數位系統設計實習

Lecture 7

指導老師:陳勇志 老師

實習課助教:黃柏皓

Outline

- Finite State Machine
  - Introduce
  - Moore Machine
  - Mealy Machine
- Binary to BCD
- LAB 7-1 ~7-3

Outline

- Finite State Machine
  - Introduce
  - Moore Machine
  - Mealy Machine
- Binary to BCD
- LAB 7-1 ~7-3

Finite State Machine (FSM)

- 有限狀態機 (Finite State Machine, FSM), 是使用一個或是多個正反器來儲存內部的狀態, 利用組合邏輯的設計輸入目前狀態,並且計算出下一個狀態輸出, 為循序邏輯電路中變化性最多的一種。

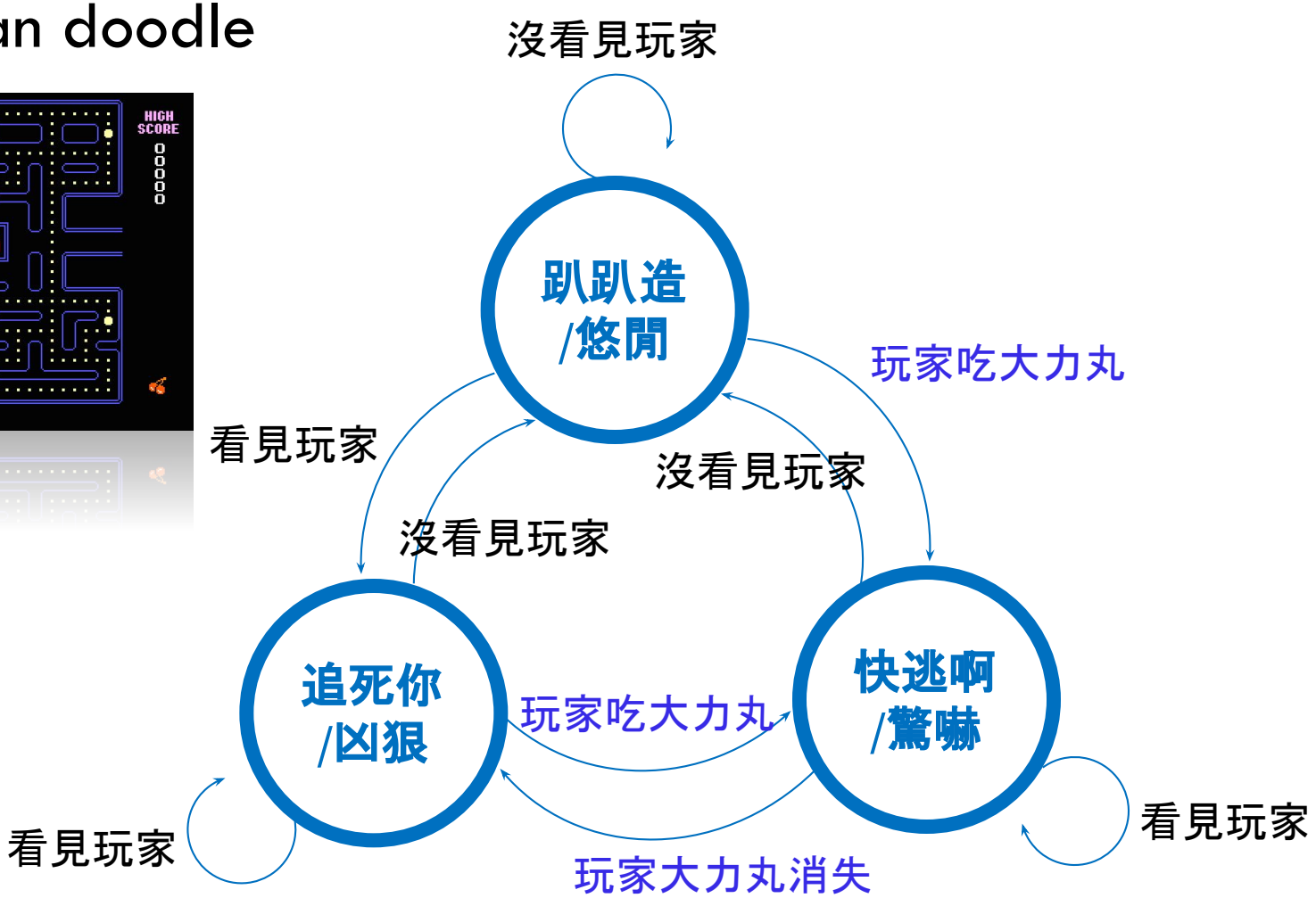
- 而有限狀態機依其輸入的訊號、輸出的訊號, 以及狀態之間的關係可以分為 Moore Machine 及 Mealy Machine 兩種。

Finite State Machine (FSM)

- **狀態 (States):** 系統存在的有限個狀態。例如，一個交通燈有紅燈、黃燈、綠燈
- **輸入 (Inputs):** 可以引起狀態轉換的事件或訊號。例如，交通燈的輸入可以是定時器觸發。
- **轉移函數 (Transition Function):** 描述系統如何從一個狀態轉移到另一個狀態。它會根據當前的狀態和輸入，決定下一個狀態。
- **起始狀態 (Start State):** 系統的初始狀態。

Finite State Machine (FSM)

- Pac-Man doodle

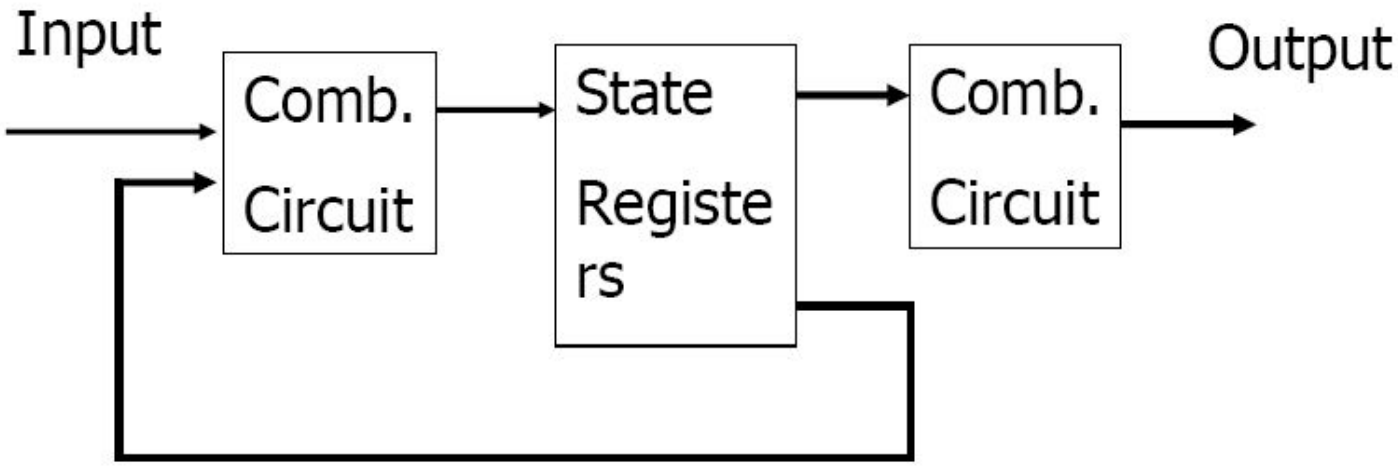


Outline

- Finite State Machine
  - Introduce
  - Moore Machine
  - Mealy Machine
- Binary to BCD
- LAB 7-1 ~7-3

Moore Machine

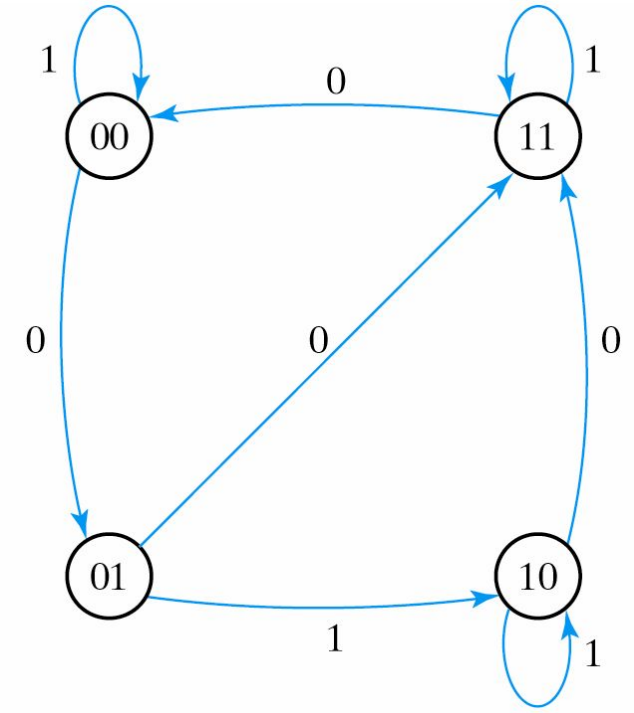
- 這種有限狀態機其輸出的訊號只和目前的狀態有關而與輸入的訊號無關(輸入間接影響輸出)。



Outputs dependent on state variables only.

Moore Machine

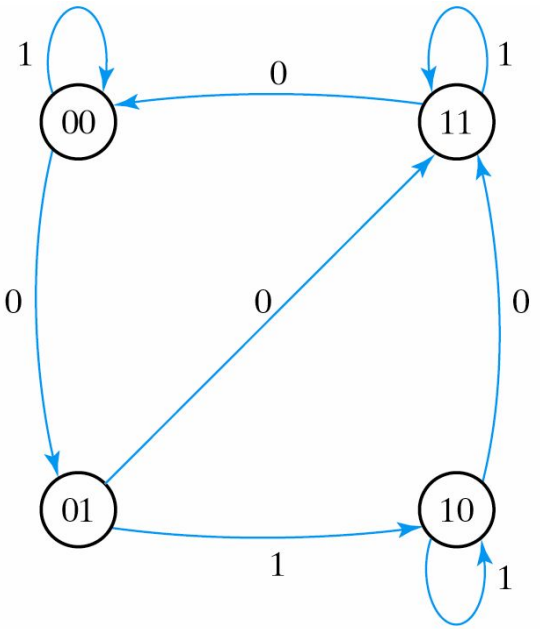
- 狀態圖 (State Diagram) 其中每個圓圈代表一個狀態，圓圈內的值代表該狀態的名稱，而有方向性的箭號代表循序電路的狀態轉移，其旁邊的記號代表該電路轉移所對應的輸入訊號的變化與輸出訊號的變化。



Moore Machine

- 由右邊狀態圖我們可將其狀態表列出:

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



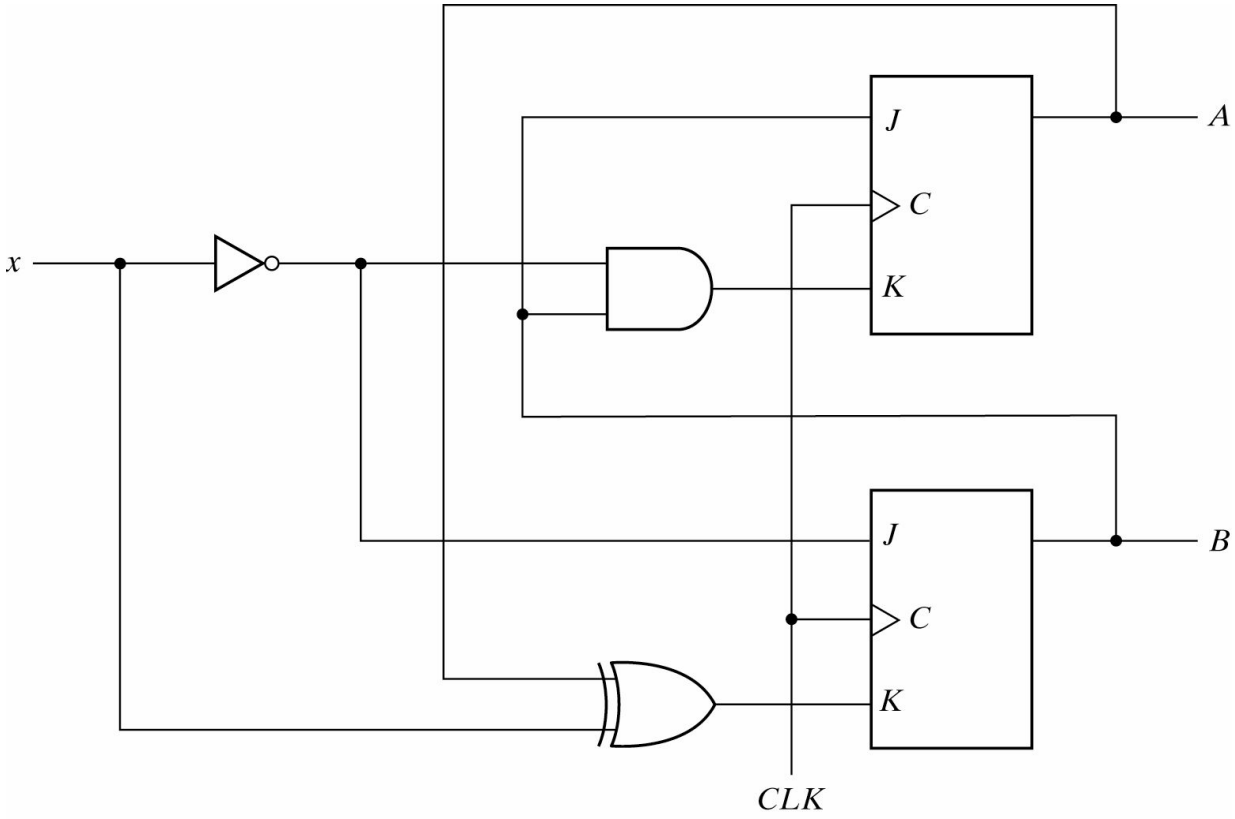
Moore Machine

- 我們選擇使用 JK 正反器, 由狀態表經卡諾圖化簡可得其方程式:

$$J_A = B$$
$$K_A = B\,x'$$
$$J_B = x'$$
$$K_B = A \oplus x'$$

Moore Machine

- 由上列方程式可得:



程式碼(1)

```
// Structural description of moore machine
module Moore_mdl (x,AB,Clock);
    input x,Clock;
    output [1:0]AB;
    wire [1:0]AB;
    wire Ja, Ka, Jb, Kb;

    assign Ka = AB[0] & ~x;
    assign Kb = AB[1] ^ x;

JK_FF J1(AB[0],Ka,Clock,AB[1],Qnot);
JK_FF J2(~x,Kb,Clock,AB[0],Qnot);
    endmodule

    //JK flip-flop
    module JK_FF (J,K,CLK,Q,Qnot);
    output Q,Qnot;
    input  J,K,CLK;
    reg  Q;
    assign Qnot = ~ Q ;
    always @ (posedge CLK)
            case ({J,K})
              2'b00: Q = Q;
              2'b01: Q = 1'b0;
              2'b10: Q = 1'b1;
              2'b11: Q = ~ Q;
            endcase
    endmodule
```

程式碼(2)

If(in == 1'b1) state = s0;
else state = s1;

```
parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;

always@ (posedge clk)
if (!rst) state=s0;
else
    case (state)
    s0:
        begin
        state=(in==1'b1)?s0:s1;
        out=2'b00;
        end
    s1:
        begin
        state=(in==1'b1)?s2:s3;
        out=2'b01;
        end
    s2:
        begin
        state=(in==1'b1)?s2:s3;
        out=2'b10;
        end
    s3:
        begin
        state=(in==1'b1)?s3:s0;
        out=2'b11;
        end
    endcase
```

```
always@ (posedge clk)
if (!rst) state=s0;
else
    case (state)
    s0: state=(in==1'b1)?s0:s1;
    s1: state=(in==1'b1)?s2:s3;
    s2: state=(in==1'b1)?s2:s3;
    s3: state=(in==1'b1)?s3:s0;
    endcase

always@(state)
    case (state)
    s0: out=2'b00;
    s1: out=2'b01;
    s2: out=2'b10;
    s3: out=2'b11;
    endcase
```

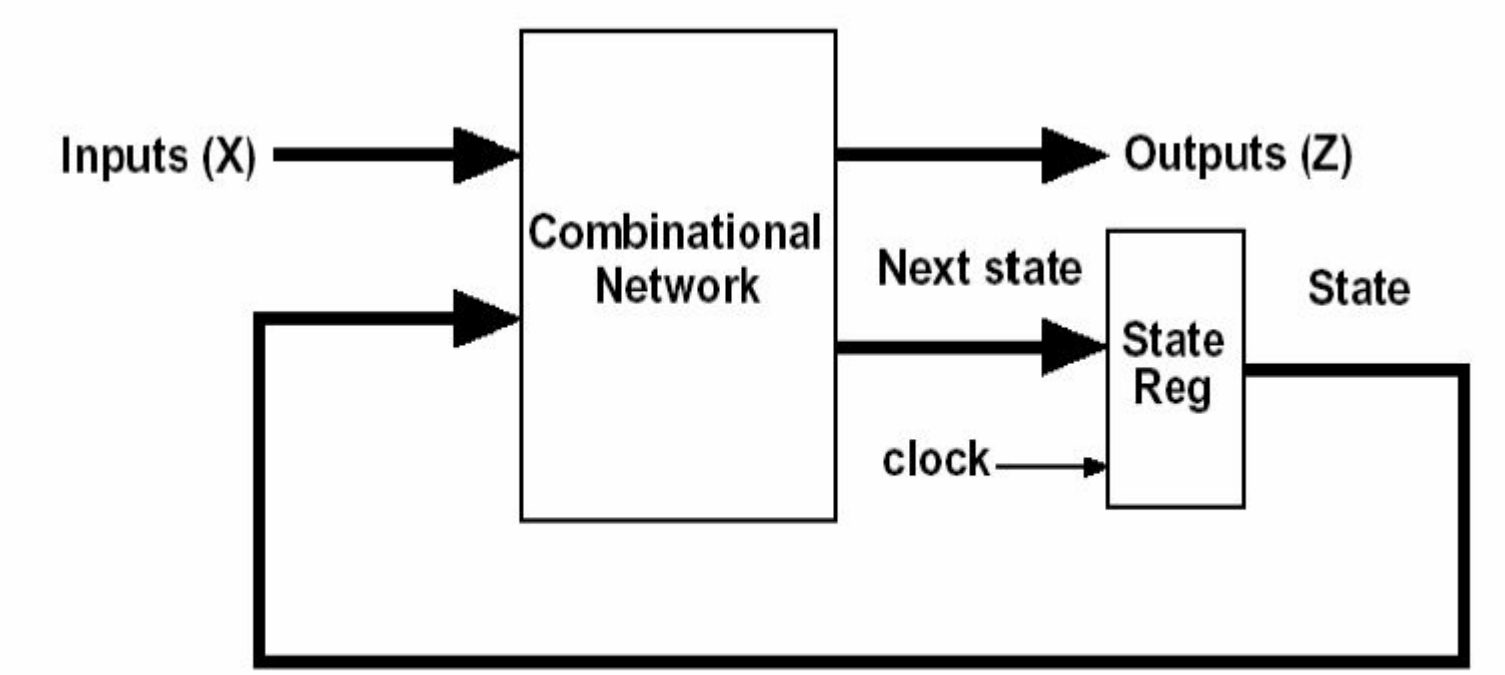
Recommend!

Outline

- Finite State Machine
  - Introduce
  - Moore Machine
  - Mealy Machine
- Binary to BCD
- LAB 7-1 ~7-3

Mealy Machine

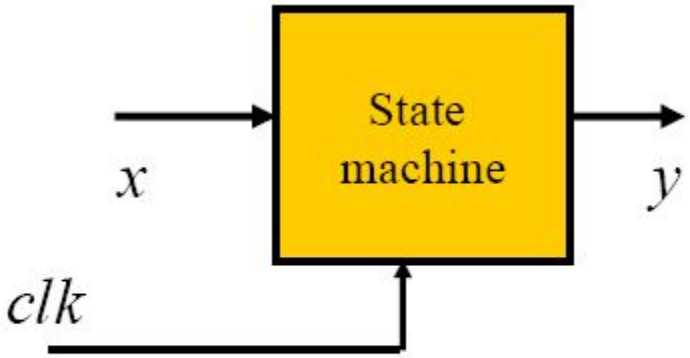
- 此有限狀態機其輸出的訊號必須考慮目前的狀態及輸入的訊號。

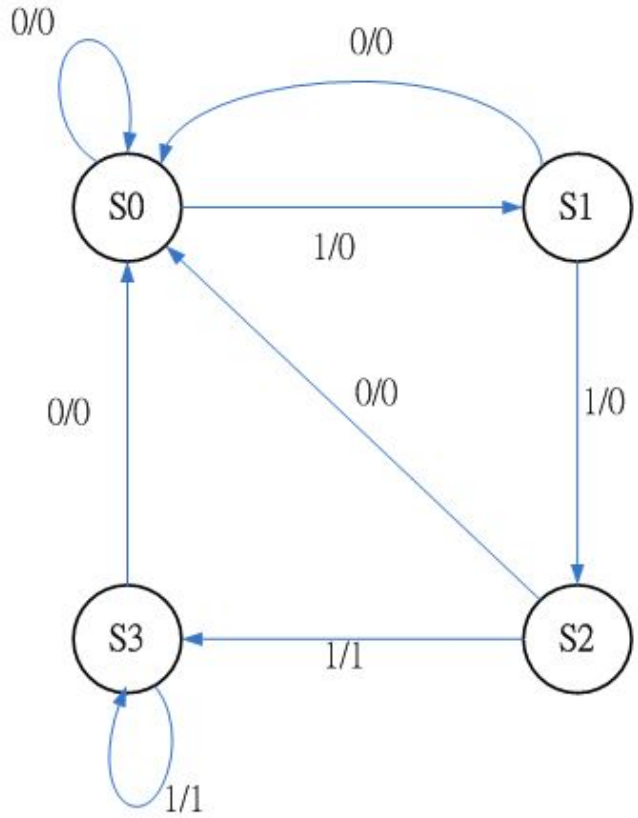


Outputs depend on inputs and state variables.

Mealy Machine

EX：A Sequence Detector

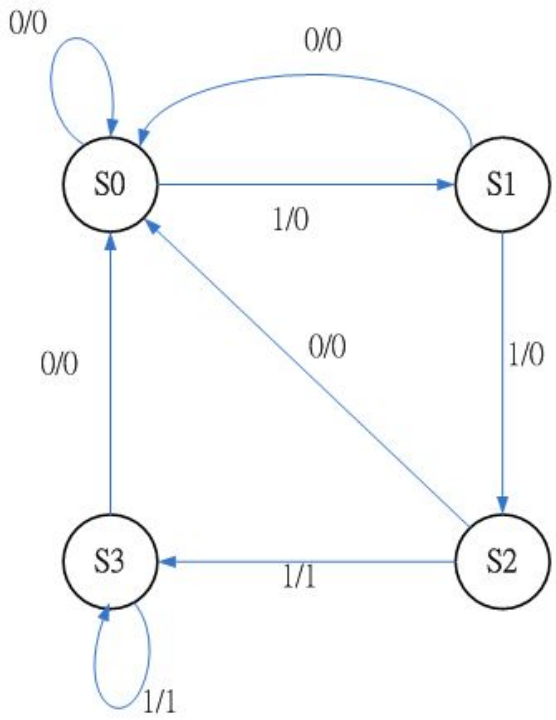




連續讀入三個 1 即輸出 1，否則輸出 0

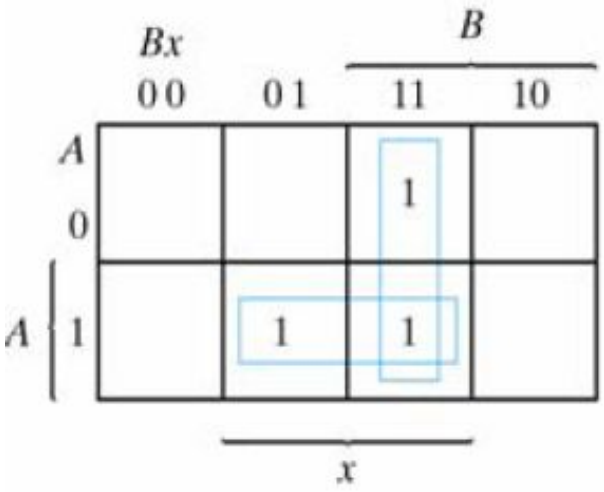
Mealy Machine

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

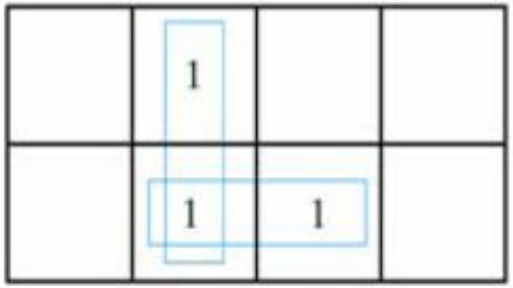


Mealy Machine

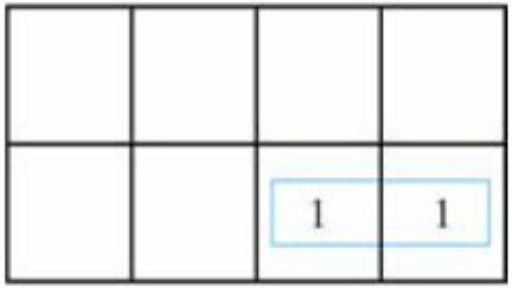
Synthesis Using D FFs:



$D_A = Ax + Bx$

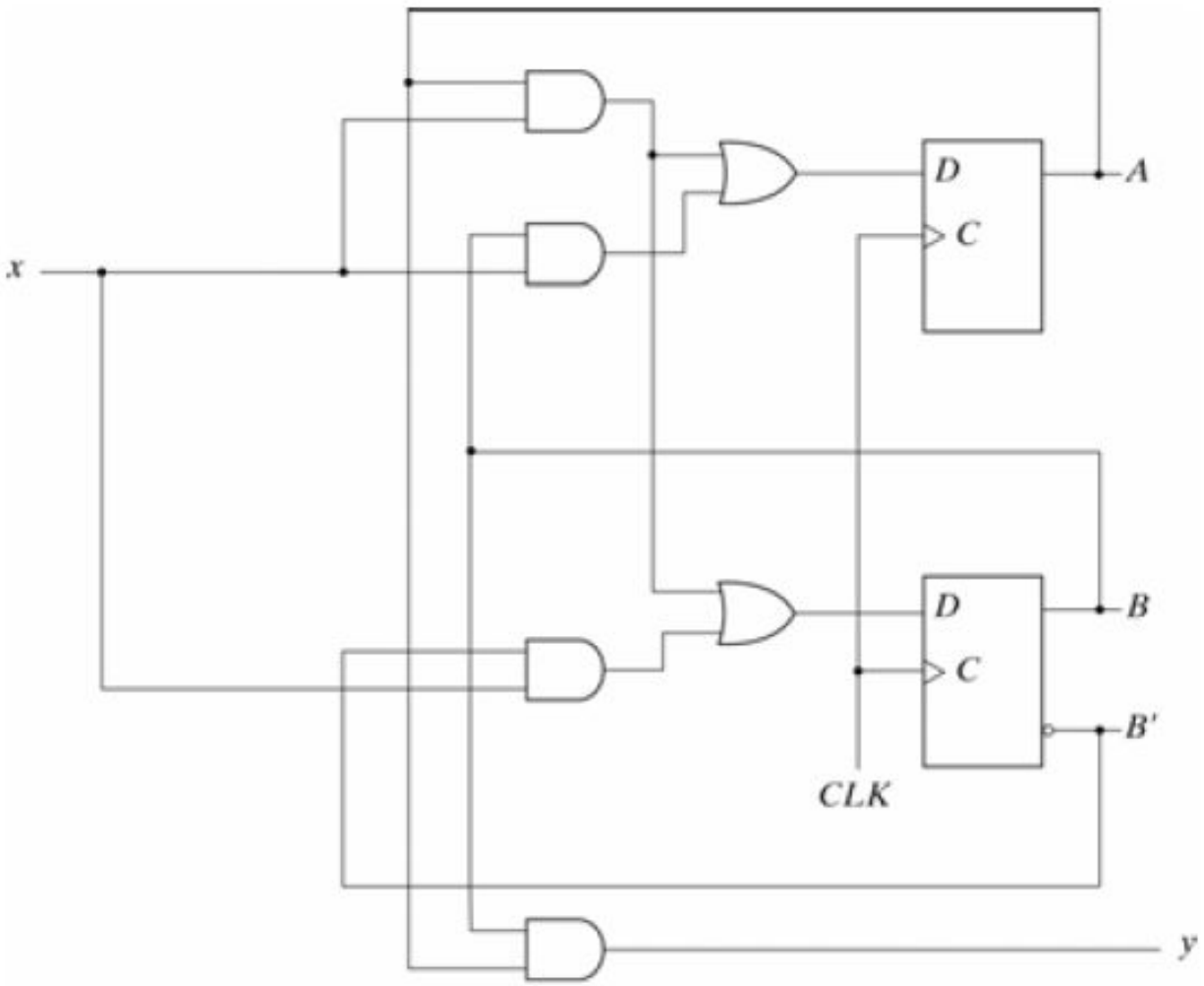


$D_B = Ax + B'x$

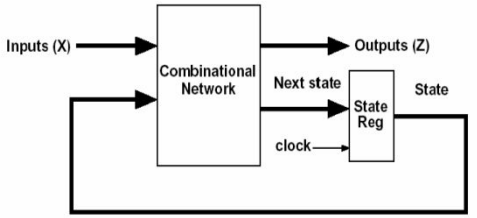


$y = AB$

Mealy Machine



Mealy Machine




```
always@(posedge clk)
if (!rst) state=s0;
else
    case (state)
    s0:
        begin
        state=(in==1'b1)?s1:s0;
        out=(in==1'b1)?2'b00:2'b00;
        end
    s1:
        begin
        state=(in==1'b1)?s2:s0;
        out=(in==1'b1)?2'b00:2'b00;
        end
    s2:
        begin
        state=(in==1'b1)?s3:s0;
        out=(in==1'b1)?2'b01:2'b00;
        end
    s3:
        begin
        state=(in==1'b1)?s3:s0;
        out=(in==1'b1)?2'b01:2'b00;
        end
    endcase
```

```
always@(posedge clk)
if (!rst) state=s0;
else state=next_state;

always@(state)
case (state)
s0:
    begin
    next_state=(in==1'b1)?s1:s0;
    out=(in==1'b1)?2'b00:2'b00;
    end
s1:
    begin
    next_state=(in==1'b1)?s2:s0;
    out=(in==1'b1)?2'b00:2'b00;
    end
s2:
    begin
    next_state=(in==1'b1)?s3:s0;
    out=(in==1'b1)?2'b01:2'b00;
    end
s3:
    begin
    next_state=(in==1'b1)?s3:s0;
    out=(in==1'b1)?2'b01:2'b00;
    end
endcase
```

Outline

- Finite State Machine
  - Introduce
  - Moore Machine
  - Mealy Machine
- Binary to BCD
- LAB 7-1 ~7-3

Binary to BCD

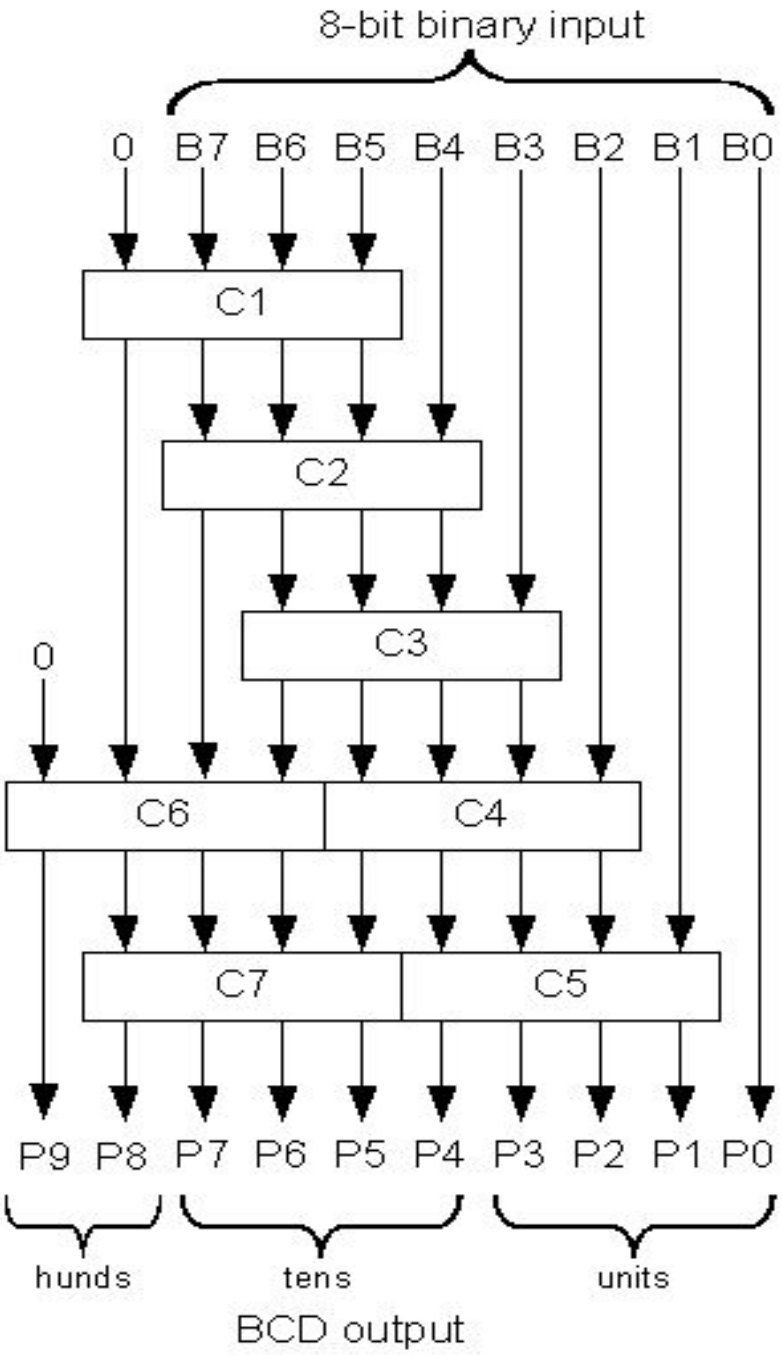
- 如果任何位元(百位、十位、個位...等)等於或大於 5, 則在該位元中新增 3。
- 將所有數字向左移動 1 位。
- 如果已經進行了 n(bits) 次計算, 就完成了！

Binary to BCD

Operation	Tens	Units	Binary
HEX			**E**
Start			1 1 1 0
Shift 1		1	1 1 0
Shift 2		1 1	1 0
Shift 3		1 1 1	0
Add 3		1 0 1 0	0
Shift 4	1	0 1 0 0	
BCD	**1**	**4**	

Binary to BCD

Operation	Hundreds	Tens	Units	Binary	
HEX				F	F
Start				1 1 1 1	1 1 1 1
Shift 1			1	1 1 1 1	1 1 1
Shift 2			1 1	1 1 1 1	1 1
Shift 3			1 1 1	1 1 1 1	1
Add 3			1 0 1 0	1 1 1 1	1
Shift 4		1	0 1 0 1	1 1 1 1	
Add 3		1	1 0 0 0	1 1 1 1	
Shift 5		1 1	0 0 0 1	1 1 1	
Shift 6		1 1 0	0 0 1 1	1 1	
Add 3		1 0 0 1	0 0 1 1	1 1	
Shift 7	1	0 0 1 0	0 1 1 1	1	
Add 3	1	0 0 1 0	1 0 1 0	1	
Shift 8	1 0	0 1 0 1	0 1 0 1		
BCD	2	5	5		

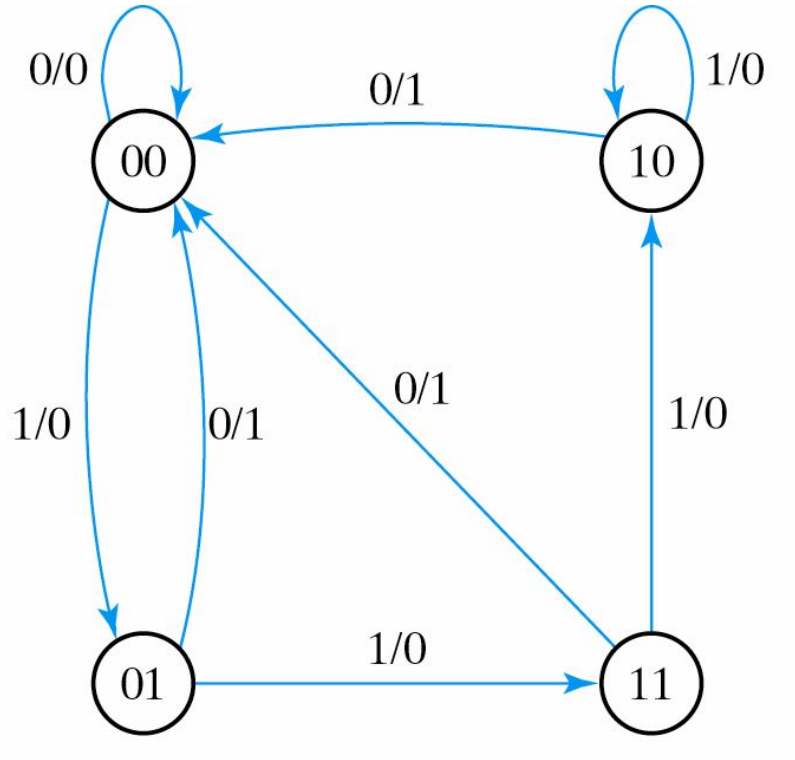

8-bit binary input
0 B7 B6 B5 B4 B3 B2 B1 B0
C1
C2
C3
0
C6
C4
C7
C5
P9 P8 P7 P6 P5 P4 P3 P2 P1 P0
hunds
tens
units
BCD output

Outline

- Finite State Machine
  - Introduce
  - Moore Machine
  - Mealy Machine
- Binary to BCD
- LAB 7-1 ~7-3

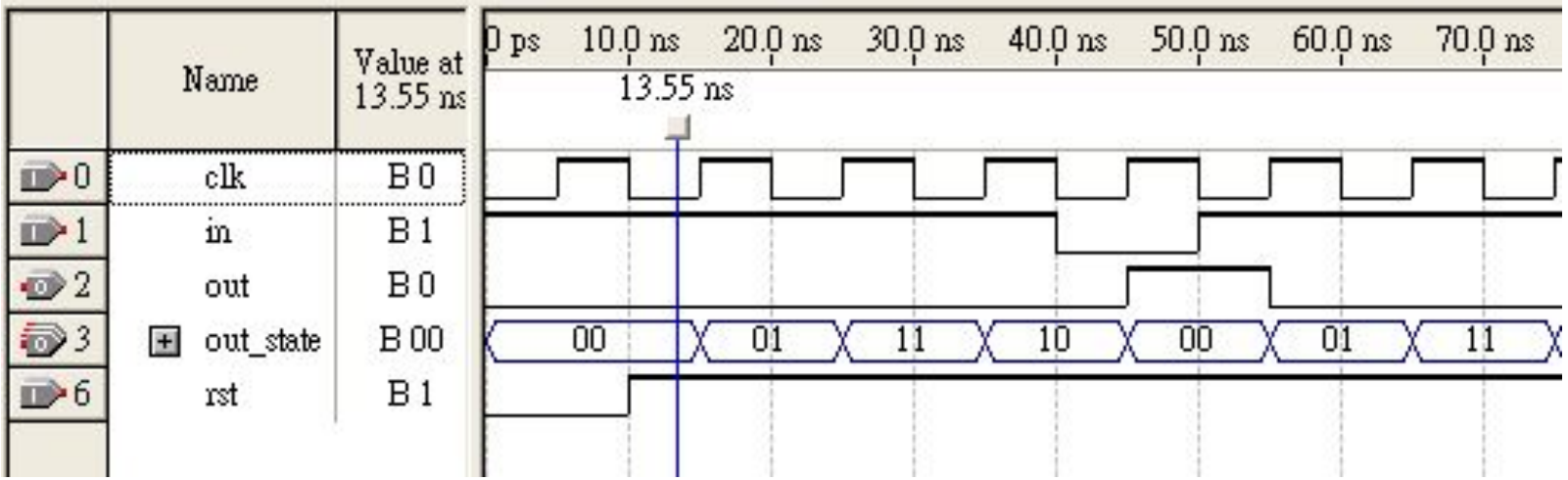
LAB 7-1

- 請使用Verilog HDL描寫出下圖 Mealy machine, 於 Quartus II 模擬訊號波型加以驗證結果。



LAB 7-1

課堂檢查:

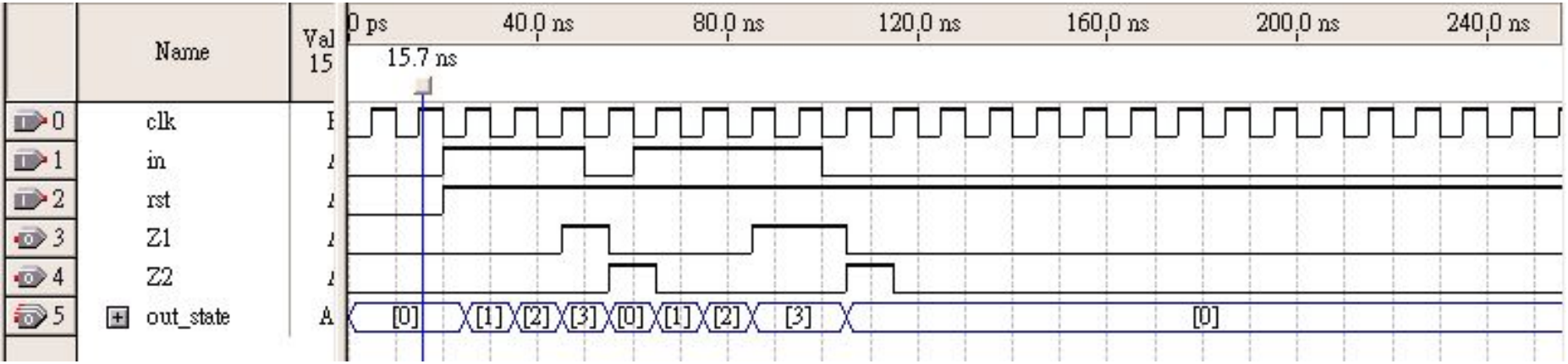

Name
Value at 13.55 ns
0 ps
10.0 ns
20.0 ns
30.0 ns
40.0 ns
50.0 ns
60.0 ns
70.0 ns
13.55 ns
0 clk B 0
1 in B 1
2 out B 0
3 out_state B 00
6 rst B 1
00
01
11
10
00
01
11

LAB 7-2

- 請設計出一 Mealy machine, 輸入為 X, 輸出為 Z1 和 Z2, 當檢測到 X連續輸入為 111 時, Z1 輸出為 1;當檢測到 X 之輸入順序為 1110時, Z2 輸出為 1。請使用 Verilog HDL 描寫出並於 Quartus II 模擬訊號波型加以驗證結果。

LAB 7-2

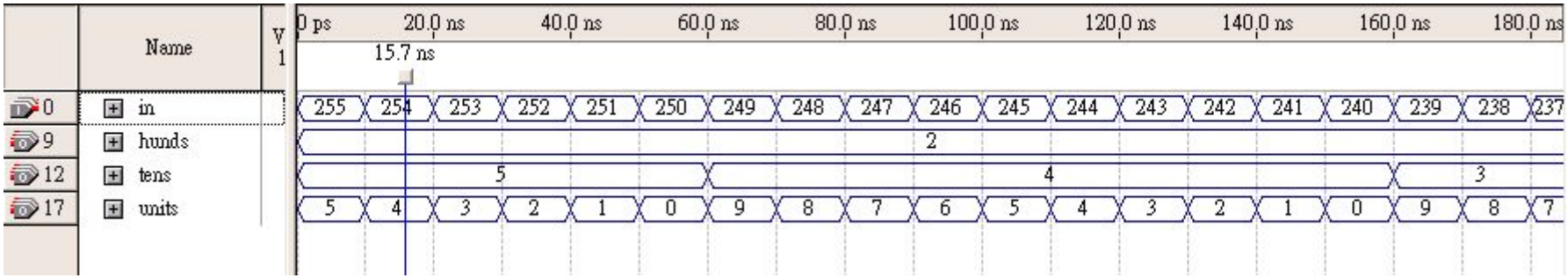
課堂檢查:


Name
Val
15
0 ps
40.0 ns
80.0 ns
120.0 ns
160.0 ns
200.0 ns
240.0 ns
15.7 ns
0 clk
1 in
2 rst
3 Z1
4 Z2
5 out_state
A
[0]
[1]
[2]
[3]
[0]
[1]
[2]
[3]
[0]

LAB 7-3

- 請使用 Verilog HDL 描寫出將 8-bit binary input 轉成 BCD output, 並於 Quartus II 模擬訊號波型加以驗證結果。

課堂檢查:



LAB7

下課前將各Lab繳交至moodle：

上傳verilog.v

波形截圖