Design procedure

Sparsh Mittal

Goal: designing hardware that will implement a desired behavior.

In contrast to a combinational circuit, which is fully specified by a truth table, a sequential circuit requires a state table for its specification.

Steps

1. From the word description and specifications of the desired operation, derive a state diagram for the circuit.

2. Reduce the number of states if necessary.

3. Assign binary values to the states.

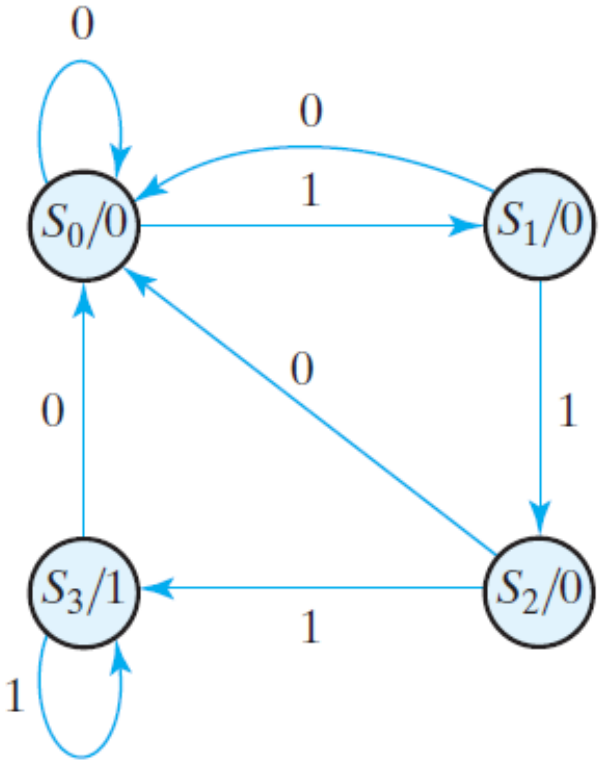
4. Obtain the binary-coded state table.

5. Choose the type of flip-flops to be used.

6. Derive the simplified flip-flop input equations and output equations.

7. Draw the logic diagram.

A circuit for detecting >=3 consecutive 1s in serial bit-stream



Start in state $S0$.

If input is 0, circuit stays in $S0$, but if input is 1, it goes to state $S1$ to indicate that a 1 was detected.

If the next input is 1, the change is to state $S2$ to indicate the arrival of two consecutive 1's, but if the input is 0, the state goes back to $S0$.

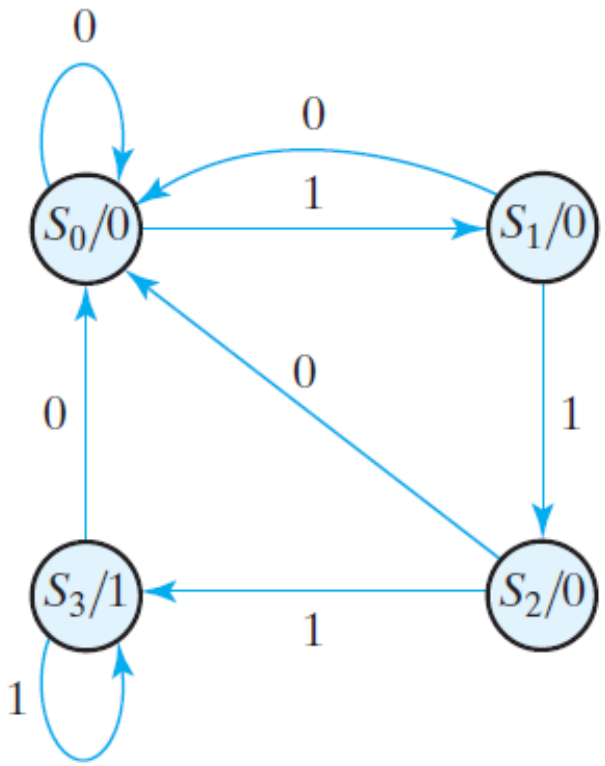
The third consecutive 1 sends the circuit to state $S3$.

If more 1's are detected, the circuit stays in $S3$. Any 0 input sends the circuit back to $S0$.

Summary: circuit stays in $S3$ as long as there are three or more consecutive 1's received.

Is it Moore or Mealy? Moore

Create state table (with D flip-flops)

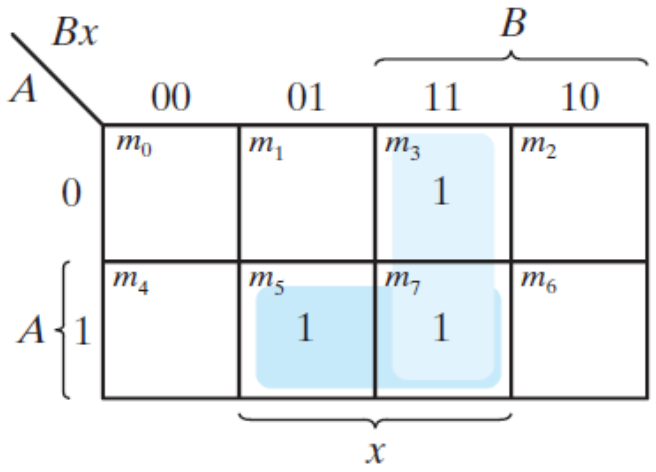


State Table for Sequence Detector

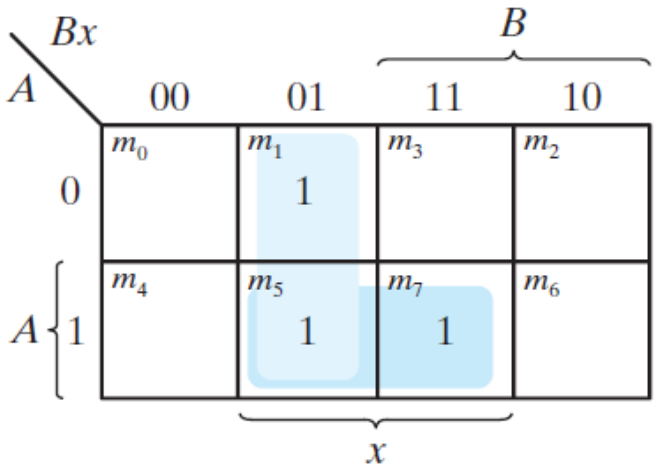
Present State		Input	Next State		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

Finding equations

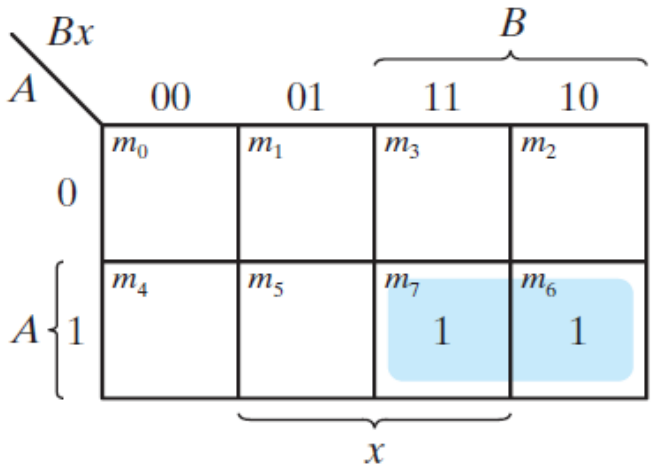
We choose two D flip-flops to represent the four states, and we label their outputs A and B . There is one input x and one output y .



$D_A = Ax + Bx$



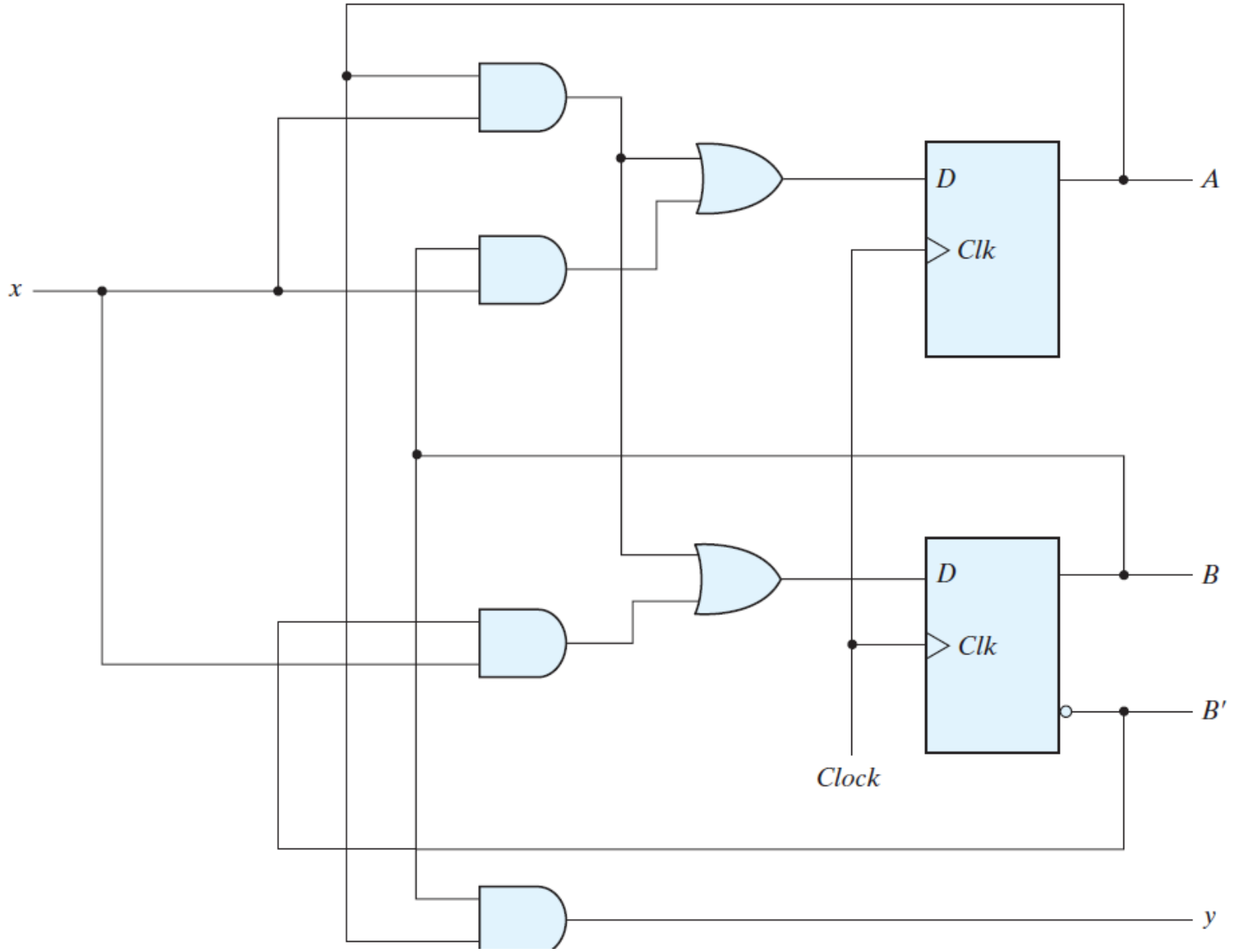
$D_B = Ax + B'x$



$y = AB$

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

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



Excitation table

The design of a sequential circuit with flip-flops other than the *D* type is complicated by the fact that the input equations for the circuit must be derived indirectly from the state table.

When *D* -type flip-flops are employed, the input equations are obtained directly from the next state.

This is not the case for the *JK* and *T* types of flip-flops.

To determine the input equations for these flip-flops, it is necessary to derive a functional relationship between the state table and the input equations.

During the design process, we usually know the transition from the present state to the next state and wish to find the flip-flop input conditions that will cause the required transition.

➔ We need a table that lists the required inputs for a given change of state. Such a table is called an *excitation table*

It has a column for the present state $Q(t)$, a column for the next state $Q(t + 1)$, and a column for each input to show how the required transition is achieved.

Table 5.1
Flip-Flop Characteristic Tables

JK Flip-Flop

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

T Flip-Flop

T	Q(t + 1)	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

There are four possible transitions from the present state to the next state.

Flip-Flop Excitation Tables

Q(t)	Q(t = 1)	J	K
0	0		
0	1		
1	0		
1	1		

(a) *JK* Flip-Flop

Q(t)	Q(t = 1)	T
0	0	
0	1	
1	0	
1	1	

(b) *T* Flip-Flop

Table 5.1
Flip-Flop Characteristic Tables

JK Flip-Flop

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

T Flip-Flop

T	Q(t + 1)	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

There are four possible transitions from the present state to the next state.

Flip-Flop Excitation Tables

Q(t)	Q(t = 1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

(a) *JK* Flip-Flop

Q(t)	Q(t = 1)	T
0	0	0
0	1	1
1	0	1
1	1	0

(b) *T* Flip-Flop

State table

State Table and JK Flip-Flop Inputs

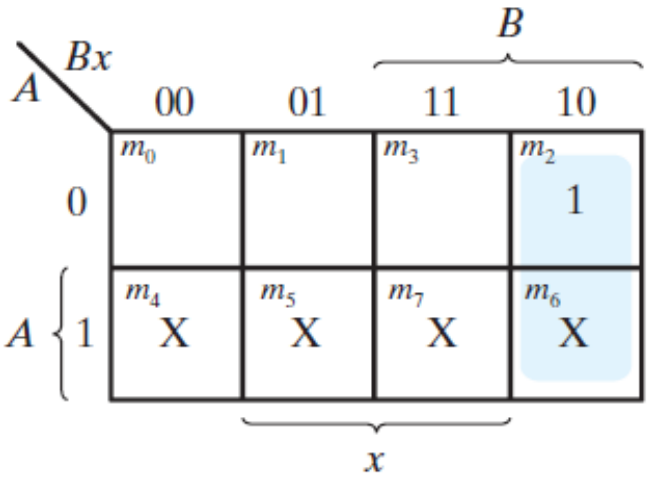
Present State		Input	Next State		Flip-Flop Inputs			
A	*B*	*x*	*A*	*B*	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

Example: We have a transition for flip-flop *A* from 0 in the present state to 0 in the next state.

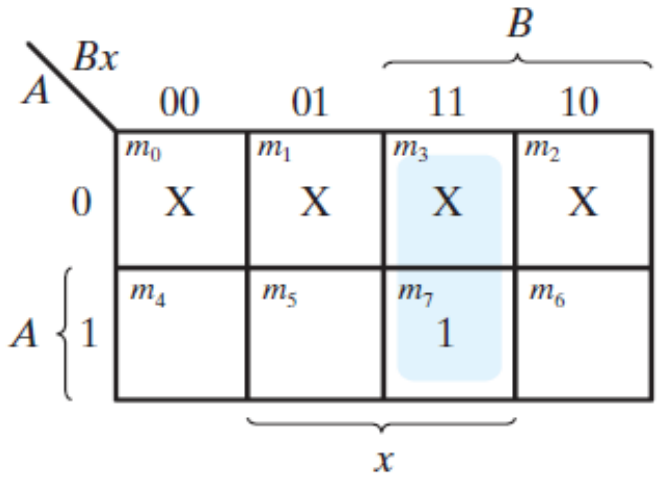
For the *JK* flip-flop, a transition of states from present state 0 to next state 0 requires that input *J* be 0 and input *K* be a don't-care.

So 0 and X are entered in the first row under *JA* and *KA*, respectively.

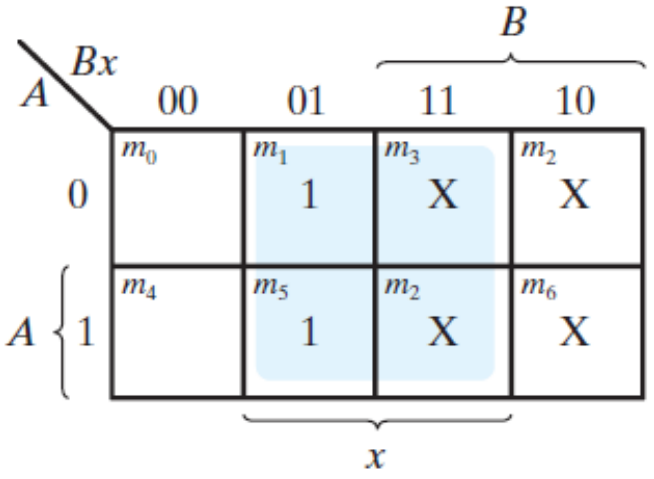
K-maps



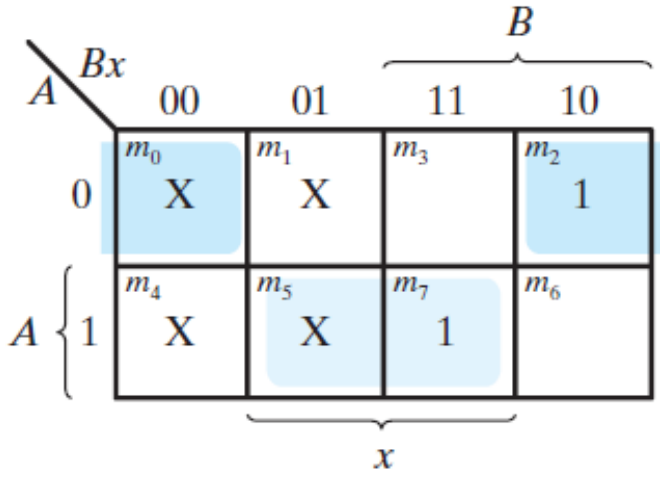
$J_A = Bx'$



$K_A = Bx$

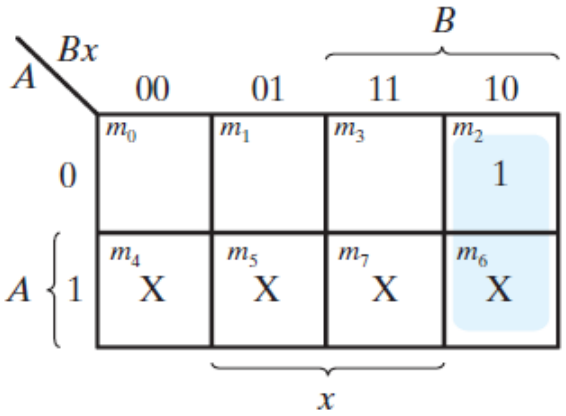


$J_B = x$

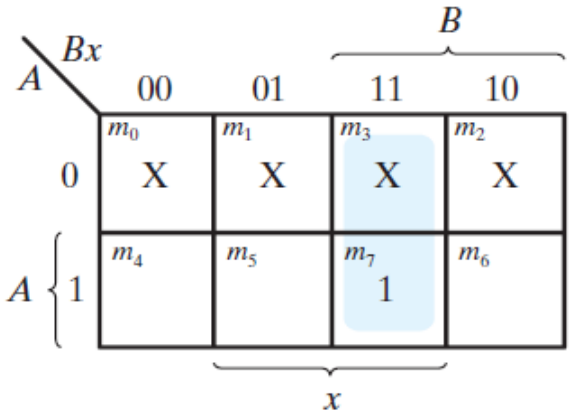


$K_B = (A \oplus x)'$

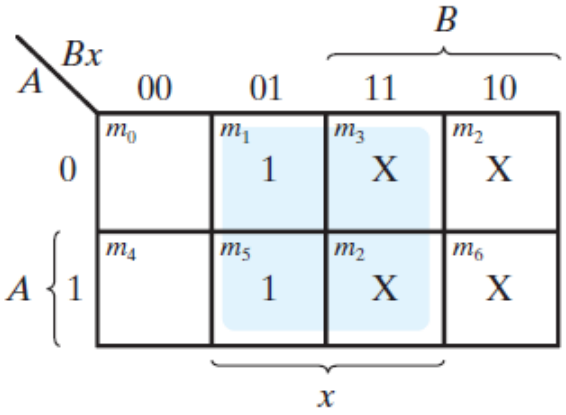
Logic diagram



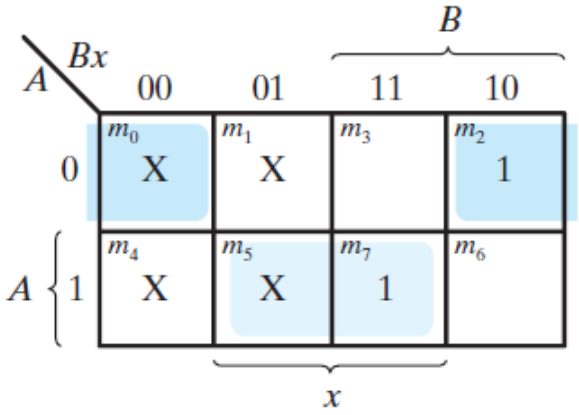
$J_A = Bx'$



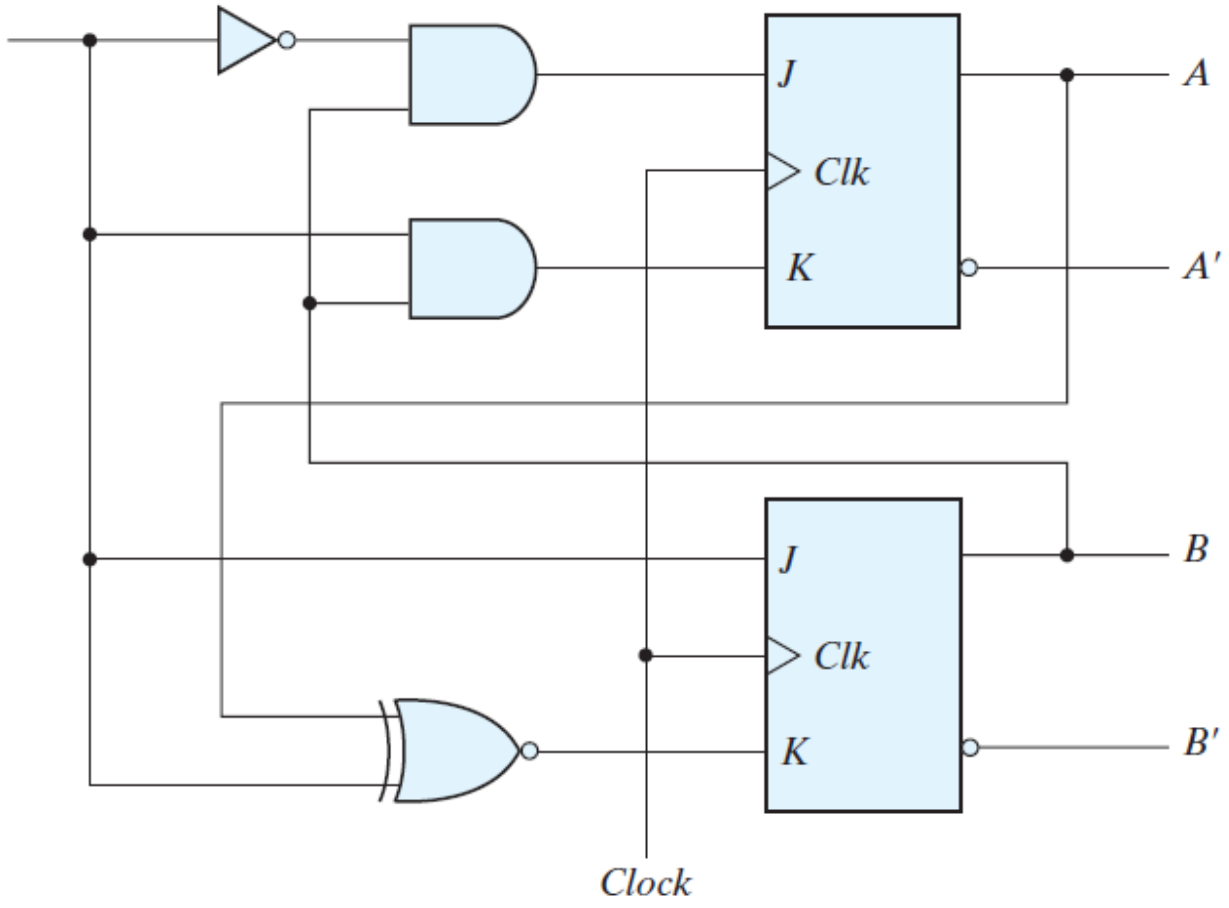
$K_A = Bx$



$J_B = x$



$K_B = (A \oplus x)'$



Comments

Advantage of using JK flip-flops:

There are so many don't-care entries ==> combinational circuit is likely to be simpler.

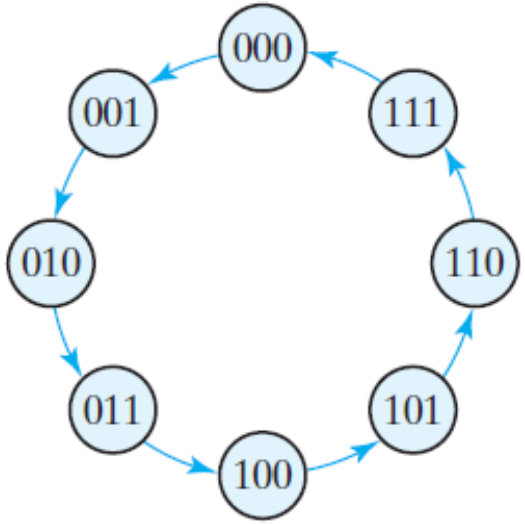
If there are unused states in the state table, there will be additional don't-care conditions in the map.

Nonetheless, D-type flip-flops are more amenable to an automated design flow.

Synthesis using T flip-flops

Three-bit binary counter

Binary counters are constructed most efficiently with *T* flip-flops because of their complement property



State-table

Consider input entries for row 001.
Present state = 001. Next state = 010.

Comparing these two counts, $A2$ goes from 0 to 0, so $TA2$ is marked with 0 because flip-flop $A2$ must not change when a clock occurs.

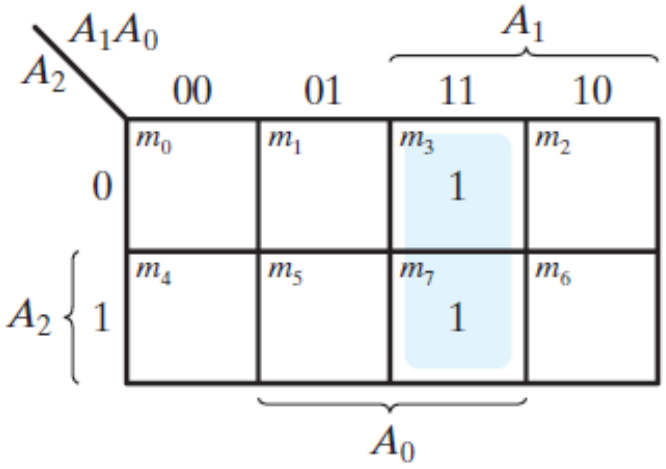
Also, $A1$ goes from 0 to 1, so $TA1$ is marked with a 1 because this flip-flop must be complemented in the next clock edge.

$A0$ goes from 1 to 0, indicating that it must be complemented, so $TA0$ is marked with a 1.

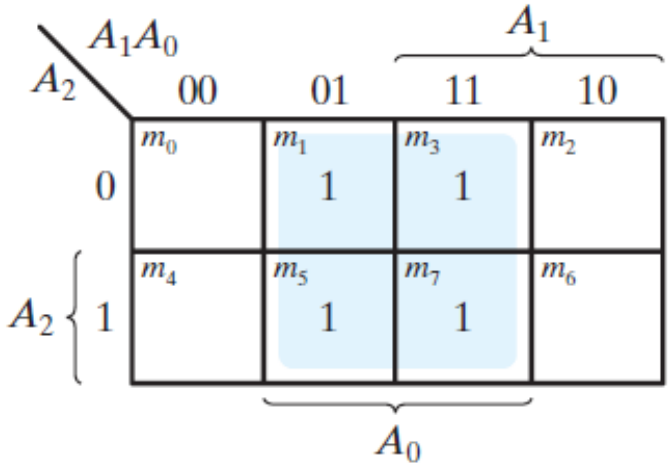
State Table for Three-Bit Counter

Present State			Next State			Flip-Flop Inputs		
A_2	A_1	A_0	A_2	A_1	A_0	T_{A2}	T_{A1}	T_{A0}
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

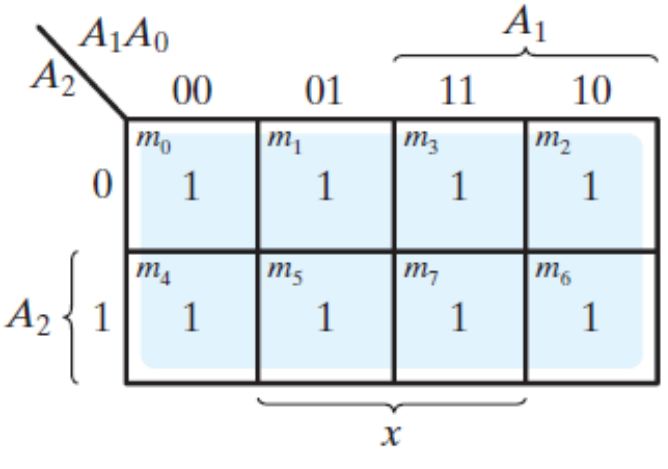
Map and logic-diagram



$T_{A2} = A_1A_0$



$T_{A1} = A_0$



$T_{A0} = 1$