УДК 681.325:519.713

СТРАТЕГИИ ПОСТРОЕНИЯ ТЕСТОВ ДЛЯ ЦИФРОВЫХ СИСТЕМ

МД. МЕХЕДИ МАСУД

Предлагаются три стратегии генерации тестов для цифровых проектов относительно класса одиночных константных неисправностей (ОКН). Объект тестирования представлен в виде структуры функциональных элементов, включающих триггеры, который задается в форме булевых уравнений на языке VHDL. Выбор стратегии зависит от размерности и структурной сложности объекта тестирования. Используются методы декомпозиции схемы на функционально независимые фрагменты, уменьшающие время построения тестов.

1. Введение

Актуальность работы определяется необходимостью выполнения верификации цифровой системы на стадии ее описания с помощью языка VHDL. Для этого следует генерировать тесты проверки соответствия исходной спецификации и модели, представленной в VHDL-среде. Учитывая, что сложность реализации цифровой системы достигает сотен тысяч и миллионов вентилей, вполне понятной является сложность проблемы генерации представительной выборки входных последовательностей, обеспечивающей верификацию проекта.

Для реализации цифрового проекта используются программируемые логические интегральные схемы (ПЛИС) – Field Programable Gate Array (FPGA), которые достойно конкурируют на рынке микроэлектронных технологий с базовыми матричными кристаллами, сигнальными процессорами. Такой успех определяется: использованием субмикронных технологий изготовления кристаллов; применением Hardware-Software Cooperation-Design; минимальным временем проектирования цифровой системы (4-5 месяцев); высоким быстродействием выполнения операций (до 500Мгц); большой степенью интеграции элементов на кристалле (до 3,5 млн.).

Вместе с преимуществами ПЛИС существуют и проблемы создания диагностического обеспечения для них: тестирование цифровых проектов большой размерности, вентильного, функционального алгоритмического уровней описания, заданных в форме графов переходов конечных автоматов, булевых уравнений, многоуровневых иерархических структур; генерация тестов проверки ОКН или исправного поведения; получение приемлемого быстродействия алгоритмов моделирования неисправностей для оценки качества тестов.

Теоретические источники работы: многозначная алгебра [1] – Hayes J.P., двухтактная кубическая алгебра [2], дедуктивный метод моделирования неисправностей [3-6] – Ермилов В.А., Armstrong D.B., Биргер А.Г., Levendel Y.H., методы генерации тестов [7] – Breuer M.A., Abramovici M.

2. Модель функционального элемента

Автоматная модель примитива имеет вид:

$$M=<X, Y, Z, f, g>,$$

где $X=(X_1,X_2,...,X_i,...,X_m)$, $Y=(Y_1,Y_2,...,Y_i,...,X_h)$, $Z=(Z_1,Z_2,...,Z_i,...,Z_k)$ – множества входных, внутренних и выходных переменных, отношения между которыми описываются уравнениями:

$$Y(t)=f[X(t-1), X(t), Y(t-1),$$

$$Z(t-1)]; Z(t)=g[X(t-1), X(t), Y(t-1), Y(t), Z(t-1)].$$

Переменные Z(t) отличаются от Y(t) тем, что первые наблюдаемы по выходным линиям, а Y(t) в этом смысле есть внутренние.

Функциональный последовательностный примитив задается компонентами:

$$F^2 = <(t-1,t),(X,Z,Y),\{A^2\}>,$$

где (t-1,t) – два автоматных соседних такта в описании функции; (X,Z,Y) – векторы входных, внутренних и выходных переменных; $\{A^2\}$ – двухтактный алфавит описания состояний (переходов) автоматных переменных [1,2]:

A^2={Q=00,E=01,H=10,J=11,O={Q,H},I={E,J},A={Q,E}, B={H,J},S={Q,J},P={E,H},C={E,H,J},F={Q,H,J},L={Q,E,J}, V={Q,E,H},Y={Q,E,H,J}, A^1={0,1,X={0,1}}, $\varnothing$ (U)}.

Форма описания примитивного элемента (ПЭ) – кубическое покрытие $C = (C_1, C_2, ..., C_i, ..., C_n)$,

где $C_i = (C_{i1}, C_{i2}, ..., C_{ij}, ..., C_{iq})$ – куб, включающий входные, внутренние, выходные координаты $C_i = (C_i^X, C_i^Y, C_i^Z)$, q=m+h+k. Для комбинационного автомата формат описания кубического покрытия $F^1 = <(t),(X,Z),\{A^1\}>$ определяется отношениями на (q=m+k)-мерном векторе переменных $C_i = (C_i^X, C_i^Z)$. Формат задает многовыходовой комбинационный примитивный элемент с m входами и k выходами. Двоичная булева функция от m переменных $Z=f(X_1,X_2,..., X_m)$ определяется при k=1.

3. Генерация тестов для комбинационных структур

Речь идет о практическом применении кубического исчисления [8] для построения проверяющих тестов методом очувствления одномерного логического пути эквипотенциальных линий, в котором присутствуют прямая и обратная фазы для выполнения процедур активизации и обеспечения соответственно.

Вместо двухтактных символов активизации [2,8] и обеспечения используются их однотактные составляющие, реализующие активность, или обеспечение в двух соседних временных фреймах. Такое упрощение позволяет значительно повысить технологичность алгоритма генерации тестов, сведя все процедуры к выполнению операций над символами алгебры Кантора $A^1 = \{0,1,X,\varnothing\}$.

В качестве исходной информации используется структурно-функциональная модель устройства, где для каждого примитива имеется кубическое покрытие $C=\{C_1, C_2, ..., C_i, ..., C_n\}$, по которому генерируются пары кубов покрытия активизации. Учитывая, что символы алфавита в парах кубов

активизации – однотактные, для выполнения обратной фазы – обеспечения используются только кубические покрытия примитивов. Результат выполнения алгоритма оформляется во множество пар векторов активизации $D = \{D_1, D_2, ..., D_i, ..., D_n\}$, $D_i = (D^1{}_i, D^2{}_i)$.

Для исключения итеративности процедур прямой и обратной импликации до начала работы алгоритма следует выполнить ранжирование линий и элементов. Первыми нумеруются в порядке возрастания входные линии схемы, после чего очередные номера присваиваются выходам тех элементов, входы которых уже занумерованы. Последними отмечаются выходы ПЭ, относящиеся к внешним выходам схемы. Аналогично осуществляется ранжирование ПЭ, с той лишь разницей, что на первом шаге нумеруются элементы, входы которых являются внешними входами объекта, а затем отмечается очередной ПЭ, имеющий уже отмеченных предшественников, последними нумеруются элементы, выходы которых принадлежат к внешнему разъему устройства.

Основные этапы К-алгоритма генерации тестов для комбинационных структур:

1. Выбор очередной входной эквипотенциальной линии j и установка в соответствующей координате нового вектора активизации символа активности (здесь и далее имеется в виду пара символов однотактного алфавита) D_j=01(10) (на первом шаге i-1, j=1). Все другие координаты вектора D, за исключением D_j, к началу построения активного пути должны иметь значения неопределенностей (XX).

2. Определение очередного примитива для выполнения прямой импликации символа активизации по правилу: линия, соответствующая входу ПЭ, должна иметь символ активности в текущем векторе D_i, выходная координата элемента по возможности не должна быть активизирована во всех предшествующих векторах D. Анализ примитивов заканчивается при достижении активности на внешних выходах схемы. Стратегия выбора пути активизации использует критерий получения минимального множества одномерных путей, покрывающих все эквипотенциальные линии объекта.

3. Обратная фаза заключается в обработке каждого примитива схемы, на выходе которого после прямой фазы существуют символы, отличные от (XX), исключая элементы, ставшие активными при выполнении прямой фазы. Расширение поля деятельности обратной импликации в целях доопределения символов активизации 01(10) связано с построением фактического пути, очувствляющего выход схемы, если первоначально, при выполнении прямой фазы была сделана попытка активизировать несущественную переменную. Для устранения итеративного характера обратной импликации очередным следует выбирать элемент, имеющий уже обработанных преемников. Суть анализа заключается в пересечении каждого куба C_l покрытия обеспечения с текущим вектором активности D_i. При непустом пересечении D_i с n кубами выполняется размножение строки D_i на n векторов, что является следствием многовариантности решения задачи обратной импликации.

4. Объединение текущих векторов, являющихся претендентами для активизации очередного входа устройства, после выполнения обратной импликации на очередном примитиве. Данная процедура имеет смысл при наличии сходящихся разветвлений, которая способна значительно уменьшить количество промежуточных решений, а значит и уменьшить время работы алгоритма. Объединение выполняется в случае различия пары строк только по одной координате – принцип минимизации, или при получении в результате выполнения координатной операции пересечения двух соседних векторов одного из них – правило поглощения.

5. После проектирования пар векторов активизации, покрывающих все линии в схеме, выполняется доопределение символа X на каждой входной координате по правилу

$$D^{1(2)}_{ij} = D^1_{ij} \cap D^2_{ij} \Leftarrow (D^1_{ij} \vee D^2_{ij} = X) \wedge (D^1_{ij} \neq D^2_{ij}) \cdot$$

В противном случае доопределение подчинено правилу $(D^1_{ij} \cap D^2_{ij} = X) \Rightarrow [D^1_{ij} = D^2_{ij} = 0(1)]$.

Координаты заменяются нулем или единицей в целях получения минимального кодового расстояния между соседними векторами теста. Возможны и другие критерии доопределения, например, с учетом максимизации проверяющих свойств двоичных наборов или минимизации входных векторов вычеркиванием одинаковых.

6. Для определения двоичных значений линий на всех тестовых последовательностях выполняется их моделирование, которое уточняет состояния невходных линий, оставшихся неопределенными даже после выполнения обратной фазы.

Полученный в результате выполнения К-алгоритма тест является полным относительно ОКН, если удалось построить только одномерные пути активизации, покрывающие все линии. В противном случае тест будет полным относительно всех ОКН всех неизбыточных или существенных линий. Минимальность полученного двоичного теста не гарантирована, поэтому есть смысл в оценке его качества с помощью программ моделирования неисправностей, что дает возможность получить минимальный набор входных проверяющих последовательностей.

Выполнение основных шагов алгоритма показано на рис. 1.

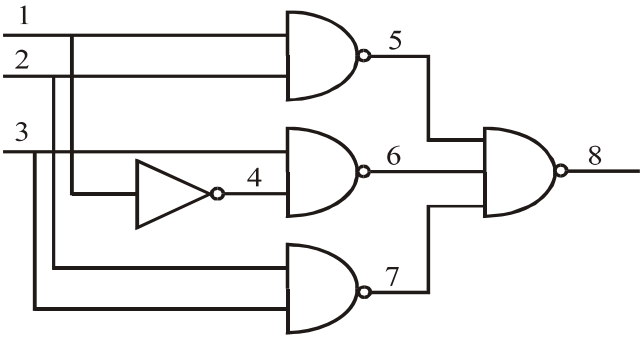

Рис. 1. Фрагмент избыточной схемы

В схеме формат записи переменных для элемента типа 2И-НЕ определяется векторами: $C^1 = \{(1,2,5); (3,4,6); (3,2,7)\}$. Для инвертора – $C^2 = (1,4)$. Для 3И-НЕ – $C^3 = (5,6,7,8)$. Покрытия соответствующих типов ПЭ, записанные по входным и выходным номерам эквипотенциальных линий, имеют вид:

$$C^1 = \begin{vmatrix} 0 & X & 1 \\ X & 0 & 1 \\ 1 & 1 & 0 \end{vmatrix}; \quad C^2 = \begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix}; \quad C^3 = \begin{vmatrix} 0 & X & X & 1 \\ X & 0 & X & 1 \\ X & X & 0 & 1 \\ 1 & 1 & 1 & 0 \end{vmatrix}.$$

Покрытия активизации упомянутых типов примитивов представлены парами кубов:

$$C^1 = \begin{vmatrix} 0 & X & 1 \\ 1 & 1 & 0 \end{vmatrix}; \begin{vmatrix} X & 0 & 1 \\ 1 & 1 & 0 \end{vmatrix}. \quad C^2 = \begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix},$$

$$C^3 = \begin{vmatrix} 0 & X & X & 1 \\ 1 & 1 & 1 & 0 \end{vmatrix}; \begin{vmatrix} X & 0 & X & 1 \\ 1 & 1 & 1 & 0 \end{vmatrix}; \begin{vmatrix} X & X & 0 & 1 \\ 1 & 1 & 1 & 0 \end{vmatrix}.$$

Для схемы (рис.2) выполнение прямой фазы определяет в формате переменных (1,2,3,4,5,6,7,8) следующие векторы активизации:

$$\begin{vmatrix} 1 & 1 & - & - & 0 & X & X & 1 \\ 0 & X & - & - & 1 & 1 & 1 & 0 \end{vmatrix}; \begin{vmatrix} 1 & - & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & - & X & 1 & X & 0 & X & 1 \end{vmatrix};$$

$$\begin{vmatrix} 1 & 1 & - & - & 0 & X & X & 1 \\ X & 0 & - & - & 1 & 1 & 1 & 0 \end{vmatrix}; \begin{vmatrix} - & 1 & 1 & - & X & X & 0 & 1 \\ - & 0 & X & - & 1 & 1 & 1 & 0 \end{vmatrix};$$

$$\begin{vmatrix} - & - & 1 & 1 & X & 0 & X & 1 \\ - & - & 0 & X & 1 & 1 & 1 & 0 \end{vmatrix}; \begin{vmatrix} - & 1 & 1 & - & X & X & 0 & 1 \\ - & X & 0 & - & 1 & 1 & 1 & 0 \end{vmatrix},$$

где знак "-" соответствует символу X. Выполнение обратной фазы для доопределения символов $D^1_{ij}(D^2_{ij})$ при выполнении условий

$$D^2_{ij}(D^1_{ij}) = X \wedge (D^1_{ij} \neq D^2_{ij})$$

на невходных линиях путем использования кубических покрытий задает тест активизации:

$$\begin{vmatrix} 1 & 1 & X & 0 & 0 & X & X & 1 \\ 0 & X & 0 & 1 & 1 & 1 & 1 & 0 \end{vmatrix}; \begin{vmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & X & 1 & X & 0 & X & 1 \end{vmatrix};$$

$$\begin{vmatrix} 1 & 1 & X & 0 & 0 & X & X & 1 \\ X & 0 & 0 & X & 1 & 1 & 1 & 0 \end{vmatrix}; \begin{vmatrix} X & 1 & 1 & X & X & X & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{vmatrix};$$

$$\begin{vmatrix} 0 & X & 1 & 1 & X & 0 & X & 1 \\ X & 0 & 0 & X & 1 & 1 & 1 & 0 \end{vmatrix}; \begin{vmatrix} X & 1 & 1 & X & X & X & 0 & 1 \\ 0 & X & 0 & 1 & 1 & 1 & 1 & 0 \end{vmatrix}.$$

Доопределение символов X на входных координатах в формате переменных (1,2,3) полученного теста в соответствии с правилами п. 5 К-алгоритма задает наборы D^X, которые после минимизации и вычеркивания одинаковых приобретают вид D^X_m:

$$D^X = (111,010,101,001,110,100,011,000,001,100,011,010),$$

$$D^X_m = (110,010,101,011,000).$$

Полученный минимизированный тест D^X_m содержит семь входных наборов, исправное моделирование которых представлено правой частью, а проверяющие способности теста – левой частью (в виде векторов проверяемых дефектов) матрицы по формату переменных (1,2,3,4,5,6,7,8):

$$\begin{vmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{vmatrix} \begin{vmatrix} 0 & 0 & . & . & 1 & . & . & 0 \\ 1 & . & 1 & . & 0 & 0 & 0 & 1 \\ . & 1 & . & 1 & 0 & 0 & 0 & 1 \\ 1 & . & 0 & 0 & . & 1 & . & 0 \\ . & 1 & . & . & 0 & 0 & 0 & 1 \\ . & . & 0 & . & . & . & . & . \\ . & . & . & . & 0 & 0 & 0 & 1 \end{vmatrix}.$$

Моделирование ОКН для наборов D^X_m позволяет сделать вывод об избыточности последних трех, которые дублируют проверку неисправностей, выполненную предыдущими векторами. Поэтому минимальным двоичным тестом являются последовательности: $D^X_{min} = (110,010,101,001)$.

Моделирование дефектов также подтверждает невозможность проверки дефекта 7^1 на полученных векторах, поскольку выход 7 относится к избыточному элементу.

Выполнение К-алгоритма, основанного на активизации минимального множества путей в схеме, от внешних входов до наблюдаемых выходов, покрывающих все линии, гарантирует построение полного проверяющего теста для тех ОКН, для которых он существует. Дальнейшая минимизация теста связана с применением средств моделирования неисправностей, что дает точную характеристику проверяющих свойств каждого вектора и теста.

4. Стратегия генерации тестов для цифровых систем

Объект тестирования – цифровые системы, описанные на языке аппаратуры VHDL в форме булевых уравнений, задающих как комбинационные схемы, так и триггерные структуры. Типы переменных определяются структурами языка VHDL: bit, bit-vector, std logic, boolean, integer.

Базовые методы генерации тестов:

1) Детерминированный. Основан на применении алгоритмов активизации одномерных путей на паре входных наборов для комбинационных схем и на упорядоченной последовательности входных векторов конечной длины для последовательностных. Преимущества определяются возможностью целенаправленного построения тестов для отдельных неисправностей. Недостатки: вычислительная сложность получения решения, если оно существует, определяется NP-полнотой; невозможность за приемлемое время получить полный тест для схем большой размерности (миллионы вентилей).

2) Алгоритмический. Основан на генерации регулярных алгоритмических входных последовательностей (бегущие 0 и 1, галоп 0 и 1, шахматный код, логарифмический), на требующей решения задачи обратной импликации для установки схемы в наперед заданное состояние. Преимущества заключаются в высоком быстродействии и проверке неисправностей для регулярных структур (сумматоры, счетчики, регистры). Недостатки определя-

ются низкой полнотой проверки неисправностей для последовательностных схем и нерегулярных комбинационных структур.

3) Псевдослучайный. Основан на использовании программных или аппаратурных перенастраиваемых или построенных генераторов псевдослучайных последовательностей для последующей подачи двоичных кодов на входы цифровой системы. Преимущества: высокое быстродействие, ориентация на полное пространство тестируемых состояний; инвариантность к структурной организации цифрового устройства. Недостатки: практическая недостижимость 100% полноты теста относительно заданных неисправностей за приемлемое время. Можно говорить о том, что алгоритмический метод есть частный случай псевдослучайного.

Для оценки качества генерируемых тестов необходимо иметь быстродействующую программу моделирования неисправностей, от которой зависит время построения тестов для цифровой системы. Она должна иметь высокое абсолютное быстродействие (тысячи наборов в секунду для схем, имеющих тысячи эквивалентных вентилей) и инвариантность по отношению к уровням описания цифровых систем (вентильный, алгоритмический, системный).

Учитывая, что все три метода генерации тестов имеют преимущества и недостатки, далее предлагаются стратегии, максимально использующие преимущества каждого из них.

1) Параллельная стратегия. Применение к модели цифрового устройства полного набора псевдослучайных тестов. Выбор из него минимального множества, покрывающего все ОКН, путем решения задачи определения покрытия всех неисправностей минимальным набором тестов. Если тест не полный, то задача покрытия рассматривается относительно полученного процента проверенных неисправностей, который не может быть уменьшен в процессе ее решения. Достоинством такого подхода является возможность распараллеливания времяемкого процесса генерации тестов путем использования нескольких процессоров или компьютеров в сети. По окончании самой длительной процедуры генерации тестов выполняется минимизация совокупного теста уже на одном мощном компьютере. Недостатки связаны с обработкой несложных (до 1000 эквивалентных вентилей) цифровых структур, для которых предлагаемая стратегия является весьма дорогостоящей.

2) Последовательная стратегия. Выполняется на одном компьютере путем последовательного генерирования псевдослучайных тестов, имеющихся в наличии:

$$T = (T^1, T^2, ..., T^i, ..., T^n).$$

После выполнения генерирования очередного теста T^i выполняется анализ покрытия неисправностей полученными наборами $Q(T^1 \cup T^2 \cup ... \cup T^i)$. Если тест неполный $Q(T^1 \cup T^2 \cup ... \cup T^i) < 100\%$ – генерируется очередная совокупность T^{i+1}, иначе – окончание последовательной стратегии. В случае фиксации результата

$$Q(T^1 \cup T^2 \cup ... \cup T^i \cup ... \cup T^n) < 100\%$$

по окончании генерирования всех тестов используется детерминированный метод, ориентированный только на генерацию тестов для еще не проверенных неисправностей цифровой системы. Преимущества определяются возможностью остановки процесса генерирования тестов при достижении требуемой полноты (100%), что предпочтительно для регулярных и несложных структур. Недостатки связаны с последовательной генерацией тестов, что предполагает значительные временные затраты при обработке схем большой размерности (тысячи эквивалентных вентилей).

3) Стратегия остаточного генерирования тестов. Учитывая высокую вычислительную сложность и точность детерминированного метода, он применяется для построения тестов к фрагментам цифровых систем или для проверки отдельных неисправностей, которые не были обнаружены тестами, сгенерированными с помощью стратегий 1 и 2. Для выбранной неисправности строится путь ее активизации до наблюдаемых выходов. Затем выполняется процедура обеспечения условий активизации, создающая входную последовательность, которая проверяет заказанный дефект. Преимущества определяются алгоритмической сходимостью при решении задачи построения теста для конкретной неисправности, если такой существует. Недостатки связаны с полным перебором решений для худшего варианта решения задачи обеспечения.

4) Использование перенастраиваемых генераторов псевдослучайных последовательностей. Генерация n_i наборов с равновероятным распределением по каждому входу нулей и единиц. Если полнота теста FC<100%, то выполняется подсчет количества переключений по всем линиям для целей перенастройки генератора по правилам прямо или обратно пропорциональным частоте появления единиц с последующей генерацией n_{i+1} наборов с измененными параметрами. Иначе – конец процедуры генерации тестов.

5) Использование множества алгоритмических генераторов, которые дают возможность получать полные тесты для регулярных структур (логические схемы, коммутаторы, дешифраторы, сумматоры, счетчики, регистры, память).

5. Анализ схемной структуры

Большая размерность цифровых проектов, определяемая миллонами эквивалентных вентилей, требует значительного времени генерации тестов в соответствии с вычислительной сложностью, задаваемой отношением

$$W(T) = [\sum_{i=1}^{M} (m_i \times q_i)]^2,$$

где m_i и q_i – число линий и кубов в покрытии i-го примитива схемы.

Для уменьшения данной оценки необходимо понижать размерность обрабатываемых объектов путем их декомпозиции на составляющие подсхемы с использованием следующих принципов:

1. Выделение входов синхронизации. Путем анализа кубических покрытий или кодов языка VHDL можно определить направление синхронизации (передний или задний фронт). Аналогично можно выполнить классификацию остальных входов путем выделения адресных, информационных, управляющих входов схемы.

2. Декомпозиция цифрового проекта на подсхемы по независимым входам синхронизации. Для каждого синхровода находятся функционально зависимые выходы, для которых определяются в качестве существенных аргументов внешние входы:

$$X_C \to \forall i[Y_i = f_i(X_C)] \to \forall j[X_j = f_i^{-1}(Y_i)].$$

Это дает возможность разбить схему на n функциональных модулей по числу независимых внешних синхровходов

$$S = \{S_1, S_2, S_{n-1}, S_n\}.$$

Взаимодействие подсхем определяется следующими отношениями:

$$S_j \bigcap_{j \neq r}^{j,r=\overline{1,n}} S_r \neq \varnothing; \bigcup_{j=1}^{n} S_j = S.$$

3. Определение подсхем, соответствующих триггерам. Осуществляется поиск всех триггеров в цифровом устройстве, для входов которого вычисляются все внешние входы, влияющие на состояние каждого триггера:

$$X^{F_i} = \{X_j | \forall j[X_j = f_i^{-1}(Y^{F_i})]\}.$$

Это дает возможность целенаправленно генерировать входные последовательности для проверки каждого триггера или использовать заранее построенные для него тесты. После определения для триггера внешних входов активизации тестом $X_F^{F_i}$ вычисляется подсхема, соответствующая триггеру F_i. Затем находятся внешние входы, отвечающие за фазу обеспечения при транспортировании теста для триггера на внешние выходы схемы:

$$X_B^{F_i} = X^{F_i} / X_F^{F_i},$$

где $X^{F_i} = \{X_F^{F_i}, X_B^{F_i}\}$ – полное множество входов, необходимое для тестирования триггера F_i.

4. Разбиение схемы S на фрагменты, число которых равно количеству выходов, с учетом следующих критериев для входов:

$$2^{|X|} \geq 2^{\left|X_1 = f_i^{-1}(Y_1)\right|} + 2^{\left|X_2 = f_i^{-1}(Y_2)\right|} + \ldots + 2^{\left|X_m = f_i^{-1}(Y_m)\right|}.$$

5. Композиция полученных разбиений путем объединения таких подсхем, которые имеют идентичное множество входных переменных или отличаются по ним не более чем q переменными:

$$S_i \cup S_j \leftarrow [(2^{|X_i|} + 2^{|X_j|} \geq 2^{|X_i \cup X_j|}) \,\&\, |S_i \cap S_j| \leq q],$$

где $X_i, X_j \in X = \{X_1, \ldots, X_i, \ldots, X_n\}$, а X – полное множество входов схемы (подсхемы), подлежащей разбиению. Параметр q следует выбирать, исходя из структуры схемы. Для двух уровневых схем он может быть соизмерим с числов входов подсхемы. Для многоуровневых и последовательностных его следует уменьшать до нуля.

Разбиение цифровой структуры на подсхемы дает возможность применять параллельную стратегию, но уже к частям цифрового устройства. В этом случае при генерации тестов используется более одного компьютера. В лучшем случае их число равно количеству подсхем, которые одновременно обрабатываются в целях генерирования полного теста. Далее полученные тесты для подсхем складываются:

$$T = (T^1 + T^2 + \ldots + T^i + \ldots + T^n),$$

если все подсхемы отвечают условию

$$\underset{i,j=\overline{1,n}}{\overset{i \neq j}{\forall}} i, j(S_i \cap S_j \neq \varnothing).$$

При этом длина теста определяется формулой

$$|T| = |T^1| + |T^2| + \ldots + |T^i| + \ldots + |T^n|.$$

В альтернативном случае, когда

$$\underset{i,j=\overline{1,n}}{\overset{i \neq j}{\forall}} i, j(S_i \cap S_j = \varnothing),$$

выполняется минимизация теста по правилу

$$T = (T^1 \cup T^2 \cup \ldots \cup T^i \cup \ldots \cup T^n).$$

Длина теста для всей схемы будет равна

$$|T| = \max(|T^1|, |T^2|, \ldots, |T^i|, \ldots, |T^n|).$$

В общем случае, когда выполняются условия

$$\underset{i,j=\overline{1,n}}{\overset{i \neq j}{\exists}} i, j(S_i \cap S_j \neq \varnothing) \,\&\, \underset{i,j=\overline{1,n}}{\overset{i \neq j}{\exists}} i, j(S_i \cap S_j = \varnothing),$$

формирование теста происходит путем анализа очередной пары подсхем для построения объединенного теста:

$$T = T^p \ddot{\cup} T^q = \begin{cases} T^p * T^q \leftarrow S_i \cap S_j \neq \varnothing; \\ T^p \ddot{\cap} T^q \leftarrow S_i \cap S_j = \varnothing, \end{cases}$$

где * – операция конкатенации или последовательного присоединения тестов для подсхем; $\ddot{\cap}$ – матричная операция пересечения тестов, которая определяется в следующем виде:

$$T = T^p \ddot{\cap} T^q = \begin{cases} T_{ij}^p = T_{ij}^p \bigcap_{i=\overline{1,n_q}}^{j=\overline{1,m}} T_{ij}^q \leftarrow |T^p| \geq |T^q|; \\ T_{ij}^q = T_{ij}^p \bigcap_{i=\overline{1,n_p}}^{j=\overline{1,m}} T_{ij}^q \leftarrow |T^p| \leq |T^q|, \end{cases}$$

здесь n_p и n_q – длина (число векторов) в тестах T^p и T^q; m – общее число линий в схеме цифрового устройства.

Таким образом, общая длина теста после объединения частичных тестов для подсхем определяется интервалом

$$\sum_{i=1}^{n}\left|T^i\right| \le |T| \le \max_i\left(\left|T^i\right|\right) \text{ или}$$

$$\sum_{i=1}^{n} n_i \le |T| \le \max_i (n_i).$$

На основе изложенной методики декомпозиции схемы и применения нескольких стратегий для генерации тестов разработана автоматическая система генерации тестов, структура которой представлена на рис. 2.

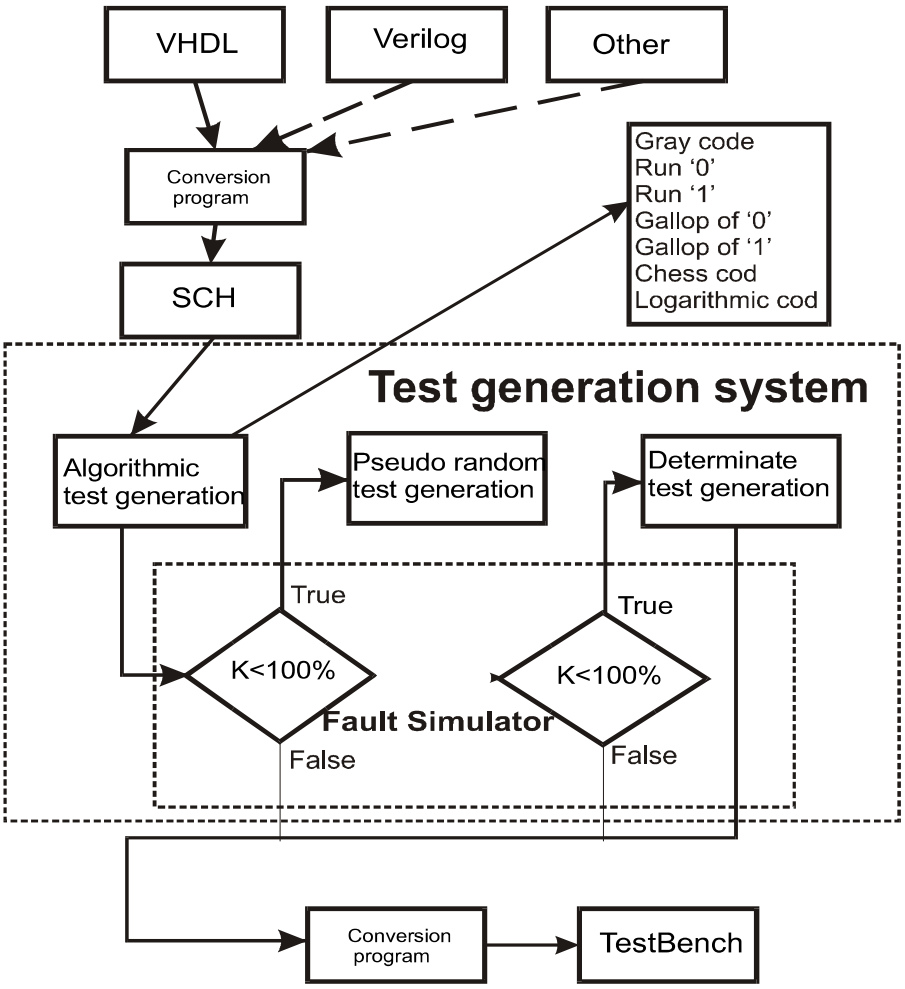

Рис. 2. Структура системы генерации тестов для цифровых объектов, описанных булевыми уравнениями

С помощью упомянутой системы обработано: 80 комбинационных схем из списка ISCAS'85; 140 комбинационных и последовательностных схем из проекта PRUS; 45 последовательностных схем большой размерности из проекта PRUS; 22 последовательностных схемы из списка ITC'99. Среднее время проектирования детерминированных тестов – 1,5 часа. Средняя сложность проекта – 1000 линий. Среднее время проектирования псевдослучайных тестов – 4 минуты. Качество тестов – более 90 %. Декомпозиция структуры цифровой системы на функциональные модули позволяет уменьшить время генерации тестов на 25%.

6. Заключение

Для цифровых объектов большой размерности предложены три стратегии генерации тестов, ориентированные на распараллеливание процессов, использование методов декомпозиции схемы на функционально законченные фрагменты. Даны оценки и процедуры разбиения цифровой системы на подсхемы и маркирования внешних входов. Предложена адаптация К-алгоритма [4] к троичному алфавиту кубического исчисления, что значительно упрощает процедуры прямой и обратной импликаций, являющихся основой детерминированного проектирования тестов.

Поступила в редколлегию 21.05.2001

Рецензент: д-р техн. наук, проф. Кривуля Г.Ф.

Литература: 1. *Hayes J.P.* A systematic approach to multivalued digital simulation// ICCD-84: Proc. IEEE Int. Conf. Comput. 1984. No. 4. P. 177-182. 2. *Хаханов В.И.* Техническая диагностика элементов и узлов персональных компьютеров.К.: ІЗМН. 1997.308 с. 3. *Ермилов В.А.* Метод отбора существенных неисправностей для диагностики цифровых схем. Общие выражения для неисправностей, возможных при эксперименте // Автоматика и телемеханика, 1971, № 1.С. 159-167. 4. *Armstrong D.B.* A deductive method of simulating faults in logic circuits. IEEE Trans. on Computers. Vol. C-21. No. 5. 1972. P. 464-471. 5. *Биргер А.Г.* Многозначное дедуктивное моделирование цифровых устройств // Автоматика и вычислительная техника 1982. №4. С.77-82. 6. *Levendel Y.H. and Menon P.R.* Comparison of fault simulation methods – Treatment of unknown signal values.- Journal of digital system. Vol.4. 1980. P.443-459. 7. *Abramovici M., Breuer M.A. and Friedman A.D.,* Digital System Testing and Testable Design, Computer Science Press, 1998. 652 p. 8. *Хаханов В.И., Сысенко И.Ю., Чамян А.Л.* Проектирование тестов для структурно-функциональных моделей цифровых схем // Радиоэлектроника и информатика. 1999. №.3. С. 49-57.

Мд. Масуд Мехеди, аспирант кафедры автоматизации проектирования вычислительной техники ХНУРЭ. Научные интересы: генерация проверяющих тестов для цифровых систем. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26.