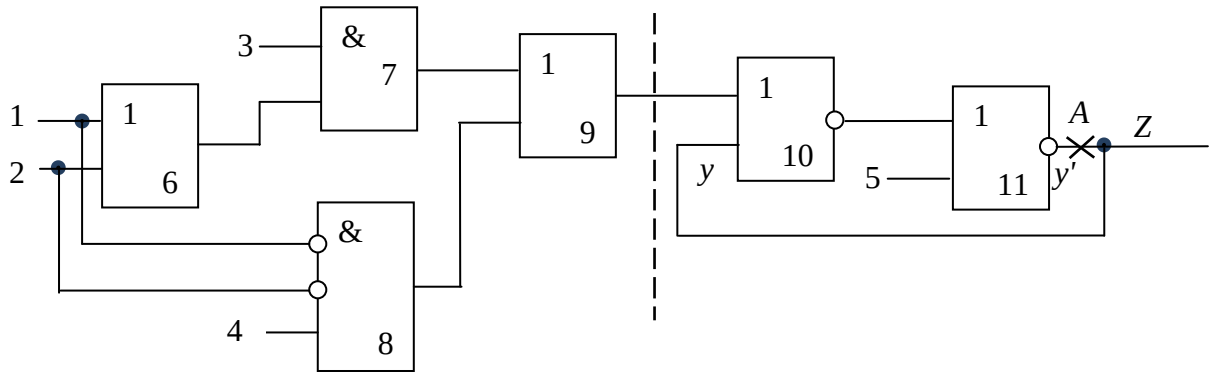


Рис. 4.2. Асинхронная модель Хаффмана

В основе алгоритма лежат оператор подстановки вырожденных покрытий элементов схемы и оператор пересечения кубов. Вырожденные покрытия составляются для входных наборов элементов в соответствии с их функциями.

Исходным шагом алгоритма является задание логических значений 0 или 1 на выходе схемы. По окончании алгоритма получаем нулевое $c(0)$ или единичное $c(1)$ покрытие входных наборов схемы, имплицирующих на ее выходе 0 или 1 соответственно.

Построение покрытия осуществляется итерационно движением по схеме от ее выхода до независимых входов путем подстановки вырожденных покрытий и пересечений кубов в ходе этих подстановок.

Пусть для схемы, показанной на рис. 4.1, необходимо построить нулевое покрытие $c_{11}(0)$. В соответствии с описанием элемента 11 записываем вырожденное покрытие для обеспечения значения 11 = 0 в формате его входов:

$$\begin{array}{ccc} 10 & 5 & 11 \\ \hline \left[\begin{array}{ccc} 1 & \times & 0 \\ \times & 1 & 0 \end{array}\right] & & \begin{array}{l} c_1 \\ c_2. \end{array} \end{array}$$

Далее, выбирая первый по порядку куб c_1, обеспечиваем подстановку значения 10 = 1 с помощью вырожденного покрытия элемента 10, в формате его входов:

$$\begin{array}{ccc} 9 & y & 10 \\ \hline \left[\begin{array}{ccc} 0 & 0 & 1 \end{array}\right] & & c_3 \end{array}$$

Объединение кубов c_1 и c_3 в единый формат дает куб:

$$\begin{array}{ccccc} 9 & y & 10 & 5 & 11 \\ \hline \left[\begin{array}{ccccc} 0 & 0 & 1 & \times & 0 \end{array}\right] & & & & c_4, \end{array}$$

в котором необходимо раскрыть значение 9 = 0, так как ***y*** является псевдовходом схемы, а координаты 10 и 11 уже раскрыты. Подстановка вырожденного покрытия для элемента 9ИЛИ: 7, 8 в кубе c_4 и переход в объединенный формат дает куб c_5:

$$\begin{array}{ccccccc} 7 & 8 & 9 & y & 10 & 5 & 11 \\ \hline \left[\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 1 & \times & 0 \end{array}\right] & & & & & & c_5. \end{array}$$

Подстановка покрытия для 7 = 0 элемента 7И:3, 6 дает кубы:

3	6	7	8	9	y	10	5	11	
0	×	0	0	0	0	1	×	0	c_6
×	0	0	0	0	0	1	×	0	c_7.

В кубы c_6 и c_7 необходимо подставить покрытие элемента 8И: 4, $\overline{1}$, $\overline{2}$, т.е.:

4	1	2	8
0	×	×	0
×	1	×	0
×	×	1	0

Пересечение с кубом c_6 дает кубы

4	1_8	2_8	3	6	7	8	9	y	10	5	11	
0	×	×	0	×	0	0	0	0	1	×	0	c_8
×	1	×	0	×	0	0	0	0	1	×	0	c_9,
×	×	1	0	×	0	0	0	0	1	×	0	c_{10}

в которых все зависимые координаты раскрыты, и пересечение с кубом c_7 дает кубы

4	1_8	2_8	3	6	7	8	9	y	10	5	11	
0	×	×	×	0	0	0	0	0	1	×	0	c_{11}
×	1	×	×	0	0	0	0	0	1	×	0	c_{12},
×	×	1	0	0	0	0	0	0	1	×	0	c_{13}

в которых необходимо раскрыть значение 6=0 элемента 6ИЛИ1, 2 путем подстановки вырожденного покрытия:

1	2	6
0	0	0

в кубы c_{11}, c_{12} и c_{13}, что дает соответственно кубы:

1_6	2_6	4	1_8	2_8	3	6	7	8	9	y	10	5	11	
0	0	0	×	×	×	0	0	0	0	0	1	×	0	c_{14}
0	0	×	1	×	×	0	0	0	0	0	1	×	0	c_{15}.
0	0	×	×	1	×	0	0	0	0	0	1	×	0	c_{16}

Кубы $\boldsymbol{c}_{15}$ и $\boldsymbol{c}_{16}$ дают пустое пересечение, так как в кубе $\boldsymbol{c}_{15}$ 1_6=0, а 1_8=1 и в кубе $\boldsymbol{c}_{16}$ 2_6=0, а 2_8=1. В следующем по порядку за кубом $\boldsymbol{c}_1$ кубе $\boldsymbol{c}_2$ все координаты раскрыты, так как 5 является входом системы. Итак, алгоритм закончен и покрытие $C^{э}_{11}(0)$ построено. Перепишем кубы в формат Ф и получим покрытие:

$C^{э}_{11}(0) =$

3	1_6	2_6	4	1_8	2_8	y	5	y'	
×	×	×	×	×	×	×	1	0	c_2
0	×	×	0	×	×	0	×	0	c_8
0	×	×	×	1	×	0	×	0	c_9 ,
0	×	×	×	×	1	0	×	0	c_{10}
×	0	0	0	×	×	0	×	0	c_{14}

где координаты кубов вырожденного покрытия соответствуют путям в схеме.

Аналогично построим и единичное эквивалентное покрытие. В нашей схеме 8 путей, следовательно, формат эквивалентного покрытия 8 координат, хотя независимых координат всего 6.

Предварительно построим эквивалентные покрытия для выделенных подсхем.

$C_9^э(1) =$

1_6	1_8	2_6	2_8	3	4
1	×	×	×	1	×
×	×	1	×	1	×
×	0	×	0	×	1

$C_9^э(0) =$

1_6	1_8	2_6	2_8	3	4
×	×	×	×	0	0
×	1	×	×	0	×
×	×	×	1	0	×
0	×	0	×	×	0
0	1	0	×	×	×
0	×	0	1	×	×

$C_{11}^э(1)=$

9	5	y
1	0	×
×	0	1

и

$C_{11}^э(0)=$

9	5	y
×	1	×
0	×	0

$C^э(y') = C_{11}^э(1) =$

1_6	1_8	2_6	2_8	3	4	5	y	y'	
1	×	×	×	1	×	0	×	1	t_1
×	×	1	×	1	×	0	×	1	t_2
×	0	×	0	×	1	0	×	1	t_3
×	×	×	×	×	0	0	1	1	t_4

Перейдем к эквивалентным покрытиям на всю схему. Для этого проведем объединение по координате 9, которая является выходом из первой подсхемы и входом во вторую. Совмещение координат производится так: в кубе (10×) из $C_{11}^э(1)$ по координате 9 стоит 1, следовательно, необходимо взять все кубы единичного покрытия $C_9^э(1)$ и подставить их в куб (10×). Куб (×01) не требует доопределения координат, так как по 9-й координате имеем безразличное значение ×. Аналогично проводится совмещение координат и для нулевого покрытия. После совмещения координат эквивалентные покрытия выглядят так:

$C^э(\bar{y}') = C_{11}^э(0)=$

1_6	1_8	2_6	2_8	3	4	5	y	y'
×	×	×	×	×	×	1	×	0
×	×	×	×	0	0	×	0	0
×	1	×	×	0	×	×	0	0
×	×	×	1	0	×	×	0	0
0	×	0	×	×	0	×	0	0
0	1	0	×	×	×	×	0	0
0	×	0	1	×	×	×	0	0

4. Переход от эквивалентных покрытий к обычным. Переход от эквивалентных покрытий к обычным осуществляется путем объединения

эквивалентных координат, пересекая их логические значения. В данном примере эквивалентные координаты 1_6 и 1_8, 2_6 и 2_8.

После объединения координат имеем обычные покрытия:

$C(y')$ =

1	2	3	4	5	y	y'
1	×	1	×	0	×	1
×	1	1	×	0	×	1
0	0	×	1	0	×	1
×	×	×	×	0	1	1

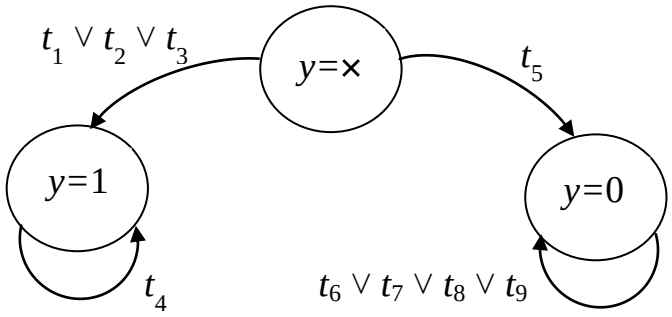
$C(\overline{y'})$=

1	2	3	4	5	y	y'
×	×	×	×	1	×	0
×	×	0	0	×	0	0
1	×	0	×	×	0	0
0	0	×	0	×	0	0
×	1	0	×	×	0	0

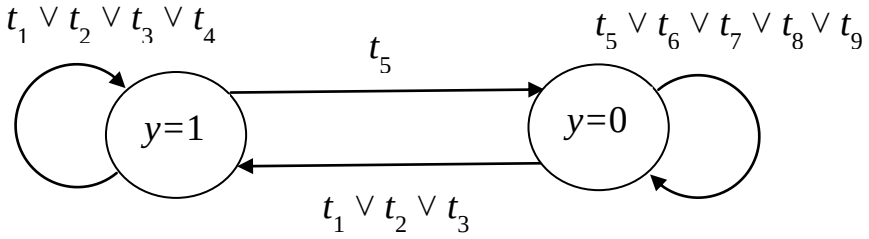
Для покрытия $C(\overline{y'})$ обозначим порядок следования сверху вниз: t_5, t_6, t_7, t_8, t_9.

Куб (010××××0) имеет пустое пересечение по координате 1, а куб (0×01×××0) – по координате 2, поэтому они исключаются из обычного покрытия.

5. Построение графа переходов схемы. Закон функционирования исходной схемы представим в виде графа переходов, где в узлах графа зафиксируем состояния схемы, а дуги укажут на переход из одного состояния схемы в другое под действием входного сигнала. Построим частично определенный граф переходов, в котором укажем установки в состояния 0 и 1 схемы из безразличного состояния × (рис. 4.3 а). Если доопределить × по координате y в покрытиях, то получим полностью определенный граф (рис. 4.3 б).



а)



б)

Рис. 4.3. а) Частично определённый и
б) полностью определенный графы переходов системы

5. **Активизация путей по эквивалентному покрытию.** Активизацию путей по эквивалентному покрытию рассмотрим в табл. 4.1.

Таблица 4.1.

Операция	$1_6\,1_8\,2_6\,2_8\,3\,4\,5\,y$	y	Примечания
∩	1 × × × 1 × 0 × × 1 × × 0 × × 0	1 0	l_1 = 3, 7 ,9, 10, 11
∩ y	1 1 × × y × 0 0	y	
1	1 1 × × 1' × 0 0 1 1 × × 0' × 0 0	$1'$ $0'$	
∩	1 × × × 1 × 0 × 0 × 0 × × 0 × 0	1 0	$l_2 = 1_6$, 6, 7 ,9, 10, 11
∩ y	y × 0 × 1 0 0 0	y	
2	1' × 0 × 1 0 0 0 0' × 0 × 1 0 0 0	$1'$ $0'$	
∩	× × 1 × 1 × 0 × 0 × 0 × × 0 × 0	1 0	$l_3 = 2_6$, 6, 7 ,9, 10, 11
∩ y	0 × y × 1 0 0 0	y	
3	0 × 1' × 1 0 0 0 0 × 0' × 1 0 0 0	$1'$ $0'$	
∩	× 0 × 0 × 1 0 × × 1 × × 0 × × 0	1 0	$l_4 = 1_8$, 8 ,9, 10, 11
∩ y	× y × 0 0 1 0 0	y	
4	× 0' × 0 0 1 0 0 × 1' × 0 0 1 0 0	$1'$ $0'$	
∩	× 0 × 0 × 1 0 × × × × 1 0 × × 0	1 0	$l_5 = 2_8$, 8 ,9, 10, 11
∩ y	× 0 × y 0 1 0 0	y	
5	× 0 × 1' 0 1 0 0 × 0 × 0' 0 1 0 0	$1'$ $0'$	
∩	× 0 × 0 × 1 0 × × × × × 0 0 × 0	1 0	l_6 = 4, 8 ,9, 10, 11
∩ y	× 0 × 0 0 y 0 0	y	
6	× 0 × 0 0 1' 0 0 × 0 × 0 0 0' 0 0	$1'$ $0'$	
∩	× × × × × × 0 1 × 1 × × 0 × × 0	1 0	l_7 =y, 10, 11
∩ y	× 1 × × 0 × × y	y	
7	× 1 × × 0 × × 1' × 1 × × 0 × × 0'	$1'$ $0'$	
∩	× × × × × × 1 × × × × × × × 0 1	0 1	l_8 = 5, 11
∩ y	× × × × × × y 1	y	
8	× × × × × × 1' 1 × × × × × × 0' 1	$0'$ $1'$	

В схеме восемь активизируемых путей. Для примера рассмотрим активизацию пути (3-7-9-10-11).

Из покрытия $C^э(y')$ выберем куб (1×××1×0×), тогда из $C^э(\overline{y'})$ выбираем такой куб, чтобы по координате 3 стоял 0 - (×1××0××0). Пересекая эти кубы, получим куб (11××y×00).

Произведем обратное пересечение $\overset{\frown}{y}$, в котором выполняется условие $y \overset{\frown}{y} 1 = 1'$ и $y \overset{\frown}{y} 0 = 0'$. Тогда с первым кубом обратное пересечение дает куб (11××1′×00), а со вторым - (11××0′×00). Таким образом, была проведена активизация пути, начинающегося с координаты 3. На этом пути контролируются все одиночные неисправности типа тождественный ноль и тождественная единица.

Аналогично проводится активизация всех остальных путей. При активизации значение по координате l_i не должно вступать в противоречие со значением 0 или 1 по другим координатам, эквивалентным данной.

6. **Синтез теста.** *Решение проверяющей задачи.* В результате активизации путей в схеме было построено по два набора на путь, которые контролируют все одиночные неисправности на данном пути. Построим проверяющую последовательность, упорядочив наборы для каждого пути следующим образом: первый – нейтральный, для него $y = y'$, а второй – переключающий, для него $y \neq y'$. Тогда проверяющая последовательность имеет вид:

P_1=

1_6	1_8	2_6	2_8	3	4	5	y	y'	
1	1	×	×	0′	×	0	0	0	t_1
1	1	×	×	1′	×	0	0	1	$\tilde{t}_1$
0′	×	0	×	1′	0	0	0	0	t_2
1′	×	0	×	1′	0	0	0	1	$\tilde{t}_2$
0	×	0	×	1′	0	0	0	0	t_3
0	×	1	×	1′	0	0	0	1	$\tilde{t}_3$
×	1′	×	0	0	1	0	0	0	t_4
×	0′	×	0	0	1	0	0	1	$\tilde{t}_4$

$\cup$

1_6	1_8	2_6	2_8	3	4	5	y	y'	
×	0	×	1	0	1	0	0	0	t_5
×	0	×	0	0	1	0	0	1	$\tilde{t}_5$
×	0	×	0	0	0	0	0	0	t_6
×	0	×	0	0	1	0	0	1	$\tilde{t}_6$
×	1	×	×	0	×	0	0	0	t_7
×	1	×	×	0	×	0	1	1	$\tilde{t}_7$
×	×	×	×	×	×	0	1	0	t_8
×	×	×	×	×	×	1	1	1	$\tilde{t}_8$

$\cup$

Решение установочной задачи. Все кубы из покрытий $C(y)$ и $C(\overline{y})$, в которых y = ×, являются установочными и решают задачу установки схемы либо в 0, либо в 1. Например, кубы (1×1×0×, ×11×0×, 00×10×) устанавливают схему в 1, а куб (××××1×) – в 0. Пусть в единичное состояние схему устанавливает набор c_1 = 1×1×0×, а в нулевое – c_2 = ××××1×.

Для составления тестовой последовательности запишем проверяющую последовательность в обычных координатах, объединив эквивалентные координаты 1_6, 1_8 и 2_6, 2_8:

$P_2 =$

1	2	3	4	5	y	y'	
1	×	0	×	0	0	0	t_1
1	×	1	×	0	0	1	$\tilde{t}_1$
0	0	1	0	0	0	0	t_2
1	0	1	0	0	0	1	$\tilde{t}_2$
0	0	1	0	0	0	0	t_3
0	1	1	0	0	0	1	$\tilde{t}_3$
1	0	0	1	0	0	0	t_4
0	0	0	1	0	0	1	$\tilde{t}_4$

$\cup$

1	2	3	4	5	y	y'	
0	1	0	1	0	0	0	t_5
0	0	0	1	0	0	1	$\tilde{t}_5$
0	0	0	0	0	0	0	t_6
0	0	0	1	0	0	1	$\tilde{t}_6$
1	×	0	×	0	0	0	t_7
1	×	0	×	0	1	1	$\tilde{t}_7$
×	×	×	×	0	1	1	t_8
×	×	×	×	1	1	0	$\tilde{t}_8$

Построим частично определенный граф переходов для синтеза теста данной схемы, пользуясь наборами из проверяющей последовательности и учитывая, что схема устанавливается в 0 под действием набора c_2 и в 1 – под воздействием c_1. Частично определенный граф приведен на рис. 4.4.

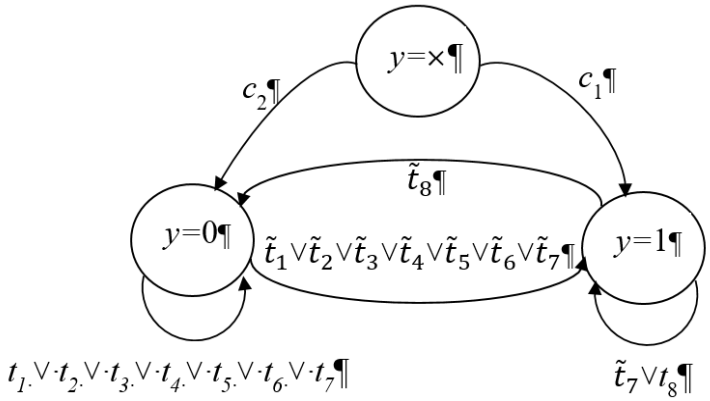


Рис. 4.4. Частично определенный граф для построения тестовой последовательности

Обходя граф по направлениям, указанным стрелками дуг, можно построить тестовую последовательность:

$T = \{c_1, \tilde{t}_7, \tilde{t}_8, t_1, t_2, t_3, t_4, t_5, t_6, t_7, \tilde{t}_1, \tilde{t}_8, \tilde{t}_2, \tilde{t}_8, \tilde{t}_3, \tilde{t}_8, \tilde{t}_4, \tilde{t}_8, \tilde{t}_5, \tilde{t}_8, \tilde{t}_6, t_8\}$.

В явном виде:

$T =$

1	2	3	4	5	y	y'	
1	×	1	×	0	×	1	c_1
1	×	0	×	0	1	1	$\tilde{t}_7$
×	×	×	×	1	1	0	$\tilde{t}_8$
1	×	0	×	0	0	0	t_1
0	0	1	0	0	0	0	t_2
0	0	1	0	0	0	0	t_3
1	0	0	1	0	0	0	t_4
0	1	0	1	0	0	0	t_5
0	0	0	0	0	0	0	t_6
1	×	0	×	0	0	0	t_7
1	×	1	×	0	0	1	$\tilde{t}_1$

$\cup$

1	2	3	4	5	y	y'	
×	×	×	×	1	1	0	$\tilde{t}_8$
1	0	1	0	0	0	1	$\tilde{t}_2$
×	×	×	×	1	1	0	$\tilde{t}_8$
0	1	1	0	0	0	1	$\tilde{t}_3$
×	×	×	×	1	1	0	$\tilde{t}_8$
0	0	0	1	0	0	1	$\tilde{t}_4$
×	×	×	×	1	1	0	$\tilde{t}_8$
0	0	0	1	0	0	1	$\tilde{t}_5$
×	×	×	×	1	1	0	$\tilde{t}_8$
0	0	0	1	0	0	1	$\tilde{t}_6$
×	×	×	×	0	1	1	t_8

Куб t_3=t_2, и они следуют друг за другом, поэтому можно удалить куб t_3 из теста. Пары наборов $\{\tilde{t}_8,\tilde{t}_4\}=\{\tilde{t}_8, \tilde{t}_5\}=\{\tilde{t}_8,\tilde{t}_6\}$, поэтому достаточно подать одну пару, чтобы проверить неисправности. Удалим пары $\{\tilde{t}_8, \tilde{t}_5\}$, $\{\tilde{t}_8,\tilde{t}_6\}$.

Устранение состязания сигналов. Проведем устранения состязаний сигналов в тестовой последовательности T. Исследования сведем в табл. 4.2 и проиллюстрируем так, как это показано на рис. 4.5.

Таблица 4.2.

	1 2 3 4 5 y y'	Примечания	Пояснения
c_1	1 × 1 × 0 × 1	A	Рис. 4.5, а
t_7	1 × 0 × 0 1 1	B	
∩	1 × 1 × 0 × 1	$c_1 \in C(y')$ и $c_1 = A$	
	× × × × 0 1 1	$c_2 \in C(y')$, $c_2 = B$	
D	1 × 1 × 0 1 1	$c_2 \cap c_1 \neq \varnothing$, $D = c_1 \cap c_2$	
$\tilde{t}_8$	× × × × 1 1 0	$c_1 \in C(\bar{y}')$, $c_1 \supset \tilde{t}_8$	Рис. 4.5, б
t_1	1 × 0 × 0 0 0	$c_2 \in C(\bar{y}')$, $c_2 \supset t_1$	
∩	× × × × 1 × 0	$D = c_1 \cap c_2$	
	1 × 0 × 1 0 0		
D	1 × 0 × 1 0 0		
t_1	1 × 0 × 0 0 0	$c_1 \in C(\bar{y}')$, $c_1 \supset t_1$	Рис. 4.5, в
t_2	0 0 1 0 0 0 0	$c_2 \in C(\bar{y}')$, $c_2 \supset t_2$	
∩	1 × 0 × × 0 0	$c_1 \cap c_2 = \varnothing$	
	0 0 × 0 × 0 0	$C_D \cap c_1 \neq \varnothing$, $C_D \cap c_2 \neq \varnothing$	
	y 0 0 0 × 0 0	$C_D \cap c_1 = D_1$	
C_D	× × 0 0 × 0 0	$C_D \cap c_2 = D_2$	
D_1	1 × 0 0 × 0 0		
D_2	0 0 0 0 × 0 0		
t_2	0 0 1 0 0 0 0	$c_1 \in C(\bar{y}')$, $c_1 \supset t_2$	Рис. 4.5, г
t_4	1 0 0 1 0 0 0	$c_2 \in C(\bar{y}')$, $c_2 \supset t_4$, $c_1 \cap c_2 = \varnothing$	
∩	0 0 × 0 × 0 0	$C_D \cap c_1 = \varnothing$, $c_D \cap c_2 = \varnothing$	
	1 × 0 × × 0 0	$D_1 = C_D \cap c_1$	
	× × 0 0 × 0 0	$D_2 = C_D \cap c_2$	
D_1	0 0 0 0 × 0 0		
D_2	1 × 0 0 × 0 0		
t_4	1 0 0 1 0 0 0	$c_1 \subset C(\bar{y}')$, $c_1 \supset t_1$	Рис. 4.5, д
t_5	0 1 0 1 0 0 0	$c_2 \supset t_5$	
∩	1 × 0 × × 0 0	$c_2 \cap c_1 = D$	
	× 1 0 × × 0 0		
D	1 1 0 × × 0 0		
t_5	0 1 0 1 0 0 0	$c_1 \in C(\bar{y}')$, $c_1 \supset t_5$	Рис. 4.5, е
t_6	0 0 0 0 0 0 0	$c_2 \in C(\bar{y}')$, $c_2 \supset t_6$	
∩	× 1 0 × × 0 0	$c_1 \cap c_2 = D$	
	× × 0 0 × 0 0		
D	× 1 0 0 × 0 0		
t_6	0 0 0 0 0 0 0	$c_1 \in C(\bar{y}')$, $c_1 \supset t_6$	Рис. 4.5, ж
t_7	1 × 0 × 0 0 0	$c_2 \supset t_7$	
∩	× × 0 0 × 0 0	$c_1 \cap c_2 = D$	
	1 × 0 × × 0 0		
D	1 × 0 0 × 0 0		
$\tilde{t}_4$	0 0 0 1 0 0 1	$c_2 \supset \tilde{t}_4$, $c_2 \supset t_8$,	Рис. 4.5, з
t_8	× × × × 0 1 1	$c_2 = t_8$	
∩	0 0 × 1 0 × 1	$D = c_1 \cap c_2$	
	× × × × 0 1 1		
D	0 0 × 1 0 1 1		

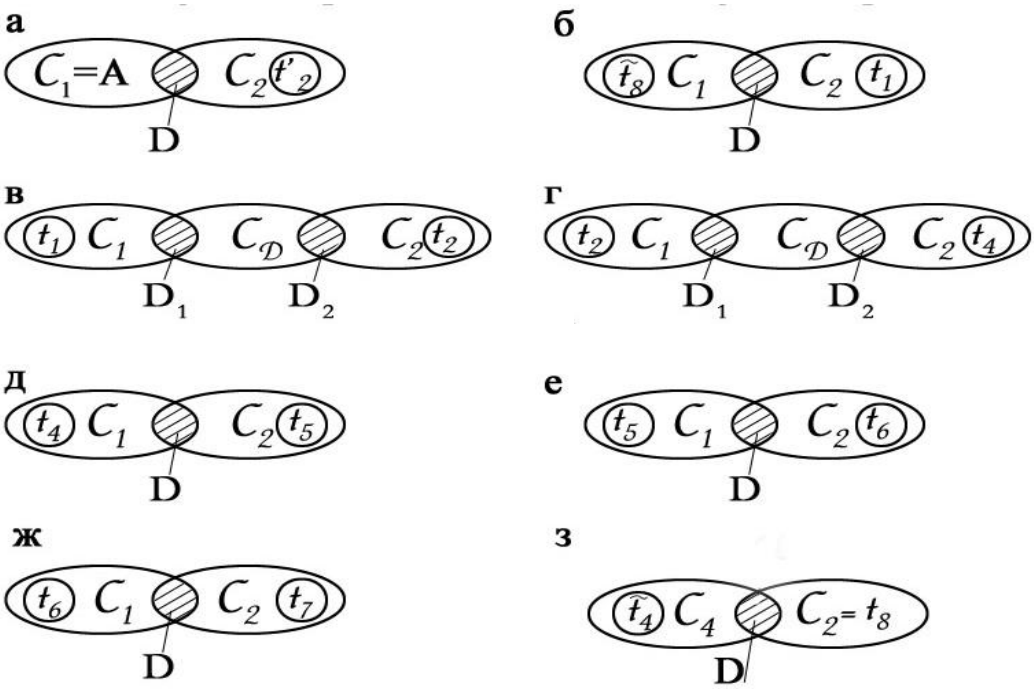


Рис. 4.5. Пояснения к таблице 4.2

Для примера рассмотрим устранение состязаний сигналов при переходе от набора $\tilde{t}_8$ к набору t_1. Выберем куб c_1 из покрытия $C(\bar{y}')$ такой, чтобы он включал в себя набор $\tilde{t}_8$, и куб c_2 из $C(\bar{y}')$, чтобы он включал набор t_1. Пересечем $c_1 \cap c_2$, пересечение непустое, следовательно, мы нашли куб $D=c_1 \cap c_2$, который необходимо поставить на переходе от $\tilde{t}_8$ к t_1 (рис. 4.5, б).

При переходе от набора t_7 к $\tilde{t}_1$ нет состязаний сигналов, так как они являются переключающими наборами (y_1' из состояния 0 переключается в состояние 1). Аналогично для $\tilde{t}_1$ - $\tilde{t}_8$, $\tilde{t}_8$ - $\tilde{t}_2$, $\tilde{t}_2$ - $\tilde{t}_8$, $\tilde{t}_8$ - $\tilde{t}_3$, $\tilde{t}_3$ - $\tilde{t}_8$, $\tilde{t}_8$ - $\tilde{t}_4$.

На переходе от t_2 к t_4 (рис. 4.5, г) оказалось, что $c_1 \cap c_2 = \varnothing$, и, следовательно, недостаточно одного набора D для устранения состязаний.

Получим серию переходных наборов. В покрытии найдем куб c_D такой, что его пересечения с c_1 и c_2 непустые. Тогда $c_D \cap c_1 = D_1$,

а $c_D \cap c_2 = D_2$.

Между наборами t_2 и t_4 необходимо поставить серию D_1, D_2, чтобы устранить состязания сигналов.

Аналогично устраняются состязания на всех переходах. После устранения состязаний сигналов получена тестовая последовательность:

$T =$

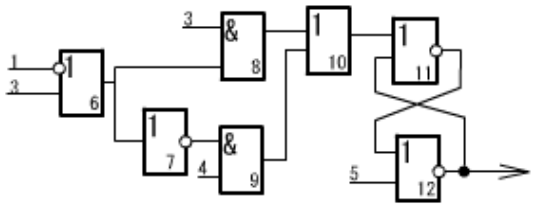
1	2	3	4	5	y	y'	
1	×	1	×	0	×	1	c_1
1	×	1	×	0	1	1	D
1	×	0	×	0	1	1	t_7
×	×	×	×	1	1	0	$\tilde{t}_8$
1	×	0	×	1	0	0	D
1	×	0	×	0	0	0	t_1
1	×	0	0	×	0	0	D_1
0	0	0	0	×	0	0	D_2
0	0	1	0	0	0	0	t_2
0	0	0	0	t_7	0	0	D_1
0	0	0	0	×	0	1	D_2

$\cup$

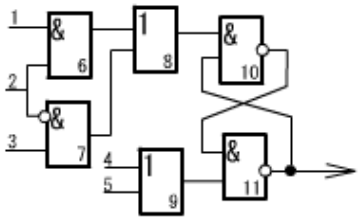
1	2	3	4	5	y	y'	
1	0	0	1	0	0	0	t_4
1	1	0	×	×	0	0	D
0	1	0	1	0	0	0	t_5
×	1	0	0	×	0	0	D
0	0	0	0	0	0	0	t_6
1	×	0	0	×	0	0	D
1	×	0	×	0	0	0	t_7
1	×	1	×	0	0	1	$\tilde{t}_1$
×	×	×	×	1	1	0	$\tilde{t}_8$
1	0	1	0	0	0	1	$\tilde{t}_2$
×	×	×	×	1	1	1	$\tilde{t}_8$
0	1	1	0	0	0	1	t_3
×	×	×	×	1	1	0	$\tilde{t}_8$
0	0	0	1	0	0	1	$\tilde{t}_4$
0	0	×	1	0	1	1	D
×	×	×	×	0	1	1	t_8

Тест T готов для контроля заданной схемы.

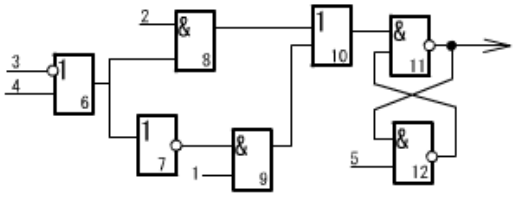
4.2. Варианты схем для домашнего задания



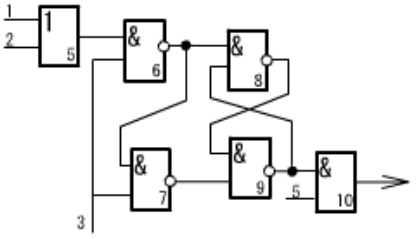
Вариант 1



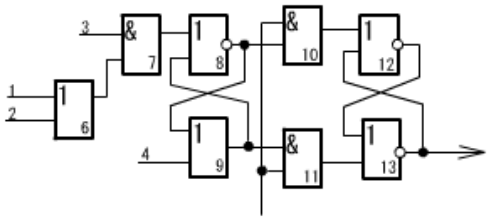
Вариант 2



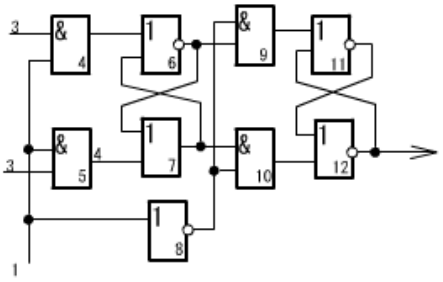
Вариант 3



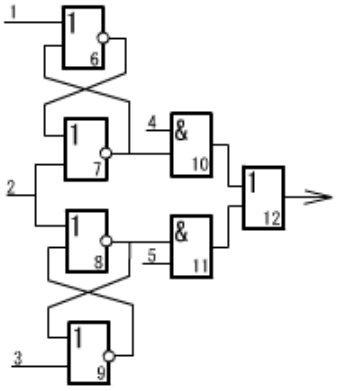
Вариант 4



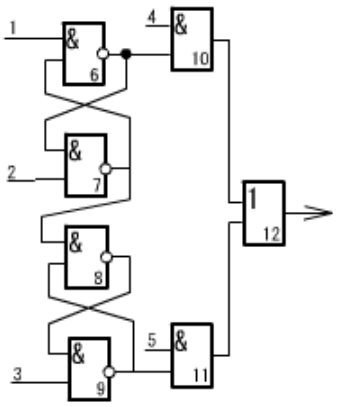
Вариант 5



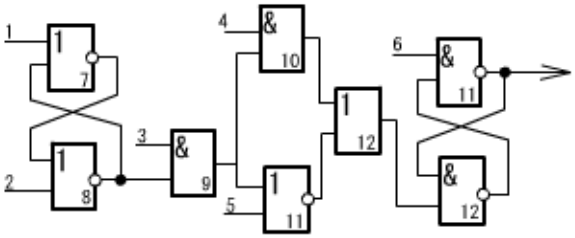
Вариант 6



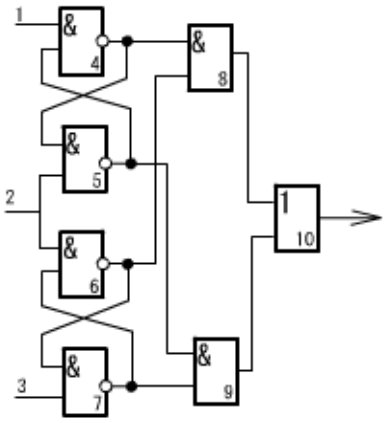
Вариант 7



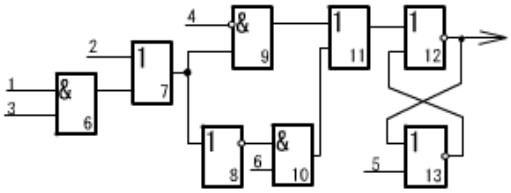
Вариант 8



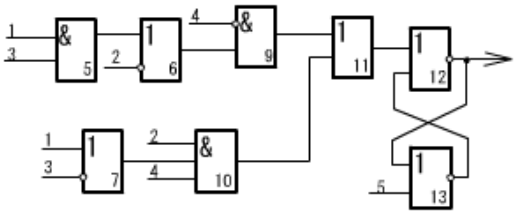
Вариант 9



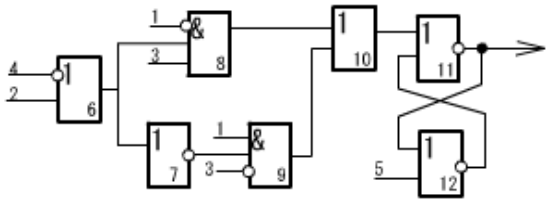
Вариант 10



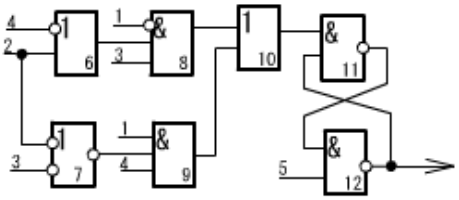
Вариант 11



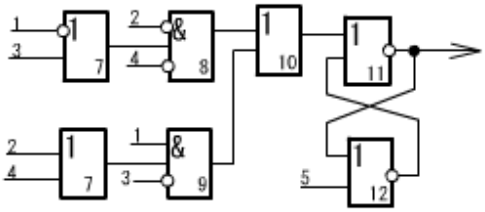
Вариант 12



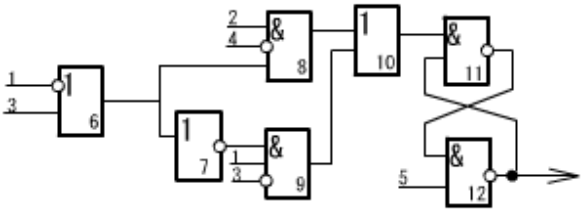
Вариант 13



Вариант 14



Вариант 15



Вариант 16

5. ВЕРИФИКАЦИИ ЦИФРОВЫХ СХЕМ

5.1. Итерационно-рекурсивная модель последовательностной схемы

Верификация предполагает наличие моделей устройства на принципиально различных уровнях представления и сравнение их между собой с целью установления их тождественного функционирования. В общем виде любое цифровое устройство может быть представлено в виде операционного устройства (ОУ) и устройства управления (УУ). Устройство управления, в отличие от ОУ, имеет, как правило, нерегулярную структуру, и задача построения его моделей является сложной проблемой и актуальна при проектировании микропроцессоров, заказных, полузаказных программируемых БИС. При построении моделей как для целей анализа, так и для целей синтеза обычно придерживаются иерархического принципа или подхода. Уровень представления модели определяется степенью её детализации. В данной работе рассматривается сопоставление между собой двух типов моделей УУ, а именно модели в виде структурно-временного автомата и его реализации в виде функционально-логической схемы. Выбор данного типа моделей обусловливается следующим. Модель в виде автомата является завершённым этапом функционального проектирования УУ. С другой стороны, функционально-логическая схема является началом технического проектирования устройства на любой элементной базе.

Методы построения моделей УУ в виде графа структурно-временного автомата достаточно исследованы и в работе не рассматриваются. В качестве модели функционально-логической схемы УУ в самом общем виде предлагается модель, представленная на рис. 5.1. Здесь вектор X есть независимые входы схемы, из которых особо выделены сигнал синхронизации T и сигнал начальной установки элементов памяти R, который может быть как асинхронным сбросом, так и синхронизированным тактом T, что в общем случае не является принципиальным. Значение сигналов обратной связи Y разбито на два подмножества Y^M и Y^S, соответственно $Y' = Y'^M \bigcup Y'^S$. Значения Y' определены как булевы функции $Y' = Y'(X,Y)$. Значения выходных сигналов также определены как $Z = Z(X,Y)$ для автомата Мили и как $Z = Z(Y)$ для автомата Мура. Для схемы также определены некоторые промежуточные сигналы Z_k, как выходы комбинационных подсхем (КС), которые являются вспомогательными и введены как следствие разбиения КС на одновыходные комбинационные подсхемы. Значение вектора сигналов Y определяет предыдущее значение состояния схемы, а значение вектора Y' - последующее состояние. В общем случае векторы Y и Y' задают некоторые классы состояний. Любой сигнал y и y' может принимать три значения: 0, 1 и ×, где × есть безразличное состояние, которое произвольным образом можно доопределить в значение 1 или 0, если это не вызывает противоречия при переходе схемы из состояния Y в состояние Y'.

Соответствие между значениями сигналов Y и Y' определяется по интервальным покрытиям $C(y') = C^x(y') \cup C^p(y')$, где $C^x(y')$ есть область установки и $C^p(y')$ есть область хранения. Если некоторое значение y' вычислено кубом $c \in C^x(y')$, то $y{=}\times$ и $y'{=}p$, если же значение y' вычислено кубом $C \in C^p(y')$, то значение $y = y' = p$.

В соответствии с предложенной структурной моделью схемы (рис. 5.1) введём понятие куба единого формата $\Phi_0 = XY^M Y^S Z_K Y'^M Y'^S Z$.

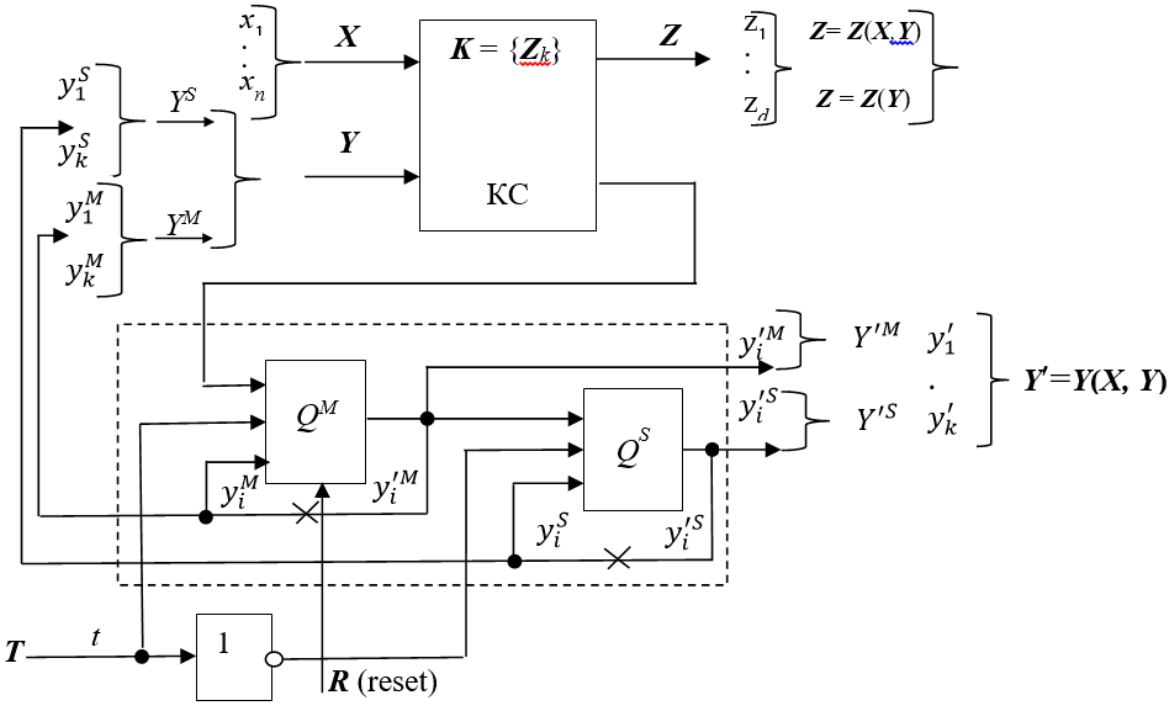


Рис. 5.1 Универсальная модель функционально-логической схемы

Данный формат в векторном виде описывает все значения сигналов и их разбиение на функциональные классы. Если не делать различия между классами Y^M, Y^S, Y'^M и Y'^S, то можно ограничиться форматом $\Phi_1 = XYZ_K Y'Z$. Далее, если Z_K отсутствуют, можно ограничиться форматом $\Phi_2 = XYY'Z$, и, наконец, для случая, когда рассматриваются только переходы схемы, можно перейти к формату $\Phi_3 = XYY'$, что является вполне допустимым при теоретическом рассмотрении состояний и переходов схемы, так как значения Z_K и Z являются импликациями от независимых входов X и сигналов обратной связи Y или Y' и их можно вычислить после построения переходов.

Определим и обозначим переход схемы в формате XYY' как XY/Y', где схема из состояния Y под действием входного вектора X переходит в состояние Y', что и показано на рис. 5.2. Установим, что вектор $Y \supseteq Y'$. Это означает доопределение некоторых сигналов обратной связи y' из $\times$ в p.

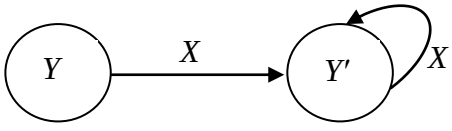


Рис. 5.2

Кроме того, заметим, что в схеме без неисправностей, а именно такие схемы и рассматриваются при верификации проектных решений, переходной процесс в ней заканчивается, что и показано на рис. 5.2 в виде петли $XY'=Y'$. Из этого следует, что из рассмотрения исключаются схемы без начальной установки, с наличием генерации сигналов и, как следствие, схемы с неоднозначным состоянием, а также схемы, в которых возможен недетерминированный переход из состояния Y под действием вектора X в некоторое состояние из множества возможных состояний, если в них при проектировании не были устранены критические состязания сигналов. Другими словами, будем рассматривать основной класс асинхронных схем, если синхросигнал T приравнять к обычным входам схемы, в виде так называемой основной модели Хаффмана (Huffman D. A. The synthesis of sequential switching circuits // J. Franklin Inst. 1954. Vol. 257, № 3, 4.).

В соответствии с предложенной моделью на ней можно поставить решение трёх основных задач:

- задачи моделирования;
- задачи построения переходов схемы в прямом направлении;
- задачи синтеза переходов в обратном направлении.

Заметим также, что при синтезе переходов для случая двухступенчатой модели элементов памяти значения сигналов из Y можно перевести в область хранения, то есть вычислять их из области $C^p(y')$. Это допущение не является принципиальным, но позволит в дальнейшем значительно сократить перебор при поиске решений. Отметим, что задача синтеза переходов схемы носит принципиально переборный характер, и в общем случае можно говорить только о направленных методах перебора и способах его сокращения.

При решении задачи моделирования исходным является вектор XY, а искомым - Y'. Значения сигналов из Y' вычисляются по методу импликации на основе операции пересечения кубов, с учётом того, что $a \bigcap b = b \bigcap a$ и, если $a \bigcap b \subseteq a$, $b \subseteq a$, и наоборот, если $a \bigcap b \subseteq b$, то $a \subseteq b$. Указанное соотношение включения необходимо отслеживать при моделировании для исследования функционального и логического риска сбоев в подсхемах и во всей схеме с целью выявления и анализа состязания сигналов. Моделирование носит итерационный характер и заканчивается на некотором (i+1) - шаге, если $Y_i' = Y_{i+1}'$. Для этого и было введено ограничение на класс рассматриваемых схем. При моделировании никаких доопределений в вектор X не вносится, и в ходе итераций происходит последовательная замена вектора Y на вектор Y'.

При решении задачи построения переходов в прямом направлении исходным вектором является вектор Y, значения координат которого являются априорно заданными и в нём запрещены любые доопределения. Векторы X и Y' первоначально задаются в виде безразличных значений x по всем координатам. Вычисления проводятся по всем координатам y' из Y' с целью их доопределения в значения p= 0 или 1, путём перебора и фиксации кубов в покрытиях $C(y') = C^x(y') \bigcup C^p(y')$. В результате такого перебора и

фиксации кубов в покрытиях $C(y')$, которые и определяют значения координат в кубе единого формата $\Phi = XYY'$, и происходит вычисление векторов X и Y'. В общем случае образуется некоторое множество переходов из состояния Y в Y', то есть XYY', что и показано на рис. 5.3.

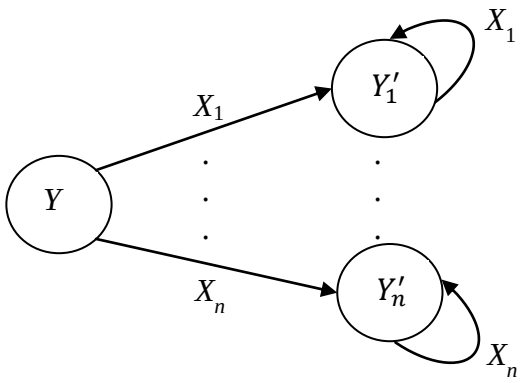


Рис.5.3 Синтез переходов схемы в прямом направлении

Указанные переходы также строятся на основе операции пересечения по формуле:

$Y'=\cap C_i(y')$, при $Y = Y_{\text{фиксированное}}$.

При синтезе переходов в обратном направлении заданным является вектор Y', который может задаваться из условий активизации сигналов в схемах, либо из условия некоторого искомого состояния, например, при решении установочной задачи. Исходные значения X и Y являются неопределёнными по всем своим координатам. Значения в векторе Y' могут доопределяться, лишь бы выполнялось соотношение включения:

$Y'_{зад.} \supseteq Y'_{выч.}$

Синтез переходов и в этом случае проводится с использованием операции пересечения кубов по следующей формуле:

$Y'=\cap C_i(y')$, для $y_i' = p$,

при условии непротиворечивости пересечения и возможных доопределений y' из × в p, что в значительной мере усложняет метод. Задача построения множества $X_iY_i \Rightarrow Y'$ носит также переборный характер и в общем случае образует некоторое множество переходов схемы, что и показано на рис. 5.4.

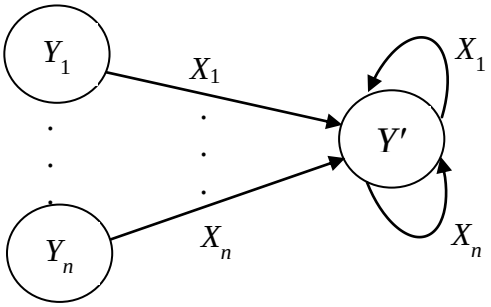


Рис. 5.4. Синтез переходов схемы в обратном направлении

Приведём различие между верификацией и тестированием устройства. При верификации сравниваются две модели (М1 и М2) различного уровня исправного устройства. При тестировании необходимо сравнивать все

возможные версии устройства, которые порождаются из-за возможности появления в нём тех или иных множеств неисправностей, с априорно заданной исправной версией. Таким образом, тестирование является более сложной и трудоёмкой задачей, и его целесообразно проводить после решения задачи верификации проекта. Задачу верификации можно решить методами математического моделирования. В этом случае необходимо построить испытательные последовательности ИП1 и ИП2 для моделей М1 и М2, соответственно. Затем произвести перекрёстное моделирование ИП2 на модели М1 с получением результата R(ИП2, М1) и моделирование ИП1 на модели М2 с получением результата R(ИП1, М2). Сравнение результатов моделирования R(ИП1, М1) с R(ИП1, М2) и R(ИП2, М2) с R(ИП2, М1), которые должны быть сопоставимы, позволяет сделать вывод о тождественности функционирования устройства, представленного моделями М1 и М2.

Отметим основные достоинства и недостатки данного подхода. Моделирование как средство анализа является сравнительно простой процедурой, однако построение испытательных последовательностей является очень сложной задачей, если рассматривать их с точки зрения полноты решаемой задачи верификации. Возможен и другой подход к решению задачи верификации. В нашем случае можно построить полный граф переходов функционально-логической схемы и сопоставить его с графом переходов структурно-временного автомата. В этом случае испытательные последовательности не строятся, в них просто нет необходимости, а идентичность моделей устанавливается путём сравнения двух графов между собой. Поэтому в решение задачи верификации входит исследование и разработка методов и алгоритмов построения графа переходов функционально-логической схемы. При построении графа переходов схемы в качестве основного метода остановимся на алгебро-топологическом подходе, при котором схема описывается в виде кубических покрытий, как наиболее машинно-ориентированном.

При построении графа переходов схемы возможны два подхода: поиск переходов схемы в прямом или обратном направлении. Любой переход схемы может быть представлен в виде вектора XYY', компонент Z при теоретическом рассмотрении может быть опущен, так как он в общем случае является импликацией от XY', т.е. $Z \Leftarrow (X,Y')$. Здесь значение вектора Y является исходным состоянием схемы и Y' - последующим состоянием, в которое схема переходит под действием входных сигналов X в исходном состоянии схемы. Так как УУ является сильно связанным устройством, можно ограничиться построением графа переходов в прямом направлении. В этом случае необходимо отдельно рассмотреть вопрос о поиске начальной вершины графа переходов схемы, а именно, переход из безразличного состояния сигналов обратной связи $Y = (\times\times...\times)$ в некоторое множество исходных состояний Y'. Поиск таких переходов можно осуществить путем пересечения покрытий $C(y')$ между собой при условии наложения

ограничений на значение сигналов обратной связи *Y*. При поиске начальных вершин:

$$Y_0' = \bigcap_i C_i(y'), \text{ при } Y = \times\times\ldots\times,$$

и при поиске последующих вершин графа переходов:

$$Y_{j+1}' = \bigcap_i C_i(y'), \text{ при } Y = Y_j'.$$

Тождество построенного графа переходов функционально-логической схемы УУ графу структурно-временного автомата позволяет сделать заключение, что рассматриваемые модели верифицированы.

5.2. Верификации моделей разного уровня методом построения графа переходов схемы

При синтезе устройств управления одним из основных этапов является разработка графов абстрактного и структурного автоматов, на основе которых проектируется функционально-логическая схема.

В качестве примера рассмотрим граф абстрактного автомата, представленный на рис. 5.5. Данный автомат выполняет функцию пересчёта.

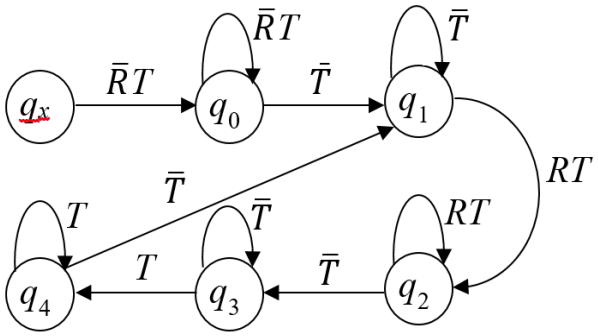


Рис. 5.5. Граф абстрактного автомата функции пересчёта

Соответствующий граф структурного автомата представлен на рис. 5.6.

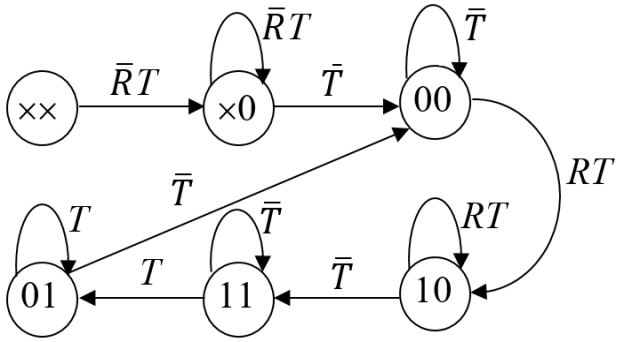


Рис. 5.6. Граф структурного автомата функции пересчета

Пусть задана реализация структурного автомата в виде функционально-логической схемы, изображённой на рис. 5.7. Переход от графа автомата к схеме является задачей синтеза схем и не входит в круг рассмотрения.

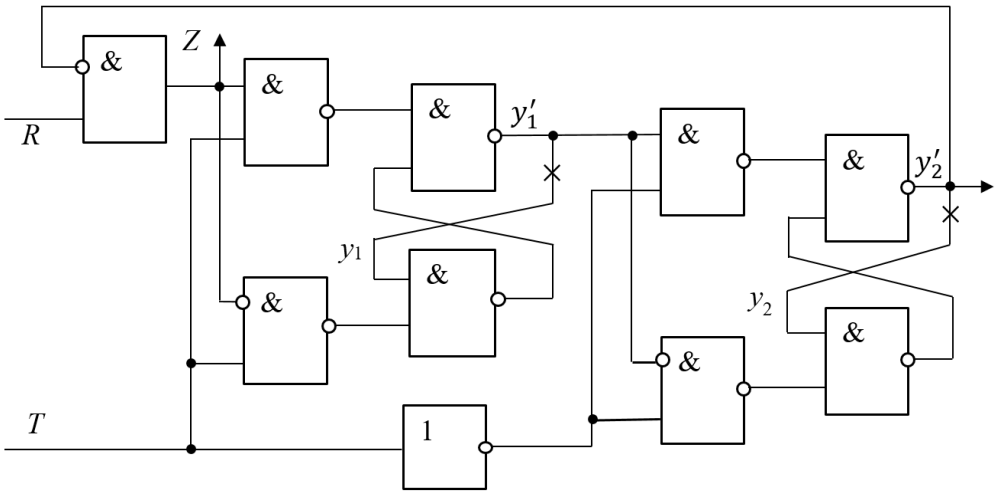


Рис. 5.7. Функционально-логическая схема

Запишем покрытия функционально-логической схемы, представленной на рис. 3.3.

$C(Z)=$

R	y_2'	Z
1	0	1
0	×	0
×	1	0

$C^1(y_1')=$

T	Z	y_1	y_1'	
1	1	×	1	C^x
×	1	1	1	C^p
0	×	1	1	C^p

$C^0(y_1')=$

T	Z	y_1	y_1'	
1	0	×	0	C^x
×	0	0	0	C^p
0	×	0	0	C^p

$C(y_1')=$

	R	T	y_2'	y_1	y_1'	
c_1	1	1	0	×	1	C^x
c_2	1	×	0	1	1	C^p
c_3	×	0	×	1	1	C^p
c_4	0	1	×	×	0	C^x
c_5	×	1	1	×	0	C^x
c_6	0	×	×	0	0	C^p
c_7	×	×	1	0	0	C^p
c_8	×	0	×	0	0	C^p

$C(y_2')=$

	T	y_1'	y_2	y_2'	
c_9	0	1	×	1	C^x
c_{10}	×	1	1	1	C^p
c_{11}	1	×	1	1	C^p
c_{12}	0	0	×	0	C^x
c_{13}	×	0	0	0	C^p
c_{14}	1	×	0	0	C^p

Для упорядочения процедуры пересечения кубов проведём их индексацию по порядку вхождения в покрытия сквозным способом. Функция Z включена в покрытия $C^1(y_1')$ и $C^0(y_1')$. Это сделано для упрощения иллюстрации методов, чтобы работать не в формате $XZYY'$, а в более простом формате XYY', что не является принципиальным.

В покрытиях явным образом выделены области C^x - установки и C^p - хранения локальных сигналов обратной связи. В процессе перебора на эти понятия опираться не будем, так как эти понятия в явном виде существуют в кубах, и они могут привлекаться лишь для направленности и сокращения перебора, а на методы и алгоритмы не должны влиять.

Первоначально рассмотрим метод пересечения каждого куба с каждым и выполним его формально, как вычислительная машина. Результаты вычислений сведены в таблицу 5.1, в которой пустые пересечения отмечены в примечании.

Таблица 5.1

∩	R	T	y_1	y_2	y_1'	y_2'	Кубы c_i^l	Примечание
c_1	1	1	×		1	0		$c_1 \cap c_{12} = \varnothing$
c_{14}	×	1		0	×	0		$c_1 \cap c_{13} = \varnothing$
$c_1 \cap c_{14}$	1	1	×	0	1	0	c_1^l	
c_2	1	×	1		1	0		$c_2 \cap c_{12} = \varnothing$
c_{14}	×	1		0	×	0		$c_2 \cap c_{13} = \varnothing$
$c_2 \cap c_{14}$	1	1	1	0	1	0	c_2^l	$c_1^l \supset c_2^l$
c_3	×	0	1		1	×		$c_3 \cap c_{11} = \varnothing$
c_9		0		×	1	1		$c_3 \cap c_{14} = \varnothing$
$c_3 \cap c_9$	×	0	1	×	1	1	c_3^l	
c_3	×	0	1		1	×		$c_3 \cap c_{12} = \varnothing$
c_{10}		×		1	1	1		$c_3 \cap c_{13} = \varnothing$
$c_3 \cap c_{10}$	×	0	1	1	1	1	c_4^l	$c_3^l \supset c_4^l$
c_4	0	1	×		0	×		$c_4 \cap c_9 = \varnothing$
c_{11}		1		1	×	1		$c_4 \cap c_{10} = \varnothing$
$c_4 \cap c_{11}$	0	1	×	1	0	1	c_5^l	
c_4	0	1	×		0	×		$c_4 \cap c_{12} = \varnothing$
c_{13}		×		0	0	0		
$c_4 \cap c_{13}$	0	1	×	0	0	0	c_6^l	
c_4	0	1	×		0	×		
c_{14}		1		0	×	0		
$c_4 \cap c_{14}$	0	1	×	0	0	0	c_7^l	$c_7^l = c_6^l$
c_5	×	1	×		0	1		$c_5 \cap c_9, c_{10}, c_{11} = \varnothing$
c_{11}		1		1	×	1		$c_5 \cap c_{13}, c_{14} = \varnothing$
$c_5 \cap c_{11}$	×	1	×	1	0	1	c_8^l	$c_8^l \supset c_5^l, c_9^l, c_{13}^l$
c_6	0	×	0		0	×		$c_6 \cap c_9 = \varnothing$
c_{11}		1		1	×	1		$c_6 \cap c_{10} = \varnothing$
$c_6 \cap c_{11}$	0	1	0	1	0	1	c_9^l	
c_6	0	×	0		0	×		
c_{12}		0		×	0	0		
$c_6 \cap c_{12}$	0	0	0	×	0	0	c_{10}^l	
c_6	0	×	0		0	×		
c_{13}		×		0	0	0		
$c_6 \cap c_{13}$	0	×	0	0	0	0	c_{11}^l	
c_6	0	×	0		0	×		
c_{14}		1		0	×	0		
$c_6 \cap c_{14}$	0	1	0	0	0	0	c_{12}^l	$c_{11}^l \supset c_{12}^l$
c_7	×	×	0		0	1		$c_7 \cap c_9, c_{10} = \varnothing$
c_{11}		1		1	×	1		$c_7 \cap c_{12}, c_{13}, c_{14} = \varnothing$
$c_7 \cap c_{11}$	×	1	0	1	0	1	c_{13}^l	
c_8	×	0	0		0	×		$c_8 \cap c_9, c_{10} = \varnothing$
c_{12}		0		×	0	0		$c_8 \cap c_{11}, c_{14} = \varnothing$
$c_8 \cap c_{12}$	×	0	0	×	0	0	c_{14}^l	
c_8	×	0	0		0	×		
c_{13}		×		0	0	0		
$c_8 \cap c_{13}$	×	0	0	0	0	0	c_{15}^l	$c_{14}^l \supset c_{15}^l, c_{10}^l$

Слева указаны движки и кубы из покрытий $C(y_1')$ и $C(y_2')$, вступающие в пересечение, в центре приведены результаты пересечений в едином формате, и справа показана индексация кубов единого формата в виде c_i^l, при i = 1, 14. В примечании также отмечены совпадающие кубы и включаемые в кубы высшего ранга.

Сведём полученные результаты в отдельную таблицу 5.2 со сквозной индексацией кубов c_i^l и информацией в примечании о поглощаемых и совпадающих кубах.

Таблица 5.2

∩	R	T	y_1	y_2	y_1'	y_2'	Кубы c_i^l	Примечание
$c_1 \cap c_{14}$	1	1	×	0	1	0	c_1^l	
$c_2 \cap c_{14}$	1	1	1	0	1	0	c_2^l	$c_1^l \supset c_2^l$
$c_3 \cap c_9$	×	0	1	×	1	1	c_3^l	
$c_3 \cap c_{10}$	×	0	1	1	1	1	c_4^l	$c_3^l \supset c_4^l$
$c_4 \cap c_{11}$	0	1	×	1	0	1	c_5^l	
$c_4 \cap c_{13}$	0	1	×	0	0	0	c_6^l	
$c_4 \cap c_{14}$	0	1	×	0	0	0	c_7^l	$c_7^l = c_6^l$
$c_5 \cap c_{11}$	×	1	×	1	0	1	c_8^l	$c_8^l \supset c_5^l$, c_9^l, c_{13}^l
$c_6 \cap c_{11}$	0	1	0	1	0	1	c_9^l	
$c_6 \cap c_{12}$	0	0	0	×	0	0	c_{10}^l	
$c_6 \cap c_{13}$	0	×	0	0	0	0	c_{11}^l	
$c_6 \cap c_{14}$	0	1	0	0	0	0	c_{12}^l	$c_{11}^l \supset c_{12}^l$
$c_7 \cap c_{11}$	×	1	0	1	0	1	c_{13}^l	
$c_8 \cap c_{12}$	×	0	0	×	0	0	c_{14}^l	
$c_8 \cap c_{13}$	×	0	0	0	0	0	c_{15}^l	$c_{14}^l \supset c_{15}^l, c_{10}^l$

Исключим поглощаемые кубы и сведём полученные результаты в таблицу 5.3.

Таблица 5.3

∩	R	T	y_1	y_2	y_1'	y_2'	Кубы c_i^l	Примечание
$c_1 \cap c_{14}$	1	1	×	0	1	0	c_1^l	$c_1^l \supset c_2^l$
$c_3 \cap c_9$	×	0	1	×	1	1	c_3^l	$c_3^l \supset c_4^l$
$c_4 \cap c_{14}$	0	1	×	0	0	0	c_6^l	$c_6^l = c_7^l$
$c_5 \cap c_{11}$	×	1	×	1	0	1	c_8^l	$c_8^l \supset c_5^l$, c_9^l, c_{13}^l
$c_6 \cap c_{13}$	0	×	0	0	0	0	c_{11}^l	$c_{11}^l \supset c_{12}^l$
$c_8 \cap c_{12}$	×	0	0	×	0	0	c_{14}^l	$c_{14}^l \supset c_{15}^l, c_{10}^l$

Фрагменты графа переходов, построенных по полученным кубам приведены на рис. 5.4.

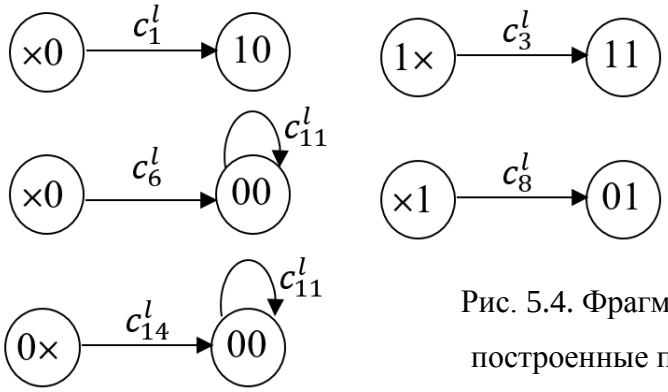


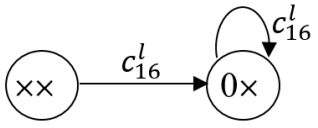
Рис. 5.4. Фрагменты графа переходов, построенные по вычисленным кубам

На первый взгляд, состояния схемы мало стыкуемы и нет состояния *Y* = ××, которое является исходным для начальной установки схемы для любого графа переходов ФЛС. Однако, если внимательно рассмотреть состояния схемы в кубах c_i^l, то можно некоторые из них склеить по принципу, аналогичному * - произведению кубов. Такую операцию обозначим как $\bigcap_*$ и её можно достаточно корректно применить для ситуации $y_i = y_i'$, так как в этом случае можно считать, что контур обратной связи замкнут и y_i, y_i' есть один и тот же сигнал. Из множества $\{c_i^l\}$ выберем два куба c_5^l и c_6^l, которые вступают в такую операцию и дают результат: $c_5^l \bigcap_* c_6^l = c_{16}^l$ = 01××0×, что и показано на рис. 35.5 а). Фрагмент графа переходов куба c_{16}^l приведён на рис. 5.5 б). Объединение графа переходов (рис. 5.4 и 5.5 б) показано на рис. 5.6.

Теперь можно провести сравнения исходного графа (рис. 5.2) и построенного (рис. 5.6). Они являются изоморфными. Если же на графе переходов выделить в явном виде вектор (*RT*) и обозначить им дуги переходов, то можно сделать заключение об их идентичности, что даёт верификацию ФЛС (рис. 5.3 а).

Кубы	*R*	*T*	y_1	y_2	y_1'	y_2'	Операция
c_5^l	0	1	×	1	0	1	∩
c_6^l	0	1	×	0	0	0	*
$c_5^l \bigcap_* c_6^l$	0	1	×	*y*	0	*y*	∩
	×	×	×	×	×	×	*
c_{16}^l	0	1	×	×	0	×	

а)



б)

Рис. 5.5. Операция $\bigcap_*$ а) и фрагмент графа переходов б)

Теперь рассмотрим построение графа переходов ФЛС на основе ограничений, налагаемых на вектор *Y*. Вычисления проведём по тем же

самым покрытиям $C(y_1')$ и $C(y_2')$ и с продолжением той же сквозной индексации получаемых кубов c_i^l, начиная с $i = 17$. Результаты вычислений сведены в таблицу 5.4.

Таблица 5.4

$\bigcap$ и огранич.	R	T	y_1	y_2	y_1'	y_2'	Кубы c_i^l	Примечание
$Y = \times\times$			×	×				$Y = \times\times \rightarrow Y' = 0\times$
c_4	0	1	×	×	0	×	c_{17}^l	$c_{17}^l = c_{16}^l$
$Y = 0$x			0	×				
$c_8 \cap c_{12}$	×	0	0	×	0	0	c_{18}^l	$c_{18}^l = c_{1414}^{\ l}$
$Y = 00$			0	0				
$c_1 \cap c_{14}$	1	1	×	0	1	0	c_{19}^l	$c_{19}^l = c_1^l$
$c_8 \cap c_{13}$	×	0	0	0	0	0	c_{20}^l	$c_{20}^l = c_{15}^l \subset c_{1818}^{\ l}$
$c_4 \cap c_{13}$	0	1	×	0	0	0	c_{21}^l	$c_{21}^l = c_6^l$
$c_4 \cap c_{14}$	0	1	×	0	0	0	c_{22}^l	$c_{22}^l = c_{2121}^{\ l}$
$c_6 \cap c_{12}$	0	0	0	×	0	0	c_{23}^l	$c_{23}^l = c_{10}^l$
$c_6 \cap c_{13}$	0	×	0	0	0	0	c_{24}^l	$c_{24}^l = c_{11}^l$
$c_6 \cap c_{14}$	0	1	0	0	0	0	c_{25}^l	$c_{25}^l = c_{12}^l \subset c_{24}^l$
$c_8 \cap c_{12}$	×	0	0	×	0	0	c_{26}^l	$c_{26}^l = c_{18}^l = c_{14}^l$
$Y = 10$			1	0				
$c_1 \cap c_{14}$	1	1	×	0	1	0	c_{27}^l	$c_{27}^l = c_1^l$
$c_2 \cap c_{14}$	1	1	1	0	1	0	c_{28}^l	$c_{28}^l = c_2^l \subset c_{27}^l$
$c_3 \cap c_9$	×	0	1	×	1	1	c_{29}^l	$c_{29}^l = c_3^l$
$c_4 \cap c_{13}$	0	1	×	0	0	0	c_{30}^l	$c_{30}^l = c_6^l$
$c_4 \cap c_{14}$	0	1	×	0	0	0	c_{31}^l	$c_{31}^l = c_{30}^l$
$Y = 11$			1	1				
$c_3 \cap c_9$	×	0	1	×	1	1	c_{32}^l	$c_{32}^l = c_3^l = c_{24}^l$
$c_3 \cap c_{10}$	×	0	1	1	1	1	c_{33}^l	$c_{33}^l = c_4^l \subset c_{28}^l$
$c_4 \cap c_{11}$	0	1	×	1	0	1	c_{34}^l	$c_{34}^l = c_5^l$
$c_5 \cap c_{11}$	×	1	×	1	0	1	c_{35}^l	$c_{35}^l = c_8^l \supset c_{34}^l$
$Y = 01$			0	1				
$c_4 \cap c_{11}$	0	1	×	1	0	1	c_{36}^l	$c_{36}^l = c_{34}^l \subset c_{35}^l$
$c_5 \cap c_{11}$	×	1	×	1	0	1	c_{37}^l	$c_{37}^{\ l} = c_{35}^{\ l}$
$c_6 \cap c_{11}$	0	1	0	1	0	1	c_{38}^l	$c_{38}^{\ l} = c_9^{\ l} \subset c_{37}^{\ l}$
$c_6 \cap c_{12}$	0	0	0	×	0	0	c_{39}^l	$c_{39}^l = c_{10}^l \subset c_{26}^l$
$c_7 \cap c_{11}$	×	1	0	1	0	1	c_{40}^l	$c_{40}^l \subset c_{35}^l$
$c_8 \cap c_{12}$	×	0	0	×	0	0	c_{41}^l	$c_{41}^l = c_{18}^l$

Результаты пересечений с учётом удаления поглощающихся кубов дают следующее множество кубов единого формата:

$C_i^{\ l} = \{c_{17}^l, c_{18}^l, c_{19}^l, c_{29}^l, c_{35}^l, c_{41}^l\}$.

Комплексное кубическое покрытие, построенное методом с ограничениями (с заказом), представлено в таблице 5.5.

Таблица 5.5

$c_i^l =$

R	T	y_1	y_2	y_1'	y_2'	Кубы c_i^l
0	1	×	×	0	×	c_{17}^l
×	0	0	×	0	0	c_{18}^l
1	1	×	0	1	0	c_{19}^l
×	0	1	×	1	1	c_{29}^l
×	1	×	1	0	1	c_{35}^l
×	0	0	×	0	0	c_{41}^l

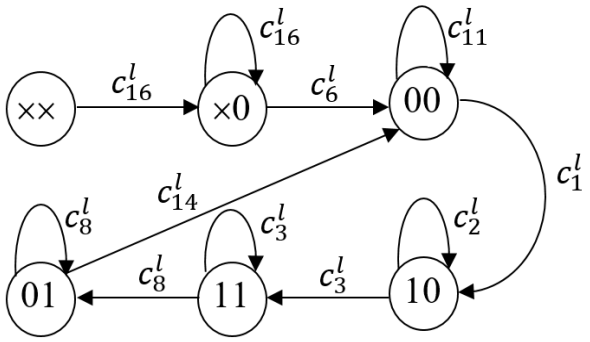


Рис. 5.6. Граф переходов ФЛС

Граф переходов ФЛС, построенный по покрытию c_i^l, вычисленному в прямом направлении методом с ограничением, приведён на рис. 5.7. Сравнивая графы переходов, представленные на рис. 5.6 и на рис. 5.7, можно установить их изоморфность и, если перейти к обозначениям дуг вектором $X = RT$, то и их идентичность. Этого и следовало ожидать.

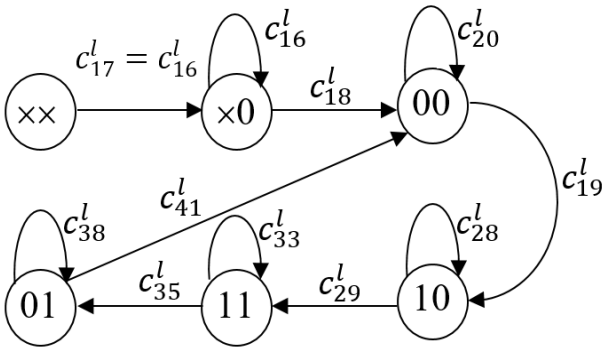


Рис. 5.7. Граф переходов схемы, восстановленный по покрытию, построенному методом пересечения с ограничениями

Отметим, что при вычислении переходов необходимо в процессе вычислений удалять поглощающиеся кубы, чтобы не раздувать промежуточные результаты, что наглядно показано в таблице 5.4.

Построение алгебро-топологической модели последовательностной схемы в виде комплексного кубического покрытия на примере схемы пересчёта со сбросом рассмотрено в Приложении 2.

5.3. Верификация моделей одного уровня абстракции

В процессе проектирования одной из важных задач является верификация проектных решений на каждом этапе проектирования. При этом должны быть соблюдены необходимые и достаточные условия верификации. Верификация может проводиться как для моделей разного уровня представления, так и для одноуровневых моделей, адекватно описывающих функционирование спроектированной схемы или устройства. Такая задача возникает при перепроектировании существующего устройства на новый элементный базис или при переводе программ на другую языковую и вычислительную платформу.

5.3.1. Методы верификации цифровых схем

Рассмотрим применение условий необходимости и достаточности при верификации схемных решений методом моделирования. Пусть задана некоторая булева функция f своими покрытиями $C^1(f)$, при $f = 1$, и $C^0(f)$ при $f = 0$, которые построены произвольным методом, например по карте Карно. Требуется верифицировать некоторое схемное решение для f в виде логической схемы N (с внешним выходом ZN), спроектированной, например, эвристическим способом в произвольном базисе с применением методов факторизации, декомпозиции или их сочетанием.

Необходимым условием верификации является совпадение реакций схемы $ZN = 1$ ($ZN = 0$) для $\forall\, c \in C^1(f)$ ($\forall\, c \in C^0(f)$. Но этого еще недостаточно для верификации схемы N.

Достаточным условием верификации является совпадение реакции схемы $ZN = 0$ ($ZN = 1$) для $\forall\, c \in C^0(f)$ ($\forall\, c \in C^1(f)$).

Таким образом, необходимым и достаточным условиями верификации схемы N является совпадение реакции схемы ZN со значениями булевой функции f для всех кубов $c \in C^1(f) \cup C^0(f)$. Комплексное покрытие $KP = C^1(f) \cup C^0(f)$ и есть метамодель схемы N.

Верификацию схемы N можно осуществить и другим способом, отличным от метода моделирования. Построим метамодель схемы для значений выхода ZN, равных 0 и 1, в виде комплексного покрытия $KP = C^0(ZN) \cup C^1(ZN)$. В этом случае требуется установить соответствие между покрытиями $C^1(f) \cup C^0(f)$ и $C^0(ZN) \cup C^1(ZN)$, которое может быть выполнено либо с использованием операции вычитания кубов покрытий (#), либо с помощью операции пересечения кубов покрытий ($\cap$). При использовании операции вычитания для верификации схемы N необходимо, чтобы $C^1(f)$ # $C^1(ZN) = \varnothing$, и достаточно, чтобы $C^1(ZN)$ # $C^1(f) = \varnothing$. Аналогично для покрытий $C^0(f)$ и $C^0(ZN)$. Из этого следует, что при использовании операции вычитания достаточно иметь только один тип покрытий: либо единичное $C^1(f)$ и $C^1(ZN)$, либо нулевое $C^0(f)$ и $C^0(ZN)$.

Применение алгебро-топологических операций вычитания и пересечения покрытий позволяет при сравнении множеств избежать точного соответствия элементов множеств между собой, поэтому $C^1(f)$ # $C^1(ZN)$ и $C^1(ZN)$ # $C^1(f)$ не соответствуют аналогам (не топологическим) обычного вычитания множеств ($A\backslash B$ и $B\backslash A$).

Итак, применив операции вычитания (#) и пересечения ($\cap$) покрытий при верификации объектов получим четыре возможных отношения:

1. $C^1(f)$ # $C^1(ZN) = \varnothing$ и $C^1(ZN)$ # $C^1(f) = \varnothing$, или $C^1(f) \cap C^0(ZN) = \varnothing$ и $C^0(f) \cap C^1(ZN) = \varnothing$ - условие полной верификации.

2. $C^1(f)$ # $C^1(ZN) \neq \varnothing$ и $C^1(ZN)$ # $C^1(f) = \varnothing$, или $C^1(f) \cap C^0(ZN) \neq \varnothing$ и $C^0(f) \cap C^1(ZN) = \varnothing$ - необходимое, но недостаточное условие верификации.

3. $C^1(f)$ # $C^1(ZN) = \varnothing$ и $C^1(ZN)$ # $C^1(f) \neq \varnothing$, или $C^1(f) \cap C^0(ZN) = \varnothing$ и $C^0(f) \cap C^1(ZN) \neq \varnothing$ - достаточное но не необходимое условие верификации.

4. $C^1(f)$ # $C^1(ZN) \neq \varnothing$ и $C^1(ZN)$ # $C^1(f) \neq \varnothing$, или $C^1(f) \cap C^0(ZN) \neq \varnothing$ и $C^0(f) \cap C^1(ZN) \neq \varnothing$ - условия верификации отсутствуют.

Во 2-м и 3-м случаях можно говорить о верификации частично определенных функций (объектов) с использованием «размытых» множеств, иначе говоря, один объект «делает» больше, чем другой. Таким образом, из вышеизложенного следует, что полная верификация существует только при выполнении соотношений 1-го случая. Данное рассуждение проведено для случая одновыходной схемы N и исходного покрытия булевой функции f, которое является простейшим случаем метамодели высшего ранга. Аналогичные методы можно применить при верификации граф-схем алгоритмов, конечных автоматов (абстрактных и структурных), многовыходных последовательностных схем и программ.

Предложенный метод применим и при верификации двух схем, спроектированных в разных элементных базисах и/или разными группами разработчиков (C_1 и C_2). В этом случае в операциях алгебро-топологического пересечения и вычитания участвуют комплексные покрытия соответствующих схем. Аналогично и с верификацией программ.

1. C_1 # $C_2 \neq \varnothing$ & C_2 # $C_1 \neq \varnothing$; $C_1 \cap C_2 = \varnothing$.
2. C_1 # $C_2 \neq \varnothing$ & C_2 # $C_1 \neq \varnothing$; $C_1 \cap C_2 \neq \varnothing$.
3. C_1 # $C_2 = \varnothing$ & C_2 # $C_1 \neq \varnothing$;
4. C_1 # $C_2 \neq \varnothing$ & C_2 # $C_1 = \varnothing$;
5. C_1 # $C_2 = \varnothing$ & C_2 # $C_1 = \varnothing$.

Интерпретация результатов верификации по комплексным покрытиям представлена на рисунке 5.8.

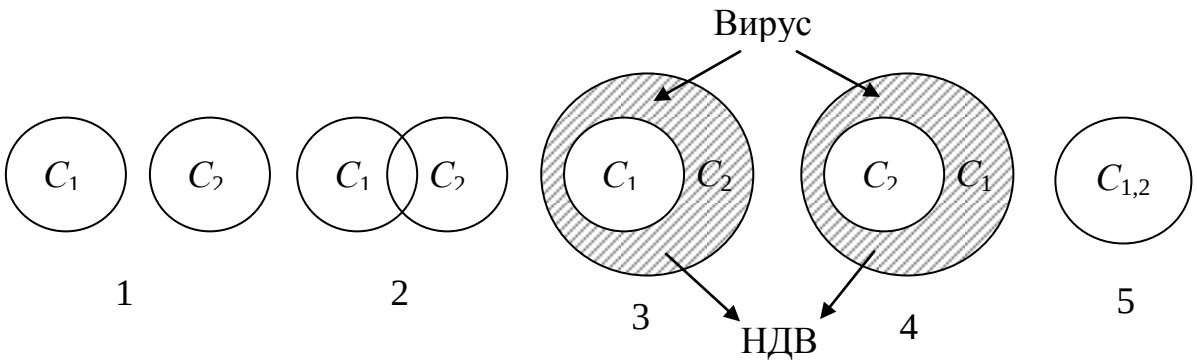


Рис. 5.8. Интерпретация результатов верификации по комплексным покрытиям

Контрольные вопросы
по темам «Тестирование и верификация цифровых схем»

1. В чём отличие между валидацией и верификацией?
2. Что понимается под формальной и функциональной верификациями?
3. Какую информацию содержат координаты эквивалентного покрытия?
4. Как формально определить наличие статического риска сбоя?
5. Чем является комплексное кубическое покрытие схемы?
6. От чего зависит последовательность входных наборов в тесте?
7. Приведите необходимое и достаточное условия верификации двух схем, разработанных по одной спецификации (одному ТЗ).
8. Что понимается под вырожденным покрытием?
9. Приведите пример частично определённого и полностью определённого графов переходов схемы.
10. Какие методы могут быть использованы для сокращения полного перебора при построении комплексного покрытия схемы?

ЗАКЛЮЧЕНИЕ

В пособии рассмотрены методы верификации вычислительных процессов, реализованных цифровыми схемами. Разобраны методы формальной и функциональной верификаций комбинационных и последовательностных схем на основе алгебро-топологических методов, слабо представленных в учебной литературе. Подробно рассмотрен синтез тестов для функциональной верификации. В задачу авторов не входило изложение методов верификации и моделирования в системах Verylog, VHDL и т.п., так как они подробно описаны в литературе и документации разработчиков этих систем. Во второй части учебного пособия будут рассмотрены методы верификации вычислительных процессов, реализованных в виде программ.

Библиографический список

1. Проектирование цифровых вычислительных машин / С.А. Майоров, Г.И. Новиков, О.Ф. Немолочнов и др. Под ред. С.А. Майорова. М.: Высш.шк., 1972. 344 с.
2. Новиков А.Ф. Дискретная математика для программистов: Учебник для вузов. 3-е изд. — СПб.: Питер, 2009. — 384 с.
3. Зыков, Анатолий Геннадьевич. Методы верификации аппаратно-программных компонентов вычислительных систем: диссертация кандидата технических наук: 05.13.12 / Зыков Анатолий Геннадьевич; [Место защиты: С.-Петерб. гос. ун-т информац. технологий, механики и оптики]. - Санкт-Петербург, 2008. - 154 с. : ил.
4. П.С. Довгий, В.И. Поляков, В.И. Скорубский. Основы теории множеств и приложение булевой алгебры к синтезу комбинационных схем. Учебное пособие по дисциплине «Дискретная математика». – СПб: НИУ ИТМО, 2012. – 109 с.
5. Немолочнов О.Ф., Зыков А.Г., Поляков В.И., Петров К.В. Учебно-исследовательская САПР верификации и тестирования вычислительных процессов программ //Научно-технический вестник СПбГУ ИТМО. – 2006. – № 9 (32). – С. 127–128.
6. Зыков А.Г., Безруков А.В., Немолочнов О.Ф., Поляков В.И., Андронов А.В. Графо-аналитические модели вычислительных процессов в САПР // Научно-технический вестник СПбГУ ИТМО. – 2011. – № 4 (74). – С. 116–120.
7. Немолочнов О.Ф., Зыков А.Г., Поляков В.И., Македонский А.А. Параллельные структуры управления вычислительными процессами в САПР // Научно-технический вестник СПбГУ ИТМО. – 2011. – № 4 (74). – С. 121–124.
8. В.Ю. Скобцов, Ю.А. Скобцов. Логическое моделирование и тестировании цифровых устройств. Донецк:ИПММ НАНУ, ДонНТУ, 2005.-436 с.
9. Скобцов Ю.А., Сперанский Д.В., Скобцов В.Ю. Моделирование, тестирование и диагностика цифровых устройств.-М.:Национальный открытый университет “ИНТУИТ”, 2012. – 439 с.
10. [http://sevntu.com.ua/conference/virt/Materials/Shkil/text3/tema3/As_tr.htm Основы теории конечных автоматов]
11. [http://abc.vvsu.ru/Books/Teh_diag/page0017.asp Состязания сигналов в цифровых схемах]
12. [http://about-evm.ru/Pages/osobennosti_logicheskogo_proektirovaniya_cu_organizaciya_vzaimodejstviya_e-lementov_v_uzlah.html Особенности логического проектирования ЦУ.]
13. Р. С. Зыбин, В. В. Кулямин, А. В. Пономаренко, В. В. Рубанов, Е. С. Чернов. Автоматизация массового создания тестов работоспособности //Программирование, 34(6): С.64-80, 2008.

14. Wiemann A. Standardized functional Verification. Springer, San Carlos, CA USA. 2008. 289 p.
15. Ganai M., Gupta A. SAT-Based Scalable Formal Verification Solutions. New York: Springer-Verlag. 2007. 338 p.
16. Kuehlmann A., Cornelis A.J. van Eijk. Combinational and Sequential Equivalence Checking // in: Logic synthesis and Verification (editors S.Hassoun, T.Sasao, R.K.Brayton). Kluwer Academic Publishers. 2002. P. 343–372.
17. Cheremisinova L., Novikov D. Simulation-based approach to verification of logical descriptions with functional indeterminacy // Information Theories & Applications (IJ ITA). 2008. V. 15. № 3. P. 218-224.
18. Новиков Д.Я., Черемисинова Л.Д. Исследование методов верификации описаний с функциональной неопределенностью на основе моделирования // Автоматика и вычислительная техника. 2012. № 5. С. 13-25.
19. Закревский А.Д., Поттосин Ю.В., Черемисинова Л.Д. Логические основы проектирования дискретных устройств. М.: Физматлит, 2007.
20. Черемисинова Л.Д., Новиков Д.Я. Формальная верификация описаний с функциональной неопределенностью на основе проверки выполнимости конъюнктивной нормальной формы // Автоматика и вычислительная техника. 2010. № 1. С. 5–16.
21. Cheremisinova L., Novikov D. SAT-Based Approach to Verification of Logical Descriptions with Functional Indeterminacy // 8th Int. Workchop on Boolean problems, Freiberg (Sachsen). Sept. 18–19, 2008. P. 59-66.
22. Cheremisinova L., Novikov D. SAT-based method of verification using logarithmic encoding // Information Science and Computing. 2009. № 15. P. 107-114.
23. Een N., Sorensson N. An Extensible SAT-solver // Theory and Applications of Satisfiability Testing (SAT 2003). Proc. of the Intern. Conf. Santa Margherita Ligure, Italy. May 5– 8, 2003. Microsoft Research. P. 502-518.
24. ModelSim: HDL Simulation / Mentor Graphics Corporation [Electronic resource]. – Mode of access: http://www.mentor.com/products/fv/modelsim/. Date of access: 2014.
25. Бибило П.Н., Кардаш С.Н., Романов В.И. СиВер – система синтеза и верификации комбинационных логических схем // Информатика. 2006. № 4 (12). С. 79-87.
26. Espresso examples / University of California Berkeley [Electronic resource]. – Mode of access: http://www1.cs.columbia.edu/~cs4861/ sis/ espressoexamples/ex. Date of access: 2014.
27. Бибило П.Н., Новиков Д.Я. Верификация логических схем, реализующих системы частичных булевых функций // Информатика. 2011. № 3. С. 68-76.

28. Магеррамов Р. В. Процесс тестирования интегральных микросхем // Молодой ученый. — 2015. — №13. — С. 154-158.
29. Лохов А.Л. Современные методы функциональной верификации цифровых HDL-проектов: методология ABV, библиотеки OVL и QVL // Современная электроника. 2010. № 1. С. 56–59.
30. Ровнягин М.М., Лебедев М.С., Чудновский А.Л. Современные методы верификации и особенности их применения // Современные информационные технологии и ИТ-образование // VI Междунар. науч.-практич. конф.: сб. избр. тр.; [под ред. В.А. Сухомлина]. М.: Изд-во ИНТУИТ.РУ, 2011. С. 1009–1020.
31. Барских М.Е., Аряшев С.И., Рогаткин Б.Ю. Современные методы функциональной верификации RTL-моделей блоков СБИС микропроцессора // Проблемы разработки перспективных микро- и наноэлектронных систем: сб. тр.; [под общ. ред. А.Л. Стемпковского]. М.: Изд-во ИППМ РАН, 2014. Ч II. С. 119–122.
32. Захаров А.В., Хисамбеев И.Ш., Котович Н.В., Кравченко А.А., Осипов А.С., Кольцов П.П., Коганов М.А., Грибков И.В., Куцаев А.С. Развитие системы стохастического тестирования микропроцессоров INTEG // Программные продукты и системы. 2010. № 2. С. 14–23.
33. Слинкин Д.И. Анализ современных методов тестирования и верификации проектов сверхбольших интегральных схем //Программные продукты и системы. 2017. №3. С. 401-408.
34. Гришкин В.М., Лопаткин Г.С., Михайлов А.Н., Овсянников Д.А. Интерфейсный метод построения моделей входных воздействий для тестирования электронных цифровых модулей // Вопросы радиоэлектроники. 2013. Т. 1. № 1. С. 80-89
35. Мельник В., Гришкин В., Михайлов А., Овсянников Д. Методика разработки тест-программ контроля и диагностики цифровых устройств с использованием САПР SimTest // Электроника: Наука, технология, бизнес. 2013. № S (128). С. 118-124.
36. Чернов А.В., Сергеева Е.А. Автокорреляционное тестирование цифровых комбинационных схем // Современные проблемы науки и образования. – 2013. – № 6.
37. URL: http://science-education.ru/ru/article/view?id=11526

Приложение 1

Устранение критических состязаний сигналов в логических схемах при реализации вычислительных процессов

Вычислительный процесс в логических схемах имеет ряд отличительных особенностей, главной из которых является параллелизм преобразования информации над значениями переменных в двоичном коде. То есть любая переменная *x, y* и *z* может принимать только два значения 0 и 1, что и позволяет работать с ними по законам формальной логики, полагая *true*=1 и *false*=0. В этом случае основная задача состоит в синхронизации по тактам в виде синхротактов с целью устранения критических состязаний сигналов, которые приводят к неоднозначности преобразования информации. Состязания сигналов, вызванные задержками на логических элементах и в линиях связи, проявляются в виде статического и динамического рисков сбоя. Статический риск сбоя делится на логический и функциональный риски. Логический риск можно устранить путем введения в схему избыточных элементов, сохраняющих постоянное значение булевых функций во время переходного процесса. Функциональный риск сбоя принципиально не устраним. Если риск сбоя в виде появления ложных сигналов запоминается на элементах памяти, то он является критическим. Устранение такого сбоя и есть задача синхронизации вычислительного процесса, а, именно, разбиение процесса во времени на такты. Вычисления, как и в обычном вычислительном процессе, проводятся по итеративным и рекуррентным формулам. Это приводит к появлению D-триггеров и 2D-триггеров. D-триггер есть элемент с одной линией обратной связи, по которой и происходит запоминание информации в один бит.

Вся логическая схема, таким образом, разделяется на комбинационную часть, в которой, собственно, и происходит преобразование информации, и блок памяти, где происходит ее запоминание для последующей обработки по алгоритмам, разделенных во времени по тактам.

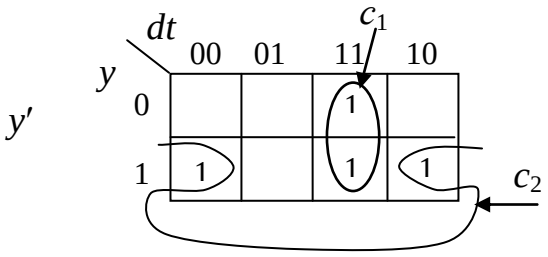
В качестве элемента памяти рассмотрим D-триггер, который запоминает данные d в виде сигналов обратной связи $y=d$ и управляется синхросигналами t и $\bar{t}$. Синхросигналы также могут принимать только два значения 0 и 1, что позволяет работать с ними по законам формальной логики в виде временных логических функций или, в другой терминологии, по законам темпоральной логики.

В большинстве источников D-триггер строят на основе RS-триггера, синтезируя из входных сигналов d и t входные сигналы R и S. Но D-триггер имеет свою логику работы, которую можно реализовать в виде последовательностной схемы.

Закон функционирования D-триггера можно выразить следующей интервальной формулой:

$$y' = \begin{cases} d, & \text{при} \quad t = 1; \\ y, & \text{при} \quad t = 0. \end{cases} \qquad (1)$$

То есть, при t = 1 триггер устанавливается в значении d на его входе, а при t = 0 происходит хранение $y' = y$ в петле обратной связи. Булеву функцию y' можно изобразить на карте Карно как функцию от трех переменных: d, y и t. На рис. П1.1 показана функция y' и ее минимальное покрытие.



$$C_{\min}(y') = \left\{ \begin{array}{ccc|c} d & t & y & y' \\ \hline 1 & 1 & \times & 1 \\ \times & 0 & 1 & 1 \end{array} \right\} \begin{array}{l} c_1 \\ c_2 \end{array}$$

Рис. П1.1. Покрытие функции y'

Минимальная дизъюнктивная нормальная форма функции y' имеет следующий вид:

$$y' = dt \vee y\bar{t} \qquad (2)$$

Построим схему D-триггера по минимальной форме (рис. П1.2.).

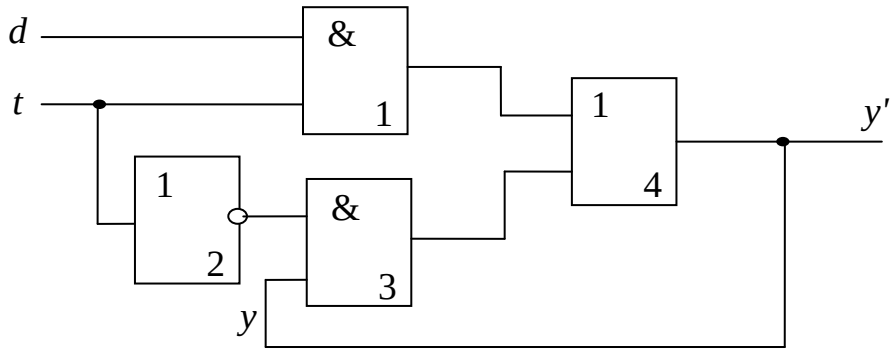


Рис. П1.2. Схема D-триггера, построенная по минимальной форме

При t=1 и d=1 терм td становится равным 1, и соответствующий ему элемент &(1) вырабатывает на выходе 1. В результате через элемент 1(4) на выходе схемы y' и в цепи обратной связи y устанавливается 1. При изменении входного сигнала d с 1 на 0, на выходе элемента &(1) появится 0, а при t=1 $(\bar{t} = 0)$ и на выходе &(3) также будет 0. Выход элемента 1(4) обнуляется, т.е. y'=y=0. Теперь, после окончания синхросигнала t $(\bar{t} = 1)$, на выходе &(3) остается 0, и триггер запоминает нулевое, ложное значение. Для устранения этой гонки Д. Хаффман предложил в цепь обратной связи ввести задержку (рис. П1.3.). Теперь единица на входе y «дождется» окончания синхросигнала $(\bar{t} = 1)$, и элемент &(3) выработает единицу, которая переключит в 1 и выход y'. Таким образом, триггер запомнит правильное, единичное значение.

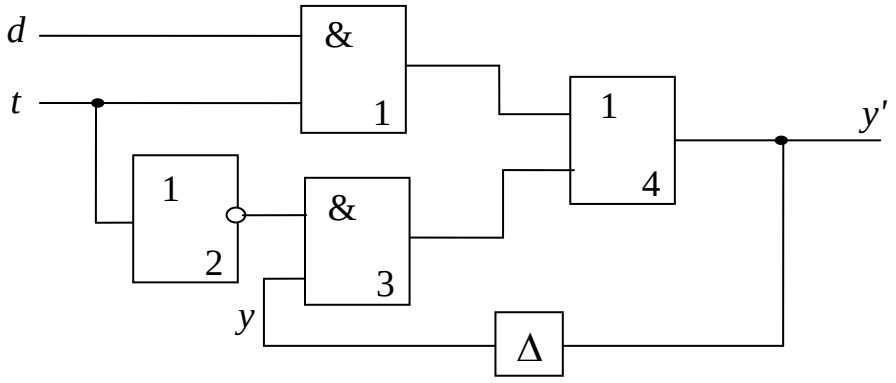
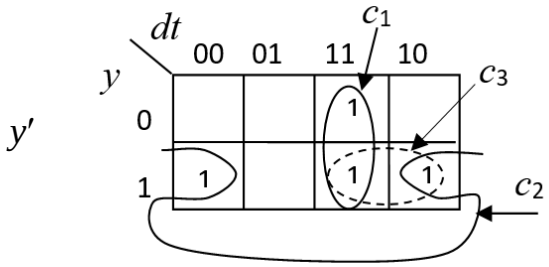


Рис. П1.3. Схема D-триггера, построенная по модели Хаффмана

Но построенная схема также не свободна от гонок и имеет риск сбоя.

Если сигнал y = 1 выработается позже, чем произойдет изменение синхросигнала t с 1 на 0, при входном сигнале $d = 1$, то на выходах элементов &(1) и &(3) появятся два 0, которые переключат триггер в ложное, нулевое значение. Для устранения этих состязаний необходимо ввести в покрытие функции y' избыточный куб c_3. На рис. П1.4 показана карта Карно функции y' и ее избыточное покрытие C_1.



$$C_1(y') = \left\{ \begin{array}{cccc|c} d & t & y & y' & \\ \hline 1 & 1 & \times & 1 & c_1 \\ \times & 0 & 1 & 1 & c_2 \\ 1 & \times & 1 & 1 & c_3 \end{array} \right.$$

Рис. П1.4. Избыточное покрытие функции y'

Дизъюнктивная нормальная форма функции y', построенная по избыточному покрытию, примет следующий вид:

$$y' = dt \vee y\bar{t} \vee yd. \qquad (3)$$

Избыточный терм yd порождает избыточный элемент &(5) в схеме (рис. П1.5).

Наличие терма, не зависящего от аргумента t, исключает возможность появления в схеме ложного значения y'=0 и, следовательно, обеспечивает устойчивое хранение y' при переходе с 0 на 1 без риска сбоя.

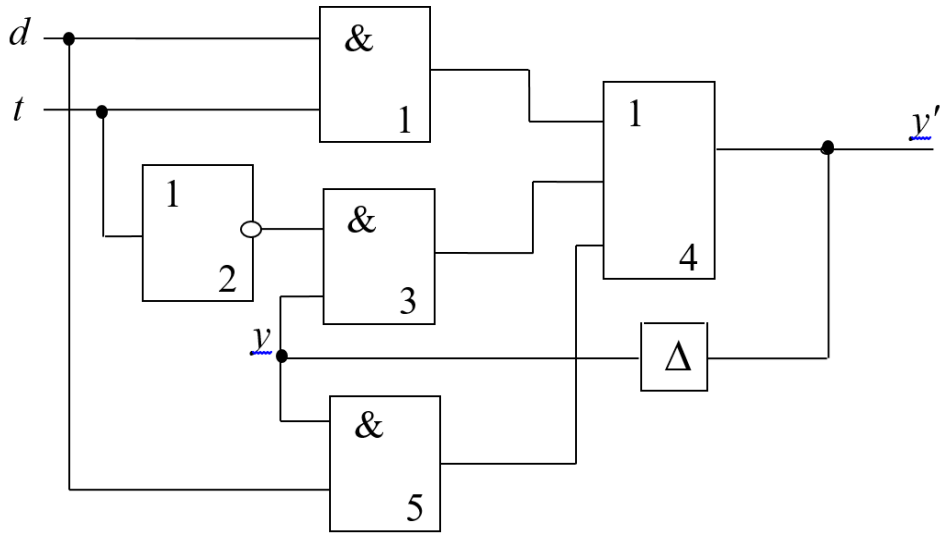


Рис. П1.5. Схема D-триггера, построенная по избыточному покрытию

Обобщая сказанное, можно сформулировать следующую теорему.

Теорема. Для устранения критических состязаний сигналов в D-триггере необходимо ввести в дизъюнктивную формулу y', описывающую закон его функционирования, избыточный терм yd и достаточно обеспечить значение этого терма $yd = 1$ на время переходного процесса Δ, то есть $d(t) = d(t+\Delta)$.

Необходимость и достаточность можно доказать методом от противного.

1. Предположим, что терм yd отсутствует, тогда в схеме существует статический риск сбоя, и он является критическим, что и было показано в предыдущих рассуждениях.

2. Предположим, что значение d хранится меньше времени переходного процесса в триггере, то есть $\Delta t < \Delta$, тогда терм yd обнулится раньше времени окончания переходного процесса Δ и, следовательно, в промежуток времени $\Delta - \Delta t \neq 0$ существуют условия возникновения статического риска сбоя. Если же $\Delta t = \Delta$, то терм $yd = 1$ на всем промежутке времени Δ переходного процесса в триггере, и это исключает появление ложного значения $y' = 0$, что и требовалось доказать.

Следствие. Хранение постоянного значения $d(t) = d(t+\Delta)$ служит основой построения 2D-триггеров, в которых отсутствуют критические состязания сигналов, так как их нет в составляющих его D-триггерах и, следовательно, 2D-триггер позволяет организовать рекуррентные вычисления любого вида устойчиво, то есть детерминировано без критических состязаний сигналов.

Из теории двойного D-триггера следует автоматная теория синхронных конечных автоматов, в которой каждое состояние $q_i \in Q$, распадается на два состояния q_i^t и $q_i^{\bar{t}}$, что, в свою очередь, порождает теорию синхронизации

вычислительных процессов на уровне автоматных моделей. Эти модели выступают как метамодель верификации вычислений в логических схемах в терминах временной (темпоральной) логики.

Выводы

Рассмотрение явления риска сбоя и методов его устранения позволяет проектировать устойчивые, детерминированные вычислительные процессы в логических схемах и разрабатывать теорию асинхронно – синхронных автоматов. Эта теория может быть использована как законченная и теоретически обоснованная основа проектирования встраиваемой логики в любых изделиях с искусственным интеллектом.

В контексте данного пособия любой конечный автомат с состояниями $q_i \in Q$ рассматривается как схема синхронизации вычислительных процессов: в АЛУ – сумматор на 2D-триггерах ($S_{i+1}=S_i \pm r$), или счетчик команд ($CK_i:=CK_{i-1} \pm k$), или сдвиговые регистры ($R:=R\times 2^m$ или $R:=R/2^m$); и любое устройство управления вычислительными процессами в цифровых схемах по данным и по управлению $t = \Delta t,\ \bar{t} = \Delta t.$

Приложение 2

Алгебро-топологическая модель последовательностной схемы в виде комплексного кубического покрытия

Процесс проектирования логических схем представляет собой последовательность этапов синтеза и анализа иерархических моделей вплоть до конкретной реализации схемы в заданном элементном базисе. Для решения задач анализа закон функционирования схемы описывается тем или иным способом. Для систем автоматизированного проектирования (САПР) одним из наиболее компактных и информационных является описание схем в виде кубических покрытий. Предложенная выше универсальная модель последовательностной схемы является основой для построения комплексного покрытия схемы. При этом схема декомпозируется на одновыходные подсхемы, каждая из которых описывается вырожденными покрытиями $C(f_i)$ в локальном формате Фл = {X,Y,Y',Z}, где X – независимые входы, Y- сигналы обратной связи, определяющие исходное состояние схемы, Y'- сигналы обратной связи, определяющие последующее состояние схемы и Z – выходные или внутренние значения комбинационных элементов (подсхем) схемы. Так как значения Z являются импликациями от независимых входов X и сигналов обратной связи Y и Y', то их можно вычислить после построения переходов схемы. Поэтому можно перейти к локальному формату
$Фл_1$ = {X,Y,Y'}, что является допустимым при теоретическом рассмотрении состояний и переходов схемы.

Построение комплексного покрытия всей схемы позволяет получить полную модель схемы для решения требуемых задач анализа. Комплексное покрытие (KC) строится в глобальном формате Ф$кс$ = {X,Y,Y',Z}. Любой куб комплексного покрытия $к_i \in KC$ трактуется как некоторое состояние логической схемы. KC строится методом пересечения множеств кубов покрытий подсхем на основе полного перебора, то есть всех возможных сочетаний. Таким образом:

$$KC = \bigcap_{i=1}^{n} C(f_i) - \{a \mid a \subseteq b; a,b \in KC\},$$

где n количество покрытий подсхем.

Простой куб $кс_i$ есть такой куб, в котором ни одну координату со значением p = 1 или 0, нельзя заменить на × без нарушения импликации $y_i' = p_i$ и $z_j = p_j$

Сокращение перебора может быть достигнуто методами импликации значений $C(y_i')$ и $C(Z)$, а также методами локальной и глобальной оптимизации на основе операции поглощения кубов.

Пусть имеется схема пересчёта (рис. П2.1), состоящая из двух триггеров: DT1 – синхронный DT триггер и DT2 – синхронный DT – триггер со сбросом, и после декомпозиции являющиеся одновыходными подсхемами.

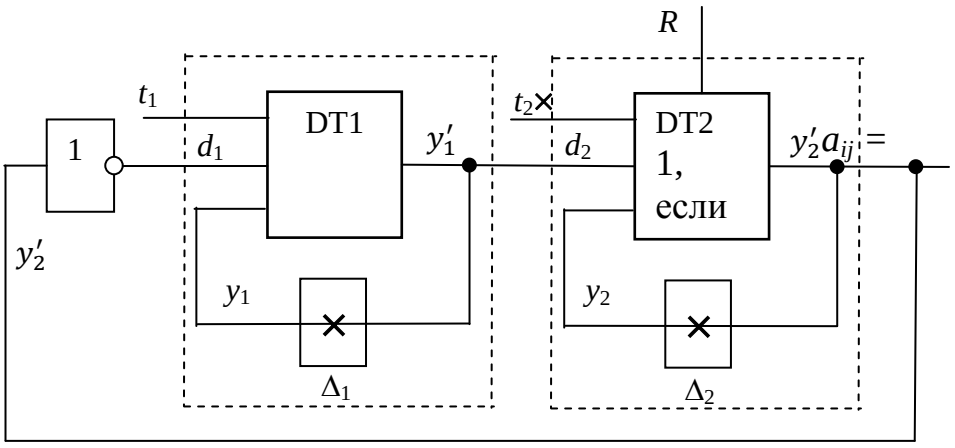


Рис. П2.1. Схема пересчёта со сбросом

Вырожденные покрытия $C(y_1')$ и $C(y_2')$ этих подсхем

$C(y_1')=$

t_1	d_1	y_1	y_1'	
1	1	×	1	c_1
×	1	1	1	c_2
0	×	1	1	c_3
1	0	×	0	c_4
×	0	0	0	c_5
0	×	0	0	c_6

$C(y_2')=$

R	t_2	d_2	y_2	y_2'	
0	1	1	×	1	c_7
0	×	1	1	1	c_8
0	0	×	1	1	c_9
1	×	×	×	0	c_{10}
×	1	0	0	0	c_{11}
×	×	0	0	0	c_{12}
×	0	×	0	0	c_{13}

В соответствии с предложенным алгоритмом пересечения и методами сокращения перебора построим комплексное покрытие схемы пересчёта в глобальном формате Фкс = {R, t_1, t_2, y_1, y_2, y_1',y_2'}.

Таблица П2.1

		R	t_1	t_2	y_1	y_2	y_1'	y_2'	Примечание
∩	c_1	×	1	×	×	×	1	0	
	c_{10}	1	×	×	×	×	×	0	
	k_1	1	1	×	×	×	1	0	– куб комплексного покрытия
∩	c_2	×	×	×	1	×	1	0	
	c_{10}	1	×	×	×	×	×	0	
	k_3	1	×	×	1	×	1	0	– куб комплексного покрытия
∩	c_3	×	0	×	1	×	1	×	
	c_9	0	×	0	×	1	×	1	
	k_m	0	0	0	1	1	1	1	– k_m куб поглощается кубом k_6
∩	c_3	×	0	×	1	×	1	×	
	c_{10}	1	×	×	×	×	×	0	– k_j куб поглощается кубом k_3 ($k_3 \supset k_j$) и из
	k_j	1	0	×	1	×	1	0	дальнейшего рассмотрения исключается.

Отметим некоторые моменты при построении КС. Покрытия в пересечение могут вступать в любом порядке при условии, что выбранный куб из некоторого покрытия должен быть пересечён со всеми кубами из

других покрытий. Приведём примеры пересечения некоторых кубов из покрытий $C(y_1')$ и $C(y_2')$ с учётом, что $d_1 = \overline{y_2'}$ и $d_2 = y_1'$ (табл. П2.1).

Отметим также, что, например, пересечение кубов c_1 и $c_2 \in C(y_1')$ с кубами c_7, c_8 и c_9 пустое (= ∅) по координате y_2' с учётом, что $d_1 = \overline{y_2'}$. Выполнив всю процедуру построения комплексного покрытия, получим покрытие всей схемы пересчета.

$KC =$

R	t_1	t_2	y_1	y_2	y_1'	y_2'	
1	1	×	×	×	1	0	k_1
×	1	0	×	0	1	0	k_2
1	×	×	1	×	1	0	k_3
×	×	0	1	0	1	0	k_4
0	0	1	1	×	1	1	k_5
0	0	×	1	1	1	1	k_6
0	1	0	×	1	0	1	k_7
0	×	0	0	1	0	1	k_8
1	0	×	0	×	0	0	k_9
×	0	1	0	×	0	0	k_{10}
×	0	×	0	0	0	0	k_{11}

Построенное комплексное покрытие является алгебро-топологической моделью схемы и позволяет решать задачи моделирования, решения установочной задачи, анализа состязаний, синтеза тестовых последовательностей, построения переходов и состояний схемы в прямом и обратном направлении при решении задачи верификации. При построении комплексного покрытия была использована только операция пересечения кубов, что упрощает программную реализацию соответствующей подсистемы САПР.

Отметим, что предложенный метод представления и построения модели логической схемы позволяет автоматизировать соответствующие этапы проектирования цифровых систем.

Донецкая Юлия Валерьевна
Зыков Анатолий Геннадьевич
Поляков Владимир Иванович

Методы верификации вычислительных процессов

Учебно-методическое пособие

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49