(一)

Design Specification

(1) input & output

main module:

```
input clk,rst; //global clk,rst
output [3:0] q; //作為 led 的輸出
wire clk_d; // F divider 產生的 clk
```

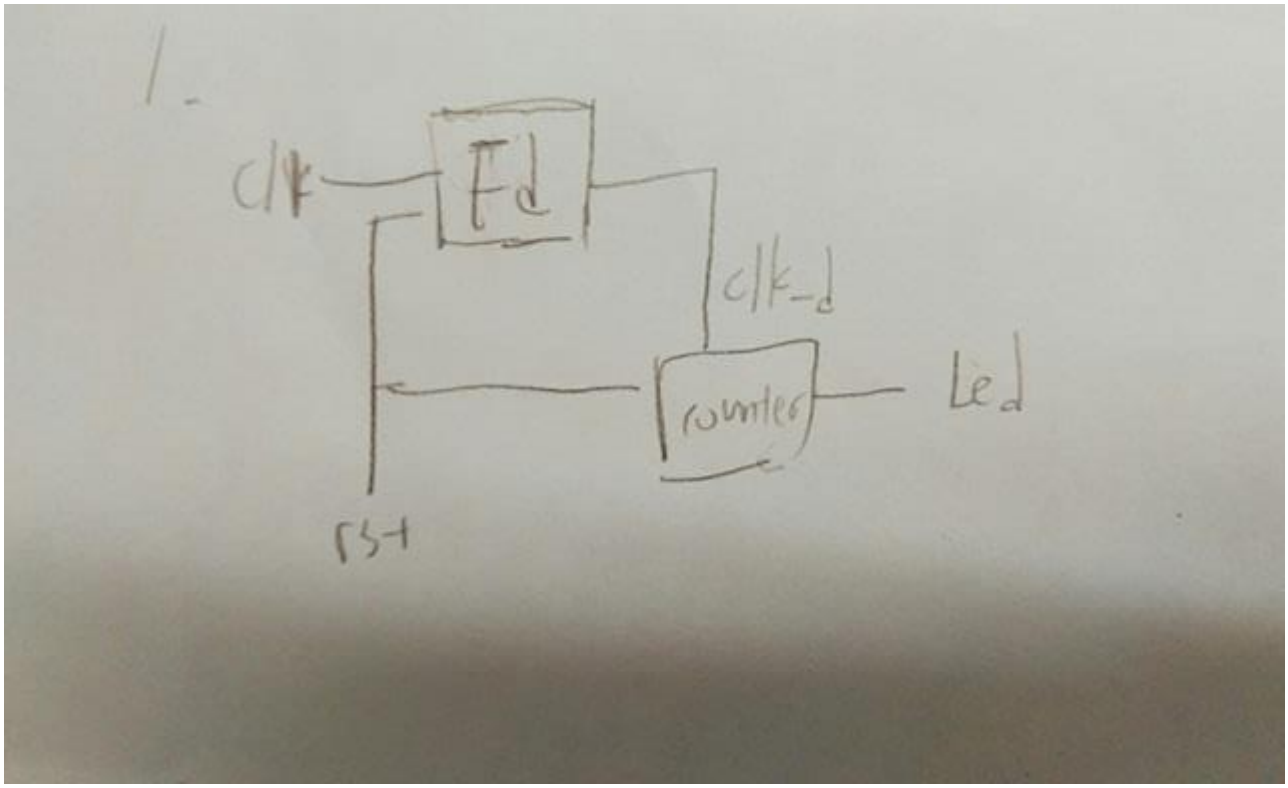
module fd:

```
input clk;
input rst;
output reg clk_out; // output
reg s;              // if(q == 50m) s = 1 else s = 0
reg [26:0]q;    //  計數
```

module counter:

```
input clk;
input rst;
output reg [3:0]out;
```

(2) block diagram



```
input clk;
input rst;
output [3:0]b;
wire clk_d;

fd U0(.clk(clk), .rst(rst), .clk_out(clk_d));
counter U1(.clk(clk_d), .rst(rst), .out(b));
```

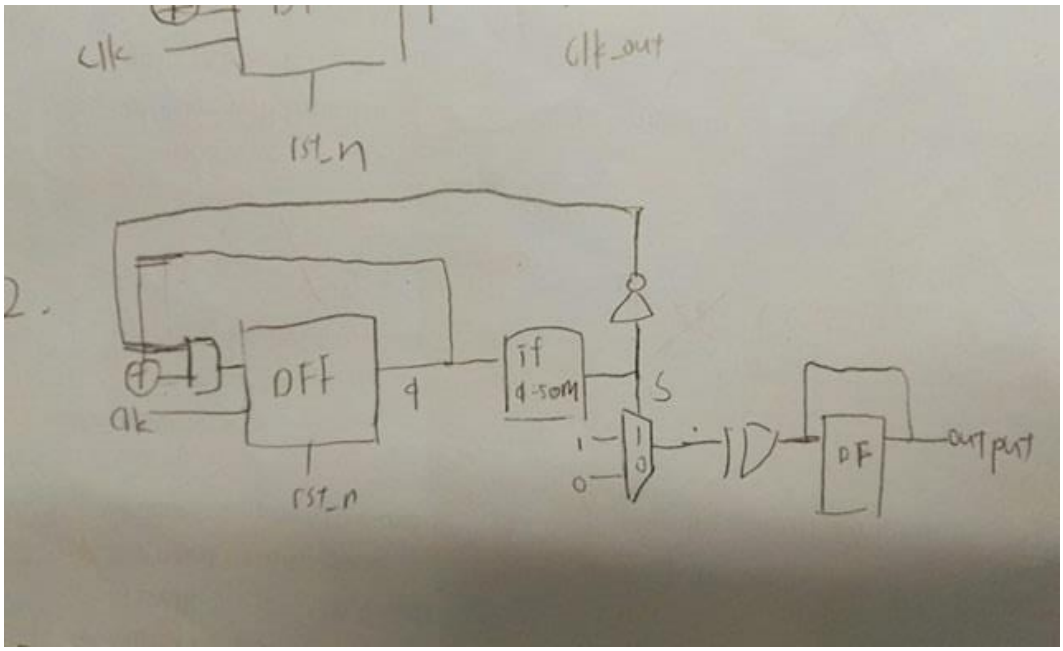
Design Implementation

(1) I/O PIN

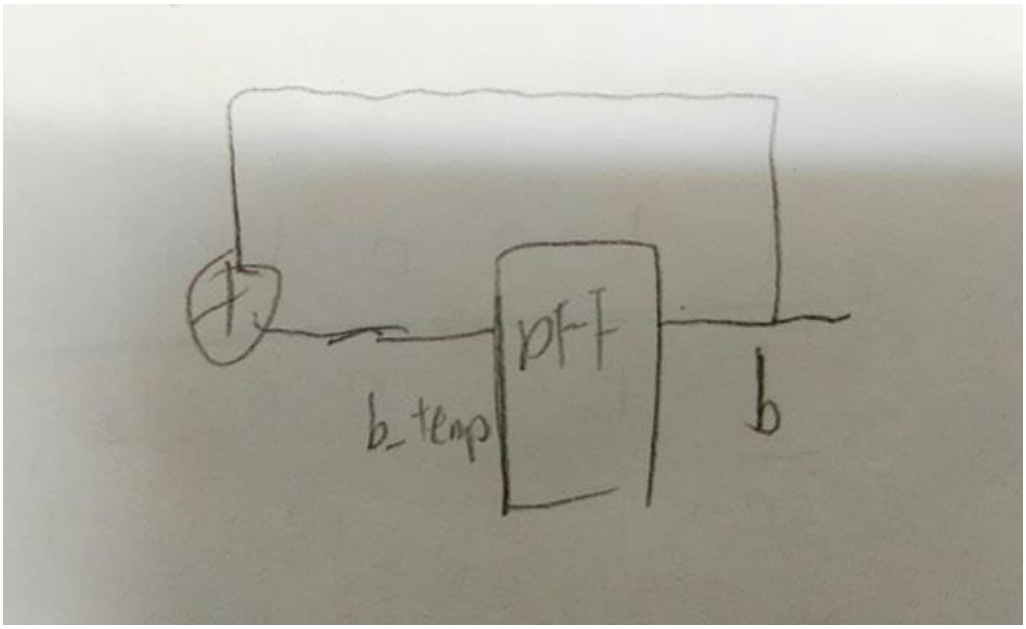
set_property PACKAGE_PIN V19 [get_ports {b[3]}]
set_property PACKAGE_PIN U19 [get_ports {b[2]}]
set_property PACKAGE_PIN E19 [get_ports {b[1]}]
set_property PACKAGE_PIN U16 [get_ports {b[0]}]
set_property PACKAGE_PIN W5 [get_ports clk]
set_property PACKAGE_PIN V17 [get_ports rst]

(2) verilog code

```
module fd(clk, rst, clk_out);
    input clk;
    input rst;
    output reg clk_out;
    reg s;
    reg [26:0]q;

    always@(posedge clk or negedge rst)
        if (~rst || s == 1) q <= 0;
        else q <= q + 1;

    always @*
        if (q == 27'd50000000) s = 1;
        else s = 0;

    always@(posedge clk or negedge rst)
        if (~rst) clk_out <= 0;
        else clk_out <= clk_out ^ s;

endmodule
```




```
module counter(clk, rst, out);
    input clk;
    input rst;
    output reg [3:0]out;

    always@(posedge clk or negedge rst)
        if (~rst) out <= 0;
        else out = out + 1;
endmodule
```



DISSCUSSION

這次的想法很簡單，就只是 1 個 f divider 接一個 counter，基本上把以前打過的 module 接一接就完成了。實驗結果就是 4 個 led 燈個當 b 的 4 個 bit，然後每秒都會看到 led 燈 output 的結果+1。

CONCLUSION

這個 lab 讓我對模組之間的連接更熟悉

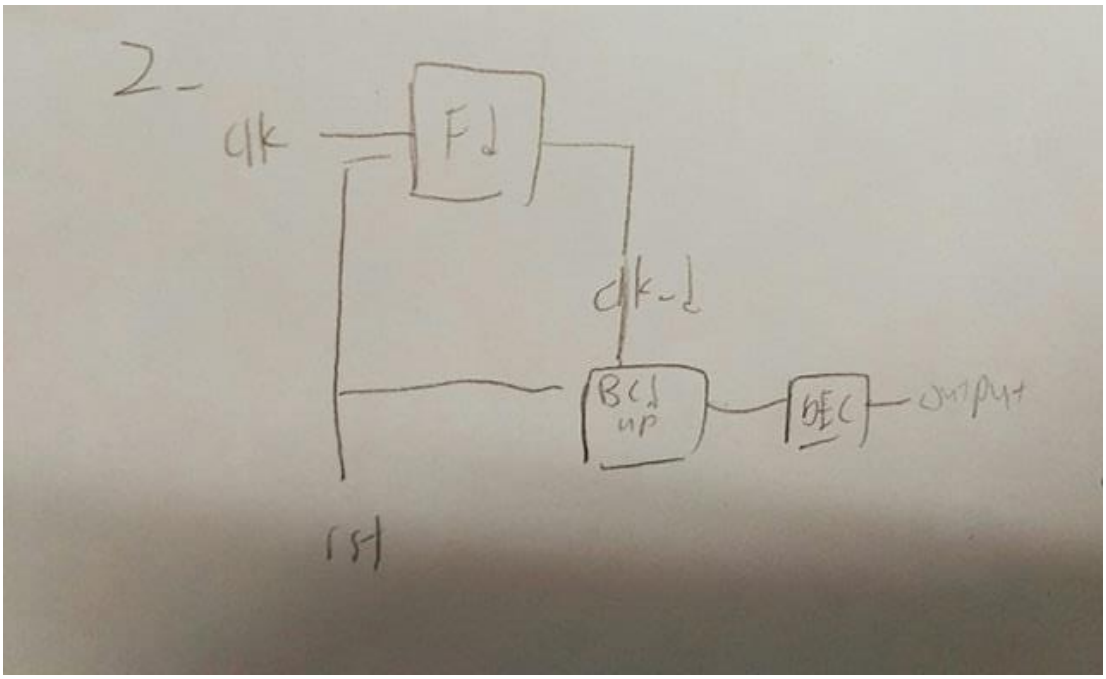
(二)

Design Specification

(1)

```
module main(clk, rst, light, ssd);
input clk;
input rst;
output [3:0]light;    // light of seven segment display
output [7:0]ssd;    // seven segment display
wire [3:0]value;    // the counting number
wire clk_d;          // clock generated by f divider

module fd(clk, rst, clk_out);
reg [26:0]q;
input clk;
input rst;
output clk_out;

module BCD_UP(clk, rst, q);
input clk;
input rst;
output reg [3:0]q;
reg [3:0]q_temp;

module DEC(in, out);
input [3:0]in;
output reg [7:0]out;
```

(2) Block diagram




```
fd U0(.clk(clk), .rst(rst), .clk_out(clk_d));
BCD_UP U1(.clk(clk_d), .rst(rst), .q(value));
DEC U2(.in(value), .out(ssd));
assign light = 4'b1110;
```

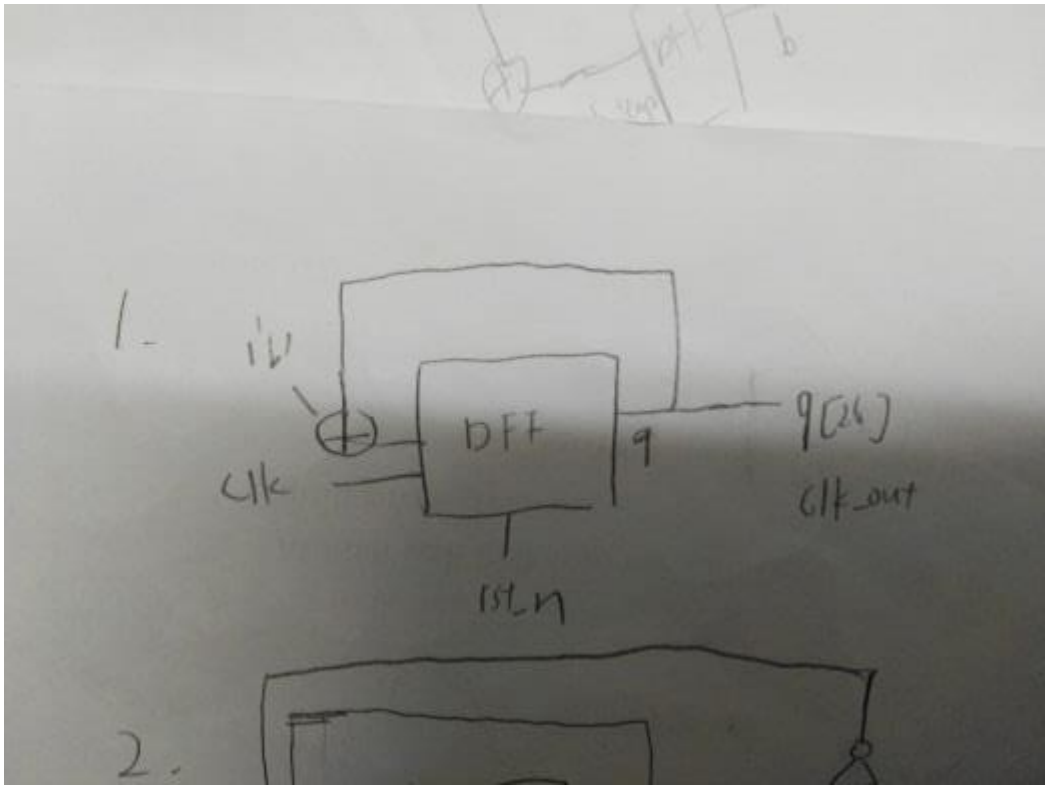
DESIGN IMPLEMENTATION

(1) I/O PINS

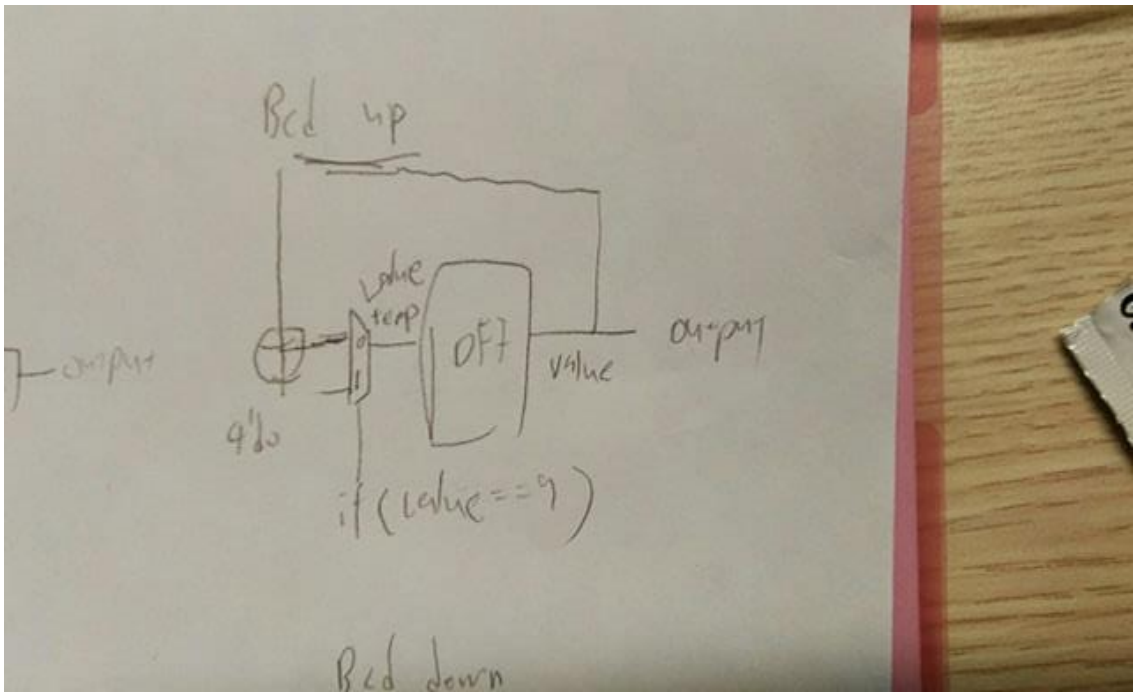
set_property PACKAGE_PIN W4 [get_ports {light[3]}]
set_property PACKAGE_PIN V4 [get_ports {light[2]}]
set_property PACKAGE_PIN U4 [get_ports {light[1]}]
set_property PACKAGE_PIN U2 [get_ports {light[0]}]
set_property PACKAGE_PIN W7 [get_ports {ssd[7]}]
set_property PACKAGE_PIN W6 [get_ports {ssd[6]}]
set_property PACKAGE_PIN U8 [get_ports {ssd[5]}]
set_property PACKAGE_PIN V8 [get_ports {ssd[4]}]
set_property PACKAGE_PIN U5 [get_ports {ssd[3]}]
set_property PACKAGE_PIN V5 [get_ports {ssd[2]}]
set_property PACKAGE_PIN U7 [get_ports {ssd[1]}]
set_property PACKAGE_PIN V7 [get_ports {ssd[0]}]
set_property PACKAGE_PIN W5 [get_ports clk]
set_property PACKAGE_PIN V17 [get_ports rst]

(2) Verilog code

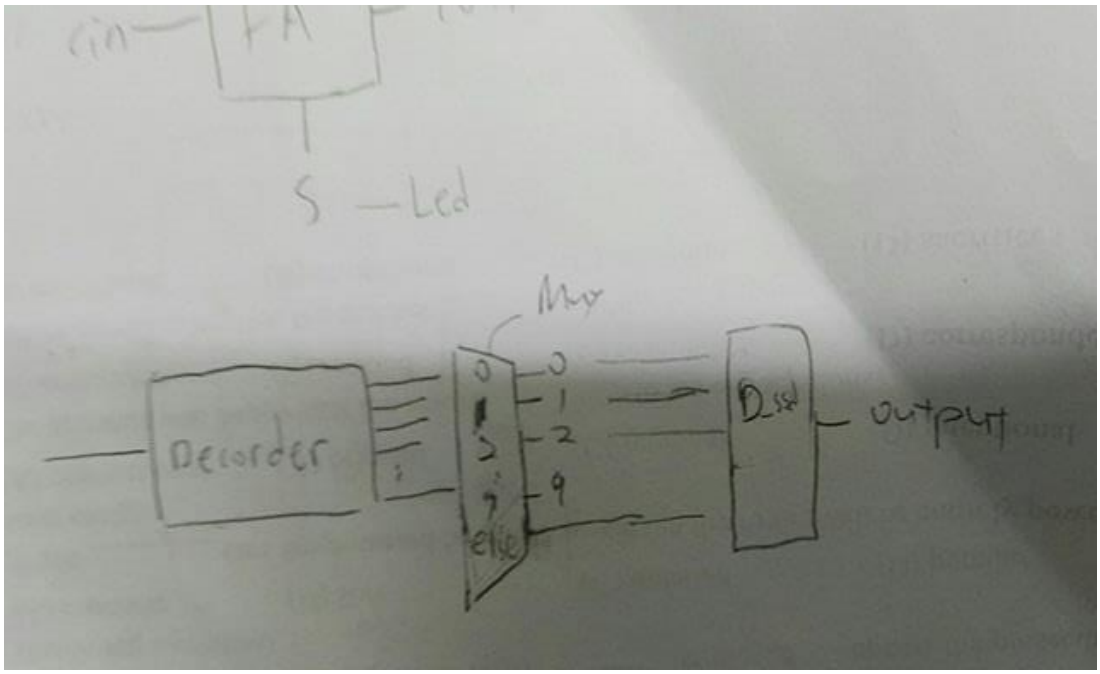
```
module fd(clk, rst, clk_out);
    reg [26:0]q;
    input clk;
    input rst;
    output clk_out;

    always @(posedge clk or negedge rst)
        if (~rst) q <= 0;
        else q <= q + 1'b1;

    assign clk_out = q[26];

endmodule
```



```
module BCD_UP(clk, rst, q);
    input clk;
    input rst;
    output reg [3:0]q;
    reg [3:0]q_temp;

    always @*
        if (q == 4'b1001) q_temp = 4'b0000;
        else q_temp = q + 1'b1;

    always @(posedge clk or negedge rst)
        if (~rst) q <= 4'b0000;
        else q <= q_temp;

endmodule
```



```
module DEC(in, out);
    input [3:0]in;
    output reg [7:0]out;

    always @*
        case(in)
        4'd0: out = `SS_0;
        4'd1: out = `SS_1;
        4'd2: out = `SS_2;
        4'd3: out = `SS_3;
        4'd4: out = `SS_4;
        4'd5: out = `SS_5;
        4'd6: out = `SS_6;
        4'd7: out = `SS_7;
        4'd8: out = `SS_8;
        4'd9: out = `SS_9;
        default: out = 8'b11111111;
        endcase
endmodule
```



DISCUSSION

這題的想法其實不難，因為我們已經做過 1HZ 的 f divider，跟這題十分相似，只是我 bcd_up module 那裏忘記宣告成 3-bit，導致我跳出來只有 0 跟 1。

Output 的結果就是從 0 數到 9 再跳回 0。

CONCLUSION

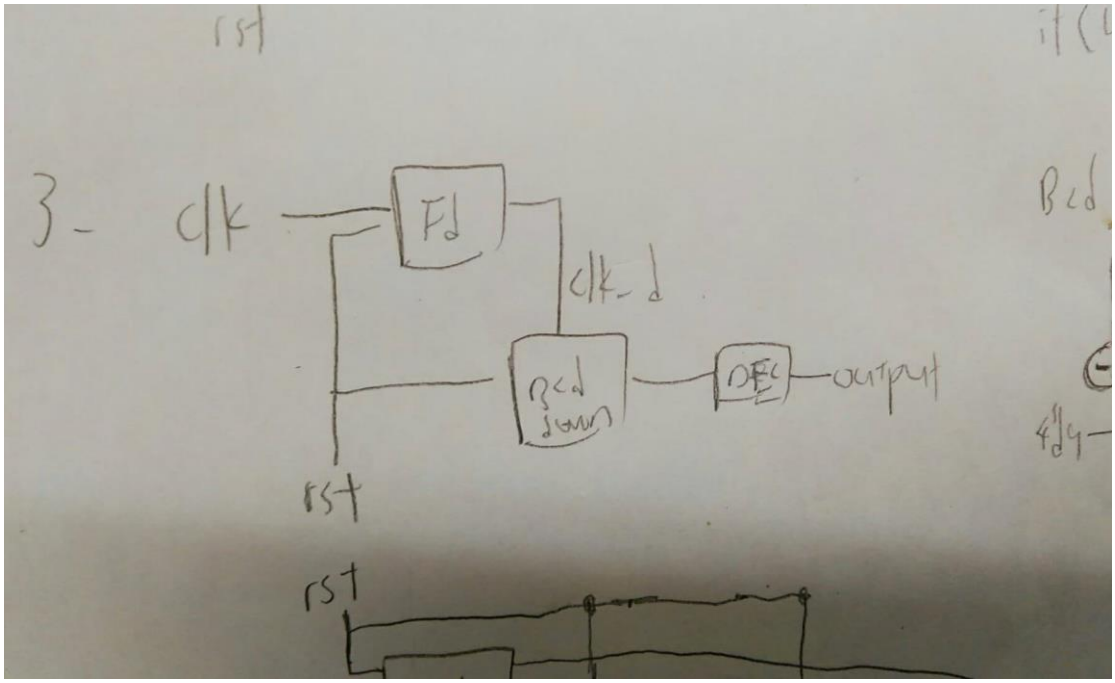
這題讓我對 VERILOG 更熟悉

(三)

Design Specification

```
module main(clk, rst, light, ssd);
        input clk;
        input rst;
        output [3:0]light;
        output [7:0]ssd;
        wire [3:0]value;
        wire clk_d;

module BCD_DOWN(clk, rst, q);
        input clk;
        input rst;
        output reg [3:0]q;
        reg [3:0]q_temp;

module DEC(in, out);      // decorder
        input [3:0]in;
        output reg [7:0]out;
```

block diagram:



```
fd U0(.clk(clk), .rst(rst), .clk_out(clk_d));
BCD_DOWN U1(.clk(clk_d), .rst(rst), .q(value));
DEC U2(.in(value), .out(ssd));
assign light = 4'b1110;
```

DESIGN IMPLEMENTATION

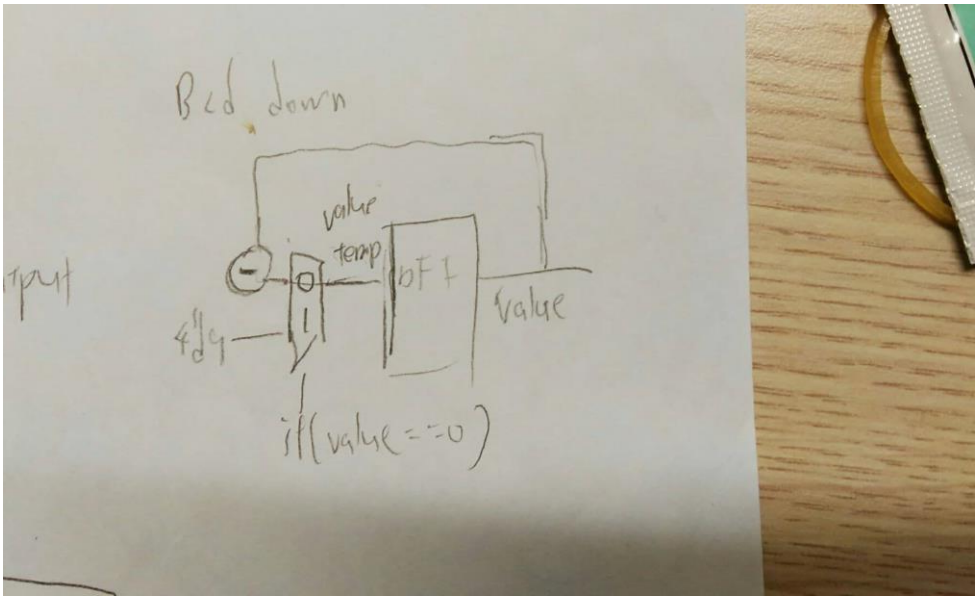
(1) I / O PINS

set_property PACKAGE_PIN W4 [get_ports {light[3]}]
set_property PACKAGE_PIN V4 [get_ports {light[2]}]
set_property PACKAGE_PIN U4 [get_ports {light[1]}]
set_property PACKAGE_PIN U2 [get_ports {light[0]}]
set_property PACKAGE_PIN W7 [get_ports {ssd[7]}]
set_property PACKAGE_PIN W6 [get_ports {ssd[6]}]
set_property PACKAGE_PIN U8 [get_ports {ssd[5]}]
set_property PACKAGE_PIN V8 [get_ports {ssd[4]}]
set_property PACKAGE_PIN U5 [get_ports {ssd[3]}]
set_property PACKAGE_PIN V5 [get_ports {ssd[2]}]
set_property PACKAGE_PIN U7 [get_ports {ssd[1]}]
set_property PACKAGE_PIN V7 [get_ports {ssd[0]}]
set_property PACKAGE_PIN W5 [get_ports clk]
set_property PACKAGE_PIN V17 [get_ports rst]

(2) CODE

Code is same as 4-2

Only bcd_down is different



```
module BCD_DOWN(clk, rst, q);
    input clk;
    input rst;
    output reg [3:0]q;
    reg [3:0]q_temp;

    always @*
        if (q == 4'b0000) q_temp = 4'b1001;
        else q_temp = q - 1'b1;

    always @(posedge clk or negedge rst)
        if (~rst) q <= 4'b0000;
        else q <= q_temp;

endmodule
```

Discussion

這題跟第 2 提根本一樣，我是直接複製貼上，再把 bcd_down 裡面+改成減，然後數到 0 時，重製成 9

結果就是從 9 數到 0 再跑回 9 重數

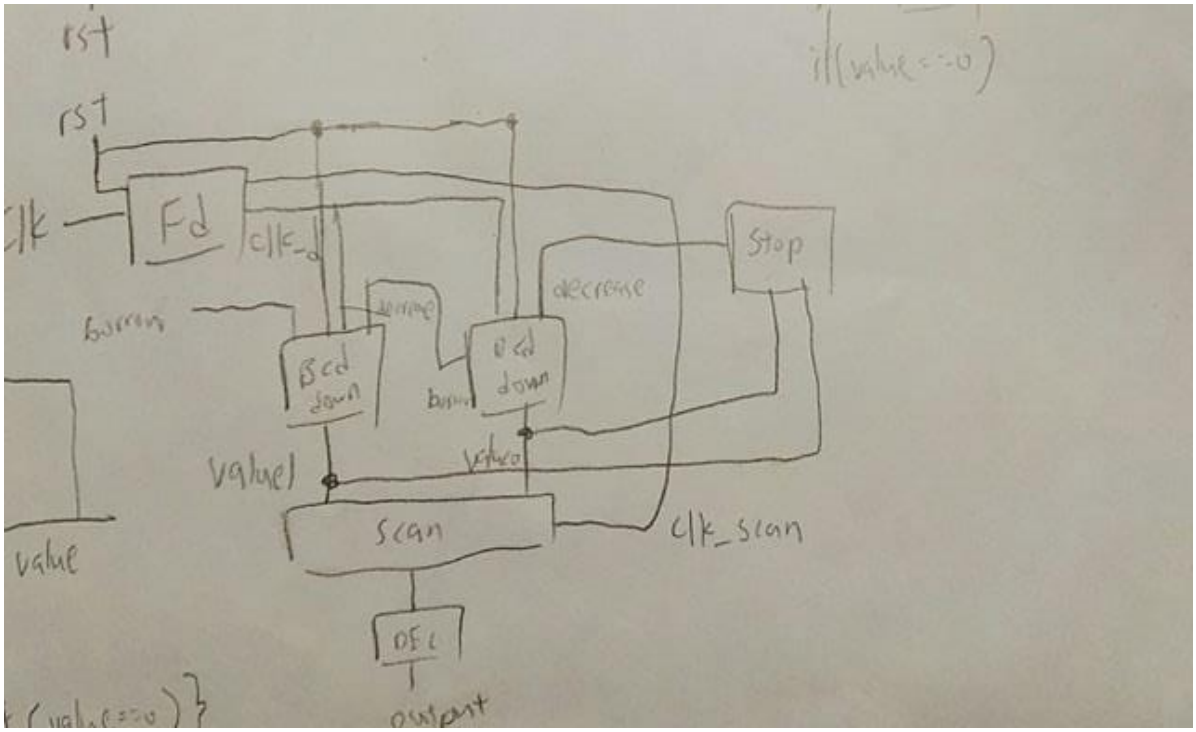
(四)

Design Specification

```
module main(clk, rst, light, ssd);
    input clk;
    input rst;
    output [3:0]light;
    output [7:0]ssd;

    wire enable;    // count enable
    wire [3:0]value0;    //  個位
    wire [3:0]value1;    //  十位
    wire clk_d;          // clock of fd
    wire clk_scan;       // clock of scan control
    wire [1:0]borrow;    //  借位
    wire [3:0]num;       //  要 input  到 decorder 的數字
```

```
module FD(clk, rst, clk_out, clk_out2);
	input clk;
	input rst;
	output reg clk_out ;    // clock for fd
	output clk_out2;        // clock for scan control
	reg s;

	reg [26:0]Q;            // 記數
	reg [26:0]Q_temp;

module BCD_DOWN(limit, clk, rst, decrease, q, borrow);
	input [3:0]limit;    // 初始值
	input clk;
	input rst;
	input decrease;      // 判斷要不要減
	output reg [3:0]q;   // output 值
	output reg borrow;    // 判斷要不要借位
	reg [3:0]q_temp;

module SCAN(control, rst, in0, in1, light, out);
	input control;
	input rst;
	input [3:0]in0;
	input [3:0]in1;
	output reg [3:0]light;
	output reg [3:0]out;

module DEC(in, out);
	input [3:0]in;
	output reg [7:0]out;

module STOP(in1, in2, out);
	input [3:0]in1;        // 個位
	input [3:0]in2;        // 十位
	output reg out;
```

block diagram




```
FD U0(.clk(clk), .rst(rst), .clk_out(clk_d), .clk_out2(clk_scan));
STOP U3(.in1(value1), .in2(value0), .out(enable));
BCD_DOWN digit1(.limit(4'b0000), .clk(clk_d), .rst(rst), .decrease(enable), .q(value0), .borrow(borrow[0]));
BCD_DOWN digit2(.limit(4'b0011), .clk(clk_d), .rst(rst), .decrease(borrow[0]), .q(value1), .borrow(borrow[1]));
SCAN U1(.control(clk_scan), .rst(rst), .in0(value0), .in1(value1), .light(light), .out(num));
DEC U2(.in(num), .out(ssd));

module
```

Design implementation

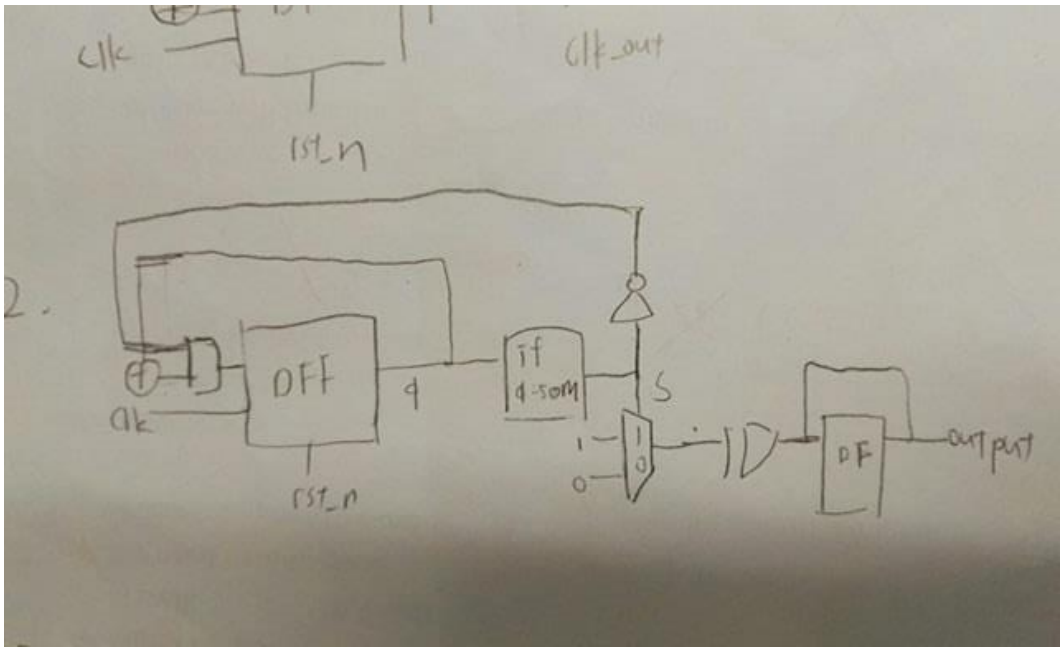
(1) I / O PINS

set_property PACKAGE_PIN W4 [get_ports {light[3]}]
set_property PACKAGE_PIN V4 [get_ports {light[2]}]
set_property PACKAGE_PIN U4 [get_ports {light[1]}]
set_property PACKAGE_PIN U2 [get_ports {light[0]}]
set_property PACKAGE_PIN W7 [get_ports {ssd[7]}]
set_property PACKAGE_PIN W6 [get_ports {ssd[6]}]
set_property PACKAGE_PIN U8 [get_ports {ssd[5]}]
set_property PACKAGE_PIN V8 [get_ports {ssd[4]}]
set_property PACKAGE_PIN U5 [get_ports {ssd[3]}]
set_property PACKAGE_PIN V5 [get_ports {ssd[2]}]
set_property PACKAGE_PIN U7 [get_ports {ssd[1]}]
set_property PACKAGE_PIN V7 [get_ports {ssd[0]}]

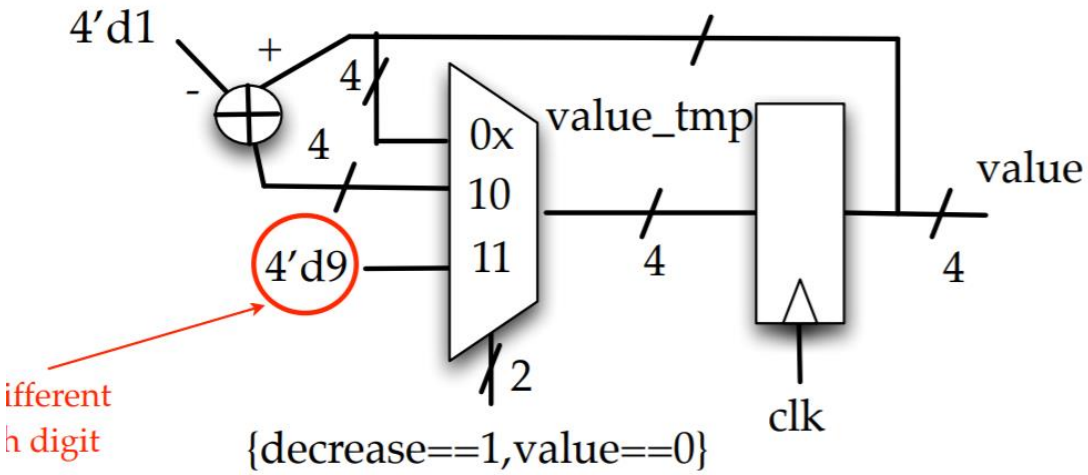
set_property PACKAGE_PIN W5 [get_ports clk]
set_property PACKAGE_PIN V17 [get_ports rst]

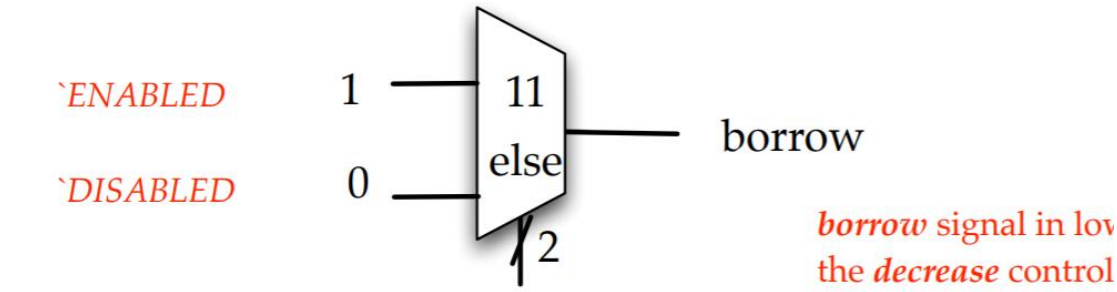
(2) CODE

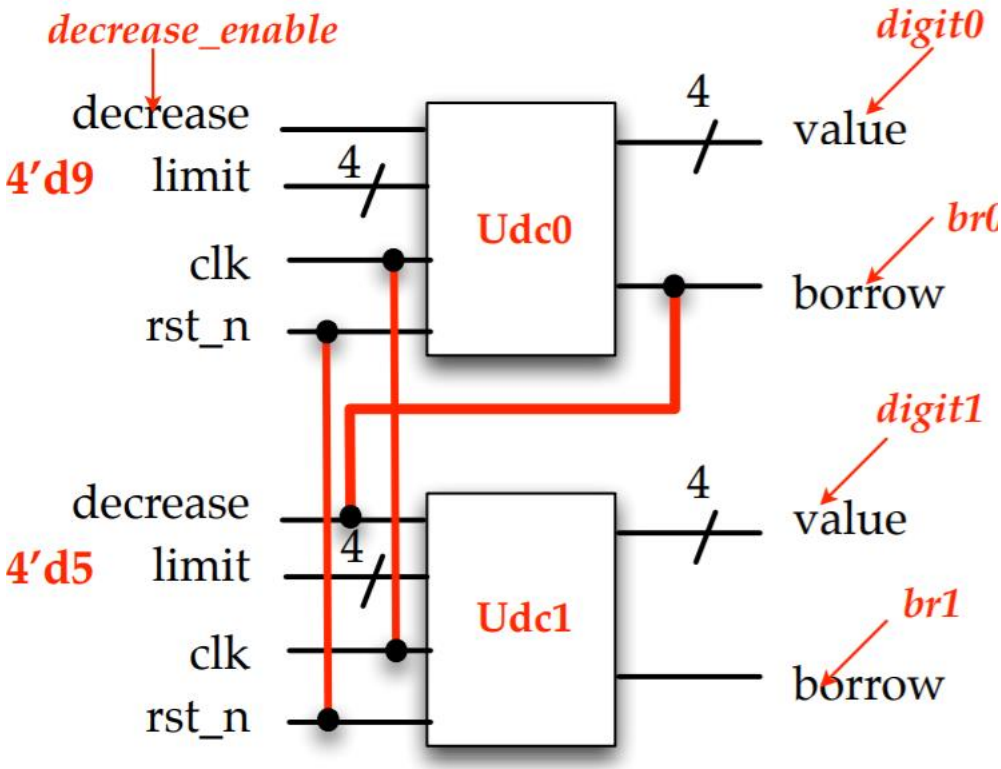
```
module FD(clk, rst, clk_out, clk_out2);
    input clk;
    input rst;
    output reg clk_out = 0;
    output clk_out2;
    reg s;

    reg [26:0]Q;
    reg [26:0]Q_temp;

    always @*
        Q_temp = Q + 1'b1;
    always @(posedge clk or negedge rst)
        if (~rst || s == 1) Q <= 27'd0;
        else Q <= Q_temp;

    always @(posedge clk or negedge rst)
        if (~rst)
            clk_out <= 0;
        else
            clk_out <= clk_out ^ s;

    always @*
        if (Q == 27'd50000000 - 1) begin
            s = 1;
        end
        else begin
            s = 0;
        end
    assign clk_out2 = Q[14];
endmodule
```



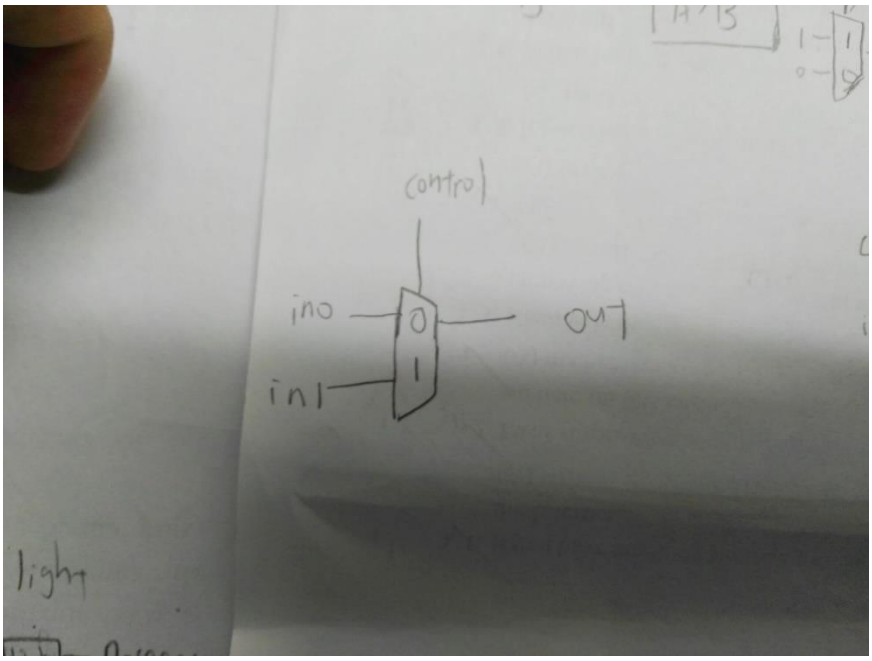

```
module BCD_DOWN(limit, clk, rst, decrease, q, borrow);
    input [3:0]limit;
    input clk;
    input rst;
    input decrease;
    output reg [3:0]q;
    output reg borrow;
    reg [3:0]q_temp;

    always @*
        if (q == 4'b0000 && decrease == 1) begin
            q_temp = 4'b1001;
            borrow = 1'b1;
        end
        else if (q != 4'b0000 && decrease == 1) begin
            q_temp = q - 1'b1;
            borrow = 1'b0;
        end
        else begin
            q_temp = q;
            borrow = 1'b0;
        end

    always @(posedge clk or negedge rst)
        if (~rst) q <= limit;
        else q <= q_temp;
endmodule
```


4'd1
+
-
4
4
0x
10
11
value_tmp
value
4'd9
4
4
2
clk
ifferent
h digit
{decrease==1,value==0}

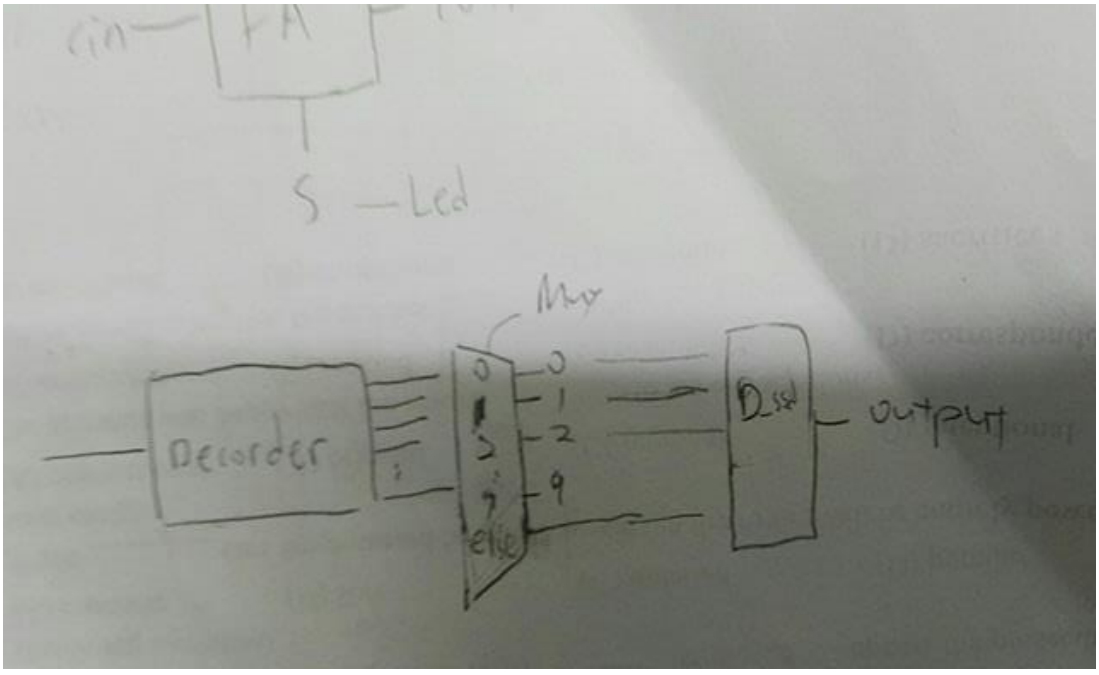

`ENABLED
1
11
else
borrow
`DISABLED
0
2
borrow signal in lov
the decrease control


decrease_enable
digit0
decrease
4'd9
limit
4
4
Udc0
value
clk
rst_n
borrow
digit1
decrease
4'd5
limit
4
4
Udc1
value
clk
rst_n
borrow
br1

```
module SCAN(control, rst, in0, in1, light, out);
    input control;
    input rst;
    input [3:0]in0;
    input [3:0]in1;
    output reg [3:0]light;
    output reg [3:0]out;

    always @*
        if (control == 1'b0) begin
            light = 4'b1110;
            out = in0;
        end
        else begin
            light = 4'b1101;
            out = in1;
        end
endmodule
```



```
module DEC(in, out);
    input [3:0]in;
    output reg [7:0]out;

    always @*
        case(in)
            4'd0: out = `SS_0;
            4'd1: out = `SS_1;
            4'd2: out = `SS_2;
            4'd3: out = `SS_3;
            4'd4: out = `SS_4;
            4'd5: out = `SS_5;
            4'd6: out = `SS_6;
            4'd7: out = `SS_7;
            4'd8: out = `SS_8;
            4'd9: out = `SS_9;
            default: out = 8'b11111111;
        endcase
endmodule
```




```
module STOP(in1, in2, out);
    input [3:0]in1;
    input [3:0]in2;
    output reg out;

    always @*
        if (in1 == 4'b0000 && in2 == 4'b0000) out = 0;
        else out = 1;
endmodule
```

discussion

這題複雜很多，down_counter 需要多加一個 borrow 跟 decrease，分別是下一位要不要減跟這一位要不要減，還要用 scancontrol 去顯示 2 個不同的數字，最後還寫了一個 stopmodule 去停在 00。

這題遇到跟以前一樣的問題，就是只會顯示 1 跟 0，但由於前面犯過了錯，我馬上去檢查哪裡少宣告成 4-bit，結果就解決了。

結果就是從 30 數到 00 然後停住。

Conclusion

這題其實花我很久，我聽完老師講之後，架構很快就打完了，可是 debug 找了很久，因為有太多地方要宣告成 4bit，導致我找到幾個後以為沒錯了，轉而去檢查其他對的地方。