# КОМПЬЮТЕРНАЯ ИНЖЕНЕРИЯ

УДК 681.326

## ОБНАРУЖЕНИЕ ОШИБОК ПРОЕКТИРОВАНИЯ В HDL-МОДЕЛЯХ КОНЕЧНЫХ АВТОМАТОВ С ИСПОЛЬЗОВАНИЕМ СИНХРОНИЗИРУЮЩИХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

*ШКИЛЬ А.С., МИРОШНИК М.А., КУЛАК Э.Н., ГРЕБЕНЮК А.С., КУЧЕРЕНКО Д.Е.*

Предлагается метод обнаружения и локализации ошибок проектирования в HDL-моделях конечных автоматов с произвольными функциями выходов. Диагностический эксперимент проводится путем обхода всех дуг автомата Мили, начиная с начальной вершины, в том числе для автоматов неисключительного класса. Для обеспечения возврата автомата с возможной ошибкой проектирования в начальное состояние предлагается использовать синхронизирующие последовательности.

### 1. Введение

Наиболее сложным и затратным этапом в современном цикле проектирования цифровых устройств (ЦУ) является функциональная верификация, т.е. 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 системах автоматизированного проектирования радиоэлектронной аппаратуры (САПР РЭА) являются языки ее описания (Hardware Description Language, HDL). Поэтому объектом верификации является модель цифрового устройства, написанная на языке описания аппаратуры, т.е. HDL-код. Важным элементом процесса верификации HDL-моделей есть поиск места возникновения и исправление ошибок проектирования в случае различия между результатами моделирования HDL-кода и спецификацией на проектируемое ЦУ.

Одним из распространенных способов исходного описания специализированного цифрового вычислительного устройства обработки данных и управления является конечный автомат, а формой его представления - таблица переходов-выходов (ТПВ) и построенный на ее основе граф переходов автомата. Исходя из этого, актуальной является задача разработки процедур верификации, поиска и локализации ошибок проектирования в HDL-моделях конечных автоматов, представленных на языках описания аппаратуры, для табличных или графовых способов задания спецификации.

Одним из способов описания моделей ЦУ в форме конечных автоматов на языке VHDL является автоматный шаблон, т.е. способ описания моделей управляющих конечных автоматов, спецификация на которые задана ТПВ или графом переходов автомата (state diagram). Это специальная структура VHDL-модели, в которой функции переходов и выходов выделены в отдельные процессы (процесс), а назначение нового состояния осуществляется в специальном процессе, связанном с синхронизацией. Заметим, что автоматный шаблон является языковым отображением модели структурного автомата Хаффмена. Выделение фрагментов HDL-кода, описывающих поведение автоматов стилем «автоматный шаблон», позволяет определить ошибки проектирования типа «неправильный переход в графе переходов автомата». Для их обнаружения и локализации можно провести стандартный диагностический эксперимент (ДЭ) над автоматами (обход всех дуг для автомата Мили, обход всех вершин для автомата Мура). Рассматриваемый класс автоматов и способы его описания должны удовлетворять условиям, приведенным в [1]. Особо следует отметить, что для проведения ДЭ исправный и все множество неисправных автоматов должны составлять *исключительный класс*, т.е. иметь различные функции выходов для каждого состояния или перехода [2]. Для неисключительного класса автоматов однократный обход всех дуг графа не гарантирует обнаружения всех ошибок проектирования.

Исходя из двойственности HDL-модели, можно определить характер возможных ошибок, их влияние на конечную реализацию (устройство после синтеза) и методы поиска. Вместо терминов «дефект» и «неисправность», принятых в технической диагностике, в дальнейшем изложении будет использоваться понятие «ошибка проектирования». Для HDL-моделей вводится модель ошибки проектирования, соответствующая ошибке в любом операторном выражении, которая не относится к синтаксическим ошибкам. Наличие ошибки проектирования определяется путем сравнения результатов моделирования ошибочного кода с эталоном (спецификацией). Для HDL-кода моделей конечных автоматов можно определить такие типы ошибок проектирования: «замена оператора» (логического или арифметического), что соответствует ошибке в функции переходов или выходов, и «замена операнда» (в операторе назначения или условном операторе), что соответствует ошибке назначения нового состояния или выходного сигнала [3].

### 2. Подготовка диагностических экспериментов для HDL-моделей конечных автоматов

Предлагается следующая технология поиска ошибок проектирования в HDL-моделях конечных автоматов. На основании обхода графа пе-

реходов автомата строится тест и по результатам его прохождения и сравнения с эталоном спецификации (функцией выходов) строится вектор экспериментальных проверок (ВЭП). Затем путем анализа таблицы маршрутов обхода графа (матрицы проверок) и путем его анализа находится ошибочная дуга (дуги) в графе переходов. При этом ошибка в HDL-коде, вероятнее всего, находится во фрагменте кода автоматного шаблона, связанного с вершиной (состоянием), откуда исходит ошибочная дуга.

В качестве элементарной проверки $P_i$ при проведении ДЭ используется реализация определенного маршрута обхода графа. Результат элементарной проверки $v_i$ считается отрицательным (0), если терминальная вершина на этом маршруте достигнута, в противном случае результат считается положительным.

Для реализации стратегии обхода всех дуг графа строится дерево решений (дерево обхода графа переходов). Результатом проведения ДЭ по обходу графа является вектор экспериментальных проверок $V = (v_1, v_2, .. v_m)$, где m – число терминальных вершин дерева решений. При этом $v_i$=0, если функция выхода автомата совпадает с эталоном (тест прошел), и $v_i$=1, если не совпадает (тест не прошел).

Место возникновения одиночной ошибки в маршруте обхода графа переходов автомата находится по левой части формулы (1), а кратной – по правой, где $M_j$ – j-я строка матрицы маршрутов обхода графа переходов автомата и ВЭП:

$$D = \cap|_{v_j=1} M_j - \cup|_{v_j=0} M_j, \qquad D = \cup|_{v_j=1} M_j - \cup|_{v_j=0} M_j. \tag{1}$$

Для построения теста реализуется стратегия обхода всех дуг графа переходов конечного автомата, начиная с начальной вершины, при условии допустимости наличия более, чем одной дуги $a_i \Rightarrow a_j$ (смешанная стратегия). Данный подход предусматривает проведение так называемого «неразрушаемого» эксперимента, в котором в конце каждой проверки автомат логически или принудительно возвращается в начальное состояние. При этом проверяются все одиночные неисправности переходов, а также исправности функций возбуждения автомата, обеспечивающих эти переходы. ДЭ над HDL-моделью конечного автомата состоит в подаче на нее входных воздействий в соответствии с выбранной стратегией обхода содержательного графа переходов автомата, получении выходных реакций в виде выходных сигналов или списка состояний автомата на Waveform, или списка обхода графа в файле, и сравнении полученных реакций с эталоном визуальным или программным путем. На основании анализа результатов сравнения делается вывод о соответствии HDL-модели спецификации. ДЭ проводится с использованием системы верификации HDL-моделей (TestBench) в среде проектирования Active–HDL, модели автоматов представлены на языке VHDL [1].

Достаточно часто при задании спецификации на конечный автомат задаются только выходные значения на определенных входных воздействиях (ТПВ или функция выходов), а граф переходов будет строиться в ходе последующих этапов проектирования. В этом случае исправный или неисправный автоматы могут выходить за пределы исключительного класса автоматов, и тест, построенный для эталонного графа переходов, может не определять некоторые ошибки проектирования.

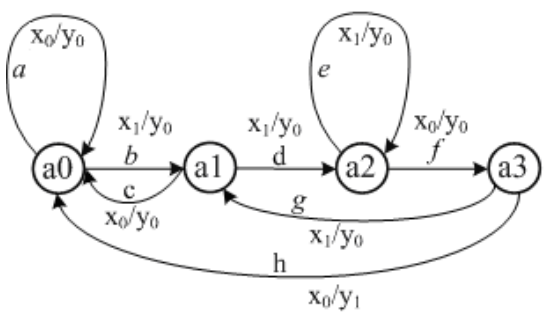
Для примера проведения ДЭ по поиску ошибок проектирования используется VHDL-модель автомата Мили, который выполняет распознавание набора 1100 во входной битовой последовательности.

На рис. 1 представлены ТПВ (а) и граф переходов (б) данного автомата Мили, а на рис. 2 - фрагмент архитектуры двухпроцессного автоматного шаблона его VHDL-модели без ошибок проектирования. Данный эталонный фрагмент кода приведен для ясности дальнейшего изложения.

Отметим, что в VHDL-модели входному сигналу x0 в коде соответствует data=0, а сигналу x1 – data=1, и, соответственно, для выходного сигнала y0 в коде соответствует y=0, а для y1 в коде – y=1. Это обусловлено особенностями графического редактора State Diagram в САПР Active-HDL.

| a | x0 | x1 |
|---|---|---|
| a0 | a0, y0 | a1, y0 |
| a1 | a0, y0 | a2, y0 |
| a2 | a3, y0 | a2, y0 |
| a3 | a0, y1 | a1, y0 |

а




б

Рис. 1. Таблица переходов-выходов (а) и граф переходов (б) автомата Мили

```
begin
p1: process (state, data)
begin
case state is
        when a0 =>
                if data='0' then nextstate<= a0; y<='0';
                                                else
nextstate<= a1;y<='0';
                end if;
        when a1 =>
                if data='1' then    nextstate<= a2; y<='0';
                                                else
nextstate<= a0; y<='0';
                end if;
                when a2 =>
                if data='0' then    nextstate<= a3; y<='0';
                                                else
nextstate<= a2; y<='0';
                end if;
                when a3 =>
                if data='0' then nextstate<= a0; y<='1';
                                                else
nextstate<= a1; y<='0';
                end if;
        end case;
end process;
p2: process (clk,reset)
begin
if reset='1' then state <= a0;
elsif clk'event and clk = '1' then state <= nextstate;
end if;
end process;
```

Рис. 2. Фрагмент эталонной VHDL-модели автомата Мили

Для описания графа переходов предложено использовать модифицированную матрицу смежности (рис. 3). Данная матрица характеризуется тем, что для каждой вершины графа (строки матрицы) в ячейках матрицы указываются имена дуг, соединяющих данную вершину с преемниками (столбцами). При этом в одной ячейке могут быть более одной дуги (мультиграф) [1].

| | a0 | a1 | a2 | a3 |
|---|---|---|---|---|
| a0 | a | b | | |
| a1 | c | | d | |
| a2 | | | e | f |
| a3 | h | g | | |

Рис. 3. Матрица смежности для графа переходов на рис. 1

Для реализации стратегии обхода всех дуг графа (в режиме «ручного» построения тестов) строится бинарное дерево решений (дерево обхода графа) по следующему алгоритму.

1. Построение дерева обхода графа начинается с начальной вершины a0. Анализируется первая строка матрицы смежности и составляется список вершин-преемников.
2. Из списка преемников выбирается первая вершина и включается в дерево. Соответствующая ячейка в матрице (дуга) помечается.
3. Для строки, соответствующей этому номеру, составляется список непомеченных преемников.
4. Построение маршрута в графе завершается терминальной (конечной) вершиной. Вершина в дереве становится терминальной, если:

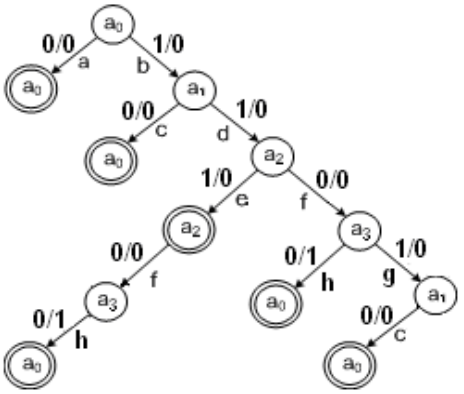
– входящая дуга является петлей (элемент главной диагонали матрицы смежности);

– очередная вершина-преемник является начальной вершиной a0 (завершен цикл работы автомата);

– все ячейки в очередной строке (при выполнении п.3) окажутся помеченными.

5. Пункты 2, 3 и 4 повторяются до тех пор, пока все дуги (ячейки матрицы смежности) не окажутся помеченными.
6. Для терминальных вершин с петлями достраиваются маршруты возврата в начальную вершину.

На рис. 4 показано дерево решений для обхода дуг графа переходов автомата (а) и маршруты обхода (б) для эталонной модели (без ошибок проектирования). При этом следует отметить, что число маршрутов обхода графа переходов равно числу терминальных вершин (вершин, не имеющих преемников) дерева решений. На рис. 4, а терминальные вершины отмечены двойной окружностью.




а

| | |
|---|---|
| 1 | a0-a-a0 |
| 2 | a0-b-a1-c a0 |
| 3 | a0-b-a1-d-a2-e-a2-f-a3-h-a0 |
| 4 | a0-b-a1-d-a2-f-a3-g-a1-c-a0 |
| 5 | a0-b-a1-d-a2-f-a3-h-a0 |

б

| M | data | a | b | c | d | e | f | g | h | y |
|---|---|---|---|---|---|---|---|---|---|---|
| $P_1$ | 00 | 1 | | | | | | | | 0 |
| $P_2$ | 10 | | 1 | 1 | | | | | | 0 |
| $P_3$ | 11100 | | 1 | | 1 | 1 | 1 | | 1 | 1 |
| $P_4$ | 11010 | | 1 | 1 | 1 | | 1 | 1 | | 0 |
| $P_5$ | 1100 | | 1 | | 1 | | 1 | | 1 | 1 |

в

Рис. 4. Стратегия обхода всех дуг графа переходов: а – дерево решений; б – маршруты обхода графа; в – матрица элементарных проверок

Стратегия обхода всех дуг графа переходов с учетом входных воздействий и значений выходов может быть представлена в виде матрицы проверок ( рис.4, в). В последнем столбце данной матрицы указываются эталонные значения функции выходов конечного автомата.

Анализируя матрицу проверок, следует отметить, что проверка $P_2$ дополняет $P_4$ для возврата автомата в начальное состояние, а $P_3$ полностью включает в себя $P_5$. Это позволяет построить матрицу совмещенных проверок, которая представлена на рис. 5.

| B | data | a | b | c | d | e | f | g | h | y |
|---|---|---|---|---|---|---|---|---|---|---|
| $P_1$ | 00 | 1 | | | | | | | | 0 |
| $P_2$ | 10 | | 1 | 1 | | | | | | 0 |
| $P_4 + P_2$ | 11010 | | 1 | 1 | 1 | | 1 | 1 | | 0 |
| $P_3 + P_5$ | 11100 | | 1 | | 1 | 1 | 1 | | 1 | 1 |

Рис. 5 Матрица совмещенных проверок для эталонного графа

При этом ВЭП будет $V=(P_1, P_2, P_4 + P_2, P_2 + P_5)$.

Для проведения ДЭ использована встроенная система верификации HDL-моделей (TestBench) в среде проектирования Active–HDL. Результаты моделирования матрицы проверок (waveform) для эталонной модели приведены на рис. 6.

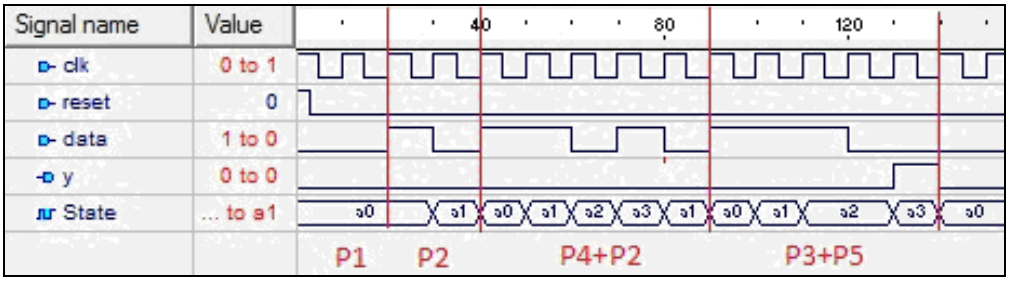



Рис. 6. Waveform для эталонной HDL-модели

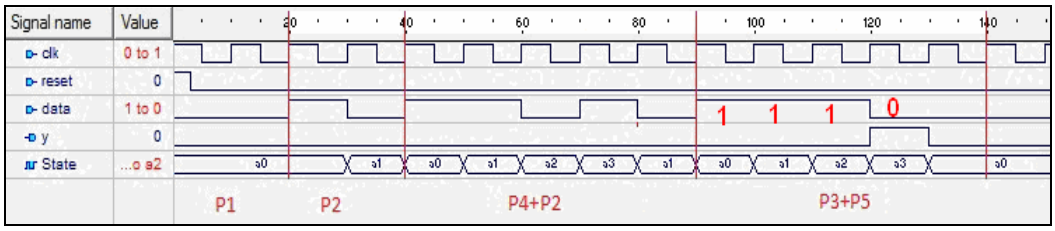



Рис. 7. Waveform для ошибки назначения нового состояния

### 3. Проведение диагностирования HDL-моделей конечных автоматов

Проведем ряд ДЭ по поиску ошибок проектирования в VHDL-моделях автоматов.

Рассмотрим ошибочную VHDL-модель с ошибкой проектирования типа «ошибочное назначение нового состояния» (фрагмент кода на рис. 7).

```
when a2 =>
        if data='0' then     nextstate<= a3; y<='0';
        else nextstate<= a3; y<='0';          --      (вместо
nextstate <= a2)
        end if;
```

Рис. 8. Фрагмент VHDL-модели автомата с ошибочным оператором назначения

Результаты моделирования VHDL-модели с ошибкой назначения приведены на рис. 8.

Результат четвертой проверки не совпадает с эталоном. Если вернуться к спецификации, то можно отметить, что y может быть равен 1 только при подаче на вход последовательности 1100. На рис. 9 видим, что y равен 1 при подаче входной последовательности 1110, т.е. ВЭП будет *V*=(0, 0, 0, 1).

```
when a2 =>
        if data='1' then nextstate<= a3;y<='0'; -- (вместо
data='0')
        else  nextstate<= a2;y<='0';
        end if;
```

Рис. 9. Фрагмент VHDL-модели с ошибочным оператором if

Место возникновения ошибки в маршруте обхода графа переходов находится по формуле. D = {b, d, e, f, h} – {a} ∪ {b, c} ∪ {b, c, d, f, g} = {b, d, e, f, h} – {a, b, c, d, f, g} = {e, h} и ошибочные дуги *e* и *h* в графе переходов определены.

Чтобы найти место возникновения ошибки проектирования, нужно вернуться к VHDL-коду модели и выполнить визуальное инспектирование участка кода для вычисления ошибочной дуги. Исходя из результатов сравнения эталонной waveform и waveform, полученной после моделирования кода с ошибкой, находится ошибочный оператор (else nextstate <= a3;).

Рассмотрим ошибочную VHDL-модель с ошибкой проектирования типа «ошибка в условном операторе if » (фрагмент кода см. на рис. 9). Результаты моделирования данной ошибочной мо-

дели приведены на рис. 10.

На основании анализа Waveform ВЭП будет *V*=(0, 0, 1, 1). Место возникновения ошибки в маршруте обхода графа переходов находится по формуле: D= {b, c, d, e, f}∩ {b, d, e, h} – {a} ∪ {b, c } = { b, d, e } – {a, b, c} = { d, e }, т.е. ошибочные дуги *d* и *e* в графе переходов найдены.

Рассмотрим ошибочную VHDL-модель с ошибкой проектирования типа «ошибка в назначении выходного сигнала» (фрагмент кода на рис. 11).

```
when a3 =>
        if data='0' then nextstate<= a0; y<='1';
                        else nextstate<= a1; y<='1';
         -- (вместо y<='0')
        end if;
```

Рис. 11. Фрагмент VHDL-модели автомата с ошибкой в назначении выходного сигнала

Результаты моделирования VHDL-модели с ошибкой в назначении выходного сигнала приведены на рис. 12.

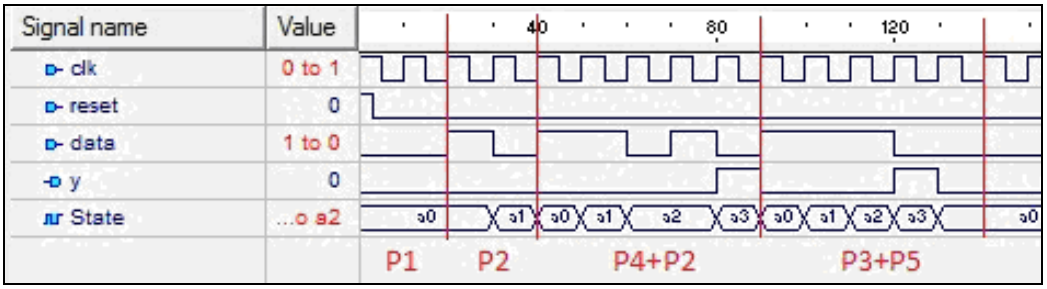



Рис. 10. Waveform для ошибки в условном операторе if «data='1' вместо data='0'»

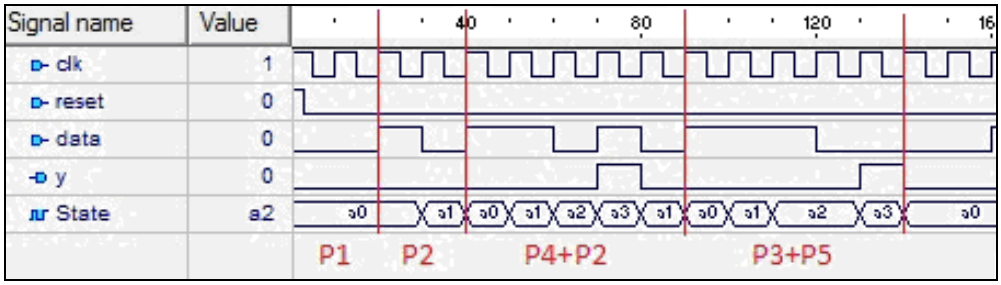



Рис. 12. Waveform для ошибки в назначении выходного сигнала «y=1 вместо y=0»

На основании анализа Waveform ВЭП будет *V*=(0,0,1,0). Место возникновения ошибки в маршруте обхода графа переходов находится по формуле: D = { b, c, d, f, g}– {a} ∪ {b, c }∪{ b, d, e, f, h } = { b, c, d, f, g } – {a, b, c, d, e, f, h} = { g }, т.е. ошибочна дуга *g* в графе переходов найдена.

Рассмотрим еще одну ошибочную VHDL-модель с ошибкой проектирования типа «ошибочное назначение нового состояния» (фрагмент кода на рис. 13).

```
when a1 =>
        if data='1' then    nextstate<= a2; y<='0';
        else nextstate<= a3; y<='0';        -- (вместо
nextstate <= a0)
        end if;
```

Рис. 13. Фрагмент VHDL-модели автомата с ошибочным оператором назначения

Результаты моделирования VHDL-модели с «ошибкой назначения нового состояния» приведены на рис. 14.

Результат по функции выходов (y) совпал с эталоном, что предполагает отсутствие ошибки в VHDL-модели. Но если внимательно проанализировать Waveform, то можно заметить, что после третьей проверки ($P_4 + P_2$) автомат не возвращается в исходное состояние a0, но функция выходов при этом дает правильный результат.

Данный результат обусловлен тем, что рассматриваемый автомат не принадлежит к так называемому исключительному классу автоматов, которые имеют различные функции выходов для каждого состояния (каждой дуги для автомата Мили). В данном автомате для всех дуг, кроме дуги h, выход y=0 (см. ТПВ и граф на рис. 1).

Рассматриваемая ошибка проектирования (nextstate <= a3 вместо nextstate <= a0) порождает новую дугу *c* из состояния a1, но функция выходов у них одинакова. Подтверждающий это новый граф переходов, полученный инструментальными средствами Code2Graphics в составе Active-HDL, представлен на рис. 15.

Данный пример подтверждает, что построение теста по простой стратегии «однократного обхода всех дуг графа» по эталонному графу не гарантирует нахождения всех ошибок проектирования в HDL-модели для неисключительного класса автоматов.

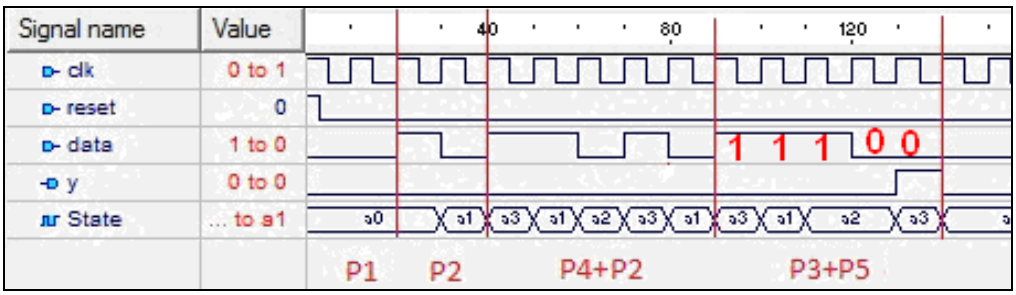



Рис. 14. Waveform для ошибки назначения «a3 вместо a0»

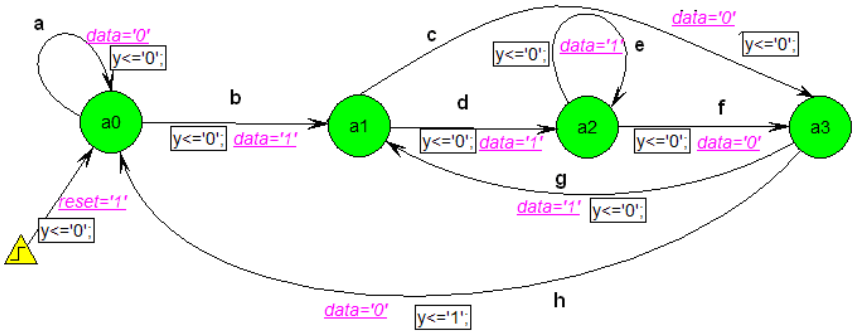



Рис. 15. Ошибочный граф переходов для ошибочного кода на рис. 13

**4. Использование синхронизирующих последовательностей в диагностических экспериментах**

Как упоминалось выше, основной задачей при проведении «неразрушаемого» диагностического эксперимента является возможность возврата автомата в начальное состояние независимо от результата очередной проверки. Для этих целей целесообразно использовать синхронизирующие последовательности конечных автоматов [4].

Входная последовательность $X_S$ автомата, которая устанавливает его в определенное конечное состояние независимо от состояния выхода и начального состояния, называется синхронизирующей последовательностью.

Если автомат $W = < X, A, Y, \delta, \lambda, a_0 >$ задан таблицей переходов-выходов, то из определения следует, что он имеет синхронизирующую последовательность тогда и только тогда, когда существует входная последовательность $X_s$ такая, что $\delta(a_i, X_j) = a_0$, $\forall a_i \in A$, $a_0 \in A$. Множество переходов $\delta(a_i, X_s) = a_0$, $\forall a_i \in A$, автомата определяет отображение множества его состояний $A$ в некоторое определенное состояние $a_0$ при подаче на автомат входной последовательности $X_s$, т.е. $a_i \xrightarrow{X_s} a_0$.

Синхронизирующая последовательность для заданного автомата может быть найдена из синхронизирующего дерева, которое является деревом-преемником, построенным по определенным правилам [4]. На рис. 16 показано построение синхронизирующих последовательностей для рассматриваемого автомата Мили. При этом следует отметить, что для состояния a3 синхронизирующей последовательности не существует.

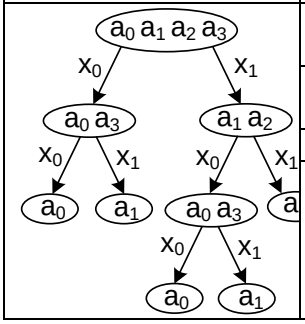
| Синхронизирующее дерево | Конечное состояние автомата | Синхронизирующие последовательности |
|---|---|---|
|  | a0 | 00<br>100 |
| | a1 | 01<br>101 |
| | a2 | 11 |
| | a3 | 110 (a0 – a3)<br>10 или 0110 (a1 – a3)<br>1 (a2 – a3) |



Рис. 16. Синхронизирующие последовательности для автомата Мили

Таким образом, если говорить о принудительном возврате в состояние a0, то синхронизирующие последовательности {00, 100} должны быть в конце каждого входного слова элементарной проверки. Используя принцип построения совмещенных проверок [5], матрица эталонных проверок (см. рис.5) примет вид, показанный на рис. 17, где для проверок $P_2$ и $(P_4 + P_2)$ выделены фрагменты «дополнений» синхронизирующих последовательностей.

| B | data | a | b | c | d | e | f | g | h | y |
|---|---|---|---|---|---|---|---|---|---|---|
| $P_1$ | 00 | 1 | | | | | | | | 0 |
| $P_2$ | 100 | 1 | 1 | 1 | | | | | | 0 |
| $P_4 + P_2$ | 110100 | 1 | 1 | 1 | 1 | | 1 | 1 | | 0 |
| $P_3 + P_5$ | 11100 | | 1 | | 1 | 1 | 1 | | 1 | 1 |

Рис. 17. Матрица эталонных совмещенных проверок с учетом добавления синхронизирующих последовательностей

Результаты моделирования VHDL-модели с ошибкой назначения нового состояния, с применением синхронизирующих последовательностей для возврата в состояние a0, приведены на рис. 18.

Анализ данной Waveform позволяет сделать два вывода. Во-первых, после завершения каждой проверки, даже для ошибочной VHDL-модели, автомат возвращается в начальное состояние, т.е. выполняется «неразрушаемый» ДЭ. Во-вторых, для проверок $P_2$ и $(P_4 + P_2)$ с синхронизирующими последовательностями выходное значение y не совпадает с эталоном, т.е. ВЭП будет $V$=(0, 1, 1, 0).

Место возникновения ошибки в маршруте обхода графа находится по формуле: D = {a, b, c}∩{ b, c, d, f, g}– {a} ∪ { b, d, e, f, h } = { b, c} – {a, b, d, e, f, h} = { c }, т.е. ошибочная дуга *c* в графе переходов найдена. Рассматривая строки VHDL-кода, соответствующие дугам, исходящим из a1, находим ошибочный оператор назначения nextstate <= a3.

Таким образом, показано, что использование синхронизирующих последовательностей позволяет строить «неразрушаемый» диагностический эксперимент и находить ошибки проектирования в том числе для конечных автоматов, не принадлежащих исключительному классу.

Следует отметить, что вопросы существования синхронизирующих последовательностей для состояния a0 произвольного автомата, вопросы принципиальной достижимости a0 в ошибочном графе не являются предметом данной статьи и требуют отдельного рассмотрения.

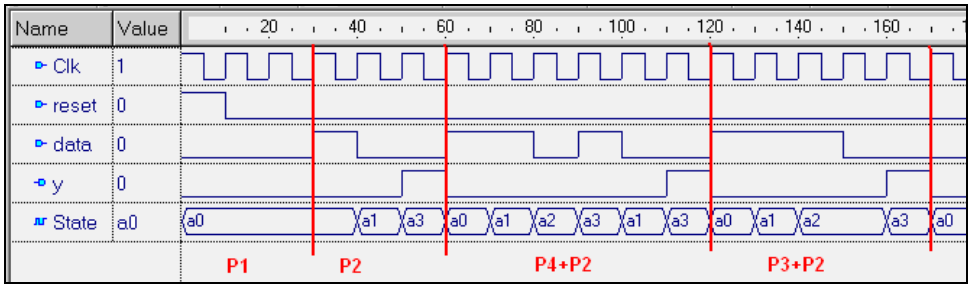



Рис. 18. Waveform для ошибки назначения «a3 вместо a0» с синхронизирующими последовательностями

## 5. Выводы

Проведенные диагностические эксперименты над VHDL-моделями конечных автоматов в форме автоматного шаблона показали, что однократный обход всех дуг графа переходов автомата Мили с возвратом в начальное состояние позволяет обнаружить и локализовать ошибки проектирования в HDL-коде типа «замена оператора» или «замена операнда». Для исключительного класса автоматов (каждая дуга имеет уникальную функцию выходов) однократный обход позволяет обнаружить одиночные и кратные ошибки проектирования, изменяющие функции переходов и выходов автомата.

Для неисключительного класса автоматов при проведении «неразрушаемого эксперимента» (с организацией возврата в начальную вершину $a_0$)) предложено использовать синхронизирующие последовательности, переводящие автомат из любого состояния в заданное. Предложенный подход позволил обнаруживать и локализовать ошибки проектирования для HDL-моделей конечных автоматов с произвольной функцией выходов.


**Литература: 1.** *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. 2. Гилл А. Введение в теорию конечных автоматов. М.: Наука, 1966. 272 с. **3.** *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. **4.** *Бережная М.А.* Синхронизирующие последовательности в конечных детерминированных автоматах / *М.А. Бережная* // Вестник НТУ “ХПИ”. 2008. № 57. С. 7-15. **5.** *Шкиль А.С.* Методы поиска ошибок проектирования в языковых моделях управляющих автоматов / *А.С. Шкиль, Э.Н. Кулак, А.С. Гребенюк (Серокурова)* // Сборник статей научно-информационного центра «Знание» по материалам XIX международной заочной научно-практической конференции: «Развитие науки в XXI веке», 2 часть, Харьков, 2016. Х.:

Научно-информационный центр «Знание», 2016. С. 57-65.

**Transliterates bibliography:**

1. *Shkil A.S.* Poisk oshibok proektiro-vaniya v HDL-modelyah tsifrovyih avtomatov / S. Almadhoun, E.E. Syirevich, A.S. Shkil // Vestnik Hersonskogo gosudarstvennogo tehnicheskogo universiteta. 2013. #2 (46). S. 377-383.

**2.** *Gill A.* Vvedenie v teoriyu konechnyih avtomatov. M.: Nauka, 1966. 272 s.

**3.** Shkil A.S. Avtomatizatsiya poiska oshibok proektirovaniya v HDL-modelyah konechnyih avtomatov / A.S Shkil, G.P.Fastovets, A.S.Serokurova // ASU i priboryi avtomatiki. 2014. Vyip.168. S. 43-52.

**4.** *Berezhnaya M.A.* Sinhroniziruyuschie posledovatelnosti v konechnyih determinirovannyih avtomatah / M.A. Berezhnaya // Vestnik NTU “HPI”. 2008. # 57. S. 7-15.

**5.** *Shkil A.S.* Metodyi poiska oshibok proektiro-vaniya v yazyikovyih modelyah upravlyayuschih avtomatov / A.S. Shkil, E.N. Kulak, A.S. Grebenyuk (Serokurova) // Sbornik statey nauchno-informatsionnogo tsentra «Znanie» po materialam XIH mezhdunarodnoy zaochnoy nauchno-prakticheskoy konferentsii: «Razvitie nauki v XXI veke», 2 chast, Harkov, noyabr 2016 – H.: Nauchno-informatsionnyiy tsentr «Znanie», 2016. – C. 57-65.

Поступила в редколлегию 23.09.2016
**Рецензент:** д-р техн. наук, проф. Кривуля Г.Ф.

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

**Мирошник Марина Анатольевна,** д-р техн. наук, зав. кафедрой информационных технологий Украинского государственного университета железнодорожного транспорта. Адрес: Украина, 61001, Харьков, пл. Фейєрбаха, 7, тел. 710-30-61.

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

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

**Кучеренко Дария Ефимовна**, канд. техн. наук, доцент кафедры АПВТ ХНУРЭ. Научные интересы: языки описания аппаратуры, экспертные системы, нечеткая логика. Адрес: Украина, 61166, Харьков, пр. Науки, 14, тел. 702-13-26.

**Shkil Alexander Sergeevich**, PhD, Associate Professor, Design Automation Department, KNURE. Scientific interests: diagnostics of digital systems, distance education. Address: Ukraine, 61166, Kharkiv, Nauka Avenue, 14, tel. 702-13-26.

**Miroshnik Marina Anatolievna,** Dr. of Tekhn. Sciences, Head of Department of Information Technologies, Ukrainian State University of Railway Transport. Address: Ukraine, 61001, Kharkov, pl. Feyerbacha, 7, tel. 710-30-61.

**Kulak Elvira Nikolaevna,** PhD, Associate Professor, Design Automation Department, KNURE. Scientific interests: automated design of digital machines. Address: Ukraine, 61166, Kharkiv, Nauka Avenue, 14, tel. 702-13-26.

**Hrebenyuk Anna Sergeevna,** post-graduate student, Design Automation Department, KNURE. Scientific interests: technical diagnostics of digital machines. Address: Ukraine, 61166, Kharkiv, Nauki Avenue, 14, tel. 702-13-26.

**Kucherenko Daria Yephimovna,** PhD, Associate Professor, Design Automation Department, KNURE. Scientific interests: languages of apparatus description, expert systems, fuzzy logic. Address: Ukraine, 61166, Kharkiv, Nauki Avenue, 14, tel. 702-13-26.