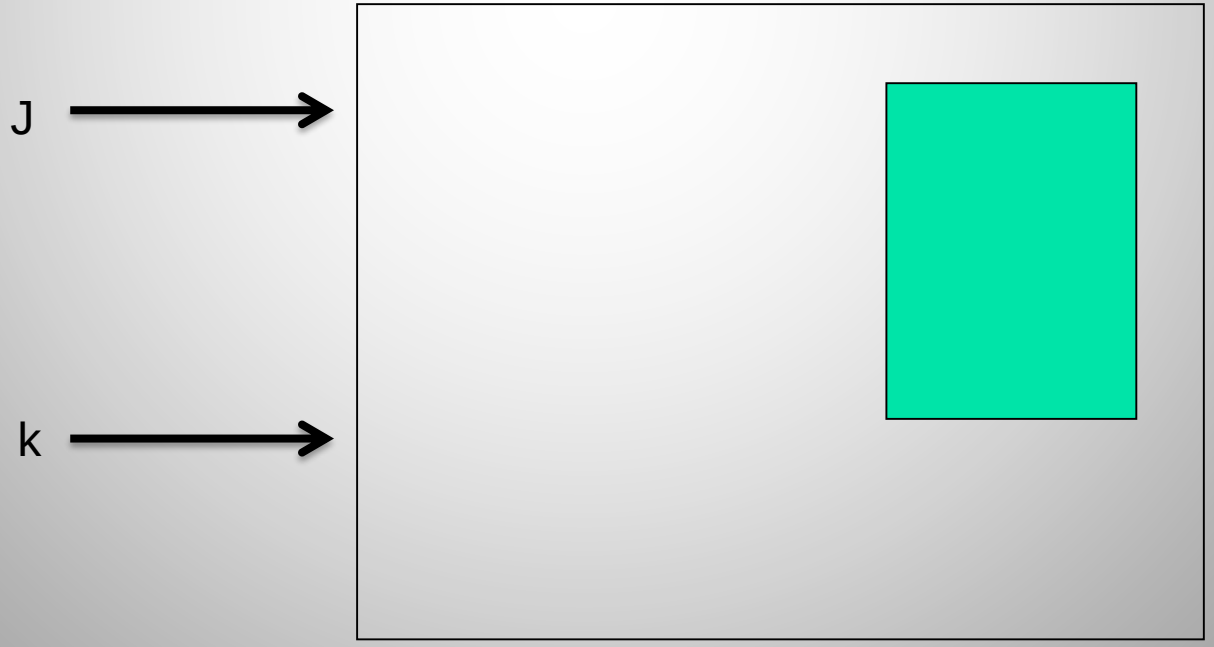
Problems and Solutions

Ch. 5 6

Problem

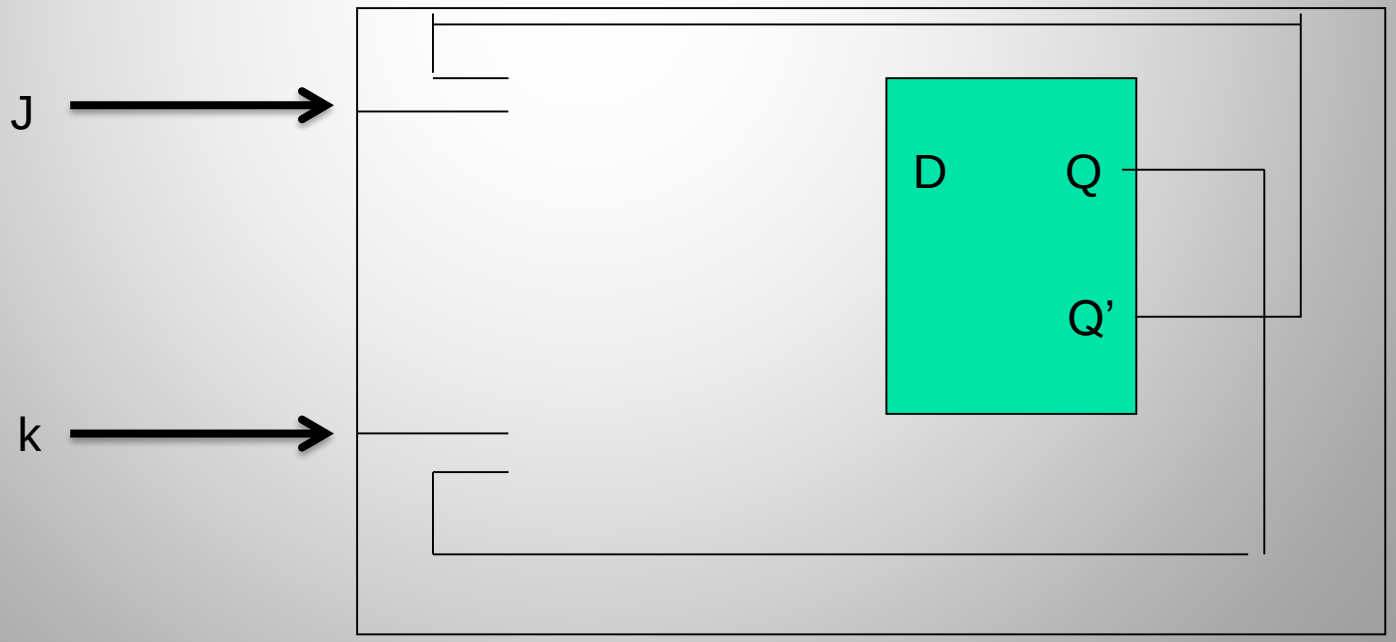
- Construct a *JK* flip-flop using a *D* flip-flop, a two-to-one-line multiplexer, and an inverter.

  - Hint *JK* flip-flop
    - $Q(t+1) = JQ' + K'Q$



Problem

- Construct a *JK* flip-flop using a *D* flip-flop, a two-to-one-line multiplexer, and an inverter.
  - Hint *JK* flip-flop
    - $Q(t+1) = JQ'+K'Q$



Problem

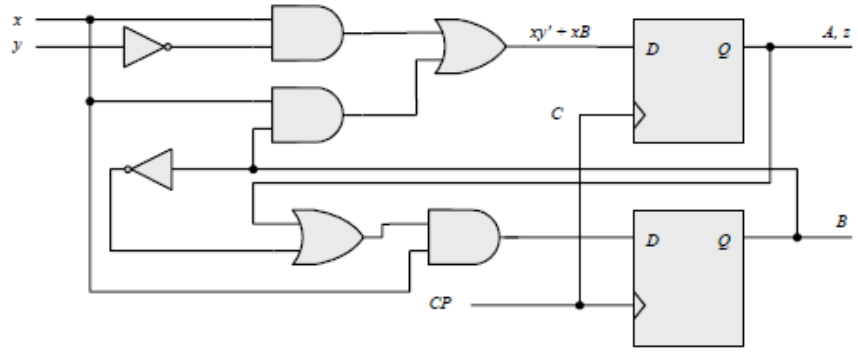
- A sequential circuit with two *D* flip-flops *A* and *B*, two inputs, *x* and *y*; and one output *z* is specified by the following next-state and output equations (HDL—see Problem 5.35):
  - $A(t + 1) = xy' + xB$
  - $B(t + 1) = xA + xB'$
  - $z = A$

(a) Draw the logic diagram of the circuit.

(b) List the state table for the sequential circuit.

(c) Draw the corresponding state diagram.

Solution

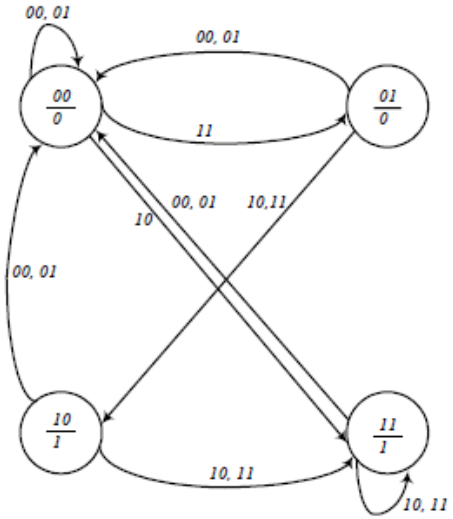


(b)

$A(t+1) = xy' + xB$
$B(t+1) = xA + xB'$
$z = A$

Present state		Inputs		Next state		Output
A	B	x	y	A	B	z
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	1	0
0	0	1	1	0	1	0
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1

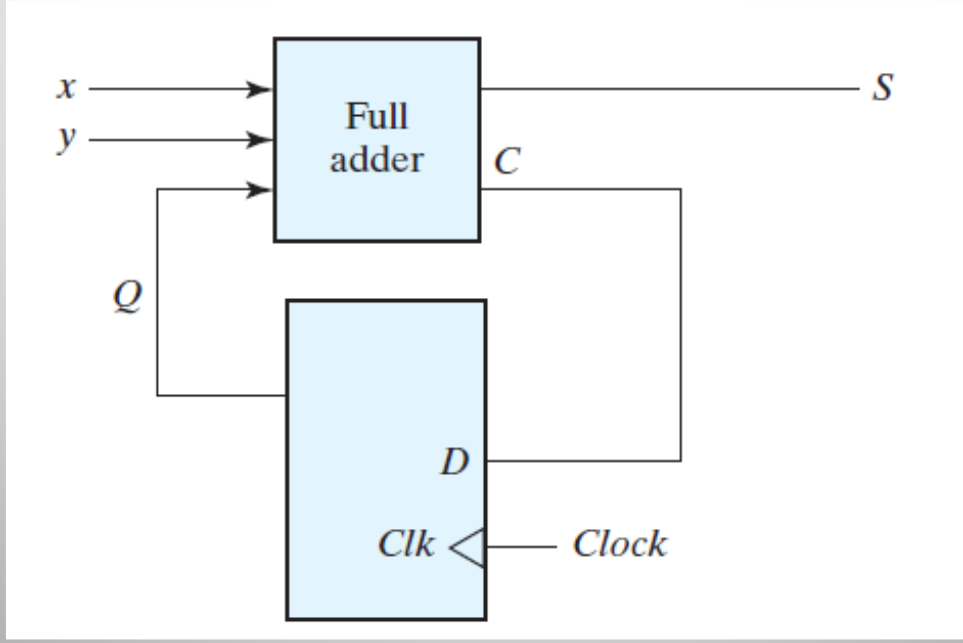
(c)



$A(t + 1) = xy' + xB$
$B(t + 1) = xA + xB'$
$z = A$

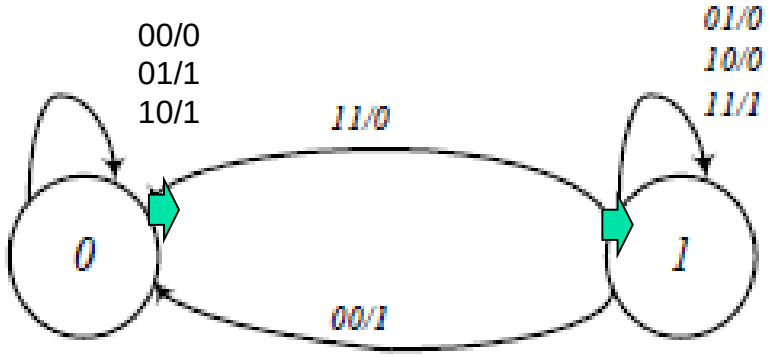
Problem

- A sequential circuit has one flip-flop *Q*, two inputs *x* and *y*, and one output *S*. It consists of a full-adder circuit connected to a *D* flip-flop, as shown in Fig. P5.7. Derive the state table and state diagram of the sequential circuit.



Solution

Present state	Inputs		Next state	Output
Q	x	y	Q	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

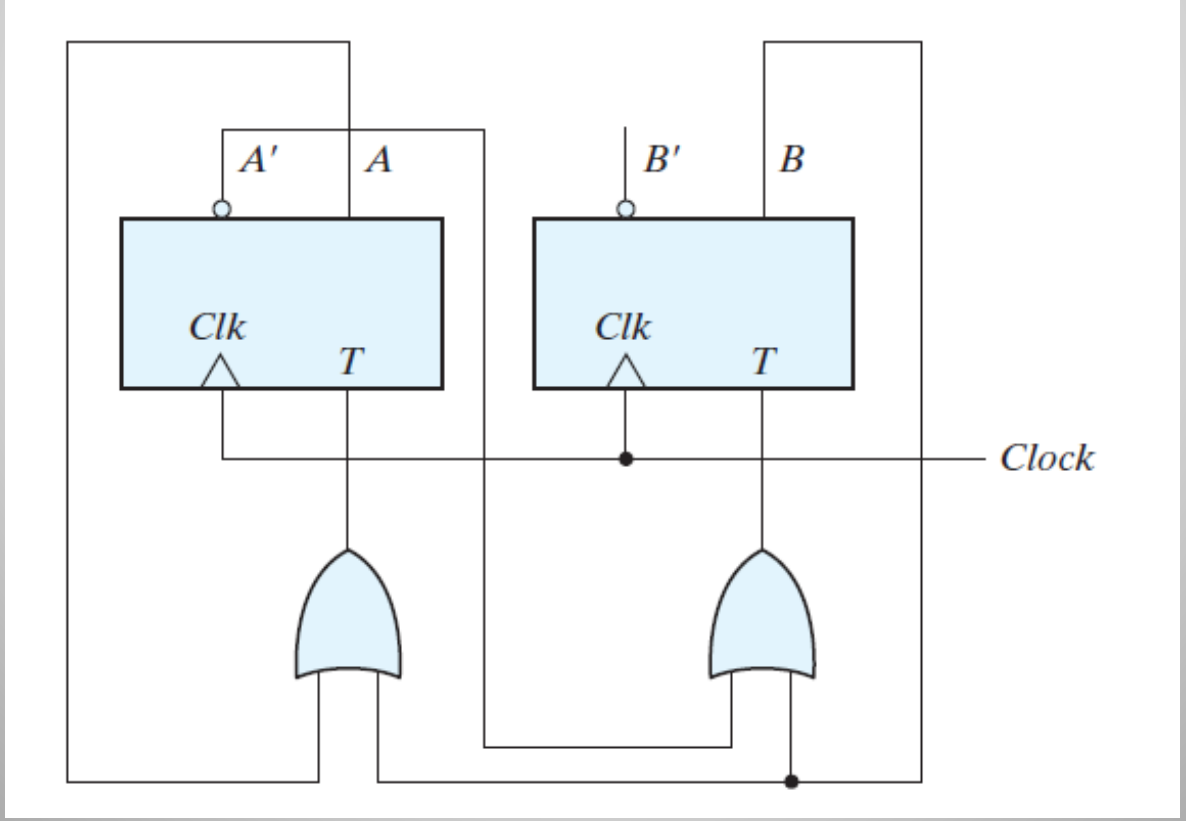


$$S = x \oplus y \oplus Q$$

$$Q(t+1) = xy + xQ + yQ$$

Problem

- Derive the state table and the state diagram of the sequential circuit shown in Fig. P5.8. Explain the function that the circuit performs



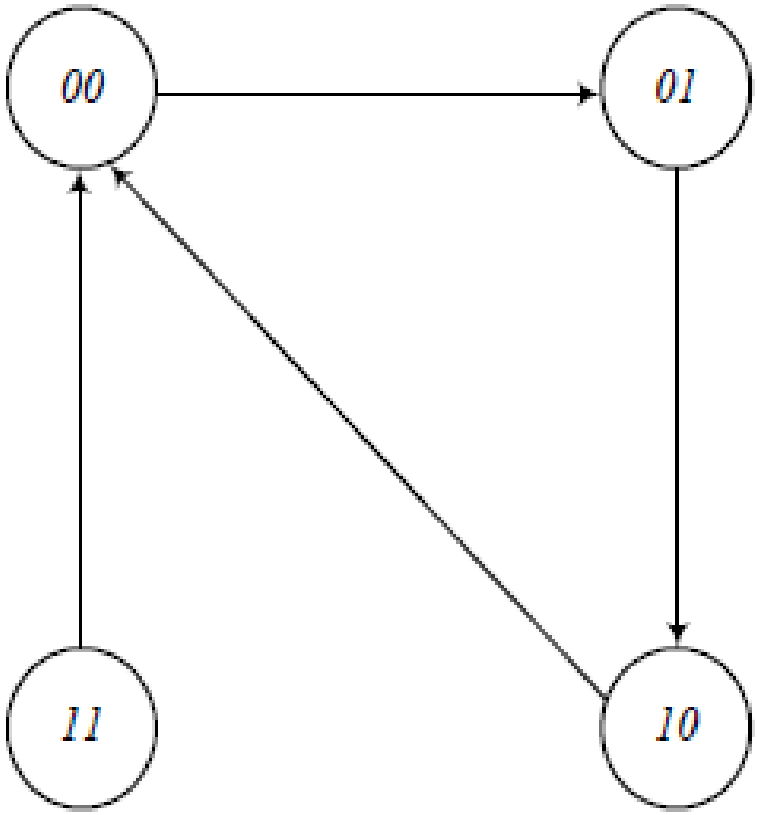
Solution

Present State		Next State		FF Inputs	
A	B	A	B	T_A	T_B
0	0	0	1	0	1
0	1	1	0	1	1
1	0	0	0	1	0
1	1	0	0	1	1

$T_A = A + B$

$T_B = A' + B$

Repeated sequence: 00 → 01 → 10



Problem

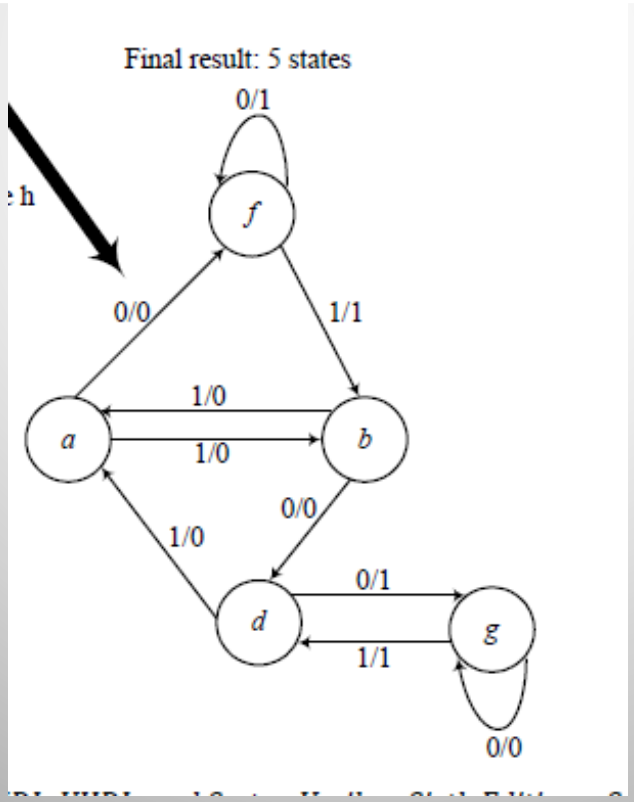
Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	f	b	0	0
b	d	c	0	0
c	f	e	0	0
d	g	a	1	0
e	d	c	0	0
f	f	b	1	1
g	g	h	0	1
h	g	a	1	0

(a)* Tabulate the reduced state table.
(c) Draw the state diagram corresponding to the reduced state table.

Solution

Present state	Next state 0	Next state 1	Output 0	Output 1
a	*f*	*b*	0	0
b	*d*	*a*	0	0
d	*g*	*a*	1	0
f	*f*	*b*	1	1
g	*g*	*d*	0	1

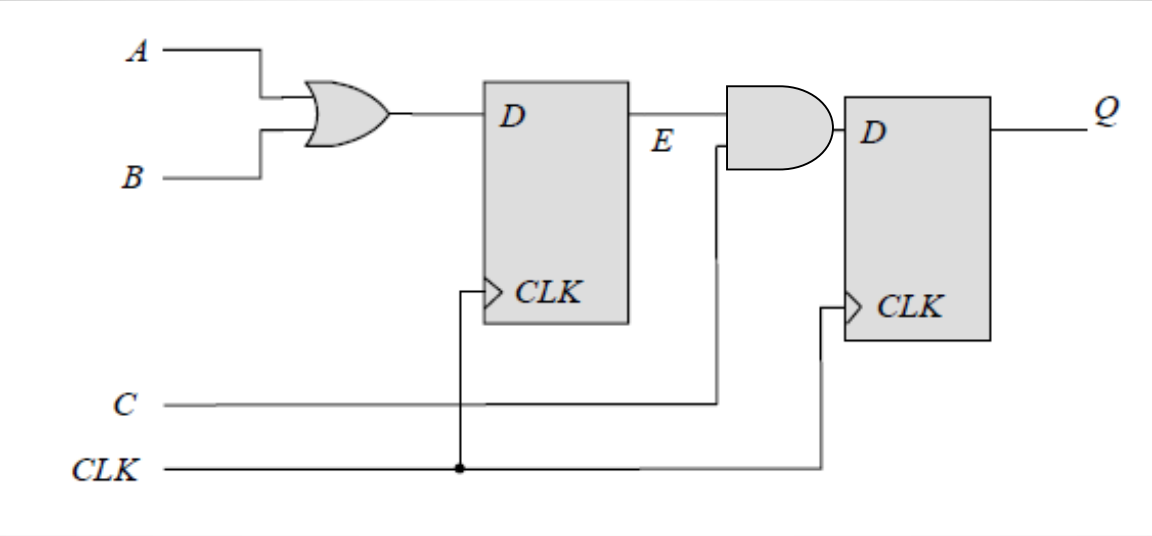
Note: Equivalent states: b = e, a = c, h = d



- Draw the logic diagram for the sequential circuit described by the following HDL code:

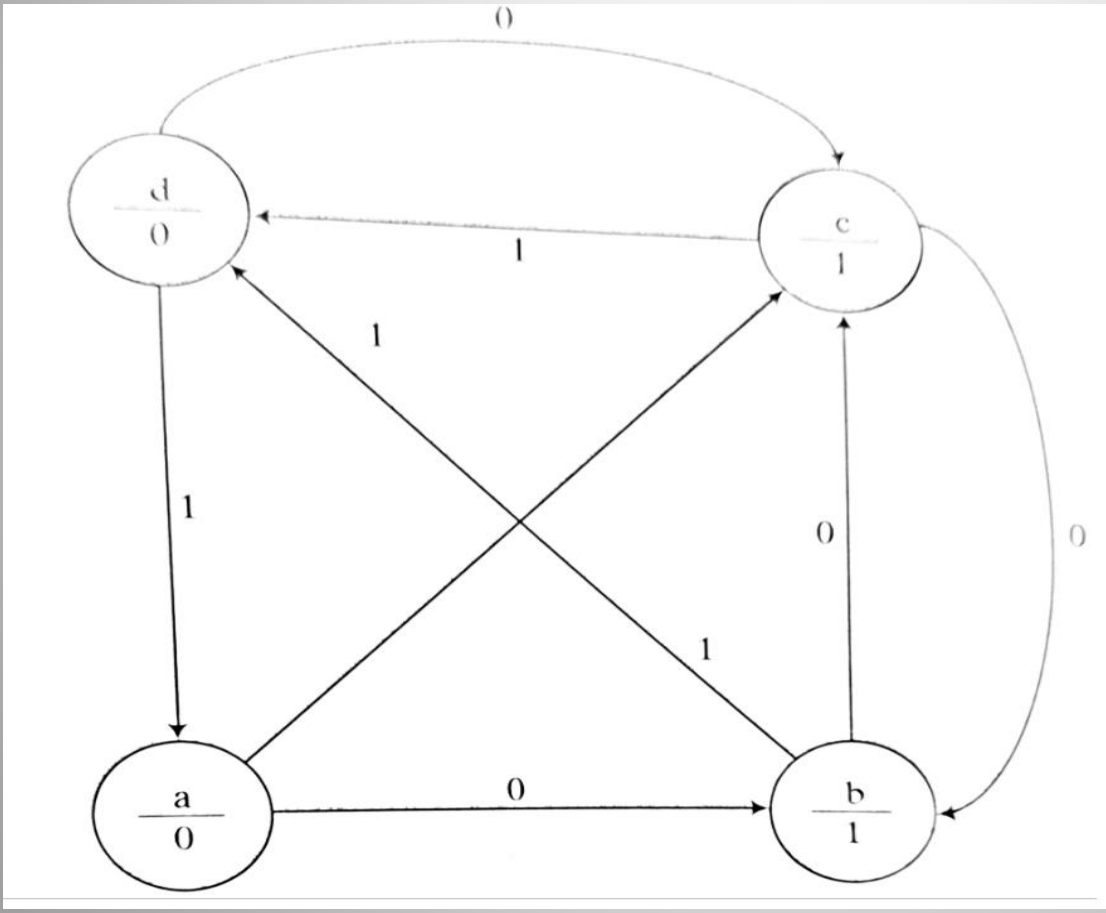
```
always @ (posedge CLK)
begin
        E <= A | B;
        Q <= E & C;
end
```

Solution



Problem

- Draw the state diagram of the machine described by the HDL model given below.



Problems (needs update)

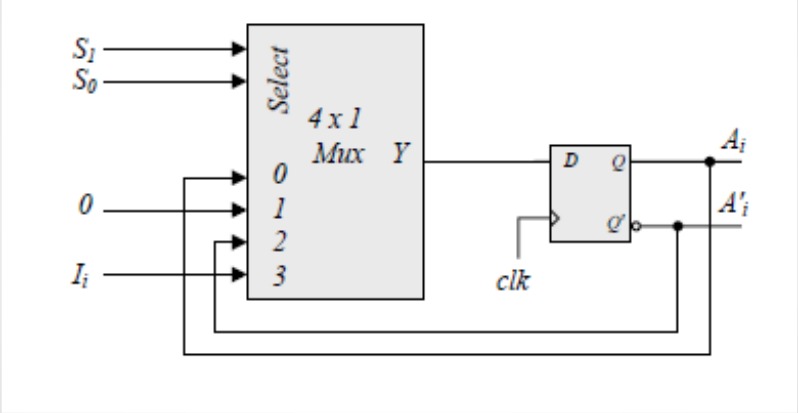
```
module Prob_5_51 (output reg y_out, input x_in, clk, reset_b);
 parameter s0 = 2'b00, s1 = 2'b01, s2 = 2'b10, s3 = 2'b11;
 reg [1:0] state, next_state;
 always @ (posedge clk, negedge reset_b) begin
  if (reset_b == 1'b0) state <= s0;
  else state <= next_state;
 always @(state, x_in) begin
  y_out = 0;
  next_state = s0;
  case (state)
   s0: begin y_out = 0; if (x_in) next_state = s1; else next_state = s0; end;
   s1: begin y_out = 0; if (x_in) next_state = s2; else next_state = s1; end;
   s2: begin y_out = 1; if (x_in) next_state = s3; else next_state = s2; end;
   s3: begin y_out = 1; if (x_in) next_state = s0; else next_state = s3; end;
   default: next_state = s0;
  endcase
 end
endmodule
```

Problem

- Draw the logic diagram of a four-bit register with four *D* flip-flops and four 4 * 1 multiplexers with mode selection inputs *s*1 and *s*0. The register operates according to the following function table.

s_1	s_0	Register Operation
0	0	No change
1	0	Complement the four outputs
0	1	Clear register to 0 (synchro-nous with the clock)
1	1	Load parallel data

Solution



Problem

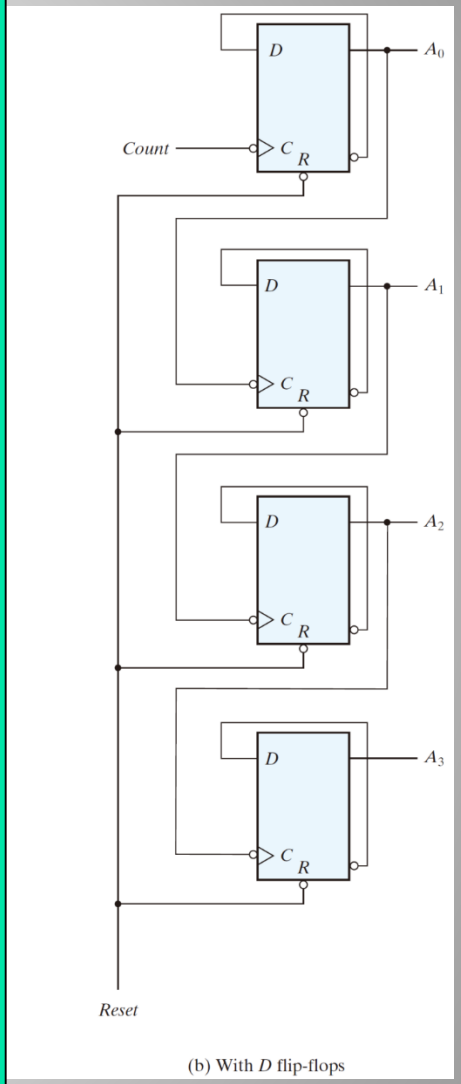
- Draw the logic diagram of a four-bit binary ripple countdown counter using:

(a) flip-flops that trigger on the positive-edge of the clock; and

(b) flip-flops that trigger on the negative-edge of the clock.

Solution

With the bubbles in C removed (positive-edge).
With complemented flip-flops connected to C.



(b) With *D* flip-flops

Problem

- Design a four-bit binary synchronous counter with *D* flip-flops.

Solution

With *E* denoting *Count_enable* in Fig. 6.12 and D-flip-flops replacing the J-K flip-flops, the toggling action of the bits of the counter is determined by: $T0 = E$, $T1 = A0\,E$, $T2 = A0\,A1\,E$, $T3 = A0\,A1\,A2\,E$. Since $DA = A \oplus TA$ the inputs of the flip-flops of the counter are determined by:

$DA0 = A0 \oplus E$;

$DA1 = A1 \oplus (A0\,E)$

$DA2 = A2 \oplus (A0\,A1\,E)$

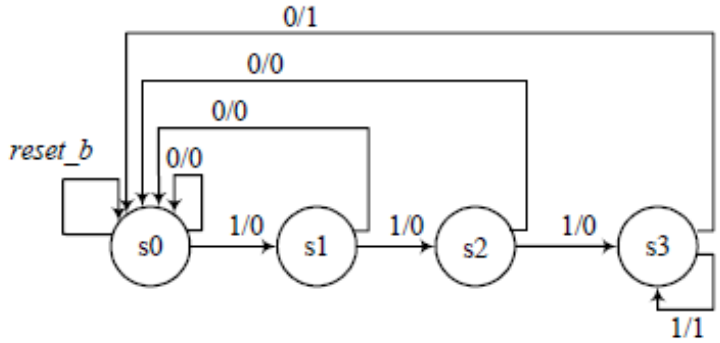
$DA3 = A3 \oplus (A0\,A1\,A2\,E)$

Problems

- Develop the state diagram for a Mealy state machine that detects a sequence of three or more consecutive 1’s in a string of bits coming through an input line.

Solution

Assumption: Synchronous active-low reset
Mealy machine, links for reset on-the-fly are implicit and not shown

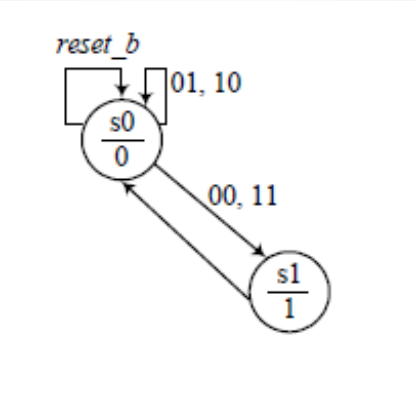


Solutions

```
module Prob_5_55 (input x_in, clk, reset_b, output reg y);
    parameter s0 = 1'd0;
    parameter s1 = 1'd1;
    parameter s2 = 1'd2;
    parameter s3 = 1'd3;
    reg [1: 0] state, next-state;

    always @ (posedge clk, negedge reset_b)
        if (reset_b == 1'b0) state <= s0;
        else state <= next_state;

    always @ (state, x_in) begin
        y = 1'b0;
        next_state = s0;

        case (state)        // Mealy machine
        s0:     if (x_in) begin y = 1'b0; next_state = s1; end
                else begin y = 1'b0; next_state = s0; end
        s1:     if (x_in) begin y = 1'b0; next_state = s2; end
                else y = 1'b0; next_state = s0; end
        s2:     if (x_in) begin y = 1'b0; next_state = s3; end
                else begin y = 1'b0; next_state = s0; end
        s3:     if (x_in) begin y = 1'b1; next_state = s3; end
                else begin y = 1'b0; next_state = s0; end
        default:    begin y = 1'b0; next_state = s0; end
    endcase
```

Problems

- A synchronous Moore machine has two inputs *x*1 and *x*2, and an output *y_out*. If both inputs have the same value, the output is asserted for one cycle; otherwise, the output is 0. Develop a state diagram


reset_b
01, 10
s0
0
00, 11
s1
1