7 ГЕНЕРАЦИЯ ТЕСТОВ КОМБИНАЦИОННЫХ СХЕМ

«Для того чтобы… процесс тестирования… гарантировал, что вычислительная система не имеет неисправных компонентов, условия тестирования… должны предпочтительнее разрабатываться на уровне самих компонентов, чем на уровне программируемых последовательностей…. Это единственный способ, которым все состояния работы каждой логической функции могут быть однозначно… определены и все логические компоненты в каждой логической функции выполнят возложенную на них задачу… таким образом производя минимальную программу, которая проверяет и выявляет неисправности...» – Ричард Д. Элдред (Richard D. Eldred), в статье 1959, «Тестовые процедуры, основанные на операторах символьной логики».

Упомянутыми выше словами Элдред (Richard D. Eldred) начал эру структурного логического тестирования схем в Datamatic. Работа Pota (Roth) в IBM обозначила настоящее начало систематической генерации тестов для определения аппаратных неисправностей в цифровых вычислителях и заложила математический базис для генерации тестовых последовательностей. Автоматическая генерация тестовых последовательностей (АГТП, ATPG) – это процесс создания последовательностей для тестирования схемы, который точно описан сетевой топологией логического уровня (схематически). Эти алгоритмы обычно работают с программным генератором неисправностей, который создает минимальный сжатый список неисправностей (см. главу 4) таким образом, разработчику нет необходимости беспокоиться о генерировании неисправностей. В определенном смысле, АГТП-алгоритмы многофункциональными в том, что они могут генерировать тестовые последовательности, они могут находить избыточную или ненужную схемную логику и они могут доказать соответствует ли одна схемная реализация другой схемной реализации [409, 410, 717].

Мы сначала опишем алгоритмы и представления, необходимые для АГТП. Затем мы познакомим с идентификацией избыточности (ИИД) – очень важным преимуществом АГТП-алгоритмов. Управляемость и наблюдаемость – критерии тестируемости (см. главу 6), используемые во всех главных АГТП-алгоритмах. Наконец, мы представим несколько ключевых комбинаций АГТП алгоритмов и покажем их поведение на примерах.

7.1 Алгоритмы и представления

Сейчас мы обсудим основные категории АГТП алгоритмов и представления поискового пространства, используемые алгоритмами.

7.1.1 Структурное и функциональное тестирование

Сначала мы объясним разницу между структурным и функциональным тестированием. Разделение тестирования на структурное и функциональное приписывается Элдреду (Richard D. Eldred). Однако, первая публикация о константной неисправности логический 0 (кн0) или 1 (кн1) для создания тестов была опубликована Гейли (Galey), Норби (Norby) и Ротом (Roth) в 1961 [237]. Позже Сешу (Seshu) и Фриман (Freeman) упомянули модель константой неисправности для параллельного моделирования неисправностей [587]. В 1963, Пудж (Poage) представил теоретический анализ константных неисправностей. Первый структурный тестовый метод использовался, чтобы проверить Honeywell Datamatic 1000 – центральный компьютер второго поколения на вакуумных лампах и диодах.

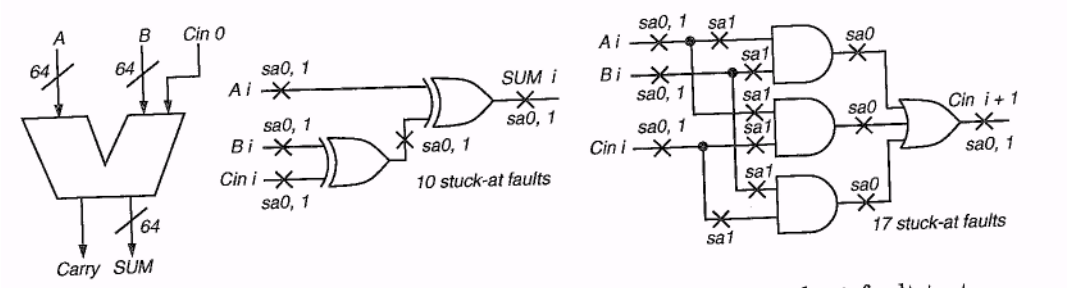

Figure 7.1: 64-bit adder: functional test vs. structural stuck-at fault test.

Функционально программы АГТП генерируют набор тестовых последовательностей, чтобы полностью изучить схемную функцию. На рисунке 7.1 изображен сумматор со сквозным переносом, и представлено очень простое (и неэффективное) логическое проектирование однобитового слоя сумматора, которого достаточно, чтобы доказать наше утверждение.С функциональной точки зрения, сумматор имеет 129 входов и 65 выходов. Таким образом, чтобы целиком протестировать функцию, нам нужно 2129 = 680,564,733,841,876,926,926,749, 214,863,536,422,912 входных наборов, производящих 265 = 36,893,488,147,419,103,232 выходных ответов. Быстрейшее автоматическое тестовое оборудование (АТО), в настоящее время, работает на частоте 1 ГГц. Это АТО должно будет работать 2.1580566142 x 1022 года, чтобы применить все эти образцы к тестируемой схеме (ТС), предполагая, что тестер и схема могут работать на частоте 1 ГГц. Поэтому, мы видим, что исчерпывающее функциональное тестирование невыполнимо, за исключением малых схем, но на сегодня большинство схем являются огромными.

Структурное тестирование, с другой стороны, только исследует минимальный набор константных неисправностей на каждой линии схемы, после отбрасывания эквивалентных неисправностей. Если мы используем эквивалентность неисправностей (смотрите главу 4), то каждый битовый слой в сумматоре будет иметь только 27 неисправностей (смотрите рис. 7.1), игнорируя эквивалентные неисправности по линиям переноса. Этот сумматор не имеет никаких избыточных аппаратурных затрат и полный структурный список неисправностей будет иметь не более чем 64 x 27 = 1,728 неисправностей. Так что нам нужно, по большей части, 1,728 тестовых последовательностей. 1 ГГц АТО должен применить эти образцы за 0.000001728 с, и так как этот набор тестовых последовательностей покрывает все возможные структурные константные неисправности в сумматоре, это охватывает те же самые неисправности: покрытие как и в труднообрабатываемом функциональном наборе тестовых последовательностей, описанных выше. Часто, проектировщик схем предоставит ограниченное подмножество функциональных тестовых последовательностей для схемы, но они обычно покрывают только от 70 до 75% количества всех неисправностей. Тестирование только 75% моделируемых ошибок – это ограниченное значение, т. к. будут обнаружены только самые грубые неисправности. Поэтому, мы видим важность алгоритмов АГТП. Векторы, которые они производят, дополняют функциональные тестовые векторы проектировщика, что позволяет увеличить покрытие константных неисправностей до уровня 98% или выше.

7.1.2 Определение автоматического генерирования тестовых последовательностей

АГТП алгоритмы вводят неисправность в схему, и затем используют разнообразные механизмы, чтобы активировать неисправность и распространить её по схеме и пронаблюдать на выводе. Выходной сигнал отличается от значения исправной схемы, и это позволяет обнаружить неисправность. Неисправность распространяется со входа И/И-НЕ вентиля к его выводу установкой других входов в 1, неуправляющее значение для И/И-НЕ. Неисправность распространяются со входа ИЛИ/ИЛИ-НЕ вентиля к его выводу установкой других входов в 0, неуправляющее значение для ИЛИ/ИЛИ-НЕ. Неисправность распространяются со входа XOR/XNOR вентиля к его выводу установкой других входов в 0 или 1 в зависимости от ситуации.

Е-лучевое тестирование [672] позволяет наблюдать внутренние сигналы схемы "проявляя" изображение схемы, которое показывает внутренние точки разветвления,

установленные в логический 0 одним цветом и установленные в логическую 1 другим цветом. Это позволяет не распространять неисправности на главные выходы (ГВых (PO)). Однако, этот метод непрактично дорог, используется только для очень специализированных приложений, и в некотором смысле преобразовывает труднообрабатываемую проблему тестирования в другую труднообрабатываемую проблему обработки изображений, так как некоторый механизм должен теперь просматривать изображение VLSI микросхемы и определять, "окрашены" ли все сигналы правильно. АГТП алгоритмы чрезвычайно важны, в том что они распространяют неисправные показания напряжения из внутренних цепей схемы к ГВых (PO), на которых АТО может исследовать напряжение и определить, правильно ли оно.

Проектирование на основе сканирования для тестирования микропроцессоров. В настоящее время, предпочтительным методом для тестирования по меньшей мере части Intel Pentium ™ и AMD K6 ™ микропроцессоров является комбинационная АГТП. Вставщик сканирующей цепочки добавляет мультиплексор специального назначения и синхронизационную аппаратуру к каждому триггеру схемы для контроля, так, чтобы в режиме сканирования, триггеры были преобразованы в большой сдвиговый регистр, и полное состояние микропроцессора может быть выдвинуто через специальный порт режима тестирования, называемый вывод сканирования (см. Главу 14).

Точно так же желаемое начальное состояние триггера может быть последовательно задвинуто в триггер через специальный порт режима тестирования, называемый вход сканирования. Этот подход преобразовывает сложную последовательную схемную АГТП проблему в комбинационную схемную АГТП проблему легче поддающуюся обработке, за счет:

1. использования 5 - 20 % площади микросхемы для аппаратных средств сканирующих цепочек в больших микросхемах.

2. замедление всех триггеров из-за введенных задержек мультиплексора сканирующей цепочки.

3. резервирования одного или большего количества дополнительных контактов для управления сканирущей цепочкой.

4. удлинения последовательности тестовых наборов. Это происходит, потому что установка машины, имеющей n триггеров в любое желаемое начальному состоянию требует n тактов сканирующей цепочки. Применение желаемой тестовой последовательности требует 1 дополнительный такт, следующий за n дополнительными тактами чтения состояния триггера (когда проявление неисправности зафиксировано в триггере, но не распространилось к выходу схемы.) Для множественного тестирования они могут пересекаться (см. Главу 14.)

Однако, проектирование на основе сканирования вместе с комбинационной АГТП - самый популярный метод тестирования микропроцессоров и других VLSI микросхем, потому что этот метод с высокой вероятностью генерирует набор тестовых последовательностей с близким к 100%-ому покрытию неисправностей. Кроме того, время на разработку тестов предсказуемо и может быть оговорено в новом графике внедрения продукта. Современная последовательная АГТП часто вызывает большие задержки разработки, из-за проблем алгоритмов и несовместимых аппаратных средств, и может задержать внедрение продукта. В Главах 14 и 16 рассмотрим проектирование на основе сканирования более подробно, но мы уже видим, что комбинационные программы АГТП чрезвычайно важны.

7.1.3 Абстракции поискового пространства

Все программы АГТП нуждаются в структуре данных, описывающей поисковое пространство для тестовых последовательностей.

Деревья Двоичного поиска. Рассмотрим двоичное дерево на рисунке 7.2 (b) для главных входов схемы (Pis) в рисунке 7.2 (a). Гоэль (Goel) [256, 258] сначала использовал эти деревья в комбинационной АГТП литературе. Дерево представляет все восемь выборов для рисунков ввода схемы. В самой верхней точке разветвления, если выбрана левая ветвь, то сигнал A установлен в 0 (ветвь A), но если выбрана правая, то A в 1. Во втором и третьем уровнях в дереве, последующие значения выбраны для других вводов схемы, сначала для B и затем для C. Это покрывает все возможные входные последоваетльности. Вершины дерева помечены подходящим машинным кодом, который соответствует входным значениям. Все АГТП

алгоритмы неявно ищут это дерево, чтобы найти тестовые последовательности, и в самом плохом случае, должны исследовать полное дерево, чтобы доказать, что неисправность нетестируема. Мы опустим подробное изучение, потому что количество листьев дерева равно числу основных входов и возрастает по экспоненте.

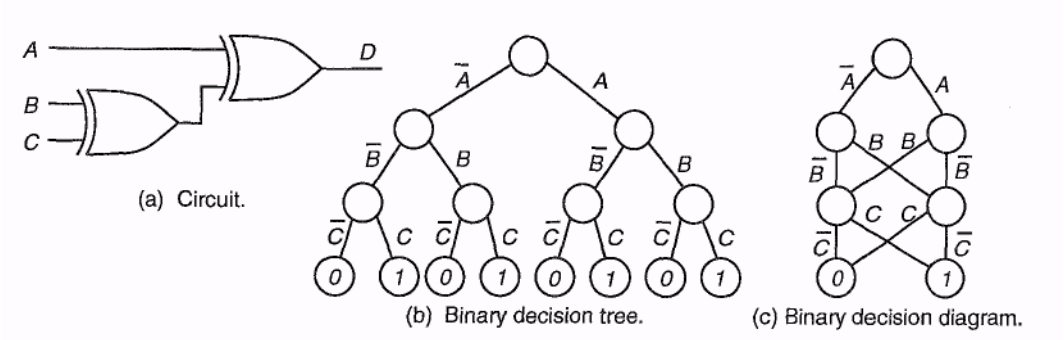

Figure 7.2: Different representations of a circuit.

Двоичные диаграммы принятия решений. Любая переключающаяся функция может быть полностью описана двоичной диаграммой принятия решений (ДДПР (BDD)), которая был изобретен Ли (Lee) в 1959 [385].

Настоящее обсуждение основано на работе Акерса (Akers), который применил ДДПР (BDD), чтобы решить проблемы тестирования [43, 44]. Рисунок 7.2 (c) показывает ДДПР (BDD) для схемы на рисунке 7.2 (a). Чтобы прочитать диаграму, мы начинаем в самом верхнем корневом узле, и следуем пути от этого узла до однойго из двух самых нижних узлов, 0 или 1, который дает значение выхода схемы. Произведение булевых литералов по пути дает элементарную дизъюнктивную форму схемы или минимальную дизъюнктивную форму схемы , в зависимости от того, заканчивается ли путь в 0-м или 1-м выходном узле. Например, крайний левый путь в ДДПР (BDD) - это A B C, который производит 0 на выходе схемы, и это совпадает с функцией схемы. Путь младшего разряда в ДДПР (BDD) - это A B C, который производит 1 на выходе схемы, также совпадает с функцией схемы. Мы можем проверить, что все ДДПР (BDD) пути совместимы с логической функцией. ДДПР (BDD) использовались для АГТП [234, 635], но в них есть проблемы вычислительной сложности, особенно для схем умножителя. Нельзя не заметить существенные потери вычислительного времени, в зависимости от порядка в котором схемные входы описаны в ДДПР (BDD)[104].

7.1.4 Результативность алгоритма

Понятие результативности АГТП алгоритма, означает, что для того, чтобы сгенерировать тестовую последовательность, алгоритм должен в конечном счете быть способен найти полное двоичное дерево решения, в случае необходимости, сгенерировать тестовую последовательность даже для трудно выявляемой неисправности. Если неисправность непроверяема, то после поиска полного дерева никаких тестов не найдено. Это означает что, эта схема ведет себя правильно даже в присутствии этой неисправности. Важно для АГТП алгоритма быть законченным, или он может не достигнуть необходимого покрытия неисправностей.

7.1.5 АГТП алгебра

АГТП алгебра - более высокий порядок представления булевой системы обозначений набора с целью одновременного представления значений "исправных" и "неисправных" схем (или автоматов). Это дает преимущество в необходимости только одного прохода АГТП, чтобы определить значения сигнала для обоих автоматов. Так как тестовый вектор требует, чтобы была разница между этими двумя автоматами, с вычислительной точки зрения самое быстрое – представить отношение обоих автоматов по алгебре, вместо того, чтобы сохранить их, отдельными. Рот (Roth) [551] показал, как активизация множественного пути, требовала протестировать некоторые комбинационные схемы, это могло быть сделано с его пятизначной алгеброй, данной в Таблице 7.1.

Table 7.1: Roth's five-valued and Muth's nine-valued algebras.

Symbol	Meaning	Roth's 5-valued algebra		Muth's 9-valued algebra	
		Good machine	Failing machine	Good machine	Failing machine
D	(1/0)	1	0	1	0
$\overline{D}$	(0/1)	0	1	0	1
0	(0/0)	0	0	0	0
1	(1/1)	1	1	1	1
X	(X/X)	X	X	X	X
$G0$	(0/X)	–	–	0	X
$G1$	(1/X)	–	–	1	X
$F0$	(X/0)	–	–	$\overline{X}$	0
$F1$	(X/1)	–	–	X	1

Позже, Мат [481] показал что, чтобы тестировать конечные автоматы, символ X должен быть расширен, чтобы покрыть случаи, когда одно из исправных или неисправных значений автоматов может быть известным, но другие значения автоматов - неизвестны. Таблица 7.1. показывает девятизначную алгебру Мата, которая также часто приносит пользу комбинационной АТГП [116]. Для того чтобы вычислить реакцию логического вентиля, чтобы ввести символы алгебры, мы разворачиваем символы в исправные схемные/неисправные схемные значения, данные в столбце 2 Таблицы 7.1. Мы тогда независимо вычисляем реакцию логического вентиля для обоих автоматов и объединяем значения выхода опять в алгебру.

7.1.6 Типы алгоритмов

Мы классифицируем различные типы АГТП алгоритмов, и представляем их степень интеграции.

Исчерпывающий. В этом подходе, для n-входной схемы, мы генерируем все 2-х входовые последовательности. По причинам, обсуждаемым выше, это неосуществимо, если схема не разделена в подсхемы логической схемы, каждая с 15 или меньшим количеством входов. Мы можем тогда выполнить исчерпывающее формирование тестовых последовательностей для каждой подсхемы. Однако, те неисправности, которые требуют множества подсхем, которые должны быть активизированы синергистическим способом в процессе тестирования, могут быть не проверены.

Случайный - Использовался с алгоритмическими методами. В 1972 в Университете Штата Иллинойс, Агравел (Agrawal) и Агравел (Agrawal) [26] предложил использование случайного генерирования последовательностей (СГП, RPG) для тестирования. Основная часть схемы СГП, RPG - имитатор ошибки, который выбирает пригодные последовательности (Рисунок 7.3.) При генерировании тестов для плат параллельного компьютера ILLIAC IV сообщили, что покрытие случайных последовательностей будет часто варьироваться между 60-80 % и что переключение на D-алгоритм обусловит преимущество программы в этом смысле. Это ограничение СГП, RPG, которое касается тестируемости схемы, было учтено Айхельбергером (Eichelberger) и другими [210] для программируемых логических матриц (ПЛМ). Стало ясно, что последовательности с одинаковой вероятностью выходов и входов, как начало СГП, RPG на рисунке 7.3, может быть не лучшим выбором [15, 16, 20]. Когда вероятности выходов и входов отличаются от 0.5, последовательности называют взвешенными случайными последовательностями (ВСП). Такие последовательности использовались Шнурманом (Schnurmann) и другими [571], Ваисукавским (Waicukauski) и Линдблумом (Lindbloom) [705], и некоторыми другими. Метод нахождения оптимального набора вероятностей для входов дается Вундерлихом (Wunderlich) [739]. О применении СГП, RPG для последовательных схем, см. обсуждение методов на основе моделирования в Главе 8. Недавняя книга Дэвида (David) [187] всесторонне раскрывает эту тему.

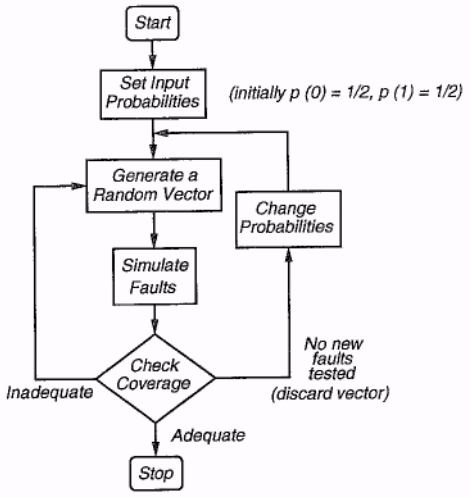

Figure 7.3: Random pattern generation method.

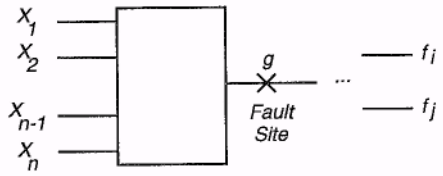

Figure 7.4: Boolean difference.

Символический - булева производная. Селлерз (Sellers) и другие [582, 583] используют Теорему Развертывания Шаннона, чтобы характеризовать Булевы схемы. Например, произвольная булева функция F(X1, X2 ..., Xn) может быть развернута по любой переменной, например X2, как:

$$F(X_1, X_2, \ldots, X_n) = X_2 \cdot F(X_1, 1, \ldots, X_n) + \overline{X_2} \cdot F(X_1, 0, \ldots, X_n)$$

Предполагая, что логическая функция g = G(Xi,X2, • • •, Xn) имеет неисправность как на рисунке 7.4, мы выразим выходы как:

$$f_j = F_j(g, X_1, X_2, \ldots, X_n) \quad ; \ 1 \le j \le m \qquad (7.1)$$
$$X_i = 0 \ \ or \ \ 1 \ \ for \ \ 1 \le i \le n$$

Обратите внимание, что эти выходные уравнения выражают выходы схемы в терминах множества первичных входовXi и сигнала с ошибкой g. Селлерз (Sellers) и другие определили булеву производную, или булеву частичную производную, схемы как:

$$\frac{\partial F_j}{\partial g} = F_j(1, X_1, X_2, \ldots, X_n) \oplus F_j(0, X_1, \ldots, X_n) \qquad (7.2)$$

Они выразили требования к обнаружению неисправности для g КН0 на выходе fj как:

$$G(X_1, X_2, \ldots, X_n) = 1 \qquad (7.3)$$

$$\frac{\partial F_j}{\partial g} = F_j(1, X_1, X_2, \ldots, X_n) \oplus F_j(0, X_1, \ldots, X_n) = 1 \qquad (7.4)$$

Уравнение 7.3 показывает, что, чтобы проверить ошибку константного нуля на g, логический вентиль G должен сделать чувствительным точку ошибки, установив её в логическую 1. Уравнение 7.4 говорит, что, чтобы обнаружить ошибку, булева производная некоторого вывода относительно точки неисправности g должна быть 1 (то есть, выход должен изменить свое значение сигнала, когда сигнал точки неисправности переключается с 1 в 0). К сожалению, из-за высокой сложности булева производная - не эффективный способ вычислять тестовые последовательности для больших схем.

Методы активизации путей. Активизация пути на уровне представления логического вентиля - в настоящее время предпочтительный АГТП метод. Подход состоит из трех шагов [550]:

1. активизация ошибки, при которой константная ошибка активизируется установкой сигнала в противоположное значение от значения ошибки. Это необходимо для уверенности в поведенческом различии между правильной схемой и дефектной схемой. Активизация ошибки также известна как активация ошибки или возбуждение ошибки.

2. распространение ошибки, при котором проявление ошибки распространено через один или более путей к главному выходу схемы. Для некоторых ошибок, необходимо одновременно распространить проявление ошибки по множественным путям, чтобы протестировать ее. Вообщем случае, количество путей может увеличиваться по экспоненте в зависимости от количества логических вентилей в схеме. Распространение ошибки также известно как активизация пути.

3. Обоснование линии, при котором внутренние назначения сигнала, предварительно сделанные для активизации ошибки или распространения ее проявления обосновываются установкой главных входов схемы.

Во втором и третьем шагах, мы можем найти конфликт, где необходимое назначение сигнала противоречит некоторым предварительно сделанным назначеним. Это вынуждает АГТП алгоритм отслеживать в обратном порядке или резервировать, то есть, отказываться от предварительно-сделанного назначения сигнала и делать альтернативное назначение.

Рассмотрим пример на рисунке 7.5 [550]. Во всех примерах в этой главе, мы обозначим главные входы и главные выходы заглавными буквами, и все остальные сигнальные линии в схеме строчными буквами. Обратите внимание, что главный вход B (основа разветвления) развеетляется на два вентиля И, выходы которых - h и i. Разветвление переходит от B до входов двух вентилей И отмеченных f и g. Часто случается, что тесты неисправностей в B отличаются от тестов неисправностей в f, которые также отличаются от тестов неисправностей на g. Именно поэтому мы должны отмечать каждую отдельную строку или провод сигнальной сети. Мы генерируем тест для B КН0. Для активизации ошибки, мы устанавливаем B в 1, и это ведет к назначениям сигнала f = D и g = D (см. Таблицу 7.1.)

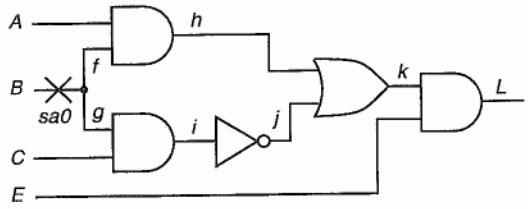

Figure 7.5: A combinational circuit example for path sensitization.

Распространение ошибки требует, чтобы мы выбрали один из трех сценариев:

1. распространение по пути f - h - k - L, или
2. распространение по пути g- i - j - k - L, или
3. одновременное распространение по обоим путям f - h - k - L и g - i - j - k - L.

Мы выбираем путь f - h - k - L для распространения. Это означает, что для каждого И вентиля по пути, входы вне пути должны быть установлены во неуправляющее значение (1), и также для каждого ИЛИ вентиля по пути, входы вне пути должны быть установлены в 0. Это приводит к назначениям сигналов A = 1, j = 0, и E = 1. Обоснование линии теперь требует, чтобы мы обосновали любые внутренние сигналы, которым назначались значения. В этом случае, единственный сигнал - j. Мы можем присвоить i = 1, чтобы обосновать j - 0 обратным логическим моделированием инвертора j. Однако, И вентиль i тогда должен иметь выход 1, но он уже имеет вход g = D. Обратное логическое моделирование, используя 5-оцененную алгебру (см. Таблицу 7.1), показывает, что нет никакого способа получить i = 1, когда вход уже был установлен в D. Поэтому, мы отслеживаем в обратном порядке и отказываемся от назначения j = 0 и пробуем альтернативное назначение j =1. Однако, это немедленно блокирует распространение ошибки по пути f - h - k - L. Мы приходим к выводу, что единственные пригодные варианты могут быть вышеупомянутыми сценариями 2 и 3 .

Мы выбираем сценарий 3. Мы изменяем наш подход распространения ошибки, и теперь назначаем A = 1, C = 1, и E = 1, чтобы гарантировать распространение ошибки. Прямое логическое моделирование результатов этих назначений i - D, h = D, j = D, и k = i, так как D

ИЛИ nD = 1. Это получено вычислением 1f0 ИЛИ 0f1 = 1f1. D-граница - это сечение, отделяющее часть схемы, помеченную X от части, помеченной D или nD, где мы включаем только D или nD наиболее близкие к выходам в границе. Наконец, L = 1 и D-граница исчезает в ИЛИ вентиле k, что означает, что ошибка непроверяема из-за множественных путей. Остается только вернуться и пробовать распространение ошибки по пути g - i - j - k - L. Мы устанавливаем C = 1, h = 0, и E = 1, чтобы распространить ошибку. Прямое логическое моделирование дает i=D, j=nD, k =nD и L = nD. Остается обосновать h = 0. Это достигается обратным логическим моделированием для И вентиля h, со вводом f = D, устанавливая вход A = 0. Единственный тест на КН0 на входе B выглядит так: ABCE = 0111, и дает выход L = 0 в исправном автомате, и L = 1 в неисправном автомате.

Эта процедура АГТП правильна только для нециклических комбинационных схем. Мы будем обсуждать процедуры для последовательных схем в Главе 8. В частности любая схема с обратными связями, триггерами, или неявными триггерами-защелками, выраженными как комбинационная логика будет часто обращать эту процедуру в бесконечный цикл. Распространение ошибки и обоснование линии в АГТП состоит из операций назначений сигналов, праямого моделирование логического вентиля, обратного моделирование логического вентиля, и обратного хода.

Булева выполнимость и методы импликации графов. Проблема булевой выполнимости означает удовлетворение булеву выражению или уравнение. N-битный булевый вектор состоит из набора n двойных переменных, Xi, i = 1,2..., n. Символически, переменная Xi или её дополнение nXi обозначается как литерал. Проблема двойной выполнимости сводится к обнаружению набора значений для rEj's, которые удовлетворит уравнению типа:

$$\sum \alpha_k \beta_k = 0 \quad (\text{non} - \text{tautology}) \quad \text{or} \quad \prod(\alpha_k + \beta_k) = 1 \quad (\text{satisfiability}) \qquad (7.5)$$

где αi и βi - любые два литерала, и суммирование, и умножение являются булевыми операциями ИЛИ и И, соответственно.

Терм или произведение в уравнении 7.5 называют булевым предложением или просто предложением. 2-SAT проблема разрешима в полиномиальный терм [190]. Когда предложения в булевом выражении содержат три литерала, проблема известна как (3-SAT) проблема с тройной выполнимостью. Решение 3-SAT имеет экспоненциальную сложность времени.

Чакрадар (Chakradhar) и другие [121, 124, 130] и Ларби (Larrabee) [383, 384] получили Булевские формулировки выполнимости для АТГП проблемы. Учитывая целевую неисправность, получают функцию мощности нейронной сети или Булевского произведения выражения сумм в терминах переменных сигнала схемы такой, что любой тест на целевую ошибку свернет функцию мощности или удовлетворит булеву выражению. Минимизация мощности показана эквивалентной булевской сумме произведений. Остальное зависит от нахождения эффективных путей решения проблемы выполнимости.

Эти методы были расширены другими [291, 605, 648], и теперь являются

наиболее быстрыми известными АГТП алгоритмами для больших схем. В этих методах, булева функция каждого логического вентиля зафиксирована в уравнениях, которые связывают входные и выходные сигналы вентилей. Рассмотрим отношения сигналов И вентиля на рисунке 7.6:

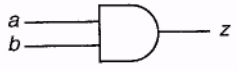

Figure 7.6: A two-input AND gate.

$$\begin{aligned} \text{If } a = 0, \text{ then } z = 0 \\ \text{If } b = 0, \text{ then } z = 0 \\ \text{If } z = 1, \text{ then } a = 1 \text{ AND } b = 1 \\ \text{If } a = 1 \text{ AND } b = 1, \text{ then } z = 1 \end{aligned} \qquad (7.6)$$

Для каждого ограничения куб разработан так, чтобы, если сигналы были последовательно помечены, то куб станет 0. Если любое значение сигнала вокруг логического вентиля несовместимо с функцией вентиля, то куб станет 1. Мы просто подводим итог кубам,

чтобы получить следующее булевское уравнение, где кубы показаны в порядке вышеуказанных условий:

$$\overline{a}\, z + \overline{b}\, z + z\, \overline{ab} + a\, b\, \overline{z} = 0 \qquad (7.7)$$

,

что упрощается:

$$\overline{a}z \; + \overline{b}z \; + ab\overline{z} = 0 \qquad (7.8)$$

Это уравнение удовлетворено только, когда a, b, и c принимают значения, которые являются совместимыми с функцией И вентиля. Первые два терма - 2-SAT термы, а третий - 3-SAT терм. Альтернативное представление называют псевдобулевым уравнением [124]. Мы преобразовываем булево выражение в псевдобулеву форму, заменяя булевские операторы OR И AND арифметическим сложением и умножением, и обрабатывая сигналы как "реальные" переменные, которые могут принять значения 0.0 и 1.0. Дополнения x заменяются 1.0-x. Для И вентиля, это форма:

$$F_{pseudo-Bool} = 2z + ab - az - bz - abz = 0.0 \qquad (7.9)$$

полученная из уравнения 7.8. Третье представление - функция [126] мощности, полученная, позволяя переменным принять любое реальное значение в диапазоне (0,1).

Альтернативный и более простой способ получать булевское представление уравнения - булевским ложным выражением [125], определенным как:

$$f_{AND}\,(a,b,z) = z \oplus (ab) = \overline{a}z + \overline{b}z + ab\overline{z} \qquad (7.10)$$

Выражение в правой части в уравнения 7.10 принимает значение логического 0 только, когда a, b, и z принимают значения, которые являются совместимыми с функцией AND. Дополнение fAND называют истинностным выражением или выражением выполнимости, которое принимает значение логической 1 только, когда значения a, b, и z совместимы с функцией AND. Эти выражения могут быть получены для любой сложной булевой функции, используя определения исключающего ИЛИ уравнения 7.10. Булевское ложное выражение может быть непосредственно преобразовано в функцию мощности нейронной сети, заменяя булевы операторы арифметическими операторами. Переменные тогда представляют состояния нейронов, которые могут принять или непрерывные значения или дискретные 0 и 1 значения как в модели Хопфилда (Hopfield) [304]. Применение модели нейронных сетей Хопфилда (Hopfield) для проблем тестирования было описано в книге [127].

Неинтуитивный аспект этих формулировок то, что выходы логического вентиля, так же как входы, появляются в выражениях. Преимущество этого подхода состоит в том, что мы можем записать функцию мощности для каждого логического вентиля (или модуль булевой функции) в схеме, и затем суммировать все эти функции в единственную функцию мощности для всей схемы. Если значение функции - 0, то все сигналы последовательно помечены; иначе они не отмечены.

Действительно эффективный способ свернуть функцию энергии или найти удовлетворяющие присвоения переменной для ложных или истинностных функций - это граф включения. Этот граф имеет узел для каждого литерала. Таким образом, булевская переменная x представлена двумя узлами, x и nx. Узел может быть "истинный" или "ложным". Для x = 1, x узел принимает истинное состояние. Для x = 0, nx узел становится истинным. Выражение "если ... то" для двух переменных представляется как дуга от представления литерала условия "если" к представлению литерала выражения "то". Граф может тогда быть преобразован в переходной замкнутый [122] граф так, чтобы, когда узел установлен в истину, все доступные узлы, также установлены в истину. Это позволяет очень эффективный анализировать значения сигнала, потому что переходной замкнутый контур может определить больше глобальных отношений сигнала в графе чем другие методы поиска ветвей и границ. Рисунок 7.7 (сплошные линии) показывает графу включения для “если ... то” выражения уравнения 7.6. Обратите внимание, что только двойные значения (те, которые содержат два литерала) могут быть представлены дугой. Узел A (пунктирные линии) обозначает оператор AND, и представляет 3-SAT терм

уравнения 7.8 (последнее выражение уравнения 7.6) [291]. Мы кратко рассмотрим эти алгоритмы в дополнительных темах(см. Раздел 7.5.4).

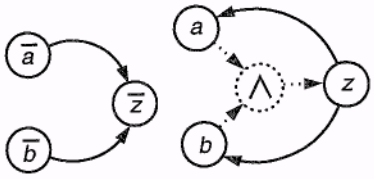

Figure 7.7: Implication graph (solid) and 3-SAT term (dotted) for 2-input AND.

Вычислительная Сложность. Ибарра (Ibarra) и Сани (Sahni) [317] анализировали вычислительную сложность АГТП. Они обнаружили, что это - NP-Complete проблема что означает, что никакое полиномиальное выражение для функции времени вычисления не было найдено, и проблема, как предполагается, имеет экспоненциальную сложность. Мы неформально обсудим, откуда это берется. В самом плохом случае, с неглавными входами, есть 2 различные входные комбинации неглавных входов, которые можно пробовать для способа поиска в глубину в двоичном дереве принятия решений. Когда триггеры присутствуют в схеме, есть потенциально 4 различные начальные состояния триггеров для АГТП, которые можно рассмотреть. Это потому что триггеры могут быть или в 0 или в 1 состоянии в схеме без ошибки и также в 0 или 1 состоянии в дефектной схеме. Таким образом, пространство состояний триггеров содержит четыре элемента (также см. раздел 8.2). Наконец, работа по прямому моделированию или обратному моделированию всех логических вентилей увеличивается пропорционально n, номеру логических вентилей. В самом плохом случае, эта работа должна быть сделана для всех потенциальных комбинаций главных входов и начальных состояний триггерных схем. Полное выражение для самого плохого случая вычислительной сложности АГТП :

$$O(n \times 2^{no\_pi} \times 4^{no\_ff})$$

Вышеупомянутое доказательство полагает, что АГТП математически эквивалентно проблеме булевской выполнимости.

Полной историей АГТП алгоритмов был процесс улучшения эвристических алгоритмов, и процедур для (1) нахождения всех необходимых присвоений сигналов для тестирования как можно раньше, и (2) поиск настолько малого пространства решений, насколько возможно. Пространство решений самого плохого случая - 2no-pl x ^no-ff. Для логического моделирования вычислительная сложность составляет O(n). Для комбинационного моделирования неисправности, сложность составляет $O(n^2)$ [277], и для последовательного моделирования ошибки, сложность оценена между O (n^2) и O (n^3), основываясь на эмпирических измерениях.

Это означает что, всякий раз, когда возможно, мы будем использовать моделирование неисправности, чтобы избежать АГТП вычислений. Например, мы используем СГП, RPG и моделирование неисправности, чтобы получить тесты. Когда это неудается, мы используем АГТП для трудно тестируемых неисправностей. Если мы находим последовательность для ошибки, то мы моделируем эту последовательность вместо всех оставшихся необнаруженных ошибок, в надежде, что мы "случайно" проверим дополнительные ошибки.

1. огромное количество различных неисправностей, возможных в больших схемах.

2. экспоненциальная сложность алгоритма (то есть, для последовательной схемы только с 20 триггерами, последовательный АГТП может продлиться дни вычислений), так как последовательный АГТП медленен на булевском уровне представления, это будет даже медленнее на аналоговом уровне представления.

3. АГТП для структур транзистора должен моделировать поведение двунаправленное, с тремя состояниями (см. главу 4). Модели неисправностей и АГТП алгоритмы существуют, но более сложны чем их двойники-логические вентили [105, 287, 538]. Хотя есть тестовые генераторы, которые могут работать на уровне транзистора [173, 212, 250, 386, 389], преобладающая тестовая методология продолжает полагаться на уровень константных неисправностей вентилей.

Таблица 7.2 показывает историю ускорения комбинационного АГТП. Ускорения выпуска продукции в таблице очень приблизительны, и должны трактоваться как оценки

порядка величины, из-за трудности в нормализации процессорных времен старших центральных процессоров, на которых были выполнены более ранние эксперименты алгоритмов, и из-за различий выполнения. Самые ранние и последние два элемента таблицы - не АГТП системы (см. раздел 7.6), но средние являются таковыми. АГТП система, используя случайную генерацию последовательности, имеет незаслуженное преимущество перед чистыми экспериментами выполнения алгоритма.

Table 7.2: History of algorithm speedups.

Algorithm	Estimated speedup over D-Algorithm (normalized to D-ALG CPU time)	Year
D-ALG [551]	1	1966
PODEM [258]	7	1981
FAN [229, 232, 233]	23	1983
TOPS [360]	292	1987
SOCRATES [576]	1574† ATPG System	1988
Waicukauski *et al.* [708]	2189† ATPG System	1990
EST [110, 253]	8765† ATPG System	1991
TRAN [122]	3005† ATPG System	1993
Recursive learning [376]	485	1995
Tafertshofer *et al.* [648]	25057	1997

Также, TRAN алгоритм и алгоритм Тафертсгофера (Tafertshofer) и других гораздо лучше, чем другие системы особенно для огромных схем, которые не отражены в этом сравнении. Мы видим, что усовершенствования появляются медленно, и что они только сохраняли темп в соответствии с Законом Мура (Moore) [477, 478].