Краткий курс HDL. Часть 6. Написание кода, зависимого от аппаратной платформы

Иосиф КАРШЕНБОЙМ
iosifk@narod.ru

Производители микросхем поставляют библиотеки компонентов, которые размещены на кристалле. Такие библиотеки могут содержать примитивы в виде отдельных транзисторов, триггеров, регистров, блоков памяти. Библиотеки для FPGA обычно «начинаются» не с транзисторов, а с буферов, триггеров и т. д.

Если мы хотим применить библиотечную функцию, поставляемую производителем микросхем, то необходимо либо воспользоваться визардом, находящимся в составе программного инструмента фирмы — изготовителя микросхем, либо найти файл описаний библиотечных функций и примитивов. Для микросхем FPGA компании Xilinx такой файл описания находится в разделе: C:\Xilinx\verilog\src\unisims\.

В данном разделе для нас наибольший интерес представляют блоки памяти, поскольку при описании памяти есть некоторая проблема, которую мы проиллюстрируем на примере использования памяти RAMB16_S36 в микросхемах Virtex2. Подробное описание данной функции читатель сможет найти в предыдущих частях статьи. Необходимо отметить следующее. У компиляторов и симуляторов разных производителей программной продукции могут отличаться синтаксисы описания узлов. В таком случае используются команды: **synthesis translate_off** и **synthesis translate_on**. С помощью этих команд производится отключение, а затем включение компилятора для синтеза.

В примере 1 приведен фрагмент кода, представляющий собой часть памяти команд для блока RAMB16_S36 для микросхемы FPGA Virtex2. В этом примере память инициализируется при помощи **defparam**. Но для работы синтезатора такое описание инициализации памяти не работает. Синтезатор работает с такой инициализацией: //synthesis attribute INIT_XX of ROM.

В этом выражении XX — это номер строки, каждая строка содержит 256 шестнадцатеричных цифр. В примере 2 приведен код, описывающий работу с двумя блоками памяти 16_S18 для микросхемы FPGA Virtex2. Если же мы хотим увеличить объем памяти, то необходимо будет использовать несколько блоков памяти. Эти блоки должны быть включены так, чтобы каждый из них выдавал часть битов данных в общем поле данных. Это позволит не делать мультиплексор по выходным данным и значительно сократит ресурсы, задействованные для размещения проекта на кристалле.

Для того чтобы облегчить процесс перехода от описания фрагмента памяти программ для встроенного микроконтроллера, приведенного в примере 1, к описанию, приведенному в примере 3, автор применял «самодельный» программный инструмент [1]. Во-первых, программа производила ассемблирование текста и формировала строковую переменную, содержащую коды, необходимые для инициализации памяти (пример 1). Во-вторых, программа сканировала файл описания модуля на языке Verilog, в данном случае это был модуль микроконтроллера, и находила следующую строку: “// ROM starts here”. Программа удаляла из исходного файла текст от строки “// ROM starts here” до строки “// ROM stops here”. После строки “// ROM starts here” программа вставляла шаблон текста, содержащий описание самой памяти, и формировала весь остальной текст, находящийся после строки “// ROM stops here” (пример 3). Кроме информации, необходимой для инициализации памяти, программа также вставляла информацию, необходимую для сопровождения проекта: имя и путь к файлу исходного текста, а также дату и время компиляции.

```
0000  00000000  Label0  :  NOP   //  Comment
0001  560055AA          :  LDI  Dest-acc : Lit-0x55AA :  //
0002  5600CC33          :  LDI  Dest-acc : Lit-0xCC33 :  //
0003  00000000          :  NOP  //
0004  80000000          :  JMP  Addr-0 :  //
0005  5600FEAB          :  LDI  Dest-acc : Lit-0xFEAB :  //
0006  5600FEAB          :  LDI  Dest-acc : Lit-0xFEAB :  //
0007  5600FEAB          :  LDI  Dest-acc : Lit-0xFEAB :  //
0008  5600FEAB          :  LDI  Dest-acc : Lit-0xFEAB :  //

defparam ROM.INIT_00 = 256'h5600FEAB5600FEAB5600FEAB800000000000000005600CC335600055AA00000000;
defparam ROM.INIT_01 = 256'h0000000000000000000000000000000000000000000000000000000005600FEAB;
```

Пример 1. Часть памяти команд для блока RAMB16_S36

```
//-----------------------------------------------
// Program space Memory INIT
//-----------------------------------------------

// ROM starts here
//defparam ROM.INIT = 36'h0;
//defparam ROM.SRVAL = 36'h0;

RAMB16_S36 ROM
      // program space
      (  .WE (1'b0),
         .EN (ram_rd_ena),
         .SSR (1'b0),            // signal to command memory,
         .CLK (clk),             // signal to memory,
         .ADDR (ps_cnt),         // 8:0
         .DI (32'h00),           // 31:0
         .DIP (4'b0000),         // 3:0
         .DOP (),                // 1:0
         .DO (ps_data)           // 31:0
      );

//     File Name — D:\PMU\Controller\Cmd2.pmu
//     Date, Time — 30.05.04 11:49:41

//synthesis attribute INIT_00 of ROM is “A8090000460404004C10000880000000000000008005000098000 00B00000000”
//synthesis attribute INIT_01 of ROM is “8191000080570000803800008 01D00008191000080160000819100000000000”
//synthesis attribute INIT_02 of ROM is “4400040300000000816D00008110000081050000810200008 0F8000080EE0000”
…
…
//synthesis attribute INIT_33 of ROM is “0000000000000000000000000000000000000000000000000000819600000000000”

//synthesis translate_off

defparam ROM.INIT = 36'h0;
defparam ROM.SRVAL = 36'h0;
defparam ROM.WRITE_MODE = “WRITE_FIRST”;
defparam ROM.INIT_00 = 256'hA8090000460404004C10000880000000000000008005000098000 00B00000000;
defparam ROM.INIT_01 = 256'h8191000080570000803800008 01D00008191000080160000819100000000000;
```

```
defparam ROM.INIT_02 = 256'h4400040300000000816D0000811000008105000081020000 80F8000080EE0000;
…
…
defparam ROM.INIT_33 = 256'h0000000000000000000000000000000000000000000000000000819600000000000;

//synthesis translate_on
// ROM stops here
```

Пример 2. Код, описывающий работу с памятью RAMB16_S36 для микросхемы FPGA Virtex2

```
//-----------------------------------------------
// Program space Memory INIT
//-----------------------------------------------

// ROM starts here
RAMB16_S18 ROMA
      // program space
      (  .WE (1'b0),
         .EN (ram_rd_ena),
         .SSR (1'b0),                  // signal to command memory,
         .CLK (clk),                   // signal to memory,
         .ADDR (ps_cnt[9:0]),
         .DI (16'h00),
         .DIP (2'b00),
         .DOP (),
         .DO (ps_data[31:16]) );

RAMB16_S18 ROMB
      // program space
      (  .WE (1'b0),
         .EN (ram_rd_ena),
         .SSR (1'b0),                  // signal to command memory,
          .CLK (clk),                  // signal to memory,
         .ADDR (ps_cnt[9:0]),
         .DI (16'h00),
         .DIP (2'b00),
         .DOP (),
         .DO (ps_data[15:0]) );

//     File Name — D:\FPGAExpr_Prj\PMU_REV30X_VD\PMU_REV30X_VD\Cmd_rev3ax_106.pmu
//     Date, Time — 12.10.2004 11:17:56
//     1K word 2 x 1024 x 18
//     A — Hi word
//     B — Low word
//

//synthesis attribute INIT_00 of ROMA is "0000000000000000000000000000000000000000000000000000000000000000"
//synthesis attribute INIT_00 of ROMB is "0000000000000000000000000000000000000000000000000000000000000000"

//synthesis translate_off

defparam ROMA.INIT = 18'h0;
defparam ROMA.SRVAL = 18'h0;
defparam ROMA.WRITE_MODE = "WRITE_FIRST";

defparam ROMB.INIT = 18'h0;
defparam ROMB.SRVAL = 18'h0;
defparam ROMB.WRITE_MODE = "WRITE_FIRST";

defparam ROMA.INIT_00 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ROMB.INIT_00 = 256'h0000000000000000000000000000000000000000000000000000000000000000;

//synthesis translate_on
// ROM stops here
```

Пример 3. Код, описывающий работу для двух блоков памяти 16_S18 в микросхемах FPGA Virtex2

По вопросам составления описаний на блоки необходимо руководствоваться документами, которые предоставляют фирмы-производители [2–5].

В следующем разделе мы рассмотрим вопросы, связанные с отладкой проекта.

Литература

1. Каршенбойм И. Г. Микропроцессор своими руками. Часть 3. Ассемблер и софт-симулятор // Компоненты и технологии. 2006. № 3, 4.
2. Chapman K. Multiplexer Selection. WP274 (v1.0). February 4, 2008. www.xilinx.com
3. Garrault P., Philofsky B. HDL Coding Practices to Accelerate Design Performance. WP231 (1.1). January 6, 2006. www.xilinx.com
4. Hill T. Using MATLAB to Create IP for System Generator for DSP. WP241 (v1.0) April 19, 2006. www.xilinx.com
5. Chapman K. Get your Priorities Right — Make your Design Up to 50% Smaller. WP275 (v1.0.1) October 22, 2007. www.xilinx.com