КОМПЬЮТЕРНАЯ ИНЖЕНЕРИЯ И ТЕХНИЧЕСКАЯ ДИАГНОСТИКА

УДК681.326

ПОИСК ОШИБОК ПРОЕКТИРОВАНИЯ В ПОВЕДЕНЧЕСКИХ HDL-МОДЕЛЯХ МЕТОДОМ ОБРАТНОГО ПРОСЛЕЖИВАНИЯ

ШКИЛЬ А.С.

Предлагается метод поиска ошибок проектирования в HDL-моделях цифровых устройств поведенческого стиля описания. В качестве ошибок проектирования рассматривается замена операндов в арифметических или логических выражениях. Для повышения глубины поиска ошибок проектирования применяется метод обратного прослеживания в эквивалентных схемах функционально-блочного уровня с использованием кубического исчисления. Приводится пример реализации метода для фрагмента VHDL-модели арифметико-логического устройства.

1. Введение и постановка задачи

Основными задачами развития рынка современной микроэлектроники являются снижение стоимости и сокращение времени проектирования, что достигается совершенствованием технического, информационного и программного обеспечения систем автоматизированного проектирования радиоэлектронной аппаратуры (САПР РЭА). Наиболее сложным и затратным этапом в современном цикле проектирования цифровых устройств (ЦУ) является функциональная верификация – процесс обнаружения, локализации и устранения ошибок в системной модели относительно спецификации, на что затрачивается до 70% общего времени проектирования. Основной формой описания проектов цифровых устройств в САПР РЭА являются языки описания аппаратуры – Hardware Description Language (HDL), поэтому объектом верификации есть HDL-код модели ЦУ, написанной на языке описания аппаратуры. Особенный вес приобретает верификация HDL-моделей на ранних стадиях проектирования, что позволяет сократить время проектирования и снизить расходы на создание цифрового изделия в целом. Важной составной частью процесса верификации HDL-моделей является поиск и исправление ошибок проектирования при фиксации несоответствия между результатами моделирования HDL-кода и спецификации на проектируемое устройство, что обусловлено как неточностью спецификации, так и человеческим фактором при написании HDL-кода.

Структура HDL-модели ЦУ зависит от формы и выбранного стиля языкового описания модели. На основании спецификации строится HDL-код модели и для него составляется список ошибок проектирования. Под ошибкой проектирования понимается ошибка в HDL-операторе, не относящаяся к классу синтаксических и нарушающая алгоритм функционирования модели устройства, заданный спецификацией. Далее рассматриваются типы ошибок проектирования: «замена оператора» (логического или арифметического) и «замена операнда» (в операторах назначения или в условных операторах).

Одной из главных задач при поиске ошибок проектирования в HDL-моделях ЦУ является отсутствие эталонной модели. При диагностировании аппаратной реализации ЦУ всегда присутствует схемная или аналитическая (табличная) модель устройства, по которой можно вычислить эталонные реакции в любой точке схемы, или имеется в наличии идеальное работоспособное устройство, эталонные реакции для которого можно получить в ходе физического эксперимента. При диагностировании HDL-модели, как правило, имеется только код, правильность которого должна быть проверена в ходе диагностического эксперимента (ДЭ), и спецификация, по которой данный код составлялся. Проблема получения эталонов при функциональной верификации состоит в том, что спецификация обычно неформальная (нет однозначного соответствия между входными воздействиями и выходными реакциями), является неполной (не все режимы, реализованные HDL-коде, описаны), что часто не позволяет получить эталонные реакции в явном виде.

Модели на языках описания аппаратуры обладают свойством двойственности. С одной стороны, они формально выглядят и ведут себя как коды на языках программирования (наличие формальных языковых конструкций и специализированных сред разработки), с другой – обладают рядом кардинальных отличий, присущих аппаратуре (сигналы, параллелизм, синтезируемость). Отсюда невозможно тестировать и диагностировать такие модели исключительно методами верификации программного обеспечения (ПО) или методами диагностирования аппаратуры. С точки зрения уменьшения размерности задачи диагностирования целесообразно подойти к HDL-коду, как к аппаратуре, и в соответствии с этим строить тесты и проводить ДЭ с использованием процедур аппаратной тестовой диагностики.

В качестве модели объекта диагностирования предлагается использовать графовое представление описания устройства на HDL, которое задается в виде двух графов. Информационный I-граф описывает преобразование данных (подобно операционному автомату в классической композиционной модели с микропрограммным управлением) без учета условных ветвей. I-граф содержит два типа вершин: операнды и функции. Типы операндов: целые числа и беззнаковые векторы.

Типы функций ограничены синтезируемым подмножеством HDL или конструкциями, которые имеют физические эквиваленты в системах синтеза и имплементации в САПР. Управляющий С-граф соответствует цепочке условий, соответствующих рассматриваемому оператору. С-граф содержит условные конструкции (например, case, if...then..., with … select), взятые из исходного описания ЦУ. С-граф – это структура с двумя типами вершин: условиями и метками. Вершины условий содержат вычисляемые операторы. Вершины меток – конечные, не имеющие входной дуги и содержащие имя метки. Результат моделирования С-графа – набор меток, по которым осуществляются переходы в I-графе. В такой интерпретации С-граф соответствует управляющему автомату [1].

Для декомпозиции исходного I-графа используются контрольные точки (КТ), аналогично наблюдаемым линиям при генерации тестов, которые позволяли «разбить» путь активизации и определить границы подграфов. При этом ошибка проектирования транспортируется на внешний выход каждого подграфа. Особенностью диагностирования HDL-модели является отсутствие эталонного HDL-кода или полной спецификации, поэтому вычислить эталонные значения сигналов во всех КТ без привлечения внешних способов невозможно. Исходя из этого, определяются два типа наблюдаемых точек, используемых при поиске места возникновения ошибки в HDL-коде. КТ первого рода – сигналы (переменные) модели, которые наблюдаются, где эталонные значения известны из спецификации. КТ второго рода – сигналы (переменные) модели, значения в которых наблюдаемы, но до начала ДЭ эти значения неизвестны.

Учитывая, что HDL-операторы, выбранные в качестве функциональных примитивных элементов (ПЭ), не содержат внутри себя ошибок, очевидно, что подача на ПЭ тестов проверки исправности является нецелесообразной. Поэтому смысл тестирования ПЭ состоит не в проверке правильности функционирования, а в идентификации его типа. Таким образом, на примитив необходимо подавать такие тестовые наборы, чтобы после анализа реакций на них можно идентифицировать тип (функцию) примитива и отличить его от других ПЭ. Такие различающие последовательности (РПС) позволяют найти ошибки, связанные с заменой операторов в HDL-коде. Ошибки в выражениях, как правило, связаны з заменой операндов в арифметических, логических или условных операторах [2].

Целью данного исследования является существенное уменьшение времени проектирования цифровых изделий путем разработки углубленного метода поиска ошибок проектирования в HDL-моделях поведенческого уровня описания в условиях неполной спецификации.

2. Общая стратегия поиска ошибок проектирования в HDL-моделях

Процесс проведения алгоритма диагностирования называют, как правило, диагностическим экспериментом. Диагностический эксперимент над HDL-кодом осуществляется в два этапа. На первом проводится безусловный эксперимент путем подачи на входы модели теста и сравнение последовательностей на внешних выходах с эталонными значениями. Если результат хотя бы на одном выходе не совпадает с эталоном, выполняется второй этап ДЭ – локализация ошибок проектирования.

По результатам проведения первого этапа диагностического эксперимента (подачи теста для определения ошибки проектирования в описании) формируется вектор экспериментальной проверки (ВЭП): $V = (v_1, v_2, .. v_m)$, где m – количество контрольных точек первого рода.

ВЭП указывает, на какие выходы влияет модельная или реальная ошибка проектирования. Координаты ВЭП вычисляются :

$$v_i = \begin{cases} 0, \text{если } S_i^{эт} = S_i^{экс}; \\ 1, \text{если } S_i^{эт} \neq S_i^{экс}, \end{cases}$$

где $S_i^{эт}$ и $S_i^{экс}$ – результаты эталонной и экспериментальной реакции соответственно в i-й КТ первого рода.

Общую методику диагностирования HDL-модели можно определить такой последовательностью шагов [2] :

1) структурная декомпозиция HDL-модели, составление графовой модели на основе I-графа и С-графа, псевдоразрыв обратных связей;

2) определение класса ошибок проектирования, составление РПС для них, тестирование модели на основе подачи РПС и сравнение с имеющимися эталонами;

3) для небольших фрагментов HDL-кода применение функционального метода поиска ошибок проектирования с применением словарей неисправностей на основе таблиц функций неисправностей;

4) сужение области подозреваемых дефектов на основе структурного метода минимизации пространства неисправностей заданного класса для многовыходовых моделей;

5) применение структурного метода поиска для подграфов с контрольными точками первого рода;

6) применение структурно-функционального метода обратного прослеживания для уточнения диагноза в подграфах, ограниченных контрольными точками первого рода.

В спецификациях HDL-моделей реальных цифровых устройств контрольных точек первого рода, в которых известны значения эталонных сигналов до начала верификации HDL-кода, как правило, немного. Поэтому локализация до подграфа с выходной КТ первого рода является недостаточной. Желательно локализацию ошибочных операторов проводить до КТ второго рода, но эталоны, как правило, отсутствуют. Выходом из этого положения является применение в подграфе, где выходом есть КТ первого рода, метода обратного прослеживания, который первоначально был ориентирован на поиск константных неисправностей в цифровых схемах в условиях отсутствия эталонов во внутренних точках. Классический метод обратного прослеживания, который схемотехники называют методом уточнения диагноза, основан на использовании следующих компонентов и условий:

1. Структурно-функциональное модель-устройство с доступными для наблюдения внутренними линиями.

2. Существенность каждого элемента.

3. Тест, который активизирует все пути в схеме.

4. Эталонные реакции на внешнем выходе модели.

5. Наличие в схеме одиночной константной неисправности.

Суть метода состоит в следующем: подается тест на схему с подозреваемой неисправностью и фиксируется реакция на внешнем выходе схемы. Если на внешнем выходе схемы реальное значение не совпадает с эталонным, то выполняется импликация назад с использованием условий существенности до тех пор, пока не исчезнет различие между реальными значениями и эталонными. Последняя из линий схемы, на которой наблюдается несовпадение, признается неисправной. Метод применяется для одновыходовых схем в предположении о наличии в них одиночных неисправностей.

Для использования данного метода при уточнении диагноза в классе эквивалентных ошибок проектирования HDL-кода необходима модификация метода. Введем понятие эквивалентной цифровой схемы функционально-блочного уровня, результаты моделирования которой совпадают с результатами моделирования HDL-кода на входных воздействиях из спецификации. Эквивалентная схема в общем случае может быть результатом функционального синтеза, но может и отличаться, поскольку многие синтезаторы значительно оптимизируют HDL-код.

Исходя из этого, принимаются следующие ограничения и предположения.

1. Модель фрагмента HDL-кода есть структурно-функциональная модель эквивалентной схемы, примитивами которой являются предопределенные операторы или логические условия HDL-кода.

2. Класс неисправностей (ошибок проектирования) – замена функционального элемента или изменение логического условия.

3. Все операнды HDL-модели (переменные и сигналы) считаются исправными (правильными). Замена операндов не рассматривается.

4. В качестве тестов используются те, которые построены на основе РПС для каждого функционального элемента и транспортированы до КТ первого рода (внешних выходов схемы).

5. Для HDL-моделей, синтезируемых в последовательностные структуры, цепи сброса и синхронизации считаются исправными. Это достигается путем предварительного инспектирования указанных сигналов и переменных в HDL-коде.

6. Неисправным (ошибочным) признается функциональный элемент, на входах которого при обратном прослеживании наблюдается последнее несовпадение с эталоном.

7. Если несовпадение с эталоном распространяется до внешнего входа, то неисправный элемент лежит на пути обратного прослеживания.

Алгоритм уточнения диагноза имеет следующие пункты.

1. Выполняется моделирование РПС-теста на реальной HDL-модели до внешнего выхода (КТ первого рода).

2. Если на выходе обнаружено несовпадение с эталоном из спецификации, то осуществляется процедура обратного прослеживания.

3. Обратное прослеживание выполняется путем пересечения текущего вектора с условиями существенности обрабатываемого ФЭ до тех пор, пока результат не станет равным $\varnothing$, или не будут достигнуты внешние входы.

4. Из непустых результатов пересечений формируется подмножество подозреваемых элементов. Если у какого-то из вариантов обратное прослеживание достигло внешних входов, то необходимо провести инспекцию HDL-кода вдоль данного пути обратного прослеживания, начиная от внешних входов в процессе уточнения диагноза. Под инспекцией кода понимается визуальный анализ перечня подозреваемых HDL-операторов в целях обнаружения (искомых) ошибок проектирования (для которых строились РПС-тесты).

Пример применения метода обратного прослеживания в HDL-моделях стиля «поток данных» приведен в [3].

3. Метод обратного прослеживания в поведенческих HDL-моделях

Рассмотрим проведение ДЭ над моделью цифрового устройства на языке VHDL, в результате которого локализован фрагмент VHDL-кода поведенческого стиля описания, внутри которого нет контрольных

точек первого рода, иначе говоря, эталонные значения для внутренних сигналов (переменных) фрагмента кода из спецификации неизвестны. Задан перечень ошибок проектирования (ограничения на перечень), которые могут присутствовать в данном фрагменте кода. Предполагается, что в VHDL-коде присутствует одиночная ошибка проектирования типа «замена операнда в условном выражении» (ошибочная ветвь в условном операторе). Необходимо повысить глубину диагностирования для обнаружения ошибок проектирования в VHDL-коде с точностью до оператора HDL (группы эквивалентных операторов).

Метод уточнения диагноза использует в качестве моделей элементов табличные описания, поэтому при рассмотрении логических операторов внутри процессов в VHDL–коде возникает проблема представления (if, case) в табличном виде. Предлагается при обратном прослеживании указанные операторы представлять не в виде элементов управляющего графа, а в виде табличных представлений мультиплексоров и АЛУ–подобных элементов, независимо от того, как указанные операторы будут синтезироваться.

Для представления внутреннего поведения компонентов в сложных VHDL-моделях служит конструкция process( ), которая определяет поведенческий стиль описания модели. Данная конструкция дает возможность использовать условные конструкции и последовательный характер выполнения операторов, что свойственно традиционным языкам программирования.

Кроме операторов параллельного назначения сигналов (ПНС) в VHDL-коде поведенческого стиля присутствуют последовательные и условные операторы (if, case), которые представляют особый интерес при выполнении процедуры обратного прослеживания, поскольку в рассматриваемой графовой модели они находятся на стыке I- и C-графов.

Рассмотрим форму и способ построения покрытий существенности условных операторов. Первоначально рассмотрим оператор if :

If((X1 **and** X2) **or** (X3 **and** X4)) Y<=D1;

else Y<=D2;

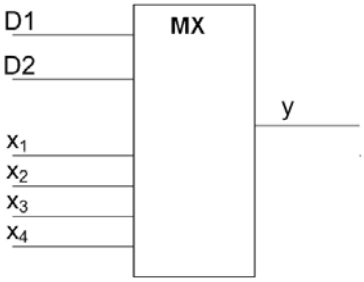
Оператор if(C), реализующий двухвыходовую условную вершину, состоит из логического условия C и двух выполняемых операторов ПНС или присваивания значений переменных. Логическое условие C в операторе if(C) представляет собой уравнение, решение которого в кубической форме представлено на рис.1.

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

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

Рис. 1. Решение уравнения по логическому условию

Таким образом, оператор if(C) может быть представлен моделью мультиплексора с двумя информационными входами, группой управляющих входов и одним (двумя) выходом (два выхода будет в случае, если назначение выполняется для различных сигналов). Эквивалентная схема мультиплексора и покрытие существенности оператора if(C) относительно информационных входов представлены на рис.2.



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

Рис. 2. Модель оператора if( )и его покрытие существенности

Кроме оператора if( ) к условным операторам можно отнести и оператор case( ), который реализует многовыходовую условную вершину. Фрагмент VHDL-кода с оператором case( ) представлен на рис. 3.

```
process( )
variable xxx : in std_logic_vector(2 downto 0);
begin
 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; end process;
```

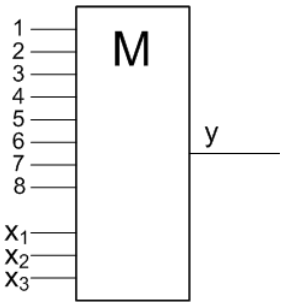
Рис. 3. Фрагмент VHDL-кода с оператором case( )

Модель оператора case( ) в форме мультиплексора и его покрытие существенности представлены на рис.4.

Покажем пример применения изложенного метода. Рассмотрим фрагмент поведенческого описания АЛУ, спецификация на который представлена в табл. 1, а VHDL-модель показана на рис. 5.

Таблица 1

Reset (R)	Control (C)	Output (Y)
0	X	1
1	0	Y=A & B
1	1	Y=A v B



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

Рис. 4. Модель оператора case( ) и его покрытие существенности

```
Entity SCH2 is
Port (R,C,A,B : in bit; Y : out bit);
End;
Architecture BEH2 of SCH2 is
begin
process(R,C,A,B)
begin
  if(R = '0')then Y <= '1';
   elsif(C = '0') then Y <=A and B;
   else Y <=A or B;
  end if;
end process; End;
```

Рис. 5. VHDL-модель фрагмента АЛУ поведенческого стиля описания

Эквивалентная структурно-функциональная модель приведена на рис. 6, что соответствует результатам синтеза рассматриваемого фрагмента VHDL-кода, а покрытие существенности оператора if( )-else( ) в виде мультиплексора 2 на 1 представлено на рис. 7.

Таблица 2

Reset									Y=A & B, Y=A v B								
1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
1	0	0	1	0	0	0	0	1	1	0	1	0	1	0	1	0	0
1	0	0	1	1	0	0	0	0	1	0	1	1	1	0	1	1	1

Проверка условных конструкций VHDL-кода состоит в построении РПС-тестов для всех ветвей условных операторов. В табл. 2 приведены РПС-тесты и эталонные реакции для трех ветвей комплексного условного оператора if()-elsif()-else() из VHDL-модели АЛУ на рис. 5. Эталонные реакции получены из спецификации (см. табл. 1).

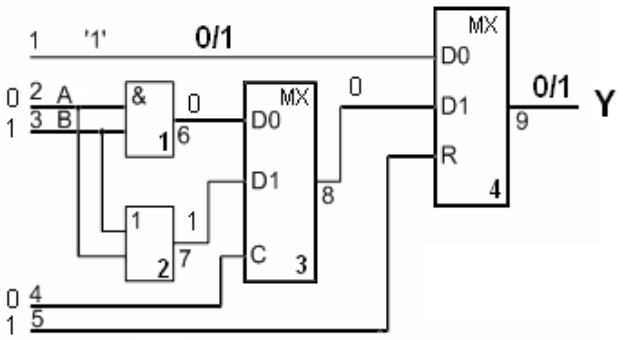
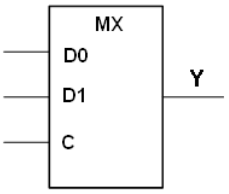


Рис. 6. Эквивалентная схема фрагмента АЛУ для РПС-теста (10010001)



D0	D1	C	Y
Z	X	0	Z
X	Z	1	Z

Рис. 7. Покрытие существенности оператора if( )-else( )

Рассмотрим ошибку проектирования в операторе ПНС логического выражения:

if(R = '0')**then** Y<='1';

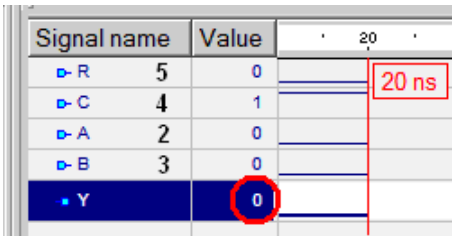
а именно **if**(R ='0')**then** Y<='0'.

Результаты моделирования ошибочного VHDL-кода приведены для входного набора РПС-теста 10010 на рис. 8, а, а для набора РПС-теста 10011 – на рис. 8, б. На входном наборе 10010 наблюдается несовпадение с эталоном (в спецификации Y=1, а в результате моделирования получено Y=0).

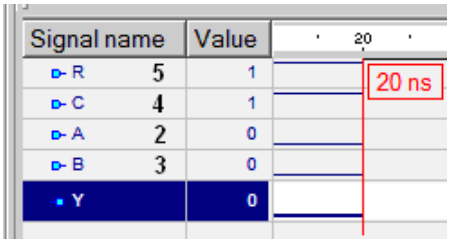
Результаты обратного прослеживания для теста 10010 приведены в табл. 3.

Таблица 3

1	2	3	4	5	6	7	8	9	
1	0	0	1	0	0	0	0	0	Эталон 1
								z	Линия 9 сущ
x				1			z	z	$\rightarrow \varnothing$
z				0			x	z	$\cap \neq \varnothing$
z	0	0	1	0	0	0	0	z	Внешний вход 1



а



б

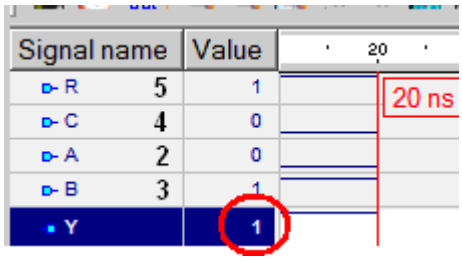
Рис.8. Результат моделирования VHDL-модели с ошибкой **if**(R=’0’)**then** Y<=’0’

Последняя существенная линия 1 является входом оператора **if**(R =’0’) **then** Y <= ’0’;, что означает – данный оператор и является ошибочным.

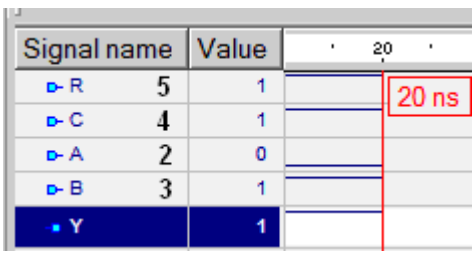
Рассмотрим другую ошибку проектирования в условном операторе:

elsif(C = ’0’) **then** Y <=A **and** B; **else** Y <=A **or** B;, а именно замену ’0’ на ’1’ в логическом условии : **elsif**(C = ’1’) **then** Y <=A **and** B;

Результаты моделирования ошибочного VHDL-кода приведены для входного набора РПС-теста 10101 на рис. 9, а, а для набора РПС-теста 10111 – на рис. 9,б. На входном наборе 10101 наблюдается несовпадение с эталоном (в спецификации Y=0, а в результате моделирования получено Y=1).



а



б

Рис.9. Результат моделирования VHDL-модели с ошибкой **elsif**(C = ’1’) **then** Y <=A **and** B

Результаты обратного прослеживания для теста 10101 приведены в табл. 4.

Несовпадение с эталоном наблюдается вдоль пути 2 – 6 – 8 – 9. Таким образом, подозреваемым является оператор **elsif**(C = ’1’) **then** Y <=A **and** B; а именно, существует наличие ошибки в логическом условии (C = ’1’) или в операторе ПНС Y<=A **and** B;. Инспекция кода подтверждает ошибку в логическом условии указанного оператора.

Таблица 4

1	2	3	4	5	6	7	8	9	
1	0	1	0	1	0	1	1	1	Эталон 0
								z	Линия 9 сущ
z				0			x	z	$\rightarrow\varnothing$
x				1			z	z	$\cap\neq\varnothing$
1	0	1	0	1	0	1	z	z	Линия 8 сущ
			0		z	x	z		$\cap\neq\varnothing$
			1		x	z	z		$\rightarrow\varnothing$
1	0	1	0	1	z	1	z	z	Линия 6 сущ
	z	1			z				$\cap\neq\varnothing$
	1	z			z				$\rightarrow\varnothing$
1	z	1	0	1	z	1	z	z	Внешний вход 2

4. Выводы

Научная новизна. Впервые разработан метод «обратного прослеживания», повышающий глубину поиска ошибок проектирования в поведенческих HDL-моделях цифровых устройств, представленных на языке описания аппаратуры VHDL в условиях неполной спецификации путем реализации обратного прослеживания на структурно-функциональной модели фрагмента HDL-кода функционально-блочного уровня с использованием операций кубического исчисления, а также методов логического моделирования.

Практическая значимость. Разработанный метод совместно с [3] позволяет осуществлять локализацию места возникновения ошибки проектирования в VHDL-моделях разных стилей описания до неверного оператора или операнда внутри подграфа, ограниченного контрольными точками первого рода. Данные методы поиска ошибок проектирования в VHDL-коде интегрированы в систему автоматизированного проектирования Active-HDL, что позволяет сократить временные затраты на поиск ошибок проектирования при функциональной верификации проектов ЦУ.

Литература: 1.*Шкиль А.С.* Структурное и функциональное диагностирование HDL-моделей цифровых устройств в САПР РЭА /*А.С. Шкиль, Е.Е. Сыревич, С. Альмадхоун, Г.П. Фастовец* // Інформаційно-керуючі системи на залізничному транспорті. 2013. № 2. С. 75-82. **2.***Шкиль А.С.* Поиск ошибок проектирования в HDL-моделях цифровых устройств / *А.С. Шкиль, Е.Е. Сыревич, С. Альмадхоун* // Інформаційні технології та комп’ютерна інженерія: міжнародна науково-практична конф., Вінниця, 19-21 травня 2010. Вінниця, ВНТУ, 2010. С. 377-378. **3.***Шкиль А.С.* Метод обратного прослеживания для поиска ошибок

проектирования в HDL-коде / *А.С. Шкиль, Е.Е. Сыревич, Д.Е. Кучеренко, С. Альмадхоун* // Радиоэлектроника. Информатика. Управление. 2009. №2. С. 86-90.

Поступила в редколлегию 11.06.2015

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

Шкиль Александр Сергеевич, канд. техн. наук, доцент кафедры АПВТ ХНУРЭ. Научные интересы: логическое моделирование, техническая диагностика компьютерных систем. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, комн. 319.

УДК004:519.713

КИБЕРФИЗИЧЕСКИЕ СТРУКТУРЫ ДЛЯ АНАЛИЗА БОЛЬШИХ ДАННЫХ

ХАХАНОВ В.И., ЛИТВИНОВА Е.И., ЗАЙЧЕНКО С.А., ГУРЕЕВ Б.Н., ШЛЯХТУН М.М.

Предлагается инфраструктура обеспечения параллельного анализа big data для поиска, распознавания и принятия решений на основе использования булеановой метрики измерения киберпространства. Она характеризуется наличием единственной логической xor-операции для определения кибер-расстояния путем циклического замыкания не менее одного объекта, что дает возможность на порядок повысить быстродействие анализа больших данных. Разрабатывается новая структурная модель анализа big data, которая характеризуется использованием облачных сервисов, киберфизических и поисковых систем, параллельных виртуальных мультипроцессоров с минимальным набором векторно-логических операций для точного поиска информации на основе предложенной булеановой метрики и нечисленных критериев качества. Это дает возможность создавать семантическую инфраструктуру «чистого» киберпространства путем компетентностной классификации и метрического упорядочения big data в масштабах киберэкосистемы планеты.

1. Введение

Киберфизическая система призвана сделать активной концепцию big data, рассматривая большие данные во взаимодействии с киберсистемами (облаками) управления, ориентированными на поиск, распознавание и принятие решений. Структурное содержание CPS (рис. 1) – совокупность коммуникационно связанных реальных и виртуальных компонентов с выраженными функциями адекватного физического цифрового мониторинга и оптимального облачного компьютерного киберуправления для обеспечения качества жизни, продукции, процессов или сервисов в заданных условиях ограничений на время и ресурсы. CPS включает компоненты: Cyber Control, Internet of Things или Cloud, Security, Intelligence, Big Data and Services, Digital Monitoring, Cyber Managing, Physical Smart Everything, Nature, Social and Tech World. Регуляторные отношения (Relationship) между компонентами CPS формируются законами, уставами предприятий и организаций, морально-этическими правилами поведения внутри социальной группы. Направление движения RoadMap – Harmony of Human, Nature and Tech киберфизической системы человечества можно определить как достижение такого интегрального уровня развития киберфизических компонентов, который обеспечит гармонию жизни человека с природой и техникой (созданным миром – Created World).

Big data – технологическая культура киберпространства, направленная на формирование динамически развивающейся инфраструктуры киберфизической экосистемы планеты путем метрико-семантической структуризации больших потоков (объемов) гетерогенных данных на основе использования интеллектуальных быстродействующих специализированных облачных фильтров параллельного мониторинга и метрического анализа извлекаемой информации для online управления физическими и виртуальными процессами.

Рыночно-привлекательные глобальные проекты сегодня выполняются под эгидой объединения физического и виртуального пространства в единое целое. Киберфизическое пространство (Cyber Physical Space)

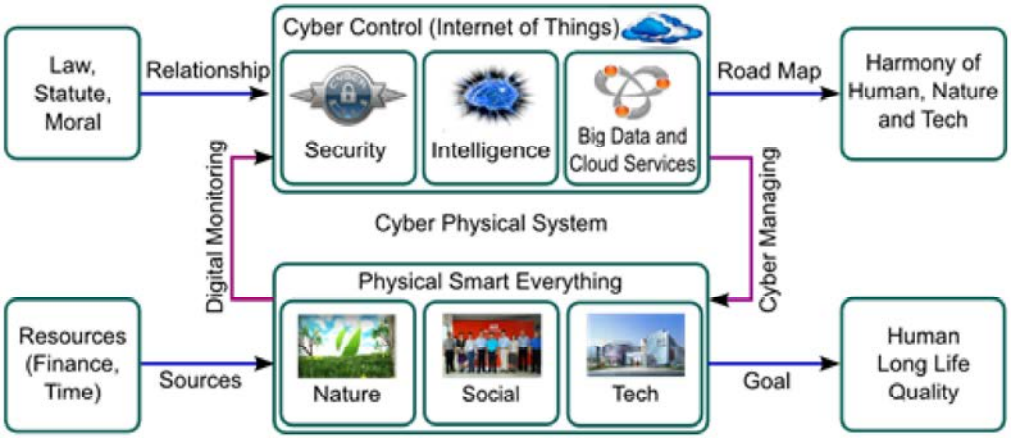


Рис. 1. Киберфизическая система управления неприродными процессами