# 邏輯設計實驗 Lab4 結報

# 105060012 張育菘

**1. Implement pre-lab1**:

**Cascade eight DFFs together as a shift register. Connect the output of the last DFF to the input of the first DFF as a ringer counter. Let the initial value of DFF output after reset be 01010101. Construct the Verilog RTL representation for the logics with verification.**

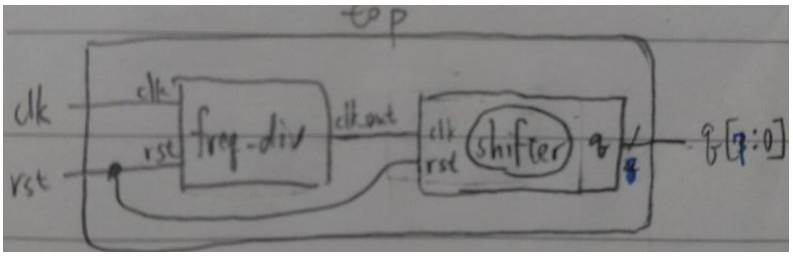
| clk | rst |
|---|---|
| W5 | V17 |

Design Specification

input clk,rst;

output [7:0]q;

block diagram :




Design Implementation

Logic function :

1. top：

   此 top module，專門拿來呼叫其他小 moudule 的。

   並接一內線 clk_out 作為 shifter 的 clk。

2. freq_div：

   此為除頻的 module，用來控制 shifter 的速度。

   always@* cnt_tmp={clk_out,cnt}+1'b1;

   always@(posedge clk or negedge rst)  //DFFs

   if(~rst){clk_out,cnt}<=27'b0; //當 rst=0 時，{clk_out,cnt} = 0(從 0 開始數)

   else{clk_out,cnt}<=cnt_tmp;

   //當 rst=1 & clk 從 0→1 時，把 cnt_tmp 存入{clk_out,cnt}

   週期為 clk 的 $2^{27}$ 倍。

3. shifter：

此作用就是做移位的工作，使所有的 bit 往左移一位，最高位則變成最低位。

此 module 就是同時有 8 的 DFFs 在作用，clk 每從 0→1 時，即移位。

```
always@(posedge clk or negedge rst)
    if(~rst) q<=8'b01010101;    //設初始值為 01010101
    else
    begin
        q[0]<=q[7];
        q[1]<=q[0];
        q[2]<=q[1];
        q[3]<=q[2];
        q[4]<=q[3];
        q[5]<=q[4];
        q[6]<=q[5];
        q[7]<=q[6];
    end
```

Result

1. rst=0→初始化設為 01010101
2. rst=1→LED 燈來回在 01010101 與 10101010 之間轉換顯示。

Discussion

1. 這次實驗教會我如何操作 shifter。

**2. Construct a ring counter similar to that of pre-lab1 but the initial value of the DFFs can be set randomly.**

| clk | rst |
|---|---|
| W5 | V17 |

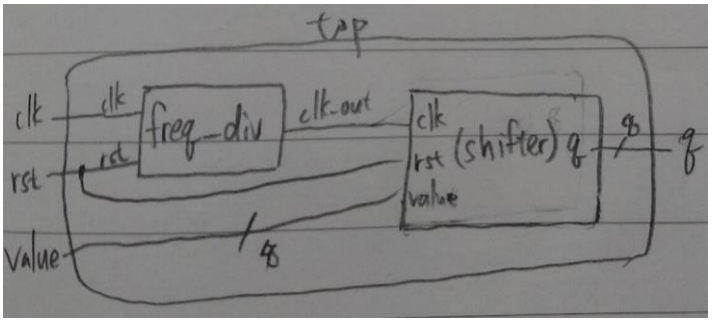
| value[7] | value[6] | value[5] | value[4] | value[3] | value[2] | value[1] | value[0] |
|---|---|---|---|---|---|---|---|
| R2 | T1 | U1 | W2 | R3 | T2 | T3 | V2 |

Design Specification

input : clk,rst;

output : [7:0]value, [7:0]q;

block diagram :




Design Implementation

Logic function :

1. top：

此 top module，專門拿來呼叫其他小 moudule 的。

並接一內線 clk_out 作為 shifter 的 clk。

2. freq_div：

此為除頻的 module，用來控制 shifter 的速度。

```
always@* cnt_tmp={clk_out,cnt}+1'b1;
always@(posedge clk or negedge rst)    //DFFs
        if(~rst){clk_out,cnt}<=27'b0;  //當 rst=0 時，{clk_out,cnt} = 0(從 0 開始數)
        else{clk_out,cnt}<=cnt_tmp;
                //當 rst=1 & clk 從 0→1 時，把 cnt_tmp 存入{clk_out,cnt}
```

週期為 clk 的 $2^{27}$ 倍。

3. shifter：

此作用就是做移位的工作，使所有的 bit 往左移一位，最高位則變成最低位。

此 module 就是同時有 8 的 DFFs 在作用，clk 每從 0→1 時，即移位。

```
always@(posedge clk or negedge rst)
        if(~rst) q<=value;    //初始值為自己輸入的 value 值
```

```
else
begin
        q[0]<=q[7];
        q[1]<=q[0];
        q[2]<=q[1];
        q[3]<=q[2];
        q[4]<=q[3];
        q[5]<=q[4];
        q[6]<=q[5];
        q[7]<=q[6];
end
```

Result

1. 圖 1 & 2 即當 value 值為 01101000 時的情況，每經一個 clk 往昨移 1 個 bit。
2. 圖 3 & 4 即當 value 值為 10000001 時的情況，每經一個 clk 往昨移 1 個 bit。

Discussion

1. 這次實驗教會我如何操作 shifter，並如何自己給定的輸入值。

**3. Use the idea from pre-lab1. We can do something on the seven-segment display. Assume we have the pattern of E, H, N, T, U for seven-segment display as shown below. Try to implement the scrolling pre-stored pattern NTHUEE with the four seven-segment displays.**

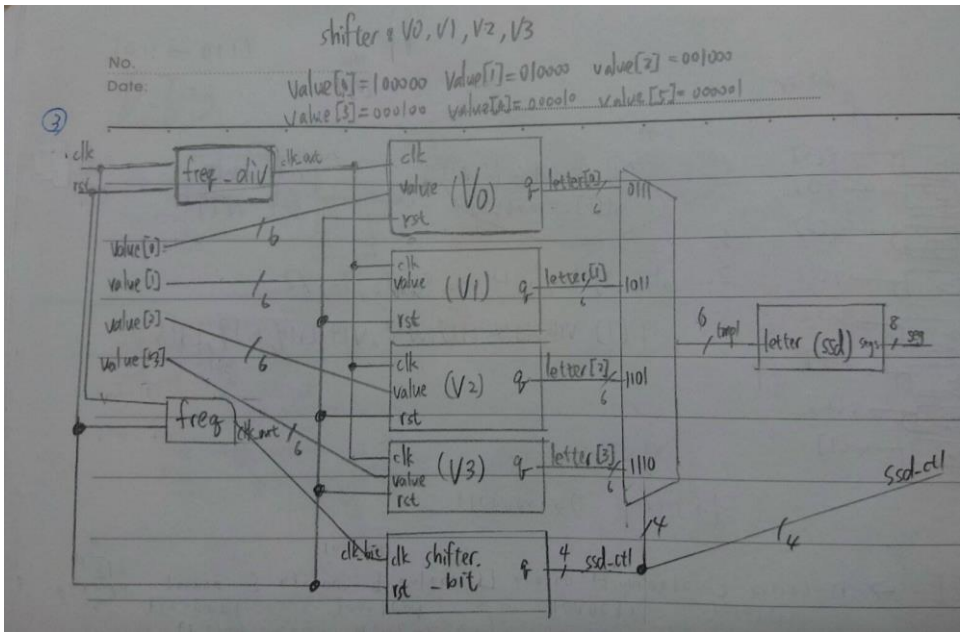
| clk | clk_out |
|---|---|
| W5 | U16 |

Design Specification

input : clk,rst;

output : [3:0]ssd_ctl, [7:0]seg;

block diagram :




Design Implementation

Logic function :

1. top：
   a. 此 top module，專門拿來呼叫其他小 moudule 的，並接一內線 clk_out 作為 shifter 的 clk。
   b. 外加給定 value 值作為 shifter 的 Input。

   value[0] = 100000 ; value[1] = 010000 ; value[2] = 001000 ; value[3] = 000100

   而此在 ssd 內，6-bit 的 one-hot 會分別代表一個字母。
   c. 還有內建一個 mux，用由 shifter_bit 得到的 ssd_ctl 選擇七段顯示器要顯示何種字母。
2. freq_div：

   此為除頻的 module，用來控制 shifter 的速度。

   always@* cnt_tmp={clk_out,cnt}+1'b1;

```
always@(posedge clk or negedge rst)    //DFFs
    if(~rst){clk_out,cnt}<=27'b0;  //當 rst=0 時，{clk_out,cnt} = 0(從 0 開始數)
    else{clk_out,cnt}<=cnt_tmp;
        //當 rst=1 & clk 從 0→1 時，把 cnt_tmp 存入{clk_out,cnt}
```

週期為 clk 的 $2^{27}$ 倍。

3. freq：

此為除頻的 module，用來控制 shifter_bit 的速度。

```
always@* cnt_tmp={clk_out,cnt}+1'b1;
always@(posedge clk or negedge rst)    //DFFs
    if(~rst){clk_out,cnt}<=16'b0;  //當 rst=0 時，{clk_out,cnt} = 0(從 0 開始數)
    else{clk_out,cnt}<=cnt_tmp;
        //當 rst=1 & clk 從 0→1 時，把 cnt_tmp 存入{clk_out,cnt}
```

週期為 clk 的 $2^{16}$ 倍。

4. shifter：

此作用就是做移位的工作，使所有的 bit 往左移一位，最高位則變成最低位。

此 module 就是同時有 6 的 DFFs 在作用，clk 每從 0→1 時，即移位。

```
always@(posedge clk or negedge rst)
        if(~rst) q<=value;    //此為在 top 給定的初始值
        else
        begin
            q <= q>>1'b1;    //每經一個 clk，q 往右移一個 bit
            q[5] <= q[0];
                //ex. 010000→001000
       End
```

5. shifter_bit：

此作用就是做移位的工作，使所有的 bit 往左移一位，最高位則變成最低位。

此 module 就是同時有 4 的 DFFs 在作用，clk 每從 0→1 時，即移位。

而此最重要的功能在於決定何時要顯示哪一個七段顯示器，因此會接上除頻後週期為 clk 的 $2^{16}$ 倍的時間做為自己的 clk，作用在於用即快速的交換顯示，使人眼誤以為是同時顯示的，因為七段顯示器一次只能顯示一種圖案，所以才要做這種處理。

```
always@(posedge clk or negedge rst)
    if(~rst) q<= 4'b1110;    //一開始只顯示最右邊的七段顯示器
    else
    begin
        q <= q<<1'b1;    ////每經一個 clk，q 往左移一個 bit
        q[0] <= q[3];
   end
```

6. ssd：

```
當 letter 為 6'd32: segs = 8'b11010101;   //七段顯示器顯示 n
當 letter 為 6'd16: segs = 8'b11100001;   //七段顯示器顯示 t
當 letter 為 6'd8: segs = 8'b10010001;    //七段顯示器顯示 H
當 letter 為 6'd4: segs = 8'b10000011;    //七段顯示器顯示 U
當 letter 為 6'd2: segs = 8'b01100001;    //七段顯示器顯示 E
當 letter 為 6'd1: segs = 8'b01100001;    //七段顯示器顯示 E
default: segs = 8'b00000000;              //七段顯示器顯示 8.
```

Result

1. rst=0(圖一)→其初始會顯示為”ntHU”，在只顯示最右邊的七段顯示器情況下只會顯示”U”。
2. rst=1(圖二&圖三)→為其左移的情形。

圖一

圖二

圖三

Discussion

1. 此設計想法是讓每一個七段顯示器各自擁有自己的shifter，只是初始值不同，使人眼產生字母是往左移的錯覺。
2. 可以用”>>”的方式代表位移，不用一個一個打出來

   ex. q <= q>>1'b1 ; q[5] <= q[0] ; 即可代表右移一個 bit 的情形。

**4. (Bonus) Display 1010 in the seven-segment display. Use the DIP switch (one bit to indicate left/right shift, three bits with one hot to display the kind of shift operation) as the control input to implement the functional/arithmetic/barrel shifter. Use one push button to control the display of the number before/after the shift operation.**

| clk | clk_out | push |
|---|---|---|
| W5 | U16 | T17 |

| en[3] | en[2] | en[1] | en[0] |
|---|---|---|---|
| L3 | P1 | N3 | P3 |

Design Specification

input : clk,rst;

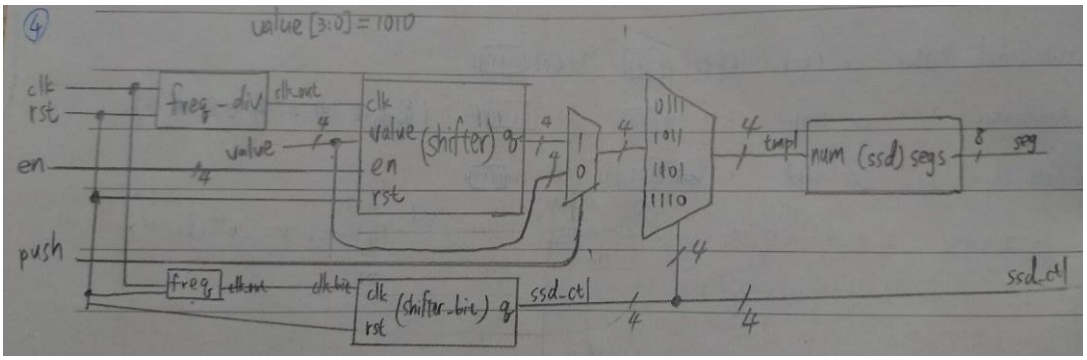
input [3:0]en;  //DIP //en[0]控制往左往右移;en[1]為 functional shifter

// en[2]為 arithmetic shifter ; en[3]為 barrel shifter

input :push;  //按鈕

output : [3:0]ssd_ctl, [7:0]seg;

block diagram :




Design Implementation

Logic function :

1. top：
   a. 此 top module，專門拿來呼叫其他小 moudule 的，並接一內線 clk_out 作為 shifter 的 clk。
   b. 外加給定 value 值作為 shifter 的 Input。value = 1010。
   c. 還有內建二個 mux，第一個用由 shifter_bit 得到的 ssd_ctl 選擇七段顯示器要顯示何種字母；第二個當 push = 1(按下按鈕)七段顯示器顯示正在運作的值，push = 0 七段顯示器顯示初始值 1010，
2. freq_div：

   此為除頻的 module，用來控制 shifter 的速度。

   always@* cnt_tmp={clk_out,cnt}+1'b1;

```
always@(posedge clk or negedge rst)    //DFFs
    if(~rst){clk_out,cnt}<=27'b0;  //當 rst=0 時，{clk_out,cnt} = 0(從 0 開始數)
    else{clk_out,cnt}<=cnt_tmp;
        //當 rst=1 & clk 從 0→1 時，把 cnt_tmp 存入{clk_out,cnt}
```

週期為 clk 的 $2^{27}$ 倍。

3. freq：

此為除頻的 module，用來控制 shifter_bit 的速度。

```
always@* cnt_tmp={clk_out,cnt}+1'b1;
always@(posedge clk or negedge rst)    //DFFs
    if(~rst){clk_out,cnt}<=16'b0;  //當 rst=0 時，{clk_out,cnt} = 0(從 0 開始數)
    else{clk_out,cnt}<=cnt_tmp;
        //當 rst=1 & clk 從 0→1 時，把 cnt_tmp 存入{clk_out,cnt}
```

週期為 clk 的 $2^{16}$ 倍。

4. shifter：

此作用就是做移位的工作，並用 mux 決定要產生何種功能。

```
//en[0] = 1 表示往右移; en[0] = 0 表示往左移
always@(posedge clk or negedge rst)
    if(~rst) q <= value;
    else
    begin
//en[3] = 1 做 barrel shifter。當往左移時，最低位補最高位的數；往右移時，最高位補最低位的數
        if(en==4'b1001)
        begin
            q <= q>>1'b1;
            q[3] <= q[0];
        end
        else if(en==4'b0001)
        begin
            q <= q<<1'b1;
            q[0] <= q[3];
        end
//en[2] = 1 做 arithmetic shifter。當往左移時，最低位補”0”；往右移時，最高位補最高位的數
        else if(en==4'b1010)
        begin
            q <= q>>1'b1;
            q[3] <= q[3];
```

```
            end
            else if(en==4'b0010)
            begin
                q <= q<<1'b1;
                q[0] <= 0;
            end
```
//en[1] = 1 做 functional shifter。當往左移時，最低位補”0”；往右移時，最高位補”0”
```
            else if(en==4'b1100)
            begin
                q <= q>>1'b1;
                q[3] <= 0;
            end
            else if(en==4'b0100)
            begin
                q <= q<<1'b1;
                q[0] <= 0;
            end
            else
            begin
                q <= q;
            end
        end
```
5. shifter_bit：

   此作用就是做移位的工作，使所有的 bit 往左移一位，最高位則變成最低位。此 module 就是同時有 4 的 DFFs 在作用，clk 每從 0→1 時，即移位。

   而此最重要的功能在於決定何時要顯示哪一個七段顯示器，因此會接上除頻後週期為 clk 的 $2^{16}$ 倍的時間做為自己的 clk，作用在於用即快速的交換顯示，使人眼誤以為是同時顯示的，因為七段顯示器一次只能顯示一種圖案，所以才要做這種處理。

```
 always@(posedge clk or negedge rst)
      if(~rst) q<= 4'b1110;    //一開始只顯示最右邊的七段顯示器
      else
      begin
          q <= q<<1'b1;    ////每經一個 clk，q 往左移一個 bit
          q[0] <= q[3];
      end
```
6. ssd：

```
always@*
    if(num==1'b0) segs = 8'b00000011; //當 num 為 0 七段顯示器顯示 0
    else segs = 8'b10011111;          //當 num 為 1 七段顯示器顯示 1
```

Result(顯示器只有第 3 個數字亮)

1. rst=0(圖一)→其初始會顯示為”1010”，在只顯示最右邊的七段顯示器情況下只會顯示”0”。
2. 圖二：當 push=0 時(未按下按鈕)，不管怎麼開開關，仍然顯示初始值”1010”。
3. 圖三、四、五：當 en[1] 為 1 時做 functional shifter，當往左移時(en[0]=0)，最低位補”0”；往右移時(en[0]=1)，最高位補”0”。
4. 圖六、七：當 en[2] 為 1 時做 arithmetic shifter，當往左移時，最低位補”0”；往右移時，最高位補最高位的數。
5. 圖八：當 en[3]為 1 時做 barrel shifter，當往左移時，最低位補最高位的數；往右移時，最高位補最低位的數。

圖一

圖二

圖三

圖四

圖五

圖六

圖七

圖八

Discussion

1. functional shifter：當往左移時(en[0]=0)，最低位補”0”；往右移時(en[0]=1)，最高位補”0”。
2. arithmetic shifter：當往左移時，最低位補”0”；往右移時，最高位補最高位的數。

3. barrel shifter，當往左移時，最低位補最高位的數；往右移時，最高位補最低位的數。

**Conclusion**：

這次還蠻好玩的，感覺可以把上次 Lab3 學到的東西做應用，還可以外加學一些功能，像是跑馬燈其實是每個七段顯示器內部自己在做循環顯示而已，真的還蠻實用的。

Reference：

馬席彬教授的上課講義