|  |
| --- |
| **實驗二** |
| *Time-of-day clock* |

|  |
| --- |
| 學生: 謝旻錡  學號: 313512078  日期: 2025/03/12 |

1. **實驗目的**

熟悉latches、flip-flop及regist，並以此基礎組合出counter，並藉由counter的功能，實作一個循環24小時制的時鐘。

時鐘輸入輸出規格:

|  |  |  |  |
| --- | --- | --- | --- |
| Input | | Output | |
|  | Select hour and minute | HEX5~4 | Hour |
|  | Set time (positive edge) | HEX3~2 | Minute |
|  | The value of setting time | HEX1~0 | Second |

1. **實驗程式碼**

STEP.1 我使用QUARTER的LPM (程式碼至於附件)，定義了second/ minute/ hour的counters，並配合frequency divider計算second/ minute/ hour時間。

STEP.2 計算出的值為二進制，所以需要使用BCD來轉換成10進制，這邊的BCD我使用Double Dabble演算法來轉換。

STEP.3 最後接上七段顯示器顯示時間。

* **LMP\_counters**

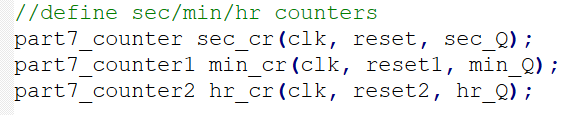


圖 1. counters

|  |  |
| --- | --- |
| 變數 | 內容 |
| clk | 時鐘訊號 (50MHz) |
| reset, reset1, reset2 | counter歸零訊號 (1: rest; 0: nothing) |
| sec\_Q, min\_Q, Hr\_Q | 記數數量 |

表格 1. Counter變數

我定義了三個counter，分別用於second, minute, hour 的clk訊號的記數。每20ns記數一次(clk: 50MHz)。

* **時間設定功能**

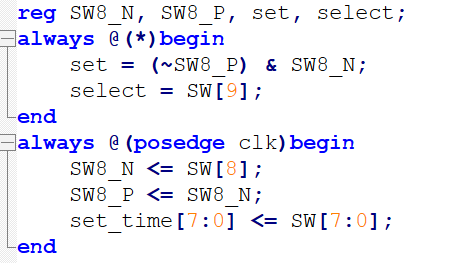


圖 2. 設定時間

|  |  |
| --- | --- |
| 變數 | 內容 |
| select | 選擇設定hour或minute (1: hour; 0: minute) |
| set | 傳入設定時間 (positive edge) |
| set\_time | 欲設定的時間 |
| SW8\_N, SW8\_P | SW8開關的現在值與上一個clk的開關值 |

表格 2. Time setting變數

宣告一個8bit的set\_time 暫存器，儲存SW[7:0]開關欲設定的時間。

為了使 set是以正緣觸發的方式設定時間，我寫段邏輯電路 。因此當SW8為0轉至1的瞬間，set的值為1，以此滿足正緣觸發的條件。

* **計時器 second/ minute/ hour timer**

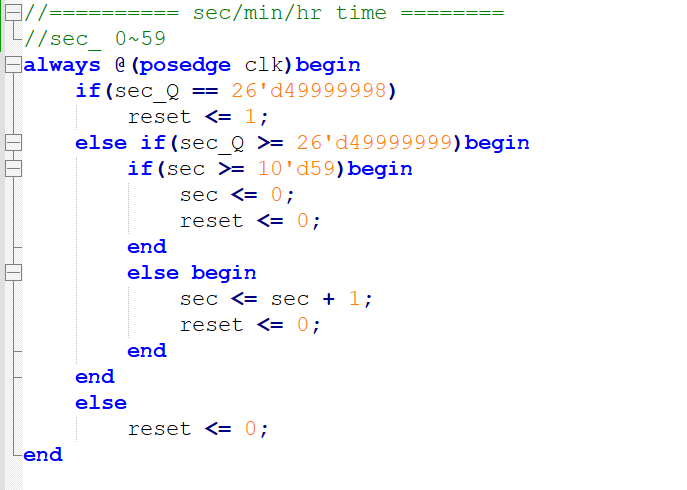


圖 3. Second timer

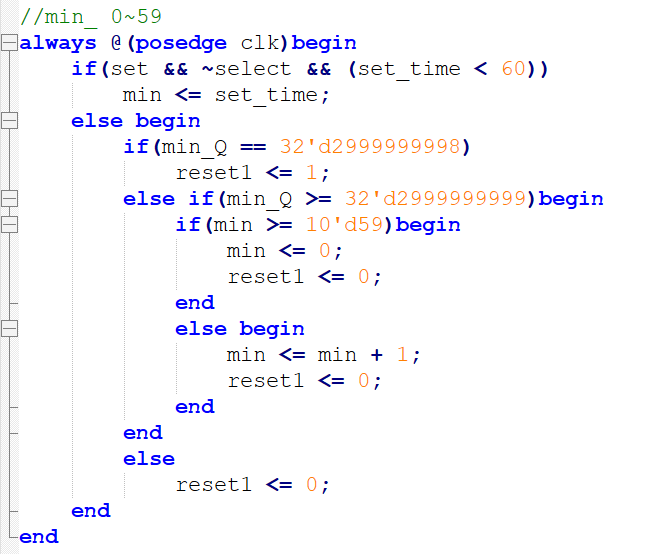


圖 4. Minute timer



圖 5. Hour timer

|  |  |
| --- | --- |
| 變數 | 內容 |
| clk | 時鐘訊號 (50MHz) |
| reset, reset1, reset2 | counter歸零訊號 (1: rest; 0: nothing) |
| sec\_Q, min\_Q, hr\_Q | 記數數量 |
| sec, min, hr | 計時數值 |
| set\_time | 欲設定的時間 |
| set | 傳入設定時間 |
| select | 選擇設定hour或minute (1: hour; 0: minute) |

表格 3. Timer變數

分別設定三個計時器來對second/ minute/ hour計時。

Second: 每當counter數了次，為一秒。如果當sec數了59秒時，sec下一秒歸零。

Minute: 每當counter數了次，為一分鐘。當min數了59秒，sec數了59秒時，下一秒min歸零。

Hour: 每當counter 數了次，為一小時。當hr數了23小時，min數了59秒，sec數了59秒時，下一秒hr歸零。

* **BCD**

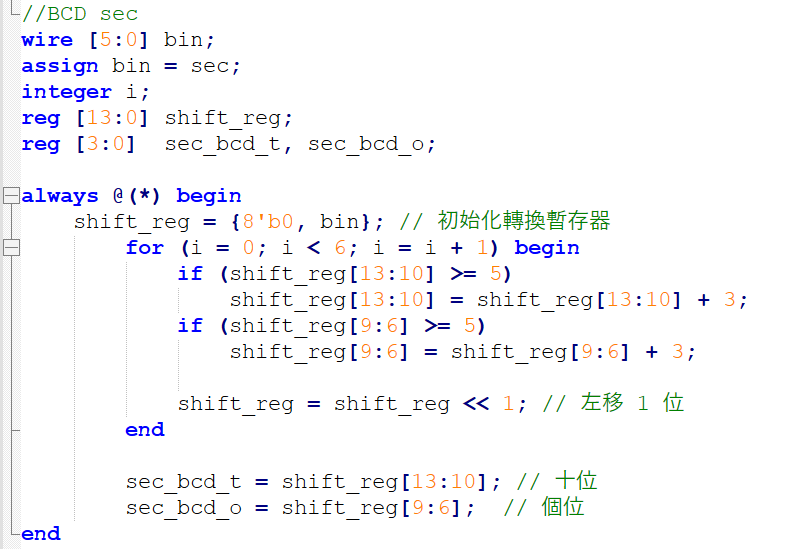


圖 6. BCD of second

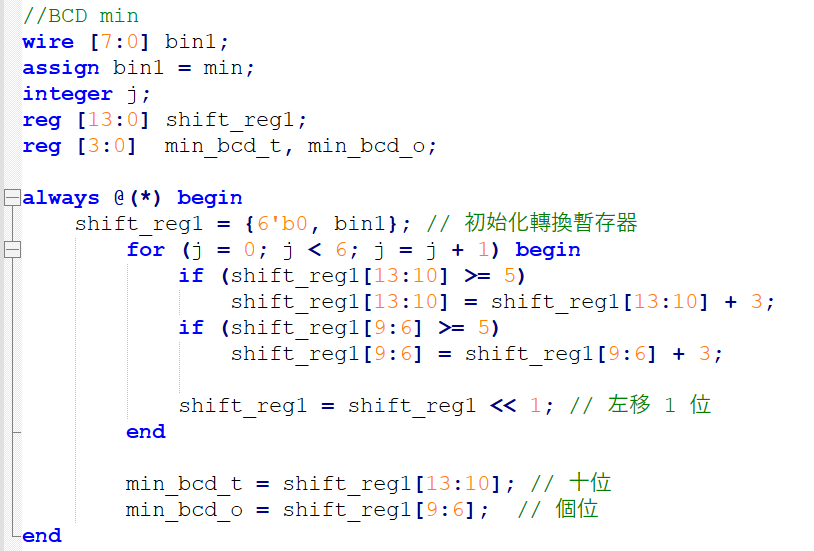


圖 7. BCD of minute

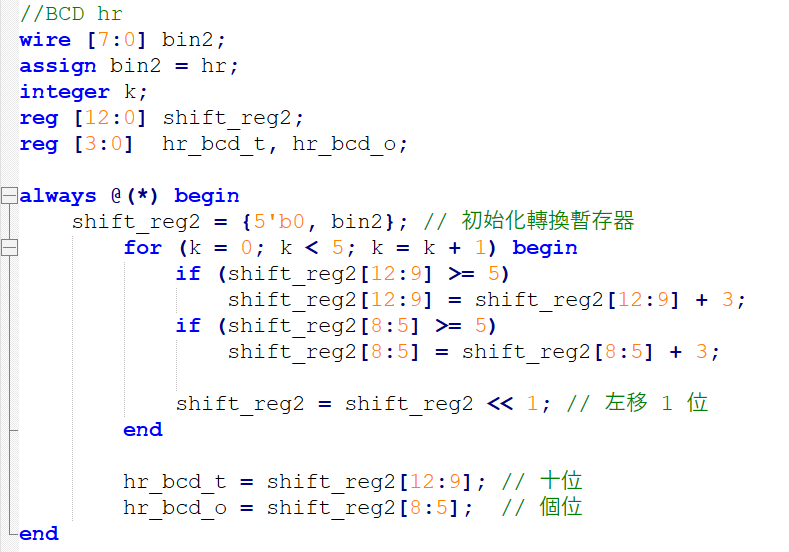


圖 8. BCD of hour

|  |  |
| --- | --- |
| 變數 | 內容 |
| bin, bin1, bin2 | 儲存sec/ min/ hr的暫存器 |
| shift\_reg | 用於sec的BCD計算的暫存空間 |
| shift\_reg1 | 用於min的BCD計算的暫存空間 |
| shift\_reg2 | 用於hr的BCD計算的暫存空間 |
| sec\_bcd\_t, sec\_bcd\_o | 秒 轉換後的10進制值 |
| min\_bcd\_t, min\_bcd\_o | 分 轉換後的10進制值 |
| hr\_bcd\_t, hr\_bcd\_o | 時 轉換後的10進制值 |

sec/ min/ hr裡儲存的時間為二進制形式，因此我藉由使用double dabble的演算法，將其轉換為二進制。

Double dabble演算法:

**遞推關係**

* 初始條件（當 i=0時）：

BCD 欄位初始化為 0。

* **主要轉換步驟（第 i 次迭代）：**

然後整體左移一位：

這個過程會持續 n 次，最終 D(n)即為 BCD 的結果。

* 7段顯示器

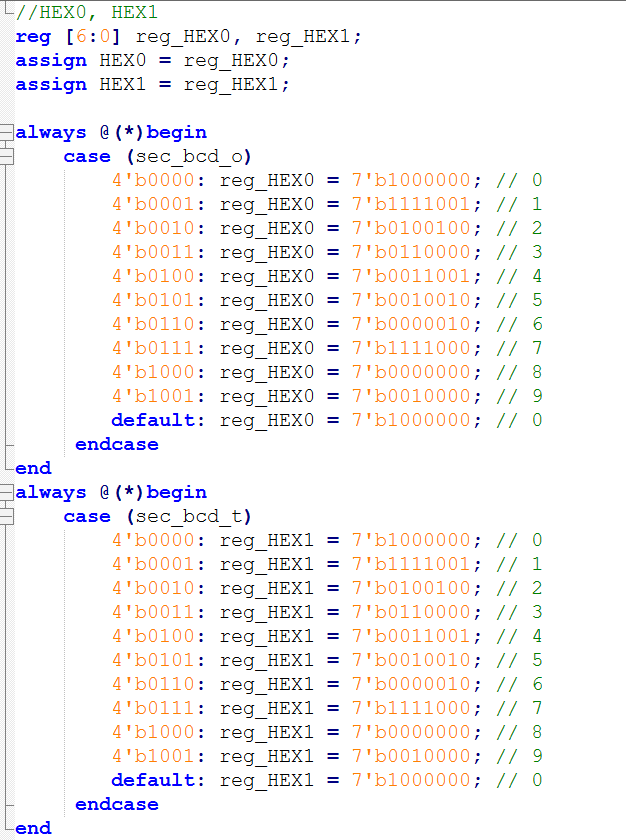


圖 9. 7段顯示器-秒

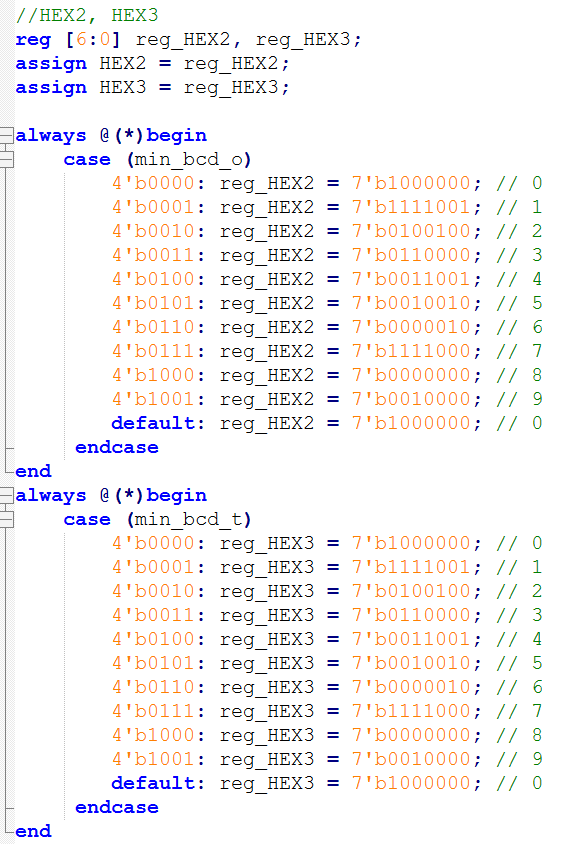


圖 10. 段顯示器-分

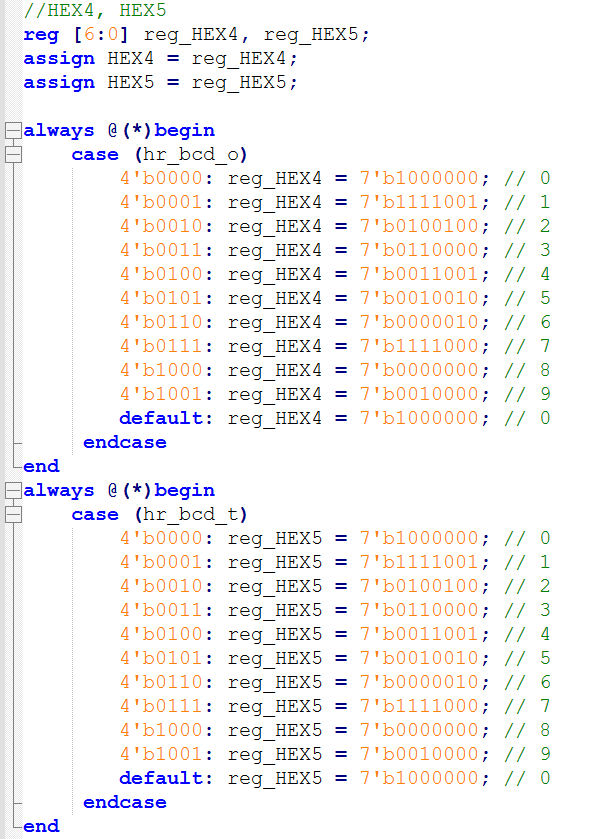


圖 11. 段顯示器-時

|  |  |
| --- | --- |
| 變數 | 內容 |
| HEX0, HEX1 | 秒的10進制顯示 |
| HEX2, HEX3 | 分的10進制顯示 |
| HEX4, HEX5 | 時的10進制顯示 |

1. **實驗結果照片(optional)**

|  |  |
| --- | --- |
|  |  |
|  |  |

1. **RTL布局(optional)**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

1. **問題與討論**
   1. RTL線路過大，我認為是因為我設計時，使用了三個counters。可以改成只用一個counter來實作，減少電路量。
   2. 如果改用一個counter製作，我會把timer在單獨拉出一個獨立的模組。這樣可以在修改時，簡潔易點。
2. **附件**

|  |
| --- |
| /\*part7 day clock\*/ |
| module part7 (SW, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, CLOCK\_50 );  input CLOCK\_50;  input [9:0] SW;  output [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;  reg reset, reset1, reset2;  wire clock;  wire [25:0] sec\_Q;  wire [31:0] min\_Q;  wire [37:0] hr\_Q;  reg [5:0] sec;  reg [7:0] min;  reg [7:0] hr;  reg [7:0] set\_time; //二進制  assign clk = CLOCK\_50;  //define sec/min/hr counters  part7\_counter sec\_cr(clk, reset, sec\_Q);  part7\_counter1 min\_cr(clk, reset1, min\_Q);  part7\_counter2 hr\_cr(clk, reset2, hr\_Q);  //SW  reg SW8\_N, SW8\_P, set, select;  always @(\*)begin  set = (~SW8\_P) & SW8\_N;  select = SW[9];  end  always @(posedge clk)begin  SW8\_N <= SW[8];  SW8\_P <= SW8\_N;  set\_time[7:0] <= SW[7:0];  end  //========== sec/min/hr time ========  //sec\_ 0~59  always @(posedge clk)begin  if(sec\_Q == 26'd49999998)  reset <= 1;  else if(sec\_Q >= 26'd49999999)begin  if(sec >= 10'd59)begin  sec <= 0;  reset <= 0;  end  else begin  sec <= sec + 1;  reset <= 0;  end  end  else  reset <= 0;  end  //min\_ 0~59  always @(posedge clk)begin  if(set && ~select && (set\_time < 60))  min <= set\_time;  else begin  if(min\_Q == 32'd2999999998)  reset1 <= 1;  else if(min\_Q >= 32'd2999999999)begin  if(min >= 10'd59)begin  min <= 0;  reset1 <= 0;  end  else begin  min <= min + 1;  reset1 <= 0;  end  end  else  reset1 <= 0;  end  end  //hr\_ 0~24  always @(posedge clk)begin  if(hr == 23 && min == 59 && sec == 59 && sec\_Q == 26'd49999999)  hr <= 0;  else if( min == 59 && sec == 59 && sec\_Q == 26'd49999999)  hr <= hr +1;  else if(set && select && (set\_time < 24))  hr <= set\_time;  else begin  if(hr\_Q == 38'd179999999998)  reset2 <= 1;  else if(hr\_Q >= 38'd179999999999)begin  if(hr >= 10'd24)begin  hr <= 0;  reset2 <= 0;  end  else begin  hr <= hr + 1;  reset2 <= 0;  end  end  else  reset2 <= 0;  end  end  //==================== BCD ==========================  //BCD sec  wire [5:0] bin;  assign bin = sec;  integer i;  reg [13:0] shift\_reg;  reg [3:0] sec\_bcd\_t, sec\_bcd\_o;  always @(\*) begin  shift\_reg = {8'b0, bin}; // 初始化轉換暫存器  for (i = 0; i < 6; i = i + 1) begin  if (shift\_reg[13:10] >= 5)  shift\_reg[13:10] = shift\_reg[13:10] + 3;  if (shift\_reg[9:6] >= 5)  shift\_reg[9:6] = shift\_reg[9:6] + 3;  shift\_reg = shift\_reg << 1; // 左移 1 位  end  sec\_bcd\_t = shift\_reg[13:10]; // 十位  sec\_bcd\_o = shift\_reg[9:6]; // 個位  end  //BCD min  wire [7:0] bin1;  assign bin1 = min;  integer j;  reg [13:0] shift\_reg1;  reg [3:0] min\_bcd\_t, min\_bcd\_o;  always @(\*) begin  shift\_reg1 = {6'b0, bin1}; // 初始化轉換暫存器  for (j = 0; j < 6; j = j + 1) begin  if (shift\_reg1[13:10] >= 5)  shift\_reg1[13:10] = shift\_reg1[13:10] + 3;  if (shift\_reg1[9:6] >= 5)  shift\_reg1[9:6] = shift\_reg1[9:6] + 3;  shift\_reg1 = shift\_reg1 << 1; // 左移 1 位  end  min\_bcd\_t = shift\_reg1[13:10]; // 十位  min\_bcd\_o = shift\_reg1[9:6]; // 個位  end  //BCD hr  wire [7:0] bin2;  assign bin2 = hr;  integer k;  reg [12:0] shift\_reg2;  reg [3:0] hr\_bcd\_t, hr\_bcd\_o;  always @(\*) begin  shift\_reg2 = {5'b0, bin2}; // 初始化轉換暫存器  for (k = 0; k < 5; k = k + 1) begin  if (shift\_reg2[12:9] >= 5)  shift\_reg2[12:9] = shift\_reg2[12:9] + 3;  if (shift\_reg2[8:5] >= 5)  shift\_reg2[8:5] = shift\_reg2[8:5] + 3;  shift\_reg2 = shift\_reg2 << 1; // 左移 1 位  end  hr\_bcd\_t = shift\_reg2[12:9]; // 十位  hr\_bcd\_o = shift\_reg2[8:5]; // 個位  end  //================ HEX5~0 =========================  //HEX0, HEX1  reg [6:0] reg\_HEX0, reg\_HEX1;  assign HEX0 = reg\_HEX0;  assign HEX1 = reg\_HEX1;  always @(\*)begin  case (sec\_bcd\_o)  4'b0000: reg\_HEX0 = 7'b1000000; // 0  4'b0001: reg\_HEX0 = 7'b1111001; // 1  4'b0010: reg\_HEX0 = 7'b0100100; // 2  4'b0011: reg\_HEX0 = 7'b0110000; // 3  4'b0100: reg\_HEX0 = 7'b0011001; // 4  4'b0101: reg\_HEX0 = 7'b0010010; // 5  4'b0110: reg\_HEX0 = 7'b0000010; // 6  4'b0111: reg\_HEX0 = 7'b1111000; // 7  4'b1000: reg\_HEX0 = 7'b0000000; // 8  4'b1001: reg\_HEX0 = 7'b0010000; // 9  default: reg\_HEX0 = 7'b1000000; // 0  endcase  end  always @(\*)begin  case (sec\_bcd\_t)  4'b0000: reg\_HEX1 = 7'b1000000; // 0  4'b0001: reg\_HEX1 = 7'b1111001; // 1  4'b0010: reg\_HEX1 = 7'b0100100; // 2  4'b0011: reg\_HEX1 = 7'b0110000; // 3  4'b0100: reg\_HEX1 = 7'b0011001; // 4  4'b0101: reg\_HEX1 = 7'b0010010; // 5  4'b0110: reg\_HEX1 = 7'b0000010; // 6  4'b0111: reg\_HEX1 = 7'b1111000; // 7  4'b1000: reg\_HEX1 = 7'b0000000; // 8  4'b1001: reg\_HEX1 = 7'b0010000; // 9  default: reg\_HEX1 = 7'b1000000; // 0  endcase  end  //HEX2, HEX3  reg [6:0] reg\_HEX2, reg\_HEX3;  assign HEX2 = reg\_HEX2;  assign HEX3 = reg\_HEX3;  always @(\*)begin  case (min\_bcd\_o)  4'b0000: reg\_HEX2 = 7'b1000000; // 0  4'b0001: reg\_HEX2 = 7'b1111001; // 1  4'b0010: reg\_HEX2 = 7'b0100100; // 2  4'b0011: reg\_HEX2 = 7'b0110000; // 3  4'b0100: reg\_HEX2 = 7'b0011001; // 4  4'b0101: reg\_HEX2 = 7'b0010010; // 5  4'b0110: reg\_HEX2 = 7'b0000010; // 6  4'b0111: reg\_HEX2 = 7'b1111000; // 7  4'b1000: reg\_HEX2 = 7'b0000000; // 8  4'b1001: reg\_HEX2 = 7'b0010000; // 9  default: reg\_HEX2 = 7'b1000000; // 0  endcase  end  always @(\*)begin  case (min\_bcd\_t)  4'b0000: reg\_HEX3 = 7'b1000000; // 0  4'b0001: reg\_HEX3 = 7'b1111001; // 1  4'b0010: reg\_HEX3 = 7'b0100100; // 2  4'b0011: reg\_HEX3 = 7'b0110000; // 3  4'b0100: reg\_HEX3 = 7'b0011001; // 4  4'b0101: reg\_HEX3 = 7'b0010010; // 5  4'b0110: reg\_HEX3 = 7'b0000010; // 6  4'b0111: reg\_HEX3 = 7'b1111000; // 7  4'b1000: reg\_HEX3 = 7'b0000000; // 8  4'b1001: reg\_HEX3 = 7'b0010000; // 9  default: reg\_HEX3 = 7'b1000000; // 0  endcase  end  //HEX4, HEX5  reg [6:0] reg\_HEX4, reg\_HEX5;  assign HEX4 = reg\_HEX4;  assign HEX5 = reg\_HEX5;  always @(\*)begin  case (hr\_bcd\_o)  4'b0000: reg\_HEX4 = 7'b1000000; // 0  4'b0001: reg\_HEX4 = 7'b1111001; // 1  4'b0010: reg\_HEX4 = 7'b0100100; // 2  4'b0011: reg\_HEX4 = 7'b0110000; // 3  4'b0100: reg\_HEX4 = 7'b0011001; // 4  4'b0101: reg\_HEX4 = 7'b0010010; // 5  4'b0110: reg\_HEX4 = 7'b0000010; // 6  4'b0111: reg\_HEX4 = 7'b1111000; // 7  4'b1000: reg\_HEX4 = 7'b0000000; // 8  4'b1001: reg\_HEX4 = 7'b0010000; // 9  default: reg\_HEX4 = 7'b1000000; // 0  endcase  end  always @(\*)begin  case (hr\_bcd\_t)  4'b0000: reg\_HEX5 = 7'b1000000; // 0  4'b0001: reg\_HEX5 = 7'b1111001; // 1  4'b0010: reg\_HEX5 = 7'b0100100; // 2  4'b0011: reg\_HEX5 = 7'b0110000; // 3  4'b0100: reg\_HEX5 = 7'b0011001; // 4  4'b0101: reg\_HEX5 = 7'b0010010; // 5  4'b0110: reg\_HEX5 = 7'b0000010; // 6  4'b0111: reg\_HEX5 = 7'b1111000; // 7  4'b1000: reg\_HEX5 = 7'b0000000; // 8  4'b1001: reg\_HEX5 = 7'b0010000; // 9  default: reg\_HEX5 = 7'b1000000; // 0  endcase  end  endmodule |

|  |
| --- |
| part7\_counter.v |
| // megafunction wizard: %LPM\_COUNTER%  // GENERATION: STANDARD  // VERSION: WM1.0  // MODULE: LPM\_COUNTER  // ============================================================  // File Name: part7\_counter.v  // Megafunction Name(s):  // LPM\_COUNTER  //  // Simulation Library Files(s):  // lpm  // ============================================================  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!  //  // 13.1.0 Build 162 10/23/2013 SJ Full Version  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Copyright (C) 1991-2013 Altera Corporation  //Your use of Altera Corporation's design tools, logic functions  //and other software and tools, and its AMPP partner logic  //functions, and any output files from any of the foregoing  //(including device programming or simulation files), and any  //associated documentation or information are expressly subject  //to the terms and conditions of the Altera Program License  //Subscription Agreement, Altera MegaCore Function License  //Agreement, or other applicable license agreement, including,  //without limitation, that your use is for the sole purpose of  //programming logic devices manufactured by Altera and sold by  //Altera or its authorized distributors. Please refer to the  //applicable agreement for further details.  // synopsys translate\_off  `timescale 1 ps / 1 ps  // synopsys translate\_on  module part7\_counter (  clock,  sclr,  q);  input clock;  input sclr;  output [25:0] q;  wire [25:0] sub\_wire0;  wire [25:0] q = sub\_wire0[25:0];  lpm\_counter LPM\_COUNTER\_component (  .clock (clock),  .sclr (sclr),  .q (sub\_wire0),  .aclr (1'b0),  .aload (1'b0),  .aset (1'b0),  .cin (1'b1),  .clk\_en (1'b1),  .cnt\_en (1'b1),  .cout (),  .data ({26{1'b0}}),  .eq (),  .sload (1'b0),  .sset (1'b0),  .updown (1'b1));  defparam  LPM\_COUNTER\_component.lpm\_direction = "UP",  LPM\_COUNTER\_component.lpm\_port\_updown = "PORT\_UNUSED",  LPM\_COUNTER\_component.lpm\_type = "LPM\_COUNTER",  LPM\_COUNTER\_component.lpm\_width = 26;  endmodule  // ============================================================  // CNX file retrieval info  // ============================================================  // Retrieval info: PRIVATE: ACLR NUMERIC "0"  // Retrieval info: PRIVATE: ALOAD NUMERIC "0"  // Retrieval info: PRIVATE: ASET NUMERIC "0"  // Retrieval info: PRIVATE: ASET\_ALL1 NUMERIC "1"  // Retrieval info: PRIVATE: CLK\_EN NUMERIC "0"  // Retrieval info: PRIVATE: CNT\_EN NUMERIC "0"  // Retrieval info: PRIVATE: CarryIn NUMERIC "0"  // Retrieval info: PRIVATE: CarryOut NUMERIC "0"  // Retrieval info: PRIVATE: Direction NUMERIC "0"  // Retrieval info: PRIVATE: INTENDED\_DEVICE\_FAMILY STRING "Cyclone V"  // Retrieval info: PRIVATE: ModulusCounter NUMERIC "0"  // Retrieval info: PRIVATE: ModulusValue NUMERIC "0"  // Retrieval info: PRIVATE: SCLR NUMERIC "1"  // Retrieval info: PRIVATE: SLOAD NUMERIC "0"  // Retrieval info: PRIVATE: SSET NUMERIC "0"  // Retrieval info: PRIVATE: SSET\_ALL1 NUMERIC "1"  // Retrieval info: PRIVATE: SYNTH\_WRAPPER\_GEN\_POSTFIX STRING "0"  // Retrieval info: PRIVATE: nBit NUMERIC "26"  // Retrieval info: PRIVATE: new\_diagram STRING "1"  // Retrieval info: LIBRARY: lpm lpm.lpm\_components.all  // Retrieval info: CONSTANT: LPM\_DIRECTION STRING "UP"  // Retrieval info: CONSTANT: LPM\_PORT\_UPDOWN STRING "PORT\_UNUSED"  // Retrieval info: CONSTANT: LPM\_TYPE STRING "LPM\_COUNTER"  // Retrieval info: CONSTANT: LPM\_WIDTH NUMERIC "26"  // Retrieval info: USED\_PORT: clock 0 0 0 0 INPUT NODEFVAL "clock"  // Retrieval info: USED\_PORT: q 0 0 26 0 OUTPUT NODEFVAL "q[25..0]"  // Retrieval info: USED\_PORT: sclr 0 0 0 0 INPUT NODEFVAL "sclr"  // Retrieval info: CONNECT: @clock 0 0 0 0 clock 0 0 0 0  // Retrieval info: CONNECT: @sclr 0 0 0 0 sclr 0 0 0 0  // Retrieval info: CONNECT: q 0 0 26 0 @q 0 0 26 0  // Retrieval info: GEN\_FILE: TYPE\_NORMAL part7\_counter.v TRUE  // Retrieval info: GEN\_FILE: TYPE\_NORMAL part7\_counter.inc FALSE  // Retrieval info: GEN\_FILE: TYPE\_NORMAL part7\_counter.cmp FALSE  // Retrieval info: GEN\_FILE: TYPE\_NORMAL part7\_counter.bsf FALSE  // Retrieval info: GEN\_FILE: TYPE\_NORMAL part7\_counter\_inst.v FALSE  // Retrieval info: GEN\_FILE: TYPE\_NORMAL part7\_counter\_bb.v TRUE  // Retrieval info: LIB\_FILE: lpm |
| part7\_counter1.v |
| / megafunction wizard: %LPM\_COUNTER%  // GENERATION: STANDARD  // VERSION: WM1.0  // MODULE: LPM\_COUNTER  // ============================================================  // File Name: part7\_counter1.v  // Megafunction Name(s):  // LPM\_COUNTER  //  // Simulation Library Files(s):  // lpm  // ============================================================  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!  //  // 13.1.0 Build 162 10/23/2013 SJ Full Version  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Copyright (C) 1991-2013 Altera Corporation  //Your use of Altera Corporation's design tools, logic functions  //and other software and tools, and its AMPP partner logic  //functions, and any output files from any of the foregoing  //(including device programming or simulation files), and any  //associated documentation or information are expressly subject  //to the terms and conditions of the Altera Program License  //Subscription Agreement, Altera MegaCore Function License  //Agreement, or other applicable license agreement, including,  //without limitation, that your use is for the sole purpose of  //programming logic devices manufactured by Altera and sold by  //Altera or its authorized distributors. Please refer to the  //applicable agreement for further details.  // synopsys translate\_off  `timescale 1 ps / 1 ps  // synopsys translate\_on  module part7\_counter1 (  clock,  sclr,  q);  input clock;  input sclr;  output [31:0] q;  wire [31:0] sub\_wire0;  wire [31:0] q = sub\_wire0[31:0];  lpm\_counter LPM\_COUNTER\_component (  .clock (clock),  .sclr (sclr),  .q (sub\_wire0),  .aclr (1'b0),  .aload (1'b0),  .aset (1'b0),  .cin (1'b1),  .clk\_en (1'b1),  .cnt\_en (1'b1),  .cout (),  .data ({32{1'b0}}),  .eq (),  .sload (1'b0),  .sset (1'b0),  .updown (1'b1));  defparam  LPM\_COUNTER\_component.lpm\_direction = "UP",  LPM\_COUNTER\_component.lpm\_port\_updown = "PORT\_UNUSED",  LPM\_COUNTER\_component.lpm\_type = "LPM\_COUNTER",  LPM\_COUNTER\_component.lpm\_width = 32;  endmodule  // ============================================================  // CNX file retrieval info  // ============================================================  // Retrieval info: PRIVATE: ACLR NUMERIC "0"  // Retrieval info: PRIVATE: ALOAD NUMERIC "0"  // Retrieval info: PRIVATE: ASET NUMERIC "0"  // Retrieval info: PRIVATE: ASET\_ALL1 NUMERIC "1"  // Retrieval info: PRIVATE: CLK\_EN NUMERIC "0"  // Retrieval info: PRIVATE: CNT\_EN NUMERIC "0"  // Retrieval info: PRIVATE: CarryIn NUMERIC "0"  // Retrieval info: PRIVATE: CarryOut NUMERIC "0"  // Retrieval info: PRIVATE: Direction NUMERIC "0"  // Retrieval info: PRIVATE: INTENDED\_DEVICE\_FAMILY STRING "Cyclone V"  // Retrieval info: PRIVATE: ModulusCounter NUMERIC "0"  // Retrieval info: PRIVATE: ModulusValue NUMERIC "0"  // Retrieval info: PRIVATE: SCLR NUMERIC "1"  // Retrieval info: PRIVATE: SLOAD NUMERIC "0"  // Retrieval info: PRIVATE: SSET NUMERIC "0"  // Retrieval info: PRIVATE: SSET\_ALL1 NUMERIC "1"  // Retrieval info: PRIVATE: SYNTH\_WRAPPER\_GEN\_POSTFIX STRING "0"  // Retrieval info: PRIVATE: nBit NUMERIC "32"  // Retrieval info: PRIVATE: new\_diagram STRING "1"  // Retrieval info: LIBRARY: lpm lpm.lpm\_components.all  // Retrieval info: CONSTANT: LPM\_DIRECTION STRING "UP"  // Retrieval info: CONSTANT: LPM\_PORT\_UPDOWN STRING "PORT\_UNUSED"  // Retrieval info: CONSTANT: LPM\_TYPE STRING "LPM\_COUNTER"  // Retrieval info: CONSTANT: LPM\_WIDTH NUMERIC "32"  // Retrieval info: USED\_PORT: clock 0 0 0 0 INPUT NODEFVAL "clock"  // Retrieval info: USED\_PORT: q 0 0 32 0 OUTPUT NODEFVAL "q[31..0]"  // Retrieval info: USED\_PORT: sclr 0 0 0 0 INPUT NODEFVAL "sclr"  // Retrieval info: CONNECT: @clock 0 0 0 0 clock 0 0 0 0  // Retrieval info: CONNECT: @sclr 0 0 0 0 sclr 0 0 0 0  // Retrieval info: CONNECT: q 0 0 32 0 @q 0 0 32 0  // Retrieval info: GEN\_FILE: TYPE\_NORMAL part7\_counter1.v TRUE  // Retrieval info: GEN\_FILE: TYPE\_NORMAL part7\_counter1.inc FALSE  // Retrieval info: GEN\_FILE: TYPE\_NORMAL part7\_counter1.cmp FALSE  // Retrieval info: GEN\_FILE: TYPE\_NORMAL part7\_counter1.bsf FALSE  // Retrieval info: GEN\_FILE: TYPE\_NORMAL part7\_counter1\_inst.v FALSE  // Retrieval info: GEN\_FILE: TYPE\_NORMAL part7\_counter1\_bb.v TRUE  // Retrieval info: LIB\_FILE: lpm |
| part7\_counter2.v |
| // megafunction wizard: %LPM\_COUNTER%  // GENERATION: STANDARD  // VERSION: WM1.0  // MODULE: LPM\_COUNTER  // ============================================================  // File Name: part7\_counter2.v  // Megafunction Name(s):  // LPM\_COUNTER  //  // Simulation Library Files(s):  // lpm  // ============================================================  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!  //  // 13.1.0 Build 162 10/23/2013 SJ Full Version  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Copyright (C) 1991-2013 Altera Corporation  //Your use of Altera Corporation's design tools, logic functions  //and other software and tools, and its AMPP partner logic  //functions, and any output files from any of the foregoing  //(including device programming or simulation files), and any  //associated documentation or information are expressly subject  //to the terms and conditions of the Altera Program License  //Subscription Agreement, Altera MegaCore Function License  //Agreement, or other applicable license agreement, including,  //without limitation, that your use is for the sole purpose of  //programming logic devices manufactured by Altera and sold by  //Altera or its authorized distributors. Please refer to the  //applicable agreement for further details.  // synopsys translate\_off  `timescale 1 ps / 1 ps  // synopsys translate\_on  module part7\_counter2 (  clock,  sclr,  q);  input clock;  input sclr;  output [31:0] q;  wire [31:0] sub\_wire0;  wire [31:0] q = sub\_wire0[31:0];  lpm\_counter LPM\_COUNTER\_component (  .clock (clock),  .sclr (sclr),  .q (sub\_wire0),  .aclr (1'b0),  .aload (1'b0),  .aset (1'b0),  .cin (1'b1),  .clk\_en (1'b1),  .cnt\_en (1'b1),  .cout (),  .data ({32{1'b0}}),  .eq (),  .sload (1'b0),  .sset (1'b0),  .updown (1'b1));  defparam  LPM\_COUNTER\_component.lpm\_direction = "UP",  LPM\_COUNTER\_component.lpm\_port\_updown = "PORT\_UNUSED",  LPM\_COUNTER\_component.lpm\_type = "LPM\_COUNTER",  LPM\_COUNTER\_component.lpm\_width = 32;  endmodule  // ============================================================  // CNX file retrieval info  // ============================================================  // Retrieval info: PRIVATE: ACLR NUMERIC "0"  // Retrieval info: PRIVATE: ALOAD NUMERIC "0"  // Retrieval info: PRIVATE: ASET NUMERIC "0"  // Retrieval info: PRIVATE: ASET\_ALL1 NUMERIC "1"  // Retrieval info: PRIVATE: CLK\_EN NUMERIC "0"  // Retrieval info: PRIVATE: CNT\_EN NUMERIC "0"  // Retrieval info: PRIVATE: CarryIn NUMERIC "0"  // Retrieval info: PRIVATE: CarryOut NUMERIC "0"  // Retrieval info: PRIVATE: Direction NUMERIC "0"  // Retrieval info: PRIVATE: INTENDED\_DEVICE\_FAMILY STRING "Cyclone V"  // Retrieval info: PRIVATE: ModulusCounter NUMERIC "0"  // Retrieval info: PRIVATE: ModulusValue NUMERIC "0"  // Retrieval info: PRIVATE: SCLR NUMERIC "1"  // Retrieval info: PRIVATE: SLOAD NUMERIC "0"  // Retrieval info: PRIVATE: SSET NUMERIC "0"  // Retrieval info: PRIVATE: SSET\_ALL1 NUMERIC "1"  // Retrieval info: PRIVATE: SYNTH\_WRAPPER\_GEN\_POSTFIX STRING "0"  // Retrieval info: PRIVATE: nBit NUMERIC "32"  // Retrieval info: PRIVATE: new\_diagram STRING "1"  // Retrieval info: LIBRARY: lpm lpm.lpm\_components.all  // Retrieval info: CONSTANT: LPM\_DIRECTION STRING "UP"  // Retrieval info: CONSTANT: LPM\_PORT\_UPDOWN STRING "PORT\_UNUSED"  // Retrieval info: CONSTANT: LPM\_TYPE STRING "LPM\_COUNTER"  // Retrieval info: CONSTANT: LPM\_WIDTH NUMERIC "32"  // Retrieval info: USED\_PORT: clock 0 0 0 0 INPUT NODEFVAL "clock"  // Retrieval info: USED\_PORT: q 0 0 32 0 OUTPUT NODEFVAL "q[31..0]"  // Retrieval info: USED\_PORT: sclr 0 0 0 0 INPUT NODEFVAL "sclr"  // Retrieval info: CONNECT: @clock 0 0 0 0 clock 0 0 0 0  // Retrieval info: CONNECT: @sclr 0 0 0 0 sclr 0 0 0 0  // Retrieval info: CONNECT: q 0 0 32 0 @q 0 0 32 0  // Retrieval info: GEN\_FILE: TYPE\_NORMAL part7\_counter2.v TRUE  // Retrieval info: GEN\_FILE: TYPE\_NORMAL part7\_counter2.inc FALSE  // Retrieval info: GEN\_FILE: TYPE\_NORMAL part7\_counter2.cmp FALSE  // Retrieval info: GEN\_FILE: TYPE\_NORMAL part7\_counter2.bsf FALSE  // Retrieval info: GEN\_FILE: TYPE\_NORMAL part7\_counter2\_inst.v FALSE  // Retrieval info: GEN\_FILE: TYPE\_NORMAL part7\_counter2\_bb.v TRUE  // Retrieval info: LIB\_FILE: lpm |