1. При коректному формуванні таблиць прив'язки та зв'язків можна обійти недоліки, які були зроблені при проектуванні бази даних;

2. Даний підхід гарантує вірність побудови складних запитів та підвищує оперативність отримання необхідної інформації;

3. Виключає необхідність для користувача володіння спеціальними інформаційно-технічними знаннями та дозволяє зосередитись на вирішенні задач управління.

ПЕРЕЛІК ПОСИЛАНЬ

1. *Голышев Л. К.* Сложные системы с развитой функцией информационно-аналитической поддержки управления. Элементы теории, методологии, практики / Л. К. Голышев. – К. : 2001. – 253 с.
2. *Конноли Т.* Базы данных: проектирование, реализация и сопровождение. Теория и практика / Т. Конноли, К. Бегг, А. Страчан. – М. : «Вильямс», 2000. – 1120 с.
3. Microsoft SQL Server 2008 [Електронний ресурс]. Електрон. текстові дані. – Режим доступу: www.microsoft.com/SQL/default.mspx. – Wednesday, 15 April, 2009.
4. Microsoft Office Access 2007 [Електронний ресурс]. Електрон. текстові дані. – Режим доступу: http://office.microsoft.com/ru-ru/access/default.aspx. – Wednesday, 15 April, 2009.
5. *Праг К. Н.* Microsoft Access 2000. Библия пользователя / К. Н. Праг, М. Р. Ирвин. – М. : Диалектика, 2001. – 1039 с.
6. 1С: Предприятие [Електронний ресурс]. Електрон. текстові дані. – Режим доступу: http://www.1c.ru/. – Wednesday, 15 April, 2009.
7. Microsoft SQL Server 2008: Reporting Services [Електронний ресурс]. Електрон. текстові дані. – Режим доступу: http://www.microsoft.com/sqlserver/2008/en/us/reporting.aspx. – Wednesday, 15 April, 2009.
8. SAP BusinessObjects. Crystal Reports [Електронний ресурс]. Електрон. текстові дані. – Режим доступу: http://www.sap.com/solutions/sapbusinessobjects/sme/reporting/crystalreports/index.epx. – Wednesday, 15 April, 2009.
9. The VSReport Designer. [Електронний ресурс]. Електрон. текстові дані. – Режим доступа: http://help.globalscape.com/help/eft5/arm/the_vsreport_designer.htm. – Wednesday, 15 April, 2009.
10. Flexible reporting to leverage your Business Intelligence capabilities [Електронний ресурс]. Електрон. текстові дані. – Режим доступу: http://www.perpetuumsoft.com/Product.aspx?lang=en&pid=21. – Wednesday, 15 April, 2009.
11. *Касьянов В. Н.* Графы в программировании: обработка, визуализация и применение / В. Н. Касьянов, В. А. Евстигнеев. – СПб. : БХВ-Петербург, 2003. – 1104 с.
12. *Андерсон Дж. А.* Дискретная математика и комбинаторика: Пер. с англ. / Дж. А. Андерсон – М. : Вильямс, 2003. – 960 с.

Надійшла 23.02.2009
Після доробки 11.05.2009

Предложен подход к получению информации из базы данных и ее представлению в терминах предметной области. В состав этого подхода входят метод привязки к предметной области и метод построения запроса к базе данных в терминах предметной области.

The approach to information obtaining from database and its representation in the terms of data domain was proposed. This approach is represented by the method of data domain binding and the method of query construction in data domain terms.

УДК 681.326

А. С. Шкиль, Е. Е. Сыревич, Д. Е. Кучеренко, Самер Альмадхоун

МЕТОД ОБРАТНОГО ПРОСЛЕЖИВАНИЯ ДЛЯ ПОИСКА ОШИБОК ПРОЕКТИРОВАНИЯ В HDL-КОДЕ

В данной статье были рассмотрены методы поиска ошибок проектирования в неструктурированном HDL-коде. Разработан метод обратного прослеживания. Проведен эксперимент над HDL-моделью цифрового устройства с использованием данного метода.

1 ПОСТАНОВКА ЗАДАЧИ

Верификация цифровых проектов, то есть аппаратных или встроенных аппаратно-программных систем, описанных на языке описания аппаратуры (Hardware Description Language – HDL), является важной задачей в процессе проектирования цифровых устройств. Часто более 70 % времени разработки затрачивается на поиск и исправление ошибок в проекте. *Целью* данной работы является разработка методов поиска дефектов/ошибок проектирования в неструктурированном HDL-коде, позволяющих сократить время проведения диагностического эксперимента и уменьшить длину диагноза. Исходя из вышесказанного, необходимо решить *задачу* адаптации метода обратного прослеживания при верификации HDL-модели и провести диагностический эксперимент по поиску ошибок проектирования в рамках верификации [1].

2 АНАЛИЗ МЕТОДОВ ПОИСКА ДЕФЕКТОВ ПРИМЕНИТЕЛЬНО К МОДЕЛЯМ ЦИФРОВЫХ УСТРОЙСТВ

В данной работе аналогично понятию дефект используется понятие ошибки проектирования. Диагно-

стический эксперимент (ДЭ) над HDL-кодом осуществляется в два этапа [2]. На первом этапе проводится безусловный эксперимент. Если результат хотя бы на одном внешнем выходе не совпадает с эталоном, выполняется второй этап ДЭ. Проблема диагностирования технического состояния модели цифрового устройства заключается в минимизации области существования ошибок проектирования относительно наперед заданного класса, которая называется множеством эквивалентных дефектов, при допустимых параметрах длины теста и числа наблюдаемых выходов. Чем больше длина теста и количество наблюдаемых выходов, тем выше разрешающая способность алгоритма диагностирования, которая должна обеспечивать определение неисправного элемента только по фактическим реакциям без доступа к контрольным точкам (КТ). При фиксации параметров длины теста и числа наблюдаемых выходов, что имеет место в алгоритмах поиска дефектов, основанных на анализе таблиц неисправностей, можно повысить глубину диагностирования путем использования структуры связей входо-выходных элементов (операндов или операторов) модели цифрового устройства. Максимально адекватной реакцией модели на тест-вектор при наличии дефекта является вектор-строка значений наблюдаемых элементов модели. В качестве примера можно выделить метод тестового диагностирования ошибок проектирования в модели ЦУ на основе использования многозначных таблиц неисправностей (МТН) и анализе структуры объекта.

В начальную область подозреваемых ошибок проектирования входят все функциональные элементы (ФЭ) либо операндовые вершины. ДЭ проводится исходя из предположения, что в коде имеется одиночная ошибка проектирования, результирующая в неисправное значение на некоторой операндовой вершине, которая является либо контрольной точкой, либо внешним выходом. Основной принцип, лежащий в основе структурных методов поиска ошибок проектирования, следующий. Если в очередной контрольной точке результат элементарной проверки отрицателен, то в область подозреваемых ошибок на очередном шаге алгоритма входят сам ФЭ и все его предшественники. Если результат проверки положителен, то все предшественники предполагаются исправными, а подозреваемая ошибка произошла среди остальных ФЭ области подозреваемых ошибок предыдущего шага алгоритма. Если транспортирование ошибки проектирования на внешний выход невозможно, то граф необходимо разбить на подграфы по принципу существования активизации. Причиной невозможности активизации может быть либо так построенный код в силу ряда причин, либо в коде имеется ошибка (предполагаемое место ошибки – подграф его предшественников).

Стандартные проверяющие тесты неадекватны по нескольким причинам [3]. Основной из них является то, что в основе построения проверяющих тестов лежат понятия транспортирования и активизации пути, по которому неисправность транспортируется на внешний выход схемы. А это не всегда представляется возможным сделать, так как в HDL-коде функциональные неисправности могут быть замаскированы дальнейшими вычислениями таким образом, что ошибка не будет наблюдаться на внешнем выходе. Например, ошибка на выходе любого функционального элемента не транспортируется на внешнем выходе, если среди его преемников есть оператор логического ИЛИ с константной 1 на одном из входов. Для декомпозиции исходного графа используются контрольные точки (аналогично контрольным точкам при генерации тестов, которые позволяли «разбить» путь активизации и определить границы подграфов). Данные контрольные точки будут выходами каждого из подграфов. Таким образом, ошибка проектирования транспортируется на внешний выход каждого подграфа. Существуют два типа контрольных точек, используемых при поиске места ошибки. Контрольные точки первого рода – сигнал (переменная) модели, эталонные значения которых известны из спецификации. Контрольные точки второго рода – сигнал (переменная) модели, значения которых наблюдаемы, но до начала эксперимента неизвестны. Структуризация исходной модели на HDL выполняется путем выделения многовыходового подграфа, который порождается (активизируется) в результате подачи на модели ЦУ специализированного теста. Входами подграфа являются внешние входы (операндовые вершины, не только физические), а выходами – КТ первого рода. По результатам проведения первого этапа диагностического эксперимента (подачи теста для определения наличия ошибки проектирования в описании) формируется вектор результатов экспериментальной проверки, который фиксирует равенство эталонной и

```
Entity SCH is Port (X1,X2,X3,X4,X5,X6,X7, X8: in bit; O15: out bit); End;
Architecture BEH of SCH is signal S9, S10, S11, s12, s13, S14: bit; begin
S9<=X1 and x2; S10<=X3 and x4;  S11<=X5 and x6;  S12<=X7 and X8;
S13<=S9 and S10; S14<= S11 and S12;   O15<=S13 nand S14; End;
```

Рисунок 1 – Пример HDL-модели цифрового устройства

экспериментальной реакции. Другими словами, определяется, на какие из множества выходов оказывает влияние ошибка проектирования.

3 МЕТОД ОБРАТНОГО ПРОСЛЕЖИВАНИЯ

В спецификациях HDL-моделей реальных цифровых устройств КТ первого рода, в которых известны значения эталонных сигналов до начала написания кода, как правило, достаточно немного. Поэтому локализация до подграфа с выходной КТ первого рода является недостаточной. Желательно локализацию ошибочных операторов проводить до КТ второго рода, но эталонов нет и взять их негде. Выходом из этого противоречия является применение в подграфе, где выходом является КТ первого рода, метода обратного прослеживания, который первоначально был ориентирован поиск константных неисправностей в цифровых схемах в условиях отсутствия эталонов во внутренних точках. Классический метод обратного прослеживания, который схемотехники иногда называли методом «доискивания», основывался на следующих положениях: есть устройства с доступными внутренними линиями и его структурно-функциональная модель; есть условие существенности для каждого элемента; есть тест, который активизирует пути в модели; есть эталонные реакции на внешнем выходе модели [4].

Для использования этого метода при «доискивании» в классе эквивалентных ошибок при диагностике HDL-кода, необходима модификация метода. Принимаются следующие положения:

1. Класс неисправности – замена функционального элемента;

2. В качестве тестов используются различающая последовательность (РПС) для каждого ФЭ;

3. Неисправным (ошибочным) признается ФЭ, на входах которого наблюдалось последнее несовпадение с эталоном.

Алгоритм выполнения «доискивания»:

1. Выполняется моделирование тестов на реальной модели до внешнего выхода (КТ);

2. Если на выходе обнаружено несовпадение с эталоном, то выполняется обратное прослеживание;

3. Обратное прослеживание идет до тех пор, пока результат не равен ∅, путем пересечения текущего вектора с условиями существенности функционального элемента;

4. Из непустых пересечений формируется подмножество подозреваемых элементов.

Пусть имеется некоторый ошибочный HDL-код (рис. 1), реализующий схему, представленную на рис. 2. Ошибка в коде: в операторе S12 функция **or** заменена на **and.**

Для этого фрагмента кода построен тест 11011100 и эталонное значение 0 на внешнем выходе (сигнал O15). В результате подачи теста на HDL-модель на ее выходе получена 1, что не совпадает с эталоном. Выполним обратное прослеживание, отмечая его результаты для наглядности на фрагменте цифровой схемы (рис. 2), эквивалентной приведенному коду.

Наиболее эффективной формой для реализации обратной импликации является кубическая форма представления моделей примитивов, поэтому для наглядности рассмотрим применение метода «доискивания» в кубической форме. Если рассматривать условие существенности в кубической форме, то они подобны D-покрытиям, но не учитывают направление перехода (инверсию). Покрытия существенности для элементов (операторов) AND и OR приведены на рис. 3.

Процедура выполнения алгоритма обратного прослеживания для фрагмента HDL-кода с предопределенными операторами параллельного назначения сигналов (ПНС) подобна процедуре обратной импликации для комбинационной схемы, при условии, что в качестве примитивных элементов выступают покрытия существенности соответствующих ПНС. Первоначально выполняется моделирование теста на реальном (неисправном) коде и результат моделирования записывается в первую строку [5]. Результат из спецификации не совпадает с эталоном, поэтому выполняется обратное прослеживание путем пересечения

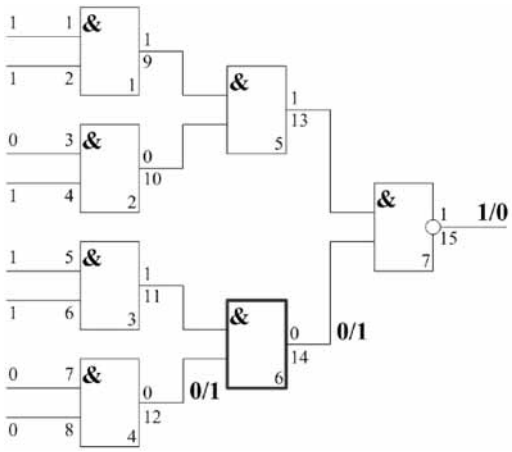


Рисунок 2 – «Доискивание» на примере структурно-функциональной схемы

X1	X2	X3	Y
Z	1	1	Z
1	Z	1	Z
1	1	Z	Z

X1	X2	X3	Y
Z	0	0	Z
0	Z	0	Z
0	0	Z	Z

Рисунок 3 – Покрытия существенности элементов AND и OR

$$C = X1 \cdot X2 \vee X3 \cdot X4$$

X1	X2	X3	X4	C
0	X	0	X	0
0	X	X	0	0
X	0	0	X	0
X	0	X	0	0
1	1	X	X	1
X	X	1	1	1

Рисунок 4 – Решение равнения по логическому условию

X1	X2	X3	X4	D1	D2	Y
0	X	0	X	X	Z	Z
0	X	X	0	X	Z	Z
X	0	0	X	X	Z	Z
X	0	X	0	X	Z	Z
1	1	X	X	Z	X	Z
X	X	1	1	Z	X	Z

Рисунок 5 – Покрытие существенности оператора IF( )

полученного вектора с покрытиями существенности элементов модели. Пересечения выполняются до тех пор, пока результат не будет равен ∅. Процедура выполнения обратного прослеживания приведена в табл. 1.

Таким образом, неисправным является элемент, у которого входом является линия 12, т. е. элемент 6, который был ИЛИ, а стал И, что и требовалось доказать. Каждый специализированный тест порождает многовыходной подграф, входами которого являются внешние входы (либо КТ первого рода), а выходами внешними выходами (либо КТ первого рода). Кроме операторов ПНС в HDL-коде присутствуют последовательные операторы. Особый интерес при выполнении обратного прослеживания представляют условные операторы (*IF, CASE*), т. к. они находятся на стыке данных и управления. Рассмотрим оператор *IF*: *IF ((x1 and x2) or (x3 and x4)) THEN y<=d1 ELSE y<=d2*. Оператор *IF*(), реализующий двухвыходовую условную вершину, состоит из логического условия *C* и двух выполняемых операторов ПНС или присваивания значений переменных. Логическое условие *C* в операторе *IF* (C) – это уравнение, решение в кубической форме представлено на рис. 4.

Таким образом, оператор *IF*() может быть представлен моделью мультиплексора с двумя информационными входами, группой управляющих входов и одним (двумя) выходами (два выхода будет в случае, если назначение выполняется для разных сигналов). Покрытие существенности оператора *IF( )* представлено на рис. 5.

Кроме оператора *IF( )* к условным операторам можно отнести и оператор *CASE( ),* который реализует многовыходовую условную вершину. Фрагмент HDL-кода с оператором *CASE( )* представлен на рис. 6.

Модель оператора *CASE( )* в форме мультиплексора и его покрытие существенности представлено на рис. 7.

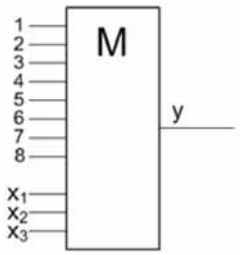
К операторам ПНС относятся также арифметические операторы, выполняемые, как правило, над многоразрядными операторами. Если рассмотреть операцию сложения одноразрядных операндов $S = A + B$, то схемной реализацией ее является одноразрядный двувходовой сумматор. Его закон функционирования и фрагмент покрытия существенности приведен на рис. 8. Следует отметить, что покрытие существенности строится динамически на основании закона функционирования сумматора. Имея покрытия существенности для одноразрядного сумматора с учетом процедур обратной импликации через арифметические операнды, легко строятся процедуры обратного продвижения существенности через многоразрядные арифметические операции. Отметим, что для арифметических операций корректными могут быть только одномерные по входам покрытия существенности.

Таблица 1 – Процедура обратного прослеживания

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	1	0	1	1	1	0	0	1	0	1	0	1	0	1	результат из спецификации 0
														z	линия 15 сущ.
												1	z	z	$\cap \neq \varnothing$
												z	1	z	$\rightarrow \varnothing$
1	1	0	1	1	1	0	0	1	0	1	0	1	z	z	линия 14 сущ.
										z	1		z		$\rightarrow \varnothing$
										1	z		z		$\cap \neq \varnothing$
1	1	0	1	1	1	0	0	1	0	1	z	1	z	z	линия 12 сущ.
						z	1				z				$\rightarrow \varnothing$
						1	z				z				$\rightarrow \varnothing$

```
variable xxx : in std_logic_vector(2 downto 0);
        case xxx is
            when «000»=> Y<=D1;        when «001»=> Y<=D2;
            when «010»=> Y<=D3;        when «011»=> Y<=D4;
            when «100»=> Y<=D5;        when «101»=> Y<=D6;
            when «110»=> Y<=D7;        when «111»=> Y<=D8;        end case;
```

Рисунок 6 – Фрагмент HDL-кода с оператором CASE( )



1	2	3	4	5	6	7	8	X1	X2	X3	Y
Z	X	X	X	X	X	X	X	0	0	0	Z
X	Z	X	X	X	X	X	X	0	0	1	Z
X	X	Z	X	X	X	X	X	0	1	0	Z
X	X	X	Z	X	X	X	X	0	1	1	Z
X	X	X	X	Z	X	X	X	1	0	0	Z
X	X	X	X	X	Z	X	X	1	0	1	Z
X	X	X	X	X	X	Z	X	1	1	0	Z
X	X	X	X	X	X	X	Z	1	1	1	Z

Рисунок 7 – Модель CASE( ) в форме мультиплексора и его покрытие существенности

A_i	B_i	C_{i-1}	S	C_i	A_i	B_i	C_{i-1}	S	C_i
0	0	0	0	0	Z	0	0	Z	0
0	1	0	1	0	0	Z	0	Z	0
1	0	0	1	0	0	0	Z	Z	0
1	1	0	0	1	Z	1	1	Z	1
0	0	1	1	0	1	Z	1	Z	1
0	1	1	0	1	1	1	Z	Z	1
1	0	1	0	1	Z	0	1	Z	Z
1	1	1	1	1	0	Z	1	Z	Z

Рисунок 8 – Закон функционирования и фрагмент покрытия существенности одноразрядного сумматора

В качестве примера рассмотрим операцию сложения двух четырехразрядных операторов $S(4) = A(4) + B(4)$ на примере $A = 5$, $B = 6$, $S = 5 + 6 = 11$. В двоичном виде $S = A + B = 0101 + 0110 = 1011$. Если $S = 10Z1$ (символ существенности находится в третьем выходном разряде), то существует только одно корректное решение при $A = 0101$ и $B = 01Z0$. В качестве проверки рассмотрим $S = A + B = 0101 + 01Z0 = 10Z1$, при условии, что $Z = \{0, 1\}$. Приведенный пример подтверждает возможность выполнения обратной импликации символа Z через многоразрядные арифметические операторы. Таким образом, имея покрытия существенности для операторов ПНС и условных операторов можно выполнять обратное прослеживание для любых фрагментов HDL-кода.

ЗАКЛЮЧЕНИЕ

Методы поиска дефектов (ошибок проектирования) для верификации HDL-моделей позволяют не только говорить о наличии ошибки проектирования, но и точно определить место ее возникновения (локализовать дефект). В работе были усовершенствован метод обратного прослеживания, который применим для «доискивания» в подграфах, не имеющих внутри себя контрольных точек первого рода.

ПЕРЕЧЕНЬ ССЫЛОК

1. *Сыревич Е. Е.* Верификация моделей цифровых устройств, представленных на языках описания аппаратуры: дисс. канд. техн. наук: 05.13.12 / Сыревич Евгения Ефимовна. – Харьков, 2007. – 176 с. – Библиогр.: с. 82–130.
2. Структурный метод поиска ошибок проектирования в моделях радиоэлектронных устройств на HDL: тез. докл. Третьего Международного радиоэлектронного форума «Прикладная радиоэлектроника. Состояние и перспективы развития» (октябрь, 2008) / отв. ред. Бондаренко. – Харьков : 2008. – 48–51 с.
3. *Шкиль А. С.* Реализация процедур импликации на графовой структуре / Шкиль А. С., Чегликов Д. И., Зинченко Д. Е. // Журн. радиоэлектронные компьютерные системы. – 2006. – Вып. 6. – С. 172–177.
4. *Abramovici M.* Digital System Testing and Testable Design /Abramovici M., Breuer M., Friedman A. – New York : Computer Science Press, 1998. – 652 p.
5. Test Generation for VHDL Descriptions Verification: Proceedings of IEEE East-West Design & Test Workshop (September, 2005). – Odessa : 2008. – Pp. 191–195 p.

Надійшла 16.03.2009
Після доробки 18.05.2009

У даній статті були розглянуті методи пошуку помилок проектування в неструктурованому HDL-коді. Був розроблений метод зворотнього простежування. Був проведений експеримент над HDL-моделлю цифрового пристрою з використанням даного методу.

In this paper the method of design error searching in unstructured HDL-code was considered. The backtracing method was developed. An experiment on HDL-model of digital device that uses this method was carried out.