第四次實驗報告

題目:4-1~4-4

姓名:羅名志

學號:0813228

繳交日期:2022/3/23

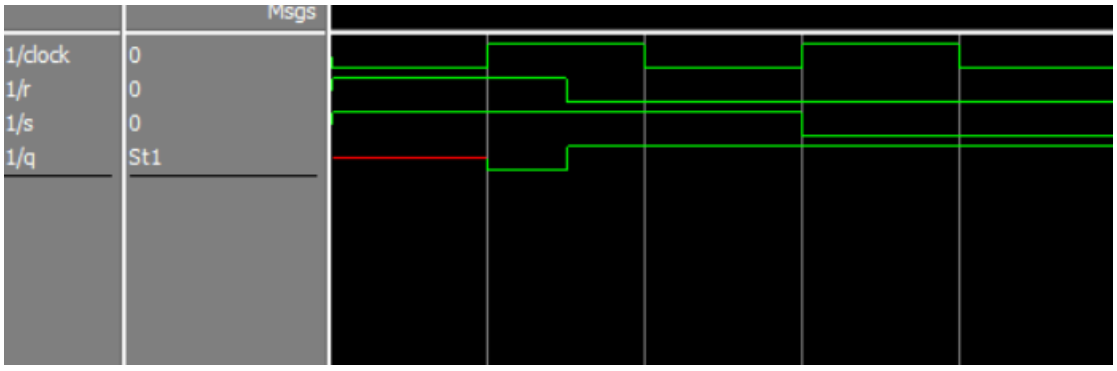
一、4-1

(1)實驗程式碼:

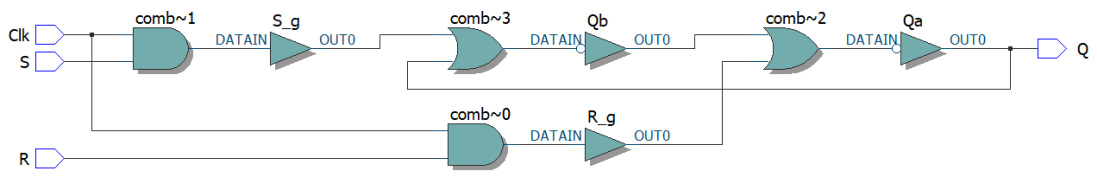
```
1  module labfour1(Clk,R,S,Q);
2  input Clk,R,S;
3  output Q;
4
5  wire R_g,S_g,Qa,Qb/*synthesis keep*/;
6
7  and(R_g,R,Clk);
8  and(S_g,S,Clk);
9  nor(Qa,R_g,Qb);
10 nor(Qb,S_g,Qa);
11
12 assign Q=Qa;
13
14 endmodule
```

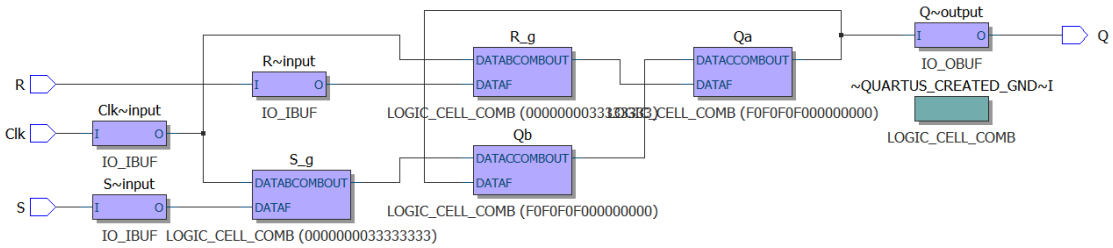
```
1  `timescale 1ns/100ps
2  module testbanch1();
3
4  reg clock;
5  reg r;
6  reg s;
7  wire q;
8
9  labfour1 u1(
10 .Clk(clock),
11 .R(r),
12 .S(s),
13 .Q(q)
14 );
15
16 always #2 clock=~clock;
17
18 initial begin
19    #0 clock=1'b0;
20    #0 r=1'b1;
21    #0 s=1'b1;
22    #3 r=1'b0;
23    #3 s=1'b0;
24 end
25 endmodule
```

(2)實驗結果:



(3)RTL 布局:





RTL 布局/Technology viewer

(4)問題與討論:

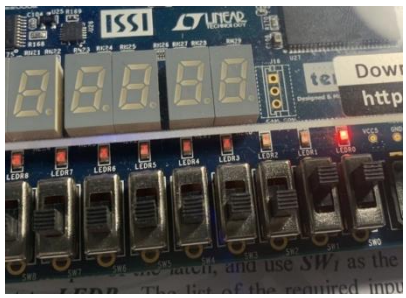
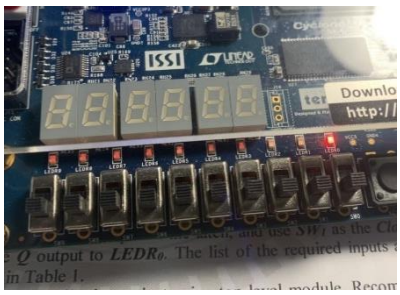
第一次使用 simulation 看波型，同時也是第一次使用 synthesis keep 的做法設計不同的 latch，這次的實驗雖然概念上都蠻簡單的，但實際操作過程有點多步驟且複雜，算是學到了許多(尤其是 test banch 的部分)，另外，由於我沒修過邏設，所以也算是第一次接觸 SR latch。

二、4-2

(1)實驗程式碼:

```
1   module labfour2(SW,LEDR);
2
3   input [1:0]SW;
4   output [0:0]LEDR;
5
6   wire R_g,S_g,Qa,Qb;
7
8   nand(R_g,SW[0],SW[1]);
9   nand(S_g,~SW[0],SW[1]);
10  nand(Qa,R_g,Qb);
11  nand(Qb,S_g,Qa);
12
13  assign LEDR[0]=Qa;
14
15  endmodule
```

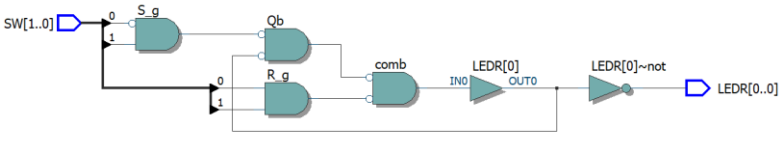
(2)實驗結果:


//上: clock=1; D=0 / clock=1; D=1

//下: clock=0; D=0 / clock=0; D= 1

(3)RTL 布局:



(4)問題與討論:

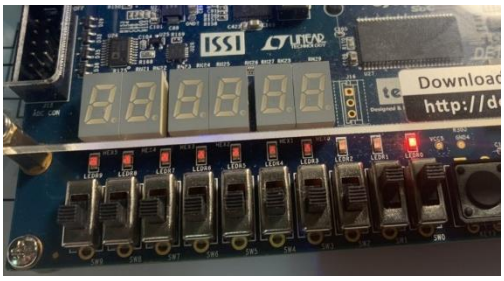
這部份想蠻久的，利用 4 個 nand 組合成 D latch 邏輯，但程式結構相對前次的 lab 簡單許多。

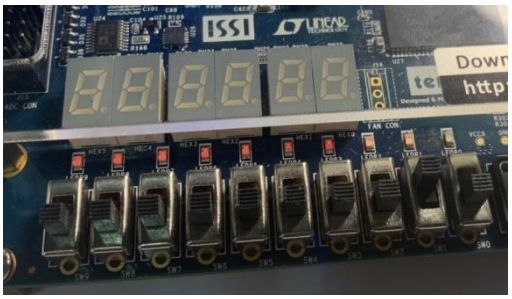
三、4-3

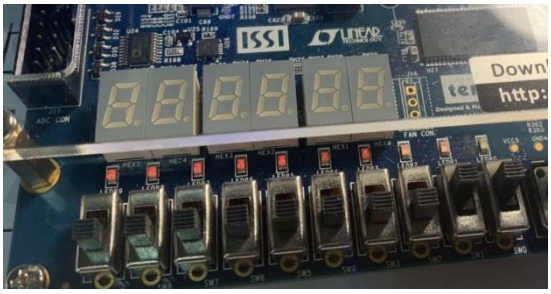
(1)實驗程式碼:

```
module labfour3(SW,LEDR);
input [1:0]SW;
output [0:0]LEDR;

wire R_g,S_g,Qa,Qb,D,Qaa,Qbb,R_gg,S_gg;

nand(R_g,SW[0],~SW[1]);
nand(S_g,~SW[0],~SW[1]);
nand(Qa,R_g,Qb);
nand(Qb,S_g,Qa);

assign D=Qa;
nand(R_gg,D,SW[1]);
nand(S_gg,~D,SW[1]);
nand(Qaa,R_gg,Qbb);
nand(Qbb,S_gg,Qaa);

assign LEDR[0]=Qaa;

endmodule
```

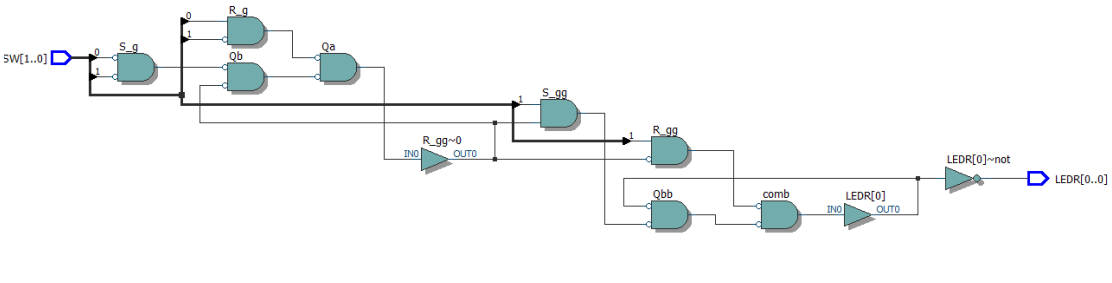
(2)實驗結果:







(3)RTL 布局:



(4) 問題與討論:

延續上個部份的概念，用兩組 nand 結合成 D flip-flop，程式概念相對簡單許多。

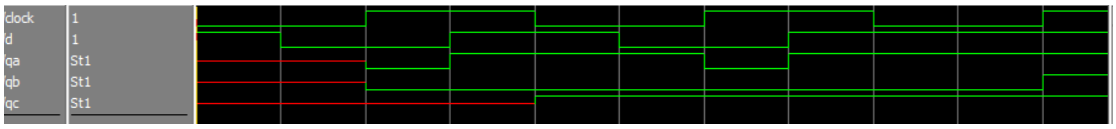
四、4-4

(1) 實驗程式碼:

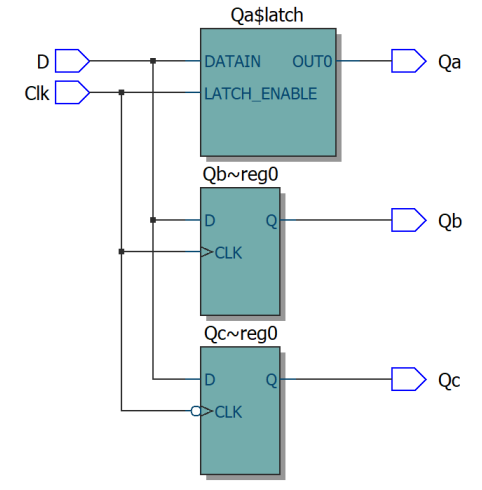
```
module labfour4(Clk,D,Qa,Qb,Qc);
input Clk,D;
output reg Qa,Qb,Qc;

always@(Clk,D) begin
    if(Clk)begin
        Qa=D;
    end
end

always@(posedge Clk) begin
        Qb=D;

end

always@(negedge Clk) begin
        Qc=D;

end

endmodule
```

```
`timescale 1ns/100ps
module testbanch2();

reg clock;
reg d;
wire qa,qb,qc;

labfour4 u1(
.Clk(clock),
.D(d),
.Qa(qa),
.Qb(qb),
.Qc(qc)
);

always #2 clock=~clock;

initial begin
    #0 clock=1'b0;
    #0 d=1'b1;
    #1 d=1'b0;
    #2 d=1'b1;
    #2 d=1'b0;
    #2 d=1'b1;

end
endmodule
```

(2) 實驗結果:



(3)RTL 布局:



(4)問題與討論:

Always 的作法讓我大開眼界，之前都一個個使用 assign 結合 and、or、not 等邏輯閘，但使用了 always 後簡單快速了許多，但需要注意 posedge、negedge 的部分，另外，也學到 test banch 需要將參數仔細設定，如宣告 wire、reg 的不同。