數位系統技術

Alarm Clock

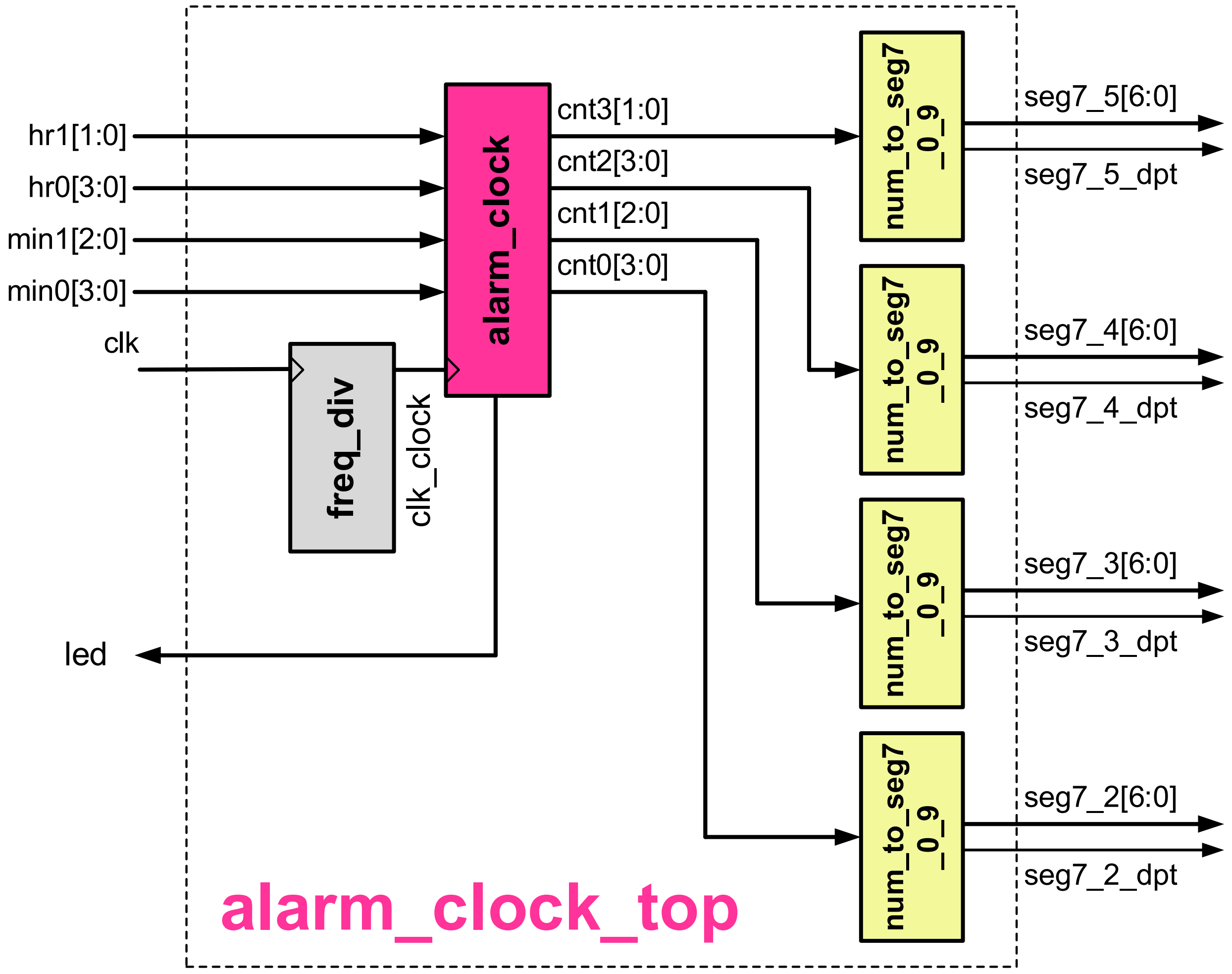
Ren-Der Chen (陳仁德)
Department of Computer Science and Information Engineering
National Changhua University of Education
E-mail: rdchen@cc.ncue.edu.tw
Spring, 2025

實驗內容

- 利用seg77設計一個具鬧鐘功能之時鐘電路。
- 鬧鐘動作之時間由外部輸入，以led亮燈表示鬧鐘正在動作，持續時間為10分鐘，之後led自動變暗。
- 將所設計之電路燒錄至FPGA晶片中，並利用實驗板週邊進行電路功能驗證。

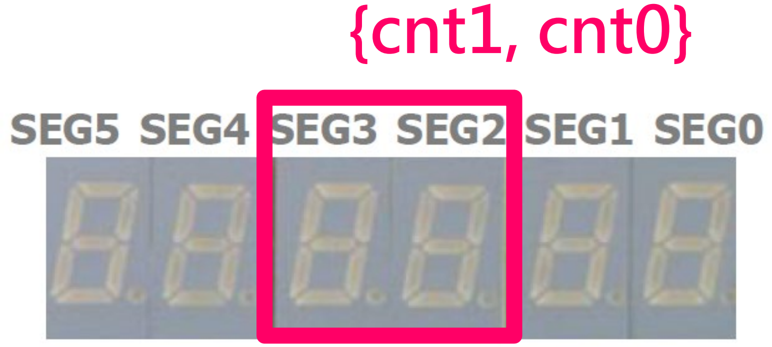
alarm_clock_top 電路方塊圖

- Create a new project – alarm_clock_top



Design 1: cnt_00_59_bcd (1/3)

- 設計一個00-59 bcd計數器cnt_00_59_bcd.v，作為時鐘之”分鐘”部分，並加入至project中。
- 計數範圍為 <u>00</u>, 01, 02, 03, …, 58, 59, <u>00</u>, 01, 02, … 。



Design 1: cnt_00_59_bcd (2/3)

```
module cnt_00_59_bcd(clk, rst, carry_out, cnt1, cnt0);
    input   clk, rst;
    output  carry_out;
    output  reg [2:0] cnt1;
    output  reg [3:0] cnt0;

    assign  carry_out =
            ({cnt1, cnt0} == {3'd5, 4'd9})
```

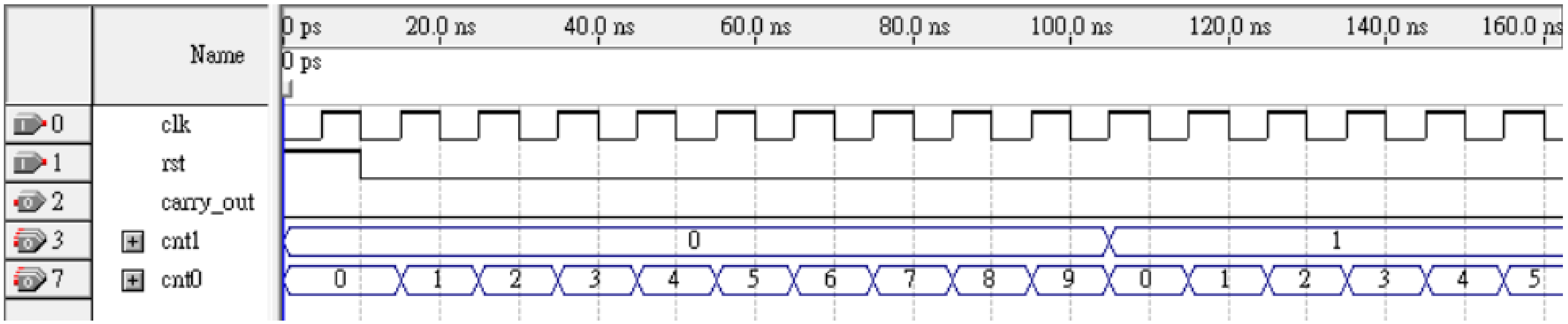
// {cnt1, cnt0} = 59 時產生進位(carry_out=1)至小時部分

Design 1: cnt_00_59_bcd (3/3)

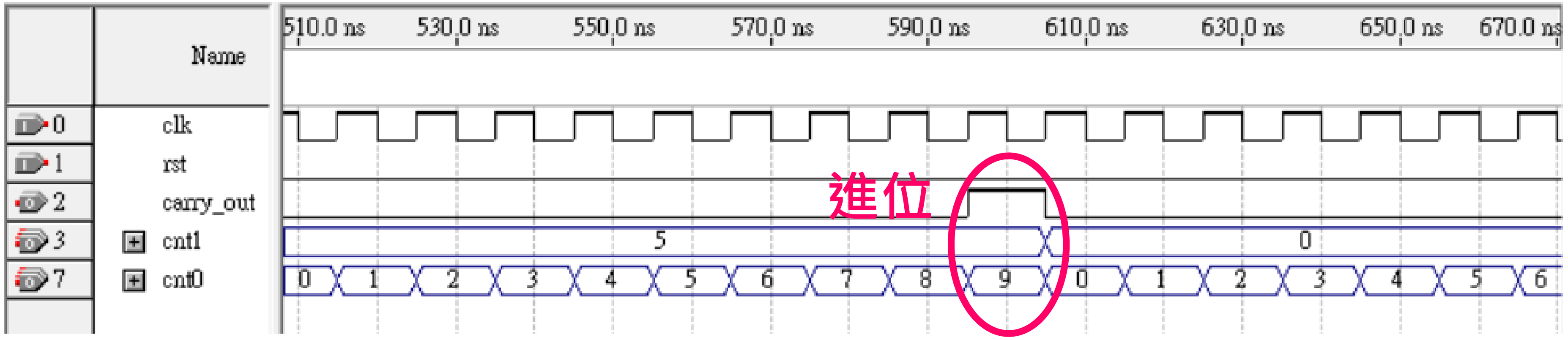
```
    always @(posedge clk or posedge rst) begin
        if (rst)
            {cnt1, cnt0} = {3'd0, 4'd0}; // 00
        else
            if ({cnt1, cnt0} == {3'd5, 4'd9}) // 59->00
                {cnt1, cnt0} = 
            else if (cnt0 == 4'd9)
                // 09->10, 19->20, 29->30, 39->40, 49->50
                {cnt1, cnt0} = 
            else
                cnt0 =              // +1
    end // always
endmodule
```

// {cnt1, cnt0} = <u>00</u>, 01, 02, 03, ..., 58, 59, <u>00</u>, 01, 02, ...

Functional Simulation for "cnt_00_59_bcd"

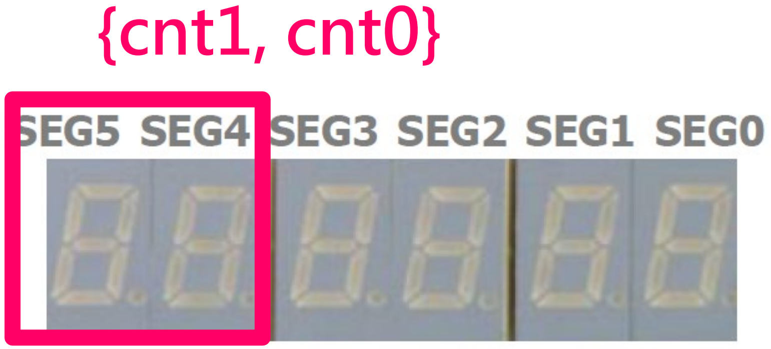


{cnt1, cnt0} = 00, 01, 02, 03, …, 58, 59, 00, 01, 02, …



Design 2: cnt_00_23_bcd (1/2)

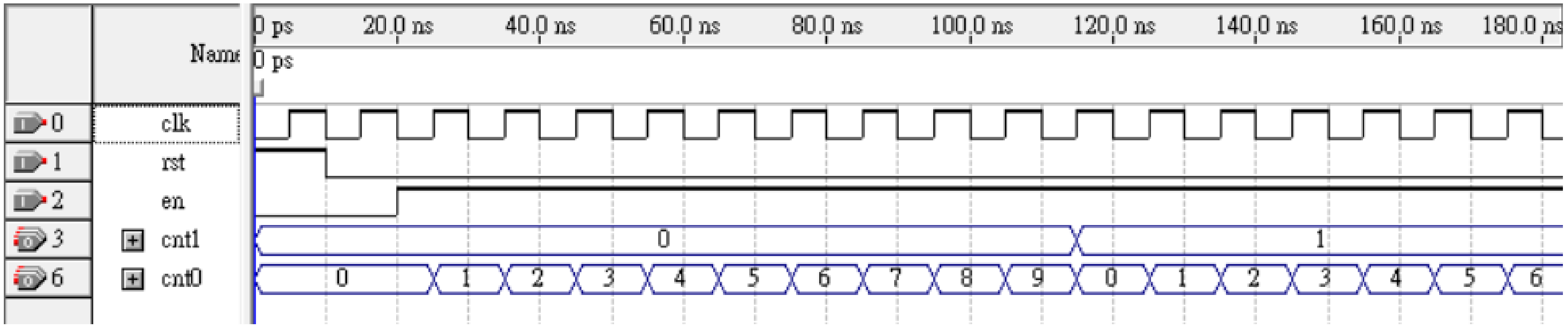
- 設計一個00-23 bcd計數器cnt_00_23_bcd.v，作為時鐘之"小時"部分，並加入至project中。
- 計數範圍為 <u>00</u>, 01, 02, 03, …, 22, 23, <u>00</u>, 01, 02, … 。



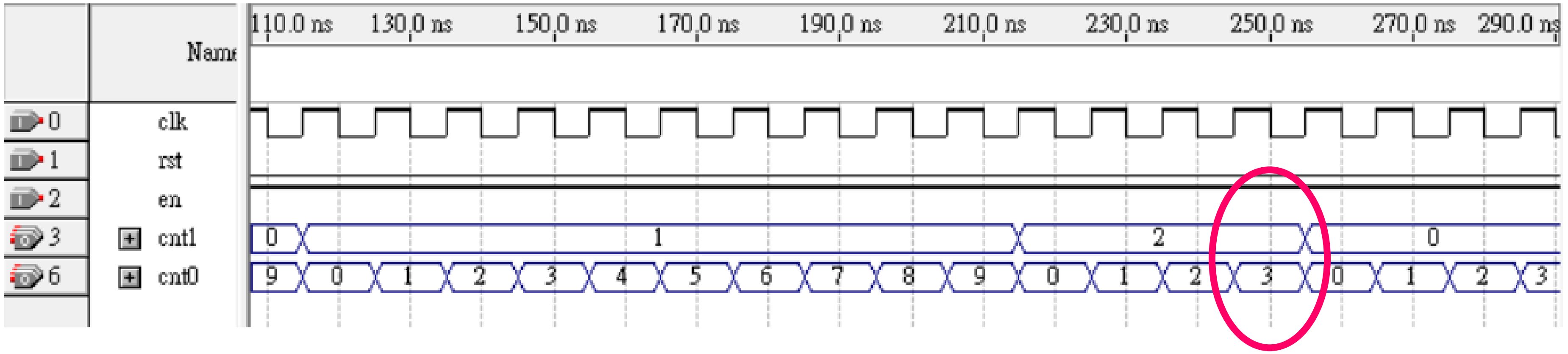
Design 2: cnt_00_23_bcd (2/2)

```
module cnt_00_23_bcd(clk, rst, en, cnt1, cnt0);
    input   clk, rst, en;
    output  reg [1:0] cnt1;
    output  reg [3:0] cnt0;

    always @(posedge clk or posedge rst) begin
        if (rst)
            {cnt1, cnt0} = {2'd0, 4'd0}; // 00
        else if (en == 1'b1) begin
            if ({cnt1, cnt0} == {2'd2, 4'd3}) // 23->00
                {cnt1, cnt0} = 
            else if (cnt0 == 4'd9)
                // 09->10, 19->20
                {cnt1, cnt0} = 
            else
                cnt0 =              // +1
        end // else if
    end // always
endmodule
```

Functional Simulation for "cnt_00_23_bcd"



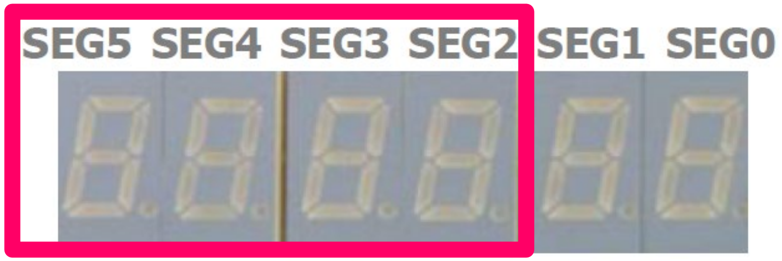
{cnt1, cnt0} = 00, 01, 02, 03, …, 22, 23, 00, 01, 02, …



Design 3: clock (1/2)

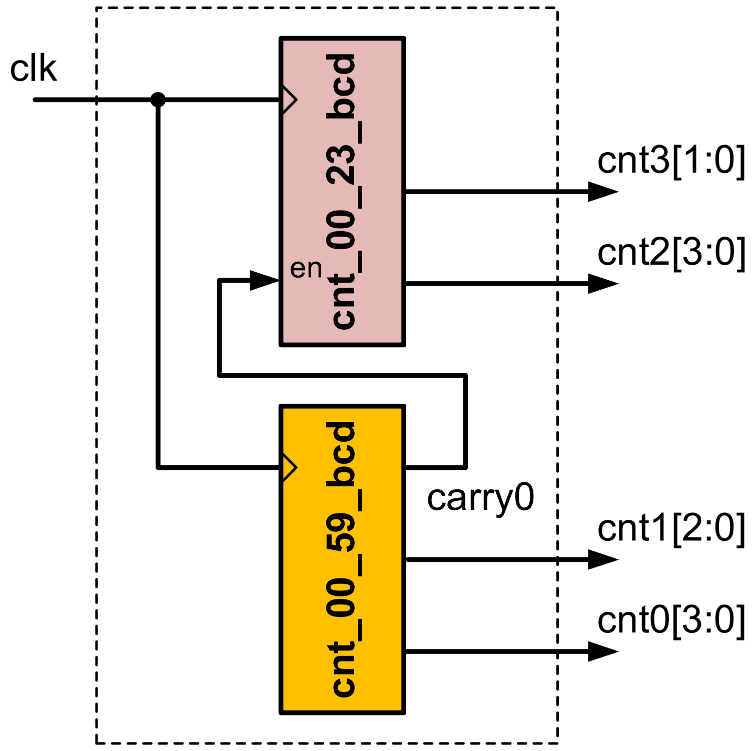
- 設計一個時鐘電路clock.v，並加入至project中。
- 電路輸出含小時及分鐘(hr:min)部分, 計數範圍為 <u>00:00</u>, 00:01, 00:02, …, 00:58, 00:59, 01:00, 01:01, …, 23:58, 23:59, <u>00:00</u>, 00:01, 00:02, … 。

{cnt3, cnt2, cnt1, cnt0}

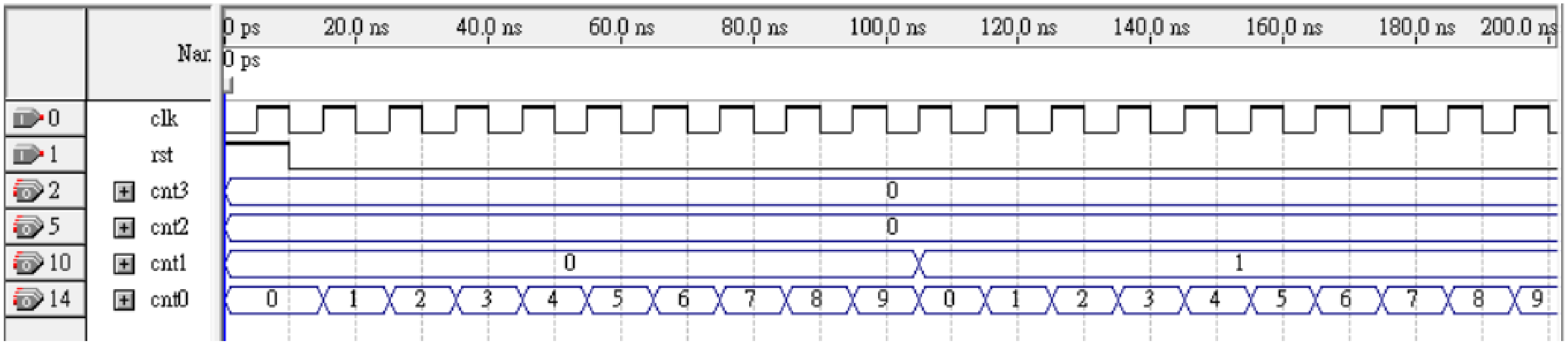


Design 3: clock (2/2)

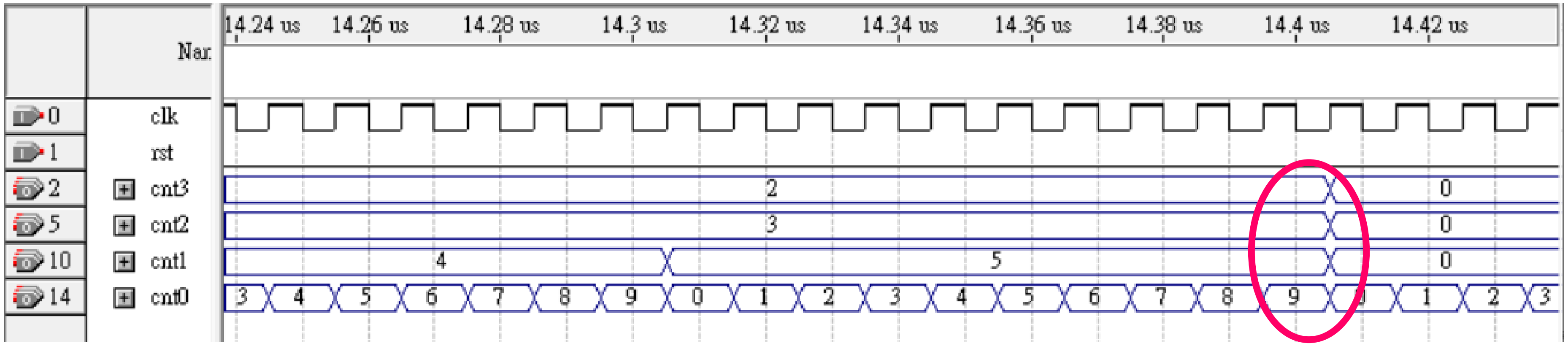
```
module clock(clk, rst, cnt3, cnt2, cnt1, cnt0);
    input   clk , rst;
    output  [1:0] cnt3;
    output  [3:0] cnt2;
    output  [2:0] cnt1;
    output  [3:0] cnt0;

    wire    carry0;








    // hour
    cnt_00_23_bcd m0
    // minute
    cnt_00_59_bcd m1
endmodule
```



Functional Simulation for “clock”



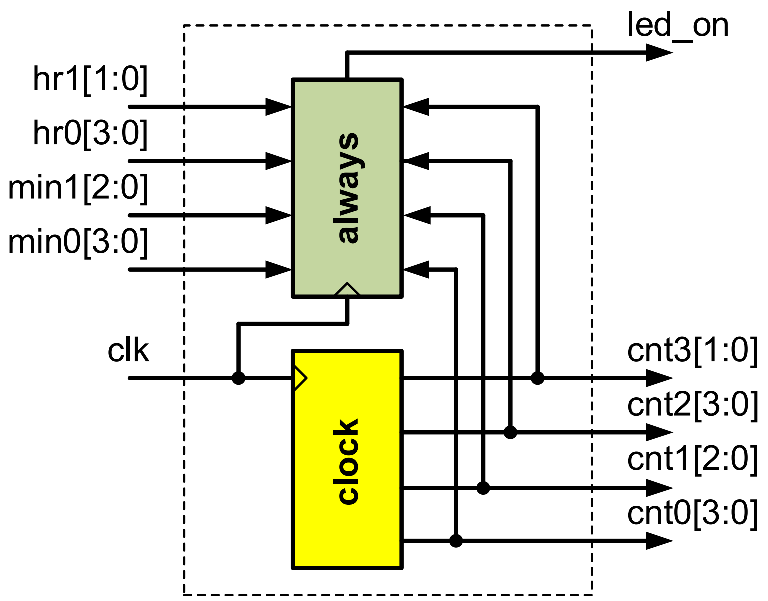
{cnt3, cnt2, cnt1, cnt0} = <u>00:00</u>, 00:01, 00:02, 00:03, …



{cnt3, cnt2, cnt1, cnt0} = 23:58, 23:59, <u>00:00</u>, 00:01

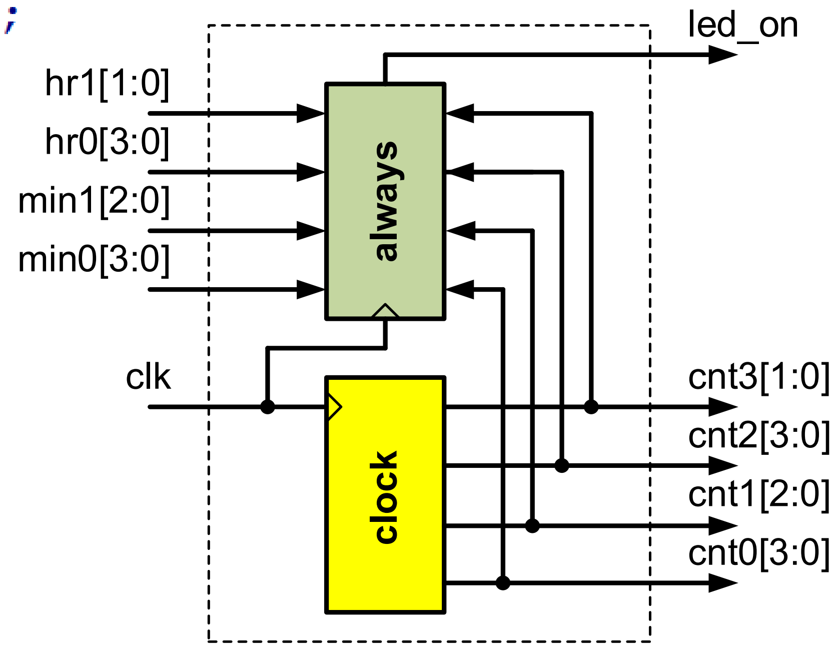
Design 4: alarm_clock (1/5)

- 利用時鐘電路clock.v，設計一個具鬧鐘功能之時鐘電路alarm_clock.v，並加入至project中。
- 當所設定之時間與目前clock計數之時間相同時， 產生led_on信號，使led燈亮，表示鬧鐘動作。
- 該led燈亮持續10 mins後，自動變暗，表示鬧鐘不再動作。



Design 4: alarm_clock (2/5)

```
module alarm_clock(clk, rst, hr1, hr0, min1, min0,
cnt3, cnt2, cnt1, cnt0, led_on);
    input   clk, rst;

    input   [1:0] hr1;
    input   [3:0] hr0;
    input   [2:0] min1;
    input   [3:0] min0;

    output  [1:0] cnt3;
    output  [3:0] cnt2;
    output  [2:0] cnt1;
    output  [3:0] cnt0;

    output  reg led_on;

    reg     [3:0] dur_cnt;
    // time for led on, 10 mins

    clock m0
```



鬧鐘有動作, led_on=1, led亮

鬧鐘無動作, led_on=0, led暗

dur_cnt: 記錄鬧鐘動作之時間, 預設為10分鐘關閉鬧鐘

Design 4: alarm_clock (3/5)

```
always @(posedge clk or posedge rst) begin
    if (rst)
        led_on = 1'b0;
    else begin
        // alarm on
        // {hr1, hr0, min1, min0} <-> {cnt3, cnt2, cnt1, cnt0}
        // {a, a, a, (1-9)} <-> {a, a, a, (0-8)}, ex. 10:28 <-> 10:27
        // {a, a, (1-5), 0} <-> {a, a, (0-4), 9}, ex. 12:30 <-> 12:29
        // {a, (1-9), 0, 0} <-> {a, (0-8), 5, 9}, ex. 15:00 <-> 14:59
        // {(1-2), 0, 0, 0} <-> {(0-1), 9, 5, 9}, ex. 20:00 <-> 19:59
        // {0, 0, 0, 0}     <-> {2, 3, 5, 9}    , ex. 00:00 <-> 23:59
```

為使鬧鐘在設定之時間啟動(led on), 當目前時間為(設定時間-1 min)時, led_on設定為1

Design 4: alarm_clock (4/5)

```
if  (
    ( (cnt3 == hr1) && (cnt2 == hr0) && (cnt1 == min1) && (cnt0 == min0-1'b1) ) ||

    ( (cnt3 == hr1) && (cnt2 == hr0) && (cnt1 == min1-1'b1) && (cnt0 == 4'd9) &&
      (min0 == 4'd0) ) ||



    ( (cnt3 == hr1-1'b1) && (cnt2 == 4'd9) && (cnt1 == 3'd5) && (cnt0 == 4'd9) &&
      (hr0 == 4'd0) && (min1 == 3'd0) && (min0 == 4'd0) ) ||

    ( (cnt3 == 2'd2) && (cnt2 == 4'd3) && (cnt1 == 3'd5) && (cnt0 == 4'd9) &&
      (hr1 == 2'd0)&& (hr0 == 4'd0)&& (min1 == 3'd0) && (min0 == 4'd0) )
    ) begin

    led_on = 1'b1;
    dur_cnt = 4'd0;
end // if
```

```
{hr1, hr0, min1, min0} <-> {cnt3, cnt2, cnt1, cnt0}
{a, a, a, (1-9)} <-> {a, a, a, (0-8)}, ex. 10:28 <-> 10:27
{a, a, (1-5), 0} <-> {a, a, (0-4), 9}, ex. 12:30 <-> 12:29
{a, (1-9), 0, 0} <-> {a, (0-8), 5, 9}, ex. 15:00 <-> 14:59
{(1-2), 0, 0, 0} <-> {(0-1), 9, 5, 9}, ex. 20:00 <-> 19:59
{0, 0, 0, 0}     <-> {2, 3, 5, 9}    , ex. 00:00 <-> 23:59
```

鬧鐘開始動作
led_on=1, led亮

Design 4: alarm_clock (5/5)

```
            if (led_on == 1'b1) begin
                dur_cnt = 
                
                if (dur_cnt ==        )
                    led_on = 1'b0;
            end // if
        end // else
    end // always
endmodule
```

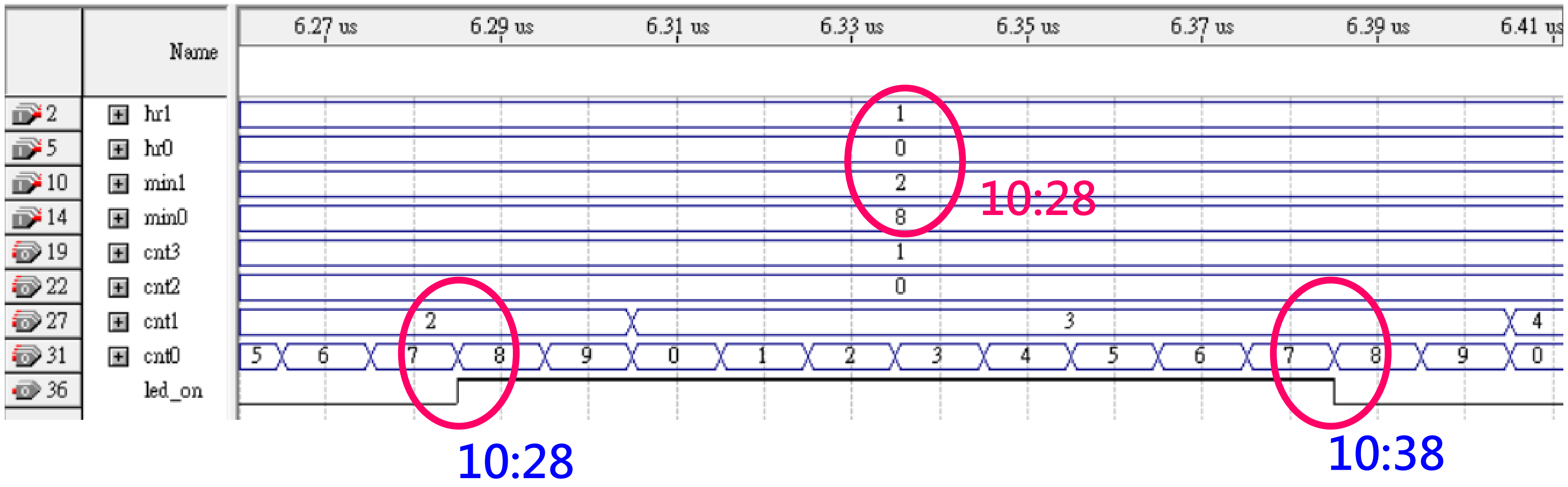
鬧鐘動作時間
遞增 1 min

鬧鐘動作時間是否已
到10 mins, 判斷10?

鬧鐘結束動作
led_on=0, led暗

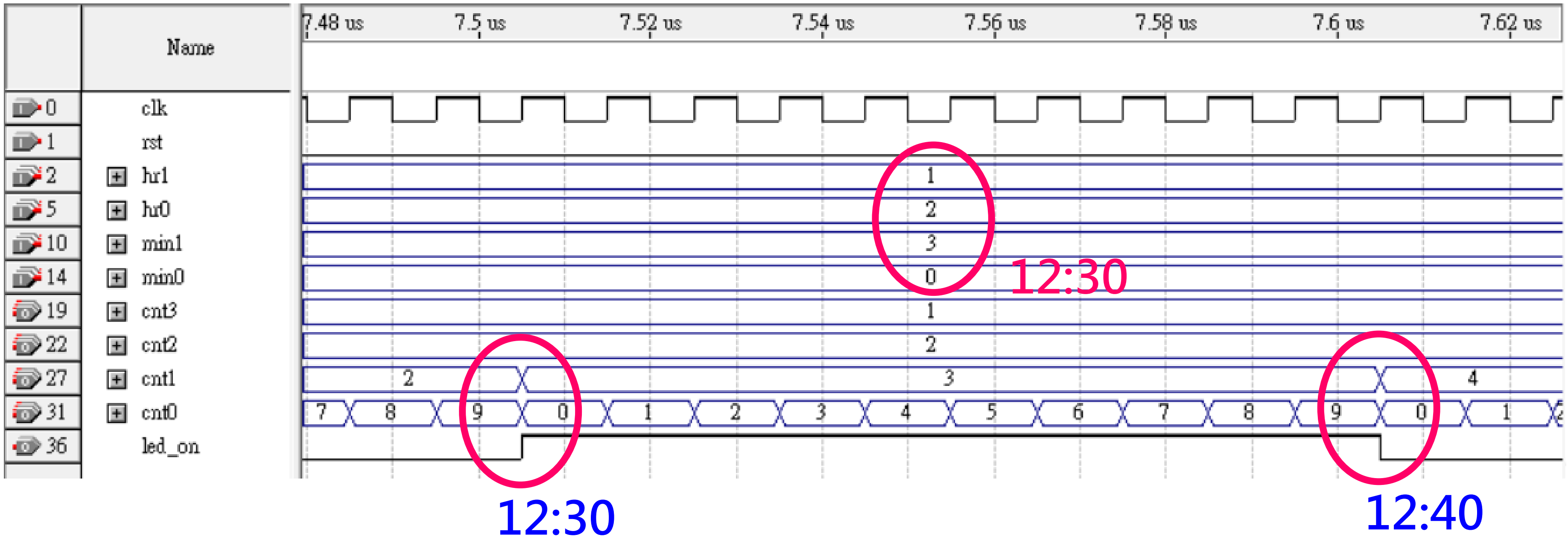
Functional Simulation for “alarm_clock” (1/5)

- {hr1, hr0, min1, min0} = 10:28



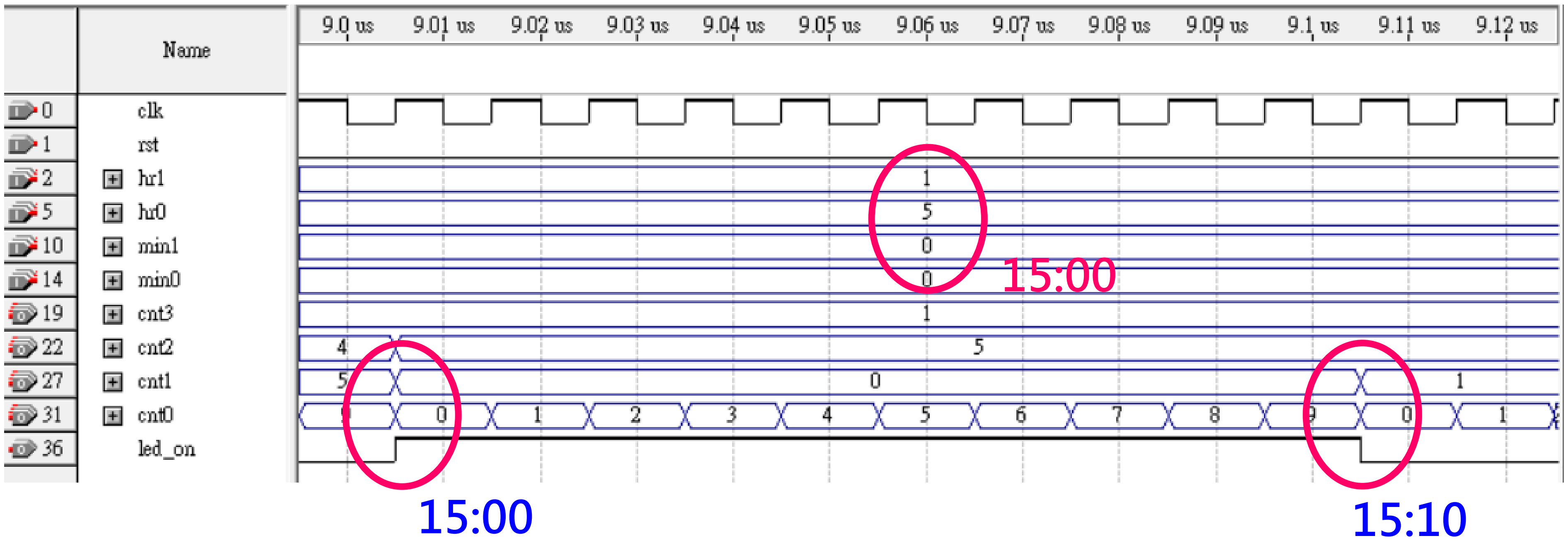
Functional Simulation for "alarm_clock" (2/5)

- {hr1, hr0, min1, min0} = 12:30



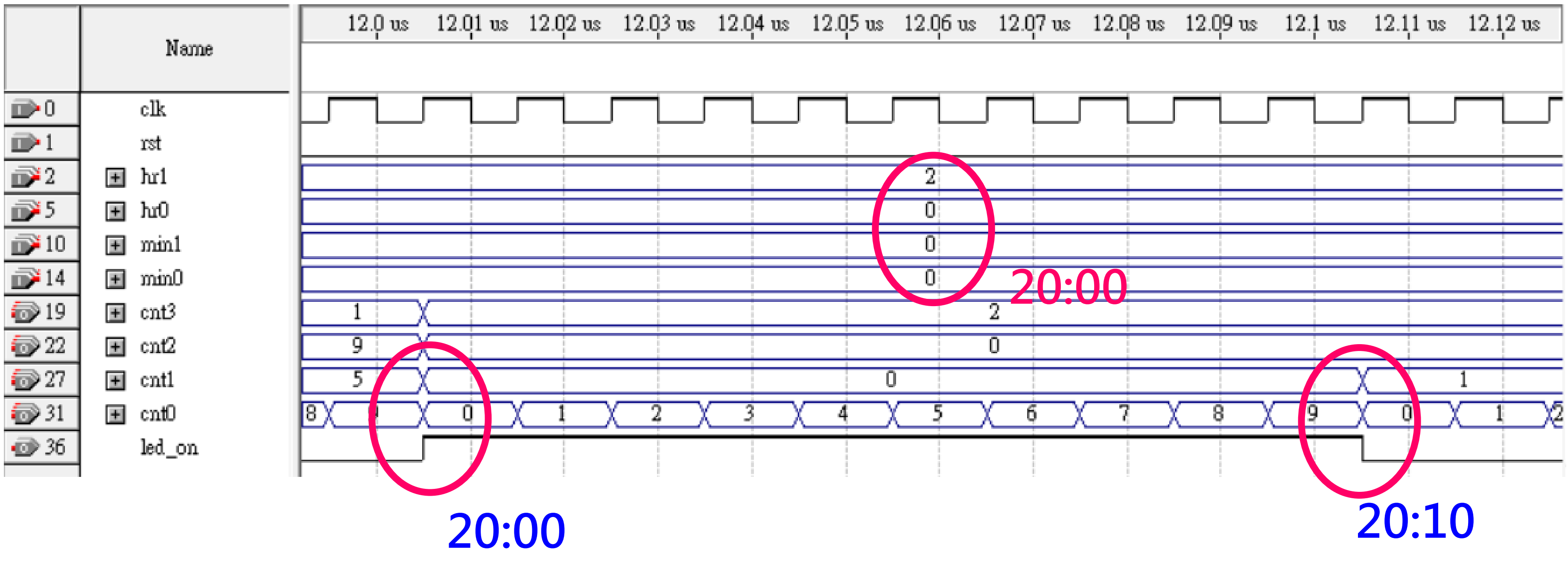
Functional Simulation for "alarm_clock" (3/5)

- {hr1, hr0, min1, min0} = 15:00



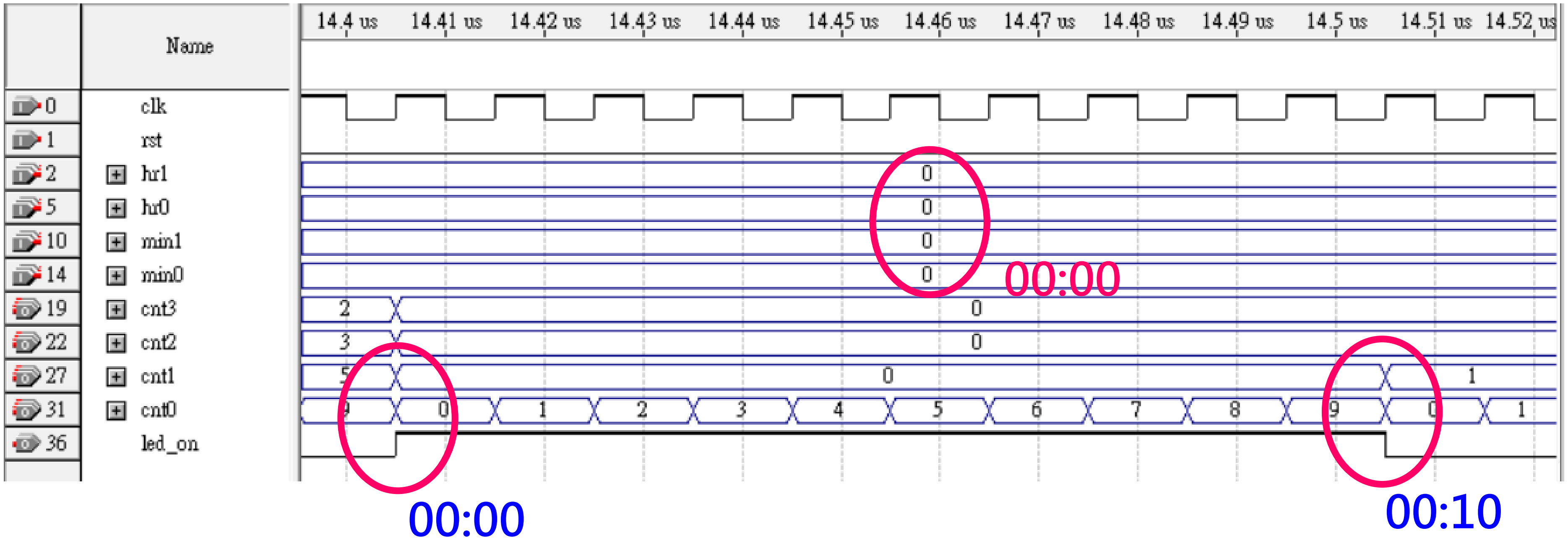
Functional Simulation for "alarm_clock" (4/5)

- {hr1, hr0, min1, min0} = 20:00



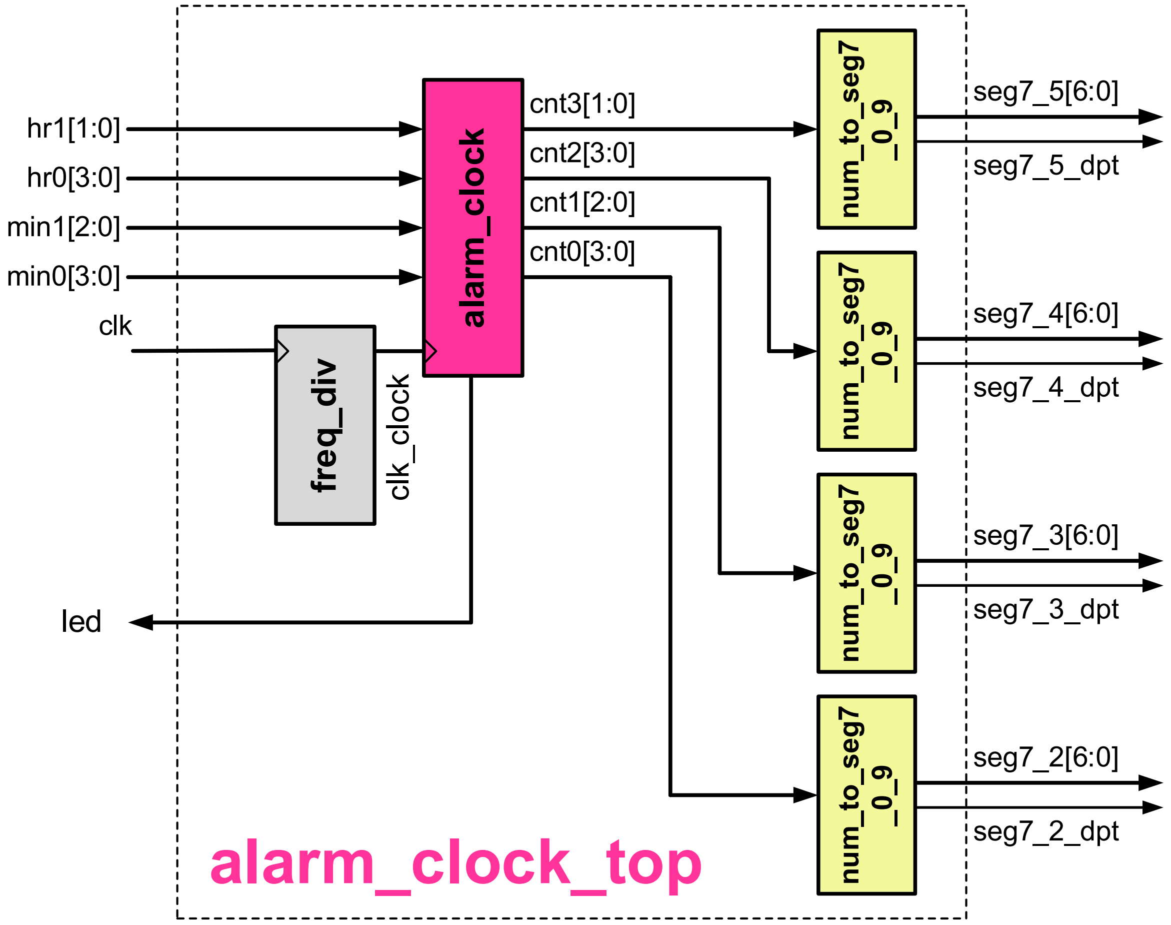
Functional Simulation for "alarm_clock" (5/5)

- {hr1, hr0, min1, min0} = 00:00



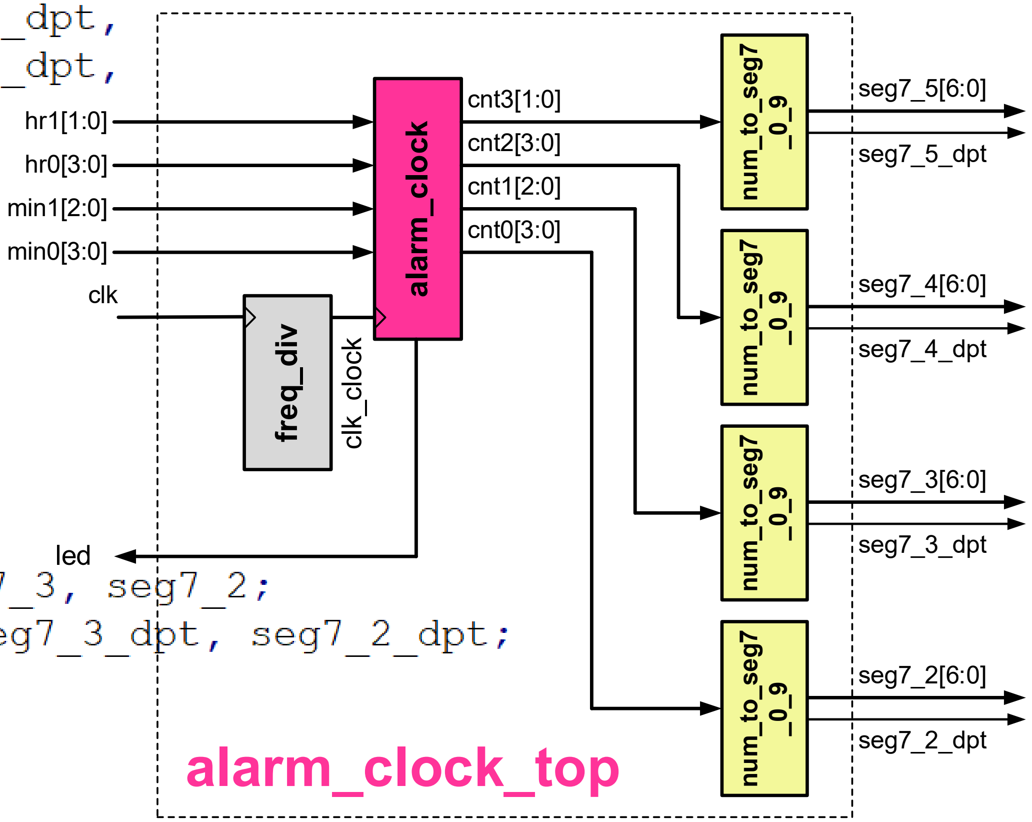
Design 5: alarm_clock_top (1/3)

- 設計alarm_clock_top.v並加入至project中，整合所有電路模組。



Design 5: alarm_clock_top (2/3)

```
module alarm_clock_top (clk, rst, hr1, hr0, min1, min0,
seg7_5, seg7_5_dpt, seg7_4, seg7_4_dpt,
seg7_3, seg7_3_dpt, seg7_2, seg7_2_dpt,
led);
      input   clk, rst;
      input   [1:0] hr1;
      input   [3:0] hr0;
      input   [2:0] min1;
      input   [3:0] min0;
```

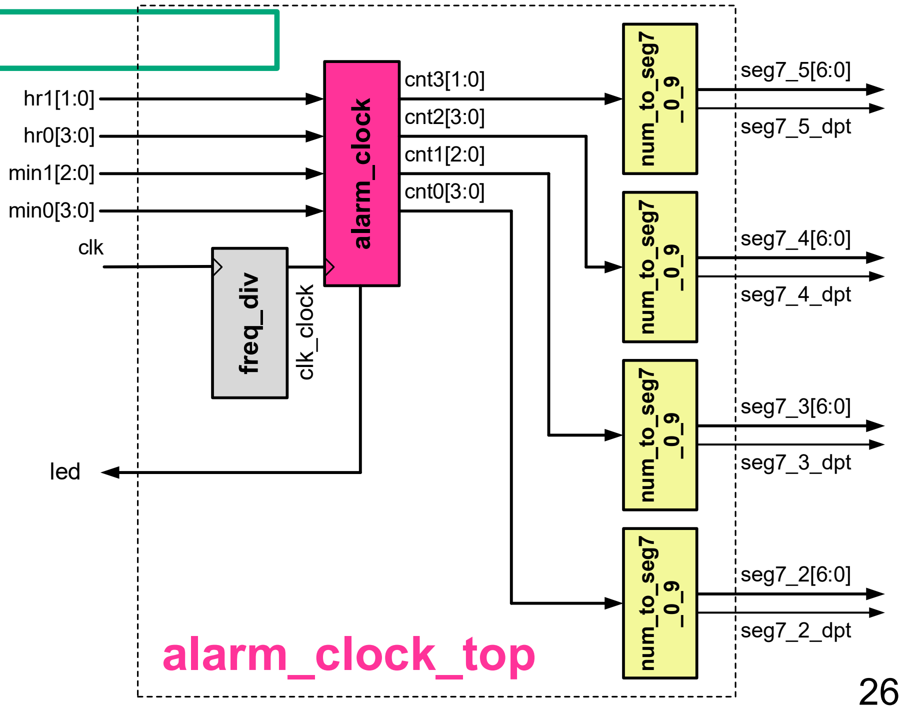


```
output  [6:0] seg7_5, seg7_4, seg7_3, seg7_2;
output  seg7_5_dpt, seg7_4_dpt, seg7_3_dpt, seg7_2_dpt;
output  led;

wire    [1:0] cnt3;
wire    [3:0] cnt2;
wire    [2:0] cnt1;
wire    [3:0] cnt0;

wire    clk_clock;
wire    led;
```

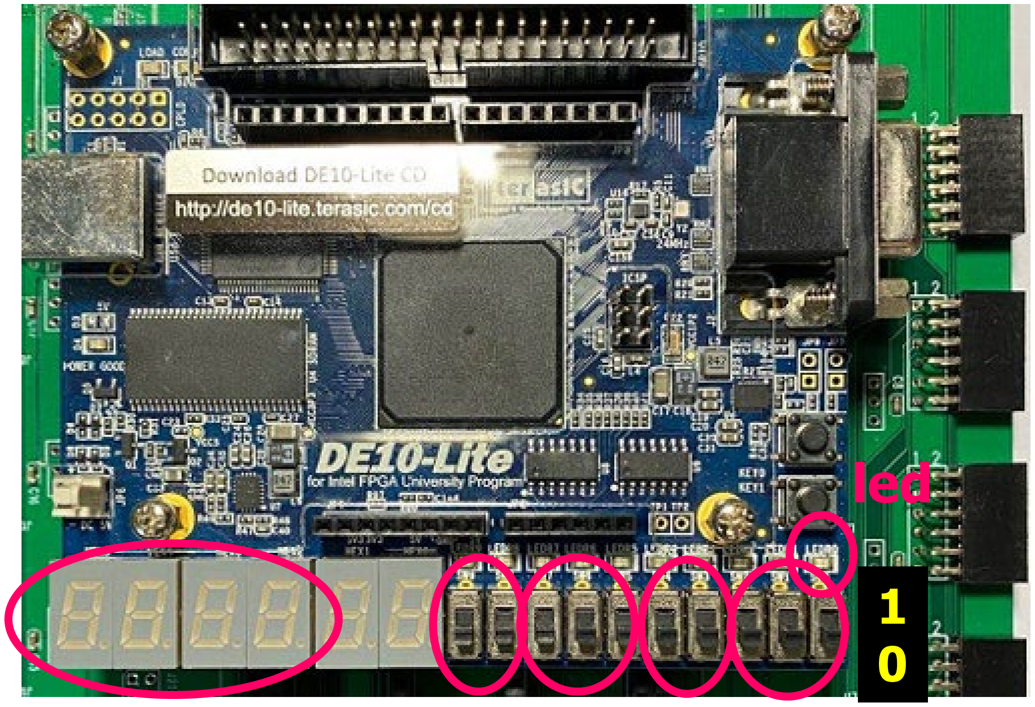
Design 5: alarm_clock_top (3/3)

```
alarm_clock      m0



num_to_seg7_0_9 m1
num_to_seg7_0_9 m2
num_to_seg7_0_9 m3
num_to_seg7_0_9 m4


freq_div #(   )  m5
endmodule
```

rst必須先反相: ~rst



Pin Assignments



hr1[1:0], hr0[2:0], min1[1:0], min0[2:0]

seg7_5, seg7_5_dpt
seg7_4, seg7_4_dpt
seg7_3, seg7_3_dpt
seg7_2, seg7_2_dpt

部分驗證
hr1: 0-2
hr0: 0-7
min1: 0-3
min0: 0-7

clk

Clocks: 50 MHZ
CLK1: P11
CLK2: N14

rst

KEY0: B8	Push: 0
KEY1: A7	Not push: 1

File List

- Verilog files
  - freq_div.v
  - num_to_seg7_0_9.v
  - cnt_00_59_bcd.v
  - cnt_00_23_bcd.v
  - clock.v
  - alarm_clock.v
  - alarm_clock_top.v
- Waveform files
  - cnt_00_59_bcd.vwf
  - cnt_00_23_bcd.vwf
  - clock.vwf
  - alarm_clock.vwf

實驗結果驗收

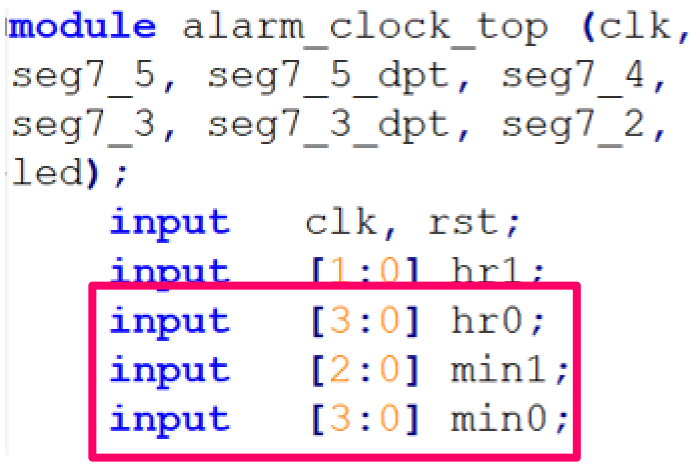
- 修改上述 alarm_clock_top.v 電路，利用led信號控制clk_timer信號，當led為0(暗)時(鬧鐘設定時間未到)，clock以高速計數。而當led為1(亮)時(鬧鐘時間已到)，clock以慢速計數10次(肉眼可觀察之程度)，之後clock恢復高速計數。
- 請老師或助教驗收以下鬧鐘

所設定之時間

(1) 03:31 (2) 06:16

(3) 12:35 (4) 17:27

(5) 20:14 (6) 21:02



```
module alarm_clock_top (clk,
seg7_5, seg7_5_dpt, seg7_4,
seg7_3, seg7_3_dpt, seg7_2,
led);
    input   clk, rst;
    input   [1:0] hr1;
    input   [3:0] hr0;
    input   [2:0] min1;
    input   [3:0] min0;
```

input信號未接switch時，預設值為1
因此hr0, min1, min0的bit數要縮減成switch個數