**Проблемы функционального проектирования самотестируемых СБИС**

С.И. Родзин

**Введение**

Затраты на синтез теста интегральных схем приближенно оцениваются соотношением W=k\*Va , где 1.5<а<2.5, V - число вентилей схемы, k - коэффициент, зависящий от структуры проектируемой схемы. В сравнении с началом 70-х годов число вентилей выросло почти на четыре порядка, что означает рост затрат W на синтез теста, примерно, на восемь порядков! Необъяснимым остается вопрос, как вообще тестируется СБИС в ходе эксплуатации? Между тем решение проблемы вытекает непосредственно из приведенного выше соотношения. Поскольку в будущем вряд ли следует ожидать уменьшения степени интеграции СБИС, то сокращение затрат на тестирование можно достигнуть лишь через структуру проектируемой схемы. Это обстоятельство является фундаментом для развития работ в области проектирования самотестируемых СБИС. Причем термин самотестирование здесь употребляется применительно к СБИС, на кристалле которых размещаются средства генерации теста, сигнатурного анализа результатов и управления тестом [1].

**1. Постановка задачи**

В данной работе предлагается метод функционального проектирования самотестируемых СБИС. Идея метода состоит в том, что для синтеза теста используется внутренняя логика проектируемой схемы, которая управляет генератором теста (ГТ), работая в цепи обратной связи ГТ, что позволяет значительно сократить аппаратные затраты на проектирование ГТ. Эти затраты определяются прежде всего числом используемых в ГТ триггеров. И хотя, как известно, минимальное число состояний не обязательно приводит к уменьшению затрат при реализации схемы, однако предлагаемый метод проектирования ГТ направлен на минимизацию числа состояний и реализацию ГТ с возможно меньшим числом триггеров. Кроме того для синтеза теста при необходимости может привлекаться сигнатурный регистр(СР),что позволяет дополнительно сократить число элементов памяти. В этом случае при проектировании может оказаться, что ГТ либо вообще не содержит триггеров, либо содержит небольшое их число, а это упрощает кодирование состояний. Отметим также, что подобного рода подход к самотестированию позволяет через СР наблюдать состояние элементов памяти проектируемой схемы, при этом не требуется разрывать их обратные связи, что, в свою очередь, приводит к сокращению общей длины теста[2].

Таким образом, цель метода состоит в том, чтобы проектируемая схема тестировалась в своем рабочем состоянии, то есть чтобы функции схемы во время теста не изменялись. Поэтому сокращение числа состояний относится только к ГТ.

Для достижения поставленной цели предлагается решить во взаимосвязи две следующие задачи:

Синтез тестовой последовательности входных векторов для обнаружения заданного класса неисправностей проектируемой схемы, имея в виду подходящую реализацию ГТ и, используя для синтеза теста внутреннюю логику проектируемой схемы;

**Проектирование ГТ на кристалле.**

В качестве заданного класса неисправностей наряду с одиночными константными неисправностями на внешних и внутренних контактах схемы рассматривается также неисправности характерные для КМОП-схем, которые могут приводит к секвенциальным отношениям в проектируемой схеме[3].

Для определения тестовой последовательности проводится трансформация последовательной схемы в виртуальную комбинационную схему. С этой целью триггеры заменяются проводящими элементами, имеющими нулевую задержку и служащими для запоминания информации; обратные связи мысленно обрываются. В качестве метода получения тестовых наборов для виртуальной комбинационной схемы можно, например, применить D-алгоритм либо его модификации. При этом разрешается образовывать не полностью определенные тестовые векторы следующего вида: T={ (x,z)1,(x,z)2,...,(x,z)q }, где x - входные наборы, z - состояния схемы. Ясно, что для различных начальных состояний могут быть получены различные тестовые последовательности. Возникает вопрос: какая из последовательностей является наиболее подходящей для аппаратной реализации ГТ? Ответ на этот вопрос потребовал проведения дальнейших исследований по установлению взаимосвязи между реализацией ГТ и синтезированным тестом. В частности, удалось доказать, что число внутренних состояний ГТ зависит от числа переходов состояний d (d - это максимальное число всех переходов состояний во время прохода теста из заданного состояния, включая и повторяющиеся переходы), причем минимальное число триггеров для реализации ГТ равно [log2d]. Требования минимизации длины теста и минимизации числа триггеров ГТ не могут выполняться одновременно. Необходим компромисс. Современные СБИС работают с высокой тактовой частотой, что делает параметр длины теста некритичным и позволяет существенно снижать аппаратные затраты при проектировании ГТ для самотестируемых СБИС.

**2. Алгоритм проектирования генератора теста**

Проведенное исследование зависимости между реализацией ГТ и синтезируемой тестовой последовательностью позволяет сформулировать следующую процедуру проектирования ГТ:

– синтез тестовой последовательности, которая обеспечивает проверку всех неисправностей заданного класса и для которой величина d является минимальной (если существует несколько таких последовательностей, то выбирается наиболее короткая из них);

– проектирование ГТ с минимальным числом триггеров таким образом, что СБИС образует вместе с ГТ самотестируемую схему.

Рассмотрим подробнее алгоритм синтеза теста с помощью ГТ. Пусть заданно множество не полностью определенных тестовых векторов T в виде пар входов и состояний. Предлагается следующий алгоритм синтеза оптимальной по отношению к величине d тестовой последовательности для самотестируемой схемы, состоящей из ГТ и проектируемой схемы.

**Алгоритм**

Начало.

Выбираем начальное состояние в искомой тестовой последовательности {xF, zF}={(x1,z1), (x2,z2)...}, где функция перехода состояний определяется обычным автоматным отношение zt+1=Fz(xt,zt), t – момент времени. Образуем множество L1={z1}.

Полагаем i:=1.

Поиск в множестве T не полностью определенных пар(x,z)j таких, которые покрывают все вектора z уже принадлежащие множеству Li. Переход к п. 5. В противном случае , переход к п. 4.

Образуем с помощью функции перехода состояний Fz множество Li+1, содержащее все те состояния , которые достигаются из состояний zε Li. Полагаем i:=i+1 и переходим к п. 3.

Образуем из Li множество Li+1, содержащее все состояния установленные в п.3 парами (x,y)j. Состояния, включенные в L1,...,Li, описывают часть последовательности (z1,...zi+1)j,z1εL1,...,zi+1εLi+1, удовлетворяющей отношению zt+1=Fz(xt,zt), и поэтому они могут прибавляться к искомой последовательности zF.

Подсчитываем частоту вхождения каждого из состояний в последовательность zF и в прибавляемую часть последовательности (z1,...,zi+1)j, а также определяем для каждого состояния величину d.

Прибавляем к последовательности zF те части последовательностей, установленных в п.п.2-5 алгоритма, которые имеют минимальное значение d. Удаляем из zF состояния, которые покрываются парами из исходного множества T. Если существует несколько последовательностей с одинаковой минимальной величиной d, то выбираем любую из них.

Образуем новое множество L1, в которое входит последнее из включенных в zF состояний. Если в множестве T остались непомеченные пары, то переход к п.2, в противном случае, переход к п.9.

Конец.

Результирующая последовательность {xF,zF} является тестом для проектируемой схемы.

Необходимо также отметить, что вместо случайного выбора последовательностей с минимальным d в п.7 алгоритма можно, например, набирать из T вектора с меньшим числом неопределенных состояний.

Исследуем оценку сложности приведенного алгоритма в зависимости от числа q пар, содержащихся в множестве T. Так при поиске пар (x,z) в п.3 алгоритма требуется выполнить q операций сравнения. Если покрытие не получается, то идет перепроверка состояний из множества L2.

Если обозначить мощность множества входных векторов X через M, то L2 содержит максимум M элементов, L3-M2 и т.д. В общем случае, считаем, что множество Li содержит максимум Mi-1 элементов. Чтобы прибавить вектор z пары (x,z)jεT, требуется выполнить максимум q\*Mi-1 операций сравнения, исходя из некоторого начального состояния. Прибавление в дальнейшем потребует не более, чем (q-1)\*Mi-1 операций сравнения. Для определения последовательности, содержащей все q векторов из T, необходимо q-раз выполнить цикл п.п.2-7 алгоритма. Следовательно, общая оценка сложности алгоритма имеет порядок O(q2). Множество T при этом является исходным для работы алгоритма. Сомножитель Mi-1 зависит от функций, реализуемых проектируемой схемой, и от исходного множества пар тестовых векторов T. Общая оценка Mi-1 в этой связи затруднительна.

Выше отмечалось, что аппаратные затраты на реализацию ГТ можно сокращать и дальше, если использовать для синтеза теста не только внутреннюю логику проектируемой схемы, но и СР. Чтобы решить эту задачу вполне достаточно установленной ранее взаимосвязи между реализацией ГТ и синтезируемым тестом. Там триггер был необходим всегда, если тестовый вектор на входах проектируемой схемы не определялся однозначно различными состояниями. В связи с тем, что нет принципиальной разницы в том находится ли этот триггер в ГТ или же в СР, можно сократить число триггеров в ГТ путем использования СР для синтеза теста. При этом однако возникает вопрос: не приводит ли использование СР для синтеза теста к ограничению его способности выполнять свою основную функцию - сигнатурную оценку результатов тестирования? Чтобы ответить на это вопрос, приведем следующие рассуждения. До тех пор, пока неисправность не приводит к искажению последовательности состояний в СР и проектируемой схеме, самотестируемая СБИС выполняет синтезированную входную последовательность xF. Если некоторая неисправность приводит к искажению xF, то это с большой вероятностью ведет к искажению состояния СР и входной последовательности проектируемой схемы (напомним, что на входы проектируемой схемы подаются выходные сигналы ГТ). Между тем, известно, что вероятность маскирования неисправностей в СР при удачном выборе функций обратной связи СР является величиной независимой от числа этих искажений. В этой связи весьма проблематично ожидать каких-то ограничений в способности СР проводить оценку результатов тестирования.

**Практическая реализация метода самотестирования**

Ниже приводятся два прикладных примера реализации предлагаемой концепции самотестирования.

Так реализация ГТ для программируемого тактового генератора на три входа тактирующего микропроцессорные устройства была спроектирована в виде программируемой логической матрицы (ПЛМ). Длина теста для обнаружения всех одиночных константных неисправностей оказалась равной 27 наборам. Схема ГТ ( без использования для синтеза теста СР) содержит два триггера и 17 вентилей. Если сравнить затраты на реализацию тактового генератора и ГТ в пересчете на число вентилей и учесть еще затраты на хранение теста в памяти, то получается не совсем удовлетворительное соотношение. Однако затраты на ГТ сокращаются при использовании для генерации теста СР. В этом случае ГТ реализуется в виде чисто комбинационной схемы на 7 вентилей и отношение затрат на ГТ и тактовые генератор равно примерно 13%. Преимущество ПЛМ-реализации ГТ состоит в том, что комбинационные схемы в этом случае проектируются особенно просто, хотя при этом требуется несколько большая по сравнению с обычным проектированием площадь кристалла.

Другим примером является реализация ГТ для 32-разрядного секционного процессора, который имеет 13 управляющих входов, 32 входа данных, 32 выхода и 28 триггеров. Тестовая последовательность была определена упрощенным способом путем попарно-параллельного построения тестов для отдельных секций и последующей их склейки. Длина теста оказалась равной 39. Соответствующая ПЛМ-реализация ГТ содержит один триггер, а общее число термов равно 55. Для управления ГТ используются два триггера процессора и 8 триггеров 32-разрядного СР, так как большинство секций тестируются попарно-параллельно и триггеры СР содержат одинаковую информацию. Затраты на ГТ составили около 10%.

Выше отмечалось, что при реализации СБИС по КМОП-технологии, наряду с неисправностями константного типа имеют место характерные для КМОП-схем St-open неисправности, приводящие в некоторых случаях к секвенциальным отношениям к схеме. В этом случае вместо одного тестового вектора для каждой j-й St-open неисправности необходимо синтезировать целый блок тестовых пар (x,z), причем первая пара обычно служит для инициализации, а остальные - для очувствления неисправности. Что касается алгоритма синтеза тестовой последовательности для ГТ, то он является расширением ранее приведенного алгоритма. В частности, отличие состоит в том, что в п.п.4 и 5 алгоритма множество Li+1 содержит все те состояния, которые достигаются из состояния zεLi не только за один, но и за несколько переходов, а в п.8 необходимо также учесть это обстоятельство при образовании нового множества Li. Общая оценка сложности алгоритма становится равной O(q3), возрастает и длина теста. Так, например, длина теста для упомянутого ранее тактового генератора становится равной 66 наборам, а ГТ для него содержит 17 вентилей. Похожий рост затрат наблюдается и для самотестируемой КМОП-схемы 32-разрядного секционного процессора, хотя сам метод самотестирования не зависит от технологии, что является его несомненным преимуществом.

**Список литературы**

1.Электроника СБИС. Проектирование микроструктур: Пер. с анг./ Под ред. Н. Айнспрука.- М.: Мир, 1989. – 256с.ил.

2.Мелихов А.Н., Родзин С.И. Проектирование генератора тестов для самотестируемых СБИС. – В трудах 12-й международной конференции: Отказоустойчивые системы и диагностика. – ЧСФР, Прага, 1989.

3.Курейчик В.М., Родзин С.И. Контролепригодное проектирование и самотестирование СБИС : проблемы и перспективы. – М.: Радио и связь, 1994. – 176с.: ил.