УДК 681.518.5

А.С. ШКИЛЬ, Э.Н. КУЛАК, А.С. СЕРОКУРОВА

ДИАГНОСТИРОВАНИЕ HDL-МОДЕЛЕЙ МИКРОПРОГРАММНЫХ АВТОМАТОВ

Рассматриваются методы поиска ошибок проектирования в HDL-моделях микропрограммных автоматов. Исходное описание автомата представляется композицией операционного и управляющего автоматов, которая описывается содержательной граф-схемой алгоритма. HDL-модель управляющего автомата дана в форме двухпроцессного автоматного шаблона автомата Мура. Диагностический эксперимент проводится путем обхода всех дуг графа переходов управляющего автомата, начиная от начальной вершины, путем эмуляции функций операционного автомата в системе верификации HDL-моделей (TestBench) в среде проектирования Active–HDL.

1. Введение

Цифровое операционное устройство (ОУ) можно представить комбинацией операционного автомата (ОА) и управляющего автомата (УА). Операционный автомат выполняет преобразование данных, а именно, выполняет микрооперации (МО), инициируемые управляющими сигналами $Y=\{y_i\}$, порядок следования которых определяется УА. Управляющий автомат определяет выполнение последовательности микроопераций на основе граф-схемы алгоритмов (ГСА) и множества оповестительных сигналов $X=\{x_j\}$, вырабатываемых ОА. Такой автомат принято называть микропрограммным управляющим автоматом [1]. В общем случае операционный автомат по способу описания не отличается от управляющего, а распределение ролей между двумя взаимосвязанными автоматами определяется наличием входных сигналов. Если входные сигналы присутствуют в обеих структурах, установить роль ведущего и ведомого достаточно сложно, поскольку любой автомат может быть описан графом переходов. Но классификация схемных стандартов операционного автомата по их функциям (триггеры, счетчики, регистры, память, коммутаторы, декодеры, АЛУ) позволяет оперировать ими как примитивами. В этом случае управляющий автомат рассматривается как способ описания взаимодействия примитивов операционного автомата во времени. Разделение конкретного проекта на УА и ОА субъективно и является следствием практического опыта и квалификации разработчика.

При существующем многообразии исходных форм описания проектов цифровых устройств (ЦУ) можно выделить наиболее популярные в мире: аналитические – языки описания аппаратуры (HDL), графические или визуальные – иерархические цифровые структуры и схемы, граф-схемы алгоритмов операционных или управляющих устройств (flow chart). Одним из распространенных способов исходного описания конечного автомата (УА) на языке описания аппаратуры является автоматный шаблон, т.е. специальная структура HDL-кода, которая строится на основе графа переходов автомата (state diagram) или прямой структурной таблицы. Построение графа переходов конечного автомата на основе других способов описания его функционирования является искусством проектировщика и особенностями инструментальных средств систем автоматизированного проектирования радиоэлектронной аппаратуры (САПР РЭА) [2].

Наиболее сложным и затратным этапом в современном цикле проектирования ЦУ является функциональная верификация, т.е. процесс обнаружения, локализации и устранения ошибок в системной модели относительно спецификации, на что затрачивается более половины общего времени проектирования. Основной формой описания проектов ЦУ в САПР РЭА являются языки описания аппаратуры, поэтому объектом верификации есть модель ЦУ, написанная на языке описания аппаратуры, т. е. HDL-модель.

Возможные ошибки проектирования в HDL-моделях определяются стилем описания HDL-кода. Под ошибкой проектирования понимается ошибка в HDL-операторе, которая не относится к классу синтаксических и нарушает алгоритм функционирования модели устройства, заданный спецификацией. Выделение фрагментов HDL-кода, описывающих пове-

дение конечных автоматов стилем «автоматный шаблон», позволяет определить ошибку проектирования типа «неправильный переход в графе переходов автомата», что соответствует ошибке в выборе текущего состояния в операторе when, ошибке выбора следующего состояния в функции переходов (a_i вместо a_j), ошибке в операторе if( ) при анализе входного сигнала, ошибке в назначении выходного сигнала. Для проведения диагностического эксперимента (ДЭ) по поиску ошибок проектирования реализуется стратегия обхода всех дуг графа переходов конечного автомата, начиная с начальной вершины. При этом проверяются все одиночные неисправности переходов, а также исправности функций автомата, обеспечивающих эти переходы [3].

ДЭ над HDL-моделью конечного автомата состоит в подаче на нее входных воздействий в соответствии с выбранной стратегией обхода содержательного графа переходов, получении выходных реакций на Waveform и сравнения полученных реакций с эталоном. На основании этого делается вывод о соответствии HDL-модели спецификации. ДЭ проводится с использованием системы верификации HDL-моделей (TestBench) в среде проектирования Active–HDL. При проведении ДЭ в простых HDL-моделях УА подача входных воздействий и сравнение полученных реакций с эталонами не представляет особых трудностей даже в режиме визуального сравнения по Waveform, так как тестовые данные подаются непосредственно на входы автомата, а реакции снимаются с его выходов.

При проведении ДЭ для HDL-моделей микропрограммных автоматов задача подачи входных воздействий и сравнения выходных реакций с эталонами усложняется. С одной стороны, при проведении ДЭ по обходу всех дуг графа автомата фактически проверяется УА, а входные данные (операнды микропрограммы) подаются на ОА и с него же снимаются выходные реакции (результат). С другой стороны, входные данные УА (оповестительные сигналы x_i) и выходные (сигналы инициализации микроопераций y_i) не определены в спецификации и непосредственно сравниваться с эталоном при проведении ДЭ не могут. Таким образом, задача разработки методики проведения ДЭ над HDL-моделью микропрограммного автомата, заданного содержательной граф-схемой алгоритма микропрограммы, является актуальной. Эталонные реакции (функции выходов ОА) при этом определяются исключительно спецификацией на операции, выполняемые операционным устройством.

2. Подготовка диагностического эксперимента для микропрограммного автомата

Особенностью проведения ДЭ по диагностированию HDL-модели (как и любого программного кода) является отсутствие эталонного программного кода, поэтому в качестве эталона можно использовать только результаты выполнения алгоритма микропрограммного ОА, определенного спецификацией. Таким образом, стратегия обхода всех дуг графа переходов УА реализуется путем построения последовательностей микроопераций обработки данных операционным автоматом.

Подготовку и проведение диагностического эксперимента по локализации ошибок проектирования в HDL-моделях микропрограммных автоматов будем рассматривать на примере микропрограммы (МКП) сложения четырехразрядных двоичных знаковых чисел в дополнительном модифицированном коде. Для данной микропрограммы в качестве УА рассматривается автомат Мура. На рис. 1,а показан фрагмент содержательной граф-схемы алгоритма указанной микропрограммы с отметками состояний УА для автомата Мура. Каждой операторной вершине ГСА (состоянию автомата Мура) соответствует набор управляющих сигналов y_i, каждый из которых инициирует выполнение определенной микрооперации ОА. Для упрощения дальнейшего изложения в данном фрагменте микропрограммы не анализируется результат сложения на переполнение разрядной сетки, что в целом не меняет структуры микропрограммы. На рис. 1, б приведен содержательный граф переходов УА Мура для данной микропрограммы, в котором латинскими буквами поименованы дуги графа.

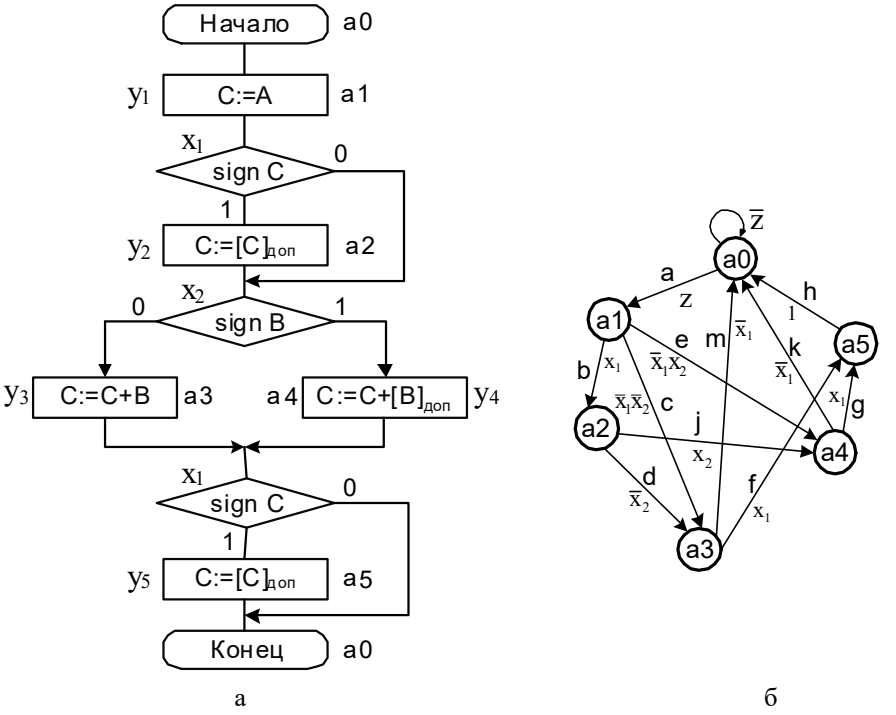
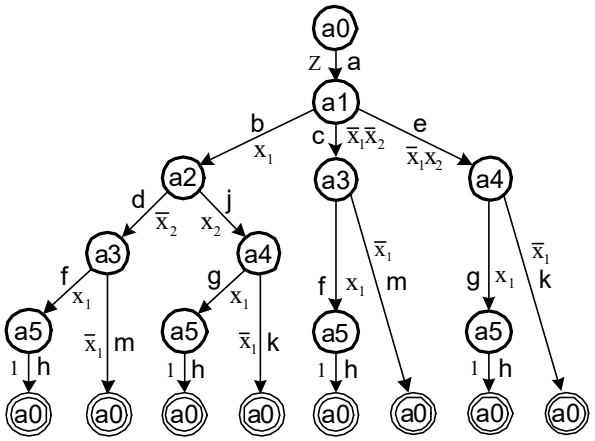


Рис. 1. Фрагмент содержательной ГСА микропрограммы сложения (а) и граф переходов управляющего автомата (б)

Для реализации стратегии обхода всех дуг графа по методике, предложенной в [3], на основании модифицированной матрицы смежности (рис.2,а) строится дерево решений для обхода путей (маршрутов) графа. В линейной ГСА, где логические условия следуют друг за другом, максимальное количество путей обхода графа автомата Мура будет 2^n, где n – количество логических условий x_i с двумя альтернативами, т.е. для данного графа переходов будет 8 маршрутов обхода. При этом условие однократного покрытия всех дуг графа при его обходе на данном этапе не анализируется. Особенность данного дерева (рис.2,б) состоит в том, что терминальной вершиной во всех маршрутах обхода графа УА является состояние a0, т.е. каждый путь в графе реализует полный цикл работы ОА.

	a0	a1	a2	a3	a4	a5
a0	$\overline{Z}$	a				
a1			b	c	e	
a2				d	j	
a3	m					f
a4	k					g
a5	h					

а



б

Рис. 2. Матрица смежности (а) и дерево решений для маршрутов обхода графа переходов управляющего автомата (б)

По дереву решений строится таблица проверок (маршруты обхода графа), по которым непосредственно и проводится диагностический эксперимент [4]. В таблице представлен полный перечень проверок P_i для маршрутов обхода графа. Каждая проверка характеризуется перечнем активизируемых дуг графа УА, перечнем управляющих сигналов для инициализации микроопераций y_i (вершин графа УА) и диапазоном значений операндов (входных данных), которые данный маршрут обхода реализуют.

Полный перечень проверок при проведении ДЭ и перечень МО

M	a	b	c	d	e	f	g	h	j	k	m	Сигналы инициализации МО	Операнды и результат	Совместимая последовательность МО
P_1	1	1		1		1		1				y_1, y_2, y_3, y_5	C=(-A)+B<0	да
P_2	1	1		1							1	y_1, y_2, y_3	C=(-A)+B>0	да
P_3	1	1					1	1	1			y_1, y_2, y_4, y_5	C=(-A)+(-B)<0	да
P_4	1	1							1	1		y_1, y_2, y_4	C=(-A)+(-B)>0	нет
P_5	1		1			1		1				y_1, y_3, y_5	C=A+B<0	нет
P_6	1		1								1	y_1, y_3	C=A+B>0	да
P_7	1				1		1	1				y_1, y_4, y_5	C=A+(-B)<0	да
P_8	1				1					1		y_1, y_4	C=A+(-B)>0	да

Любой маршрут обхода графа микропрограммного автомата порождает выполнение последовательности микроопераций, которые реализуют определенную часть ГСА микропрограммы (алгоритма обработки данных). Поэтому перед проведением ДЭ возникает необходимость рассмотрения возможности совместного выполнения определенных микроопераций. В теории проектирования микропрограммных автоматов используется понятие функциональной и структурной совместимости микроопераций [1]. Микрооперации называются функционально совместимыми, если в один момент времени они присваивают результаты разным операндам. Микрооперации называются структурно-совместимыми, если они одновременно не используют одни и те же аппаратные ресурсы ОА. В дальнейшем изложении управляющий сигнал инициализации микрооперации y_i и соответствующая ему микрооперация будут использоваться как синонимы.

При организации ДЭ по обходу графа микропрограммы возникает необходимость введения понятия совместимости последовательности микроопераций (во времени выполнения). Последовательность микроопераций (управляющих сигналов y_i) называется совместимой, если существуют значения операндов ОА, которые позволяют реализовать данную последовательность. Если таких операндов не существует, то последовательность микроопераций называется несовместимой. Например, в операции сложения знаковых чисел с переполнением разнознаковые операнды никогда не дадут переполнения разрядной сетки. Таким образом, если в заданной последовательности микроопераций для сложения разнознаковых чисел присутствует y_i, реализующая выдачу сигнала переполнения разрядной сетки, данная последовательность является несовместимой. Несовместимость последовательности микроопераций бывает позитивной (в последовательности y_i присутствуют лишни микрооперации) или негативной (в последовательности y_i отсутствуют необходимые микрооперации). Это следует учитывать при формировании путей обхода графа переходов УА.

Если проанализировать таблицу, то можно сделать вывод, что последовательность микроопераций, реализующая проверку $P_4 = \{y_1, y_2, y_4\}$, определяет наличие положительной суммы (отсутствует МО y_5) при отрицательных операндах (МО $\{y_2, y_4\}$), что невозможно, и данная последовательность МО несовместима. Аналогично, последовательность МО, реализующая проверку $P_5 = \{y_1, y_3, y_5\}$, определяет наличие отрицательной суммы (присутствует МО y_5) при положительных операндах (МО y_3 при отсутствии МО y_2) и также является несовместимой. Таким образом, проверки P_4 и P_5 нереализуемы и вектор экспериментальных проверок (ВЕП) будет $V = (P_1, P_2, P_3, P_6, P_7, P_8)$. При этом, если проанализи-

ровать оставшиеся строки табл.1, то отсутствие проверок P_4 и P_5 не нарушает полноту ДЭ по обходу всех дуг графа (во всех столбцах, соответствующих дугам графа {a, b, c, d, e, f, g, h, j, k, m}, присутствуют единицы, и условие однократного покрытия всех дуг графа соблюдается). Такая ситуация может быть не всегда, но рассмотрение условий получения однократного покрытия при обходе всех дуг графа не входит в задачи данного исследования.

В данной работе результат проведения ДЭ определяется путем визуального сравнения результатов моделирования HDL-модели на временной диаграмме (Waveform) с эталонными значениями (результатами выполнения операции сложения для разных операндов). При подборе операндов для реализации указанных проверок необходимо учитывать следующее:

– значения операндов должны быть такими, чтобы не было переполнения разрядной сетки, так как в рассматриваемом фрагменте микропрограммы отсутствует анализ переполнения;

– значения операндов для всех проверок должны быть такими, чтобы результат выполнения операции сложения для них различался.

На рис.3 приведен один из возможных вариантов значений входных данных и результатов для проверок {P_1, P_2, P_3, P_6, P_7, P_8}, что является эталонными значениями для проведения ДЭ. Для упрощения формирования эталонов на данном рисунке приведены обратные и дополнительные коды рассматриваемых операндов и результатов.

Обратные и дополнительные коды отрицательных операндов

Десятичные числа	-2	-3	-4	-5	-8	-9
Прямой код	11 0010	11 0011	11 0100	11 0101	11 1000	11 1001
Обратный код	11 1101	11 1100	11 1011	11 1010	11 0111	11 0110
Дополнит. код (+1)	11 1110	11 1101	11 1100	11 1011	11 1000	11 0111

M	Операнды и знак результата	Десятичные данные и результат	Двоичные данные и результат выполнения сложения в дополнительном коде	Результат в прямом коде (+1)
P_1	C=(-A)+B<0	(-9)+5 = (-4)	11 0111 + 00 0101 = 11 1100	11 0100
P_2	C=(-A)+B>0	(-3)+5 = 2	11 1101 + 00 0101 =1 00 0010	00 0010
P_3	C=(-A)+(-B)<0	(-3)+(-5) = (-8)	11 1101 + 11 1011 = 1 11 1000	11 1000
P_6	C=A+B>0	9+5 = 14	00 1001 + 00 0101 = 00 1110	00 1110
P_7	C=A+(-B)<0	3+(-5) = (-2)	00 0011 + 11 1011 = 11 1110	11 0010
P_8	C=A+(-B)>0	9+(-3) = 6	00 1001 + 11 1101 = 1 00 0110	00 0110

Рис. 3. Подготовка эталонов для проведения диагностического эксперимента

В заключение подготовки ДЭ приведем эталонную HDL-модель УА Мура (рис.4), построенную по графу переходов УА (см. рис.1,б). Как уже упоминалось выше, при проведении реального ДЭ эталонного кода не существует (есть только HDL-код «неизвестного происхождения» с возможными ошибками проектирования), но для наглядности изложения и иллюстрации процедур локализации ошибок в HDL-коде эталонный код и результаты его моделирования привести целесообразно.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity FSM is
    port (Reset, Z, Clk, x1, x2: in STD_LOGIC;
                y1, y2, y3, y4, y5: out STD_LOGIC);
end;

architecture Moore of FSM is
    type State_type is (a0, a1, a2, a3, a4, a5);
    signal State, NextState: State_type;
begin
    Sreg0_CurrentState: process (Clk, reset)
    begin
                if Reset='1' then State <= a0;
                elsif Clk'event and Clk = '0' then State <= NextState;
                end if;
    end process;

    Sreg0_NextState: process (State, x1, x2, Z)
    begin
                case State is
                        when  a0=> if Z='1' then NextState <= a1;
                                   else NextState <= a0;
                                   end if;
                        when  a1=> if x1='1' then NextState <= a2;
                                   elsif x2='1' then NextState <= a4;
                                   else NextState <= a3;
                                   end if;
                        when  a2=> if x2='1' then NextState <= a4;
                                   else NextState <= a3;
                                   end if;
                        when  a3=> if x1='1' then NextState <= a5;
                                   else NextState <= a0;
                                   end if;
                        when  a4=> if x1='1' then NextState <= a5;
                                   else NextState <= a0;
                                   end if;
                        when  a5=> NextState <= a0;
                        when others => NextState <= a0;
                end case;
    end process;
    y1 <= '1' when State=a1 else '0';
    y2 <= '1' when State=a2 else '0';
    y3 <= '1' when State=a3 else '0';
    y4 <= '1' when State=a4 else '0';
    y5 <= '1' when State=a5 else '0';
end;
```

Рис. 4. Эталонная HDL-модель УА, построенная по графу переходов

3. Построение TestBench

Любая САПР на основе HDL имеет в своем составе встроенную систему верификации HDL-моделей (TestBench). Данная система позволяет подавать на верифицируемую HDL-модель (UUT) входные воздействия, моделировать их, снимать выходные реакции и выводить результат на систему визуального отображения временных диаграмм (Waveform). Также существует возможность проводить верификацию с использованием специальной тестовой программы (HDL-кода) в рамках TestBench (рис.5).

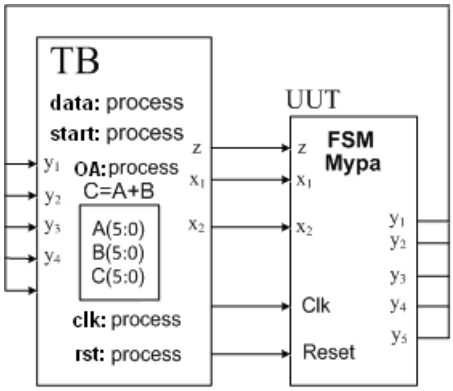


Рис. 5. Структура системы верификации (TestBench) при проведении ДЭ над УА

Тестовая программа TestBench (рис.6) состоит из ряда процессов:

clk**: process,** который формирует синхросигнал со скважностью 2T, где T – полупериод синхросигнала;

rst: **process**, который формирует сигнал сброса исходя из числа проверок ДЭ;

start: **process**, который формирует сигналы запуска автомата и определяет полный цикл проведения ДЭ;

data**: process,** который обеспечивает подачу входных воздействий на HDL-модель УА и формирует цикл работы микропрограммы;

OA**: process,** который запускается сигналами инициализации микроопераций y_i, эмулирует работу ОА и вырабатывает оповестительные сигналы x_i с использованием операторов условного назначения сигналов when.

В тестовой программе также используются:

переменная endsim – признак конца моделирования;

константа T - 1/2 периода синхросигнала Clk;

константа N - максимальная длина пути проверки с учетом сброса и запуска;

сигналы A, B, C – операнды и результат в двоичном виде;

сигналы Adec, Bdec, Cdec – операнды и результат в десятичном виде.

Сигнал сброса Reset активизируется в начале каждого цикла проверки ДЭ, для реализации неразрушаемого эксперимента, позволяющего в случае невозврата автомата в начальное состояние (по каким-либо причинам) обеспечить его гарантированную установку в начальное состояние.

ОА срабатывает по переднему фронту, а УА – по заднему. Это необходимо в случае использования УА Мура. Если ОА и УА срабатывают по одному фронту, они не укладываются в один такт, выполняя полный цикл взаимодействия – генерацию управляющих сигналов и выполнения МО, инициируемых этими управляющими сигналами. Следствием этого является неверное формирование признаков состояния ОА и неправильная генерация следующих управляющих сигналов УА.

В автомате Мили таких проблем не возникает, ОА и УА могут срабатывать по одному фронту. Эти особенности природы разных типов УА не зависят от системы моделирования HDL-моделей. Моделирование выполнялось в системах VCS 2014.12, Riviera-PRO EDU 2015.06, Active-HDL и дало идентичные результаты.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity fsm_tb1 is
end fsm_tb1;
architecture TB_ARCHITECTURE of fsm_tb1 is
    component fsm
    port(Reset, Z, Clk, x1, x2: in std_logic;
y1, y2, y3, y4, y5: out std_logic);
    end component;
    signal Reset, Z, Clk, x1, x2, y1, y2, y3, y4, y5: std_logic;
    signal A, B, C: STD_LOGIC_VECTOR(5 downto 0);
    signal Adec, Bdec, Cdec: integer;
    shared variable endsim: BOOLEAN:=false; — конец моделирования
    constant T: time:= 5 ns;              — 1/2 периода Clk
    constant N: integer:= 7;  — максимальная длина пути проверки
begin
    — Unit Under Test port map
  UUT : fsm port map (Reset => Reset, Z => Z, Clk => Clk, x1 => x1, x2 => x2, y1 => y1, y2 => y2, y3 => y3,
y4 => y4, y5 => y5);
    — Add your stimulus here ...
    clock: process                          — синхронизация
    begin
                if not endsim then Clk<='0'; wait for T;
                                   Clk<='1'; wait for T;
                else wait;
                end if;
    end process;
    rst: process                            — сброс
    begin
                if not endsim then Reset <='1'; wait  for 2*T;
  Reset <='0'; wait  for(N-1)*2*T;
                else wait;
                end if;
    end process;
    start: process                          — запуск
    begin
       if not endsim then Z <='0'; wait  for 2*T;
                          Z <='1'; wait  for 2*T;
                          Z <='0'; wait  for(N-2)*2*T;
      else wait;
      end if;
    end process;
    data: process                           —входные данные
    begin
    A <="111001"; B <="000101"; wait for N*2*T;  — проверка P1
    A <="110011"; B <="000101"; wait for N*2*T;  — проверка P2
    A <="111001"; B <="110101"; wait for N*2*T;  — проверка P3
    A <="001001"; B <="000101"; wait for N*2*T;  — проверка P4
    A <="000011"; B <="110101"; wait for N*2*T;  — проверка P5
    A <="001001"; B <="110011"; wait for N*2*T;  — проверка P6
      endsim := true;
      wait;
    end process;
OA: process (Clk, y1, y2, y3, y4, y5)    -— Эмуляция ОА
    begin
    if Clk'event and Clk = '1' then
    if  (y1 = '1') then C <= A;
    elsif(y2 = '1')then C<=C(5 downto 4)&((not C(3 downto 0))+1);
    elsif(y3 = '1')then C <= C + B;
    elsif(y4='1')then C<=C+(B(5 downto 4)&((not B(3 downto 0))+1));
    elsif(y5 = '1')then C <= C(5 downto 4)&((not C(3 downto 0))+1);
     end if;
    end if;
    end process;
    x1<= '1'when C(5 downto 4)="11" else '0';
    x2<= '1'when B(5 downto 4)="11" else '0';
    — Отображение_результата_в_виде десятичных_чисел
Adec<=-(ieee.std_logic_signed.conv_integer('0' & A(3 downto 0))) when A(5 downto 4)="11" else
    (ieee.std_logic_signed.conv_integer('0' & A(3 downto 0)));
Bdec<=-(ieee.std_logic_signed.conv_integer('0' & B(3 downto 0))) when B(5 downto 4)="11" else
    (ieee.std_logic_signed.conv_integer('0' & B(3 downto 0)));
Cdec<=-(ieee.std_logic_signed.conv_integer('0' & C(3 downto 0))) when C(5 downto 4)="11" else
    (ieee.std_logic_signed.conv_integer('0' & C(3 downto 0)));
end TB_ARCHITECTURE;
```

Рис. 6. Фрагмент тестовой программы TestBench для проведения ДЭ

Данная структура TestBench является универсальной и может использоваться в качестве шаблона для проведения верификации любых микропрограммных УА, при этом необходимо менять только ОА: process (для соответствующей МКП) и подавать соответствующие данные в data: process. Для удобства визуального наблюдения на Waveform входные данные и результат преобразуются в десятичные числа.

На рис.7 изображена временная диаграмма работы устройства сложения (ОА+УА) на основе эталонного кода УА, приведенного на рис.4.

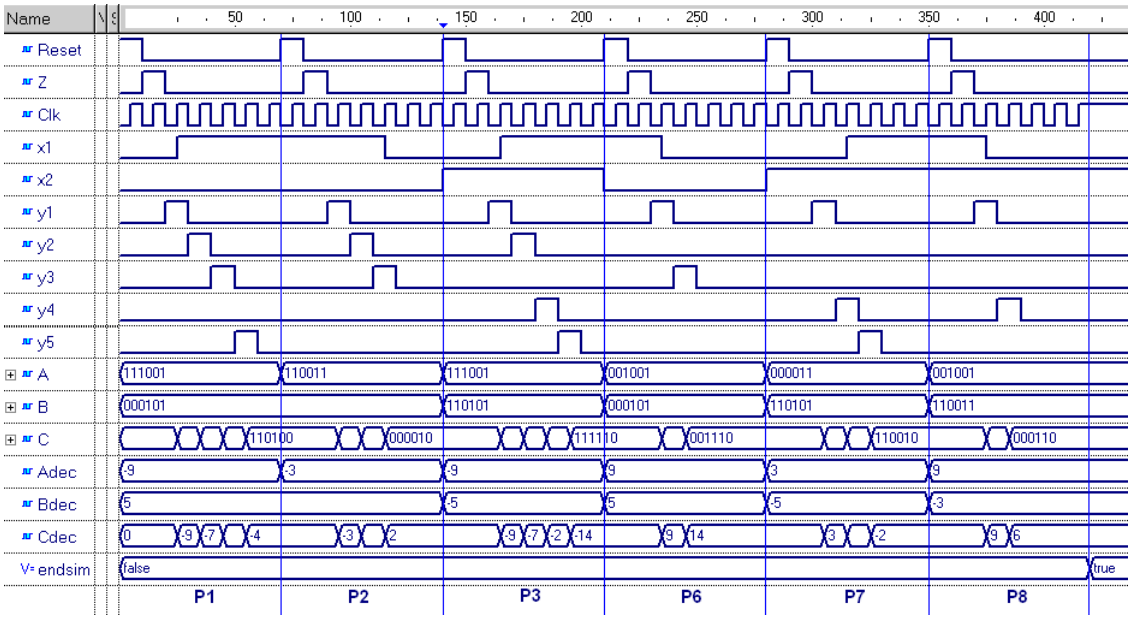


Рис. 7. Временная диаграмма моделирования работы эталонной HDL-модели устройства сложения с УА Мура

4. Поиск ошибок проектирования в HDL-модели управляющего автомата

Для демонстрации методики поиска ошибок проектирования внесем ошибку в эталонный HDL-код. Предположим, что вместо перехода a2 – a3 реализуется переход a2 – a5. При этом не выполняется микрооперация y3 при условии, что была выполнена y2. Фрагмент ошибочного кода приведен на рис.8.

```
when a2=> if x2='1' then NextState <= a4;
                    else NextState <= a5;
                    end if;
```

Рис. 8. Фрагмент ошибочного HDL-кода, реализующего переход {a2 – a5} вместо {a2 – a3}

Диагностический эксперимент проводится путем реализации системы проверок {P_1, P_2, P_3, P_6, P_7, P_8}. На рис.9 приведена временная диаграмма моделирования ошибочного кода. При анализе Waveform следует учитывать, что Cdec=Adec+Bdec в десятичном виде.

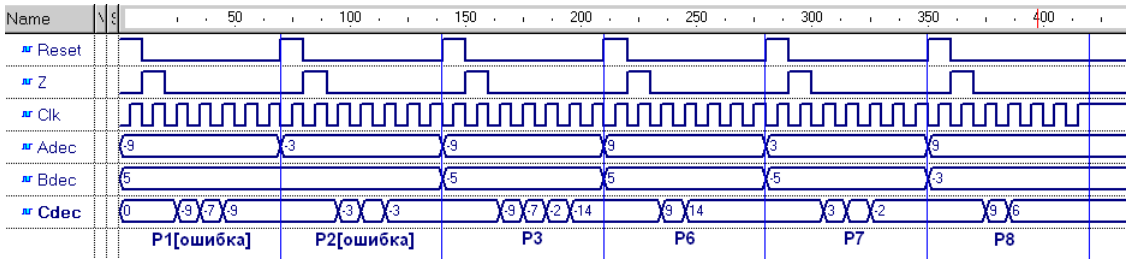


Рис. 9. Временные диаграммы проведения ДЭ для ошибочного HDL-кода УА

В результате анализа проведения ДЭ проверки P_1 и P_2 положительны (результат не совпал с эталоном), а остальные отрицательны, т.е. ВЭП будет V=(1, 1, 0, 0, 0, 0).

Место возникновения ошибки в HDL-коде (перечень подозреваемых дуг графа, D) определяется путем анализа маршрутов обхода графа переходов по формуле, где M_j – j-я строка таблицы проверок (таблица):

$$D = \cap|_{v_j=1} M_j - \cup|_{v_j=0} M_j, \quad (1)$$

$$D = \{a,b,d,f,h\} \cap \{a,b,d,m\} - \{a, b, g, h, j\} \cup \{a, c, m\} \cup \{a, e, g, h\} \cup \{a, e, k\} =$$
$$= \{a, b, d\} - \{a, b, c, e, g, h, j, k\} = \{d\}.$$

Таким образом, ошибочный оператор находится в группе операторов, связанных с состоянием a2 по ветви x2=0 (см. рис.1,б), что соответствует присутствующей ошибке проектирования.

5. Выводы

Предложен метод диагностирования HDL-модели микропрограммного автомата путем проведения диагностического эксперимента по обходу всех дуг графа управляющего автомата, начиная с начальной вершины. При этом осуществляется инициализация всех микроопераций операционного автомата, а операнды и результаты проверяются по спецификации реализуемой микропрограммы. Диагностический эксперимент проводится на примере микропрограмм сложения знаковых чисел в дополнительном модифицированном коде для управляющего автомата Мура. Введено понятие совместимых последовательностей микроопераций, которые обеспечивают проведение диагностического эксперимента для непротиворечивых числовых значений операндов. Локализация ошибок проектирования в HDL-модели управляющего автомата осуществляется до группы операторов HDL-кода, связанных с реализацией ошибочного перехода в графе автомата. Данные (оповестительные сигналы) на управляющий автомат подаются путем эмуляции функций операционного автомата в системе верификации HDL-моделей (Test Bench) САПР Active-HDL. Полученные результаты также проверялись в САПР VCS 2014.12 и Riviera-PRO EDU 2015.06.

Вопросы автоматизации выбора конкретных числовых значений операндов для квазиоптимального обхода всех дуг графа переходов управляющего автомата являются предметом дальнейших исследований.

Список литературы: 1. *Майоров С.А.* Структура электронных вычислительных машин / *С.А. Майоров, Г.И. Новиков*. Л.: Машиностроение, 1979. 384 с. **2.** *Шкиль А.С.* Модель процесса перехода от содержательного графа микропрограммы к графу автомата / *А.С. Шкиль, В.И. Хаханов, Е.В. Ковалев* // АСУ и приборы автоматики. 2000. Вып. 112. С. 112-120. **3.** *Шкиль А.С.* Поиск ошибок проектирования в HDL-моделях цифровых автоматов / *С. Альмадхоун, Е.Е. Сыревич, А.С. Шкиль* // Вестник Херсонского государственного технического университета. 2013. №2 (46). С. 377-383. **4.** *Шкиль А.С.* Автоматизация поиска ошибок проектирования в HDL-моделях конечных автоматов / *А.С Шкиль, Г.П. Фастовец, А.С Серокурова* // АСУ и приборы автоматики. 2014. Вып.168. С. 43-52.

Поступила в редколлегию 22.09.2015

Шкиль Александр Сергеевич, канд. техн. наук, доцент кафедры АПВТ ХНУРЭ. Научные интересы: диагностика цифровых систем, дистанционное образование. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 702-13-26.

Кулак Эльвира Николаевна, канд. техн. наук, доцент кафедры АПВТ ХНУРЭ. Научные интересы: автоматизированное проектирование цифровых автоматов. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 702-13-26.

Серокурова Анна Сергеевна, аспирантка кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика цифровых автоматов. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 702-13-26.