實驗主題(Lab13)

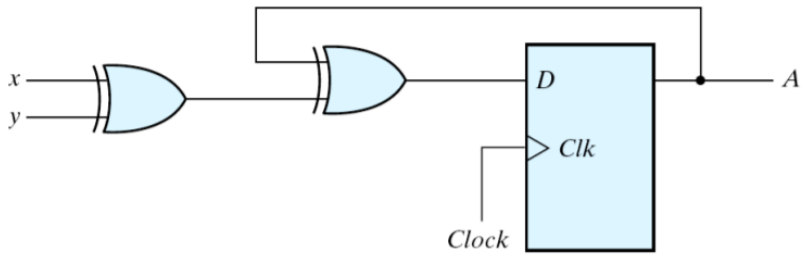
B083022053

黃啟桓

Exercise 1:

Exercise 1: Analysis of Sequential Circuit (D-FF) (1/2)

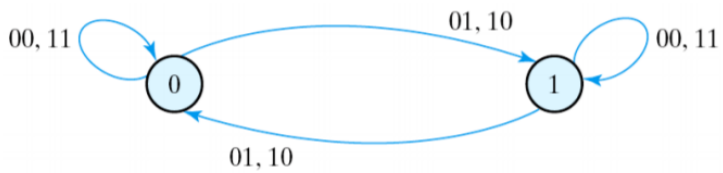
- Simulate the sequential circuit shown in Fig. 5.17
  - Drive the input equation and state equation of the sequential circuit shown in Fig. 5.17(a)
  - Drive the state table and the state diagram of the sequential circuit shown in Fig. 5.17(a)
  - Write the Verilog HDL description of the state diagram (i.e., Behavioral model)
  - Write the Verilog HDL description of the logic circuit diagram (i.e., Structural model)
  - Write a Verilog HDL stimulus with a sequence of inputs: 00, 01, 11, 10. Verify that the response is the same for both descriptions (first reset A to 0).



(a) Circuit diagram

Present state	Inputs		Next state
A	x	y	A
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(b) State table

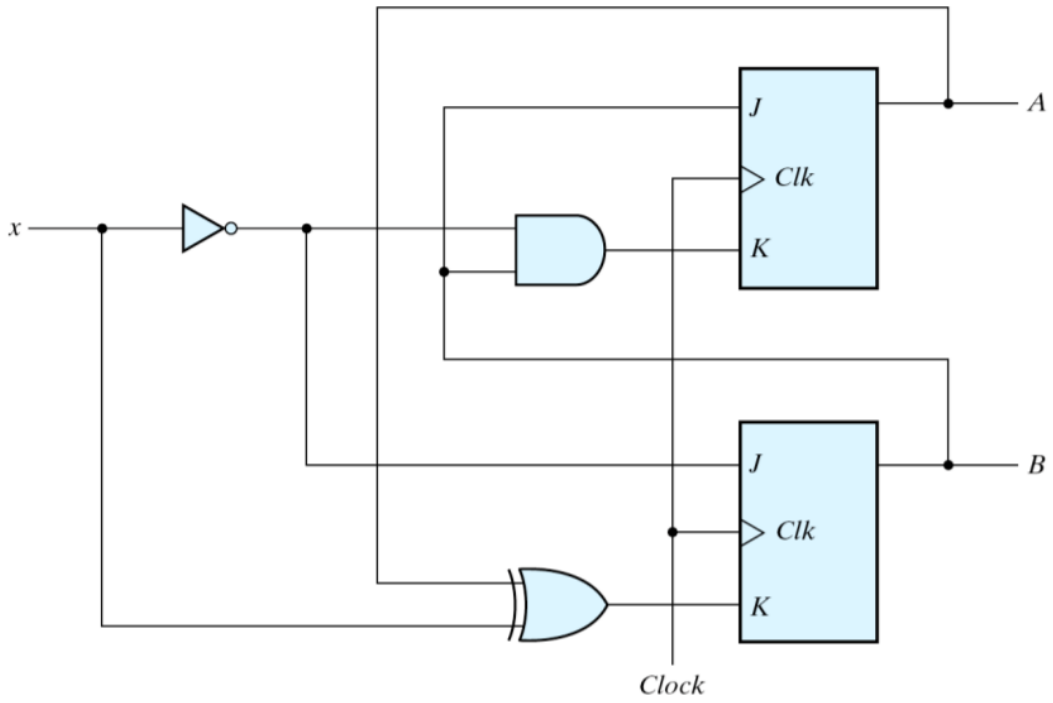


(c) State diagram

Exercise 2:

Exercise 2: Analysis of Sequential Circuit (JK-FF) (1/2)

- Simulate the sequential circuit shown in Fig. 5.18
  - Drive the input equation and state equation of the sequential circuit shown in Fig. 5.18
  - Drive the state table and the state diagram of the sequential circuit shown in Fig. 5.18
  - Write the Verilog HDL description of the state diagram (i.e., Behavioral model)
  - Write the Verilog HDL description of the logic circuit diagram (i.e., Structural model)
  - Write a Verilog HDL stimulus with a sequence of inputs: 0, 1, 0, 0. Verify that the response is the same for both descriptions (first reset A and B to 0)



實驗內容

一、 Exercise 1:

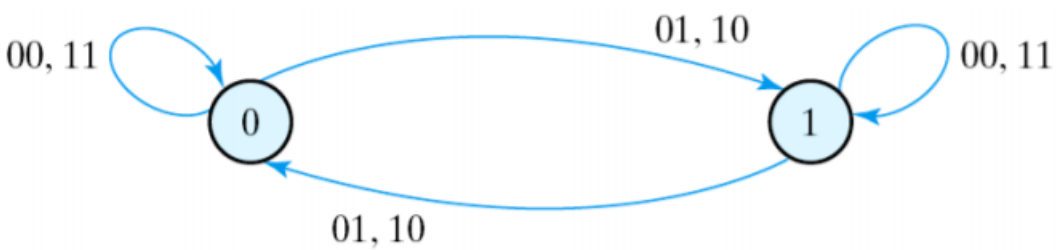
- Drive the input equation and state equation of the sequential circuit shown in Fig. 5.17(a)

※input equation: $D_A = A \oplus x \oplus y$

※state equation: $A(t+1) = A(t) \oplus x \oplus y$

- Drive the state table and the state diagram of the sequential circuit shown in Fig. 5.17(a)

Present State	Input		Next State
A	x	y	A
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



(c) State diagram

- Write the Verilog HDL description of the state diagram (i.e., Behavioral model)

exercise_1a.v × testbench.v × exercise_1b.v × Untitled 7 ×

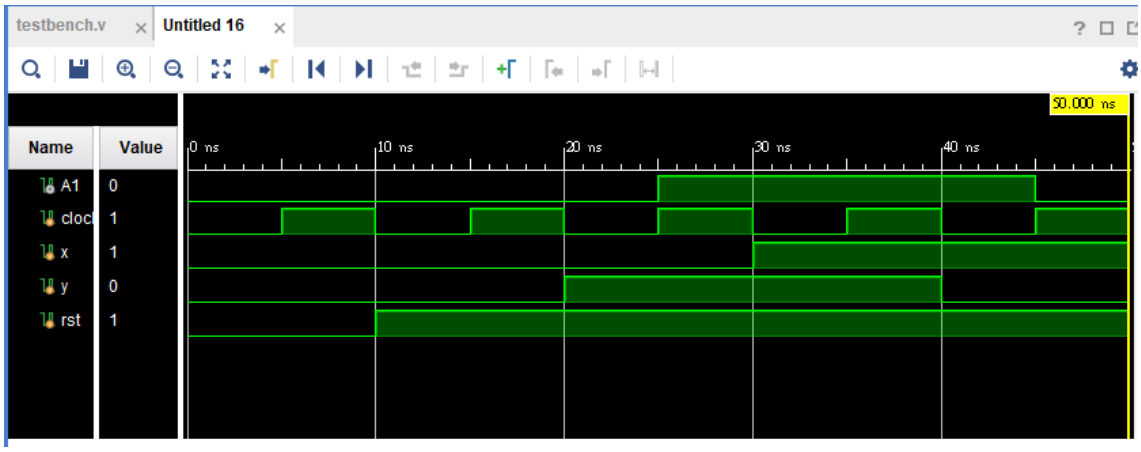
C:/Users/ncpc/project_1/project_1.srcs/sources_1/new/exercise_1a.v

```
1   `timescale 1ns / 1ps
2
3
4   module exercise_1a (A, clock, x, y, rst);
5       output A;
6       input clock, x, y, rst;
7       reg state;
8
9
10      parameter S0 = 1'b0, S1 = 1'b1;
11      always @ (posedge clock, negedge rst)
12      if(rst == 0) state = 0;
13      else
14      case (state)
15          S0:if ( (x^y) == 0) state <= S0; // Initialize to state S0
16            else state <= S1;
17          S1:if ( (x^y) == 1) state <= S0; // Initialize to state S0
18            else state <= S1;
19      endcase
20      assign A = state; // Output of flip-flops
21
22  endmodule
```

Parameter S0 = 1’b0 預設 S0 為 1’b0。

always @(posedge clk,negedge rst)：當達成 clk 正緣或 rst 負緣時執行 always 的 block。

If – else 一般會執行到分號(；) 也就是一行程式碼，若程式碼比較長則需要加 begin – end 或 case – endcase ，前者類似於{} ，後者類似於 switch - case (一對多 多工器)



實驗結果如上 clk 正緣時，A 才改變

用 rst = 0 來讓 A 初始值變為 0

- Write the Verilog HDL description of the logic circuit diagram (i.e., Structural model)

exercise_1a.v × | testbench.v × | **exercise_1b.v** × | Untitled 7

C:/Users/ncpc/project_1/project_1.srcs/sources_1/new/exercise_1b.v

```
module D_flip_flop (Q, Clock, D, Reset);
    output Q;
    input Clock, D, Reset;

    wire n1,n2,n3,n4,n5,n6;


    nand(n1, n3, Clock, Reset);
    nand(n2, n1, Clock, n4);
    nand(n3, n1, n4);
    nand(n4, n2, D, Reset);
    nand(n5, n1, n6);
    nand(n6, n2, n5 , Reset);

    buf(Q, n5);
endmodule

module exercise_1b (A, clock, x, y, rst);
    output A;
    input clock, x, y, rst;
    wire n1;
    xor(n1, A, x, y); // Output of flip-flops
    D_flip_flop DFF1(A, clock, n1, rst);

endmodule
```

D_flip_flop = D flip-flop 的 block

引入變數 rst 也是單純為了設定初值

實驗結果如上 clk 正緣時，A 才改變

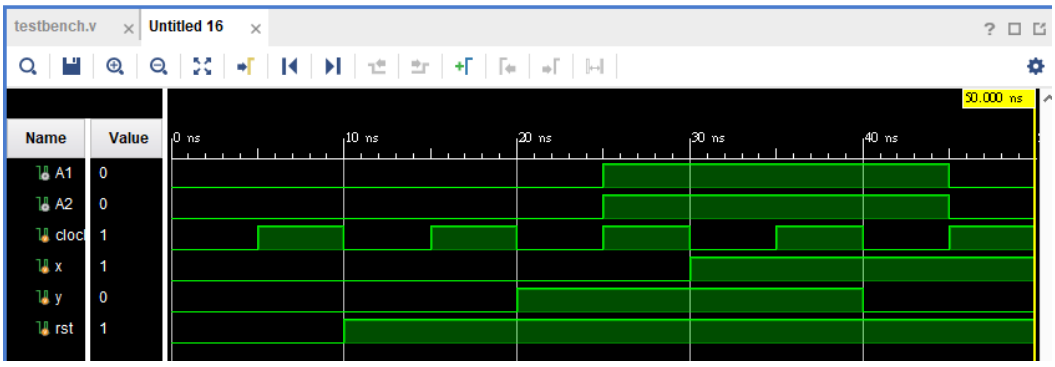
用 rst = 0 來讓 A 初始值變為 0

- Write a Verilog HDL stimulus with a sequence of inputs: 00, 01, 11, 10. Verify that the response is the same for both descriptions (first reset A to 0).

Reset A to 0 -> 在 D flip flop 加入 reset 控制 A

只要控制 testbench 的 4、5 行就能選擇只輸出 behavior model 或是 structural model 的結果

```
1. module testbench();
2. wire A1,A2;
3. reg clock, x, y, rst;
4. exercise_1a M1 (A1, clock, x, y, rst);
5. exercise_1b M2 (A2, clock, x, y, rst);
6.     initial
7.      #50 $finish;
8.     initial begin
9.          rst = 0;
10.     #10 rst = 1;
11.    end
12.    initial begin
13.        clock = 0;
14.        forever #5
15.            clock = ~clock;
16.    end
17.    initial  begin
18.         x <= 1'b0;y <= 1'b0;
19.     #20 x <= 1'b0;y <= 1'b1;
20.     #10 x <= 1'b1;y <= 1'b1;
21.     #10 x <= 1'b1;y <= 1'b0;
22.     end
23. endmodule
```



二、Exercise 2:

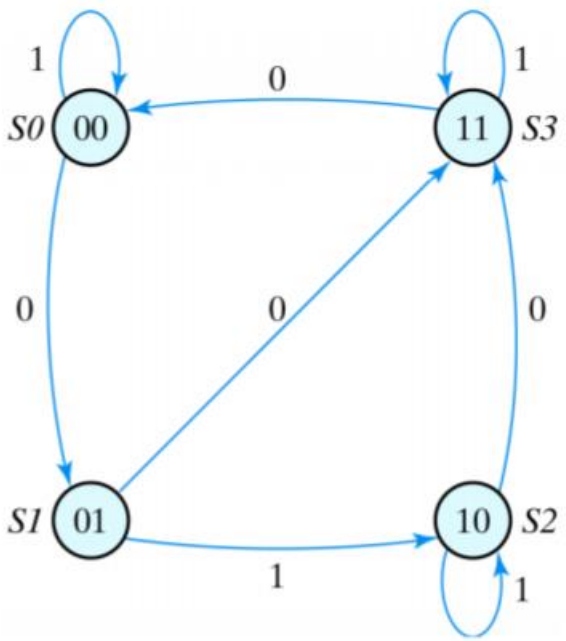
- Drive the input equation and state equation of the sequential circuit shown in Fig. 5.18

※input equation: D=JQ'+K'Q

※state equation: Q(t+1)= JQ(t)'+K'Q(t)

- Drive the state table and the state diagram of the sequential circuit shown in Fig. 5.18

Present State		Input	Next State	
A	B	x	A	B
0	0	0	0	1
0	0	1	0	0
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1



- Write the Verilog HDL description of the state diagram (i.e., Behavioral model)

exercise_2a.v × testbench.v × Untitled 4 ×

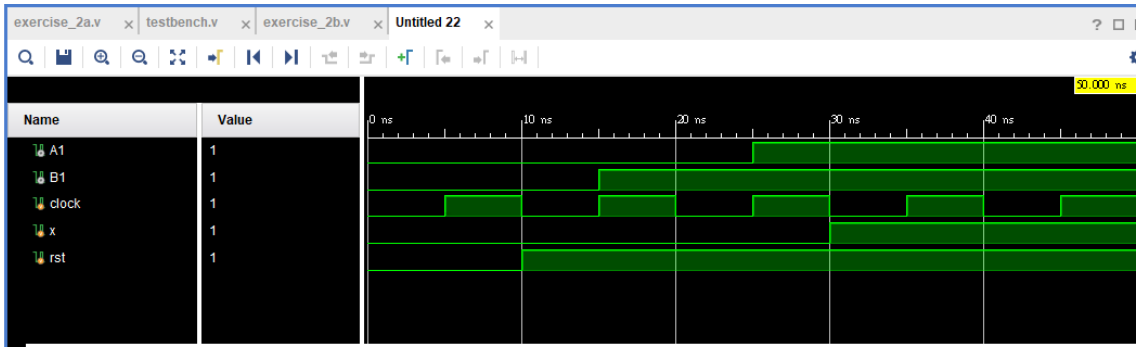
C:/Users/ncpc/project_2/project_2.srcs/sources_1/new/exercise_2a.v

```
1   `timescale 1ns / 1ps
2
3
4   module exercise_2a (A, B, x, clock,rst);
5       output A, B;
6       input x, clock, rst;
7
8       reg [1: 0] state;
9       parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11
10
11      always @ (posedge clock, negedge rst)
12      if(rst == 0) state = 2'b00;
13      else
14      case (state)
15          S0: if (x) state <= S0; else state <= S1;
16          S1: if (x) state <= S2; else state <= S3;
17          S2: if (x) state <= S2; else state <= S3;
18          S3: if (x) state <= S3; else state <= S0;
19      endcase
20      assign A = state[1]; // Output of flip-flops
21      assign B = state[0]; // Output of flip-flops
22  endmodule
```

Parameter S0 = 2’b00 預設 S0 為 2’b00。

always @(posedge clk,negedge rst)：當達成 clk 正緣或 rst 負緣時執行 always 的 block。

If – else 一般會執行到分號(；) 也就是一行程式碼，若程式碼比較長則需要加 begin – end 或 case – endcase ，前者類似於{} ，後者類似於 switch - case (一對多 多工器)



實驗結果如上 clk 正緣時，A 才改變

用 rst = 0 來讓 A 初始值變為 0

- Write the Verilog HDL description of the logic circuit diagram (i.e., Structural model)

exercise_2a.v | testbench.v | exercise_2b.v | Untitled 22

C:/Users/ncpc/project_2/project_2.srcs/sources_1/new/exercise_2b.v

```verilog
`timescale 1ns / 1ps

module JK_ff (Q, clock, J, K, Reset);
    output Q;
    input J, K, clock, Reset;
//  reg Q;
    wire Q_, K_, D_in, n1, n2,n3,n4;

    reg Q;
    not (Q_ , ~Q);
    always @ (posedge clock, negedge Reset)
    if (Reset == 0) Q <= 1'b0;
    else Q <=  J&~Q|~K&Q;
    endmodule

module exercise_2b (A, B, x, clock, rst);
    output A, B;
    input x, clock, rst;

    wire J1,J2,K1,K2,n1,n2,x_;
    not (x_ , x);
    buf (J1 , B);
    and (K1 , B , x_);
    not (J2 , x);
    xor (K2 , A , x);

    JK_ff MM2(A, clock, J1, K1, rst);
    JK_ff MM3(B, clock, J2, K2, rst);
endmodule
```

JK_ff 為 JK flip_flop block

實驗結果如上 clk 正緣時，A 才改變

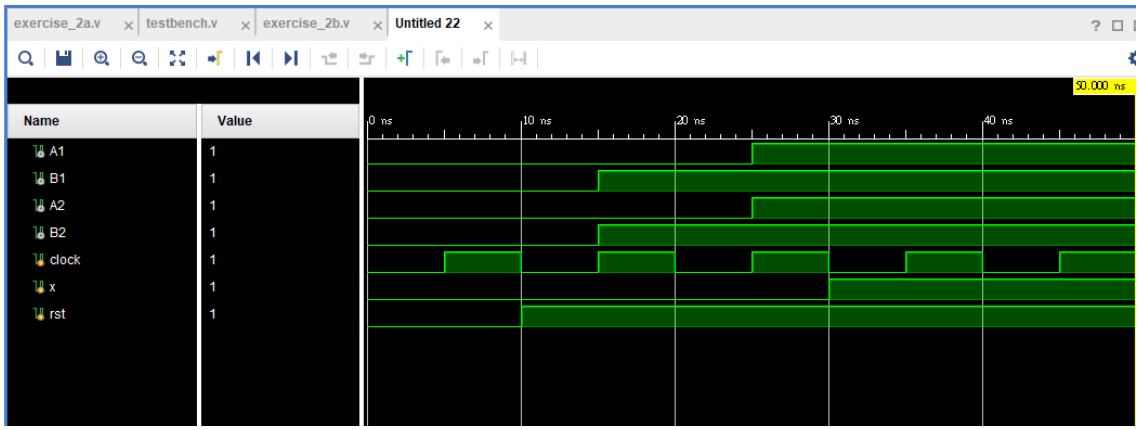
用 rst = 0 來讓 A 初始值變為 0

- Write a Verilog HDL stimulus with a sequence of inputs: 0, 1, 0, 0. Verify that the response is the same for both descriptions (first reset A and B to 0)

Reset A and B to 0 -> 在 JK flip flop 加入 reset 控制 A B

只要控制 testbench 的 4、5 行就能選擇只輸出 behavior model 或是 structural model 的結果

```
1. module testbench();
2. wire A1,A2;
3. reg clock, x, rst;
4. exercise_2a M1 (A1, B1, x, clock, rst);
5. exercise_2b M2 (A2, B2, x, clock, rst);    initial
6.      #50 $finish;
7.     initial begin
8.          rst = 0;
9.      #10 rst = 1;
10.    end
11.    initial begin
12.        clock = 0;
13.        forever #5
14.            clock = ~clock;
15.    end
16.    initial  begin
17.         x <= 1'b0;y <= 1'b0;
18.     #20 x <= 1'b0;y <= 1'b1;
19.     #10 x <= 1'b1;y <= 1'b1;
20.     #10 x <= 1'b1;y <= 1'b0;
21.     end
22. endmodule
```



實驗心得

1. Structural level modeling 中邏輯閘使用函數表示
2. Structural level modeling 可以很直觀的轉換成 Schematic
3. Structural level modeling 需要宣告邏輯閘之間的線路
4. 所有的 modeling 都需要注意該邏輯閘是否有交換律或結合律