УДК 681.325:519.713

СИСТЕМА ГЕНЕРАЦИИ ТЕСТОВ ЦИФРОВЫХ ПРОЕКТОВ В СРЕДЕ ACTIVE-HDL

ХАХАНОВ В.И., РУСТИНОВ В.А., ГОРБУНОВ Д.М., КОВАЛЕВ Е.В., МАСУД М.Д. МЕХЕДИ, ХАК Х.М. ДЖАХИРУЛ

Предлагаются средства генерации тестов для цифровых проектов, создаваемых в среде Active-HDL. Входное описание проектов основано на использовании языков: VHDL, Verilog, а также графического представления цифрового автомата. Получаемые тесты используются для верификации цифровых проектов средствами моделирования Active-HDL. Для оценки полноты покрытия неисправностей используется программная реализация метода кубического моделирования.

1. Введение

Цель создания автоматической системы генерации тестов заключается в верификации цифровых проектов в системе Active-HDL, реализуемых в программируемые логические интегральные схемы (ПЛИС) – Field Programable Gate Array (FPGA), Complex Programable Logic Device (CPLD), которые достойно конкурируют с базовыми матричными кристаллами, сигнальными процессорами.

Задачи, решаемые системой генерации тестов:

1. Построение проверяющих тестов для цифровых автоматов, заданных содержательными граф-схемами алгоритмов. Языки описания: VHDL, Verilog.

2. Генерация тестов для цифровых проектов, заданных булевыми уравнениями. Язык описания: VHDL.

3. Определение качества тестов путем моделирования одиночных константных неисправностей (ОКН) [1-3].

Объект диагностирования – цифровой проект, ориентированный на его реализацию в аппаратуре FPGA, CPLD.

Методы исследований: теория множеств, теория автоматов, теория графов, кубическое исчисление, метод активизации одномерных путей для генерации тестов, алгоритмические и псевдослучайные тестовые генераторы, метод кубического моделирования неисправностей, система Active-HDL фирмы Aldec Inc.

Фирма Aldec (USA, Las-Vegas) более 15 лет работает на рынке высоких технологий проектирования цифровых систем на основе FPGA, CPLD. Специализируется на создании компиляторов с языков описания аппаратуры высокого уровня VHDL, Verilog, EDIF, а также имеет собственные разработки аппаратурных и программных симуляторов. Стратегические партнеры фирмы: Xilinx, Sinopsis, Synplicity. Основной продукт фирмы – Active-HDL (v.3.5, 3.6, 4.1, 4.2), система ввода, моделирования и верификации цифровых проектов – 20 000 инсталляций имеет следующие функции: ввод проекта, управление разработкой, моделирование,отладка и верификация проектов, построение тестов.

2. ASFTEST – генератор тестов для конечных автоматов

Система Active-HDL предназначена для проектирования сложных цифровых устройств и позволяет выполнять их описание на языках VHDL и Verilog, а также с помощью программ-надстроек в графическом формате содержательного графа автомата или в графическом формате описания структуры устройства. В состав системы входят программы: управления проектом, ввода проекта на языках описания аппаратуры VHDL и Verilog HDL, ввода графического иерархического описания схемы, ввода содержательного графа автомата, компиляторы, логического моделирования, просмотра временных диаграмм, формирования тестовых воздействий оператором, просмотра списков, интегрированный отладчик, допускающий пошаговое выполнение программы, генераторы шаблонов и различные сервисные программы. Процесс проектирования с применением указанных программ представлен на рис. 1.

Существует несколько путей интеграции программного модуля и передаваемых данных в систему: 1) Модуль, встраиваемый в исполняемый код. Достоинства: защита от нелицензионного использования, возможно сокращение исполняемого кода при использовании одинаковых по отношению к базовой программе библиотек, возможность прямого доступа к данным из основной программы. Недостатки: модуль должен быть написан на том же языке, что и вся система, нет возможности внесения оперативных изменений. Возможны конфликты в случае использования одних и те же имен для глобальных переменных. 2) Динамически загружаемая библиотека. Достоинства: оптимально используется память, модифицирование модуля без изменения базовой программы, передача данных через ОЗУ. Недостатки: более сложный механизм передачи данных, для тестирования необходим фрагмент базовой программы. 3) Исполняемая программа. Достоинства: изменение модуля независимо от базовой программы, для тестирования может быть достаточно только файлов с данными и самой программы. Недостатки: данные могут передаваться только в виде файла, что в некоторых случаях резко снижает скорость обработки данных; возможность нелицензионного использования, если не предусмотрена защита.

Для интеграции программы в САПР был выбран 3-й вариант, как самый простой, удобный для тестирования и не требующий доработки базовой программы. Таким образом, для работы системы автоматической генерации тестов требуется трансляция из входного формата и генерация выходных данных в одном из стандартных форматов. В качестве входного принят формат графического представления цифрового автомата. В качестве выходного используются языки описания аппаратуры VHDL и Verilog HDL. В программе предусмотрена генерация файла макрокоманд, позволяющая эффективно взаимодействовать разработчику с системой проектирования.

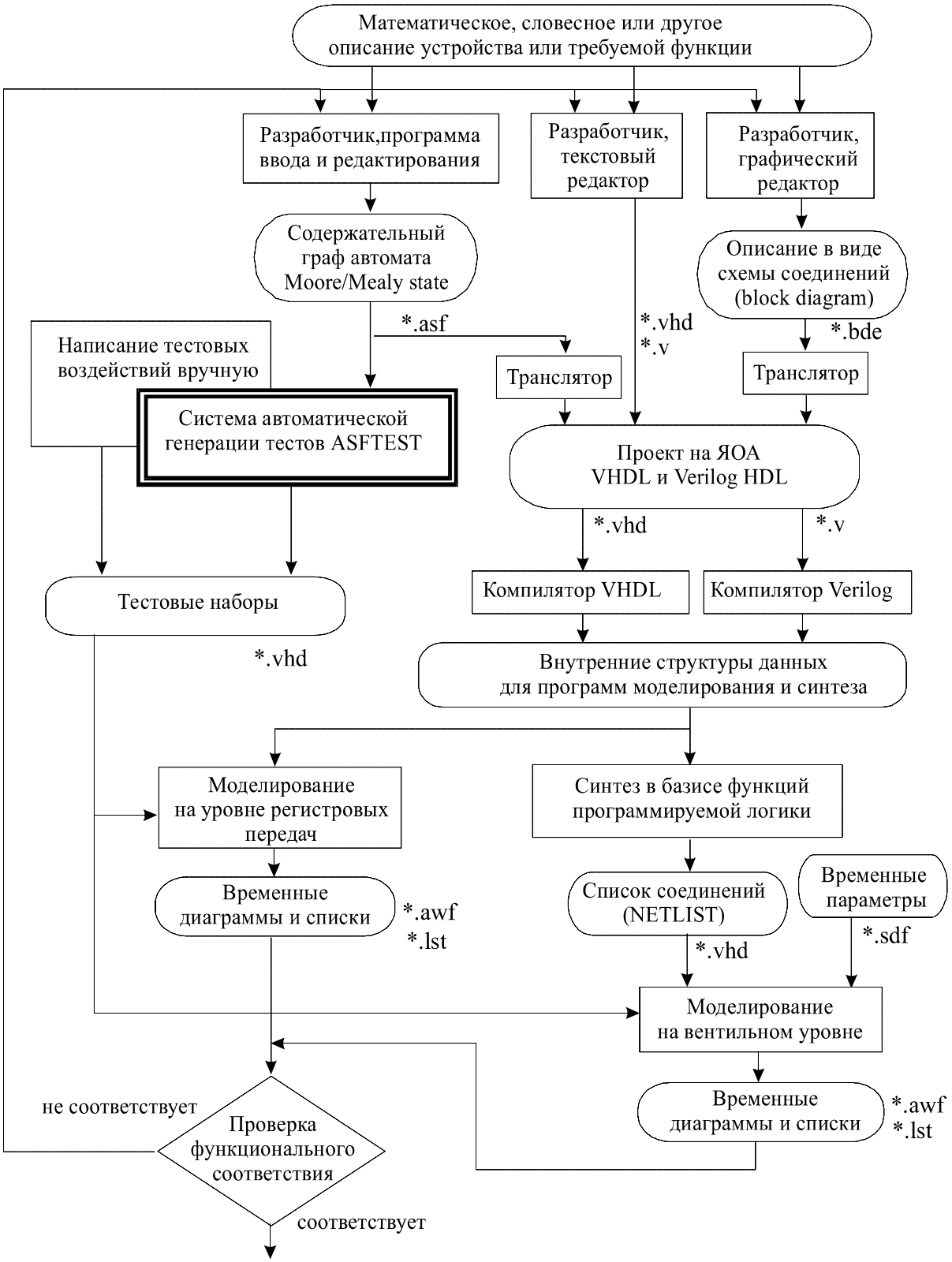

Рис.1. Структура пакета VHDL

2.1. Описание структуры программы ASFTEST и внутренних преобразований

Схема прохождения данных в программе ASFTEST приведена на рис.2. В качестве входной информации в программу лексического и семантического анализа поступает ASF-файл, который преобразуется в таблицу данных об объектах. Полученная таблица подвергается нескольким преобразованиям, связанным с ускорением обработки входных данных. Каждому объекту присваивается уникальный номер и они сортируются. Выполняется восстановление связей, так как один объект может быть частью другого. Трансляция во внутренние структуры данных совершается в несколько этапов. Вначале происходит трансляция констант, описаний сигналов и переменных во внутреннее представление. Далее производится формирование списка автоматов, выделение памяти для состояний и

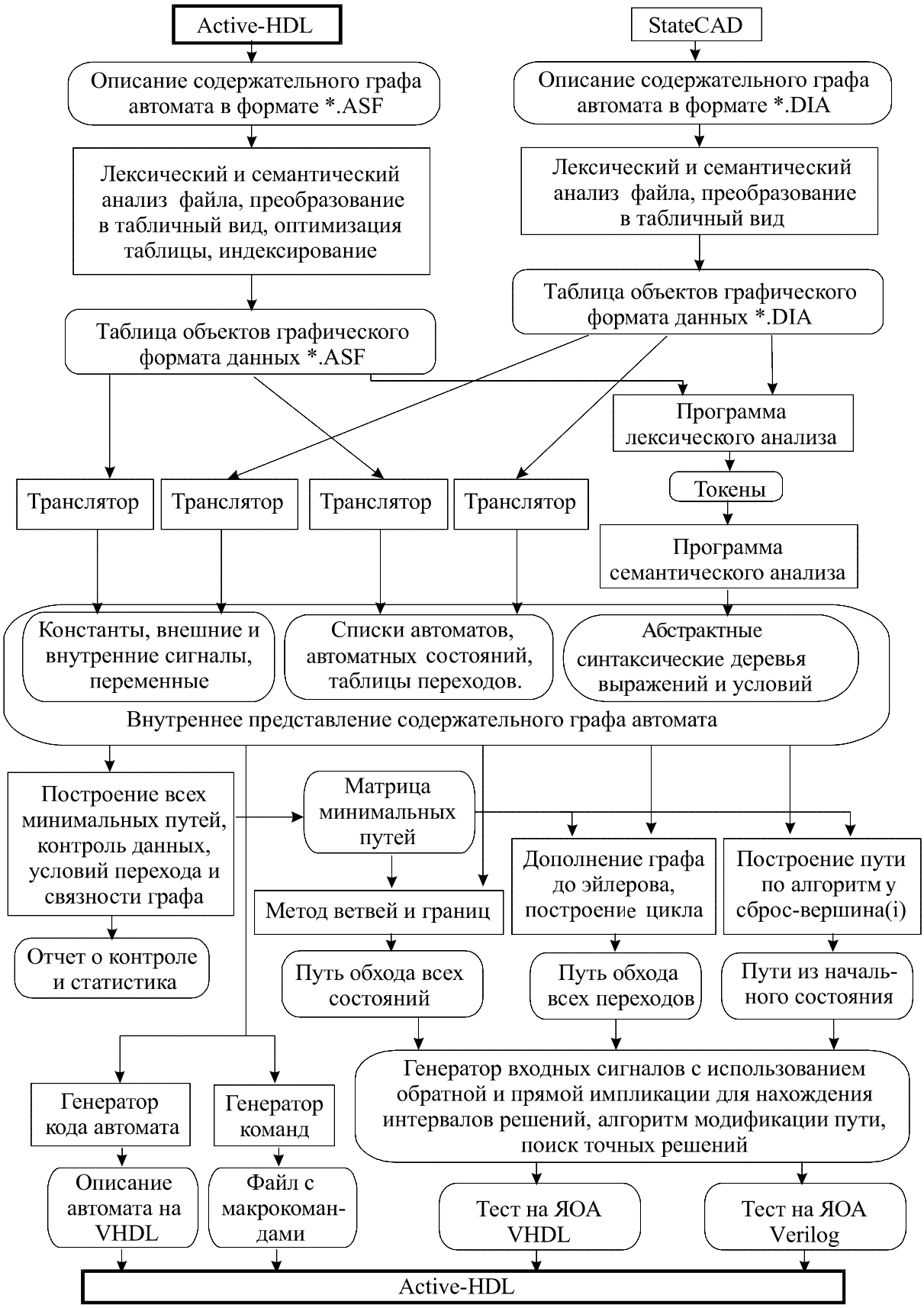

Рис. 2. Структура пакета ASFTEST

переходов. При формировании для каждого автомата списков состояний и переходов выполняется лексический и семантический разбор условий перехода (функция возбуждения) и действий, связанных с данным переходом. В результате выполнения указанных операций для каждого условия и выражения строится абстрактное синтаксическое дерево. Данные также могут поступать в формате DIA (система StateCAD). Этот формат значительно отличается от формата ASF, поэтому наиболее про-

стым способом трансляции его во внутреннее представление данных было написание отдельного транслятора (см. рис.2). После того как получено внутреннее представление данных, выполняется их модификация и анализ. Производится добавление условий перехода в состояние по умолчанию только для тех состояний, для которых такой переход не будет тождественно-ложным, к каждому состоянию добавляются глобальные переходы. Анализируются условия перехода на наличие тождественно ложных условий. Наличие тождественно-ложных условий недопустимо при построении теста по стратегии 2 (проверка всех переходов). Для контроля правильности перевода во внутреннее представление данных может генерироваться VHDL-код автомата. Выполняется построение матрицы минимальных путей. С ее помощью проверяется связность полученного графа. В случае, если граф не является связным, построение теста проверяющего все состояния невозможно. Поэтому выполнение программы прерывается с выдачей соответствующего сообщения.

После того как построена матрица минимальных путей, выполняется построение теста по одному из алгоритмов. При использовании стратегии обхода всех состояний по методу ветвей и границ строится последовательность обхода всех состояний. Для стратегии обхода всех переходов строится путь, проходящий через все дуги полученного графа, результат записывается в виде последовательности состояний. При реализации стратегии сброса в i-е состояние выдаются 2n пар элементов матрицы путей минимальной длины, где n – число состояний автомата.

Полученная последовательность состояний должна быть активизирована с помощью подачи на входные линии значений, вычисляемых с помощью процедуры обратной импликации. Однако получение таких значений не всегда возможно вследствие наличия внутренних переменных. В случае, если активизация заданной последовательности состояний невозможна, то последовательность модифицируется путем введения дополнительного цикла, в результате прохождения по которому интервал возможных значений необходимой внутренней переменной меняется. Такой процесс будет происходить до тех пор, пока не будет достигнуто необходимое значение внутренней переменной или максимальное количество итераций. В последнем случае выдается соответствующее сообщение об ошибке.

После построения теста выполняется его трансляция на необходимый язык описания аппаратуры (VHDL или Verilog HDL, в зависимости от того, какой язык выбран автором модели).

После построения теста выполняется генерация файлов макрокоманд для работы в программе Active-HDL. Необходимо добавить файлы со сгенерированными тестовыми последовательностями и файлы с макрокомандами в проект и запустить на выполнение один из файлов макрокоманд. В результате произойдет компиляция проекта и тестовых воздействий, а затем запуск программы моделирования и просмотра временных диаграмм, по которым можно делать выводы о корректности проекта.

2.2. Функции ASFTEST

Программа ASFTEST предназначена для автоматической генерации тестовых наборов. Входной информацией является содержательный граф автомата, представленный в формате ASF. Программа является частью САПР Active-HDL. В результате работы программы формируются минимизированные тестовые последовательности в виде файла на языке VHDL (Verilog) для проверки правильности (в англоязычной литературе используется термин – validation) и верификации проекта с помощью программ моделирования на языках VHDL и Verilog. Возможна генерация теста по 3 различным стратегиям, в зависимости от целей моделирования и необходимой полноты теста. Также программа выполняет анализ модели, проверяя ее корректность, генерирует описание на языке VHDL, макрокоманды для среды Active-HDL, фиксирует в файле отчета статистическую информацию.

Требования к аппаратному и программному обеспечению. Программа ASFTEST выполнена в виде консольного приложения Win32. Название исполняемого файла “asftest.exe”. Он предназначен для запуска программы в операционных системах Windows 95, Windows 98, Windows NT 4.0, а также совместимых с ними и имеющих соответствующие динамически подгружаемые библиотеки (kernel32.dll и ctrdll.dll). Исполняемый код сгенерирован для процессора семейства intel 486 и может исполняться на процессорах семейства x86, совместимых с ними и поддерживающих команды intel 486.

Требования к оперативной памяти. Программа работает с динамическим распределением памяти, поэтому требования к свободной оперативной памяти зависят от входных данных. Вся обрабатываемая программой информация (входной файл, выходные файлы, промежуточные результаты) располагается в оперативной памяти. Минимальные требования к оперативной памяти (для входного файла минимальных размеров) 200 Кбайт. Приблизительный объем необходимой оперативной памяти M в байтах можно посчитать по формуле:

$$M = 12 \cdot nstates^2 + 24 \cdot (nvar + 5) \cdot ntest,$$

где nstates – количество состояний в содержательном графе автомата; nvar – суммарное количество сигналов, констант и переменных; ntest – возможное количество тестовых наборов. Так как программа предназначена для обработки относительно небольшого числа состояний (до 100), то первое слагаемое (при максимальном числе вершин) может быть ограничено 120 000 байтов. В данной версии программы количество тестовых наборов ограничено 1000, а число переменных относительно невелико (возьмем 100 как максимальное). Получим M=24*(100+5)*1000=2 400 000. Таким образом для

большинства задач необходимое количество оперативной памяти не будет превышать 3 мегабайта, что приемлемо практически для любых компьютеров с операционной системой Windows.

Требования к свободному месту на жестком диске. Исполняемый файл занимает 224 Кбайта дискового пространства. Поскольку вся промежуточная информация хранится в оперативной памяти, то свободное место необходимо только для записи выходных данных. Так как эти данные записываются на языке VHDL (Verilog) в виде текста, то при количестве тестовых наборов ntest> 200 и переменных nvars>100, особенно с именами из большого числа букв nname>60 со значительным количеством символов для присваивания, переноса строки и отступа в начале строки nadd>40 могут занимать значительное место дисковой памяти Mdisk, которое можно приблизительно посчитать как Mdisk = ntest*(nvars*(nname+nadd)). Таким образом, Mdisk > 2 Мбайт.

3. TESTBUILDERv.2.0

Программа предназначена для автоматического проектирования тестов относительно класса ОКН цифровых проектов, описанных на языке булевых уравнений.

Функции программы:

1. Псевдослучайная генерация тестов на основе встроенных генераторов двоичных кодов с равномерным распределением нулей и единиц.

2. Детерминированная генерация двоичных тест-векторов, активизирующих одномерные логические пути в схеме.

3. Моделирование ОКН в целях определения качества полученного теста.

4. Форматирование теста в рамках стандарта языка VHDL – Testbench.

Tester.exe является консольным Win32 приложением и имеет демонстрационное назначение (исходный код, файл **Tester.cpp**). Данный модуль является примером вызова интерфейсных функций динамической библиотеки **TBuilder.dll**, реализующей основную функциональность системы, поэтому файл **TBuilder.dll** должен находиться в одном каталоге с **Tester.exe**.

Программа выполняет рекурсивную обработку выбранной схемы, до тех пор пока не будет нажата клавиша **[Esc]** (в случае нажатия другой клавиши схема будет обработана повторно). После выдачи запроса на ввод с клавиатуры (в общем случае, когда загрузка схемы закончена) схема (файл Name.sch) может быть изменена и обработана повторно.

Синтаксис:

Tester.exe <command> <Name[.sch]> [[-switch_1] ... [-switch_N]]

Command:

t – сгенерировать тест (*input*: Name.sch; *output*: Name.wap, Name.tst);

m – выполнить моделирование входных наборов (*input*: Name.sch, Name.imd; *output*: Name.omd);

g – сгенерировать псевдослучайный файл входных воздействий (Name.imd) и выполнить моделирование этих наборов, результат будет занесен в файл Name.omd и Name.tst (если указана опция **-t**, при этом если указана опция **-c**, то тест будет продолжен);

r – сгенерировать тест на основе случайных наборов (файл Name.imd не создаётся). Данный режим, по сравнению с предыдущим, имеет некоторое преимущество по быстродействию, так как не выполняет предварительную запись входных наборов на диск и последующую их загрузку, а, вместо этого, формирует входные вектора непосредственно в памяти (*output*: Name.omd, Name.tst).

Name [.sch] – имя файла описания схемы (расширение может быть опущено) и соответственно имя «проекта». Все создаваемые файлы будут иметь имя **Name** и расширение, соответствующее их типу. Для выполнения моделирования пользовательских входных наборов файл входных воздействий должен иметь имя Name.imd. **Name** может содержать полный либо относительный путь.

Switches:

-l<N> – установить предел времени (N) на обработку одной линии (единица измерения – десятая доля секунды);

-b<N> – установить размер буфера (N) векторов;

-n<N> – определить количество векторов (N), генерируемых псевдослучайным датчиком (для последующего моделирования);

-r<N> – установить предельное значение (N) на количество «возвратов» к предыдущему решению при достижении пустого решения (опустошение буфера);

-x<0/1> – определить символ (0 либо 1) для доопределения X-вых координат в процессе построения теста (режим **t**);

-c – продолжить построение существующего теста (если файл Name.tst отсутствует, то он создаётся «с нуля»);

-s – сохранять все пары активизации, включая бесполезные (не вошедшие в тест);

-z – применить статическое сжатие теста;

-a – создать / использовать файл пар активизации (Name.wap) (только режим **t**);

-t – создать / использовать файл теста (Name.tst);

-m – создать / использовать файл результатов моделирования (Name.omd);

-d – использовать файл описания схемы (Name.sch) (имеет смысл только с ключом **-p**);

-p – создать текстовый файл протокола (Name.prt). Состав протокола определяется четырьмя опциями, указанными выше. Термин «использовать» отно-

сится только к этой опции (т.е. использовать при генерации протокола).

Если в процессе обработки схемы возникают какие-либо ошибки, то соответствующие сообщения помещаются в текстовый файл error.log.

В качестве примера построения теста для схемы (рис.3) может быть следующая распечатка:

Test =

No	012345	%
1 – Vector	01X010	
Faults =	10.101	41.67
Total =	10.101	41.67
2 – Vector	100000	
Faults =	1X1101	58.33
Total =	1X1101	58.33
3 – Vector	11X111	
Faults =	00.000	41.67
Total =	XX1X0X	83.33
4 – Vector	X01001	
Faults =	0000X0	58.33
Total =	XXXXXX	100.00

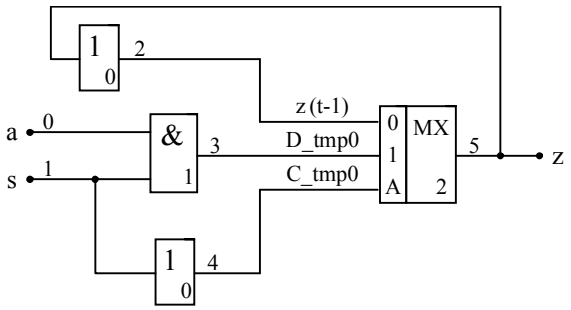

Рис. 3. Пример цифровой схемы

Программой обработано: 10 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 – 2 часа. Средняя сложность проекта – 1000 линий. Среднее время проектирования псевдослучайных тестов – 5 минут. Качество тестов – более 90 %.

4. FAULT SIMULATOR

Предназначен для моделирования ОКН цифровой схемы, описанной на функциональном уровне в виде кубических покрытий.

Задачи, решаемые программой:

1. Моделирование ОКН по кубическим покрытиям функциональных элементов.

2. Моделирование дополнений к состоянию линий схемы по кубическим покрытиям функциональных элементов.

3. Генерация алгоритмических и псевдослучайных тестов.

4. Оптимизация длины теста путем положительного приращения его качества.

5. Оптимизация количества алгоритмических генераторов путем решения задачи покрытия.

Исходное описание объекта тестирования: язык описания аппаратуры VHDL, представление схемы в виде булевых уравнений.

Результат работы программы : тест для цифрового проекта, представленный в формате VHDL (Testbench). Структура программного пакета приведена на рис. 4.

Программой поддерживается дружественный интерфейс по установке значений на входах схемы (включая установку генераторов). Вывод результатов моделирования осуществляется визуально (в цифровом виде или в виде диаграмм) и в файл формата tst, который преобразуется в формате VHDL (Testbench).

В результате моделирования списки проверяемых дополнений могут быть считаны из внутренних структур данных (программным путём) или записаны в формате tst. (предпочтительней первое, так как при переходе от списков дополнений к спискам проверяемых ОКН происходит потеря информации). Списки проверяемых ОКН могут быть выведены на экран.

При реализации програм-

Рис. 4. Система генерации тестов

мы использовался компилятор Microsoft Visual C++ 6.0 библиотеки Objective Grid 7.0 (для визуального представления данных и результатов в табличном виде) и mod.lib(библиотека, обеспечивающая функциональность системы).

Для автоматического занесения структуры схемы используется экземпляр класса CScheme, реализованный в библиотеке ManipSch.lib, который инкапсулирует файл формата sch.

4.1. Объектная модель

Классом – контейнером, предоставляющим пользователю библиотеки различные функции, является класс Modeling.

Предоставляются следующие функции:

– добавление различных определённых пользователем генераторов;

– установка\считывание значений и списков проверяемых дополнений на всех линиях;

– моделирование поведения схемы (получение значений на всех линиях по значениям на входах);

– получение списков проверяемых дополнений.

Каждый экземпляр класса gate (выходная линия примитива) содержит: значение на линии за последние 2 такта моделирования; текущий список проверяемых дополнений; номера линий, которые являются аргументами для данной линии.

Класс Calc инкапсулирует кубическое покрытие и служит для получения значения на линии по её аргументам и получения списков проверяемых дополнений [1].

Класс Csave служит для сохранения результатов моделирования в формате tst, а также для сбора статистики. Он содержит такую информацию, как процент проверенных, проверяемых ОКН; приращение проверенных ОКН на текущем такте моделирования. Также класс предназначен для работы с транзакциями – возможность сделать "мгновенный снимок" и затем восстановить это состояние.

Генераторы на входные линии спроектированы таким образом, что пользователь библиотеки легко (при помощи механизма виртуальных функций) может добавить свои собственные генераторы без перекомпиляции библиотеки.

4.2. Принцип работы

Такт моделирования – время\действие между подачей на вход модели входного набора и получением выходного. Итерация моделирования – поочерёдная установка текущих значений по всем линиям в схеме.

Как уже было отмечено, все линии содержат информацию о значении двух тактов моделирования – текущего и предыдущего. После установки значений на входах текущие значения невходовых линий присваиваются предыдущим.

Далее выполняется итерация моделирования с использованием только одного значения на каждой линии - текущего. Следовательно, какие значения будут на входах каждого элемента (по крайней мере на первой итерации моделирования), зависит также и от порядка обработки линий. В данном случае порядок обработки линии зависит от её расположения во внутреннем массиве, которое определяется расположением линии в файле sch. В этом файле линии отранжированы или почти отранжированы, что позволяет исключить наихудший вариант с точки зрения быстродействия, когда итерация моделирования осуществляется в порядке убывания ранга вершины (линии).

Такт моделирования считается оконченным, когда на двух итерациях моделирования отсутствуют изменения на линиях или количество итераций превысило 50. В последнем случае на тех линиях, значения которых изменялись за последние 2 итерации, ставится значение X.

Аналогично вводятся понятия такта и итерации установки списков дополнений.

Списки дополнений идентичны спискам ОКН в случае моделирования комбинационной схемы.

Различие между списками дополнений и ОКН заключается в том, что для первых необходимо знать не только тот факт, что ОКН проявляется на какой-то линии, но и то, на каком такте моделирования (при каком входном векторе) это происходит.

Программно это реализовано следующим образом. Список дополнений (класс List) состоит из 2-х списков. Первый оптимизирован по быстродействию, но позволяет хранить только индексы линий – в нём содержится список дополнений, полученных на текущем такте моделирования. Операции над вторым списком реализованы сравнительно медленно, но он позволяет хранить списки проверяемых дополнений за предыдущие такты моделирования. Это создаёт определённые неудобства при операциях над ним, а также при переходе от одного такта к другому, когда может потребоваться перевод всего "быстрого" списка в "медленный". С теоретической точки зрения проблемой является "разбухание" списков дополнений после уже нескольких десятков тактов моделирования в схеме, у которой обратные связи заводятся на элементы XOR или им подобные.

После установки значений на выходах и запуска соответствующей процедуры (функции-члена) начинается такт установки списков дополнений.

Критерием окончания такта является, как и при установке значений, неизменность списка дополнения за 2 итерации.

Итерация установки списков дополнения учитывает возможность генерации списков. Если результирующий список меньше списка, который был на предыдущей итерации, то формируется список удаления, который влияет на установку нового списка и действителен до конца такта [1].

Важным моментом является то, что в качестве аргументов вентиля (линии) при вычислении списка берётся текущее значение, если эта линия была обработана раньше, и значение на предыдущем такте моделирования, если эта линия ещё не обработана (на текущей итерации). Другими словами, это означает то, что при вычислении списков проверяемых дополнений линии, которые получаются с большим рангом, считаются обратными связями (только при вычислении списков!).

Сказанное выше можно проилюстрировать блок-схемой, представленной на рис. 5.

4.3. Генераторы

В классе Modeling, который обеспечивает моделирование поведения схемы, а предусмотрен такой режим работы, при котором пользователь библиотеки имеет возможность не просто подавать на входы схемы какие-либо значения, но и устанавливать генераторы, которые позволяют автоматизировать процесс установки значений на входах схемы. Генераторы могут быть добавлены в любой момент работы со схемой.

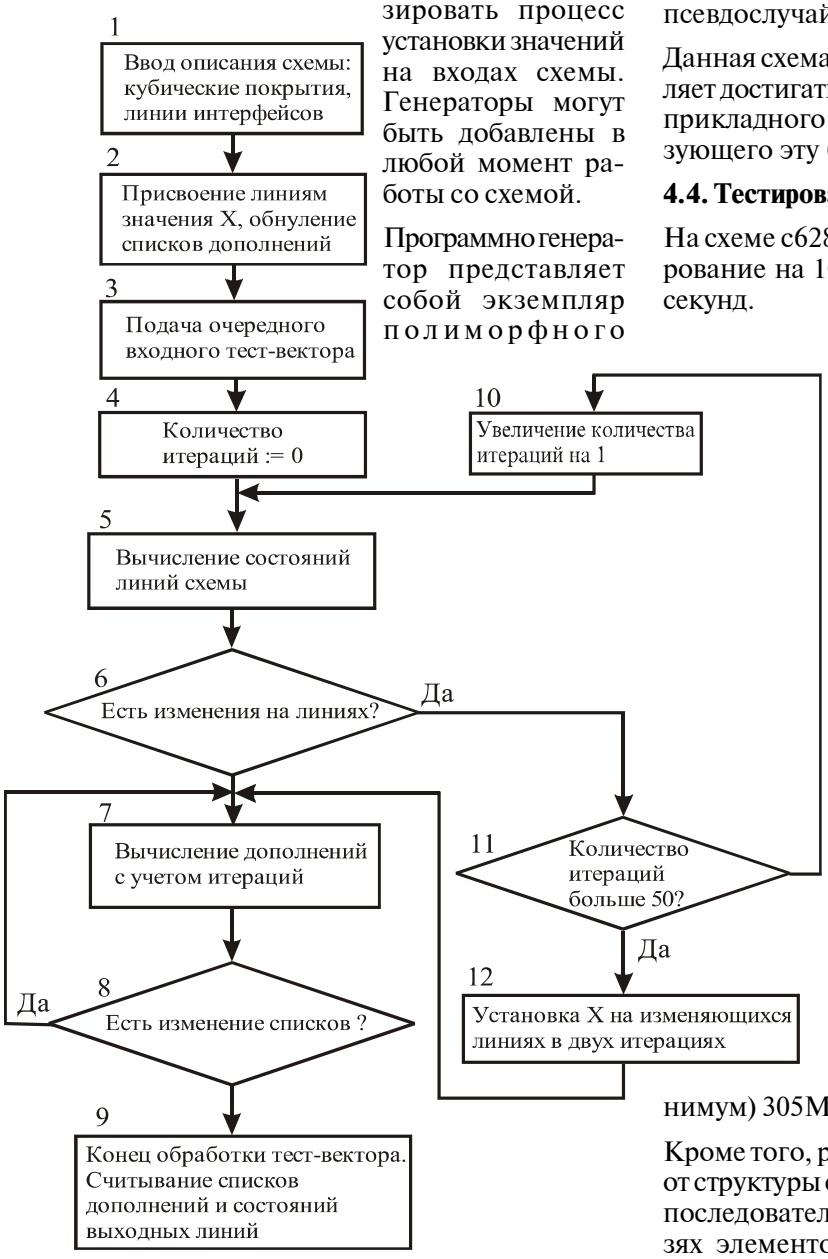

Рис. 5. Процедура моделирования тест-вектора

Программно генератор представляет собой экземпляр полиморфного класса, который является потомком класса Modeling: _gen_test. В классе Modeling реализован список генераторов, в который пользователю можно добавлять свои собственные. Генераторы вызываются в порядке помещения в список (генератор должен реализовать функцию Generate) и может изменять значения всех входов(на практике следует придерживаться правила – каждый генератор "отвечает" только за "свои" входные линии во избежание путаницы). Кроме того, из генератора можно управлять вызовом всех генераторов, помещённых до него – если функция-член Generate возвратит false, то считается, что необходимо обращаться к генераторам более высокого приоритета(помещённых раньше), в противном случае такого обращения не происходит. Кроме того, каждый генератор должен предоставить функции сброса соответствующих значений на входах(Reset) и проверки значений на входах (Verify).

В библиотеке реализованы генераторы кода Грея, псевдослучайного появления 0 и 1 и CLK.

Данная схема генерации входных значений позволяет достигать требуемой гибкости при реализации прикладного программного обеспечения, использующего эту библиотеку.

4.4. Тестирование

На схеме c6288, содержащей 2448 линий, моделирование на 100 векторах было произведено за 75 секунд.

На схеме c7552, содержащей 3722 линии моделирование на 50 векторах было произведено за 26 секунд.

Потребляемая оперативная память в обоих случаях не превышала 6Мбайт. Тестирование проводилось на CELERON500.

Результаты, полученные на комбинационных схемах не отражают специфику решённых задач, поскольку списки дополнений могли бы быть заменены на списки проверяемых ОКН. При оценке верхнего предела величины обрабатываемых схем необходимо учитывать, что на каждую линию в программе должно быть выделено не менее N/8 байт, где N-число линий в схеме. Подставляя число линий с 7552, получаем теоретическую минимальную потребность в памяти около 1.7Мбайт, что согласуется с практическими данными. Следовательно, для схемы с числом линий около 50000 потребуется (минимум) 305Мбайт.

Кроме того, результаты(скорость) зависят также и от структуры схемы(см. результаты), от её "степени последовательности", от наличия в обратных связях элементов типа XOR, которые приводят к "протаскиванию" списков проверяемых дополнений на следующие такты моделирования.

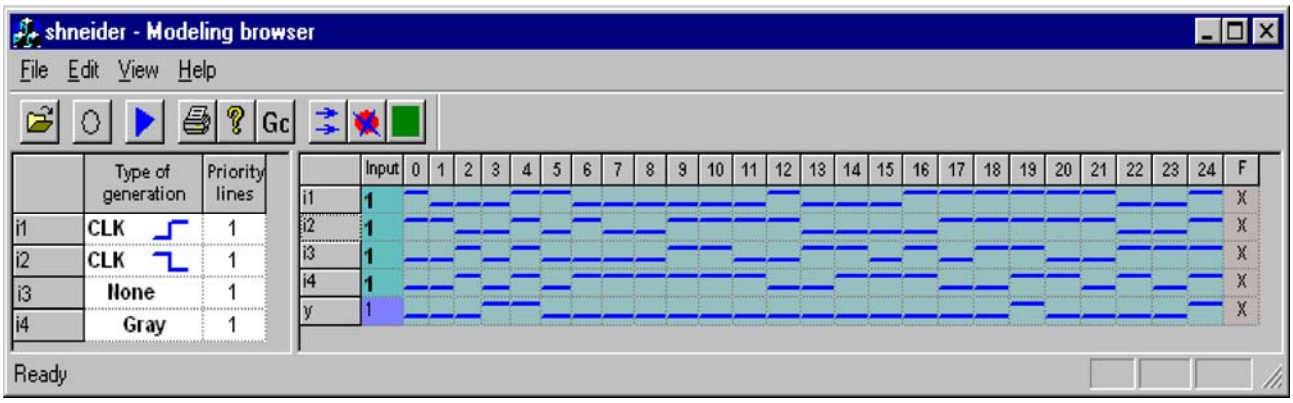

Рис. 6. Окно результатов моделирования

Была написана демонстрационная программа, которая использует все возможности библиотеки Fault Simulation.

Программа позволяет конечному пользователю ознакомиться с возможностями библиотеки и результатами её работы.

В качестве описания схемы используется файл формата sch. Выходная информация преобразуется (с потерей информации) в формат представления результатов моделирования и списков ОКН - tst.

4.5. Интерфейс программы моделирования

Интерфейс программы согласуется с концепцией SDI (Single Document Interface). Окно программы разделено на 2 части. В первой находятся названия входных линий, с возможностью установки на них различных генераторов (код Грея, CLK по фронту, CLK по спаду, синхронная и асинхронная установка псевдослучайных значений). Во второй части находятся результаты моделирования. Управление списком отображаемых линий осуществляется при помощи диалогового окна, появление которого инициируется кнопкой Observation (Ctrl +'+').

Таблица результатов моделирования (рис.6) может отображать следующие данные (управляется при помощи меню View или соответствующих клавиш):

– цифровые значения на линиях за предыдущие такты моделирования (Ctrl +d);

– значения на линиях за предыдущие такты моделирования в виде диаграмм (Ctrl +d);

– значения проверяемых ОКН на каждом такте (Ctrl +W);

– последний столбец всегда показывает проверенные ОКН.

Независимо от режима изображения доступна следующая информация, которая появляется в виде подсказки (hint)при наведении курсора мыши на соответствующий столбец:

– процент проверяемых ОКН и процент ОКН, которые были добавлены к проверенным ОКН на данном тестовом векторе (при наведении на столбец соответствующего входного вектора);

– процент проверенных ОКН всеми тестовыми векторами (при наведении на последний столбец с ОКН).

Все результаты, отображаемые в таблице результатов, автоматически заносятся в tst – файл.

Сброс результатов моделирования происходит по нажатию кнопки Reset.

Пользователь может в конкретный момент времени работать в одном из режимов, при этом возможно свободное переключение между ними с сохранением предыдущих результатов:

1.Непосредственная установка значений на входных линиях схемы – кнопка Automatic Code выключена (отжата). Устанавливается по умолчанию. Пользователю необходимо из выпадающего списка, соответствующего каждой входной линии, выбрать требуемые значения для каждой входной линии и нажать кнопку Simulation или клавишу F7.

2. Генерация значений для требуемых линий – заданное число тактов. Для переключения в этот режим необходимо нажать кнопку Automatic Code и для установки генераторов выбрать их в таблице в левой части окна для каждого входа. Далее необходимо нажать кнопку Automatic Generate Test, после чего в появившемся диалоговом окне ввести требуемое число векторов и определить необходимые параметры.

3. Пошаговое выполнение второго режима. Кнопка Automatic Code нажата и для генерации одного вектора используется кнопка Simulation или клавиша F7.

4. Последовательное применение различных генераторов (не затрагивая линии, на которые поставлены генераторы CLK) и выбор требуемых результатов. Для работы в этом режиме необходимо установить в таблице генераторов (левая часть окна) генераторы CLK на требуемые линии (если нужно), а затем нажать кнопку Tests Generators, что вызовет появление диалогового окна с предложением выбрать типы используемых генераторов и максимальное количество тестовых векторов на моделирование с помощью каждого генератора. После вывода результатов (рис. 7) можно выбрать одновременно несколько вариантов, которые будут отображены в таблице и записаны в файл . При этом необходимо учитывать (для последовательностных схем), что моделирование при помощи каждого генератора производилось независимо (каждый раз

начиная с текущего такта), а при выборе нескольких типов результатов они будут промоделированы второй раз, но уже последовательно.

Таблица генераторов находится в левой части окна и предназначена для установки генераторов на входные линии. Пользователь имеет возможность управлять приоритетом линии в генераторах (определять, как часто её значение будет изменяться). Значения линий с меньшим приоритетом (столбец Priority lines) изменяются чаще. Этот столбец имеет другой смысл при установке на линии псевдослучайного генератора. В последнем случае в это поле вводится "затравка" для генератора псевдослучайных чисел.

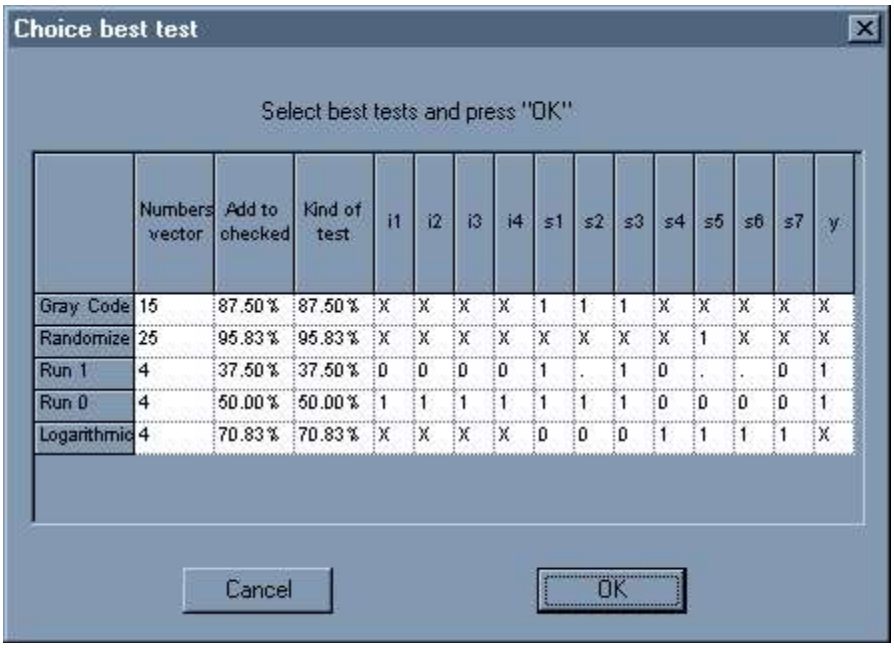

Рис. 7. Окно выбора оптимального теста

Выводы

Предложенная система генерации тестов может быть использована для верификации цифровых проектов, реализованных в стандарте VHDL.

Литература: 1. *Хаханов В.И.* Кубическое моделирование неисправностей и генерация тестов для цифровых систем. В кн.: Ежегодный отчет ХТУРЭ. 1999-2000. С.139-146.2. *Хаханов В.И., Ковалев Е.В., Ханько В.В., Мехеди М.М.* Система генерации тестов для проектирования цифровых автоматов в среде ACTIVE-HDL. АСУ и приборы автоматики. Харьков: ХТУРЭ. Вып.111.С. 5-22. 3. *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.

Поступила в редколлегию 16.06.2000

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

Хаханов Владимир Иванович, д-р техн. наук, профессор кафедры АПВТ ХТУРЭ. 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.

Рустинов Владимир Алексеевич, канд. техн. наук, зам. директора ХАЭР. Научные интересы: автоматизация проектирования и диагностика вычислительных систем. Увлечения: теннис, горные лыжи, плавание. Адрес: Украина, 61057, Харьков, пер. Театральный, 11/13, тел. 43-11-83.

Горбунов Дмитрий Михайлович, студент ХТУРЭ. Научные интересы: объектное моделирование, разработка программного обеспечения. Увлечения: поэзия. Адрес: Украина, 61057, Харьков, пер. Театральный, 11/13, тел. 43-11-83.

Ковалёв Евгений Владимирович, аспирант кафедры АПВТ ХТУРЭ. Научные интересы: техническая диагностика компьютерных устройств и систем. Увлечения: иностранные языки. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26.

Масуд М.Д. Мехеди, аспирант кафедры АПВТ ХТУРЭ. Научные интересы: диагностика устройств и сетей. Увлечения: шахматы, футбол, теннис. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26.

Хак Х.М. Джахирул, аспирант кафедры АПВТ ХТУРЭ. Научные интересы: диагностика устройств и сетей. Увлечения: шахматы, футбол, теннис. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26.