Design of Synchronous Sequential Logic

2016/12/12

范真瑋

Ex1

實驗內容：

Exercise 1: Design of Sequence Detector (D-FF) (1/2)

- Design a sequence detector to detect a sequence of three or more consecutive 1's in a string of bits coming through an input line
  - The state diagram and state table of the sequence detector are shown in Fig. 5.27 and Table 5.11
  - Design the sequence detector with D flip-flops
  - 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., a Structural model)
  - Write an Verilog HDL stimulus with a sequence of inputs: 0111110110. Verify that the responses are the same for both descriptions (first reset all flip-flops to 0).

Exercise 1: Design of Sequence Detector (D-FF) (2/2)

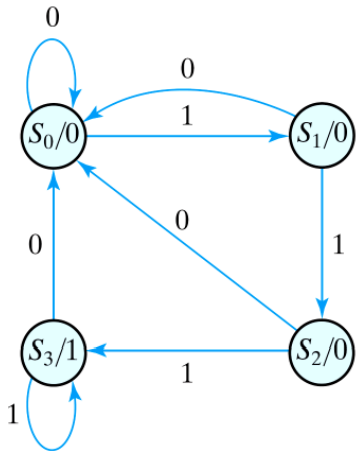


Fig. 5.27 State diagram for sequence detector

Table 5.11
State Table for Sequence Detector

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	1
1	1	1	1	1	1

程式碼：

(Behavioral model)

```
21  module ex1_state(
22   output y_out,
23   input x_in, clock, reset
24  );
25   reg [1: 0] state;
26   parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;
27   always @ (posedge clock, negedge reset)
28   if (reset == 0) state <= S0; // Initialize to state S0
29   else case (state)
30   S0: if (x_in) state <= S1; else state <= S0;
31   S1: if (x_in) state <= S2; else state <= S0;
32   S2: if (x_in) state <= S3; else state <= S0;
33   S3: if (x_in) state <= S3; else state <= S0;
34   endcase
35   assign y_out = (state == S3); // Output of flip-flops
36
37  endmodule
```

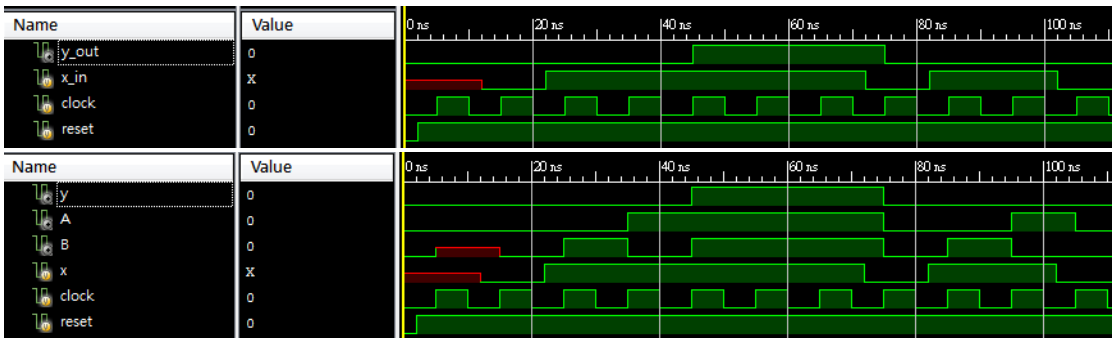
```
25  module ex1_state_tb;
26
27     // Inputs
28     reg x_in;
29     reg clock;
30     reg reset;
31
32     // Outputs
33     wire y_out;
34
35     // Instantiate the Unit Under Test (UUT)
36     ex1_state uut (
37        .y_out(y_out),
38        .x_in(x_in),
39        .clock(clock),
40        .reset(reset)
41     );
42
43   initial #200 $finish;
44  initial begin clock = 0; forever #5 clock = ~clock; end
45
46  initial begin
47  reset = 0;
48  #2 reset = 1;
49  #10 x_in = 0;
50  #10 x_in = 1;
51  #10 x_in = 1;
52  #10 x_in = 1;
53  #10 x_in = 1;
54  #10 x_in = 1;
55  #10 x_in = 0;
56  #10 x_in = 1;
57  #10 x_in = 1;
58  #10 x_in = 0;
59  end
60
61  endmodule
```

(Structural model)

```
21  module ex1_circuit(output y,A,B, input x,clock,reset);
22  wire t1,t2,t3,t4,t5;
23
24  and(t1,A,x);
25  and(t2,B,x);
26  or(t3,t1,t2);
27  d_flip_flop da(A,t3,clock,reset);
28  and(t4,~B,x);
29  and(y,B,A);
30  or(t5,t1,t4);
31  d_flip_flop db(B,t5,clock,reset);
32
33  endmodule
34
35  module d_flip_flop(output Q, input d,clk,reset);
36  wire t1,s,r,t2,q;
37
38  nand(t1,t2,s);
39  nand(s,t1,clk,reset);
40  nand(r,s,clk,t2);
41  nand(t2,r,d,reset);
42  nand(Q,s,q);
43  nand(q,Q,r,reset);
44
45  endmodule
```

```
25  module ex1_circuit_tb;
26
27     // Inputs
28     reg x;
29     reg clock;
30     reg reset;
31
32     // Outputs
33     wire y;
34     wire A;
35     wire B;
36
37     // Instantiate the Unit Under Test (UUT)
38     ex1_circuit uut (
39        .y(y),
40        .A(A),
41        .B(B),
42        .x(x),
43        .clock(clock),
44        .reset(reset)
45     );
46
47  initial #200 $finish;
48  initial begin clock = 0; forever #5 clock = ~clock; end
49
50  initial begin
51  reset = 0;
52  #2 reset = 1;
53  #10 x = 0;
54  #10 x = 1;
55  #10 x = 1;
56  #10 x = 1;
57  #10 x = 1;
58  #10 x = 1;
59  #10 x = 0;
60  #10 x = 1;
61  #10 x = 1;
62  #10 x = 0;
63  end
64
65  endmodule
```

實驗結果及分析：



可以發現 output 只和 state 有關，為 Moore machine，且當有連續 3 個或以上的 1 時，output 為 1

Synthesis using D flip-flops

- The flip-flop input (excitation) equations
  - $A(t+1) = D_A(A, B, x) = \Sigma(3,5,7)$
  - $B(t+1) = D_B(A, B, x) = \Sigma(1,5,7)$
- The output equation
  - $y(A, B, x) = \Sigma(6,7)$
- Logic minimization using the *K* map
  - $D_A = Ax + Bx$
  - $D_B = Ax + B'x$
  - $y = AB$

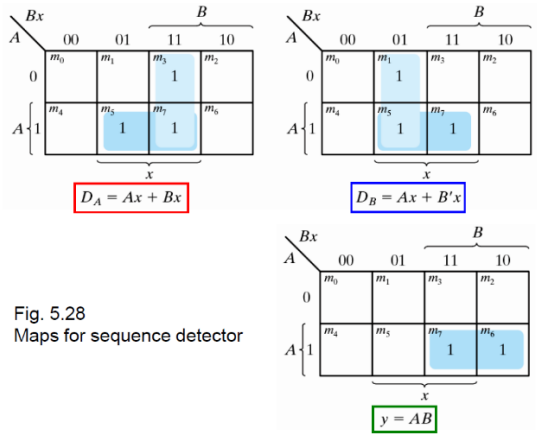


Fig. 5.28
Maps for sequence detector

Sequence detector

- The logic diagram

$D_A = Ax + Bx$
$D_B = Ax + B'x$
$y = AB$

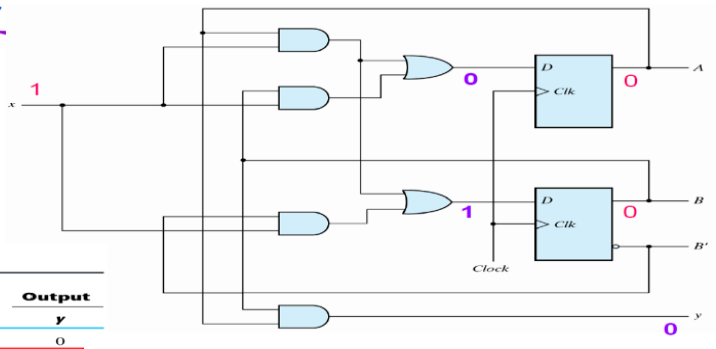


Fig. 5.29
Logic diagram of a Moore-type sequence detector

Table 5.11
State Table for Sequence Detector

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

Ex2

實驗內容：

Exercise 2: Design of Sequence Detector (JK-FF)

- Design a sequence detector to detect a sequence of three or more consecutive 1's in a string of bits coming through an input line
  - The state diagram and state table of the sequence detector are shown in Fig. 5.27 and Table 5.11
  - Design the sequence detector with JK flip-flops
  - 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., a Structural model)
  - Write an Verilog HDL stimulus with a sequence of inputs: 0111110110. Verify that the responses are the same for both descriptions (first reset all flip-flops to 0).

程式碼：

(Behavioral model)

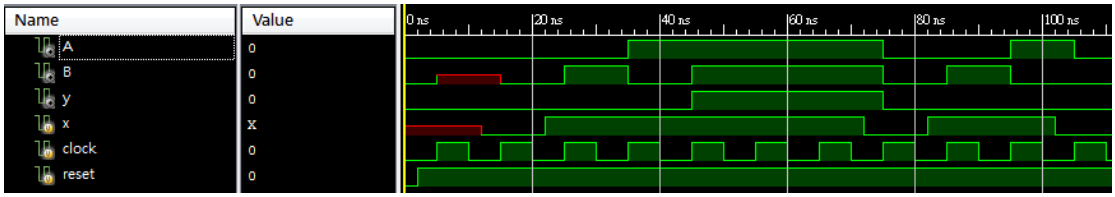
同 Ex1

(Structural model)

```
21  module ex2_circuit(output y,A,B, input x,clock,reset);
22  wire t1,t2;
23
24  and(t1,x,B);
25  jk jka(A,t1,~x,clock,reset);
26  or(t2,~A,~x);
27  jk jkb(B,x,t2,clock,reset);
28  and(y,A,B);
29
30  endmodule
31
32  module jk(output Q, input j,k,clk,reset);
33  wire t1,s,r,t2,q,d,t3,t4;
34
35  and(t3,j,~Q);
36  and(t4,~k,Q);
37  or(d,t3,t4);
38  nand(t1,t2,s);
39  nand(s,t1,clk,reset);
40  nand(r,s,clk,t2);
41  nand(t2,r,d,reset);
42  nand(Q,s,q);
43  nand(q,Q,r,reset);
44
45  endmodule
```

```
25  module ex2_circuit_tb;
26
27      // Inputs
28      reg x;
29      reg clock;
30      reg reset;
31
32      // Outputs
33      wire A;
34      wire B;
35
36      // Instantiate the Unit Under Test (UUT)
37      ex2_circuit uut (
38          .y(y),
39          .A(A),
40          .B(B),
41          .x(x),
42          .clock(clock),
43          .reset(reset)
44      );
45
46  initial #200 $finish;
47  initial begin clock = 0; forever #5 clock = ~clock; end
48
49  initial begin
50  reset = 0;
51  #2 reset = 1;
52  #10 x = 0;
53  #10 x = 1;
54  #10 x = 1;
55  #10 x = 1;
56  #10 x = 1;
57  #10 x = 1;
58  #10 x = 0;
59  #10 x = 1;
60  #10 x = 1;
61  #10 x = 0;
62  end
63
64  endmodule
```

實驗結果及分析：



Synthesis using JK flip-flops

$Q(t)$	$Q(t+1)$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

State Table and JK Flip-Flop Inputs

Present State		Input	Next State		Flip-Flop Inputs			
A	*B*	*x*	*A*	*B*	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	0	0	0	X	X	1
0	1	1	1	0	1	X	X	1
1	0	0	0	0	X	1	0	X
1	0	1	1	1	X	0	1	X
1	1	0	0	0	X	1	X	1
1	1	1	1	1	X	0	X	0

JA = Bx

KA = x'

JB = x

KB = A' +x'

y = AB