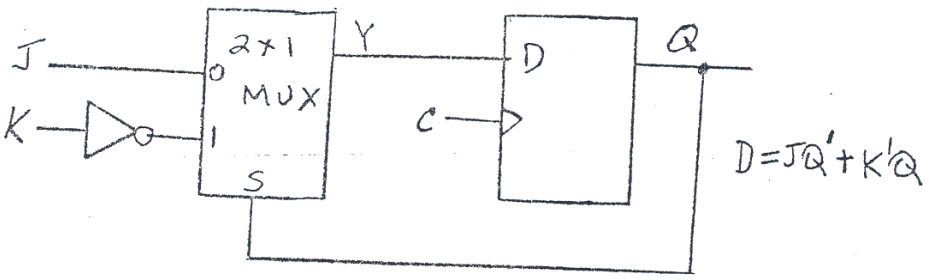
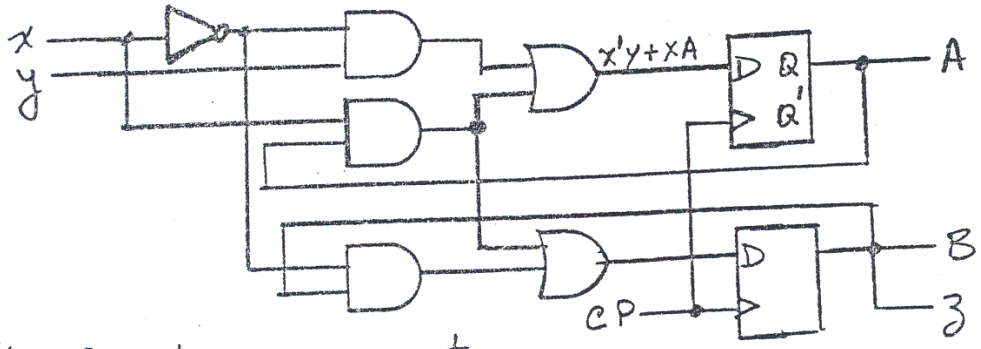
5-2



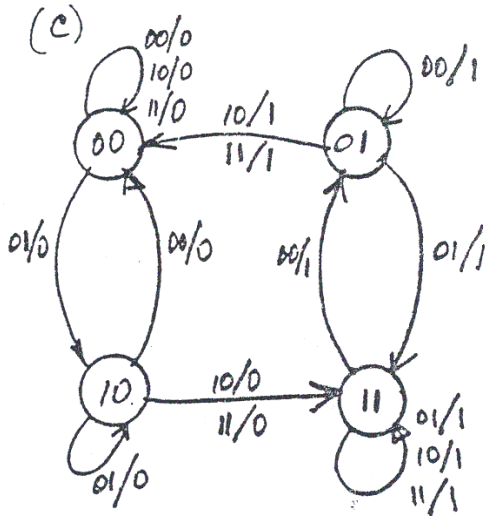
5-6 (a)



(b)

Present state A B	inputs x y	Next state A B	output z
0 0	0 0	0 0	0
0 0	0 1	1 0	0
0 0	1 0	0 0	0
0 0	1 1	0 0	0
0 1	0 0	0 1	1
0 1	0 1	1 1	1
0 1	1 0	0 0	1
0 1	1 1	0 0	1
1 0	0 0	0 0	0
1 0	0 1	1 0	0
1 0	1 0	1 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	1 1	1

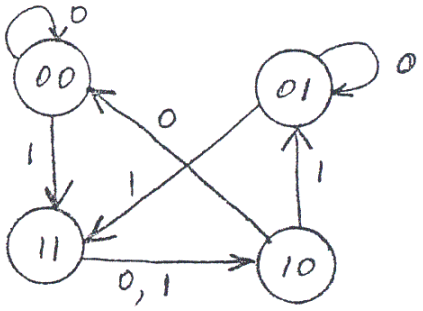
(c)



5-9 ✓

$$A(t+1) = J_A A' + K' A$$
$$= xA' + BA$$

$$B(t+1) = J_B B' + K_B' B$$
$$= xB' + A'B$$



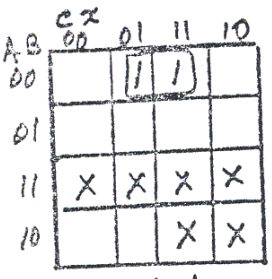
5-12

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

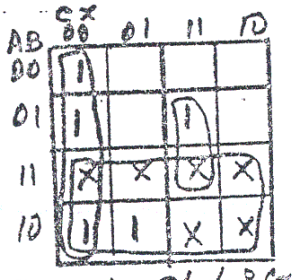
5-19 (a)

Present state ABC	Input x	Next state ABC	output y
000	0	011	0
000	1	100	1
001	0	001	0
001	1	100	1
010	0	010	0
010	1	000	1
011	0	001	0
011	1	010	1
100	0	010	0
100	1	011	0

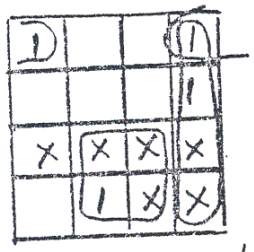
$d(A,B,C,x) = \Sigma(10,11,12,13,14,15)$



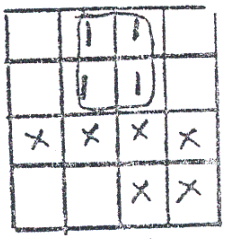
$DA = A'B'x$



$DB = A + C'x' + BCx$

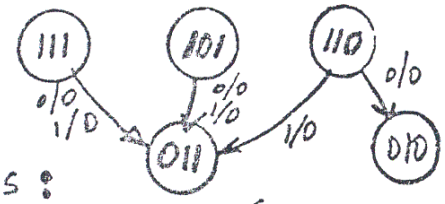


$DC = Cx' + Ax + A'B'x'$



$y = A'x$

self-correcting



(b) Use JK flip flops: same state table as in part (a).

Flip-flop inputs JA	KA	JB	KB	JC	KC
0	X	1	X	1	X
1	X	0	X	0	X
0	X	0	X	X	0
1	X	0	X	X	1
0	X	X	0	0	X
0	X	X	1	0	X
0	X	X	1	X	0
0	X	X	0	X	1
X	1	1	X	0	X
X	1	1	X	1	X

$JA = B'x$ $\quad KA = 1$

$JB = A + C'x'$ $\quad KB = C'x + Cx'$

$JC = Ax + A'B'x'$ $\quad KC = x$

$y = A'x$

Self-correcting because $KA = 1$

5-24

```
module Async_preset_clear_FF (Q,D,CLK,preset,clear);
output Q;
input D,CLK,preset,clear;
reg Q;
always @ (posedge CLK or negedge preset or negedge clear)
  if (~ preset)  Q = 1'b1;
  else if (~ clear)  Q = 1'b0;
  else  Q = D;
endmodule
```

5-26

$Q(t + 1) = JQ' + K'Q.$

When $Q = 0$, $Q(t + 1) = J$;

When $Q = 1$, $Q(t + 1) = K'$

```
always @ posedge (CLK)
  if (Q == 0)  Q = J;
        else  Q = ~K;
```