УДК 681.326: 519.713

МОДЕЛИ ЦИФРОВЫХ АВТОМАТОВ ДЛЯ ПРОЕКТИРОВАНИЯ ТЕСТОВ В СРЕДЕ ACTIVE-HDL

ХАХАНОВ В.И., ШКИЛЬ А.С., КОВАЛЕВ Е.В.

Представлены модели цифровых автоматов в среде проектирования Active-HDL. Предлагается метод обеспечения переходов при построении тестов по граф-модели автомата путем выполнения обратной импликации по функции возбуждения. Функция возбуждения, заданная в виде языковых конструкций VHDL, преобразуется в предикатную функцию, задаваемую кубическими покрытиями в многозначном алфавите.

1. Введение

Проектирование вычислительных устройств ориентируется на два основных направления. Первое связано с новыми технологиями разработки универсальных и специализированных процессоров для компьютерных систем большой производительности. Второе направление относится к области разработки и проектирования специализированных вычислительных управляющих устройств на основе использования программируемых логических интегральных схем (ПЛИС). Заметный прогресс в росте производства и применения ПЛИС в последние несколько лет связан со снижением их стоимости за счет внедрения технологии Hardware-Software Cooperation, высокой степенью интеграции (до 12 млн. вентилей на кристалле), уменьшением времени реализации проекта (2 - 4 недели) на основе использования Field Programable Gate Array (FPGA), Complex Programable Logic Device (CPLD), наличием эффективных программных средств автоматизированного проектирования цифровых вычислительных устройств от описания проекта на системном уровне до получения карты прошивки.

При существующем многообразии исходных форм описания проектов можно выделить наиболее популярные в мире: аналитические – языки описания аппаратуры, графические или визуальные – иерархические цифровые структуры и схемы, графы переходов автоматов.

Одной из распространенных форм исходного описания специализированного цифрового вычислительного устройства является граф переходов автомата. Достоинства упомянутой формы заключаются в наглядности и технологичности представления функций, которые просто преобразуются в табличную форму описания автомата в виде кубических покрытий. Поэтому такой способ представления проекта присутствует в системах проектирования ведущих фирм мира.

Одной из важных задач, которая должна эффективно решаться в любой САПР ПЛИС, является генерация тестов на функциональном уровне описания в виде графов переходов автомата для класса одиночных неисправностей переходов [1].

В отношении существующих стратегий генерации тестов по модели графа переходов отметим, что любая из них реализуется путем обхода его вершин или дуг. При этом возникает задача обеспечения реализации требуемого перехода, соответствующего дуге графа, при условии наличия нетривиальных функций возбуждения, определяемых форматом операторов языка описания аппаратуры [2]. Задача обеспечения условий перехода сводится к выполнению обратной импликации на предикатной [3] функции возбуждения с использованием кубических покрытий для описания примитивов в многозначном алфавите.

2. Концепция представления моделей цифровых объектов

Концептуальная модель объекта, оперирующая дискретной информацией, представлена триадой компонентов [1]:

$$F = <f, t, h>,$$

где f, t, h – дискретные параметры описания функций, времени, структуры.

Компонент f отображает многообразие аналитических, табличных и графических форм представления моделей. Относительно t классификация моделей определяет синхронные (не учитывающие временные параметры объектов и примитивов), асинхронные (использующие реальные или модельные задержки элементов), дельта-троичные (учитывающие модельные задержки примитивных элементов (ПЭ) и разброс времени переключения входных сигналов объекта) и с нарастающей неопределенностью (момент переключения входных сигналов элементов учитывает разброс параметров задержки прохождения сигналов от внешних входов к выходам). Параметр h классифицирует степень подробности задания структуры и идентифицирует: функциональную или автоматную модель для представления поведения комбинационного или последовательностного ПЭ; структурно-функциональную или итеративную, а также чистую структуру, представляющую собой орграф.

Проблема верификации реализации проекта в целях анализа тождественности функций, структуры временных соотношений для конкретной спецификации рассматривается в рамках самой полной f-t-h-структуры, в то время как генерация проверяющих тестов вполне может обходиться f-h-моделью объекта.

Такой вывод находит подтверждение в публикациях последних трех лет в области проектирования различных форм описания поведения цифровых объектов, практического опыта разработки упомянутых систем и ознакомления с техническими характеристиками систем моделирования и генерации тестов. Применение структурно-функциональных (h-f)-моделей вентильного, автоматного, алгоритмического уровней детализации является достаточным для решения задач, логического синхронного моделирования, исправного поведения и неисправностей, генерации статических тестов, проектирования алгоритмов контроля и поиска дефектов.

Параметр времени в таких структурах – понятие относительное, определяющее фреймы (такты) автоматных состояний. Длительность каждого модельного такта есть величина, всегда превосходящая максимальную задержку цифровой схемы. При таких условиях адекватность структуры реальному объекту уменьшается по номинальным временным характеристикам, но вследствие этого появляется возможность реального решения задач диагностирования в пространстве <функция, структура> при неявном задании модельного времени.

Одно из возможных взаимодействий пространства функций и структуры во времени (модельном) представлено универсальной концепцией автомата первого рода, который дифференцируется на управляющую и операционную части: SM={CM, OM}.

В этом случае управляющий автомат на поведенческом уровне представлен множеством входных, внутренних и выходных состояний:

$$X = \{X_1,...,X_i,...,X_m\},$$
$$Y = \{Y_1,...,Y_j,...,Y_h\}, \quad (1)$$
$$Z = \{Z_1,...,Z_r,...,Z_k\}.$$

Каноническое задание управляющего автомата определяется функциями переходов и выходов автомата первого рода:

$$Y_t = f(X_t, Y_{t-1}); \quad Y_t = g(X_t, Y_{t-1}), \quad (2)$$

где $t-1, t$ – фреймы автоматного времени.

В процессе проектирования конечного автомата разработчик не заботится о строгом разделении на управляющую и операционную части. Его целью является уменьшение времени проектирования и аппаратурных затрат автомата в целом. Для этого используются дополнительные переменные памяти в виде оповестительных сигналов. В этом случае автоматная модель последовательностного примитива представляется в виде

$$M=<X, Y, Z, f, g>, \quad (3)$$

где $X=(X_1,X_2,...,X_i,...,X_m)$, $Y=(Y_1,Y_2,...,Y_i,...,X_h)$, $Z=(Z_1,Z_2,...,Z_i,...,Z_k)$ – множества входных, внутренних и выходных автоматных переменных, отношения между которыми описываются обобщенными уравнениями конечного автомата:

$$Y(t)=f[X(t-1), X(t), Y(t-1), Z(t-1)];$$
$$Z(t)=g[X(t-1), X(t), Y(t-1), Y(t), Z(t-1)]. \quad (4)$$

Переменные Z(t) отличаются от Y(t) тем, что первые наблюдаемы по выходным линиям, а Y(t) в этом смысле есть внутренние. Формат автоматных переменных, соответствующий (4), для записи функции имеет следующий вид:

X(t–1)	Y(t–1)	Z(t–1)
X(t)	Y(t)	Z(t)

.

Конечный автомат произвольной сложности может быть представлен таблицей истинности или кубическим покрытием, как одной из наиболее простых и технологичных для компьютерного анализа форм описания. В этом случае функциональный последовательностный примитивный элемент задается компонентами:

$$F^2 = <(t-1,t),(X,Z,Y),\{A^2\}>, \quad (5)$$

где (t-1,t) – два автоматных соседних такта в описании функции; (X,Z,Y) – векторы входных, внутренних и выходных переменных; $\{A^2\}$ – двухтактный алфавит описания состояний (переходов) автоматных переменных [1].

3. Модель автомата для среды проектирования Active-HDL

Применительно к системе Active-HDL автоматная модель существенно усложняется. Если рассматривать операционное устройство преобразования информации в виде декомпозиции операционного и управляющего автомата (УА), то объект исследования – УА, входом которого являются оповестительные сигналы X. Для анализа УА очень важно знать закон формирования указанных сигналов X, что особенно важно при работе с небулевыми алфавитами входных сигналов. Но на этапе функционального моделирования (анализа) проектировщик обычно не имеет полной информации об операционном автомате (ОА). Поэтому целесообразно включить в рассматриваемую модель УА ту часть ОА, которая связана только с формированием оповестительных сигналов и не затрагивает структур обработки данных. Эта модель представлена на рис. 1.

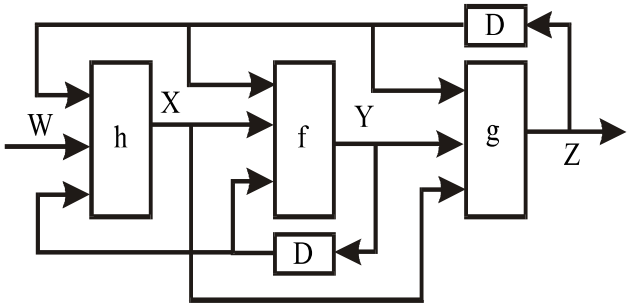

Рис. 1. Модель автомата для Active-HDL

Функции операционного устройства, которое далее будем именовать как W-автомат W=<X, Y, Z, f, g, h>, определяются характеристическими уравнениями:

$$Y_t = f(X_t, Y_{t-1}, Z_{t-1});$$
$$Z_t = g(X_t, Y_{t-1}, Z_{t-1}); \quad (6)$$
$$X_t = h(W_t, Y_{t-1}, Z_{t-1}).$$

Здесь h представляет собой функцию возбуждения или инициализации переходов в конечном автомате, где $X_t = \{0,1\}$ – выходная функция для активизации перехода (если $X_t = 1$); W – управляющие воздействия в стандарте языка VHDL [2,4]:

$$W= \{W^{BO}, W^{BI}, W^{BIV}, W^{STDL}, W^{STDLV}, W^{I}\}, \quad (7)$$

где $W^{BO} = \{false, true\}$ – переменные типа boolean; $W^{BI} = \{0,1\}$ – переменные типа bit; $W^{BIV} = (W^{BI}...W^{BI})$ – переменные типа bit_vector; $W^{STDL} = \{X$ – Forcing Unknown, 0 – Forcing 0, 1– Forcing 1, Z – High Impedance, W – Weak Unknown, L – Weak 0, H –

Weak 1, "–" – Don't care} – переменные типа std_logic; $W^{STDLV} = (W^{STDL}...W^{STDL})$ – переменные типа std_logic_vector; W^{I} = [-2147483647 – 2147483647] – переменные типа integer.

В общем случае для W-автомата **процедура генерации тестов** предполагает выполнение четырех шагов.

1. Построение теста обхода всех дуг графа по таблице переходов.

2. Решение задачи обратной импликации по функции X=h(W) для вычисления входных последовательностей в терминах стандарта VHDL.

3. Решение задачи установки во времени и в пространстве, если для активизации переходов используются сигналы из множества Z.

4. Модификация пути решения установочной задачи в случае отсутствия решения для выбранных сигналов из множества Z.

Проблема получения теста обхода всех вершин и дуг графа переходов связана с решением задачи нахождения условий перехода:

$$(X_j, Y_i, Z_j) \Rightarrow (Y_i \to Y_j).$$

Формально она сводится к выполнению обратной импликации на функции возбуждения $X_{ij} = (W_j, Y_i, Z_i)$ $(X_{ij} = \{0,1\})$ для рассматриваемого перехода $Y_i \to Y_j$. Если значение $X_{ij} = 1$, то переход выполняется, в противном случае ($X_{ij} = 0$) – нет. В общем случае входная переменная $W_r \in (X_j, Y_i, Z_i)$ определяется типами:

$$W_r \in W = \{W^{BO}, W^{BI}, W^{BIV}, W^{STDL}, W^{STDLV}, W^{I}\}. \quad (8)$$

Определение 1. Предикатным [3] называется примитив (W-элемент), имеющий в качестве входных переменных типы из множества W, а значение выхода определено на множестве сигналов {false, true} или {0, 1}.

Определение 2. Схемная структура, составленная из предикатных элементов, определяет предикатную функцию возбуждения или h-функцию. Внешние входные переменные такой структуры определены на множестве W. Значения внутренних и выходных линий предикатной функции равны {0, 1}. Переменные типа $\{W^{BIV}, W^{STDL}, W^{STDLV}, W^{I}\}$ могут быть только внешними входами. Аналитическая запись предикатной функции включает скобки, знаки конъюнкции, дизъюнкции, отрицания, другие логические и арифметические операции, поддерживаемые в стандарте VHDL.

4. Кубическая форма представления предикатных уравнений

Любому предикату, представленному в аналитической форме, можно поставить в соответствие таблицу истинности или кубическое покрытие (КП). Таким образом, функция возбуждения есть взаимосвязанная структура типов функций, каждая из которых на выходе имеет двоичное значение. Это означает, что к такой схеме можно применить модифицированный π -алгоритм, который по единичному состоянию функции возбуждения определит решение в виде входных сигналов, обеспечивающих требуемый переход $S_i \to S_j$.

Все аргументы предикатной функции h могут быть приведены к двум типам данных: логическому и целочисленному. Особенностью представления целочисленных переменных является то, что они задаются в виде интервалов (области определения), путем указания минимального и максимального значения.

Определение 3. Операция инверсии $T = \overline{A}$ для целочисленной переменной A, заданной в интервале $[a_1, a_2]$ и имеющей область определения $Arange = [arange_1, arange_2]$, есть инверсия интервала $T = (T1, T2) = ([tl_1, tl_2][t2_1, t2_2])$, реализуемого с помощью выражений

$$[tl_1, tl_2] = [arange_1, a_1 - 1],$$
$$[t2_1, t2_2] = [a_2 + 1, arange_2]. \quad (9)$$

Примечание. Операция инверсии может приводить к получению двух интервалов значений переменной.

Пример 1. Дано $A = [20,30]$, $Arange = [0,1000]$. Выполнить операцию $T = \overline{A}$.

В соответствии с (9) получим $T = ([0,19],[31,1000])$.

Одним из способов нахождения решений предикатного уравнения является его представление в виде ДНФ или единичного покрытия ($C^1 \subset C$). Но для реализации π -алгоритма кроме единичного требуется еще и нулевое покрытие ($C^0 \subset C$). Рассмотрим процедуру инверсии единичного кубического покрытием целиком, которая соответствует применению правила Де-Моргана для ДНФ.

Определение 4. Операция инверсии над КП, описывающим двухуровневую реализацию функции в виде ДНФ, включает две процедуры:

1. Инверсия каждого куба покрытия: для куба ранга r состоит в замене его k кубами ранга (n-1), где n – мерность (количество координат) куба, k= (n-r) – количество координат куба не равных X. В кубах ранга (n-1) значащие координаты заменяются их дополнением в алфавите кубического исчисления (для двоичного алфавита – инверсией).

2. Попарное пересечение всех кубов из полученных множеств с отбрасыванием пустых результатов пересечений с последующим поглощением одинаковых кубов в соответствии с аналогичным пунктом π -алгоритма [1].

Пример 2. Дана функция от 4-х переменных f(0,1,4,5,6,7,9,13)=1 получить нулевое и единичное покрытие. Выполнить переход от 1-покрытия к 0-покрытию и сравнить полученные результаты.

Минимизация заданной функции с помощью карты Карно с представлением в кубическом виде и в форме булевых уравнений:

X_3X_4 / X_1X_2	00	01	11	10
00	1_{11}	1_{12}	0_{13}	0_{14}
01	1_{21}	1_{22}	1_{23}	1_{24}
11	0_{31}	1_{32}	0_{33}	0_{34}
10	0_{41}	1_{42}	0_{43}	0_{44}

Склеивание по 1: 1-й терм – $1_{11},1_{12},1_{21},1_{22}$, 2-й терм – $1_{21},1_{22},1_{23},1_{24}$, 3-й терм – $1_{12},1_{22},1_{32},1_{42}$. Склеивание по 0: 1-й терм – $0_{33},0_{34},0_{43},0_{44}$, 2-й терм – $0_{31},0_{34},0_{41},0_{44}$, 3-й терм – $0_{13},0_{43},0_{14},0_{44}$.

Результат склеивания в виде КП имеет вид:

$$C^0 = \begin{vmatrix} 1 & X & X & 0 \\ 1 & X & 1 & X \\ X & 0 & 1 & X \end{vmatrix}; \quad C^1 = \begin{vmatrix} 0 & X & 0 & X \\ 0 & 1 & X & X \\ X & X & 0 & 1 \end{vmatrix}.$$

Выполнение перехода от единичного КП к нулевому с использованием правил Де-Моргана для КП:

1. Инверсия каждого куба $C^1 \subset C$:

$$C^1(0X0X) = \begin{vmatrix} 1 & X & X & X \\ X & X & 1 & X \end{vmatrix}; \quad C^1(01XX) = \begin{vmatrix} 1 & X & X & X \\ X & 0 & X & X \end{vmatrix};$$

$$C^1(XX01) = \begin{vmatrix} X & X & 1 & X \\ X & X & X & 0 \end{vmatrix}.$$

2. Попарное пересечение с последующим поглощением:

$$\begin{vmatrix} 1 & X & X & X \\ X & X & 1 & X \end{vmatrix} \cap \begin{vmatrix} 1 & X & X & X \\ X & 0 & X & X \end{vmatrix} = \begin{vmatrix} 1 & X & X & X \\ 1 & 0 & X & X \\ 1 & X & 1 & X \\ X & 0 & 1 & X \end{vmatrix} = \begin{vmatrix} 1 & X & X & X \\ X & 0 & 1 & X \end{vmatrix},$$

$$\begin{vmatrix} 1 & X & X & X \\ X & 0 & 1 & X \end{vmatrix} \cap \begin{vmatrix} X & X & 1 & X \\ X & X & X & 0 \end{vmatrix} = \begin{vmatrix} 1 & X & 1 & X \\ 1 & X & X & 0 \\ X & 0 & 1 & X \\ X & 0 & 1 & 0 \end{vmatrix} = \begin{vmatrix} 1 & X & 1 & X \\ 1 & X & X & 0 \\ X & 0 & 1 & X \end{vmatrix}.$$

Результат совпадает с нулевым покрытием.

Наиболее общей формой представления предикатных уравнений является скобочная. Рассмотрим построение КП по скобочной форме (СФ) предикатного уравнения. Любой условный оператор языка VHDL может быть представлен предикатным уравнением в виде СФ. Определим правила построения КП по СФ предикатного уравнения:

1. СФ представляется в виде дерева синтаксического разбора по операциям OR, AND, NOT. В нижнем ярусе указанного дерева располагаются переменные без инверсий.

2. Для каждой переменной нижнего яруса записывается куб ранга (n-1) с прямым предикатным значением на координате куба, соответствующей данной переменной, n-мерность куба покрытия (для двоичного алфавита – 1).

3. В вершине NOT выполняется правило Де-Моргана для инвертируемого покрытия.

4. В вершине AND реализуется попарное пересечение кубов входящих покрытий с поглощением одинаковых результатов.

5. В вершине OR осуществляется объединение входящих покрытий в одно.

Пример 3. Построить $C^1 \subset C$ для скобочной формы:

$$y = (ab \vee cd)(bd(\overline{a \vee f}) \vee cf). \qquad (10)$$

Разложим указанную функцию в виде дерева синтаксического разбора и на нем покажем построение 1-покрытия для пяти переменных abcdf (рис.2).

Полученное КП эквивалентно ДНФ:

$$y = \bar{a}bd\bar{f} \vee \bar{a}bcf \vee cdf,$$

которая может быть найдена в результате раскрытия скобок в уравнении (10).

5. Процедура выполнения π-алгоритма для предикатной функции

Основой π-алгоритма является итеративная процедура, описанная выражением

$$E^i = \hat{P}_i[\tilde{P}_i(E^{i-1} \bigcap_{i-1}^{p} C^i)], \qquad (11)$$

выполняемая на множестве входных, внутренних и выходных переменных <X, Z, Y>), где p – число примитивов в схемной структуре; $\tilde{P}_i$ – оператор минимизации, предназначенный для склеивания кубов, отличающихся по одной переменной; $\hat{P}_i$ – оператор поглощения, служащий для уменьшения числа векторов за счет исключения избыточных; $E = (E^0, E^1, ..., E^i, ..., E^p)$ – таблицы решений, получаемые после обработки каждого КП по правилам (11); C^i – КП i-го примитива схемной структуры.

Все основные шаги представленного ниже **алгоритма** ориентированы на повышение его быстродействия в целях получения входных решений при заданных состояниях выходных переменных.

1. Поскольку h-функция есть комбинационная схема, то данное обстоятельство позволяет ранжировать предикатные элементы и линии. Сначала отмечаются внешние входы, затем выходы тех элементов, входы которых уже занумерованы. Итерации повторяются до полной отметки всех линий схемы. Аналогично выполняется ранжирование ПЭ. На первом шаге нумеруются примитивы, выходы которых являются внешними, затем элементы, имеющие уже отмеченные преемники. Процедура заканчивается при полной нумерации ПЭ.

2. Определение исходного вектора решений $E^0 = (< X...X >< X...X >< 0...1 >)$ двоичными условиями по выходным переменным (невыходным координатам присваиваются значения X).

3. Исходный вектор решения E^0 пересекается с каждым кубом покрытия элемента $E^0 \cap C_t^i$, выход которого является внешним. В результате получаются решения в виде векторов множества E^1, где индекс при E обозначает номер обработанного примитива.

4. Пересечение множества векторов E^{i-1}, полученного после обработки покрытия C^{i-1}, с кубами очередного ПЭ $E^{i-1} \cap C_t^i$. Пересечение строки решения с кубом покрытия равно пусто, если выполняется условие

$$\exists j(E_j^{i-1} \bigcap_{j=1}^{q} C_{tj}^i = \varnothing). \qquad (12)$$

Если вектор решения E^{i-1} имеет символы X по выходным координатам анализируемого КП C^i, то он (вектор) освобождается от пересечения и переносится в таблицу формируемых кубов E^i.

5. Выполнение операторов минимизации и поглощения на множестве полученных векторов $E_{min}^i = \hat{P}_i[\tilde{P}_i(E^i)]$, что после обработки очередного примитива дает существенное уменьшение числа промежуточных решений для схем со сходящимися разветвлениями и для двухуровневых цифровых структур.

6. После обработки всех примитивов из векторов решения E^p убираются столбцы, соответствующие внутренним переменным, после чего к строкам покрытия E на оставшемся множестве переменных <X,Z> применяется оператор минимизации и поглощения $E_{min}^p = \hat{P}[\tilde{P}(E^p)]$. Входные переменные, определенные на всех кубах символами X, также исключаются из результирующего покрытия как несущественные. Полученная совокупность векторов на множестве входных и выходных существенных переменных E_{min}^p есть искомое кубическое покрытие h-функции, которое в общем случае является минимизированным или тупиковым решением.

Модификация алгоритма по отношению к изложенному в [1] заключается в использовании оператора пересечения в пункте 4 вместо подстановки, а также в выполнении операций минимизации и поглощения после обработки каждого КП. В совокупности такая модификация приводит к уменьшению времени получения КП в несколько раз. Также алгоритм инвариантен по отношению к порядку обработки примитивов, поскольку решение есть пересечение кубических покрытий взаимосвязанных элементов в целях определения общего пространства состояний. Можно считать необязательным ранжирование линий и элементов, но его выполнение ускоряет процесс построения покрытия. Однако с позиции быстродействия включение в алгоритм процедур минимизации и поглощения векторов после обработки очередного ПЭ доминирует над другими попытками улучшения π -алгоритма. Алгоритм инвариантен к количеству выходов комбинационной схемы в целях получения совместного минимизированного КП.

Перед рассмотрением основных шагов π -алгоритма, проанализируем способ получения КП отдельных примитивов по конструкциям языка VHDL.

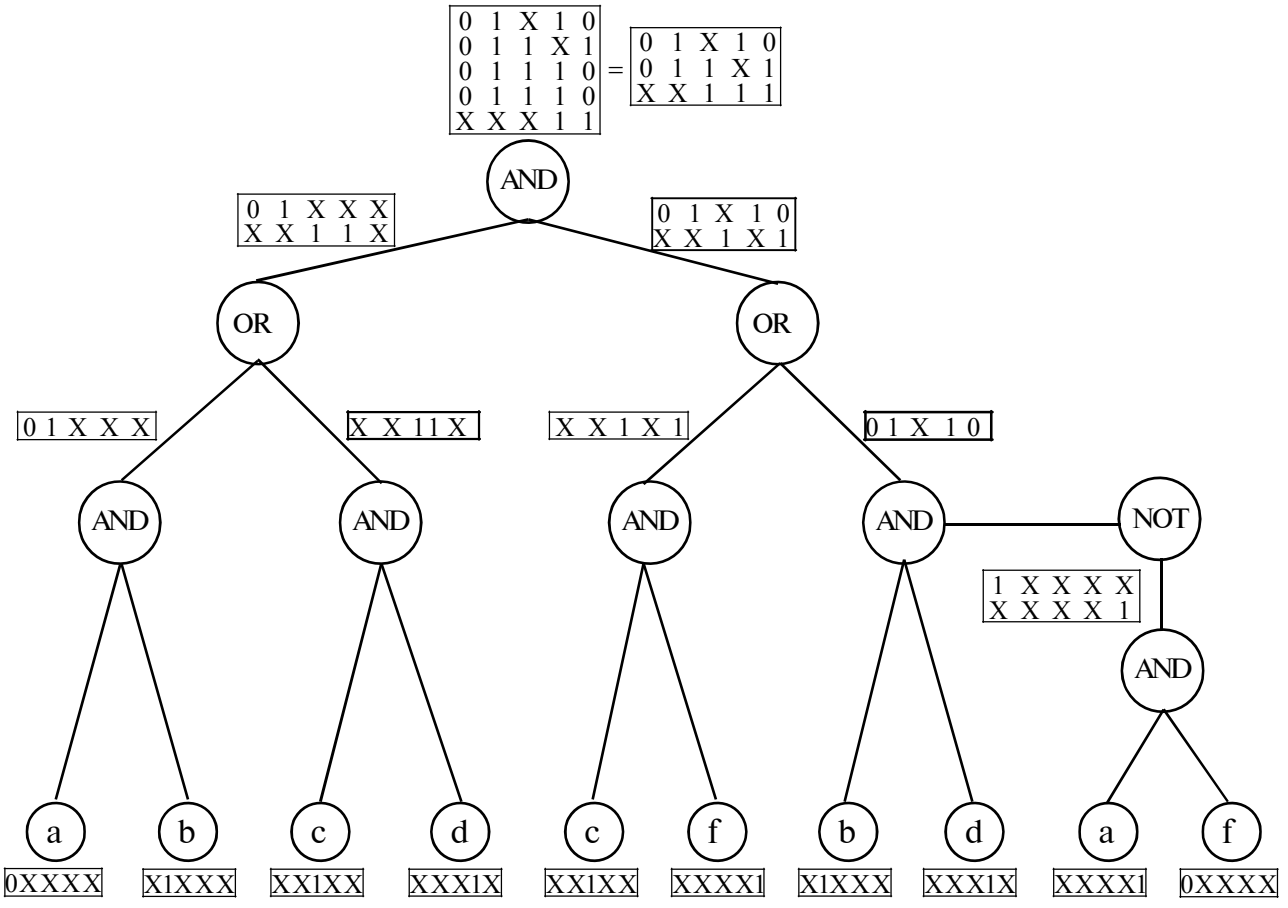

Рис. 2. Дерево синтаксического разбора

Пример 4. Записать в виде функции возбуждения следующее условие перехода: если A больше 20 и C равно переднему фронту (E) или D=1, то выполняется переход $S_1 \to S_2$. Отсутствие переднего фронта обозначается символом F. Значение переменной A целого типа определено в интервале $0 - 2^{16}$.

Указанное условие перехода соответствует фрагменту кода на языке VHDL с оператором Case:

Sreg0<=S1; Case Sreg0 is when S1 =>

if((A>20 and (CLK='1' and CLK'event) or D='1')) then Sreg0<=S2;

Выражение внутри оператора if ( ) соответствует предикатному уравнению функции возбуждения. Заметим, что выражение (CLK='1' and CLK='event'), соответствующее переднему фронту, записывается символом E в алфавите A^2 [1].

Составим дерево синтаксического разбора (рис. 3) указанного функционала в терминах переменных (A_I, C_L, D_B, F) и выполним построение 1-покрытия.

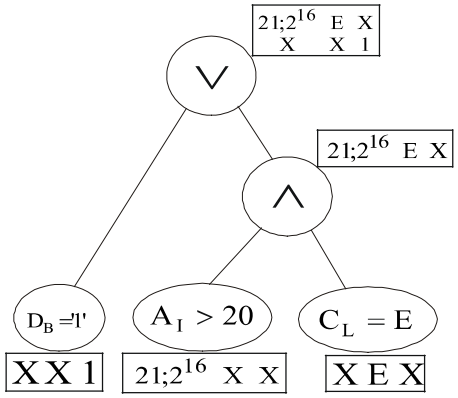

Рис. 3. Дерево разбора для предикатной функции и построение 1-покрытия

В целях получения 0-покрытия к полученному 1-покрытию применяется правило Де-Моргана:

$$\overline{\left|\begin{array}{ccc} 21-2^{16} & E & X \end{array}\right|} = \left|\begin{array}{cc} \overline{21-2^{16}} & \begin{array}{cc} \underline{X} & X \\ E & X \end{array} \\ X & \end{array}\right| = \left|\begin{array}{ccc} 0-20 & X & X \\ X & F & X \end{array}\right|;$$

$$\left|\overline{XX1}\right| = \left|XX0\right|.$$

В результате получается КП:

$$\left|\begin{array}{ccc|c} A_I & C_L & D_B & F \\ \hline 21-2^{16} & E & X & 1 \\ X & X & 1 & 1 \\ 0-20 & X & 1 & 0 \\ X & F & 1 & 0 \\ X & X & 0 & 0 \end{array}\right|.$$

Схема, составленная из подобных примитивов, обрабатывается модифицированным π -алгоритмом, в результате чего находится решение, удовлетворяющее условию перехода. Кроме того, такие примитивы могут составлять многоярусную схему, поскольку выходные значения элементов определены в двоичном алфавите.

Пример 5. Определить входные условия, устанавливающие предикатную схему, представленную на рис. 4, в единичное значение на основании применения модифицированного π -алгоритма [1,5].

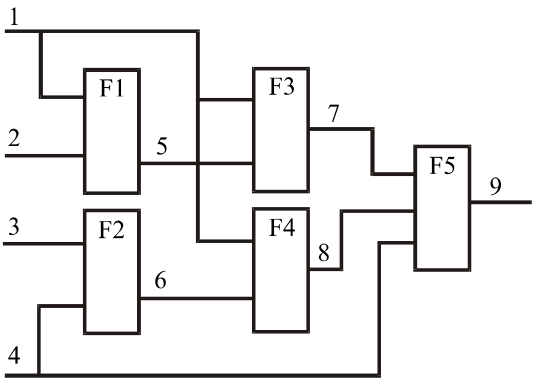

Рис. 4. Предикатная функция возбуждения

Функциональные описания предикатных элементов представлены следующими кубическими покрытиями:

$$C(F1) = \left|\begin{array}{cc|c} 1 & 2 & 5 \\ \hline E & 16-2^{16} & 1 \\ F & X & 0 \\ E & 0-16 & 0 \end{array}\right|; \quad C(F2) = \left|\begin{array}{cc|c} 3 & 4 & 6 \\ \hline 0 & 16-2^{16} & 1 \\ 1 & X & 0 \\ X & 0-16 & 0 \end{array}\right|;$$

$$C(F3) = \left|\begin{array}{cc|c} 1 & 5 & 7 \\ \hline E & 1 & 1 \\ F & X & 1 \end{array}\right|; C(F4) = \left|\begin{array}{cc|c} 1 & 6 & 8 \\ \hline E & 0 & 1 \\ F & 1 & 0 \\ E & 1 & 0 \\ F & 0 & 1 \end{array}\right|; C(F5) = \left|\begin{array}{ccc|c} 7 & 8 & 4 & 9 \\ \hline 0 & 1 & 0-30 & 1 \\ 1 & X & 31-2^{16} & 0 \\ X & 0 & X & 0 \\ 1 & X & 0-30 & 1 \end{array}\right|.$$

Согласно описанному ранее модифицированному π -алгоритму на первом шаге выполняется пересечение КП для предикатных примитивов F5, F4, что дает следующий результат:

$$E^4 = C(F5) \cap C(F4) = \left|\begin{array}{cccc|ccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \hline E & . & . & 0-30 & . & 0 & 0 & 1 & 1 \\ E & . & . & 0-30 & . & 0 & 1 & 1 & 1 \\ F & . & . & 0-30 & . & 0 & 0 & 1 & 1 \\ F & . & . & 0-30 & . & 0 & 1 & 1 & 1 \end{array}\right|.$$

Здесь точками идентифицируются безразличные значения координат, которые в процессе выполнения алгоритма могут быть доопределены любыми типами из множества W_r (8).

Минимизация полученных векторов дает куб

$$E^4_{min} = \left|\begin{array}{cccc|ccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \hline Y & . & . & 0-30 & . & 0 & X & 1 & 1 \end{array}\right|,$$

пересечение которого с покрытием примитива F3 дает следующий результат, не подлежащий минимизации:

$$E^3 = E^4_{min} \cap C(F3) = \left|\begin{array}{cccc|ccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \hline E & . & . & 0-30 & 1 & 0 & 1 & 1 & 1 \\ F & . & . & 0-30 & X & 0 & 1 & 1 & 1 \end{array}\right|.$$

Пересечение этих векторов с покрытием примитива F2 формирует неминимизируемое множество

$$E^2 = E^3 \cap C(F2) = \left|\begin{array}{cccc|ccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \hline E & . & 0 & 16-30 & 1 & 0 & 1 & 1 & 1 \\ F & . & 0 & 16-30 & X & 0 & 1 & 1 & 1 \end{array}\right|.$$

На последнем шаге пересечение E^2 и C(F1) формирует результат

$$E^1 = E^2 \cap C(Fl) = \left|\begin{array}{cccc|ccccc} 1 & & 2 & & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ E & 16-2^{16} & & & 0 & 16-30 & 1 & 0 & 1 & 1 & 1 \\ E & 16-2^{16} & & & 0 & 16-30 & 1 & 0 & 1 & 1 & 1 \end{array}\right|,$$

минимизируемый в один входной вектор

$$E^1_{min} = \left|\begin{array}{cccc|ccccc} 1 & & 2 & & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ E & 16-2^{16} & & & 0 & 16-30 & 1 & 0 & 1 & 1 & 1 \end{array}\right|,$$

который устанавливает предикатную функцию возбуждения в единичное значение.

6. Заключение

В статье представлены структурно-графовые модели цифровых автоматов, ориентированные для проектирования условий перехода при построении тестов цифровых устройств. Предложена модель предикатного примитива, служащего основой при вычислении и анализе комбинационных функций возбуждения. В качестве входного фигурирует алфавит описания шести типов входных переменных, который поддерживает стандарт VHDL. Показан способ получения нулевого и единичного покрытий предикатной функции возбуждения по поведенческому описанию фрагмента цифрового устройства.

Представлен модифицированный π -алгоритм получения входных решений для активизации заданного перехода из одного состояния в другое путем анализа кубических покрытий предикатных примитивов, составляющих комбинационную схему функций возбуждения. Выполнение модифицированного π -алгоритма проиллюстрировано на примере.

Модели и алгоритмы реализованы в виде программного продукта, который предназначен для интегрирования в промышленную систему проектирования Active-HDL [4] в целях построения тестов для верификации цифровых автоматов, описанных на языках описания аппаратуры VHDL, Verilog.

Литература: 1. *Хаханов В.И.* Техническая диагностика элементов и узлов персональных компьютеров. К.: ІЗМН. 1997. 308 с. 2. *IEEE* Standard Language Reference Manual. NewYork: The Institute of Electrical and Electronics Engineers, 1993. 3. *Шабанов-Кушнаренко Ю.П.* Теория интеллекта. Математические средства. Х.: Вища шк. Изд-во при Харьк. ун-те, 1984. 144 с. 4. *Active*-VHDL Series. Book #1 - #4. Reference Guide. ALDEC Inc. 1998. 206 p. 5. *Бондаренко М.Ф., Кривуля Г.Ф., Рябцев В.Г., Фрадков С.А., Хаханов В.И.* Проектирование и диагностика компьютерных систем и сетей. К.: НМЦ ВО. 2000. 306 с.

Поступила в редколлегию 16.06.2000

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

Хаханов Владимир Иванович, д-р техн. наук, профессор кафедры АПВТ ХТУРЭ. Научные интересы: техническая диагностика вычислительных устройств, систем, сетей и программных продуктов. Увлечения: баскетбол, футбол, горные лыжи. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26.

Шкиль Александр Сергеевич, канд. техн. наук, доцент кафедры АПВТ ХТУРЭ. Научные интересы: техническая диагностика вычислительных устройств. Увлечения: теннис. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26.

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