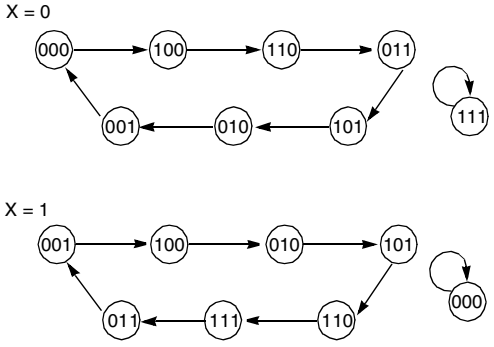
Solutions to Problems Marked with a * in
Logic and Computer Design Fundamentals, 4th Edition

Chapter 5

5-7.*

Present state			Input	Next state		
A	*B*	*C*	*X*	*A*	*B*	*C*
0	0	0	0	1	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	1	0
1	0	1	0	0	1	0
1	0	1	1	1	1	0
1	1	0	0	0	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	0	1	1



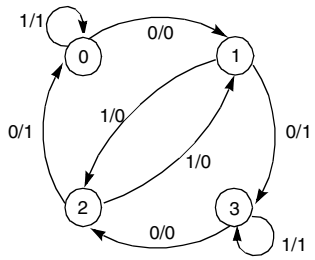
State diagram is the combination of the above two diagrams.

5-11.*

$S_A = B \qquad S_B = \overline{X \oplus A}$

$R_A = \overline{B} \qquad R_B = X \oplus A$

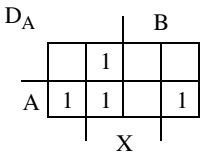
Present state		Input	Next state		Output
A	*B*	*X*	*A*	*B*	*Y*
0	0	0	0	1	0
0	0	1	0	0	1
0	1	0	1	1	1
0	1	1	1	0	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	1



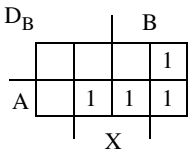
Format: X/Y

5-13.*

Present state		Input	Next state	
A	*B*	*X*	*A*	*B*
0	0	0	0	0
0	0	1	1	0
0	1	0	0	1
0	1	1	0	0
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	0	1



$D_A = A\overline{X} + \overline{B}X$

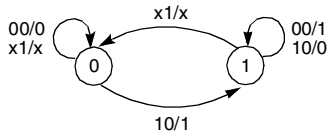


$D_B = AX + B\overline{X}$

Logic diagram not given.

5-18.*

Format: XY/Z (x = unspecified)



Present state	Inputs		Next state	Output
Q(t)	*X*	*Y*	*Q(t+1)*	*Z*
0	0	0	0	0
0	0	1	0	X
0	1	0	1	1
0	1	1	0	X
1	0	0	1	1
1	0	1	0	X
1	1	0	1	0
1	1	1	0	X

5-26.*

To use a one-hot assignment, the two flip-flops A and B need to be replaced with four flip-flops Y4, Y3, Y2. Y1.

No Reset State Specified.

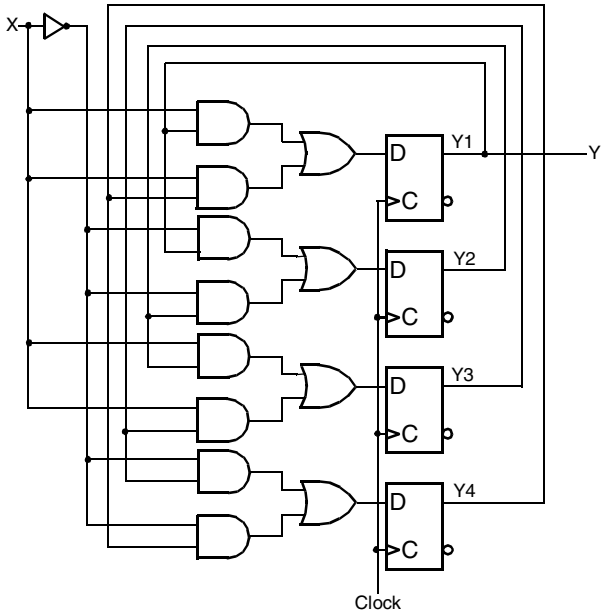
Present State		Input	Next State		Output
A B	*Y4 Y3 Y2 Y1*	*X*	*A' B"*	*Y4'Y3'Y2'Y1*	*Z*
0 0	0 0 0 1	0	0 1	0 0 1 0	1
0 0	0 0 0 1	1	0 0	0 0 0 1	1
0 1	0 0 1 0	0	0 1	0 0 1 0	0
0 1	0 0 1 0	1	1 0	0 1 0 0	0
1 0	0 1 0 0	0	1 1	1 0 0 0	0
1 0	0 1 0 0	1	1 0	0 1 0 0	0
1 1	1 0 0 0	0	1 1	1 0 0 0	0
1 1	1 0 0 0	1	0 0	0 0 0 1	0

$D1 = Y1' = X \cdot Y1 + X \cdot Y4$

$D2 = Y2' = \overline{X} \cdot Y1 + \overline{X} \cdot Y2$

$D3 = Y3' = X \cdot Y2 + X \cdot Y3$

$D4 = Y4' = \overline{X} \cdot Y3 + \overline{X} \cdot Y4$

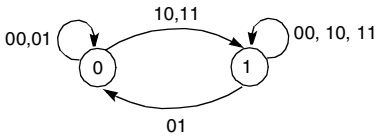


5-27.*

a)

S	R	Q	
0	0	Q	No Change
0	1	0	Reset
1	0	1	Set
1	1	1	Set

b)

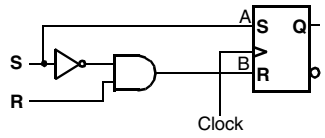


c)

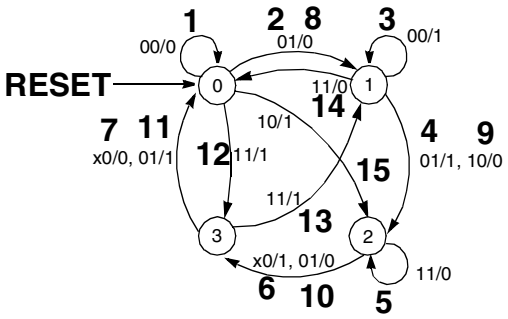
Present state	Input		Next state		
Q	*S*	*R*	*Q(t+1)*	*A*	*B*
0	0	0	0	0	x
0	0	1	0	0	x
0	1	0	1	1	0
0	1	1	1	1	0
1	0	0	1	x	0
1	0	1	0	0	1
1	1	0	1	x	0
1	1	1	1	x	0

A = S

B = $\overline{S}$R



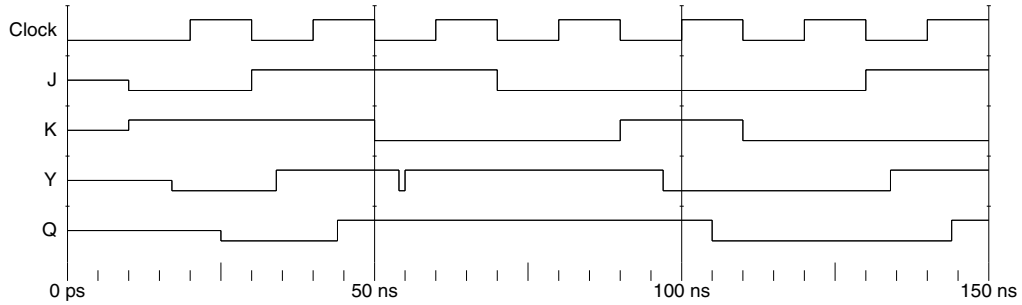
*5-31.



Format: XY/Z

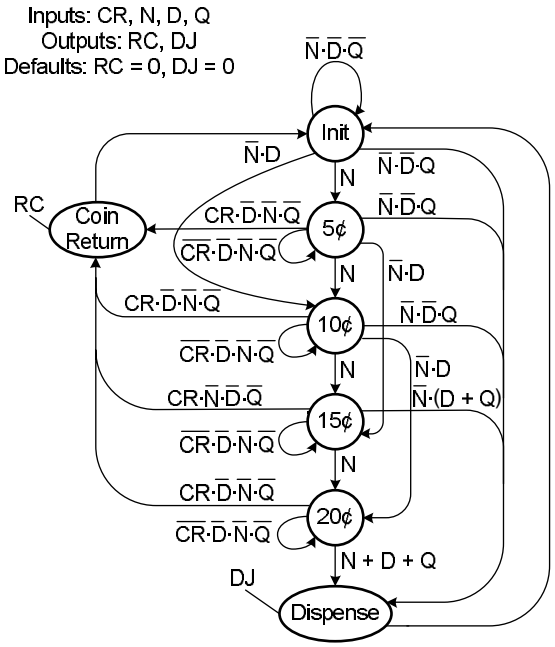
Reset, 00, 01, 00, 01, 11, x0, x0, 01, 10, 01, 01, 11, 11, 11, 10.

5-33.*



This simulation was performed without initializing the state of the latches of the flip-flop beforehand. Each gate in the flip-flop implementation has a delay of 1 ns. The interaction of these delays with the input change times produced a narrow pulse in Y at about 55 ns. In this case, the pulse is not harmful since it dies out well before the positive clock edge occurs. Nevertheless, a thorough examination of such a pulse to be sure that it does not represent a design error or important timing problem is critical.

5-37.*



5-40.*

```
library IEEE;
use IEEE.std_logic_1164.all;

entity mux_4to1 is
     port (
          S: in STD_LOGIC_VECTOR (1 downto 0);
          D: in STD_LOGIC_VECTOR (3 downto 0);
          Y: out STD_LOGIC
     );
end mux_4to1;

-- (continued in the next column)
```

```
architecture mux_4to1_arch of mux_4to1 is
begin

process (S, D)
    begin
    case S is
         when "00" => Y <= D(0);
         when "01" => Y <= D(1);
         when "10" => Y <= D(2);
         when "11" => Y <= D(3);
         when others => null;
    end case;

end process;
end mux_4to1_arch;
```

5-45.*

```
library IEEE;
use IEEE.std_logic_1164.all;
entity jkff is
   port (
      J,K,CLK: in STD_LOGIC;
      Q: out STD_LOGIC
   );
end jkff;

architecture jkff_arch of jkff is
signal q_out: std_logic;
begin

state_register: process (CLK)
begin
  if CLK'event and CLK='0' then  --CLK falling edge

-- (continued in the next column)
```

```
    case J is
        when '0' =>
            if K = '1' then
                q_out <= '0';
            end if;
        when '1' =>
            if K = '0' then
                q_out <= '1';
            else
                q_out <= not q_out;
            end if;
        when others => null;
    end case;
  end if;
end process;

Q <= q_out;
end jkff_arch;
```

5-49.*

```
module problem_6_39 (S, D, Y) ;

input [1:0] S ;
input [3:0] D ;
output Y;
reg Y ;

// (continued in the next column)
```

```
always @(S or D)
     begin
          if (S == 2'b00) Y <= D[0];
          else if (S == 2'b01) Y <= D[1];
          else if (S == 2'b10) Y <= D[2];
          else Y <= D[3];

     end
endmodule
```

5-53.*

```
module JK_FF (J, K, CLK, Q) ;

input J, K, CLK ;
output Q;
reg Q;
```

```
always @(negedge CLK)
    case (J)
        0'b0: Q <= K ? 0: Q;
        1'b1: Q <= K ? ~Q: 1;
    endcase
endmodule
```