РОССИЙСКАЯ АКАДЕМИЯ НАУК

АВТОМАТИКА И ТЕЛЕМЕХАНИКА

(ОТДЕЛЬНЫЙ ОТТИСК)

МОСКВА

Автоматика и телемеханика, № 9, 2010

Техническая диагностика

© 2010 г. **Л.А. ЗОЛОТОРЕВИЧ, канд. техн. наук, А.В. ИЛЬИНКОВА**
(Белорусский государственный университет, Минск)

РАЗРАБОТКА ТЕСТОВ ДЛЯ АНАЛИЗА КОНТРОЛЕПРИГОДНОСТИ СБИС НА ВЕРХНИХ УРОВНЯХ ПРОЕКТИРОВАНИЯ

Анализируется состояние проблемы контроля сверхбольших интегральных схем (СБИС). Рассматриваются задачи направленного построения тестов контроля на системном уровне представления объекта или уровне межрегистровых передач (RTL) на языке VHDL. Класс функциональных неисправностей, рассматриваемых при направленном построении теста, соответствует неисправностям константного типа реализаций СБИС на элементах соответствующих библиотек проектирования. Предлагается метод направленного построения теста, который позволяет на ранних этапах проектирования анализировать зависимость контролепригодности от применяемых технологических библиотек проектирования.

1. Введение

Требования к надежности цифровых систем постоянно возрастают не только в областях, в которых отказ может привести к катастрофическим событиям, но и во всех других применениях. Одна из самых важных компонент обеспечения высокой надежности электронных систем – способность определить наличие или отсутствие ошибки функционирования системы [1]. Проблема построения тестов на всем интервале развития интегральной схемотехники является одной из наукоемких проблем, которые до настоящего времени не получили эффективного ни теоретического, ни практического решения. Задача построения тестов принадлежит к классу NP-трудных проблем [2]. Моделирование неисправностей и построение тестов контроля цифровых устройств исторически выполнялось на вентильном уровне. Однако размерность данной задачи применительно к проектам современных СБИС в целом на уровне вентильного представления ограничивает возможность ее эффективного решения. Стремление анализировать контролепригодность проекта на самых начальных этапах проектирования цифровой системы, желательно еще до проведения декомпозиции проекта с целью определения программно реализуемой части проекта, положило начало исследованиям по иерархическому построению тестов – HTG (hierarchical test generation) [3].

В разделе 2 дается анализ состояния проблемы контроля СБИС и особенности контроля «систем на кристалле». В разделе 3 рассматривается математическая платформа для разработки модели объекта на системном уровне или уровне RTL. В разделе 4 предлагается метод построения иерархических тестов контроля. Метод основан на представлении объекта на системном или RTL уровне в виде управляющей и операционной частей с помощью диаграмм принятия решений как математической платформы для решения задачи. В разделе 5 рассматривается пример построения

моделей функциональных неисправностей, соответствующих реальным неисправностям объекта, реализуемого в разных технологических библиотеках.

2. Состояние проблемы контроля цифровых устройств и систем

Существующие в настоящее время системы контроля основаны на следующих подходах:

• Функциональный контроль, который предполагает исследование устройства на рабочей скорости и проводится в течение рабочего функционирования устройства в режиме on-line.

• Тестовое диагностирование объекта в режиме off-line на основе разработанных тестов контроля в различных классах неисправностей (константного типа, обрывы, замыкания и др.) в режиме приостановки эксплуатации объекта и применении внешнего диагностического оборудования для тестирования систем – ATE (Automatic Test Equipment).

Особое внимание в последнее время уделяется проблеме контроля неисправностей типа «задержка», которые обусловлены ошибками проектирования, дестабилизирующими воздействиями внешней среды, конструктивно-технологическими особенностями, влияющими на физическую длину межсоединений и др. Применяемые методы контроля неисправностей задержки основываются как на методах временного моделирования, так и на статическом временном анализе. Особенностью методов статического временного анализа цифровых структур является то, что они не предполагают вычисления реализуемых функций.

Появившаяся в начале века и быстро развивающаяся тенденция проектирования систем на кристалле SoC (System-on-Chip), стремительный переход в субмикронный диапазон сокращает область применения внешнего диагностического оборудования для тестирования систем (ATE) в связи с применением методов контролепригодного проектирования – DFT (design for testability) и средств встроенного самотестирования – BIST (built-in self-test). Такой подход к обеспечению контроля не требует внешнего оборудования, которое при этом интегрируется в состав активной инфраструктуры непосредственно на кристалле. Один из методов реализации идеи BIST – применение методов компактного тестирования, в частности сигнатурного анализа, основанного на применении сдвигового регистра с линейной обратной связью – LFSR (linear feedback shift register), который может быть и генератором псевдослучайной последовательности.

Применение методов встроенного самотестирования оказывается полезным на всех этапах жизненного цикла изделия. При трудном доступе к некоторым блокам системы со стороны внешних входов (проблема контроля глубоких схем), при решении задачи сохранения интеллектуальной собственности (intellectual property) при повторном применении IP-блоков самотестирование является основным средством контроля. Существенным недостатком метода является то, что не гарантируется полнота проверки объекта. Поэтому данный метод хорошо совмещать с применением заранее разработанной тестовой последовательности, которая включается в последовательность, применяемую при сигнатурном анализе. Таким образом, проблема построения тестов остается актуальной наряду с применением разных методов обеспечения контролепригодности и контроля объектов.

Ниже рассматривается задача построения тестов и анализа контролепригодности объектов на самых ранних этапах проектирования. Предлагаемый метод позволяет уменьшить размерность задачи построения тестов контроля СБИС, так как рассматривается описание СБИС на языке VHDL на уровне межрегистровых передач, которое намного меньше, чем описание структуры объекта на логическом уровне.

В отличие от известных классов неисправностей, заимствованных из области тестирования программных средств, рассматриваемые неисправности соответствуют неисправностям реальных физических объектов.

3. Представление объекта на поведенческом уровне и уровне RTL

При построении тестов на основе структурного представления объекта на практике применяются структурные методы моделирования и модели неисправностей чаще всего константного типа. При построении тестов в расширенном классе неисправностей, свойственных МОП-технологии, применяются переключательные модели цифровых устройств [4] и неисправности типа ПЗТ («постоянно-закрытый транзистор»). Задача построения тестов на уровне системного представления объекта, а также на уровне RTL в отличие от функционально-логического структурного представления объекта требует разработки математической платформы для построения модели, эффективной для работы с программным кодом (в рассматриваемом случае с кодом на языке VHDL).

С целью выбора математической платформы для описания цифровой системы на поведенческом уровне или уровне RTL рассмотрим различные формы диаграмм принятия решений DD (Decision Diagram), которые в последнее время широко применяются при решении задач проектирования и построения тестов контроля объектов цифровой электроники на структурном уровне символьными методами. Известны работы по представлению систем булевых функций в виде двоичных диаграмм принятия решений BDD (Binary Decision Diagrams), по их редуцированию [5]. На уровне BDD эффективно решаются задачи построения тестов цифровых структур, заданных булевыми функциями, в классе неисправностей константного типа входных переменных [6]. Известны представления объектов в виде структурно синтезируемых бинарных диаграмм принятия решений – SSBDD (Structurally Synthesized Binary Decision Diagram) [7]. В SSBDD в отличие от BDD заложена информация о структуре схемы, реализующей соответствующую функцию. Данная особенность SSBDD является существенной, так как позволяет расширить класс рассматриваемых неисправностей при построении теста на ее основе по сравнению с BDD. На рис. 1 приведен пример построения SSBDD по вентильному представлению объек-

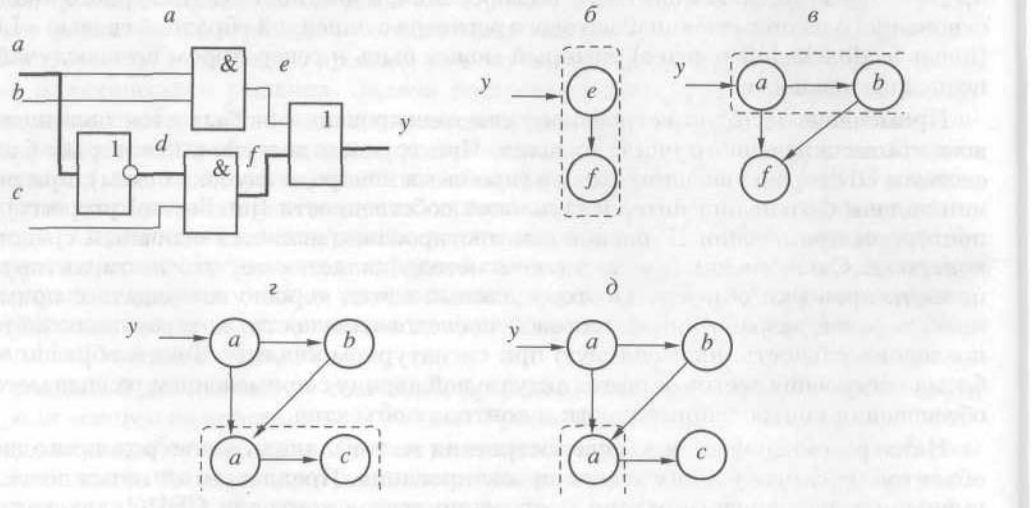

Рис. 1. Фрагмент структуры и соответствующее SSBDD представление.

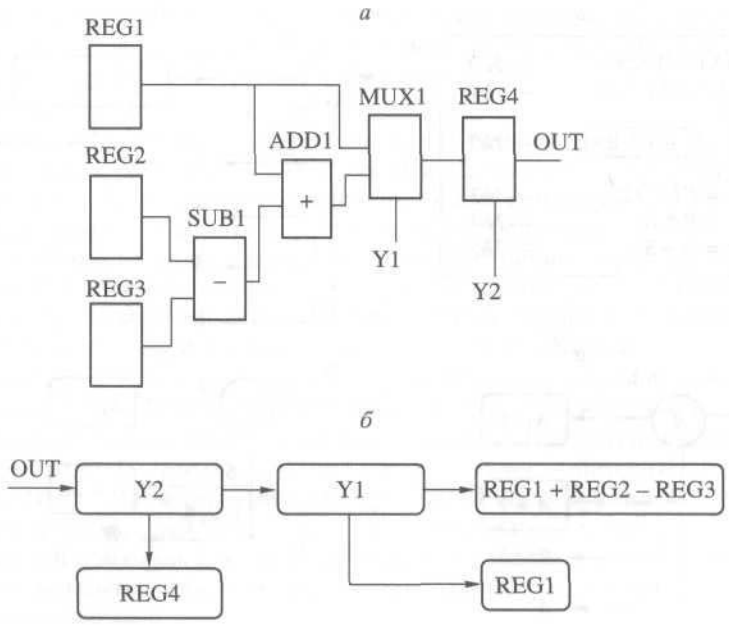

Рис. 2. Цифровое устройство *а*) на уровне RTL; *б*) SSBDD представление устройства.

та [6]. В соответствии с соглашением стрелка вправо от узла соответствует логической 1 и стрелка вниз – логическому 0. В отличие от BDD терминальные узлы, соответствующие логическим состояниям 0 и 1, на SSBDD опущены. Построение SSBDD начинается от вентиля, который является источником выходного сигнала. Затем каждый элемент заменяется его функцией в виде BDD. На рис. 1,*б* приведено SSBDD для элемента ИЛИ. На рис. 1,*б*–*д* показан процесс последовательного построения SSBDD фрагмента структуры, приведенной на рис. 1,*а*. На каждом этапе один элемент заменяется его представлением в виде BDD. На SSBDD можно проследить непосредственные отношения между узлами и путями сигнала в соответствующей схеме (рис. 1,*д*).

Поясним порядок построения SSBDD для элемента ИЛИ (рис. 1,*б*). Стрелка вправо, соответствующая выходной переменной y, идет к узлу, относящемуся к линии e (или к линии f), а от узла, связанного с линией e, идет стрелка вниз к узлу, соответствующему линии f, что следует понимать так: $y = 1$ при $e = 1$, а также $y = 1$ при $e = 0$, $f = 1$; $y = 0$ при $e = f = 0$. Следует при этом иметь в виду, что если от некоторого узла на диаграмме нет стрелки вправо (вниз), то по умолчанию подразумевается, что она имеется и идет к терминальному узлу, равному 1 (0). На рис. 1,*в* добавлено SSBDD представление узла e, соответствующего выходу элемента И. На рисунке штриховыми стрелками показаны замены всех узлов их SSBDD – представлениями соответствующих вентилей ($\overline{a}$ – это есть SSBDD – представление инвертора с входом a). SSBDD – представление применяется [6] при разработке методов построения тестов в классе неисправностей константного типа как на входных, так и на внутренних линиях структуры на функционально-логическом уровне идентификации.

Рассмотрим возможность описания объекта на уровне RTL в виде SSBDD. На рис. 2,*а* приведена схема некоторого объекта на уровне межрегистровых передач [7]. Потоками информации в схеме управляют переменные Y1 и Y2. Если Y1 и Y2

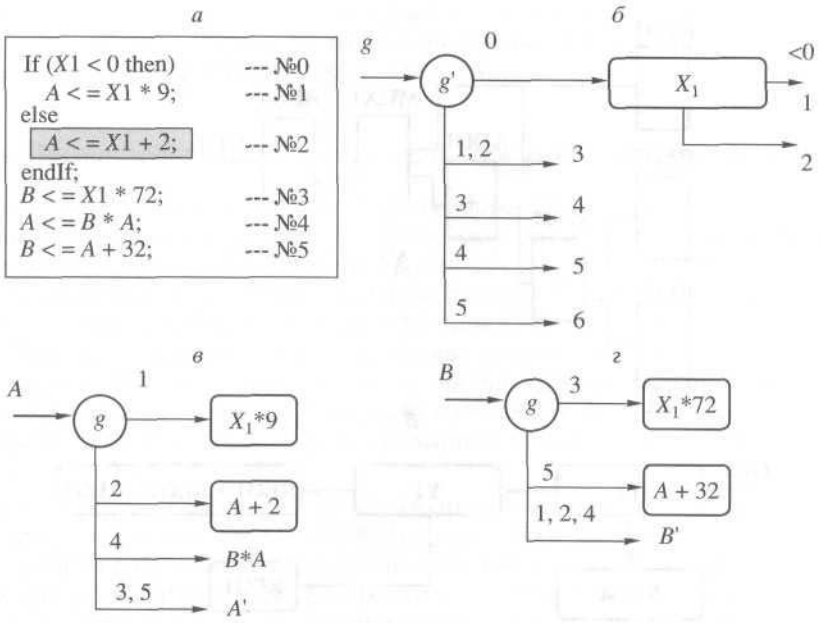

Рис. 3. DD потока управления и потоков данных для фрагмента программного кода.

являются булевыми переменными, то можно представить эту схему в виде SSBDD (рис. 2,*б*).

Следует отметить свойство BDD и SSBDD – все переменные принимают булевы значения, что ограничивает их применение рассмотрением уровня вентильной структуры. При переходе на уровень межрегистровых передач значения переменных могут быть различными (целочисленные, многозначные и т.д.). Поэтому представить объект на уровне RTL в виде SSBDD, как это показано на рис. 2, оказывается чаще всего невозможным.

Как математическую платформу для построения поведенческой модели цифровой системы, представленной в виде управляющей и операционной частей, будем использовать DD потоков данных и DD потока управления, которые строятся на основе VHDL-описания объекта. Программный код такого описания содержит операторы языка VHDL, составляющие его синтезируемое подмножество.

Пути данных распространяются через функциональные блоки, которые могут представлять собой комбинационные или последовательностные схемы. При описании системы каждая выходная и внутренняя переменная описывается в виде DD потока данных. Нетерминальные вершины DD потока данных содержат значения сигналов управления, а оконечные, терминальные, узлы описывают соответствующие данные, т.е. состояния первичных входов, регистров, операции, константы. Управляющая часть описывается конечным автоматом, таблица переходов которого описывается единственной DD. Нетерминальные узлы представляют текущее состояние и входные условия (состояния) управляющей части, а терминальные – следующие логические состояния, которые управляют распространением и обработкой информации в системе.

Описание объекта на языке VHDL на RTL-уровне не несет информации о структурном наполнении объекта, тем не менее на этапе структурного синтеза каждый фрагмент описания будет реализован в виде некоторой структуры в заданном элементном базисе.

Рассмотрим фрагмент поведенческого описания проекта на языке VHDL, приведенный на рис. 3, и его представление в виде DD [7]. На рис. 3,*а* приведен код программы, на рис. 3,*б* – DD потока управления, на рис. 3,*в*,*г* – DD потока данных для переменных A и B.

В приведенном фрагменте, когда входной переменной является X1, внутренними и/или выходными переменными являются A и B, через g' обозначено предыдущее состояние системы. Под состоянием системы будем понимать состояние всех переменных программного кода. Предыдущее состояние системы – это состояние системы до момента переключения в некоторое новое состояние, определяемое DD потока управления и DD потоков данных (рис. 3). Если предыдущее состояние системы – это состояние перед выполнением оператора 0, то следующее состояние g при $X1 \geq 0$ будет определяться выполнением оператора № 2: $A <= X1 + 2$.

В данном случае DD потока управления представляет собой ориентированный граф без петель, в котором нетерминальные вершины представляют логические состояния (условия), терминальные вершины – операции (действия), а связи указывают подмножество узлов (состояний), для которых будет выбираться узел преемника. В отличие от BDD переменные нетерминальных узлов могут быть любыми булевыми (описание флагов, логические условия и т.д.) или целыми числами (описание команд, областей контроля и т.д.). Переменные терминальных узлов помечены константами, переменными (булево или целое число) или выражениями для вычисления значения целого числа.

На поведенческом уровне DD потока данных генерируется для каждой внутренней и выходной переменной проекта. Такая DD потока данных имеет столько ветвей, сколько раз переменная появляется слева в операторе назначения. Кроме того, генерируется единственная DD потока управления, которая описывает последовательность условий (состояний) активизации пути. Диаграммы, полученные на основе программного кода на языке VHDL, будем использовать как вычислительную модель системы при иерархическом построении тестов.

4. Направленное построение тестов контроля на основе DD потока данных и DD потока управления

Рассмотрим задачу построения теста в следующей постановке. Цифровая система представлена на RTL-уровне или уровне описания поведения. Структурная реализация системы отсутствует, но известны технологические библиотеки элементов, которые доступны на этапе его синтеза. Необходимо построить тест контроля объекта, при этом функциональные модели рассматриваемых неисправностей должны соответствовать моделям неисправностей константного типа, свойственных реальному физическому объекту. Исходными данными для решения задачи является VHDL-описание объекта. По данному описанию необходимо построить DD потоков данных и DD потока управления, которые затем необходимо перевести в множество предикатов изменения состояния системы, и затем с использованием предикатов будут определяться ограничения при активизации и распространении неисправностей.

В качестве рабочей примем научную гипотезу Рота [8], получившую большое развитие при направленном построении тестов контроля логических структур: для построения теста необходимо очувствить (активизировать) неисправность, распространить влияние неисправности хотя бы к одному выходу и доопределить значения входных переменных. В отличие от неисправности константного типа, рассматриваемой Ротом в случае идентификации объекта на структурном уровне, в данном случае будет рассматриваться ошибочный оператор некоторого программного кода, т.е. оператор-мутант, который описывает на функциональном уровне поведение реального объекта с некоторой неисправностью константного типа. Примеры построения подобных операторов приведены в разделе 5.

Положим, что на рис. 3 система находится в состоянии g, которое соответствует оператору № 2 A <= X1 + 2. Чтобы проверить правильность выполнения данного оператора (выделен на рис. 3,*а*), надо вначале привести систему к состоянию g перед выполнением оператора № 2. Условиями активизации являются $g' = 0$, а также X1 >= 0. Для распространения эффекта неисправного оператора на выходы системы необходимо обеспечить условия транспортировки:

1) B $\neq 0$ (выведено из выражения A <= B * A);

2) X1 $\neq 0$ (выведено из выражения B <= X1 * 72 $\neq 0$).

При невыполнении данных условий переменная A будет всегда иметь значение 0 и неисправность рассматриваемого оператора обнаружена быть не может.

Приведем общий алгоритм построения теста на уровне RTL или на уровне поведенческого представления объекта. Предполагаем, что функциональные неисправности для каждого оператора (операторы-мутанты) получены. Методика получения операторов-мутантов для конструкции if-then-else приведена в главе 5.

Общий алгоритм построения теста:

1. По VHDL-описанию объекта строим DD потоков данных и DD потока управления (используется синтезируемое подмножество языка VHDL).

2. Выбираем очередной оператор программного кода.

3. Выбираем очередную функциональную неисправность оператора.

4. Извлекаем функциональные ограничения для очувствления данной неисправности.

5. Извлекаем функциональные ограничения для распространения эффекта неисправности к выходам системы.

6. Выполняем разрешение ограничений.

7. Если ограничения удовлетворены, то сравниваем реакции системы по выполнению программного VHDL-кода с исходным оператором и с оператором-мутантом. Если реакции различные, то фиксируем найденный тест контроля определенной группы неисправностей.

8. В противном случае, переходим к п. 3 или к п. 2, или на выход при окончании процесса или по времени.

Иерархический подход к генерации тестов позволяет на самом высоком уровне анализировать способность системы к тестированию, уменьшает размерность задачи. При этом можно сравнивать контролепригодность объекта в зависимости от используемой технологической элементной базы. Можно генерировать тесты для различных возможных проектов реализаций (различные технологические библиотеки) и выбирать решение, которое является лучшим с точки зрения способности к тестированию.

5. Построение моделей функциональных неисправностей

Известно, что средства синтеза объектов, описанных на языке VHDL, ориентированы на определенное подмножество операторов языка, так называемое синтезируемое подмножество [9]. К тому же они ориентированы на применение соответствующих технологических библиотек компонентов, так что каждая конструкция операторов языка из его синтезируемого подмножества реализуется определенной структурой. Поставим задачу построения и моделирования функциональных неисправностей цифровых объектов комбинационного типа на поведенческом или RTL уровне их описания таким образом, чтобы они соответствовали неисправностям их физических реализаций. Для иллюстрации подхода используем одну из часто при-

меняемых конструкций языка VHDL, к примеру if-then-else для выбора одного из двух входных сигналов:

```
if Y = '0' then
   Z <= X1;
else
   Z <= X2;
end if
```

и для выбора одного из четырех входных сигналов:

```
if Y = '00' then
   Z <= X1;
elsif Y = '10' then
   Z <= X2;
elsif Y = '01' then
   Z <= X3;
elsif Y = '11' then
   Z <= X4;
end if
```

и рассмотрим ее некоторые аппаратные реализации на основе мультиплексора. На рис. 4 приведена реализация функции на основе дизъюнктивной нормальной формы (ДНФ) (M1–M3, Z – имена вентилей структуры), на рис. 5 та же функция реализована на основе конъюнктивной нормальной формы (КНФ) (целесообразно рассмотреть все известные реализации рассматриваемой композиции операторов).

Сгенерируем тесты контроля неисправностей константного типа для обеих реализаций схем (моделируемые неисправности пронумерованы на рисунках: нечетные но-

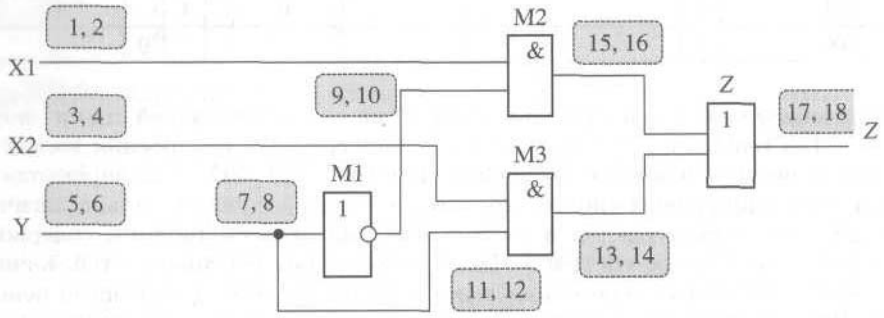

Рис. 4. ДНФ-реализация оператора if-then-else для выбора одного из двух сигналов.

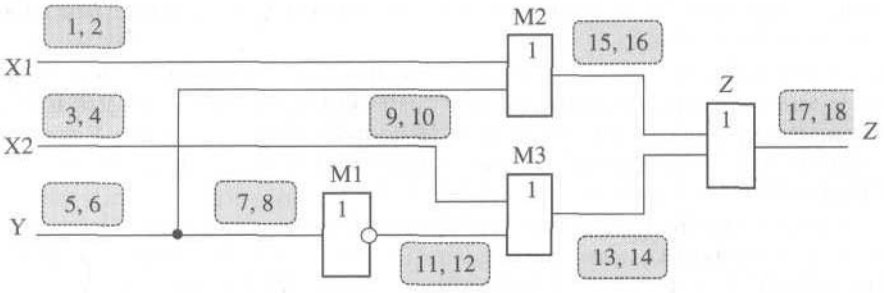

Рис. 5. КНФ-реализация оператора if-then-else для выбора одного из двух сигналов.

Таблица 1. Тесты и покрываемые неисправности константного типа

Неисправности →	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Тест-векторы ↓	ДНФ-реализация схемы																		
011	1			0		0						0		0				0	
010	0		1				1						1		1		1		1
101	0				1	1		1			1				1		1		1
100	1	0					0		0	0						0		0	
	КНФ-реализация схемы																		
100	1	0					0		0			0		0		0		0	
010	0		1				1				1						1		1
001	0				1	1		1					1		1				1
011	1			0		0				0				0		0		0	

Таблица 2. Группы эквивалентных неисправностей

Группы неисправностей →		1	2	3	4		5	6	7	8
Неисправности →	0	1, 8, 9, 15	2, 12	3, 11, 13	4, 7, 10		5	6	14, 16, 18	17
Тест-векторы ↓	ДНФ-реализация схемы									
011	1			0			0			0
010	0		1					1	1	
101	0				1		1		1	
100	1	0						0		0

мера обозначают неисправности типа "const 0" на соответствующей линии, четные – "const 1"). Для генерации тестов использовались средства построения тестов и моделирования неисправностей собственной системы VLSI_SIM моделирования и построения тестов цифровых систем, представленных на функционально-логическом уровне [10]. Результаты сведем в табл. 1. Первый столбец таблицы содержит тесты, второй – логическое состояние выхода схемы без неисправностей, начиная с 3-го столбца – логические состояния выхода схемы с соответствующими неисправностями. Во всех столбцах начиная с 3-го столбца логическое состояние указывается только в том случае, если оно не совпадает с состоянием выхода схемы без неисправности. Очевидно, табл. 1 можно значительно сократить за счет объединения одинаковых столбцов, соответствующих эквивалентным неисправностям. Объединяя эквивалентные неисправности, покрываемые соответствующими входными наборами, в группы, получаем табл. 2.

Анализируя табл. 2, можно выделить группы 1–4 неисправностей, которые покрываются только единственным входным набором. В то же время эти наборы покрывают и неисправности из групп 5–8. Неисправности, входящие в группы 5–8, можно исключить из дальнейшего рассмотрения, так как они будут обнаружены при тестировании неисправностей из групп 1–4.

Для неисправностей группы 1 по структурной схеме вычисляем функцию неисправности, т.е. функцию, которую реализует схема при внесении, предположим, неисправности № 1 (const 0 на первом входе конъюнктора с именем M2) (рис. 4): $Z = X2 \wedge Y$. Для всех других неисправностей группы № 1 функция неисправностей будет идентичной. Ниже приведены функции неисправностей 1–4:

Таблица 3. Функциональные неисправности

Группы	ДНФ-реализация		КНФ-реализация	

1	`if Y = '0' then`
	`Z <= 0;`
	`else`
	`Z <= X2;`
	`end if`
			`if Y = '0' then`
			`Z <= 0;`
			`else`
			`Z <= X2;`
			`end if`
				`if Y = '0' then`
				`Z <= X1 and X2;`
				`else`
				`Z <= X2;`
				`end if`

2	`if Y = '0' then`
	`Z <= 1;`
	`else`
	`Z <= X2;`
	`end if`
		`if Y = '0' then`
		`Z <= X1 or X2;`
		`else`
		`Z <= X2;`
		`end if`
			`if Y = '0' then`
			`Z <= 1;`
			`else`
			`Z <= X2;`
			`end if`	

3	`if Y = '0' then`
	`Z <= X1;`
	`else`
	`Z <=0;`
	`end if`
			`if Y = '0' then`
			`Z <= X1;`
			`else`
			`Z <= 0;`
			`end if`
				`if Y = '0' then`
				`Z <= X1;`
				`else`
				`Z <= X1 and X2;`
				`end if`

4	`if Y = '0' then`
	`Z <=X1;`
	`else`
	`Z <=1;`
	`end if`
		`if Y = '0' then`
		`Z <= X1;`
		`else`
		`Z <= X1 or X2;`
		`end if`
			`if Y = '0' then`
			`Z <= X1;`
			`else`
			`Z <= 1;`
			`end if`	

для ДНФ-реализации групп

$$\begin{aligned} &№\,2 \;-\; Z = \neg Y \vee X2 \wedge Y; \\ &\qquad\quad Z = X1 \wedge \neg Y \vee X2; \\ &№\,3 \;-\; Z = X1 \wedge \neg Y; \\ &№\,4 \;-\; Z = Y \vee X1 \wedge \neg Y; \\ &\qquad\quad Z = X1 \vee X2 \wedge Y; \end{aligned}$$

для КНФ-реализации для группы № 1 – $Z = X2 \wedge Y$;

$$\begin{aligned} &\qquad\quad Z = X1 \wedge X2 \vee X2 \wedge Y; \\ &№\,2 \;-\; Z = X2 \vee \neg Y; \\ &№\,3 \;-\; Z = X1 \wedge \neg Y; \\ &\qquad\quad Z = X1 \wedge X2 \vee X1 \wedge \neg Y; \\ &№\,4 \;-\; Z = X1 \vee Y. \end{aligned}$$

Проанализируем функцию неисправности $Z = X2 \wedge Y$. Данной функции соответствует следующая конструкция:

```
if Y = '0' then
   Z <= 0;
else
   Z <= X2;
end if.
```

Анализируя функции неисправностей для рассматриваемых групп построим множество моделей поведенческих (функциональных) неисправностей. В табл. 3 приведены все модели функциональных неисправностей для обеих реализаций схемы. Если удалить из 12 полученных моделей одинаковые и оставить по одной для каждой группы неисправностей, то из полученного множества моделей неисправностей можно выделить четыре поведенческие неисправности (табл. 4), обнаружение которых обеспечивает обнаружение всех неисправностей константного типа, соответствующих реальному объекту, представленному на вентильном уровне в виде ДНФ или КНФ реализаций. Аналогично можно построить функциональные тесты (табл. 5)

Таблица 4. Неисправные модификации оператора if-then-else (один из двух)

Неисправности типа «И»		Неисправности типа «ИЛИ»	

`if Y = '0'`
`then`
`Z <= X1 and X2;`
`else`
`Z <= X2;`
`end if`
	`if Y = '0'`
	`then`
	`Z <= X1;`
	`else`
	`Z <= X1 and X2;`
	`end if`
		`if Y = '0'`
		`then`
		`Z <= X1 or X2;`
		`else`
		`Z <= X2;`
		`end if`
			`if Y = '0'`
			`then`
			`Z <= X1;`
			`else`
			`Z <= X1 or X2;`
			`end if`

Таблица 5. Неисправные модификации оператора if-then-else (один из четырех)

Неисправности типа «И»		Неисправности типа «ИЛИ»	

`if Y='00'then`
`Z <= X1 or X2;`
`elsif Y='10'then`
`Z <= X2;`
`elsif Y='01'then`
`Z <= X3;`
`elsif Y='11'then`
`Z <= X4;`
`end if`
	`if Y='00'then`
	`Z <= X1 or X3;`
	`elsif Y='10'then`
	`Z <= X2;`
	`elsif Y='01'then`
	`Z <= X3;`
	`elsif Y='11' then`
	`Z <= X4;`
	`end if`
		`if Y='00'then`
		`Z <= X1;`
		`elsif Y='10'then`
		`Z <= X1 or X2;`
		`elsif Y='01'then`
		`Z <= X3;`
		`elsif Y='11' then`
		`Z <= X4;`
		`end if`
			`if Y='00'then`
			`Z <= X1;`
			`elsif Y='10'then`
			`Z <= X2 or X4;`
			`elsif Y='01'then`
			`Z <= X3;`
			`elsif Y='11'then`
			`Z <= X4;`
			`end if`

`if Y='00'then`
`Z <= X1;`
`elsif Y='10'then`
`Z <= X2;`
`elsif Y='01'then`
`Z <= X1 or X3;`
`elsif Y='11'then`
`Z <= X4;`
`end if`
	`if Y='00'then`
	`Z <= X1;`
	`elsif Y='10'then`
	`Z <= X2;`
	`elsif Y='01'then`
	`Z <= X3 or X4;`
	`elsif Y ='11'then`
	`Z <= X4;`
	`end if`
		`if Y='00'then`
		`Z <= X1;`
		`elsif Y='10'then`
		`Z <= X2;`
		`elsif Y='01'then`
		`Z <= X3;`
		`elsif Y='11' then`
		`Z <= X2 or X4;`
		`end if`
			`if Y='00'then`
			`Z <= X1;`
			`elsif Y='10'then`
			`Z <= X2;`
			`elsif Y='01'then`
			`Z <= X3;`
			`elsif Y='11'then`
			`Z <= X3 or X4;`
			`end if`

`if Y='00'then`
`Z <= 0;`
`elsif Y='10'then`
`Z <= X2;`
`elsif Y='01'then`
`Z <= X3;`
`elsif Y='11'then`
`Z <= X4;`
`end if`
	`if Y='00'then`
	`Z <= X1;`
	`elsif Y='10'then`
	`Z <= 0;`
	`elsif Y='01'then`
	`Z <= X3;`
	`elsif Y='11' then`
	`Z <= X4;`
	`end if`
		`if Y='00'then`
		`Z <= X1;`
		`elsif Y='10'then`
		`Z <= X2;`
		`elsif Y='01'then`
		`Z <= 0;`
		`elsif Y='11' then`
		`Z <= X4;`
		`end if`
			`if Y='00'then`
			`Z <= X1;`
			`elsif Y='10'then`
			`Z <= X2;`
			`elsif Y='01'then`
			`Z <= X3;`
			`elsif Y='11'then`
			`Z <= 0;`
			`end if`

для конструкции упомянутых выше операторов if-then-else, реализованной в виде четырехканального мультиплексора [11].

6. Выводы

Предложен общий подход к направленному построению теста контроля цифровой системы при ее описании на поведенческом уровне или уровне межрегистровых передач. В основе метода использована идея активизации многомерного пути, которая получила теоретическое и практическое решение при направленном построении теста на уровне структурного представления объекта. Особенность предложенного подхода заключается в использовании моделей реальных неисправностей объекта при формировании множества рассматриваемых функциональных неисправностей.

СПИСОК ЛИТЕРАТУРЫ

1. Основы технической диагностики / Под ред. П.П. Пархоменко. М.: Энергия, 1976.
2. *Ibarra O.H., Sahni S.* Polynomially complete fault detection problems // IEEE Trans. Comput. 1975. V. C-24. № 3. P. 242–249.
3. *Murray B.T., Hayes J.P.* Hierarchical test generation using precomputed tests for modules // Int. Test Conf. 1988. P. 221–229.
4. *Золоторевич Л.А., Юхневич Д.И.* Переключательное квазистатическое моделирование СБИС. Сравнение методов по точности моделей // АиТ. 1998. № 9. С. 130–141.
5. *Бибило П.Н., Леончик П.В.* Оптимизация многоуровневых представлений систем булевых функций на основе диаграмм двоичного выбора // 6-я междунар. конф. «Автоматизация проектирования дискретных систем». 14–15 ноября 2007. С. 162–169.
6. *Ubar R.* Test synthesis with alternative graphs // IEEE Design Test Comput. 1996. V. 13. № 1. P. 48–57.
7. *Jervan G., Peng Z., Ubar R.* Test cost minimization for hybrid BIST // IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Syst. (DFT'00). 2000. P. 283–291.
8. *Roth J.P.* Diagnosis of automata failures: a calculus and a method // IEEE Trans. Comput. 1966. V. 15. № 7. P. 278–291.
9. IEEE Standard for VHDL register transfer level (RTL) Synthesis // IEEE Std 1076.6-1999. 1999. P. 1–73.
10. *Золоторевич Л.А., Сидоренко О.М., Юхневич Д.И.* Моделирование и генерация тестов для изделий микроэлектроники и устройств цифровой электроники // Матер. науч.-практ. конф. «Проблемы разработки, внедрения и эффективного использования сквозных САПР в Республике Беларусь». Минск, 2000.
11. *Золоторевич Л.А.* Моделирование неисправностей СБИС на поведенческом уровне на языке VHDL // Информатика. 2005. № 3. С. 135–144.

Статья представлена к публикации членом редколлегии П.П. Пархоменко.

Поступила в редакцию 01.07.2009