HDL Behavior Model using FPGA

Ch5. Synchronous sequential circuit

SOP

- Write Verilog code
- Compile
- Run simulation
- Programmer to FPGA, verify the results
  - Assign board I/O pins => find FPGA I/O pins => assignment editor

Outline

- **Ch5: Sequential logic**
  - **Basics of behavior modeling**
  - **HDL model of latches and flip-flops**
    - » **Example 5.2: D flip-flop (Lab#1)**
    - » **Example 5.4: JK flip-flop (Lab#2)**
  - **Mealy and Moore Models of finite state machine**
    - » **Example 5.5: Mealy Zero detector (Lab#3)**
    - » **Example 5.6: Moore Model FSM (Lab#4)**
    - » **Example 5.7: binary counter**

Synthesizable HDL Models of Sequential Circuits

- **Behavioral Modeling**
  - **Abstract representations of the functionality of digital hardware**
  - **Don't specify the internal details of the circuits**
- **Single-pass behavior**
  - **Keywords: initial, *begin … end, fork … join***
  - **Used in test bench, never to model a circuit**
- **Cyclic behavior**
  - **Keyword: always**
- **A module can contain an arbitrary number of initial or always behavior**
  - **Execute concurrently, launches at time t=0**

Example of clock

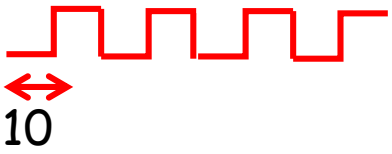
Example: a free-running clock for a specified number of cycles

```
initial
  begin
    clock = 1'b0;
    repeat (30)
      #10 clock = ~clock;
  end
```

```
initial
  begin
    clock = 1'b0;
  end

initial 300 $finish;
always #10 clock = ~clock;
```

次序不拘



Example: an infinitely free-running clock

```
initial begin clock = 0; forever #10 clock = ~clock; end
```

Delay control & event control operator

- **Delay control operator**
  - **Ex. #10**

- **Event control operator**
  - **Ex. @(posedge clock)**

```
always @ (event control expression) begin
    // Procedural assignment statements that execute when the condition is met
end
```

1. Assignment is made when the statement is executed
2. Left-hand side must be type **reg**

Always: level and edge sensitive

Examples:

```
// level sensitive: execute if a change occurs in A, B, or C

always @ (A or B or C)

// edge sensitive

always @(posedge clock or negedge reset)        // Verilog 1995

always @(posedge clock, negedge reset)          // Verilog 2001, 2005
```

Blocking and nonblocking procedural assignment

Two procedural blocking assignments:

```
B = A
C = B + 1
```

Execute sequentially in the order

C=A+1 after execution

Two nonblocking assignments:

```
B <= A
C <= B + 1
```

Execute concurrently

Right-hand side is evaluated and store in a temp. location

C="original value of B" +1 after execution

Outline

- **Ch5: Sequential logic**
  - **Basics of behavior modeling**
  - **HDL model of latches and flip-flops**
    - » **Example 5.2: D flip-flop**
    - » **Example 5.4: JK flip-flop**
  - **Mealy and Moore Models of finite state machine**
    - » **Example 5.5: Mealy Zero detector**
    - » **Example 5.6: Moore Model FSM**
    - » **Example 5.7: binary counter**

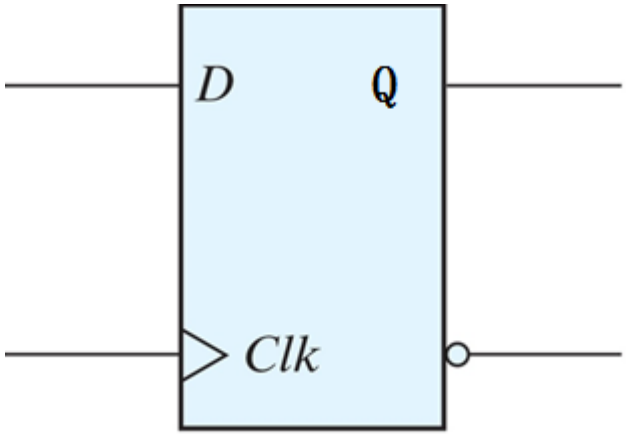
D Flip-Flops without reset

HDL Example 5.2

HDL Example 5.2

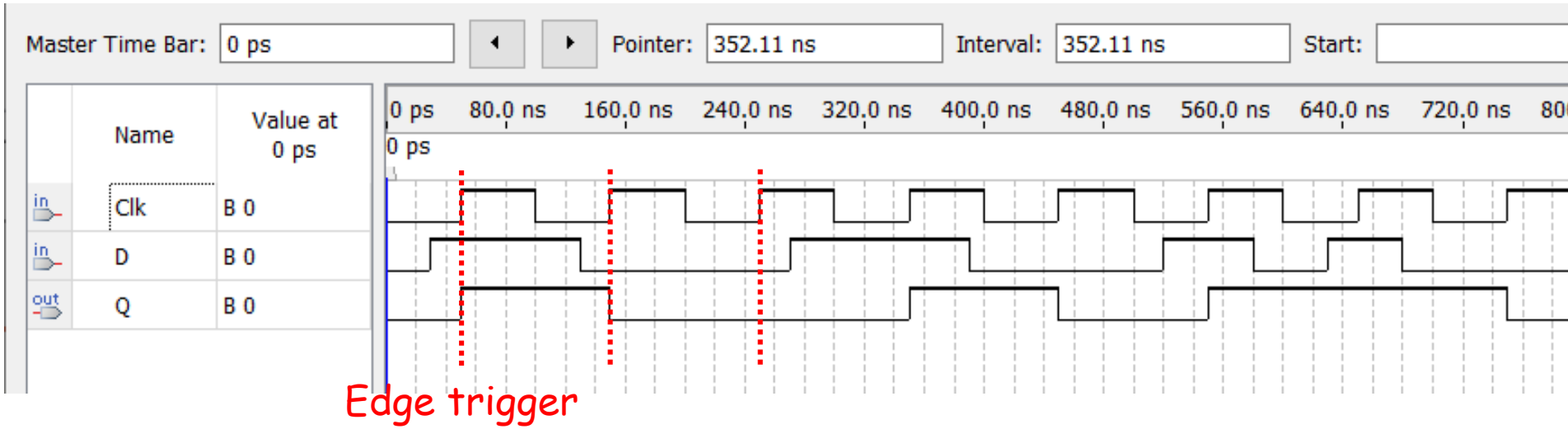
```
// D flip-flop without reset
module D_FF (Q, D, Clk);
  output Q;
  input  D, Clk;
  reg    Q;
  always @ (posedge Clk)
    Q <= D;
endmodule
```

// positive edge trigger



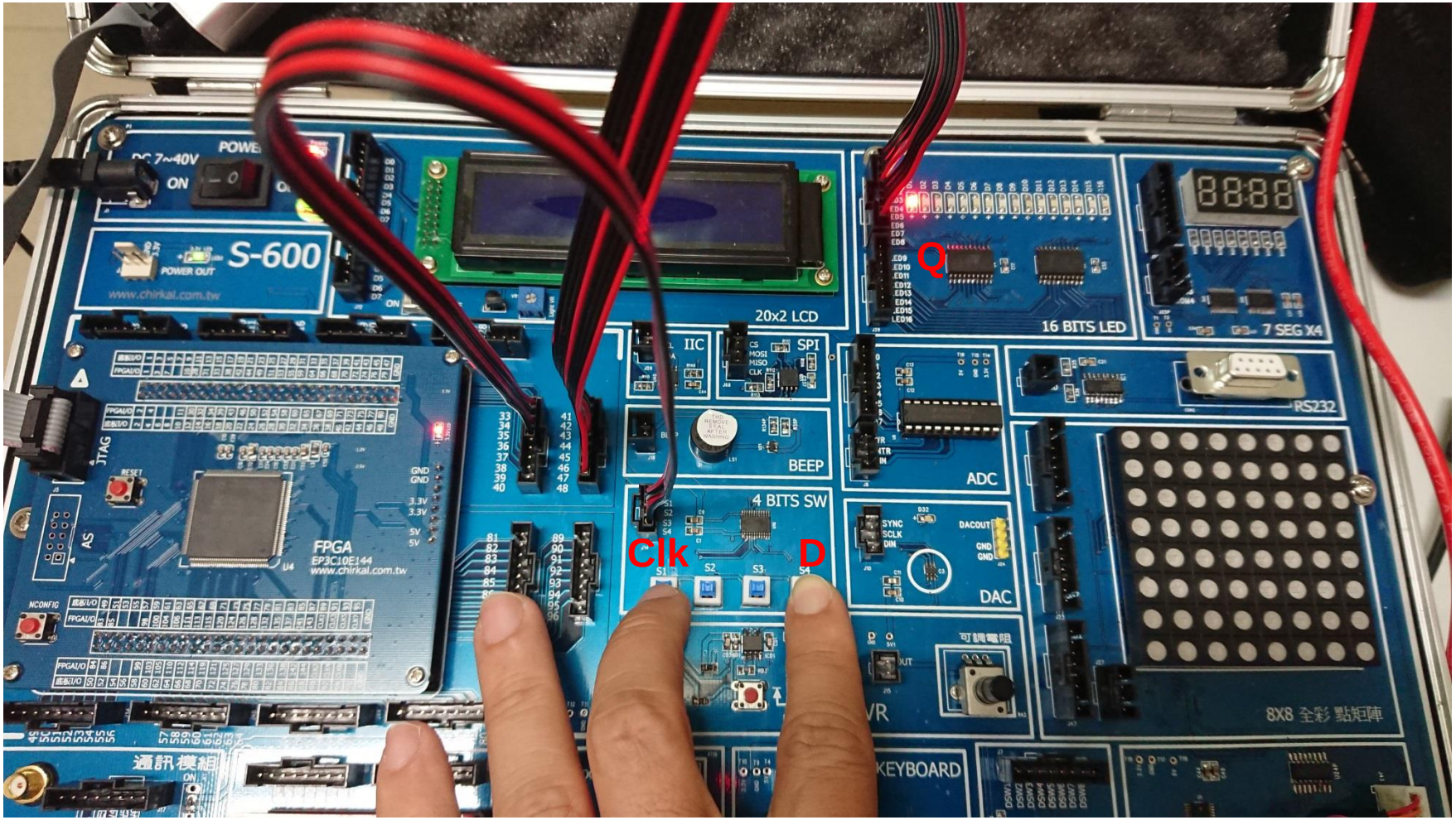
D-FF without reset: Simulation waveform

- **Edge-triggered by Clk**



* Clk 用週期性訊號，D 值用手選設定

Lab#1: D FF without reset



Outline

- **Ch5: Sequential logic**
  - **Basics of behavior modeling**
  - **HDL model of latches and flip-flops**
    - » **Example 5.2: D flip-flop**
    - » **Example 5.4: JK flip-flop**
  - **Mealy and Moore Models of finite state machine**
    - » **Example 5.5: Mealy Zero detector**
    - » **Example 5.6: Moore Model FSM**
    - » **Example 5.7: binary counter**

JK Flip-Flop

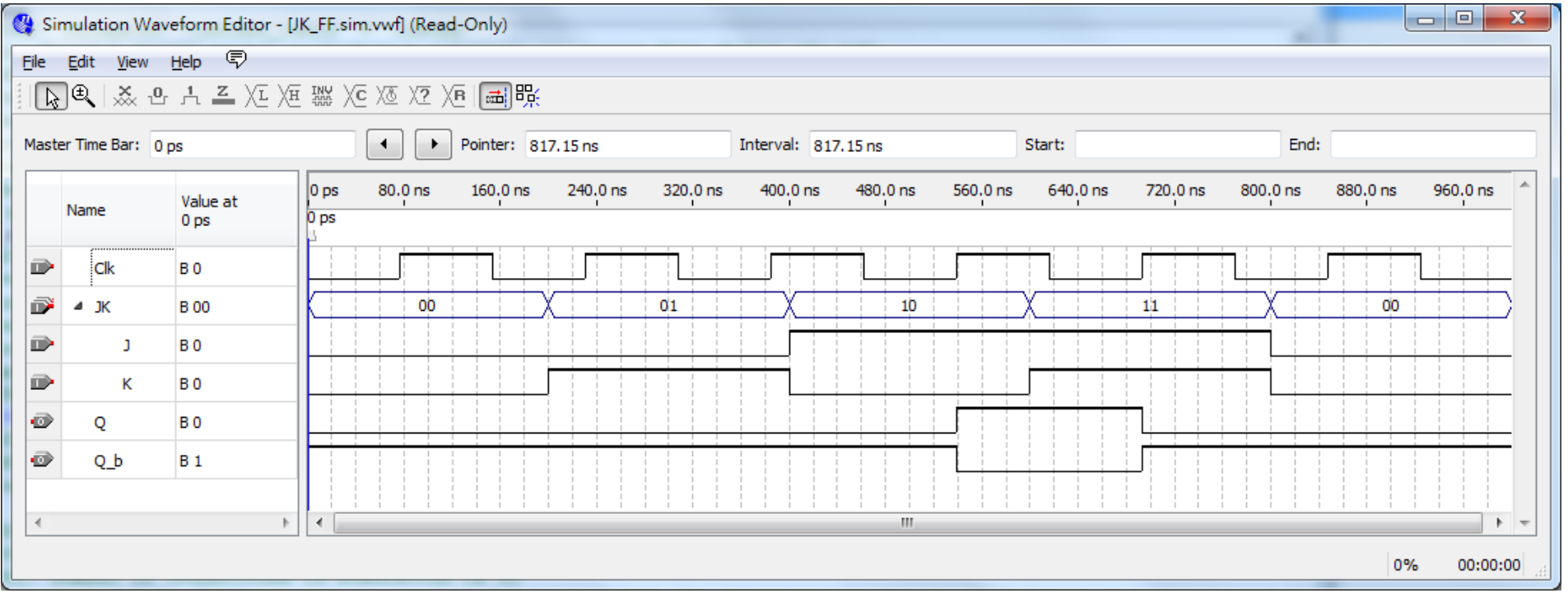
- Functional description of JK flip-flop

HDL Example 5.4

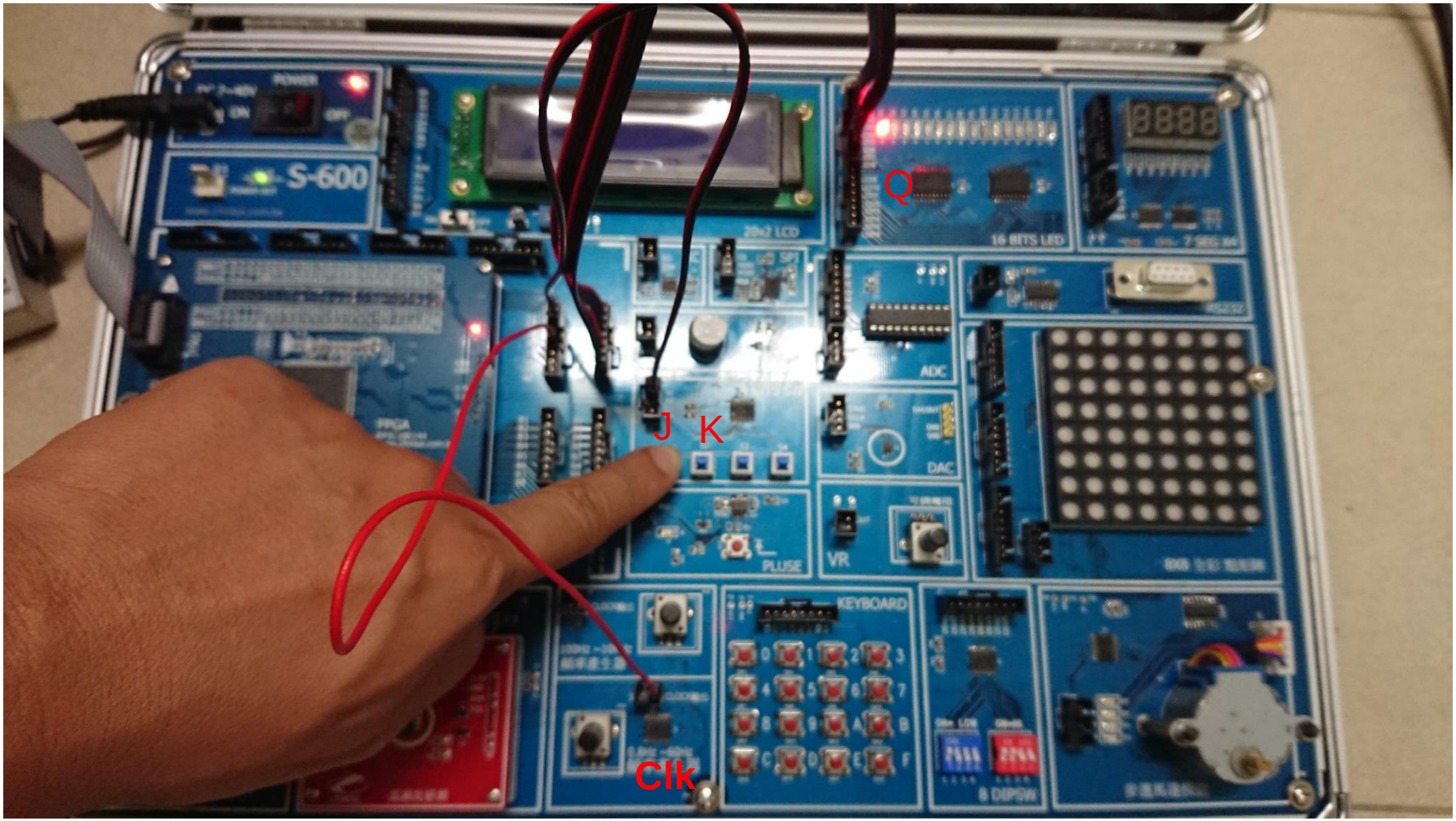
```
// Functional description of JK flip-flop (V2001, 2005)
module  JK_FF (input J, K, Clk, output reg Q, output Q_b);
  assign Q_b = ~ Q ;
  always @ (posedge Clk)
    case ({J,K})
      2'b00: Q <= Q;
      2'b01: Q <= 1'b0;
      2'b10: Q <= 1'b1;
      2'b11: Q <= ~Q;
    endcase
endmodule
```

J	K	$Q(t+1)$	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

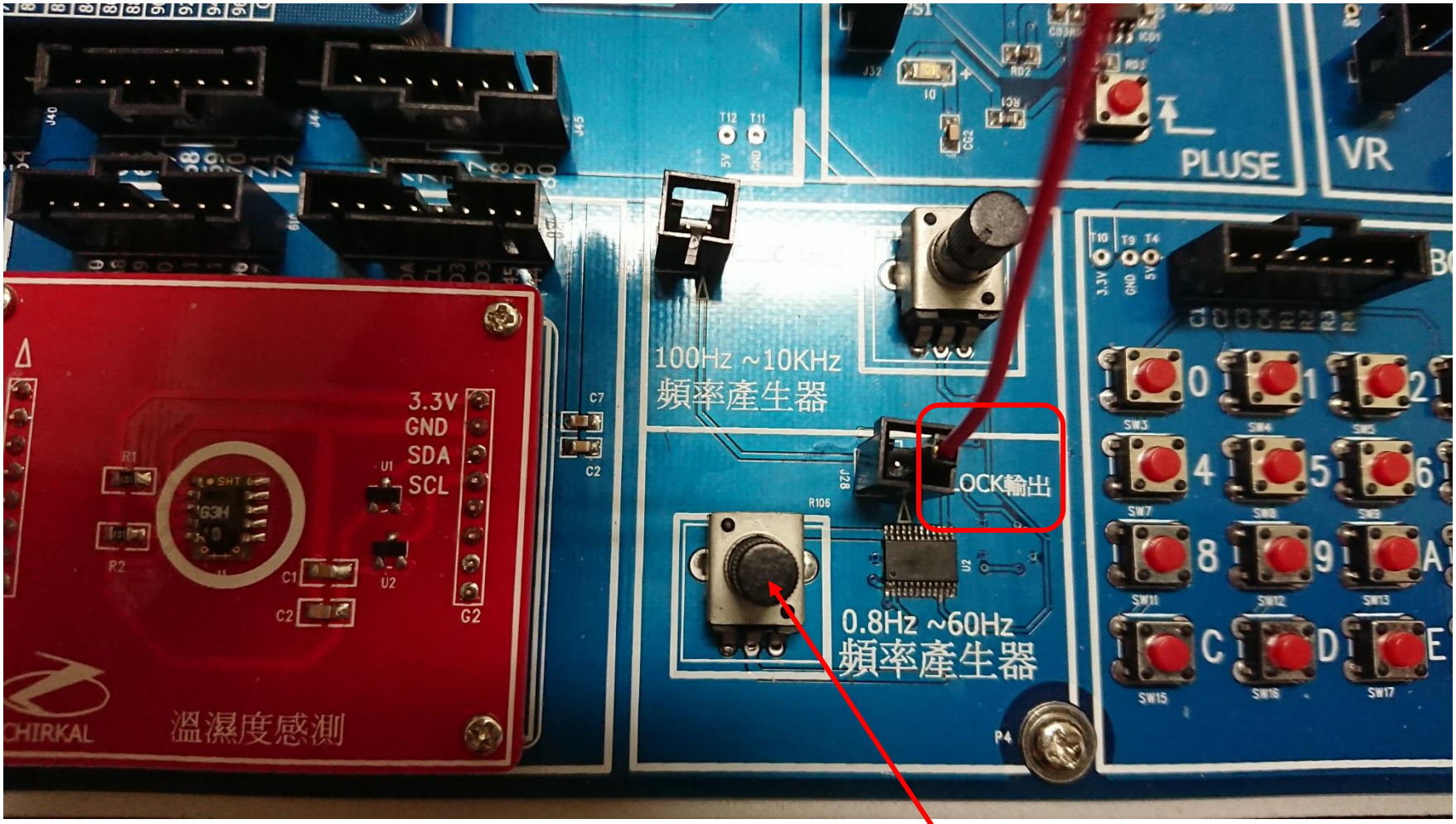
JK FF: waveform simulation



Lab#2: JK FF


S-600
20x2 LCD
16 BITS LED
Q
ADC
DAC
J
K
PLUSE
VR
KEYBOARD
8 DIPSW
Clk

使用頻率產生器當作 clock



調整頻率，右旋頻率變大

Outline

- **Ch5: Sequential logic**
  - **Basics of behavior modeling**
  - **HDL model of latches and flip-flops**
    - » **Example 5.2: D flip-flop**
    - » **Example 5.4: JK flip-flop**
  - **Mealy and Moore Models of finite state machine**
    - » **Example 5.5: Mealy Zero detector**
    - » **Example 5.6: Moore Model FSM**
    - » **Example 5.7: binary counter**

Mealy and Moore models

- **Mealy model: the outputs are functions of both the present state and inputs (Fig. 5-15)**
  - **the outputs may change if the inputs change during the clock pulse period**
    - » **the outputs may have momentary false values unless the inputs are synchronized with the clocks**
- **Moore model: the outputs are functions of the present state only (Fig. 5-20)**
  - **The outputs are synchronous with the clocks**

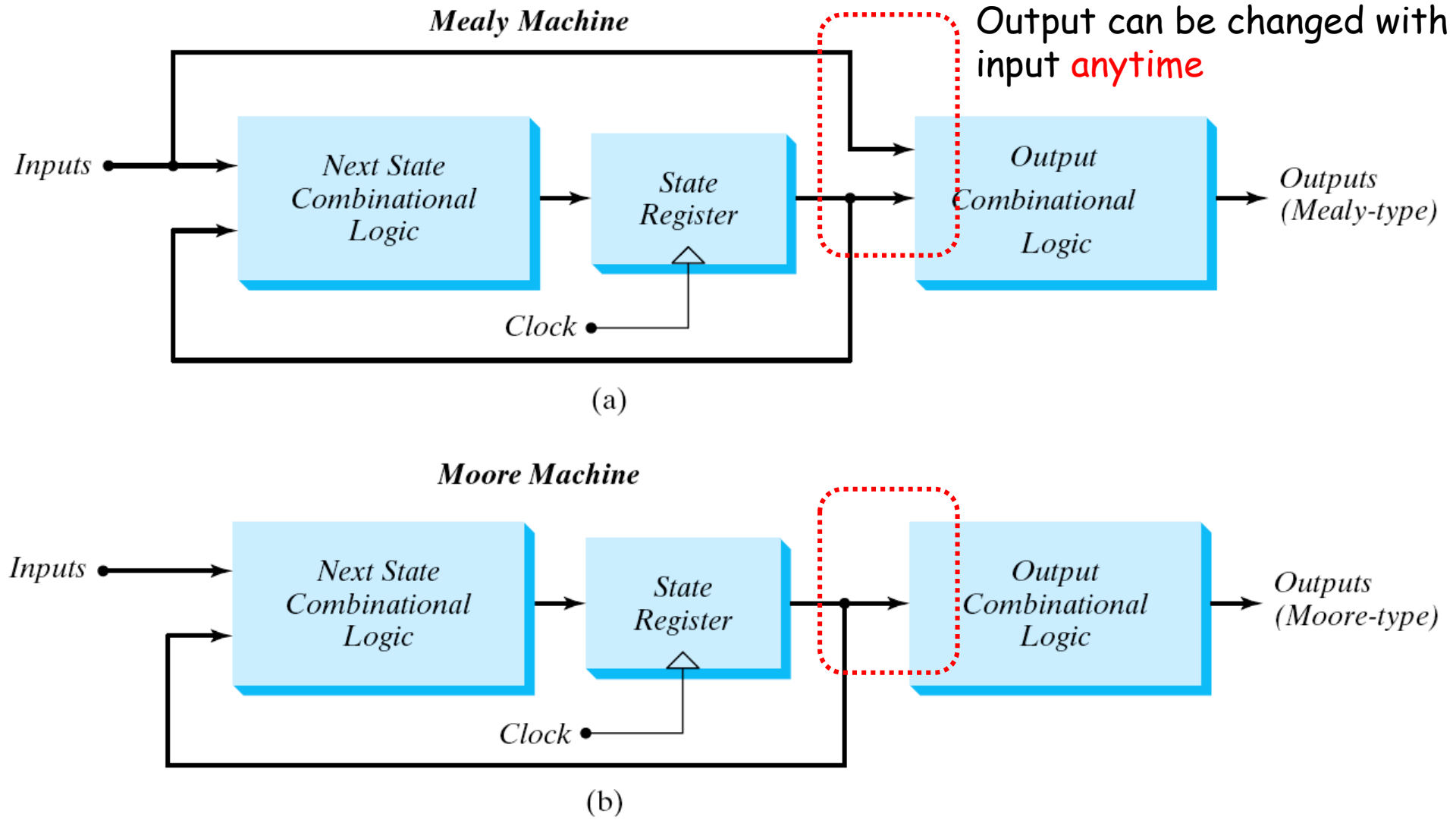
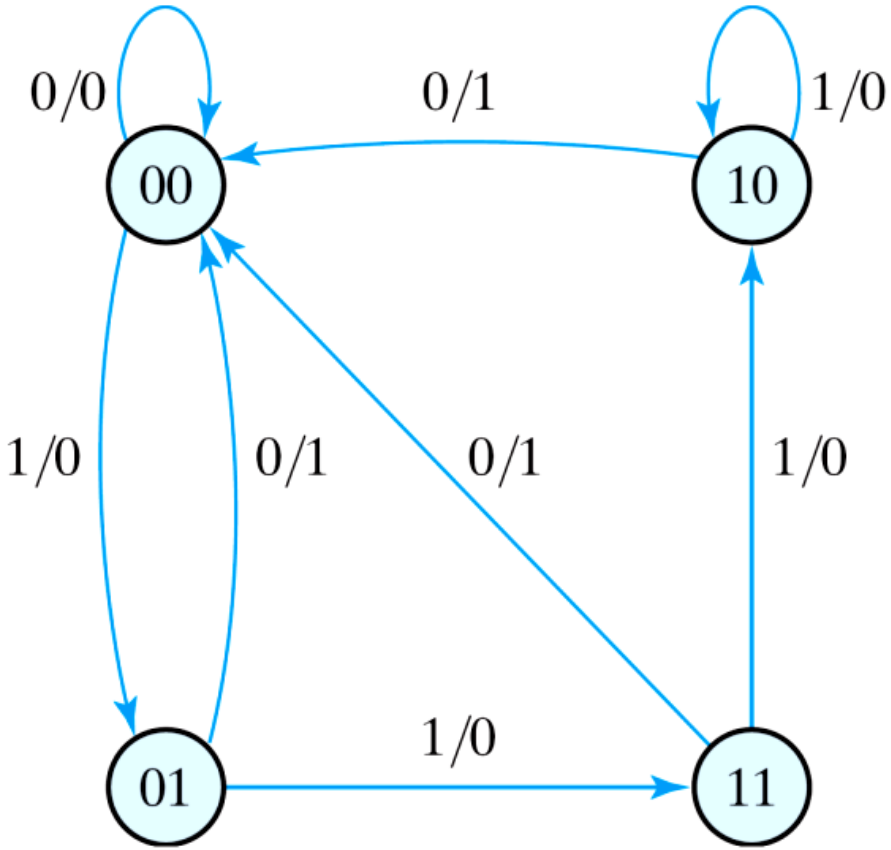


Fig. 5.21 Block diagram of Mealy and Moore state machine

Mealy machine
0-detector from a stream of 1's

- **State diagram**



state:

00 00 01 11 00

input:

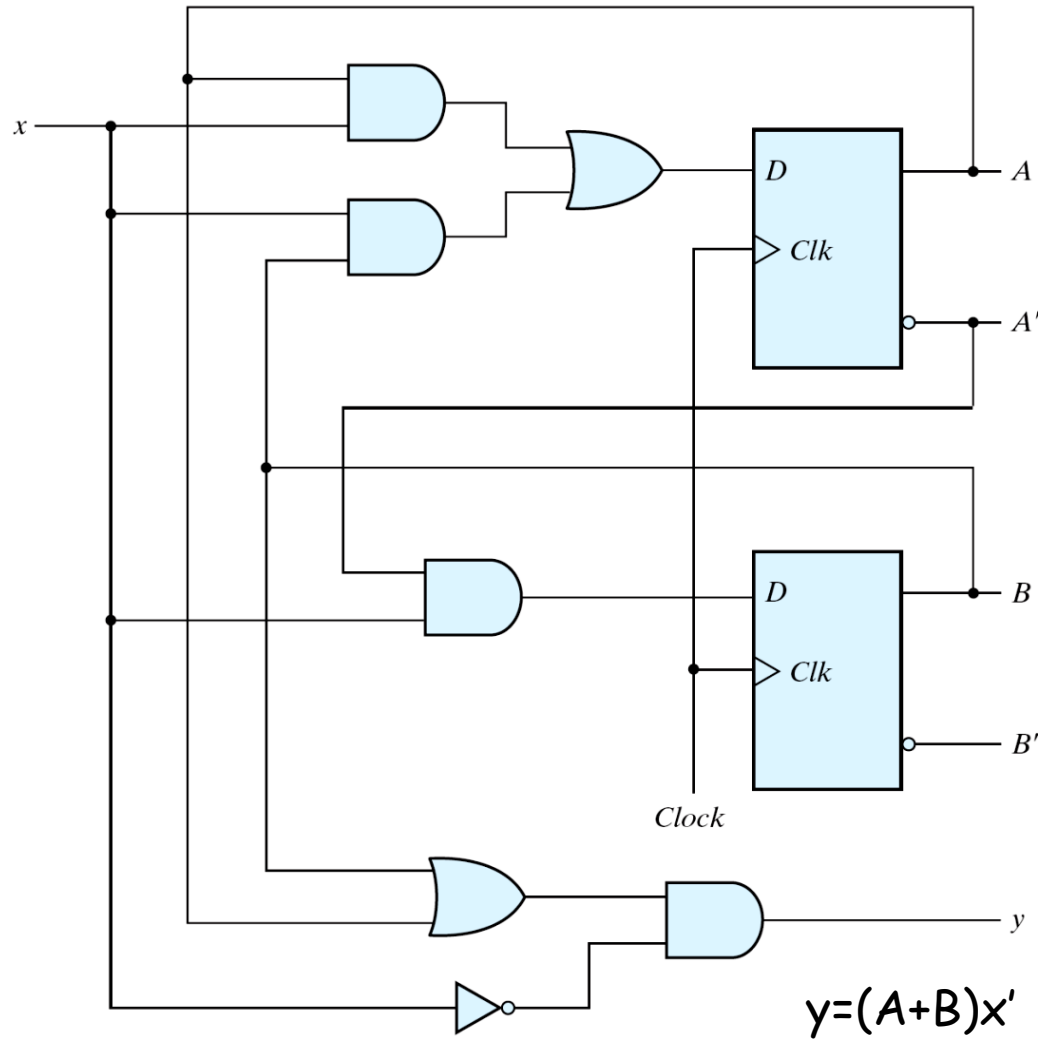
0 1 1 0

output:

0 0 0 1

Wrong!

In Lab#2 10/6



state:

00 00 01 11 00

input:

0 1 1 0

output:

0 0 0 0

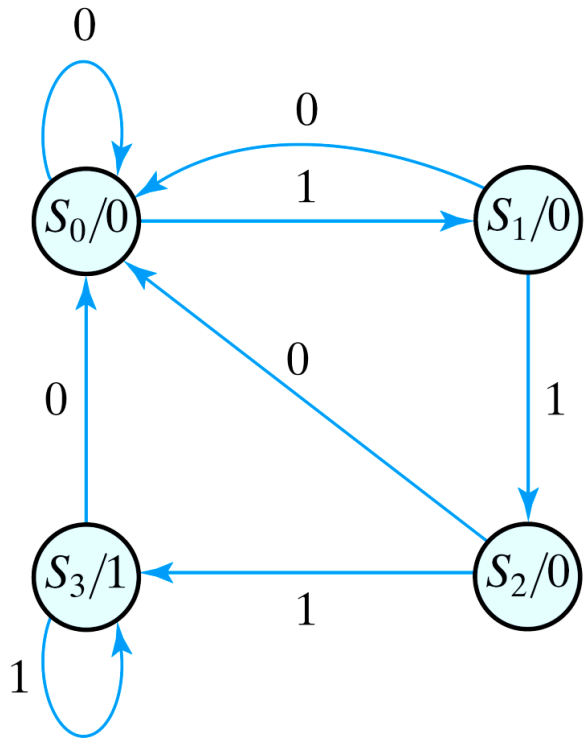
input:

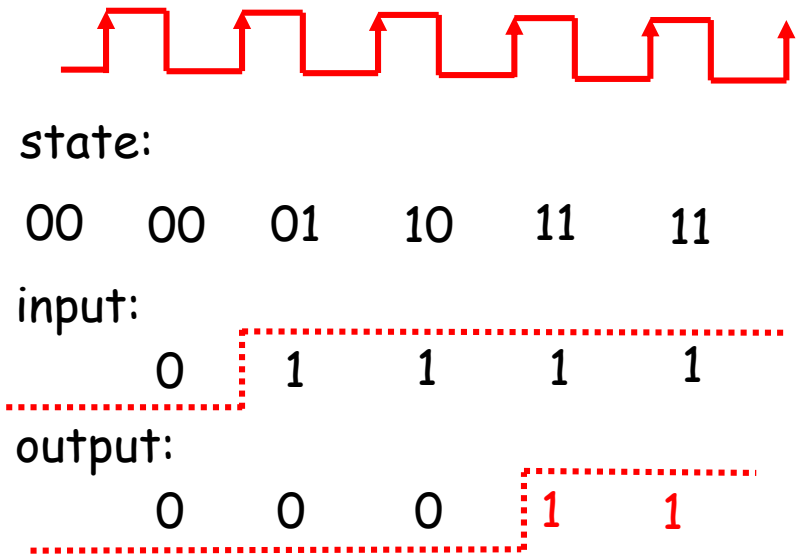
0 1 1 0

output:

0 0 0 1 0

Moore Machine sequence of three or more consecutive 1's





Outline

- **Ch5: Sequential logic**
  - **Basics of behavior modeling**
  - **HDL model of latches and flip-flops**
    - » **Example 5.2: D flip-flop**
    - » **Example 5.4: JK flip-flop**
  - **Mealy and Moore Models of finite state machine**
    - » **Example 5.5: Mealy Zero detector**
    - » **Example 5.6: Moore Model FSM**
    - » **Example 5.7: binary counter**

*** 3 behavior models for each components

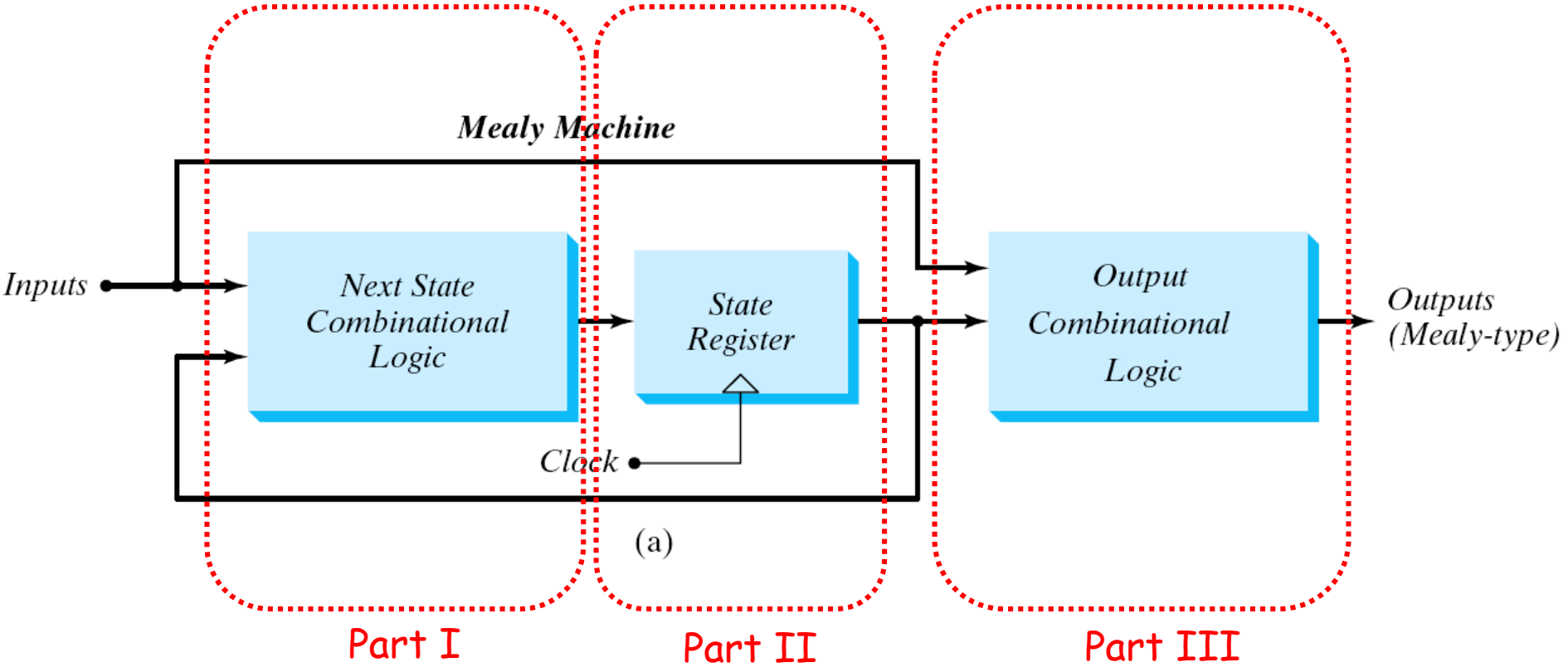


Fig. 5.21 Block diagram of Mealy and Moore state machine

HDL Example 5-5

HDL Example 5.5: Mealy HDL model

HDL Example 5.5

```
// Mealy FSM zero detector (See Fig. 5.16)                    Verilog 2001, 2005 syntax
module Mealy_Zero_Detector (
  output reg y_out,
  input      x_in, clock, reset
);
  reg [1: 0]          state, next_state;    // for 4 states
  parameter           S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;  // constants
  always @ (posedge clock, negedge reset)   // Verilog 2001, 2005 syntax
    if (reset == 0) state <= S0;
    else state <= next_state;
```

Part II

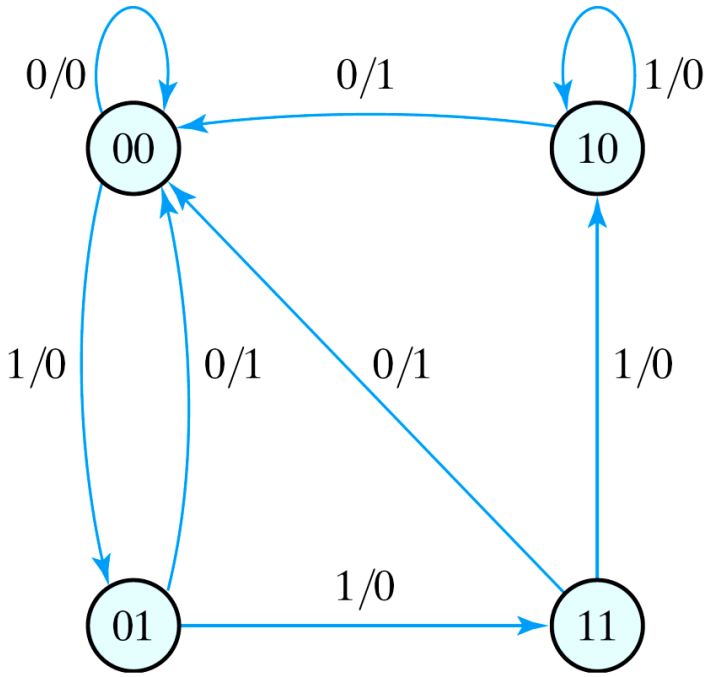
HDL Example 5-5 (Continued)

```
always @ (state, x_in)                          // Form the next state
  case (state)
    S0:       if (x_in)   next_state = S1; else next_state = S0;
    S1:       if (x_in)   next_state = S3; else next_state = S0;
    S2:       if (~x_in)  next_state = S0; else next_state = S2;
    S3:       if (x_in)   next_state = S2; else next_state = S0;
  endcase
```

Part I

```
always @ (state, x_in)                          // Form the output
  case (state)
    S0:           y_out = 0;
    S1, S2, S3: y_out = ~x_in;
  endcase
endmodule
```

Part III




```
module t_Mealy_Zero_Detector;                   // test bench
  wire    t_y_out;
  reg     t_x_in, t_clock, t_reset;

Mealy_Zero_Detector M0 (t_y_out, t_x_in, t_clock, t_reset);
initial #200 $finish;
initial begin t_clock = 0; forever #5 t_clock = ~t_clock; end
                                                // clock
```

HDL Example 5-5 (Continued)

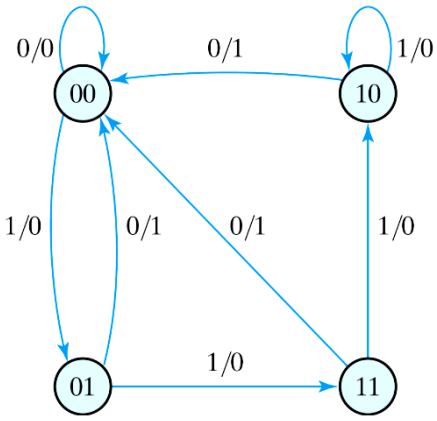
```
initial fork
      t_reset = 0;
  #2 t_reset = 1;
  #87 t_reset = 0;
  #89 t_reset = 1;
  #10 t_x_in = 1;
  #30 t_x_in = 0;
  #40 t_x_in = 1;
  #50 t_x_in = 0;
  #52 t_x_in = 1;
  #54 t_x_in = 0;
  #70 t_x_in = 1;
  #80 t_x_in = 1;
  #70 t_x_in = 0;
  #90 t_x_in = 1;
  #100 t_x_in = 0;
  #120 t_x_in = 1;
  #160 t_x_in = 0;
  #170 t_x_in = 1;
  join
endmodule
```

// fork … join 內的敘述都對應到 t=0 的時間

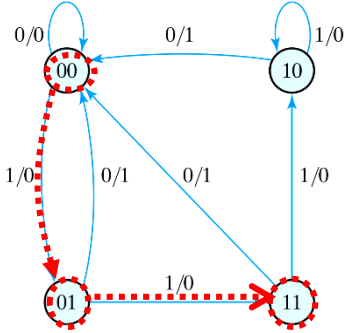
兩個一樣的?

State diagram=>code
可合併PartI+PartIII

```
always @(state, x_in)
   case(state)
      S0: if(x_in) begin next_state=S1; y_out=1'b0; end
          else begin next_state=S0; y_out=1'b0; end
      S1: if(x_in) begin next_state=S3; y_out=1'b0; end
          else begin next_state=S0; y_out=1'b1; end
      S2: if(!x_in) begin next_state=S0; y_out=1'b1; end
          else begin next_state=S2; y_out=1'b0; end
      S3: if(x_in) begin next_state=S2; y_out=1'b0; end
          else begin next_state=S0; y_out=1'b1; end
   endcase
```



Mealy_Zero_Detector



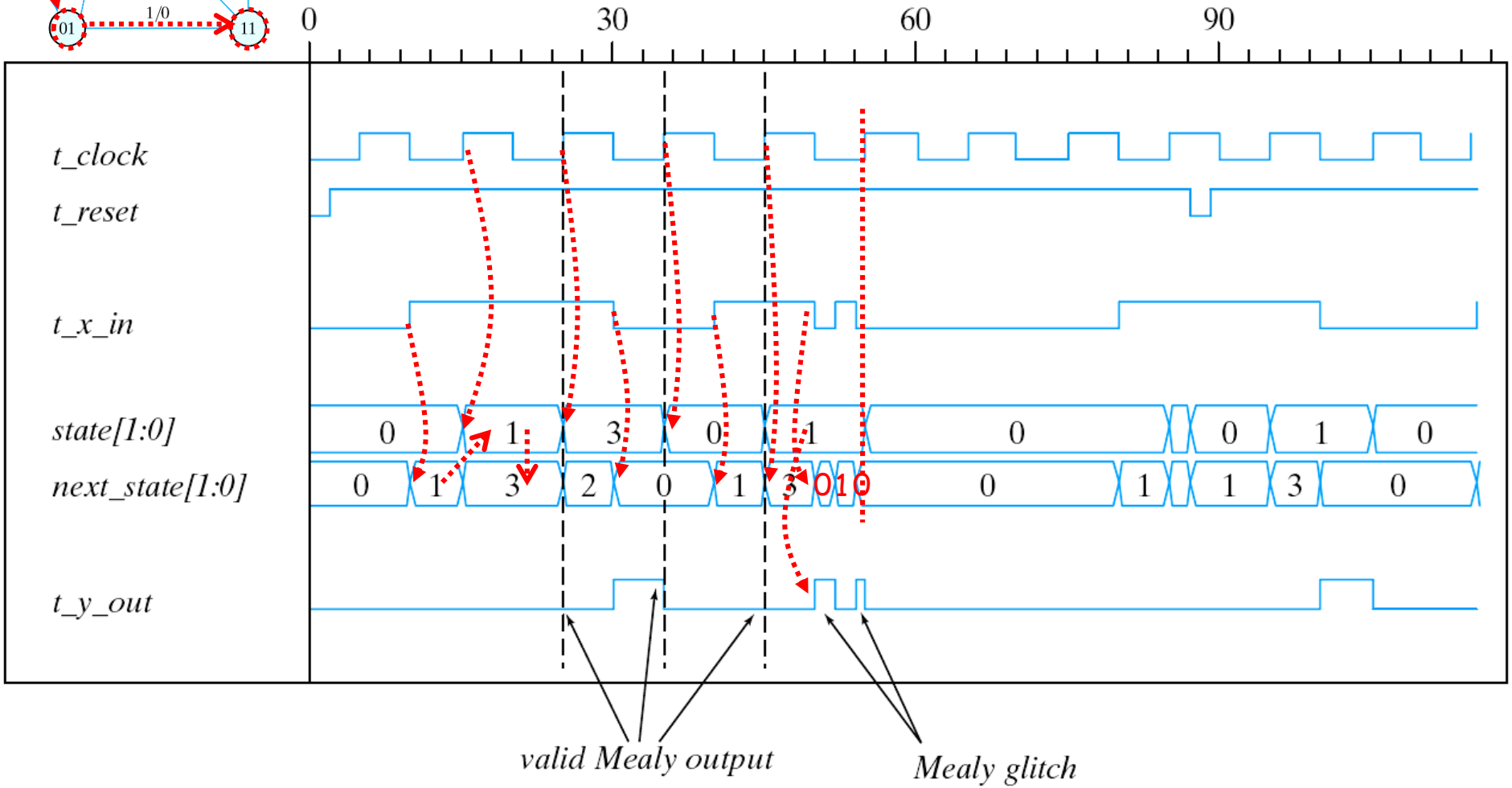
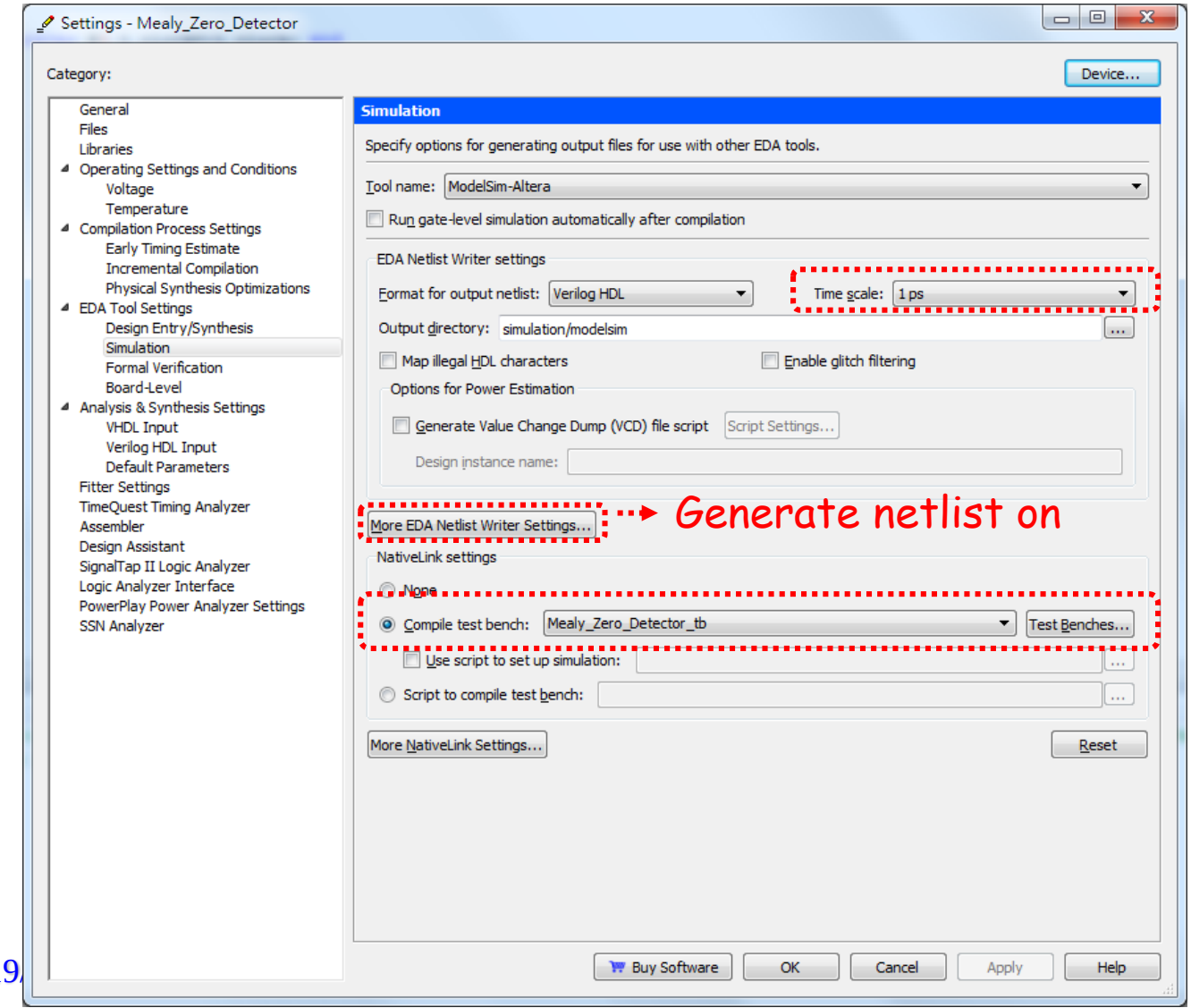


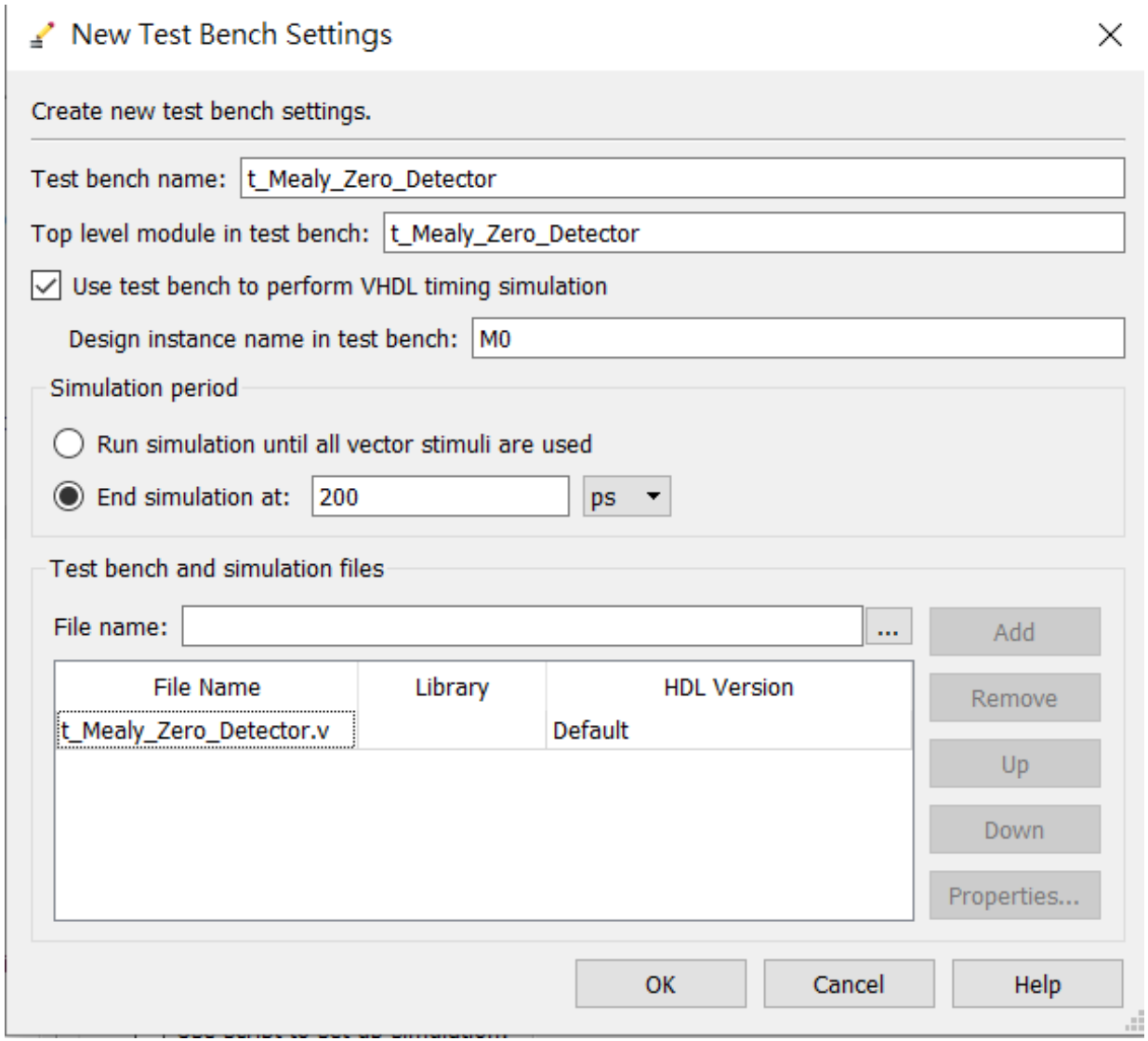
Fig. 5.22 Simulation output of *Mealy_Zero_Detector*

Run test bench on ModelSim(1)

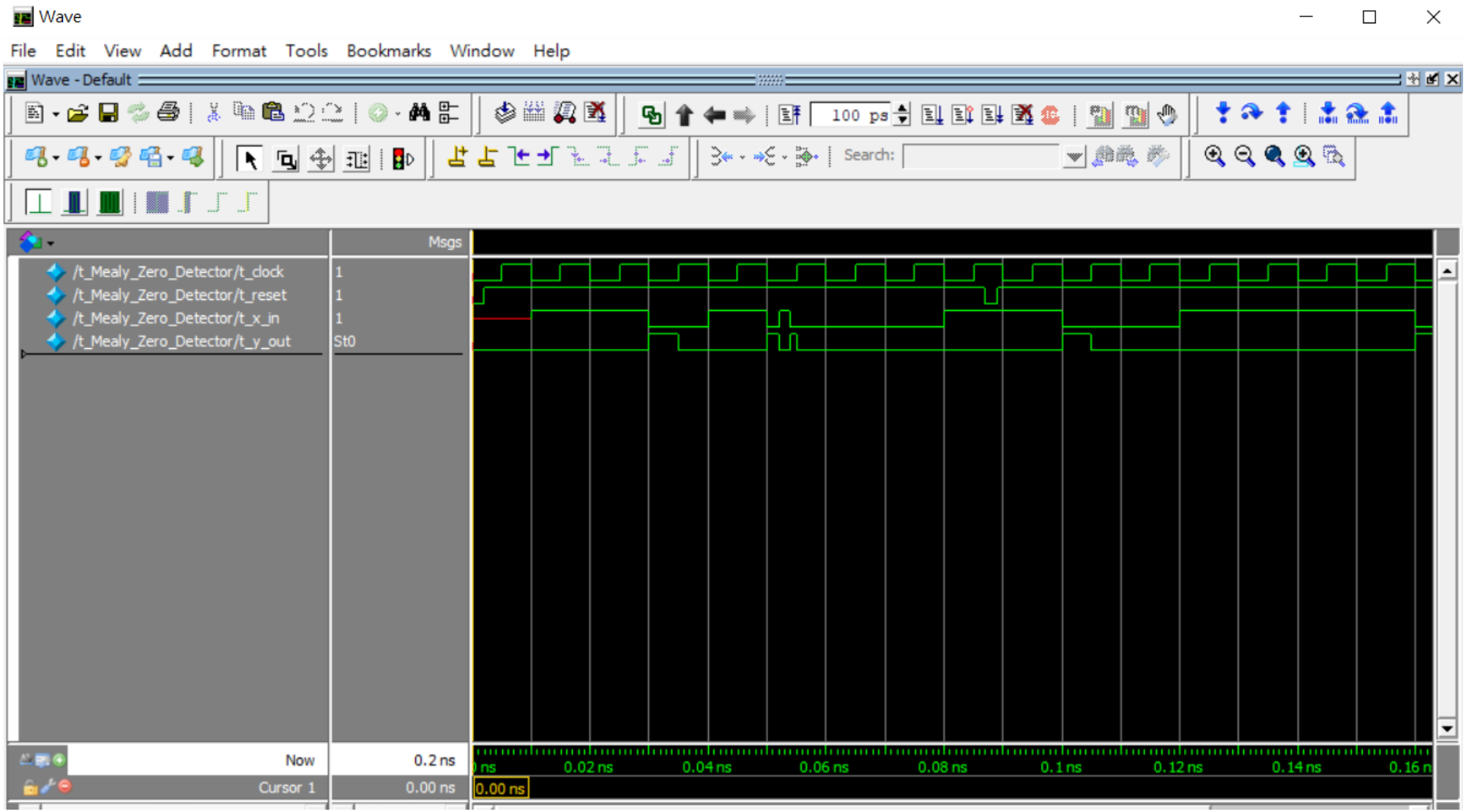
- **Assignment -> setting**



Run test bench on ModelSim(2)



Lab3-1: Run test bench on ModelSim (3)



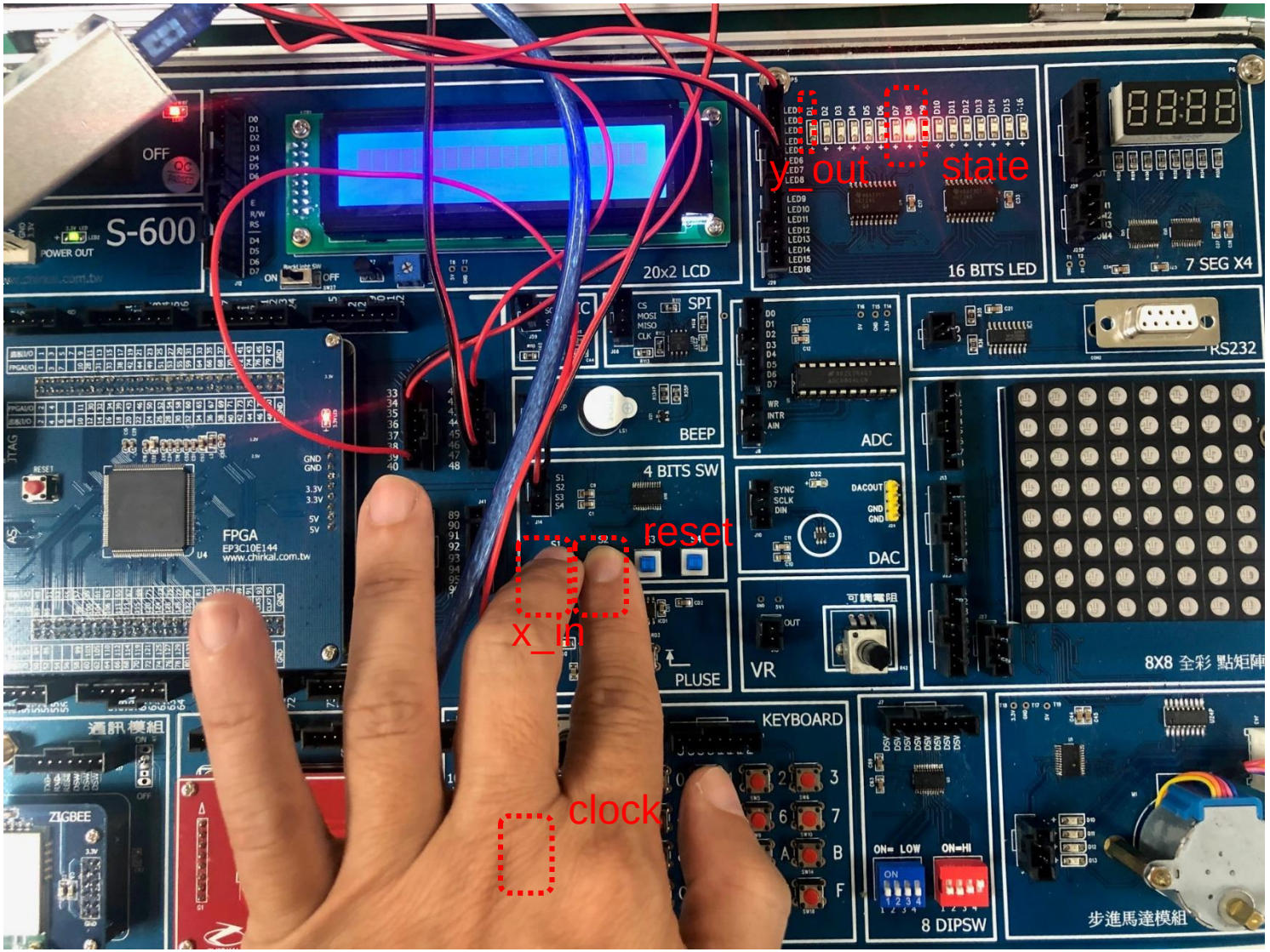
Lab3-2: FPGA

- Modify your code to let the state output to leds
- module Mealy_Zero_Detector(output reg y_out, output s1,s0, input x_in, clock, reset);

```
…
        assign s1 = state[1];
        assign s0 = state[0];
…
```

- Caution: reset must be 1 to let the system work
- 使用頻率產生器當作 clock

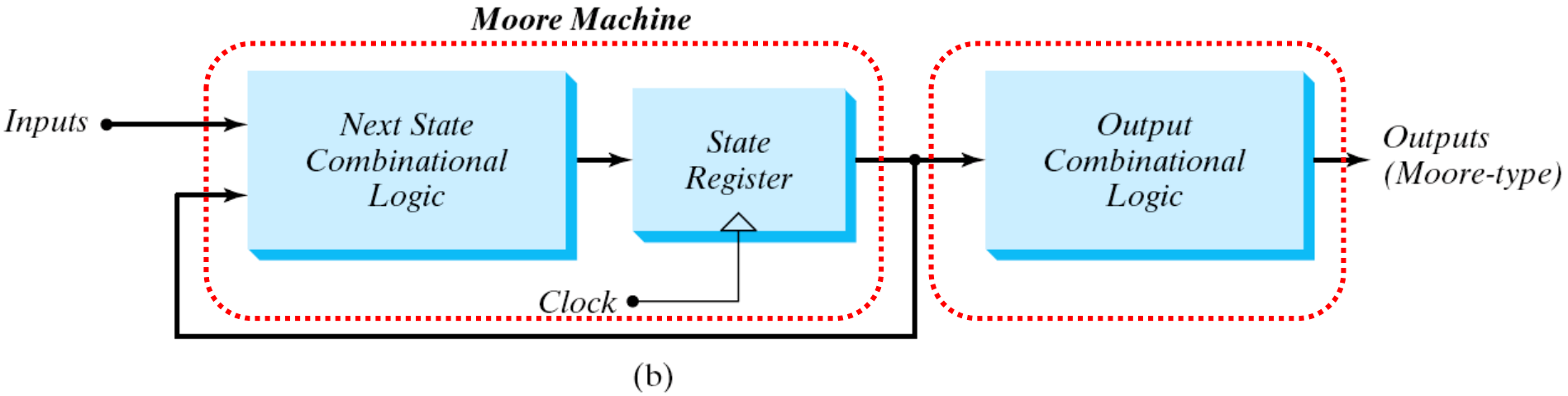
Lab3-2: FPGA


y_out
state
reset
x_in
clock
S-600
20x2 LCD
16 BITS LED
7 SEG X4
SPI
BEEP
ADC
RS232
4 BITS SW
DAC
FPGA
EP3C10E144
www.chirkal.com.tw
VR
PLUSE
KEYBOARD
8X8 全彩 點矩陣
ZIGBEE
8 DIPSW
步進馬達模組

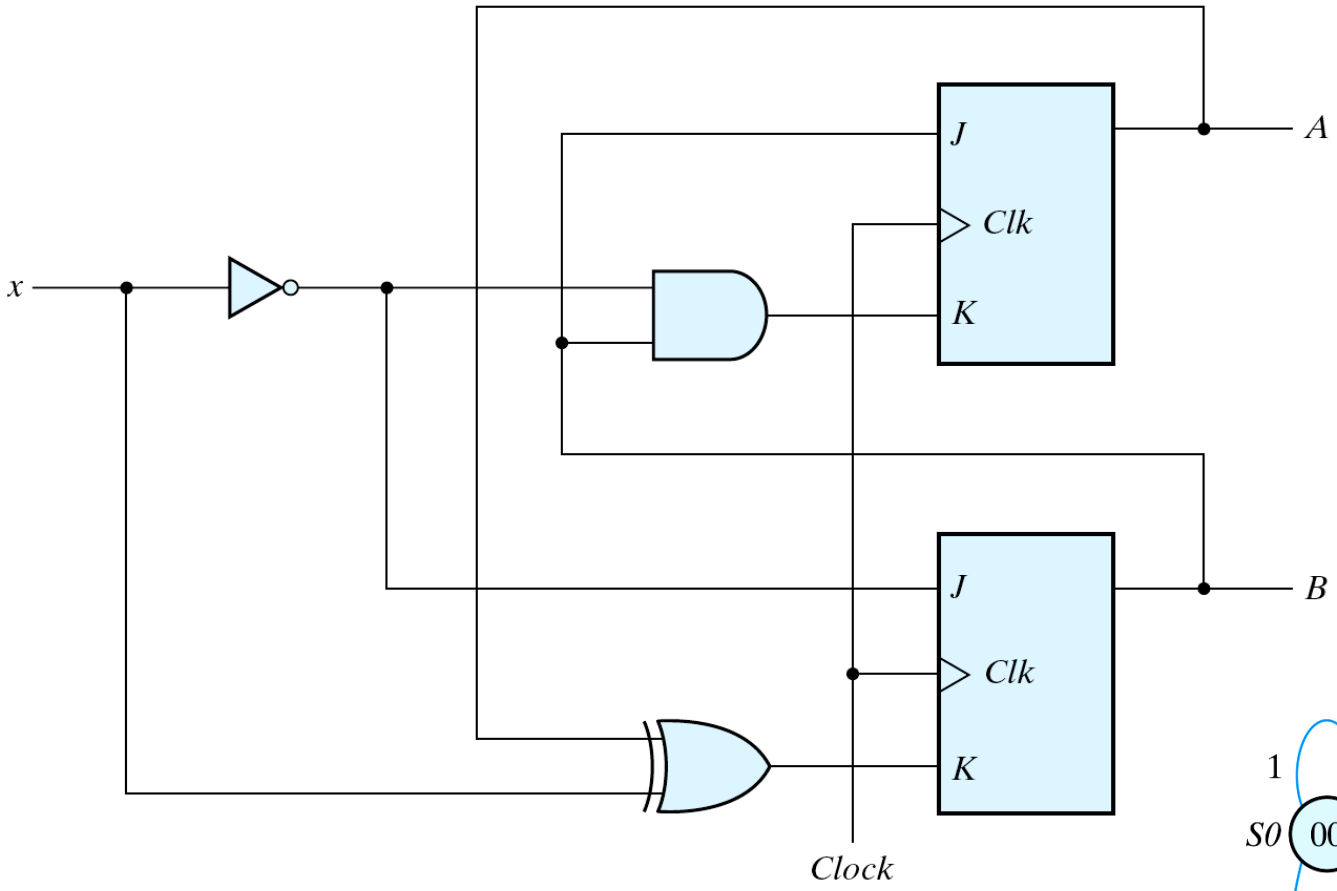
Outline

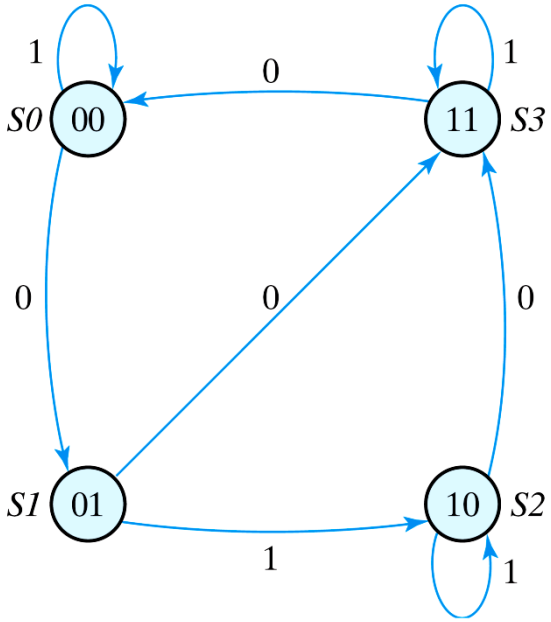
- **Ch5: Sequential logic**
  - **Basics of behavior modeling**
  - **HDL model of latches and flip-flops**
    - » **Example 5.2: D flip-flop**
    - » **Example 5.4: JK flip-flop**
  - **Mealy and Moore Models of finite state machine**
    - » **Example 5.5: Mealy Zero detector**
    - » **Example 5.6: Moore Model FSM**
    - » **Example 5.7: binary counter**

Moore machine



(b)





此例中 Output = state

沒有獨立的輸出電路

HDL Example 5-6: *Moore Model FSM*

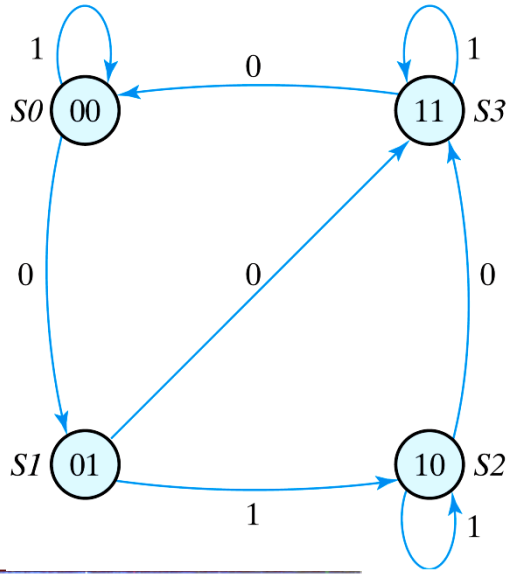
HDL Example 5.6

```
// Moore model FSM (see Fig. 5.19)                    Verilog 2001, 2005 syntax
module Moore_Model_Fig_5_19 (
  output [1: 0]          y_out,
  input                  x_in, clock, reset
);
  reg [1: 0]             state;
  parameter              S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;

  always @ (posedge clock, negedge reset)
    if (reset == 0) state <= S0;                      // Initialize to state S0
    else case (state)
      S0:    if (~x_in)  state <= S1; else state <= S0;
      S1:    if (x_in)   state <= S2; else state <= S3;
      S2:    if (~x_in)  state <= S3; else state <= S2;
      S3:    if (~x_in)  state <= S0; else state <= S3;
    endcase

  assign y_out = state;      // Output of flip-flops

endmodule
```



Simulation Output of HDL Example 5-6

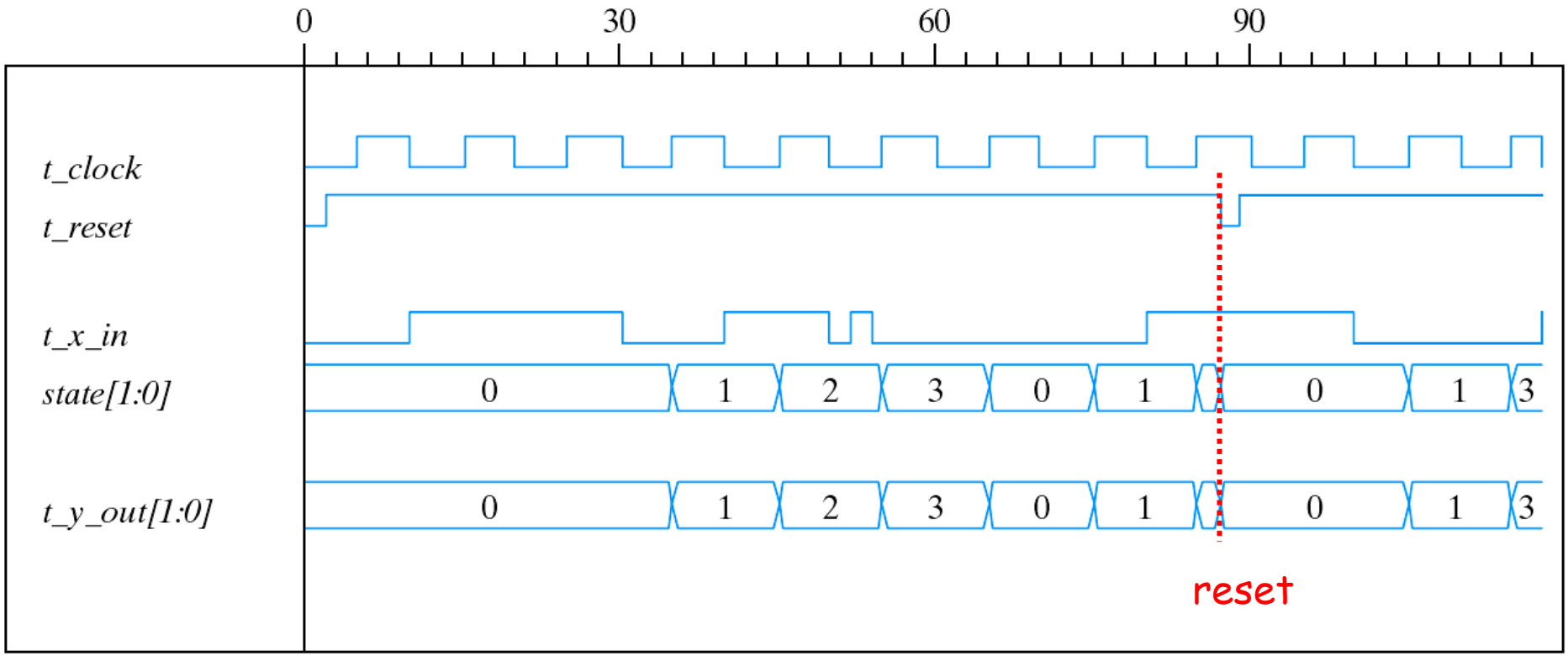
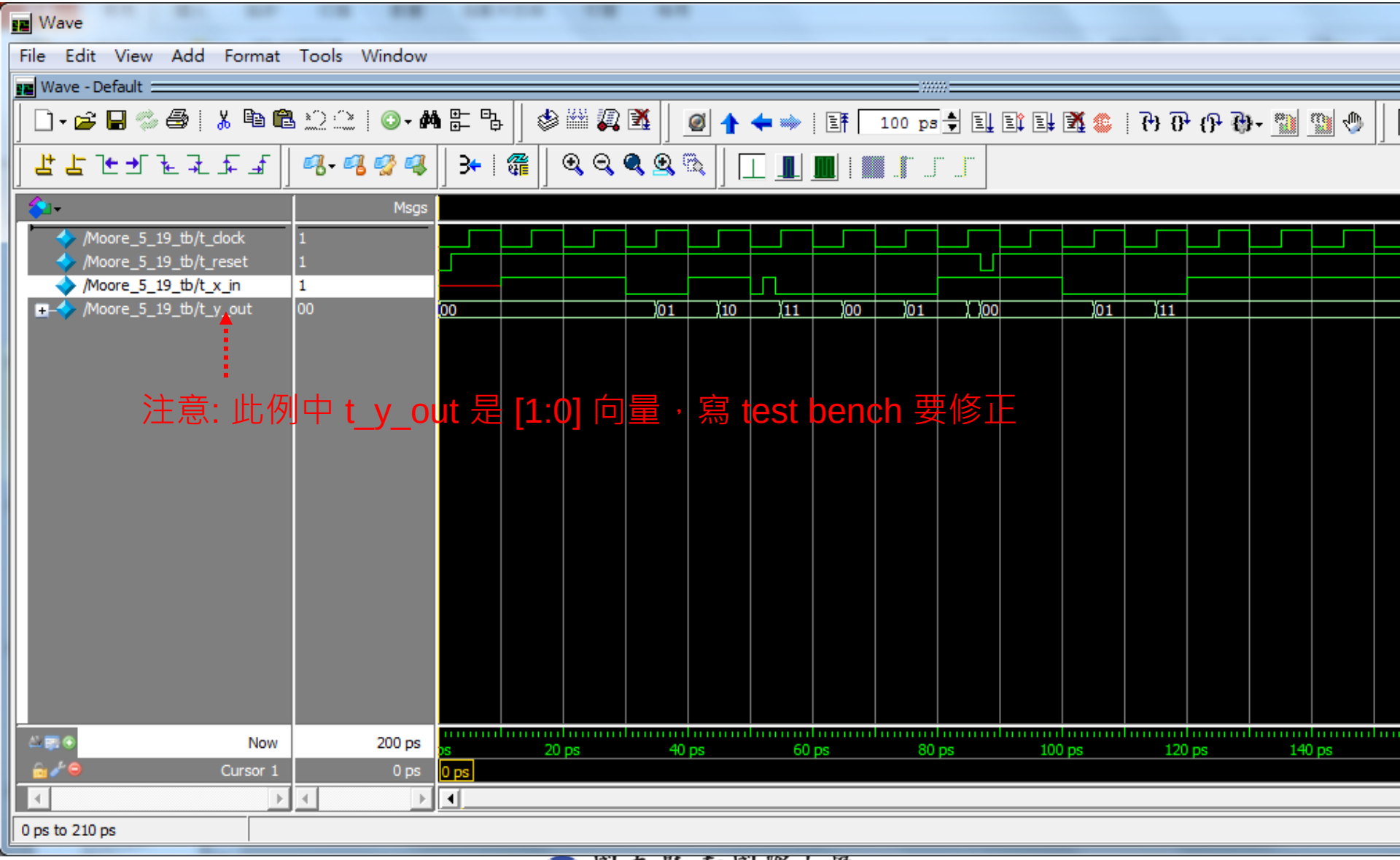


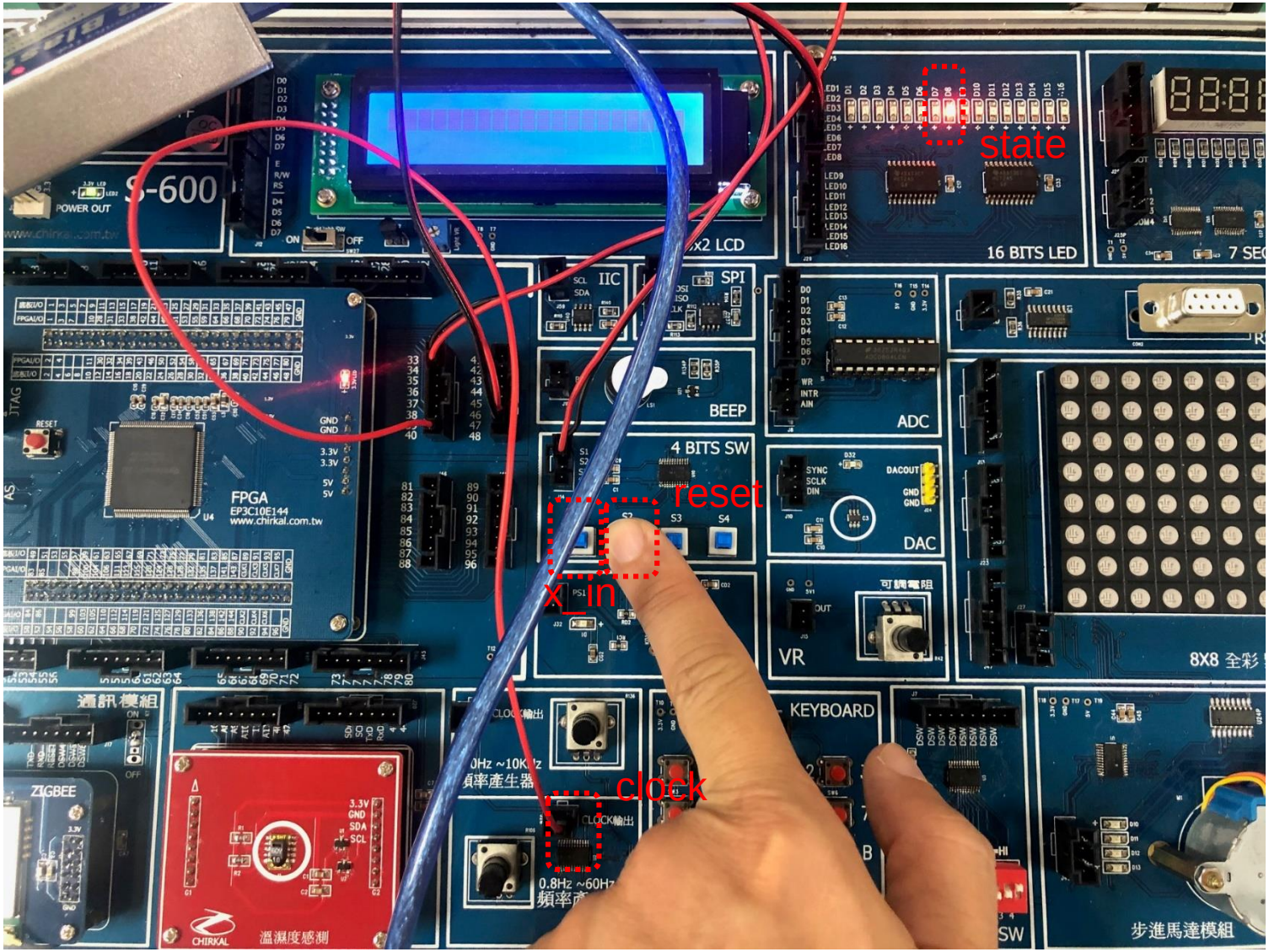
Fig. 5.23 Simulation output of HDL Example 5.6

Lab4-1: ModelSim results

// use the same test bench input sequence in Example 5-5



Lab4-2: FPGA

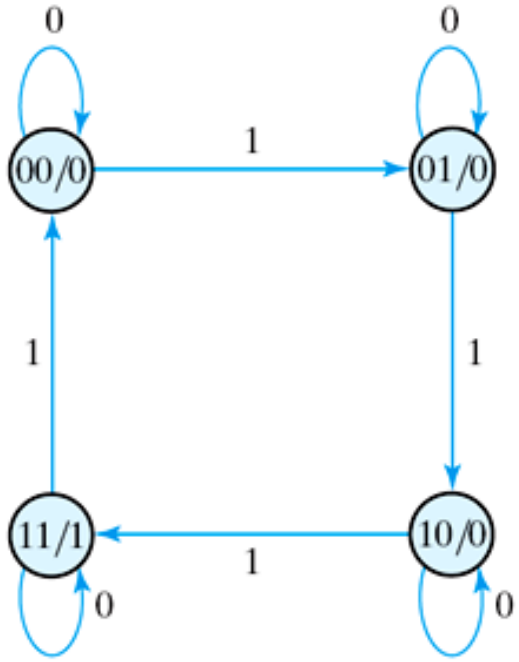

state
reset
x_in
clock
16 BITS LED
4 BITS SW
BEEP
ADC
DAC
VR
KEYBOARD
FPGA
EP3C10E144
www.chirkal.com.tw
ZIGBEE
溫濕度感測
步進馬達模組

Outline

- **Ch5: Sequential logic**
  - **Basics of behavior modeling**
  - **HDL model of latches and flip-flops**
    - » **Example 5.2: D flip-flop**
    - » **Example 5.4: JK flip-flop**
  - **Mealy and Moore Models of finite state machine**
    - » **Example 5.5: Mealy Zero detector**
    - » **Example 5.6: Moore Model FSM**
    - » **Example 5.7: binary counter**

Structural Description of Clocked Sequential Circuits

- HDL Example 5.7: Binary Counter



HDL Example 5.7

```
// State-diagram-based model (V2001, 2005)
module Moore_Model_Fig_5_20 (
  output  y_out,
  input   x_in, clock, reset
);
  reg [1: 0]          state;
  parameter           S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;

  always @ (posedge clock, negedge reset)
    if (reset == 0) state <= S0;                    // Initialize to state S0
    else case (state)
      S0:   if (x_in)  state <= S1; else state <= S0;
      S1:   if (x_in)  state <= S2; else state <= S1;
```

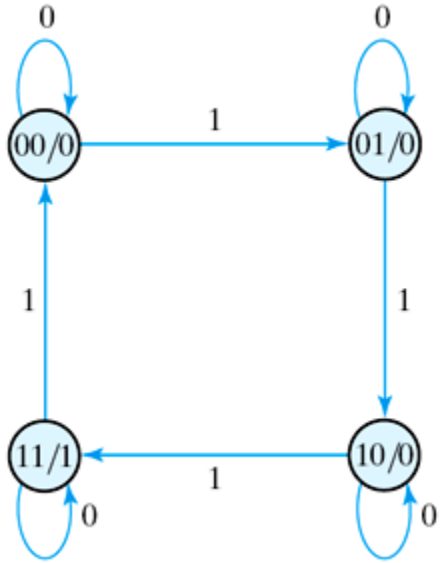
HDL Example 5-7 (Continued)

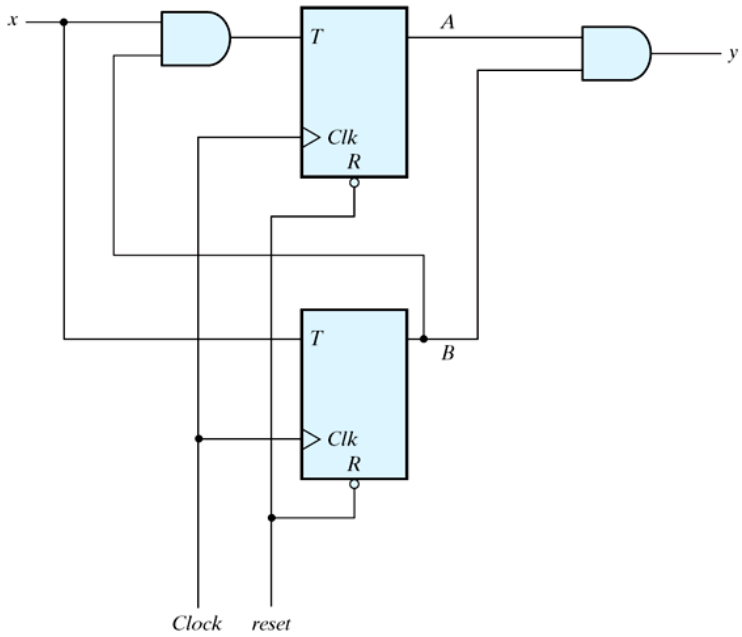
```
    S2:    if (x_in)  state <= S3; else state <= S2;
    S3:    if (x_in)  state <= S0; else state <= S3;
   endcase

  assign y_out = (state == S3);          // Output of flip-flops
endmodule

// structural model   // data-flow + behavior model
module Moore_Model_STR_Fig_5_20 (
  output   y_out, A, B,
  input    x_in, clock, reset
);
  wire     TA, TB;

// Flip-flop input equations
  assign TA = x_in & B;
  assign TB = x_in;
// Output equation
  assign y_out = A & B;
```





HDL Example 5-7 (Continued)

```
// Instantiate Toggle flip-flops
  Toggle_flip_flop  M_A (A, TA, clock, reset);
  Toggle_flip_flop  M_B (B, TB, clock, reset);
endmodule
```

T-FF

```
module Toggle_flip_flop (Q, T, CLK, RST_b);
  output   Q;
  input    T, CLK, RST_b;
  reg      Q;

  always @ (posedge CLK, negedge RST_b)
    if (RST_b == 0) Q <= 1'b0;
    else if (T) Q <= ~Q;
endmodule
```

```
// Alternative model using characteristic equation
//  module Toggle_flip_flop (Q, T, CLK, RST_b);
//  output  Q;
//  input   T, CLK, RST_b;
//  reg     Q;
```

HDL Example 5-7 (Continued)

```
// always @ (posedge CLK, negedge RST_b)
//   if (RST_b == 0) Q <= 1'b0;
//   else  Q <= Q ^ T;
// endmodule
```

```
module t_Moore_Fig_5_20;    // test bench
 wire              t_y_out_2, t_y_out_1;
 reg               t_x_in, t_clock, t_reset;
 wire A, B;
 Moore_Model_Fig_5_20          M1(t_y_out_1, t_x_in, t_clock, t_reset);
 Moore_Model_STR_Fig_5_20      M2 (t_y_out_2, A, B, t_x_in, t_clock, t_reset);
```

HDL Example 5-7 (Continued)

```
initial #200 $finish;
initial begin
    t_reset = 0;
    t_clock = 0;
    #5 t_reset = 1;
  repeat (16)
    #5 t_clock = ~t_clock;
end
initial begin
        t_x_in = 0;
  #15 t_x_in = 1;
  repeat (8)
    #10 t_x_in = ~t_x_in;
end
endmodule
```

Simulation Output of HDL Example 5-7

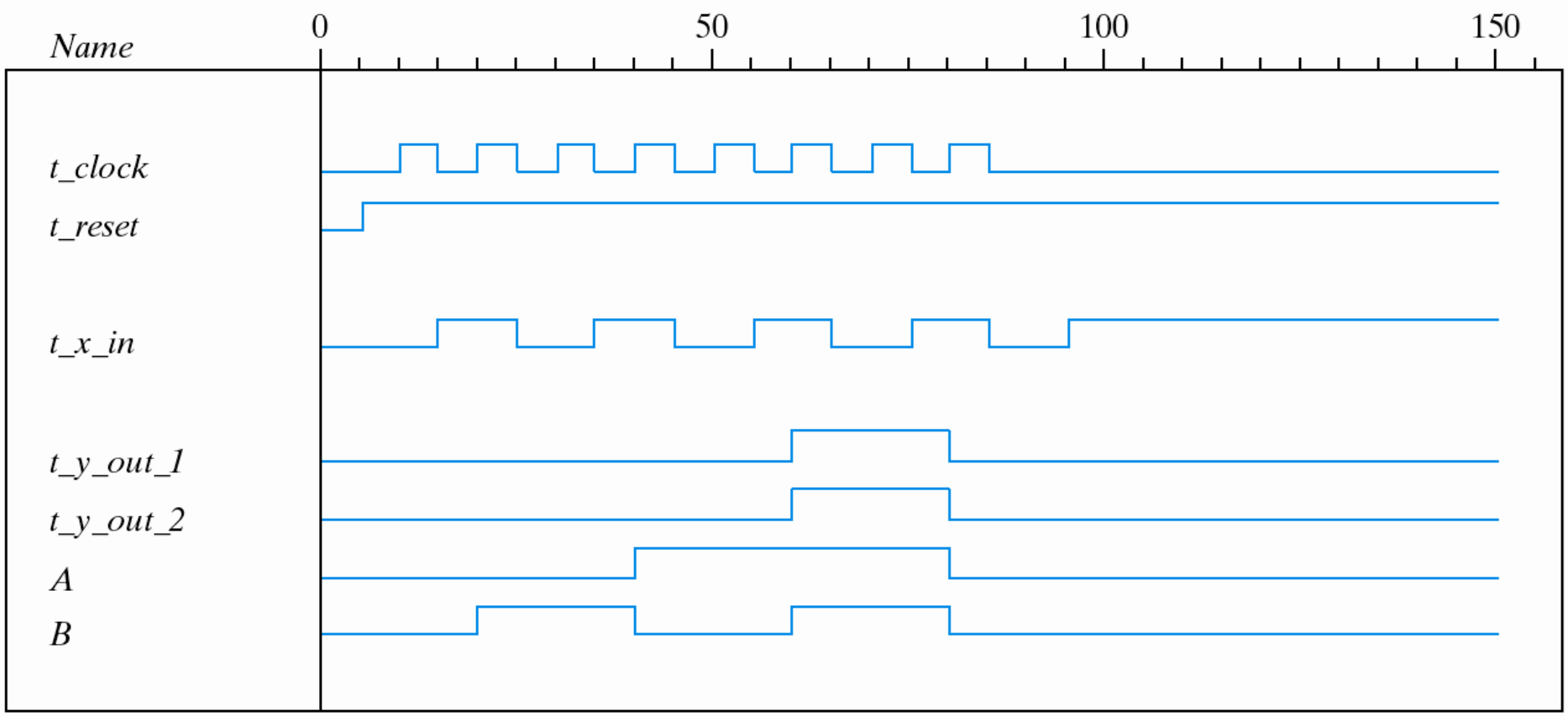


Fig. 5.24 Simulation output of HDL Example 5.7

ModelSim result