ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

УДК 621.38

П.Н. Бибило, А.Л. Соловьев

ФУНКЦИОНАЛЬНЫЕ VHDL-МОДЕЛИ ЭЛЕМЕНТОВ FPGA СЕМЕЙСТВА SPARTAN 3 ДЛЯ КОНВЕРТАЦИИ ПРОЕКТОВ ЦИФРОВЫХ СИСТЕМ В ЗАКАЗНЫЕ СБИС

На основе анализа структуры микросхемы FPGA xc3s1000-4ft256 и экспериментального изучения функций элементов предлагаются синтезируемые VHDL-модели тех логических элементов, которые входят в конфигурируемые логические блоки FPGA семейства Spartan 3. Необходимость в таких моделях возникает при конвертации реализованных на FPGA проектов цифровых систем в проекты, пригодные для синтеза в библиотеках проектирования заказных СБИС.

Введение

Широко распространенной практикой проектирования цифровых систем является этап их предварительной реализации на FPGA. Такая реализация позволяет быстро оценить сложность и быстродействие проекта и получить экспериментальный образец действующей аппаратуры. Однако для специальных применений требуется использование только отечественной элементной базы. Перевод проекта, реализованного на FPGA, в описание, пригодное для реализации на отечественной элементной базе, является важной научно-технической задачей. Если все спецификации исходного, т. е. реализованного на FPGA, проекта цифровой системы заданы на языке VHDL, то реализация такого проекта на заказной СБИС значительно облегчается [1]. Проблема усложняется, если FPGA-проект должен быть переведен в заказную СБИС из формата топологического редактора (FPGA Editor) системы WebPack ISE (Integrated Software Environment – САПР фирмы Xilinx) [2] автоматизированного проектирования программируемых логических интегральных схем фирмы Xilinx. В данном редакторе после автоматического размещения элементов и трассировки соединений (этап Place & Route) проект может быть подвергнут «ручной» модификации. Внутренними форматами представления проектов на этапе топологического проектирования в системе ISE являются форматы NCD (Native Description Language) и текстовый формат XDL (Xilinx Design Language). В литературе рассматривались задачи использования внутреннего формата XDL описания FPGA-проектов для реализации в составе программируемых микросхем специальных подсхем – макросов. Например, в [3] такая задача рассматривалась для микросхем семейств Virtex II и Virtex II Pro.

Основными блоками FPGA xc3s1000-4ft256 семейства Spartan 3 являются блоки ввода-вывода, конфигурируемые логические блоки (Configurable Logical Block – CLB), блочная память (Block RAM – BRAM) и 18-разрядные умножители (Multiplier) [4]. Процессы перепроектирования макроэлементов ОЗУ и умножителей для их реализации в составе заказных СБИС представляют собой отдельные задачи. Далее рассматривается перепроектирование той части проекта, которая реализуется в виде совокупности сетей CLB. Конвертацию таких проектов FPGA предлагается выполнять, решая две основные задачи конвертации.

1. Задачи конвертации проектов FPGA

Первой задачей конвертации является задача получения структурных описаний проектов, реализованных на FPGA. Чтобы структурные описания были синтезируемыми, необходимо иметь синтезируемые функциональные (поведенческие) описания всех элементов, которые используются в полученных структурных описаниях. Таковыми являются элементы, составляющие конфигурируемые логические блоки. Будем называть эти элементы базисными элементами CLB, а совокупность базисных элементов CLB обозначать *S*.

Второй задачей конвертации, которая и рассматривается в данной статье, является *задача разработки синтезируемых функциональных (поведенческих) VHDL-описаний* элементов, составляющих множество *S*.

Заметим, что интерфейсы VHDL-описаний элементов множества *S* имеются в фирменной документации на FPGA семейства Spartan 3, доступной по сети Интернет [5]. В служебных библиотеках системы проектирования ISE имеются соответствующие VHDL-описания, которые, однако, являются несинтезируемыми и пригодны лишь для целей моделирования проекта после выполнения этапа размещения и трассировки проекта. В этих описаниях, пригодных для моделирования, большое число параметров языка VHDL используется для задания временных характеристик элементов FPGA. Однако при синтезе схем по VHDL-описаниям временные зависимости в исходных описаниях либо игнорируются, либо заменяются другими, определяемыми временными задержками элементов целевой библиотеки синтеза. При конвертации FPGA-проектов важно правильно формировать функции проектов, не принимая во внимание задержки сигналов, поэтому функциональные VHDL-описания элементов из *S* могут быть составлены без указания конкретных временных задержек, т. е. временные задержки элементов из *S* полагаются далее равными нулю.

Рассматриваемая в данной статье задача имеет большое практическое (техническое) значение, так как ее решение, а именно получение синтезируемых функциональных описаний для элементов из *S*, позволяет выполнять синтез исходного проекта в другом базисе логических элементов, входящих в ту или иную целевую библиотеку проектирования заказной СБИС. Для целей синтеза схем заказных СБИС применялся промышленный синтезатор LeonardoSpectrum, который был настроен на пользовательскую библиотеку проектирования, описанную в [1, с. 342]. Множества синтезируемых и несинтезируемых конструкций языка VHDL для этого синтезатора изучались в [6].

2. Конфигурируемый логический блок

Основной структурной единицей FPGA является блок CLB. Он состоит из двух секций левой руки (SLICEM, где M – от слова Memory) и двух секций правой руки (SLICEL, где L – от слова Logic), входы и выходы которых подсоединены к переключательной матрице соединений (Swich Matrix). Через эту матрицу блоки CLB соединены с линиями глобальных соединений. На рис. 1 изображение секции SLICEM получим тогда, когда штриховые линии будут заменены непрерывными. Если же на рис. 1 удалить те элементы и те их соединения, которые изображены штриховыми линиями, то получим изображение секции SLICEL.

Рассмотрим сначала более простую секцию SLICEL (рис. 1 без штриховых линий). Она состоит из двух LUT (Look-Up Table), двух программируемых элементов памяти (FFX, FFY), нескольких логических элементов (И (AND), «сумма по модулю два» (XOR)) и мультиплексоров с одним управляющим входом (MUX2). Связи структурных элементов CLB являются *программируемыми*, их можно оставлять в схеме либо удалять из нее с помощью так называемых программируемых соединений, что позволяет реализовать в одной секции SLICEL разнообразные логические схемы, выполняющие различные функции. Пример программируемого соединения (точки коммутации проводников) можно увидеть в правом верхнем углу на рис. 1 перед выходным полюсом YB. Другие программируемые соединения показаны на входах элементов треугольниками. Программируемый элемент LUT является универсальным логическим элементом и может реализовать любую булеву функцию не более чем от четырех переменных. Элементы памяти могут функционировать как FF (Flip-Flop – триггер, синхронизируемый фронтом) либо как Latch (триггер с потенциальным управлением (защелка)). Таким образом, в секции SLICEL можно реализовать схему с памятью – конечный автомат.

Структура секции SLICEM (рис. 1 со штриховыми линиями) усложнена по сравнению с секцией SLICEL. Усложнение касается реализации LUT. Данные программируемые элементы могут быть настроены на реализацию логических функций – аналогично, как и в секции SLICEL. Дополнительно они могут выполнять функции блока памяти RAM либо функции сдвигового регистра. Два LUT (16-разрядный сдвиговый регистр) могут быть соединены в один 32-разрядный сдвиговый регистр, три – в 48-разрядный и т. д. Для этого в состав SLICEM вводится вспомогательный элемент WSGEN, осуществляющий общую синхронизацию сдвигов, и другие вспомогательные элементы.

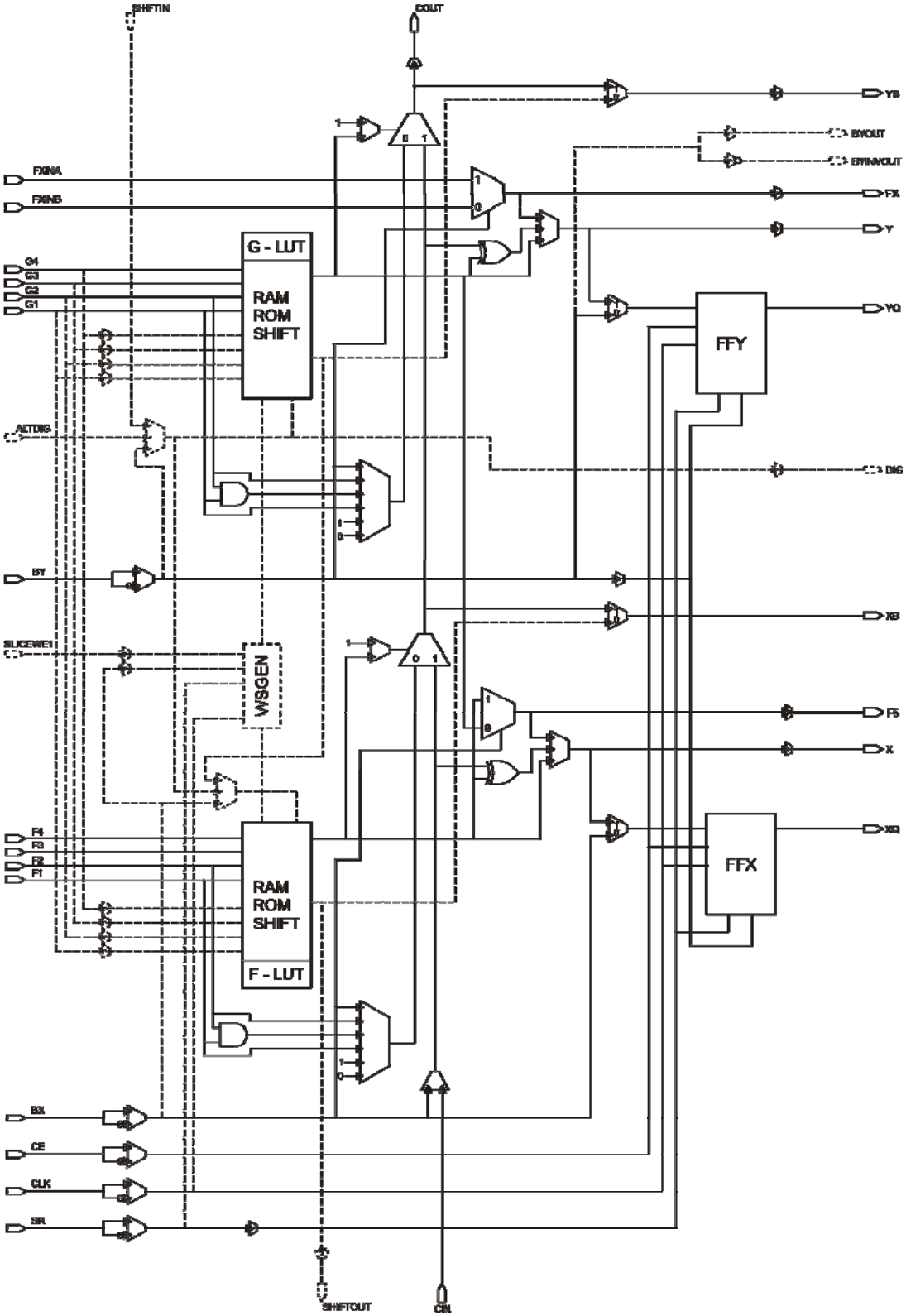

Рис. 1. Структура секций SLICEM и SLICEL

3. VHDL-модели программируемых элементов LUT

В зависимости от значений параметров конфигурирования CLB программируемый элемент LUT может быть использован в качестве:

– комбинационного элемента, реализующего одну полностью определенную булеву функцию, зависящую не более чем от четырех переменных;

– блока памяти (RAM);
– сдвигового регистра заданной длины (не более 16 бит).

3.1. Функциональная модель программируемого элемента LUT, используемого как комбинационный элемент

В структурном описании FPGA-проекта, заданном в формате XDL системы проектирования WebPack ISE 13.1 [2], логические функции программируемого элемента LUT, используемого как комбинационный элемент, описываются с помощью скобочных логических выражений, содержащих только четыре оператора: «@» – сумму по модулю 2, «~» – отрицание, «*» – конъюнкцию и «+» – дизъюнкцию. Так, например, выражение

D=(~A1*(A3@A4))+(A1*A2*A4)

задает логическую функцию

$$D = (\overline{A1}\ \&\ (A3 \oplus A4)) \vee (A1\ \&\ A2\ \&\ A4),$$

которая на VHDL представляется оператором <= назначения сигнала

D <= (not A1 and (A3 xor A4)) or (A1 and A2 and A4).

В VHDL логические операторы имеют следующие обозначения: not (отрицание), and (конъюнкция), or (дизъюнкция), xor (сумма по модулю 2).

Другой формат структурного описания FPGA-проекта, получаемый после выполнения этапа Place & Route проектирования FPGA в системе ISE, использует задание функции, реализуемой элементом LUT, в виде четверки чисел. Такое задание соответствует представлению 16-разрядного булева вектора – столбца значений булевой функции, зависящей от четырех переменных. Например, в последнем столбце табл. 1 записан набор 6CA0 из четырех чисел ($6_{16} = 6_{10}, C_{16} = 12_{10}, A_{16} = 10_{10}, 0_{16} = 0_{10}$). Данный набор кодирует столбец значений булевой функции так, как показано в табл. 1. На языке VHDL табличное представление булевой функции записывается очевидным образом – составляется совершенная ДНФ функции, которая записывается с помощью операторов VHDL (листинг 1).

Таблица 1

ADR0	ADR1	ADR2	ADR3	D	Код (6CA0)
0	0	0	0	0	
0	0	0	1	0	0
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	1	A
0	1	1	0	0	
0	1	1	1	1	
1	0	0	0	0	
1	0	0	1	0	
1	0	1	0	1	C
1	0	1	1	1	
1	1	0	0	0	
1	1	0	1	1	6
1	1	1	0	1	
1	1	1	1	0	

Листинг 1. Функциональная модель программируемого элемента LUT, реализующего логическую функцию D (табл. 1)

```
library IEEE;
use ieee.std_logic_1164.all;
entity LUT4_MM is -- "6CA0"
port (ADR0, ADR1, ADR2, ADR3: in std_logic;
      O : out std_logic);
end;
architecture beh_LUT4_MM is
begin
O <= (ADR0 and (not ADR1) and ADR2 and (not ADR3)) or
     (ADR0 and ADR1 and ADR2 and (not ADR3)) or
     ((not ADR0) and ADR1 and (not ADR2) and ADR3) or
     (ADR0 and ADR1 and (not ADR2) and ADR3) or
     (ADR0 and (not ADR1) and ADR2 and ADR3) or
     ((not ADR0) and ADR1 and ADR2 and ADR3);
end;
```

Если в структурном описании проекта элемент LUT используется как блок памяти (RAM), то функциональной моделью RAM является логическая функция, реализуемая в этом блоке памяти. Функциональная модель блока памяти, реализующего логическую функцию, не отличается от рассмотренной функциональной модели комбинационного элемента, реализующего ту же функцию.

3.2. VHDL-модель программируемого элемента LUT, используемого как сдвиговый регистр

Особенностью использования LUT как сдвигового регистра является возможность варьирования длиной регистра – длина регистра адресуется и может быть от 1 до 16 бит. Напомним, что сдвиговые регистры реализуются в LUT, которые входят в состав только секций SLICEM. Структурная схема сдвигового регистра SRLC16E в виде цепочки триггеров FF и мультиплексора MUX с четырьмя управляющими входами изображена на рис. 2 [7], функциональная синтезируемая VHDL-модель – в листинге 2. Набор <A3, A2,A1, A0> значений сигналов интерпретируется как адрес (число *i* в двоичной системе счисления), задающий длину регистра. Задание значения *i* адреса позволяет направить на выход мультиплексора MUX сигнал с выхода *i*-го триггера FF. Длина регистра задается значением параметра INIT в VHDL-модели сдвигового регистра SRLC16E (листинг 2).

Листинг 2. Функциональная модель LUT, реализующего сдвиговый регистр SRLC16E

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity SRLC16E is
      Generic(INIT:std_logic_vector(15 downto 0):=X"0000");
    Port (     D : in std_logic;
               CE : in std_logic;
              CLK : in std_logic;
               A0 : in std_logic;
               A1 : in std_logic;
               A2 : in std_logic;
               A3 : in std_logic;
                Q : out std_logic;
              Q15 : out std_logic);
end SRLC16E;
architecture Behavioral of SRLC16E is
signal A :   std_logic_vector (3 downto 0);
signal sig : std_logic_vector (15 downto 0) :=INIT;
begin
A <= (A3, A2, A1, A0);
process(CLK)
begin
```

```
if (CLK'event and CLK='1') then
      if (CE='1')then
          sig<=sig(14 downto 0) & D;
      end if;
end if;
end process;
Q15<=sig(15);
Q<=
sig(0) when A=X"0" ELSE
sig(1) when A=X"1" ELSE
sig(2) when A=X"2" ELSE
sig(3) when A=X"3" ELSE
sig(4) when A=X"4" ELSE
sig(5) when A=X"5" ELSE
sig(6) when A=X"6" ELSE
sig(7) when A=X"7" ELSE
sig(8) when A=X"8" ELSE
sig(9) when A=X"9" ELSE
sig(10) when A=X"A" ELSE
sig(11) when A=X"B" ELSE
sig(12) when A=X"C" ELSE
sig(13) when A=X"D" ELSE
sig(14) when A=X"E" ELSE
sig(15) when A=X"F" ;
end Behavioral;
```

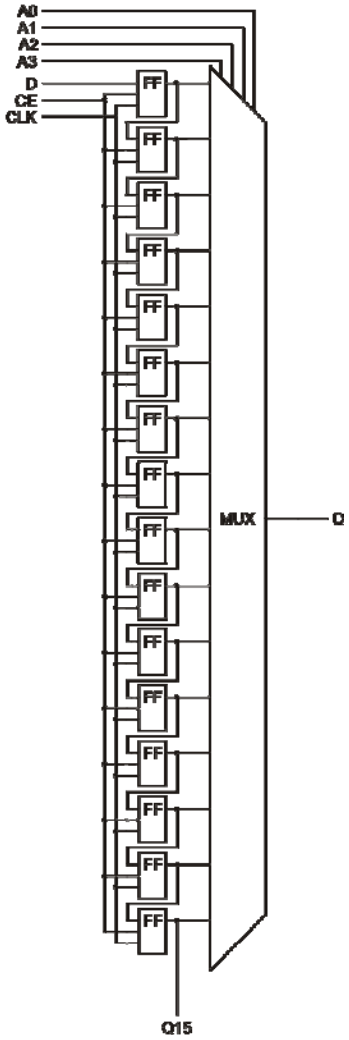

Рис. 2. Сдвиговый регистр адресуемой длины

4. VHDL-модели программируемых элементов памяти

Реализация каждого настраиваемого элемента памяти FFX, FFY (см. рис. 1) в логических блоках SLICEL, SLICEM снабжается четырьмя параметрами конфигурации:

– типом элемента памяти (FF, Latch);

– типом сброса: синхронным (SYNC) либо асинхронным (ASYNC);

– установкой начального значения, значение INIT0 свидетельствует о начальной установке элемента памяти в нуль, INIT1 – в единицу;

– сбросом в единицу (SRHIGH) либо в нуль (SRLOW).

Кроме того, различные реализации элементов памяти кроме обязательных входов D (данные) и CK (в триггерах типа FF – синхросигнал, в триггерах типа Latch – сигнал разрешения) могут иметь дополнительные входы: CE – разрешение, SR – сброс (при значении SRHIGH триггер устанавливается в единицу, при SRLOW – в нуль), REV – установка в единицу (если REV используется, то сброс SR всегда SRLOW). Далее через Q будет обозначаться выход элемента памяти. Функциональная модель триггера FF с полным набором входных сигналов представлена в листинге 3.

Листинг 3. Функциональная модель триггера FF

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity FF is
Port
(D,CE,CK,SR,REV : in std_logic;
             Q  : out std_logic);
end;
architecture Beh of FF is
signal temp : std_logic:='0';  --INIT0
begin
process (SR,REV,CK)
begin
  if (SR='1') then  -- ASYNC
   temp <='0';       -- SRLOW
  elsif (REV='1') then temp <='1';
  elsif (CK'event and CK’last_value =’0’ and CK='1') then
      if (CE='1') then temp<=D;
      end if;
  end if;
end process;
Q<=temp;
end Beh;
```

Функциональная модель триггера, представленная в листинге 4, реализуется в FPGA триггером типа Latch, имеющим асинхронный сброс (ASYNC), с начальной установкой в нуль (INIT0), сбросом в нуль (SRLOW) и полным набором входных сигналов. Заметим, что при синтезе схемы в библиотеке элементов, заданной пользователем, начальные значения сигналов игнорируются, а при синтезе схемы FPGA начальные значения сигналов поддерживаются, и это нужно принимать во внимание при замене схемы FPGA заказной СБИС.

Листинг 4. Функциональная модель триггера типа Latch

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Latch is
Port
(D,CE,CK,SR,REV : in std_logic;
             Q  : out std_logic);
end;
architecture Beh of Latch  is
signal temp : std_logic:=’0’;
```

```
begin
process (SR,REV,CE,CK)
begin
if SR ='1' then temp<='0';
 elsif (REV='1') then temp<='1';
 elsif (CE='1') then
    if (CK='1') then  temp<=D;
    end if;
end if;
end process;
Q<=temp;
end Beh;
```

Если же в структурном описании появляются триггеры, у которых некоторые входы отсутствуют (входы D и CK триггеров являются обязательными), то используется второй подход к формированию VHDL-модели триггера. Он состоит в «уменьшении» числа входов функциональной модели триггера и исключении соответствующих операторов из VHDL-программы. Например, в табл. 2 представлены параметры, а в листинге 5 – «урезанная» модель триггера FF_CE_I0_A_L, у которого отсутствуют входы SR, REV.

Таблица 2

Входные сигналы	Имеются (+), отсутствуют (–)	Настройки	Да (+), нет (–)
D	+	INIT0	+
CK	+	INIT1	–
CE	+	ASYNC	+
SR	–	SYNC	–
REV	–	SRLOW	+
		SRHIGH	–

Листинг 4. Функциональная модель триггера FF_CE_I0_A_L

```
Entity  FF_CE_I0_A_L is
   Port (D : in std_logic;
         CK : in std_logic;
         CE : in std_logic;
          Q : out std_logic );
end FF_CE_I0_A_L;
architecture Behavioral of  FF_CE_I0_A_L  is
signal temp : std_logic:='0';    --INIT0
begin
process (CK)
begin
    if (CK'event and CK’last_value =’0’ and CK='1') then
       if (CE='1') then  temp<=D;
       end if;
    end if;
end process;
Q<=temp;
```

Заметим, что имя FF_CE_I0_A_L триггера формируется по определенным правилам, т. е. в зависимости от типа триггера, используемых входов и значений настроек. Кроме того, при конвертации FPGA-проектов требуются модели триггеров с дополнительным входным сигналом, который осуществляет начальную установку триггера в нуль либо в единицу.

С учетом того что схемные реализации элементов памяти могут различаться не только значениями параметров конфигурации, но и различным числом входов, получается довольно значительное число функциональных моделей триггеров, которые требуются для получения синтезируемых описаний проектов, реализованных в микросхемах FPGA.

5. Верификация моделей элементов CLB

В табл. 3 приведен список основных элементов CLB, требующихся для конвертации рассматриваемого класса проектов. Все построенные функциональные VHDL-модели элементов CLB являются моделями с нулевыми задержками: при моделировании выходные реакции любого из элементов появляются в тот же момент времени, когда изменяется хотя бы один входной сигнал этого элемента.

Таблица 3

Имя элемента	Тип элемента		Функция
	Комбинационный элемент	Элемент памяти	
XOR2	+		Сумма по модулю 2
AND2	+		Конъюнкция
MUX2	+		Мультиплексор
INV	+		Инвертор
BUF, OBUF	+		Передача сигнала
MULT18X18	+		Умножитель
FF		+	Триггер
LATCHE		+	Триггер-защелка
SRLC16E		+	Сдвиговый регистр
RAMD16		+	Блок памяти

Функциональные модели элементов CLB проверены при конвертации 30 FPGA-проектов в синтезируемые VHDL-описания. Исходные описания реализованных на FPGA проектов представляли собой либо алгоритмические VHDL-описания, либо логические схемы в графическом редакторе системы ISE, либо созданные в системе State-CAD [1] графические описания конечных автоматов. Проверка правильности VHDL-моделей элементов CLB осуществлялась на основе сравнения результатов моделирования исходных FPGA-проектов и конвертированных VHDL-описаний проектов. Для тех проектов, которые были представлены в виде VHDL-описаний, функциональная эквивалентность исходного и конвертированного проектов была подтверждена в системе FormalPro формальной верификации фирмы Mentor Graphics [8]. Проверка свойства синтезируемости для конвертированных VHDL-проектов осуществлялась путем синтеза логических схем в целевых библиотеках проектирования заказных СБИС. В качестве основной целевой библиотеки была использована библиотека логических элементов, приведенная в [1, с. 342]. Для всех проектов было достигнуто одинаковое поведение (по результатам окончания тактов моделирования) исходного проекта и конвертированного. Таким образом, конвертация обеспечивает функциональную эквивалентность исходного FPGA-проекта и конвертированного проекта. Однако величины длины такта (тактовая частота) для исходного и реализованного на заказной СБИС конвертированного проектов различаются. Это обусловлено технологической базой: FPGA семейства Spartan 3 реализуются по своей технологии (90 нм SRAM КМОП), а реализация заказной СБИС может быть выполнена по другой технологии, например по отечественной КМОП-технологии и более «крупным» технологическим нормам.

Заключение

Предложенные VHDL-модели элементов конфигурируемых логических блоков микросхем FPGA семейства Spartan 3 обеспечивают поддержку автоматизированного перепроектирования FPGA на заказные СБИС. Получение синтезируемых VHDL-описаний позволяет применять промышленные системы проектирования на базе VHDL – проводить моделирование, сравнивать поведение исходного и конвертированного проектов, синтезировать логические схемы, размещаемые в составе заказных СБИС.

Список литературы

1. Бибило, П.Н. Системы проектирования интегральных схем на основе языка VHDL. StateCAD, ModelSim, LeonardoSpectrum / П.Н. Бибило. – М. : СОЛОН-Пресс, 2005. – 384 с.

2. Зотов, Ю.В. Проектирование цифровых устройств на основе ПЛИС фирмы XILINX в САПР WebPack ISE / Ю.В. Зотов. – М. : Горячая линия – Телеком, 2003. – 624 с.

3. An XDL-based busmacro generator for customizable communication interfaces for dynamically and partially reconfigurable systems / C. Claus [et al.] // Field Programmable Logic and Applications (FPL2010) : 20th Intern. Conf. – Milano, 2010.

4. Кузелин, О.М. Современные семейства ПЛИС фирмы Xilinx : справочное пособие / О.М. Кузелин, Д.А. Кнышев, Ю.В. Зотов. – М. : Горячая линия – Телеком, 2004. – 440 с.

5. Spartan-3 Libraries Guide for HDL Designs [Electronic resource]. – Mode of access : http://www.xilinx.com/itp/xilinx10/books/docs/spartan3_hdl/spartan3_hdl.pdf. – Date of access : 26.03.2012.

6. Бибило, П.Н. О несинтезируемых конструкциях языка VHDL / П.Н. Бибило // Современная электроника. – 2008. – № 5. – С. 68–71.

7. Using Look-Up Tables as Shift Registers (SRL16) in Spartan-3 Generation FPGAs [Electronic resource]. – Mode of access : http://www.xilinx.com/support/documentation/application_notes/xapp465.pdf. – Date of access : 26.03.2012.

8. Лохов, А. Обзор средств функциональной верификации компании Mentor Graphics / А. Лохов // Современная электроника. – 2005. – № 5. – С. 50–54.

Поступила 27.01.12

Объединенный институт
проблем информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: bibilo@newman.bas-net.by

P.N. Bibilo, A.L. Soloviev

FUNCTIONAL VHDL-MODELS OF FPGA-ELEMENTS OF SPARTAN 3 FAMILY FOR CONVERSION OF PROJECTS OF DIGITAL SYSTEMS INTO ASIC

Based on the structure of FPGA xc3s1000-4ft256 and experimental study of element functions, the synthesized VHDL models of FPGA elements of Spartan 3 family are proposed for a conversion of digital systems projects into ASIC. The need for such models appears when converting projects of digital systems implemented in FPGA into the projects suitable for a synthesis of ASIC design.