Функциональные модели триггеров и их реализация в FPGA

Пётр Бибило, Артём Соловьев (г. Минск, Белоруссия)

Рассматриваются алгоритмические VHDL-модели элементов памяти (триггеров) и их реализации в программируемых логических интегральных схемах типа FPGA фирмы Xilinx.

Современные синтезаторы логических схем позволяют преобразовать описания проектов цифровых схем в описания сконфигурированных логических интегральных схем (ПЛИС). При этом допускается смешанный стиль описания: одни части проектов могут быть заданы в графическом виде схемами, другие – в виде алгоритмических описаний на языках VHDL и Verilog. При схемном задании блоков проекта возникает необходимость создания функциональных VHDL-описаний элементов различных логических элементов, как комбинационных, так и элементов памяти в виде триггеров.

Функциональные описания комбинационных элементов представляют различные формы задания логических функций. Описания элементов памяти должны быть представлены синтезируемыми конструкциями, которые после синтеза будут заменены триггерами в построенной логической схеме. При алгоритмических описаниях проекта требуется использовать также синтезируемые конструкции языка VHDL, которые соответствуют триггерам. Для конкретности будем рассматривать ПЛИС типа FPGA семейства Spartan 3 [1], а в качестве языка описания проекта – VHDL [2]. В качестве среды проектирования выбрана система ISE фирмы Xilinx версии 13.1 [3].

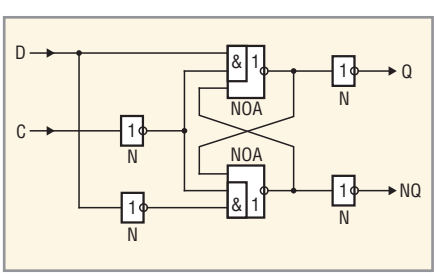


Рис. 1. Структурная схема триггера LAT

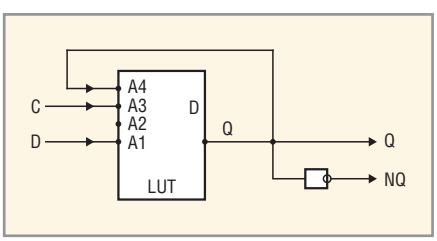


Рис. 2. Реализация триггера LAT в FPGA

В публикациях [1, 4] рассматривались VHDL-модели некоторых триггеров, но не изучались их реализации в FPGA. Данная статья посвящена вопросам реализации триггеров в FPGA, типичным ошибкам при задании триггеров, моделям триггеров и анализу их реализации в FPGA, что может быть полезным при перепроектировании FPGA на заказные СБИС.

Напомним, что основными логическими блоками FPGA являются конфигурируемые логические блоки (CLB), включающие по две секции SliceL и SliceM, каждая из которых содержит два программируемых логических элемента LUT и два синхронных программируемых элемента памяти. Программируемый элемент LUT является универсальным логическим элементом и может реализовать любую булеву функцию не более чем от четырёх переменных.

Элементы памяти могут функционировать как FF (Flip-Flop – синхронизируемый фронтом триггер) либо как Latch (триггер с потенциальным управлением – защёлка). Следует отметить, что в отечественной и зарубежной литературе имеются терминологические различия в классификации триггеров. В данной статье авторы придерживаются терминологии, принятой в зарубежной научно-технической литературе при описании структурных элементов FPGA.

Типичная ошибка проектирования – задание триггера в виде структурного описания схемы из логических элементов

Пользователи САПР программируемых логических интегральных схем, не знакомые с основами схемной реализации VHDL-конструкций, часто пытаются описать триггер в виде схемы из логических элементов или в виде взаимосвязанных логических выражений, что в конечном счёте одно и то же. Например, на рисунке 1 изображена схема триггера LAT. Данный триггер может быть нарисован в графическом редакторе в виде логической схемы либо описан на языке VHDL (см. листинг 1), где: C – вход разрешения, D – вход данных.

Листинг 1. Структурное описание триггера LAT

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity LAT is
port (D, C: IN std_logic;
       Q, NQ: OUT std_logic);
end;
architecture str of LAT is
component NOA
port (A:IN std_logic;
B:IN std_logic;
C:IN std_logic;
       y:OUT std_logic);
end component;
component N
port (A:IN std_logic;
       y:OUT std_logic);
end component;
signal w1, w2, w3, w4 :
std_logic;
begin
i5: N port map (C, w1);
i6: N port map (D, w2);
i1: NOA port map (A =>D, B =>w1,C
=> w4, Y => w3);
i2: NOA port map (A =>w2, B =>
w1, C =>w3, Y => w4);
i8: N port map (w3,Q);
i7: N port map (w4, NQ);
end;
```

Элемент NOA реализует функцию $y = \overline{((A\,\&\,B) \vee C)}$, элемент N является инвертором. После выполнения процедуры синтеза по описанию, представленному в листинге 1, реализацией в FPGA является совсем не элемент памяти, а LUT (см. рис. 2), причём в схеме имеется обратная связь. После завершения синтеза в ISE выдаётся пред-

упреждение о наличии в схеме логического цикла (обратной связи), и при моделировании схема не выполняет функции триггера.

Алгоритмические модели триггеров

Для правильной реализации в FPGA, триггеры должны быть описаны не на структурном уровне (в виде соединений логических элементов, представленных соответствующими операторами port map языка VHDL), а алгоритмическими средствами с помощью параллельных операторов процесса (process) и входящих в процесс последовательных операторов [5]. В этом случае синтезатор логической схемы XST, входящий в состав ISE, распознает и правильно заменит алгоритмическую конструкцию триггером.

Реализацию триггера в логических блоках SliceL, SliceM (см. рис. 3) FPGA можно увидеть в топологическом редакторе (FPGA Editor). Такие реализации снабжаются четырьмя параметрами конфигурации, которые также присутствуют в XDL-формате представления реализованного проекта.

Перечислим параметры конфигурации настраиваемого элемента памяти:

- тип элемента памяти (FF или Latch);
- тип сброса (SYNC или ASYNC);
- установка начального значения (INIT0 свидетельствует о начальной установке элемента памяти в лог. 0, INIT1 – в лог. 1);
- сброс в лог. 1 (SRHIGH) или в лог. 0 (SRLOW).

Кроме того, различные реализации элементов памяти, кроме обязательных входов (D – данные, CK – синхросигнал в триггерах типа FF и сигнал разрешения в триггерах типа Latch), могут иметь дополнительные входы: CE – разрешение; SR – сброс (при значении SRHIGH триггер устанавливается в лог. 1, при SRLOW – в лог. 0); REV – установка в лог. 1 (если REV используется, то сброс SR всегда SRLOW). Далее в статье Q обозначает выход элемента памяти; интерфейсная часть описания триггеров иногда опущена.

Представленная в листинге 2 алгоритмическая модель реализуется в

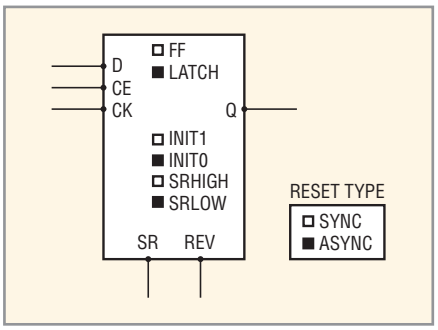


Рис. 3. Программируемый элемент памяти в логических блоках FPGA

программируемом элементе памяти FPGA в виде Latch (с входом разрешения CK) с начальной установкой в лог. 0, сбросом в лог. 0 (четвёртый параметр конфигурации имеет значение SRLOW). Следует отметить, что синтезатор установил данный параметр по умолчанию, а комментарии на VHDL начинаются с двух дефисов и продолжаются до конца строки.

Листинг 2. Алгоритмическая модель триггера LAT (типа Latch)

```
architecture Beh of LAT is
signal temp : STD_LOGIC='0';
--INIT0 (начальная установка)
```

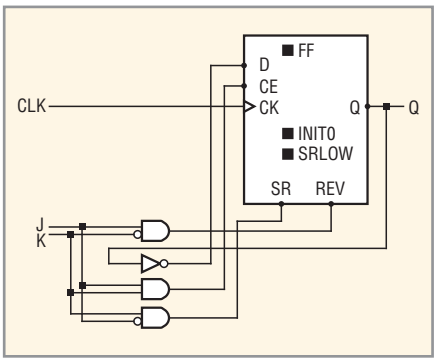


Рис. 4. Реализация JK-триггера в FPGA

```
begin
process (C)
begin
if(C='1') then temp<=D; end if;
end process;
Q <= temp; NQ <= not temp;
end Beh;
```

Алгоритмическая модель триггера, представленная в листинге 3, реализуется в FPGA (см. рис. 3) триггером типа Latch, имеющим асинхронный сброс (ASYNC), с начальной установкой в лог. 0 (INIT0), сбросом в лог. 0 (SRLOW) и полным набором входных сигналов. При синтезе схемы в библиотеке элементов, заданной пользователем, начальные значения сигналов игнорируются, а при синтезе схемы FPGA начальные значения сигналов поддерживаются. Это следует принимать во внимание при замене схемы FPGA на заказную СБИС.

Листинг 3. Алгоритмическая 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;
```

В листинге 4 приводится модель триггера типа FF, управляемого положительным фронтом, с асинхронным сбросом и установкой.

Листинг 4. Алгоритмическая 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='1') then
if (CE='1') then
temp<=D;
end if;
end if;
end process;
Q<=temp;
end Beh;
```

В процессе на листинге 4 для описания асинхронного сброса и установки первыми анализируются значения сигналов SR и REV. При написании модели триггера FF с синхронным сбросом, выражение if (CK'event and CK='1') для проверки переднего фронта синхросигнала CK должно быть проанализировано первым в процессе, затем следует проверить значения сигналов сброса и установки.

В алгоритмических моделях триггеров выражения

```
if (CK'event and CK='1')

  и

if (CK'event and CK='0')
```

определяют триггер, синхронизируемый передним и задним фронтом соответственно.

Реализация JK-триггера в FPGA

Сложные триггеры реализуются в FPGA программируемыми элементами памяти с дополнительной комбинационной логикой на основе LUT и других элементов из состава логических блоков.

Листинг 5. Алгоритмическая модель JK-триггера

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity JK is
Port ( J : in STD_LOGIC;
K : in STD_LOGIC;
CLK : in STD_LOGIC;
Q : out STD_LOGIC);
end JK;
architecture Beh of JK is
begin
process (CLK)
variable       temp :
STD_LOGIC:='0';
begin
if(CLK'event and CLK='1') then
if (K='1' and J='0') then
temp:='0';
elsif (K='0' and J='1') then
temp:='1';
elsif (K='1' and J='1') then
temp:=not temp;
end if;
end if;
Q<=temp;
end process;
end Beh;
```

Приведённая в листинге 5 модель реализуется схемой (см. рис. 4), которая размещается в FPGA в трёх блоках SliceL (синтез производился в Xilinx ISE 13.1 с запрещением размещения триггеров блоках ввода-вывода (IOB)); в одном из блоков SliceL задействован только элемент памяти FF, а комбинационные логические элементы реализованы в двух других блоках SliceL.

Реализация RS-триггера в FPGA

Если не оговаривать значения выходов RS-триггера (рассматривается RS-триггер с двумя выходами – прямым Q и инверсным NQ) для входной комбинации $R = 1$, $S = 1$, то алгоритмическая модель RS-триггера (см. листинг 6) реализуется в FPGA одним элементом – триггером Latch. Если же для входной комбинации $R = 1$, $S = 1$ полагать значения прямого и инверсного выходов равными лог. 0, то для реализации RS-триггера требуется два элемента памяти. В обоих случаях реализация таких триггеров невозможна без дополнительной комбинационной логики.

Листинг 6. Алгоритмическая модель RS-триггера

```
process (R,S)
variable temp: std_logic:='0';
begin
 if (R='1') then temp:='0';
        elsif (S='1') then
temp:='1';
        end if;
        Q <= temp;
        NQ <= not temp;
end process;
```

Заключение

С учётом того, что схемные реализации элементов памяти могут отличаться не только значениями параметров конфигурации, но и различным числом входов, для восстановления структурно-функциональных описаний проектов, реализованных в FPGA, необходимо использовать большой набор базовых функциональных моделей триггеров.

При помощи различных конструкций язык VHDL позволяет описывать разнообразные аспекты поведения цифровых систем. Многие конструкции VHDL при схемной реализации приводят к элементам памяти. Не только приведённые выше описания триггеров реализуются в FPGA настраиваемыми элементами памяти, – к корректным реализациям приводят и описания на основе других VHDL-операторов. При этом следует учитывать особенности реализации VHDL-конструкций различными синтезаторами, которые используют разные целевые библиотеки. Однако общепринятыми в литературе [2, 5] VHDL-моделями описания триггеров являются алгоритмические модели, использующие операторы процесса и операторы выделения передних и задних фронтов сигнала синхронизации с помощью различных атрибутов данного сигнала.

Литература

1. *Кузелин О.М., Кнышев Д.А., Зотов Ю.В.* Современные семейства ПЛИС фирмы Xilinx: Справочное пособие. Горячая линия – Телеком, 2004.
2. *Бабак В.П., Корченко А.Г., Тимошенко Н.П., Филоненко С.Ф.* VHDL: Справочное пособие по основам языка. Додэка-XXI, 2008.
3. *Зотов Ю.В.* Проектирование цифровых устройств на основе ПЛИС фирмы Xilinx в САПР WebPack ISE. Горячая линия – Телеком, 2003.
4. *Суворова Е.А., Шейнин Ю.Е.* Проектирование цифровых систем на VHDL. БХВ-Петербург, 2003.
5. *Pedroni V.A.* Circuit Design with VHDL. MIT Press: London, 2004.

Новости мира News of the World

Ignis Innovation разработала AMOLED-дисплей с разрешением 300 точек на дюйм

Компания Ignis Innovation объявила о том, что ею разработан AMOLED-дисплей с разрешением более 300 точек на дюйм и матрицей RGB. Ранее столь высокая плотность изображения для экранов AMOLED была доступна лишь с использованием матрицы типа PenTile, которая неоднократно подвергалась критике.

Новая технология производства дисплеев включает в себя новые элементы управления, которые позволяют компенсировать неравномерности изображения и существенно повысить срок службы экрана. Согласно Ignis, такая технология позволяет выпускать дисплеи, аналогичные по плотности существующим Retina Display, используемым в iPhone 4 и 4S, но значительно превосходящим их по качеству отображения цветов, контрастности и углам обзора.

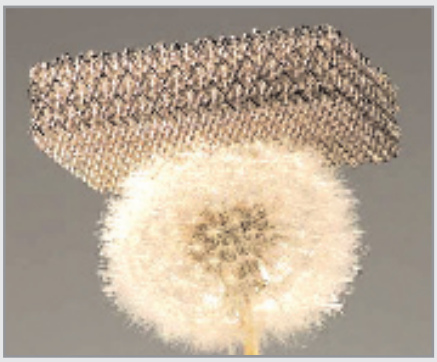
Компания уже объявила о том, что не планирует самостоятельно производить дисплеи по собственной технологии. Тем не менее, пока неизвестно, какие производители займутся выпуском подобных экранов. Вполне вероятно, что среди них будет Samsung, которая является мировым лидером в области технологии AMOLED.

http://gsmarena.com/

Создан самый лёгкий в мире материал

Учёные из исследовательской лаборатории HRL Калифорнийского университета создали материал с плотностью всего 0,9 мг/см^3. Сверхлёгкая и упругая губка из металлических волокон в 100 раз легче пенополистирола и считается самой лёгкой структурой из металла. Новый материал заставляет пересмотреть пределы прочности для пористых материалов из-за своей уникальной микрорешёточной архитектуры. Состоящий на 99,99% из воздуха материал изготовлен из полых металлических трубок нанометрового, микронного и миллиметрового размера с толщиной стенок в 1000 раз тоньше человеческого волоса.

Из специального фотополимера создаётся необходимая структура, которая покрывается никель-фосфорным сплавом. Затем полимер удаляется растворителем. По словам создателей, в создании структуры используется принцип архитектурных металлоконструкций по примеру Эйфелевой башни или моста «Золотые ворота». Благодаря этому материал демонстрирует уникальные свойства упругости и сопротивления сжатию. Исследователи считают, что созданный материал будет широко использоваться в различных областях от аэрокосмической промышленности до строительства.



http://www.hrl.com/

Самые эффективные в мире органические светодиоды на пластиковой подложке

Как утверждает группа учёных из университета Торонто, им удалось разработать самые эффективные в мире органические светодиоды (OLED), разместив их на пластиковой подложке. Это означает, что у производителей появилась альтернатива укреплённому стеклу, которое считается сегодня единственно возможным выбором. Более того, пластик позволит делать дисплеи гибкими, а новая технология обещает сделать их при этом недорогими.

В настоящий момент большинство OLED-панелей производится на укреплённом тяжёлыми металлами стекле, которое обеспечивает высокую яркость при низком потреблении энергии. Единственная проблема на сегодняшний день – это цена, такие панели имеют высокую себестоимость; кроме того, они достаточно тяжёлые, негибкие и, как результат, хрупкие.

Разработанная канадцами технология имеет в своей основе слой оксида тантала толщиной 50…100 нм, он по сути является продвинутым оптическим тонкоплёночным покрытием. Если нанести его на гибкую пластиковую подложку, можно получить высокоэффективную OLED-панель, которая может существовать без традиционного укреплённого стекла. К сожалению, на текущий момент отсутствует технология, позволяющая перенести данное решение на рельсы массового производства, эту цель разработчики достигнут в ближайшие годы.

http://www.ubergizmo.com/