УДК 519.713:681.326

ТЕХНОЛОГИЯ МОДЕЛИРОВАНИЯ И СИНТЕЗА ТЕСТОВ ДЛЯ СЛОЖНЫХ ЦИФРОВЫХ СИСТЕМ

ХАХАНОВ В.И., КОЛЕСНИКОВ К.В., ПАРФЕНТИЙ А.Н., ХАХАНОВА И.В., ОБРИЗАН В.И., МЕЛЬНИКОВА О.В.

Предлагаются быстродействующие программные средства для синтеза тестов и моделирования неисправностей, ориентированные на обработку сложных цифровых систем, включающих сотни тысяч эквивалентных вентилей на стадии проектирования. Все алгоритмы и методы, реализованные в программах, ориентированы на значительное (в десятки и сотни раз) уменьшение времени генерации тестов, моделирования дефектов для цифровых проектов, благодаря использованию новых технологий структурного и логического анализа, реконфигурированию моделей объектов в процессе обработки. Описываются структуры данных, ориентированные на компилятивно-интерпретативную реализацию алгоритмов, что делает систему гибкой по отношению к оперативной модификации моделей и быстродействующей в части транспортирования списков константных и заказных неисправностей.

1. Введение

Актуальность проектирования новых средств анализа дефектов определяется необходимостью значительного повышения быстродействия генерации тестов для структурно- и функционально-сложных цифровых систем, имплементированных в кристаллы FPGA, ASIC. В качестве прототипов разработанных средств можно рассматривать автоматические системы тестирования известных фирм: Cadence, Mentor Graphics, Synopsys, Logic Vision[1], ориентированные на эффективную обработку кристаллов, насчитывающих до 100 тысяч вентилей. Здесь существуют два фактора, рассматриваемые как недостатки прототипов: 1) высокая стоимость упомянутых систем – сотни тысяч долларов; 2) время анализа становится неприемлемым, если в качестве объекта выступает устройство, имеющее миллионы вентилей. Поэтому актуальным представляется решение проблемы существенного повышения (на порядок) быстродействия анализа сложной цифровой системы на стадии ее проектирования в целях построения тестов верификации и анализа их качества. В рамках решения упомянутой проблемы используются новые технологии: структурный анализ цифровых проектов; реконфигурирование модели в процессе обработки, совмещение преимуществ дедуктивного и параллельного методов моделирования, экспертный выбор генераторов тестов из множества алгоритмических, детерминированных и псевдослучайных.

Объект тестирования – цифровой проект, описанный на языке VHDL, представленный в виде системы булевых уравнений, имплементируемых в кристаллы программируемой логики.

[1] – [www.cadence.com, www.logicvision.com, www.simucad.com, www.syntest.com, www.synopsys.com, www.mentorgraphics.com]

Цель – разработка быстродействующего программного комплекса тестирования проектируемых цифровых изделий на этапах их формализации, синтеза, имплементации, что позволяет существенно уменьшить время верификации цифровых систем, благодаря генерированию тестов и оцениванию их качества путем моделирования константных и заказных неисправностей.

Практические задачи:

1. Проектирование инструментальных средств оценивания качества тестов путем программной реализации параллельного, дедуктивного, дедуктивно-параллельного, дедуктивно-параллельного с обратным прослеживанием, приближенного дедуктивного алгоритмов анализа дефектов.

2. Создание инструментальных средств синтеза тестов проверки неисправностей на основе программной реализации алгоритмических, детерминированных и псевдослучайных генераторов.

3. Реализация дружественного многооконного интерфейса пользователя, позволяющего: отображать результаты моделирования неисправностей и исправного поведения, осуществлять выбор режимов синтеза тестов и анализа их качества, отображать структурные фрагменты цифровых схем, выводить статистическую информацию о структуре объекта тестирования и результатах его обработки.

4. Валидное тестирование и верификация средств моделирования и синтеза на тестовых примерах из библиотек ISCAS и проектах цифровых изделий большой размерности. Сравнительный анализ характеристик системы с лучшими мировыми аналогами.

Задачи исследования:

1. Разработка структурной компилятивно-интерпретативной модели регистрового уровня описания цифровой схемы для синтеза тестов и моделирования неисправностей.

2. Создание алгоритмов структурно-функционального анализа комбинационных и последовательностных цифровых систем в целях определения множества сходящихся разветвлений и реконфигурации структуры схемы в процессе синтеза тестов и моделирования дефектов.

3. Разработка дедуктивно-параллельного, дедуктивно-параллельного с обратным прослеживанием, квази-точного дедуктивного алгоритмов анализа дефектов.

4. Создание алгоритмических, детерминированных и псевдослучайных генераторов, включающих реализацию эволюционных алгоритмов, активизацию одномерных путей для комбинационных и последовательностных схем.

Исходная информация для разработки алгоритмов синтеза тестов, моделирования неисправностей, структурного анализа проектируемого объекта представлена публикациями: BDP-метод (Backtraced Deductive-Parallel) моделирования неисправностей [1,2,8,11,14], дедуктивные модели транспортирова-

ния дефектов [3,4-7,10], параллельный метод обработки списков неисправностей функционального элемента [4-6], алгоритм обратного прослеживания примитивов [8,10,11,15] при обработке цифрового устройства, структурный анализ цифровых систем в целях определения сходящихся разветвлений для комбинационных и последовательностных схем [8,14,16], технологии верификации и тестирования в процессе автоматизированного проектирования дискретных объектов [9,10,12], промышленная система моделирования и отладки цифровых проектов Active-HDL [13], с которой взаимодействует комплекс SIGETEST.

2. Структура программного комплекса

В соответствии с задачами, решаемыми программным комплексом, были разработаны следующие основные модули, представленные на рис. 1:

1. Конвертор во внутренние структуры данных. Использует компилятор VHDL2BOOL. Продукт фирмы Aldec [13]. Осуществляет преобразование исходного файла, представленного на языке описания аппаратуры VHDL (Verilog), к формату BNF. Представляет собой описание цифрового устройства в виде булевых уравнений. Содержит трансляторы кодов из поддерживаемых системой форматов описания цифрового проекта во внутренние структуры данных – программы BE2SCH, RTL2BESF, EDIF2BESF. Результат работы указанных программ выводится на экран и хранится во внутреннем бинарном формате, который необходим для работы других приложений.

2. Блок автоматизированной генерации тестовых последовательностей. Реализует синтез тестовых наборов тремя группами методов: алгоритмическими, детерминированными и псевдослучайными. Первая группа представлена тремя методами. Это алгоритмы – классический детерминированный, кубический и кратных символов. Данные методы являются наиболее точными, однако непригодны для тестирования больших проектов, поскольку время их обработки составляет десятки часов. Тем не менее в рассматриваемой системе они используются для синтеза тестов сильнопоследовательностных фрагментов схем, регистровых и счетных структур.

Вторая группа методов представлена тремя стратегиями создания тестовых последовательностей. Это – псевдослучайная, алгоритмическая и метод генетических алгоритмов. Генерация псевдослучайных последовательностей используется для комбинационных схем большой размерности, а также для задания начальной популяции при реализации метода генетических алгоритмов. Подсистема реализации последнего реализует следующую стратегию: выбор подмножества алгоритмических тестов в качестве начальной популяции в зависимости от структурной и функциональной сложности объекта; определение фитнесс-функции популяции, являющейся оценкой качества теста или вектора; выполнение процедур кроссинговера и мутации на основе использования не случайных, но детерминированных процедур; формирование особей-потомков для следующей популяции. Для эффективного применения программных средств метода генетических алгоритмов необходимо квалифицированное задание параметров обработки схемы, вычисляемых на основе анализа структуры тестируемого цифрового устройства. Подсистема синтеза тестов алгоритмическими методами использует следующие генераторы: логарифмический, код Грея, сигнатурные регистры, бегущий 0, бегущая 1, шахматный код, галоп 0, галоп 1, регулярные тест-коды проверки регистровых и счетных структур.

3. Моделирование неисправностей с использованием интерпретативных и компилятивных моделей. На первом этапе производится топологический анализ схемы в целях выявления глобальных обратных связей и сходящихся разветвлений. Необходимость

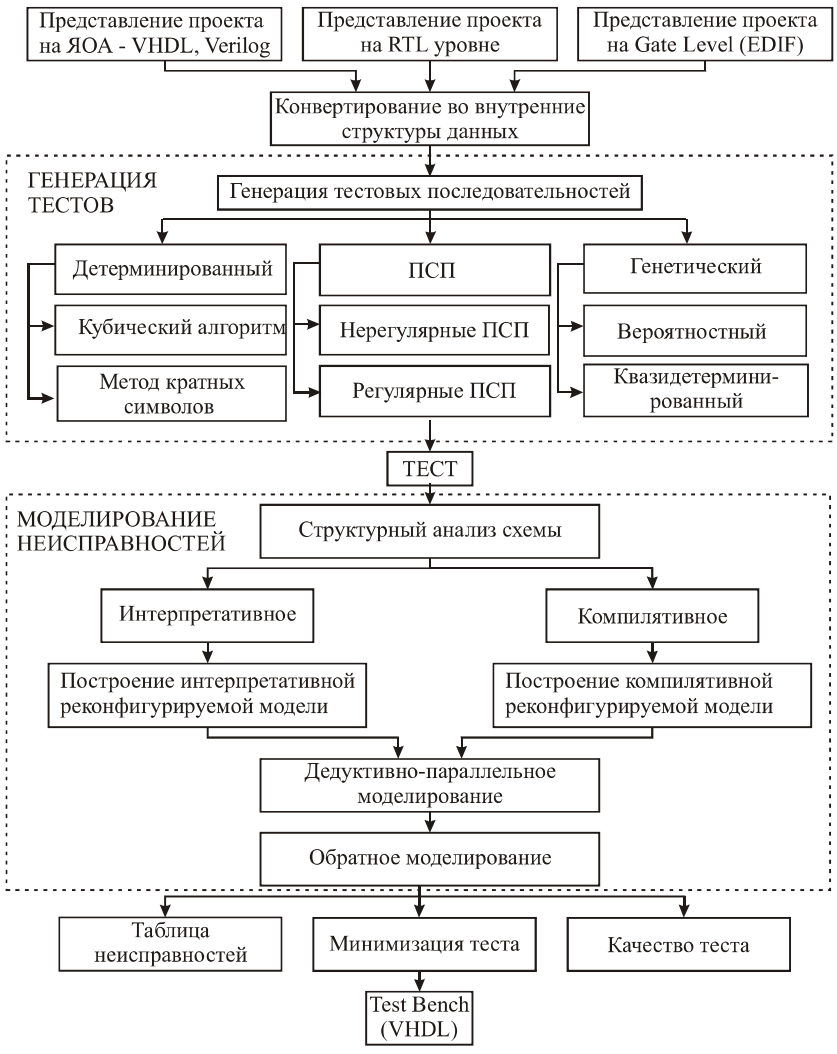


Рис. 1. Структура программного пакета

определения последних связана с повышением адекватности алгоритма обратного моделирования методом суперпозиции векторов проверяемых дефектов всех примитивов. В случае присутствия в схеме сходящихся разветвлений выполняется дедуктивно-параллельный анализ проверяемости дефектов разветвлений на выходах схемы, иначе – переход ко второму этапу – алгоритму обратного моделирования неисправностей.

Анализ дефектов на основе процедуры обратной суперпозиции требует уже линейных затрат памяти и времени в функции от числа эквипотенциальных линий и квадратичных затрат для обработки сходящихся разветвлений:

$$Q = (r^2 / W) + n_r + n_p + (n - r - r^0),$$

где (r^2 / W) – время моделирования неисправностей r сходящихся разветвлений, число которых определяется как $r = 0.2 \times n$; $n_r = n$ – время реконфигурирования примитивов схемы на входном наборе; $n_p = n$ – время поиска подграфов линий, соответствующих непроверяемым сходящимся разветвлениям; $(n - r - r^0) = n - 0.2 \times n - 0.4 \times n = 0.4 \times n$ – время выполнения процедуры суперпозиции на множестве линий схемы без сходящихся разветвлений и предшественников для непроверяемых сходящихся разветвлений. Учитывая фактические значения указанных параметров в функции от числа линий схемы, можно получить следующую оценку быстродействия BDP-метода:

$$Q = [(0.2 \times n)^2 / W] + n + n + (n - 0.2 \times n - 0.4 \times n) =$$
$$= [(0.2 \times n)^2 / W] + 2.4 \times n).$$

Таким образом, выигрыш в быстродействии предложенного метода тем больше, чем меньше процент сходящихся разветвлений в схеме цифрового устройства.

4. Обработка диагностической информации. Минимизация теста. В силу требований совместимости соседних тестовых наборов, а также многотактного транспортирования ОКН по линиям обратных связей при обработке последовательностных схем, процедура минимизации теста для таких структур имеет отличия. Все наборы разбиваются на самоустанавливающиеся тестовые сегменты. Их можно переупорядочивать по критерию уменьшения качества, сохраняя нумерацию векторов внутри каждого сегмента неизменной. Можно исключать отдельные сегменты из теста, если качество последнего не уменьшается. Такие действия позволяют устранять избыточность теста, минимизируя его.

Форматирование теста в стандарт VHDL – TestBench. Конвертирование полученного теста, удовлетворяющего условиям полноты, в формат языка VHDL (Verilog) в целях его последующего использования в среде Active-HDL для верификации проекта цифрового устройства.

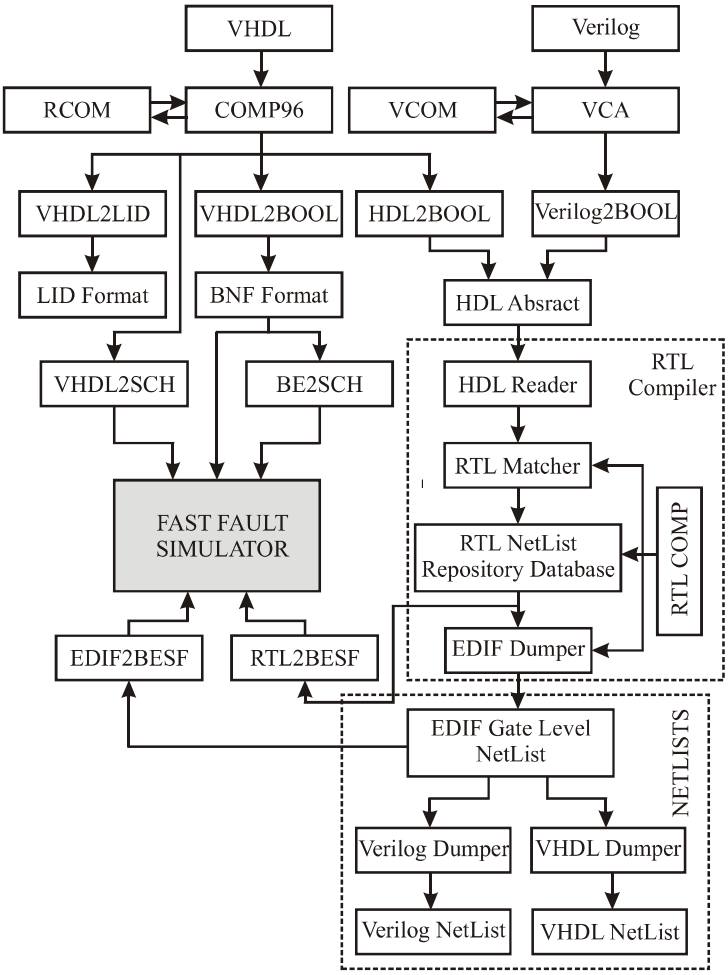


Рис. 2. Информационная поддержка HDL-стандартов IEEE

3. Форматы, поддерживаемые системой SIGETEST

Цель: Создание условий инвариантности работы средств синтеза тестов и моделирования неисправностей по отношению к существующим языкам описания аппаратуры: System C, Verilog, VHDL. Для этого разработаны трансляторы с подмножеств упомянутых языков во внутренний универсальный формат, из которого, в свою очередь, можно получить описание объекта в одном из трех аппаратурных языков. Поддерживается библиотека RTL элементов фирмы XILINX. Место системы анализа дефектов, интегрированной в автоматическую среду генерации тестов, а также совместимость и взаимодействие внутренних форматов данных с языками описания аппаратуры показаны на рис. 2.

3.1. BNF-формат. Предназначен для описания цифрового устройства в виде системы булевых уравнений с использованием ограниченного числа операторов и элементов. Ориентирован на автоматическое построение модели устройства для выполнения алгоритмов анализа неисправностей и генерации тестов. BNF-структура дает полное представление о вентильном эквиваленте устройства, подобно списку соединений (NetList). Программа VHDL2BOOL предназначена для компиляции кода VHDL в формат BNF. Таким образом, устройство, заданное на языке высокого уровня, автоматически

получает описание на вентильном уровне как комбинационных, так и последовательностных схем. Операторы, которые могут использоваться в формате BNF: '|' – OR; '&' – AND; '!' – NOT; '^' – XOR.

Особенность представления комбинационных и последовательностных схем заключается в том, что BNF-структура содержит строго определенную библиотеку последовательностных примитивов: 18 вентильных эквивалентов триггеров и защелок, каждый из которых при описании имеет обозначение – ключевое слово pragma asyn (t-1), а также название триггера, например, flip-flop(C_high,D). Более сложные последовательностные элементы, такие как регистры, счетчики, синтезируются в BNF-формат путем композиции из существующих библиотечных примитивов. Пример задания двух цифровых схем показан в листингах 1 и 2.

Листинг 1. BNF-описание комбинационной схемы c17 из библиотеки тестовых примеров ISCAS'89:

```
gat10=!(gat1&gat3);
gat11=!(gat3&gat6);
gat16=!(gat2&gat11);
gat19=!(gat11&gat7);
gat22=!(gat10&gat16);
gat23=!(gat16&gat19);
```

Листинг 2. BNF-описание последовательностной цифровой схемы:

```
process equations begin, line: 14
pragma asyn(t-1);
{ C_tmp0=s;
D_tmp0=s&(a);
latch(C_high,D)
S_tmp0=D_tmp0&C_tmp0;
R_tmp0=C_tmp0&!D_tmp0;
z(t)=S_tmp0|(!R_tmp0&z(t-1)); }
state.out file begin
state.out file end
process equations end, line: 14
```

В примере после ключевого слова pragma asyn(t-1) следует набор уравнений триггера-защелки. Ключевое слово latch(C_high,D) определяет тип триггера.

3.2. SCH-формат. Для программной реализации детерминированных алгоритмов генерации тестов необходимо иметь модель схемы в формате SCH, ориентированном на кубическое (табличное) описание функций примитивов.

Формат SCH состоит из четырех компонентов:

$$S = \{S_h, S_s, S_l, S_n\},$$

соответственно, раздел заголовка; структура схемы; описание примитивов; таблица соответствий линий схемы.

1) Заголовок. Включает общую информацию о схеме: количество линий, внешних входов и выходов, число обратных связей и сходящихся разветвлений.

2) Структура схемы. Включает список всех линий и их объединения в примитивы; номер примитива, нагруженного на линию и его входные линии. Все линии обозначаются в соответствии с принципом: первыми нумеруются линии памяти, затем внутренние и выходные. Численное значение номера выхода каждого элемента комбинационной части должно быть больше номера любого из его входов.

3) Описание примитивов. Включает библиотеку, содержащую функциональное описание каждого элемента в виде кубического покрытия в алфавите {0, 1, X, U}. При использовании нескольких элементов одного типа в схеме в библиотеке хранится только одно описание. Поэтому ее размерность определяется не общим количеством элементов в схеме, а числом типов в ней.

4) Имена линий. Включает список линий и соответствующие им номера. До начала обработки схемы строковые имена линий обозначаются десятичными числами. Соответствия значений имен и номеров линий записываются в таблицу и хранятся в отдельном разделе.

3.3. RTL DataBase Repository. Предсталяет собой выходной формат программного модуля RTL Compiler, разработанного фирмой Aldec. Является системой предварительного синтеза проекта, заданного на языке описания аппаратуры VHDL. Выполняет функцию синтаксического разбора произвольного VHDL-кода, описывающего цифровую систему. Обрабатывается поведенческое и структурное описание, при этом уровень вложенности иерархических структур неограничен. RTL-формат представляет собой планарное неиерархическое структурное соединение стандартных элементов из библиотеки Repository: защелки, триггеры, сумматоры, компараторы, регистры, счетчики, мультиплексоры, дешифраторы.

Описание цифровой системы на поведенческом уровне в языке VHDL и после синтеза в формате RTL представлено листингами 3 и 4 соответственно:

Листинг 3. VHDL-описание триггера-защелки:

```
library ieee;
use ieee.std_logic_1164.all;
entity dff is
    port (
    DIN : in std_logic;
    CLK : in std_logic;
    RES : in std_logic;
    SET : in std_logic;
    Qout: out std_logic  );
end entity;
architecture dff of dff is
begin
    process (CLK)
```

```
    begin
        if RES = '1' then
                    Qout <= '0'; --(others => '0');
        elsif SET = '1' then
                    Qout <=  '1'; --(others => '1');
        elsif CLK'event and CLK = '1' then
                    Qout <= DIN;
        end if;
    end process;
end dff;
```

Листинг 4. RTL-структура триггера-защелки:

```
.primitive (REGISTER)
    .parameter (SIZE)
    .output (Q, SIZE, REG)
    .input (D, SIZE, "1")
    .input (CLK, 1, "ACTIVE_HIGH")
    .input (RST, 1, "ASYNCH")
    .input (SET, 1, "ASYNCH")
    .input (CE, 1, "ACTIVE_HIGH")
    .attribute (PRIORITY, "RST")
    .attribute (REG_TYPE, "LATCH")
.end primitive
```

Для формальной верификации полученной структуры регистрового уровня используется транслятор RTL2VHDL, который позволяет путем исправного моделирования тестовых последовательностей в среде Active-HDL верифицировать структурно-функциональные модели.

3.4. Формат EDIF Gate Level. Предназначен для представления схемы на вентильном уровне и задает синтезированную структуру с учетом ПЛИС, в которую имплементируется цифровое устройство. В зависимости от типа ПЛИС, фирмы-производителя программируемой логики, выбранного характера синтеза, уровня оптимизации проекта имплементированное цифровое устройство содержит, помимо вентильных элементов, примитивы (black board), внутренняя структура которых не доступна в явном виде. При транслировании формата EDIF во внутренние структуры данных системы SIGETEST описание указанных элементов раскрывается с помощью открытых IP-core. Иначе black board необходимо досинтезировать, имея VHDL-описание. Листинг 5 иллюстрирует пример формата EDIF для триггера-защелки, представленной в листинге 4.

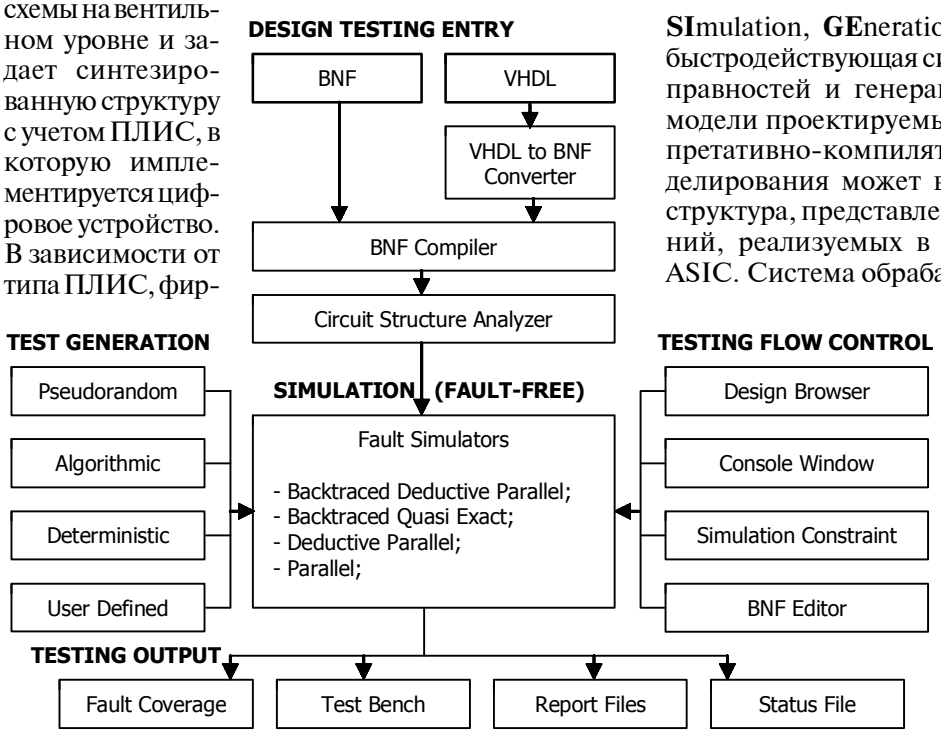


Рис. 3. Пользовательская структура программного пакета SIGETEST

Листинг 5. Фрагмент кода EDIF-формата синтезированной структуры триггера-защелки:

```
(cell PWR (cellType GENERIC)
  (view prim (viewType NETLIST)
   (interface
     (port Y (direction OUTPUT)))
   (property area (integer 0))))
(cell OB33PH (cellType GENERIC)
  (view prim (viewType NETLIST)
   (interface
     (port PAD (direction OUTPUT))
     (port A (direction INPUT)
     (property load (integer 1))
```

Таким образом, разработанная система SIGETEST поддерживает следующие форматы: вентильное описание на языке VHDL (транслятор VHDL2SCH), булевых уравнений в виде BNF (транслятор BNF2SCH), RTL Data Base Repository (транслятор RTL2BESF), вентильное описание в виде EDIF.

4. SIGETEST для пользователя

SImulation, **GE**neration of **TEST** (**SIGETEST**) – быстродействующая система моделирования неисправностей и генерации тестов, использующая модели проектируемых цифровых систем интерпретативно-компилятивного типа. Объектом моделирования может выступать любая цифровая структура, представленная в виде булевых уравнений, реализуемых в кристаллах CPLD, FPGA, ASIC. Система обрабатывает сложные цифровые проекты, насчитывающие сотни тысяч логических вентилей на стадии после синтеза (gate level description). Упрощенная структура основных доступных для пользователя модулей представлена на рис. 3.

Система имеет интегрированную среду, реализующую графический интерфейс высокого уровня. Ввод проектов осуществляется в виде

VHDL- или BNF-описания. Поддерживаемые операции: AND, OR, NOT, XOR. Также поддерживаются шинные структуры. Компилятор преобразует BNF-описание к виду внутренних структур данных, удобных для моделирования. Он имеет название SDP, используется структурным анализатором и ядром моделирования.

Ядро моделирования включает алгоритмы исправного и неисправного моделирования: Parallel, Backtraced Quasi Exact, Deductive-Parallel и Backtraced-Deductive-Parallel.

Генератор тестов включает набор алгоритмов для генерации тестовых последовательностей: псевдослучайный, детерминированный, алгоритмический. Также имеется возможность загрузки тестовых последовательностей из файла, указанного пользователем.

Результатом работы программы является Test Bench в формате VHDL. Система предоставляет также информацию о результатах исправного и неисправного моделирования, качество покрытия неисправностей, статистику моделирования, файлы отчетов. Результаты моделирования можно просмотреть с помощью окна Fault Coverage, представляющего собой многозначную таблицу неисправностей.

SIGETEST имеет средства для управления и мониторинга процесса синтеза тестов. Моделирование можно ограничить во времени или задать число тестовых наборов, которые необходимо промоделировать. Имеется также возможность ограничения процентом покрытия неисправностей генерируемыми наборами. В процессе моделирования система предоставляет информацию о прогрессе моделирования в процентах от общего числа векторов или наперед заданного временного интервала.

В системе имеется документация, дающая полную информацию об использовании интерфейса программы, поддерживаемых форматах, примеры работы с программой и рекомендации выхода из возникающих экстремальных ситуаций.

SIGETEST ориентирована на интеграцию с современными средствами синтеза и моделирования, такими как ALDEC Active-HDL, Riviera, SYNOPSYS Design Compiler.

5. Пользовательский интерфейс

Главное окно программы представляет собой разновидность многодокументного интерфейса и состоит из двух стационарных окон, а также клиентской области. Стационарными являются: окно со списком файлов, которые участвуют в моделировании, и окно консоли, предназначенное для вывода сообщений о работе программы. Клиентская область предназначена для отображения различных типов окон, таких как информатор загруженной схемы, результаты моделирования, отображение покрытия неисправностей наборами, окно редактирования файла. В верхней части главного окна находятся панель инструментов и меню, причем часть его команд дублирована в панели инструментов для удобства работы пользователя.

Загрузка схемы осуществляется выбором пункта меню «File/Open Design...». Пользователь может выбрать файл в форматах VHDL, BNF, SDP. Формат SDP получается преобразованием формата BNF и применяется непосредственно при моделировании. Программа автоматически создает SDP-формат, если загружается файл BNF. Если рядом с последним обнаружен одноименный файл формата SDP, то программа предложит пользователю загрузить его. Пользоваться этим целесообразно в случае, если разработчик уверен, что обнаруженный файл формата SDP был получен из неизмененного файла BNF. В противном случае следует заново сгенерировать файл SDP. Также для загрузки схемы необходим файл с описанием ее интерфейса, имеющий такое же имя, но с расширением «IDX». В случае загрузки VHDL-файла IDX создается автоматически при помощи внешнего генератора.

После успешной загрузки файла в окне появляется дерево файлов, а в клиентской области – окно с информацией о загруженной схеме: имя файла, количество входов, выходов, тип схемы (комбинационная или последовательностная), общее число линий схемы, число сходящихся разветвлений, число загруженных векторов теста. Последний используется в генераторе типа «User» и загружается при выборе пункта меню «File/Open User Vectors File...». Чтобы просмотреть содержимое какого-либо файла, пользователю достаточно осуществить его активизацию двойным щелчком левой кнопки мыши.

Параметры моделирования пользователь задает на панели инструментов. Первый из них – тип генератора – может быть алгоритмическим, псевдослучайным и пользовательским. Для алгоритмического типа генератора (из восьми возможных) необходимо указать также подтип в окне справа от типа. Следующими параметрами являются числовой ограничитель моделирования и единицы его измерения (векторы или миллисекунды). Таким образом, пользователь может установить продолжительность моделирования количеством векторов или желаемым качеством теста. Следующим параметром (по умолчанию отключенным) является ограничитель моделирования по достижению определенного качества теста (измеряется в процентах). Последним по расположению, но не по значимости является выбор метода моделирования дефектов. Представлено 4 реализации анализа: дедуктивный, обратного прослеживания, обратного прослеживания дедуктивно-параллельный, сочетающий в себе комбинацию первых двух, и параллельный. Выбрав параметры моделирования, пользователь активизирует процесс нажатием кнопки «Run Simulation». Имеется возможность прервать моделирование активизацией кнопки «Stop Simulation». По завершению процесса в клиентской области появляется окно со следующей информацией: количество обработанных тест-векторов, время моделирования, достигнутое качество теста и скорость обработки векторов. Также пользователь может осуществить просмотр покрытия неисправностей тест-вектора-

ми, выбрав кнопку или пункт меню «Faults Coverage». В нижней строке окна будет показано вектор проверенных тестом неисправностей. В верхней части – векторы исправного поведения и проверяемых неиспрвностей для каждого входного набора. Переключение режимов осуществляется путем нажатия кнопки «To faults mode / To logic mode». В режиме отображения векторов проверяемых неисправностей рядом с номером каждого набора указывается качество покрытия неисправностей в процентах. Также пользователь имеет возможность получить информацию о моделировании из лог-файлов. Они являются текстовыми, что даёт возможность их распечатки для целей документирования. Лог-файлы содержат такую же информацию, как и информационные окна программы.

Выбрав пункт меню Help / FaultGUI Help, пользователь может получить общую справку о системе моделирования: форматы входных и выходных файлов, общие вопросы моделирования неисправностей, а также пути решения проблемных ситуаций. В справке имеется обучающий алгоритм, в которой шаг за шагом, с иллюстрациями и подробными комментариями рассмотрен процесс обработки тестового примера – цифровой схемы – от загрузки файла до получения результатов синтеза тестов и моделирования неисправностей.

6. Конвертор, компилятор, структурный анализатор

Конвертор представляет собой утилиту для преобразования VHDL-файлов в формат BNF. Последний имеет более предпочтительную структуру для компиляции, чем VHDL, поскольку он определен на более простом синтаксисе и не содержит иерархию в проекте. Булевы уравнения VHDL-файла конвертируются сразу в аналогичные структуры BNF. Для раскрытия иерархии используются более сложные алгоритмы. Это происходит по методике анализа «снизу-вверх»: осуществляется раскрытие иерархии на более низком уровне, а затем выше, пока не будет достигнут top-level. После этого все компоненты файла становятся связанными в структуру схемы. Для VHDL-модели находится интерфейс модуля верхнего уровня путём анализа иерархии проекта.

Компилятор – это утилит для преобразования файлов BNF-формата во внутреннюю структуру SDP для неисправного моделирования. Синтаксический анализ проводится методом рекурсивного спуска. В процессе анализа структуры схемы производится оптимизация, разбиение на двухвходовые элементы, нахождение сходящихся разветвлений, обратных связей. Реализован механизм разрешения конфликтов, если несколько символьных идентификаторов соответствуют одной физической линии.

7. Ядро моделирования

Ядро моделирования реализует набор функций для исправного и различных типов неисправного моделирования. Можно выделить следующие этапы работы:

1) Чтение файла формата SDP, проверка его корректности, инициализация внутренних структур данных, построение реконфигурируемой модели на основании структуры графа схемы.

2) Исправное моделирование производится в трёхзначной логике {0,1,X}. Для последовательностных схем хранится информация о результатах исправного моделирования на предыдущей итерации, а также осуществляется контроль за возникновением режима генерации.

3) Реконфигурирование модели схемы и последующая реализация дедуктивно-параллельного моделирования дефектов сходящихся разветвлений.

4) Моделирование ОКН древовидных структур, не имеющих сходящихся разветвлений, методом обратной суперпозиции на текущем тест-векторе.

Описанные стадии моделирования составляют базовый Backtraced-Deductive-Parallel метод. Путем модификации его отдельных компонентов можно получать: Deductive-Parallel-; Backtraced Quasi Exact-; Parallel-; Deductive- методы, которые включены в штатные режимы моделирования системы SIGETEST.

Программная реализация алгоритмов моделирования комбинационных и последовательностных схем существенно отличается. Это связано с тем, что для обработки последовательностных схем необходимо хранить информацию на двух итерациях в целях обнаружения режимов генерации списков неисправностей. В связи с этим затраты времени и памяти для анализа последовательностных схем вдвое больше, чем для комбинационных такой же размерности.

Отладчик ядра (Simulation.exe) служит для тестирования и верификации программно реализованных методов моделирования. Он не имеет дружественного интерфейса (ввод настроек осуществляется при помощи командной строки) и предназначен для тестирования правильности работы алгоритмов моделирования и синтеза тестов путем генерирования структурно-сложных примеров цифровых систем большой размерности.

8. Тестирование и верификация

Цель тестирования – поиск ошибок в SIGETEST путем анализа полученных результатов на заданных входных данных и их сравнение с эталонными. Анализ функционирования программного комплекса моделирования и синтеза тестов был проведен на более чем 600 комбинационных и последовательностных схемах (триггеры, регистры, счетчики, сумматоры, мультиплексоры и схемы на их основе). Тестовые примеры были сгенерированы или взяты из библиотек конференций и с сайтов фирм, лидеров в области проектирования и тестирования. При этом существенным представлялось многообразие структур: комбинационная, последовательностная, вариация количества входов, выходов, сходящихся разветвлений, обратных связей. Был проведен анализ результатов исправного моделирования и неисправностей методами: Backtraced

Quasi Exact, Deductive-Parallel, Backtraced-Deductive-Parallel и Parallel.

Результаты исправного моделирования проверялись на пакетах Active-HDL 5.2, Nemesis, ModelSim. Для этой цели была разработана программа конвертации булевых уравнений в VHDL-модель. Сравнение проводилось в автоматическом режиме при помощи соответствующих скрипт-файлов командной строки и встроенного макро-языка Active-HDL.

Верификация SIGETEST проводилась в два этапа:

1. Результаты моделирования неисправностей, полученные каждым из методов, сравнивались с теоретически ожидаемыми данными, просчитанными вручную и при помощи пакетов Nemesis, HITECH-PROOFS.

2. Файлы анализа дефектов, полученные различными методами, сравнивались между собой.

На основании проведенного анализа строился вывод о наличии или отсутствии ошибок в работе программы. Формировался список ошибок с указанием их типов в целях их дальнейшего устранения. В процессе тестирования были выявлены следующие особенности Quasi Exact метода: 1. Для некоторых последовательностных схем результаты моделирования неисправностей на тестовом наборе не совпадают с данными, полученными тремя другими методами. 2. При моделировании неисправностей на схемах, содержащих сходящиеся разветвления, могут возникать неточности, предсказанные теорией. Во избежание этого необходимо перемоделировать все сходящиеся разветвления одним из трех других методов.

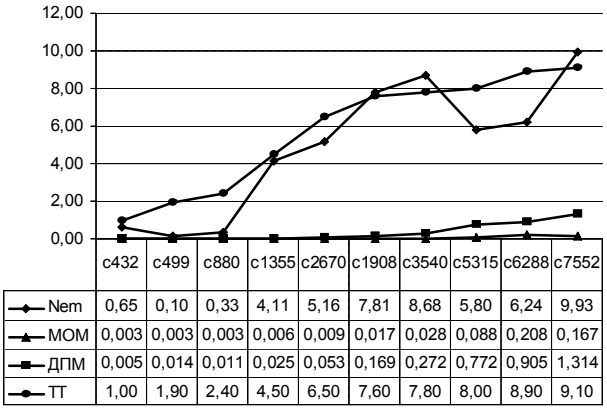
9. Заключение

Система моделирования неисправностей и синтеза тестов ориентирована на обработку сверхсложных цифровых систем на основе ПЛИС, содержащих миллионы вентилей. Тестовые эксперименты программной реализации метода на сотнях цифровых комбинационных и последовательностных схем дали хорошие результаты по быстродействию по сравнению с традиционными алгоритмами параллельного и дедуктивного анализов. Отдельные примеры оценивания быстродействия разработанного метода (обработка тест-примеров на 1000 входных последовательностей) и существующих базовых показаны на рис. 4. Ускорение моделирования составляет не менее десяти раз. На рис. 5 представлены результаты анализа быстродействия трех методов моделирования цифровых схем на одном и том же компьютере, время обработки 1000 векторов. Показано существенное преимущество предложенного ОДП-метода перед дедуктивно-параллельным. Выигрыш в быстродействии более существенен для схем большой размерности. Число сходящихся разветвлений в тест-схемах в среднем составляет 20% от общего количества линий. Преимущества в быстродействии квази-точного метода (см. рис.5) связаны со снижением адекватности моделирования неисправностей сходящихся разветвлений. На рис. 6 и 7 показаны затраты времени и памяти для обработки больших схем тремя разработанными методами. Таким образом, основным научным результатом, внедренным в SIGETEST, является усовершенствование дедуктивно-параллельного метода моделирования неисправностей цифровых систем, которое заключается в:

1) создании обобщенной модели процесса дедуктивно-параллельного анализа цифровой схемы на основе процедуры обратной суперпозиции, имеющей линейную вычислительную сложность от числа линий схемы, не принадлежащих к сходящимся разветвлениям;

2) разработке дедуктивных алгоритмов структурно-функционального анализа схем в целях определения множества сходящихся разветвлений и реконфигурации структуры для реализации процедуры суперпозиции;

3) создании внутренней интерпретативно-компилятивной модели цифрового устройства для эффективного исправного анализа логических элементов и их неисправностей одиночного константного типа.



	c432	c499	c880	c1355	c2670	c1908	c3540	c5315	c6288	c7552
Nem	0,65	0,10	0,33	4,11	5,16	7,81	8,68	5,80	6,24	9,93
МОМ	0,003	0,003	0,003	0,006	0,009	0,017	0,028	0,088	0,208	0,167
ДПМ	0,005	0,014	0,011	0,025	0,053	0,169	0,272	0,772	0,905	1,314
TT	1,00	1,90	2,40	4,50	6,50	7,60	7,80	8,00	8,90	9,10

Рис. 4. Анализ быстродействия: Nem – система Nemesis; МОМ – ОДП-метод моделирования; ДПМ – дедуктивно-параллельный метод; TT– Turbo Tester

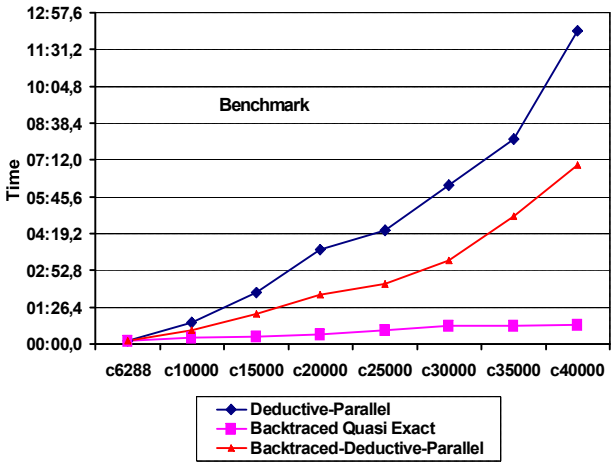


Рис. 5. Анализ быстродействия трех методов моделирования

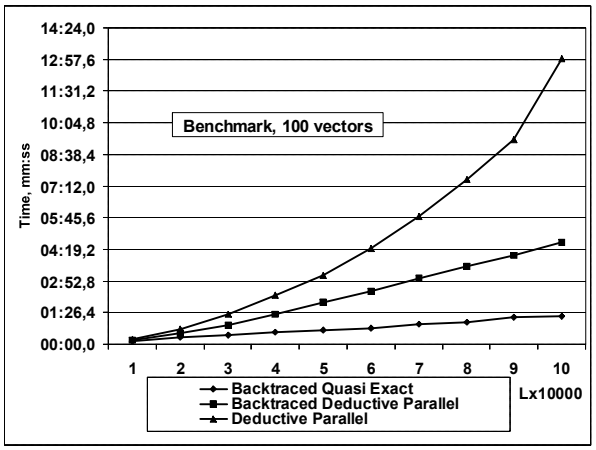


Рис. 6. Анализ быстродействия методов на автоматически генерируемых цифровых схемах

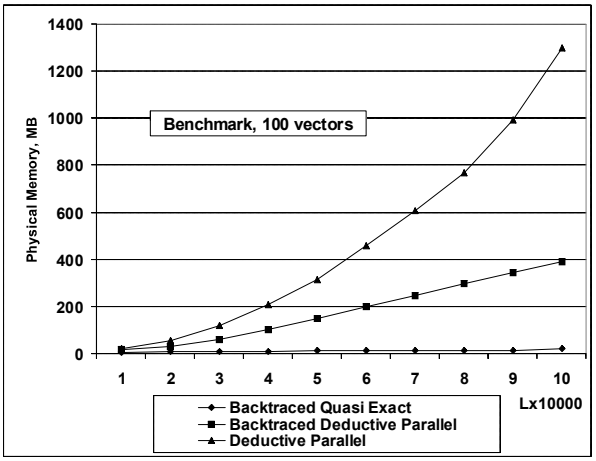


Рис. 7. Анализ затрат оперативной памяти для трех методов на автоматически генерируемых цифровых схемах

Дальнейшее развитие системы SIGETEST связано с усовершенствованием структурного анализа в целях повышения быстроедействия моделирования неисправностей цифровых схем, а именно:

1) исследование компенсации и возникновения ложных дефектов для всех промышленно используемых типов триггеров;

2) теоретическое доказательство корректности выполнения обратной суперпозиции списков неисправностей для триггерных структур;

3) исследование проблемы структурного анализа для асинхронных схем с глобальными обратными связями.

Литература: 1. *Hahanov V.I., Babich A.V., Hyduke S.M.* Test Generation and Fault Simulation Methods on the Basis of Cubic Algebra for Digital Devices. Proceedings of the Euromicro Symposium on Digital Systems Design DSD2001. Warsaw, Poland. September, 4-6, 2001. P. 228-235. 2. *Хаханов В.И., Хак Х.М. Джахирул, Масуд М.Д. Мехеди.* Модели анализа неисправностей цифровых систем на основе FPGA, CPLD // Технология и конструирование в электронной аппаратуре. 2001. № 2. С. 3-11. 3. *Хаханов В.И., Сысенко И.Ю., Хак Х.М. Джахирул, Масуд М.Д. Мехеди.* Кубическое моделирование неисправностей цифровых проектов на основе FPGA, CPLD // Радиоэлектроника, информатика, управление. 2001. № 1. С. 123-129. 4. *Levendel Y.H., Menon P.R.* Comparison of fault simulation methods − Treatment of unknown signal values // Journal of digital systems. Vol. 4. 1980. P. 443-459. 5. *Abramovici M., Breuer M.A. and Friedman A.D.* Digital System Testing and Testable Design, Computer Science Press, 1998. 652 p. 6. *Хаханов В.И.* Техническая диагностика элементов и узлов персональных компьютеров. К.: ІЗМН. 1997.308 с. 7. *Убар Р.Р.* Анализ диагностических тестов для комбинационных цифровых схем методом обратного прослеживания неисправностей // Автоматика и телемеханика. 1977. №8. С.168-176. 8. *Семенец В.В., Хаханова И.В., Хаханов В.И.* Проектирование цифровых систем с использованием языка VHDL. Харьков: ХНУРЭ, 2003. 492 с. 9. *Хаханов В.И., Соколов А.Ю., Чамян А.Л.* Детерминированный метод генетических алгоритмов для синтеза тестов верификации цифровых систем // Радиоэлектроника и информатика. 2001. № 4. С. 125-129. 10. *Хаханов В.И., Сысенко И.Ю., Колесников К.В.* Дедуктивно-параллельный метод моделирования неисправностей на реконфигурируемых моделях цифровых систем // Радиоэлектроника и информатика. 2002. №1. С. 98-105. 11. *Хаханов В.И., Колесников К.В., Хаханова А.В.* BDP-метод моделирования неисправностей для синтеза тестов цифровых проектов // Радиоэлектроника и информатика. 2002. №2. С. 60-66. 12. *Бондаренко М.Ф., Кривуля Г.Ф., Рябцев В.Г., Фрадков С.А., Хаханов В.И.* Проектирование и диагностика компьютерных систем и сетей. К.: НМЦ ВО. 2000. 306 с. 13. *Active*-VHDL Series. Book #1 - #4. Reference Guide. ALDEC Inc. 1998. 206 p. 14. *Хаханов В.И., Убар Р.-Й.Р.* Технологии проектирования систем на кристаллах. Моделирование неисправностей сверхбольших цифровых проектов // АСУ и приборы автоматики. 2002. Вып.122. С. 16-35. 15. *Jutman A., Ubar R., Hahanov V., Skvortsova O.* Practical works for on-line teaching design and test of digital circuits. Proceedings of the 9-th IEEE International Conference on Electronics, Circuits and System ICECS 2002. Dubrovnik, Croatia, September 15-18, 2002. P. 1223-1227. 16. *Hahanov V.I., Skvortsova O.B., Sysenko I.Y., Chamyan H.L.* ATPG System and Test Generation Methods for Digital Devices. 8-th Bienuual Baltic Electronics Conference. Tallinn, Estonia. October 6-9, 2002. P. 299-302.

Поступила в редколлегию 16.11.2002

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

Хаханов Владимир Иванович, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика вычислительных устройств, систем, сетей и программных продуктов. Увлечения: баскетбол, футбол, горные лыжи. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26. E-mail: hahanov@kture.kharkov.ua

Колесников Константин Васильевич, ст. преподаватель кафедры КС Черкасского государственного технологического университета. Научные интересы: тестирование цифровых систем и сетей. Увлечения: живопись − постимпрессионизм, поэзия − серебряный век, символизм. Адрес: Украина, 18006, Черкассы, бульв. Шевченко, 460, тел. 43-74-28, e-mail: kvvk601@mail.ru

Парфентий Александр Николаевич, студент ХНУРЭ. Научные интересы: техническая диагностика вычислительных устройств. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26.

Хаханова Ирина Витальевна, канд. техн. наук, доцент кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика и проектирование вычислительных устройств. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26.

Обризан Владимир Игоревич, студент ХНУРЭ. Научные интересы: техническая диагностика вычислительных устройств. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26.

Мельникова Ольга Вячеславовна, студентка ХНУРЭ. Научные интересы: техническая диагностика вычислительных устройств. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26.