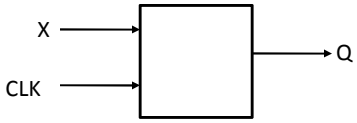
OnePulse – Asynchronous System to Allow One Pulse

Signal Behavior

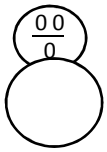
- Understand problem – in all aspects
- Describe problem – state desired behavior. It is imperative that you understand desired behavior in normal environment.
- Create a primitive state diagram of desired behavior. Identify desired output sequences for specific input sequence.

Input-Output Signals



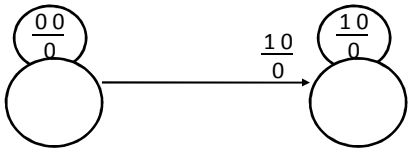
State Diagram

$\frac{X\ CLK}{Q}$



X CLK
Q

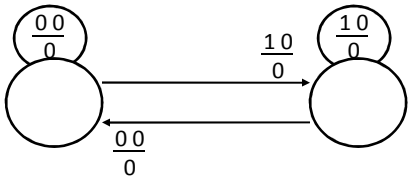
State Diagram



00
0

X CLK
Q

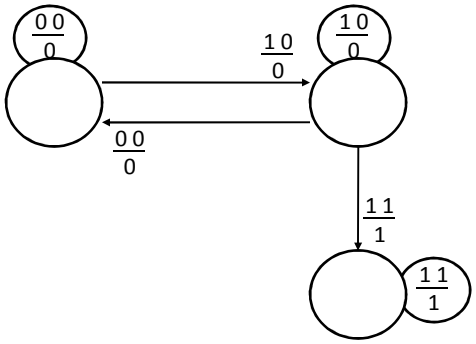
State Diagram



00
0

X CLK
Q

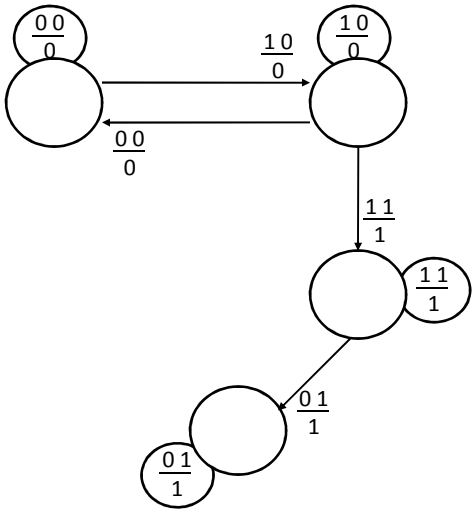
State Diagram



00
0

X CLK
Q

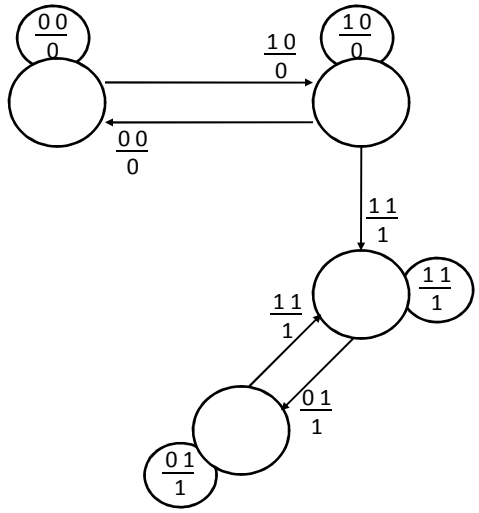
State Diagram



00
0

X CLK
Q

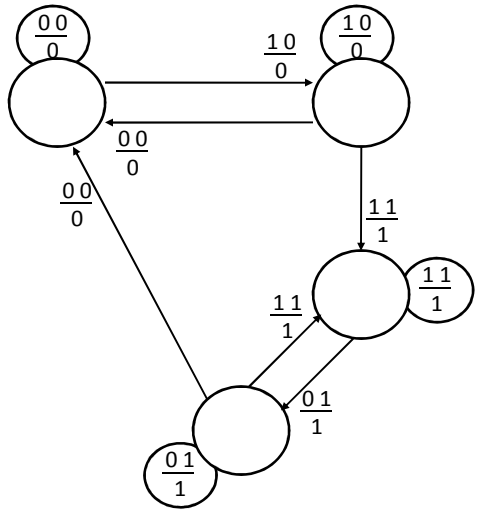
State Diagram



00
0

X CLK
Q

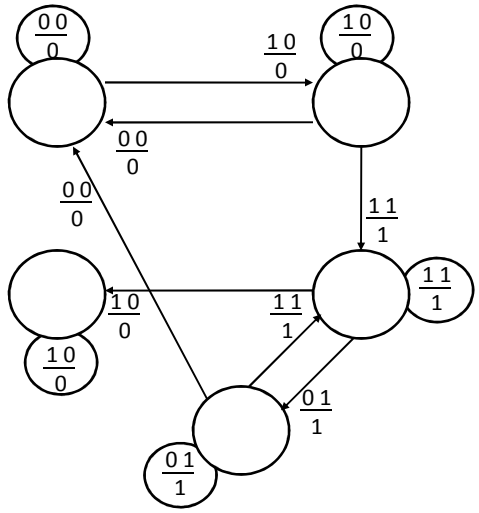
State Diagram



00
0

X CLK
Q

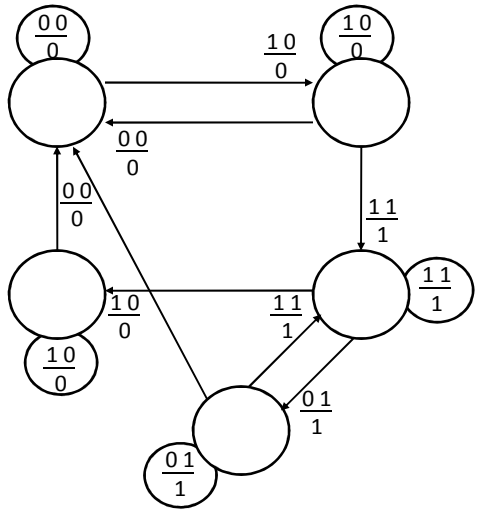
State Diagram



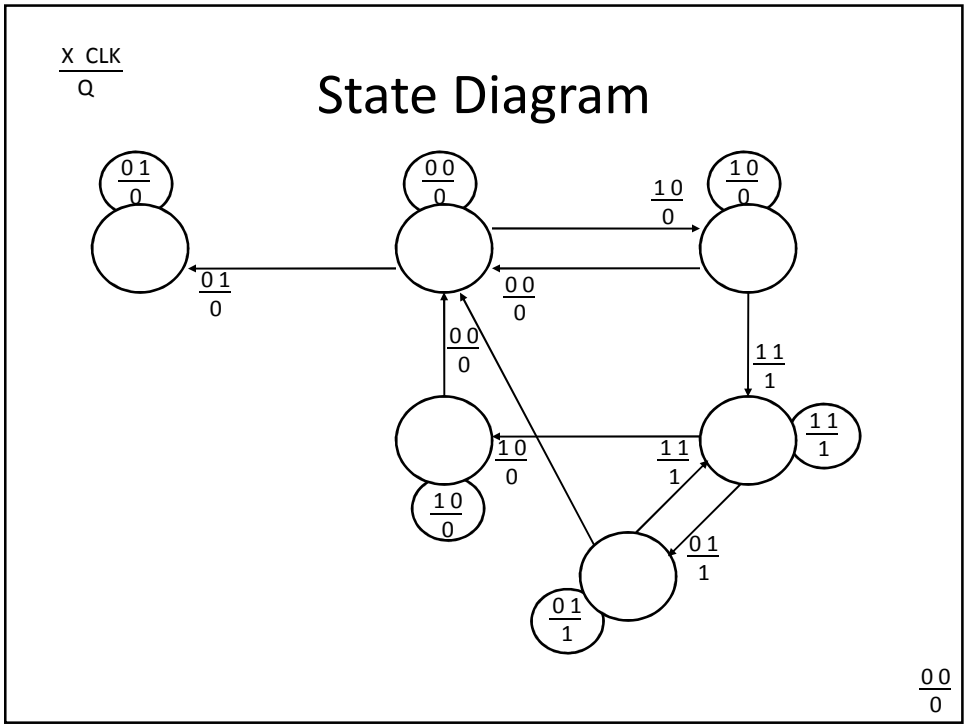
00
0

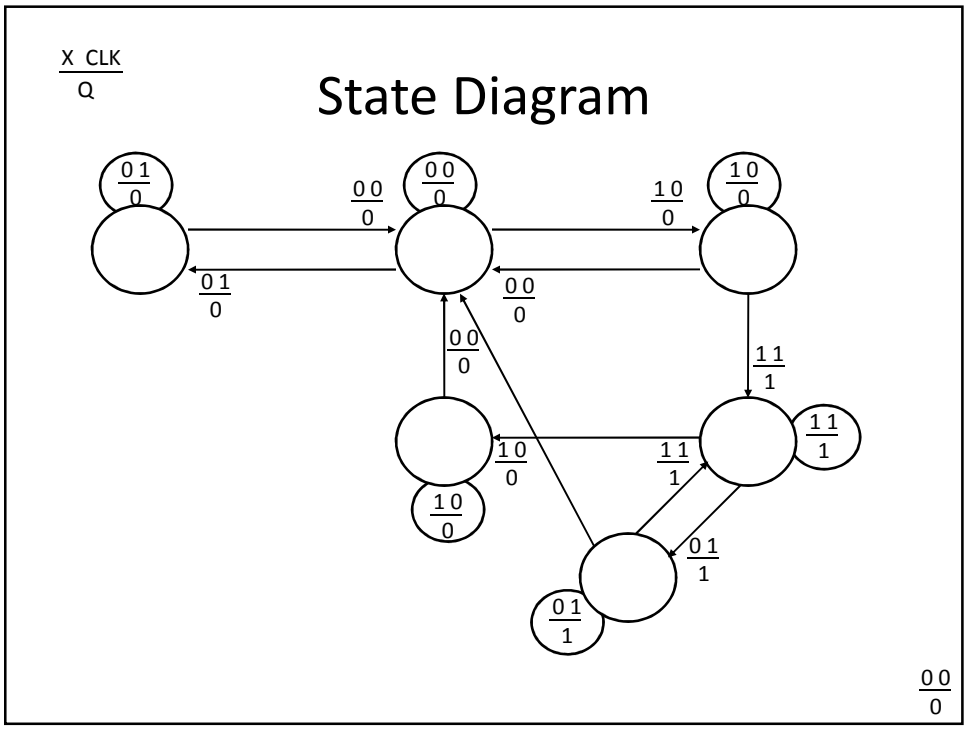
X CLK
Q

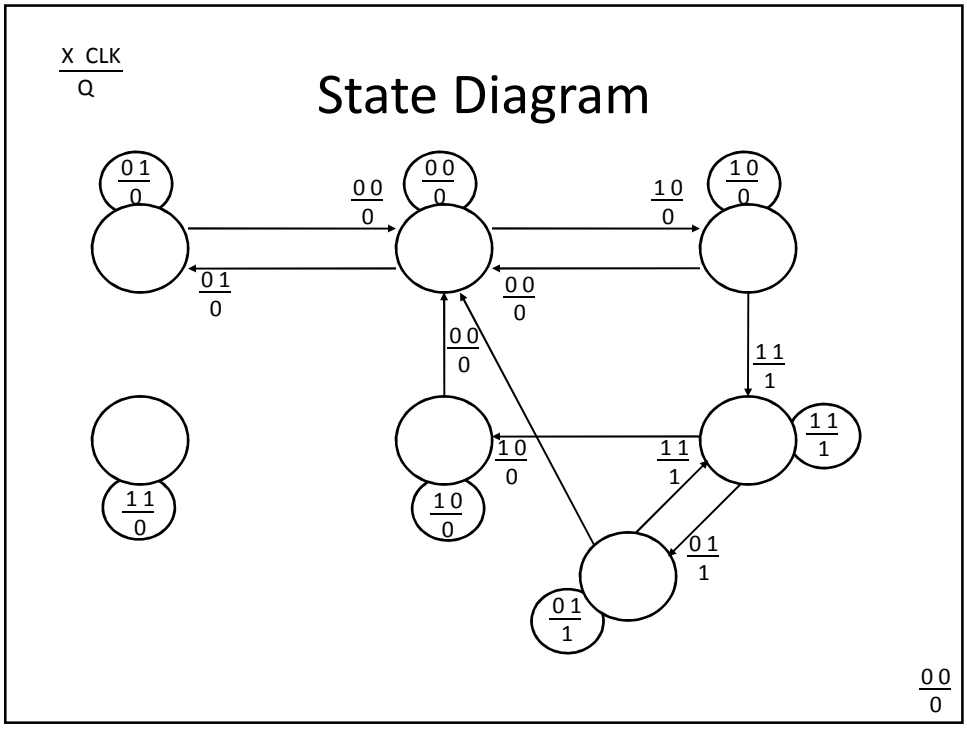
State Diagram

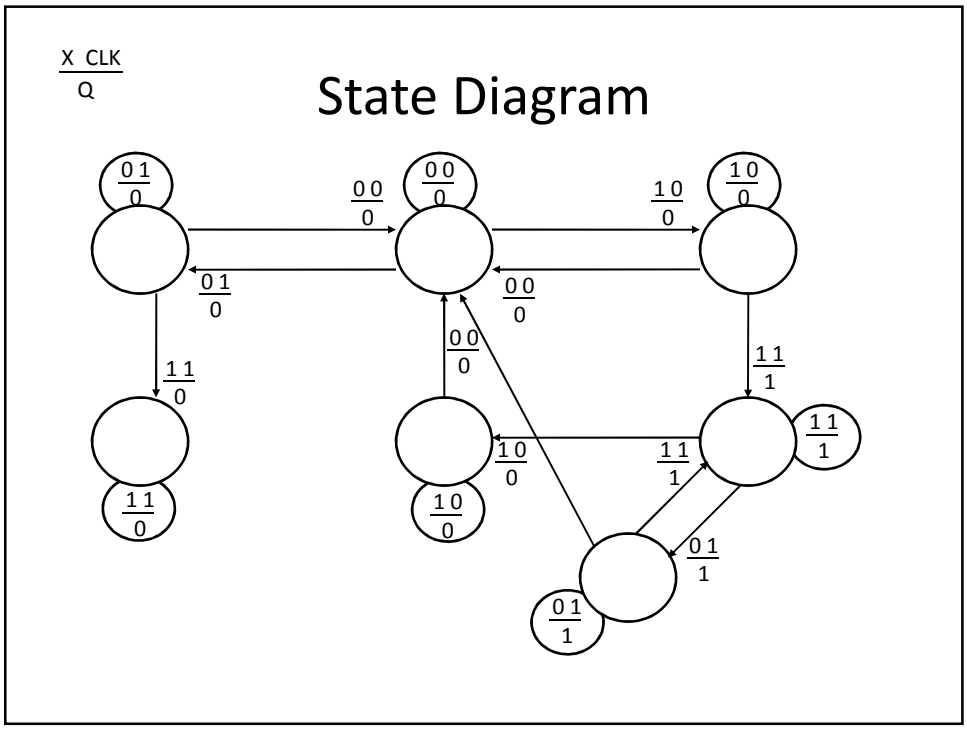


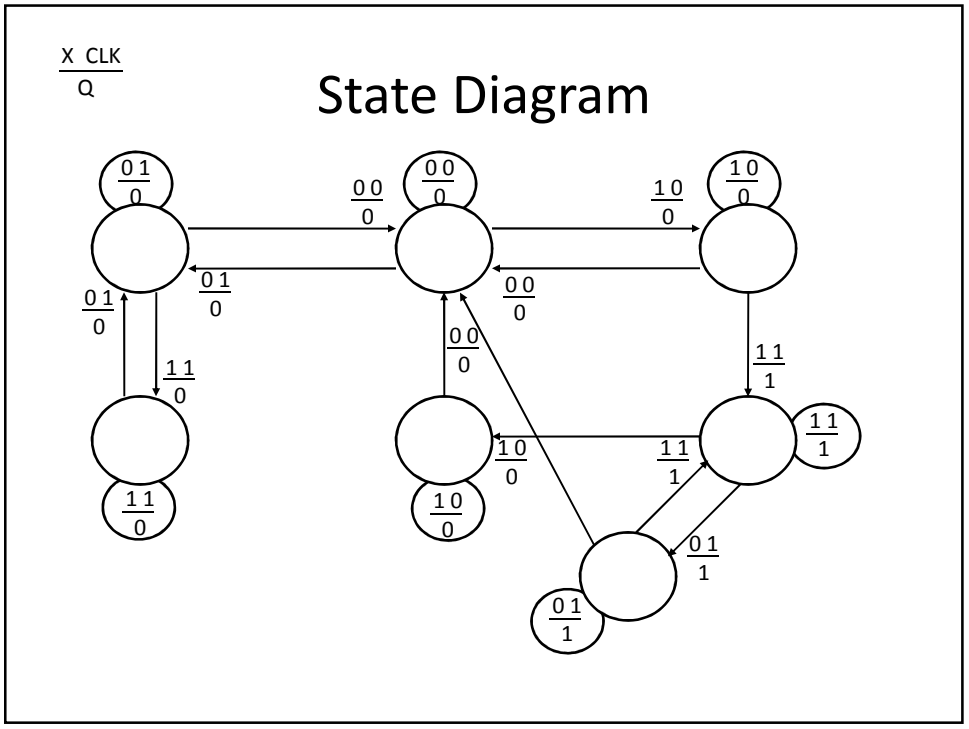
00
0

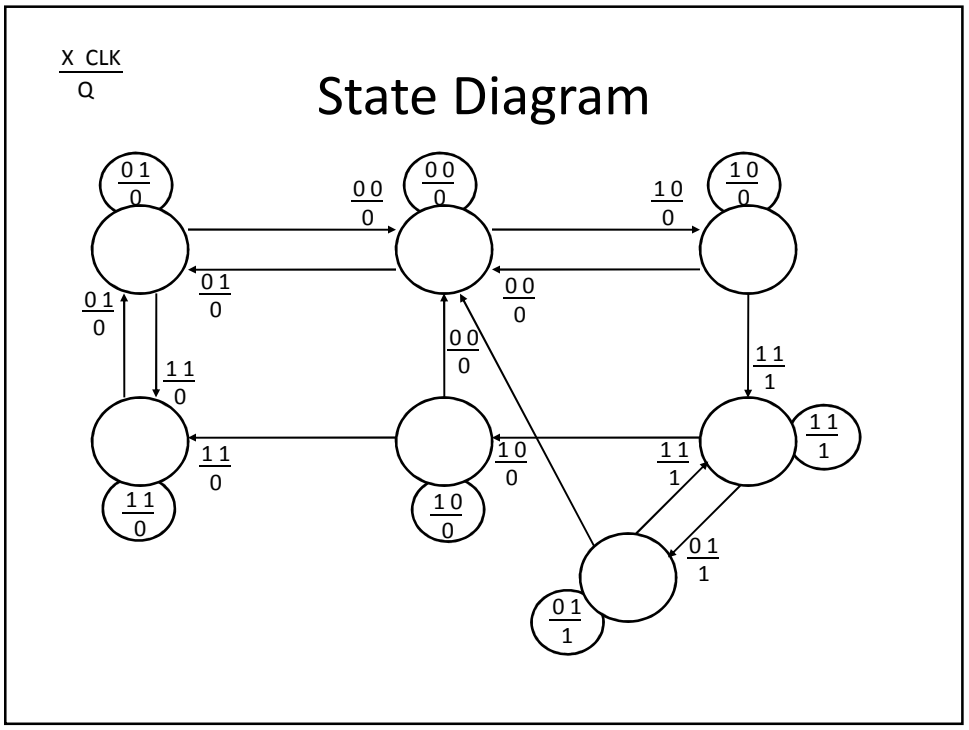

X CLK
Q
State Diagram
01 0
00 0
10 0
01 0
00 0
10 0
00 0
11 1
10 0
11 1
11 1
10 0
01 1
01 1
00 0


X CLK
Q
State Diagram
01 0
00 0
00 0
10 0
01 0
00 0
00 0
11 1
10 0
11 1
11 1
10 0
01 1
01 1
00 0


X CLK
Q
State Diagram
01 0
00 0
10 0
11 0
11 1
10 0
01 1
00

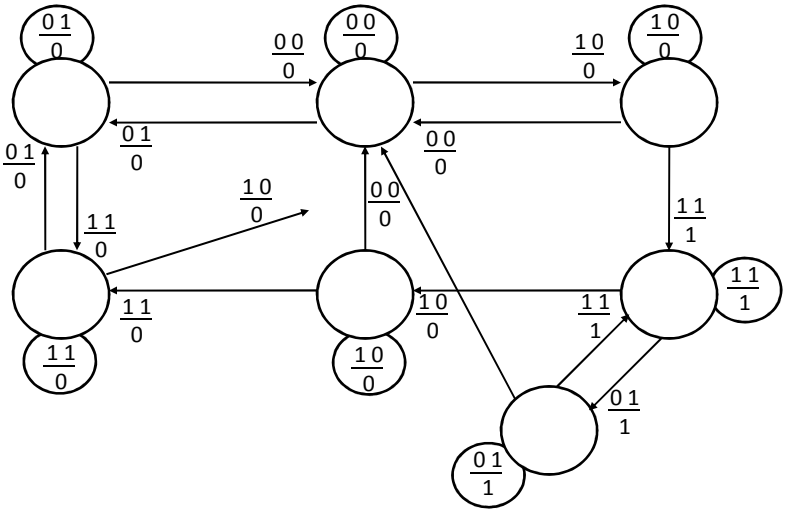

X CLK
Q
State Diagram
01 0
00 0
10 0
11 0
11 1
10 0
01 1


X CLK
Q
State Diagram
01 0
00 0
10 0
01 0
00 0
01 0
00 0
11 0
11 1
11 0
10 0
10 0
11 1
11 1
01 1
01 1


X CLK
Q
State Diagram
01 0
00 0
10 0
01 0
00 0
01 0
00 0
11 0
11 1
11 0
10 0
10 0
11 1
11 1
01 1
01 1

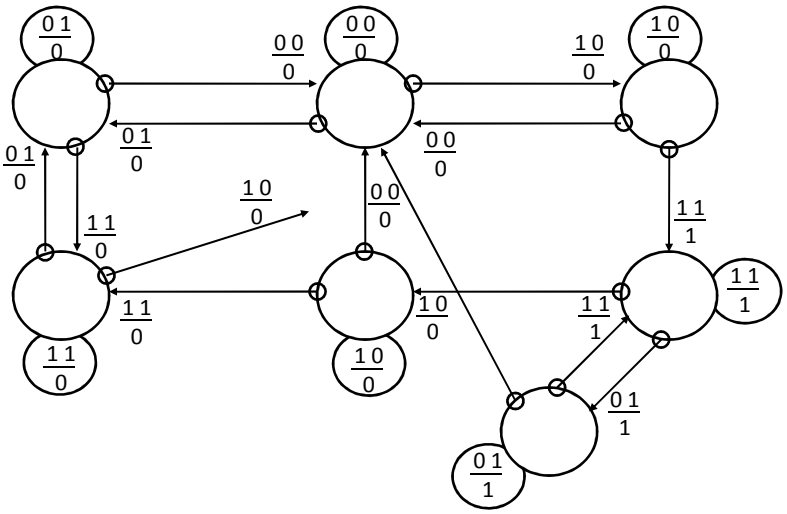
X CLK
Q

State Diagram



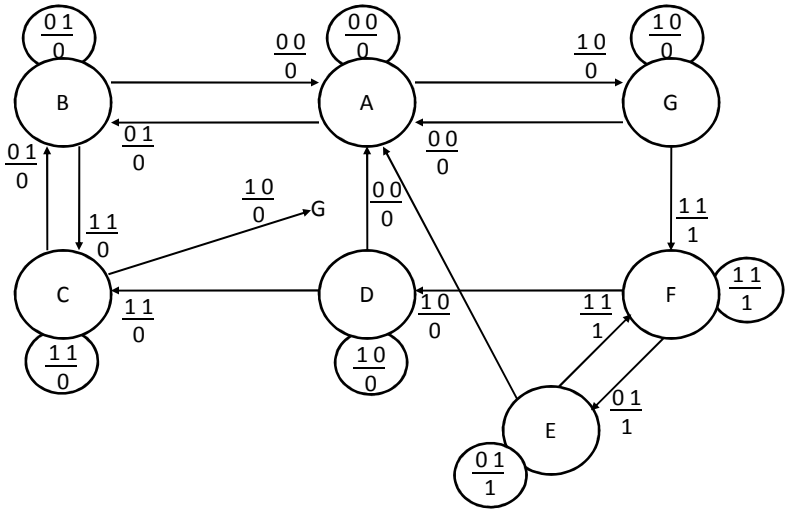
X CLK
Q

State Diagram



X CLK
Q

State Diagram



Next Step: Primitive State Table

Primitive State Table: one row for each state in state diagram. Each stable state (the state that is associated with that row) is circled to indicate that it is stable, and possible next states are included in the row (un-circled)

Primitive State Table

00	01	11	10
(A)	B		G

Primitive State Table

00	01	11	10
(A)	B		G
A	(B)	C	
	B	(C)	G
A		C	(D)
A	(E)	F	
	E	(F)	D
A		F	(G)

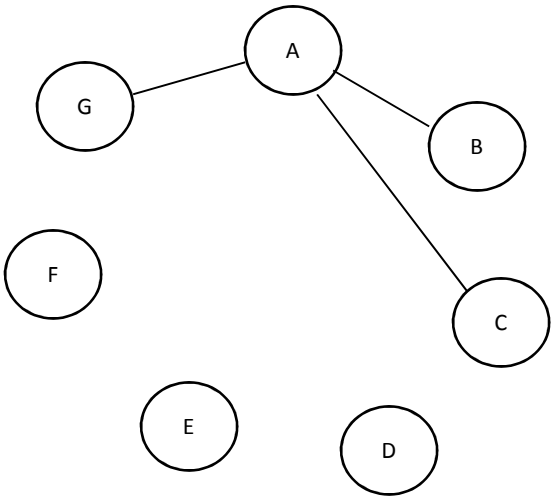
Primitive State Table

	00	01	11	10
0	(A)	B		G
0	A	(B)	C	
0		B	(C)	G
0	A		C	(D)
1	A	(E)	F	
1		E	(F)	D
0	A		F	(G)

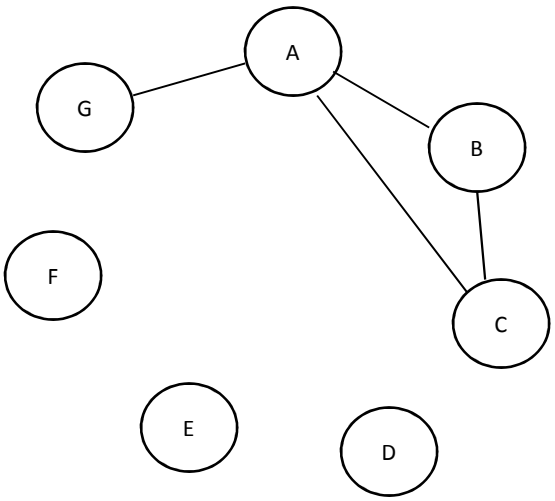
Next Step: Merge Diagram

Merge Diagram: Determine which states can merge with other states in a single row of a revised state table. Rows can merge when the entries in corresponding columns are equal or empty. Merge diagram contains one 'bubble' for each row in the primitive state table.

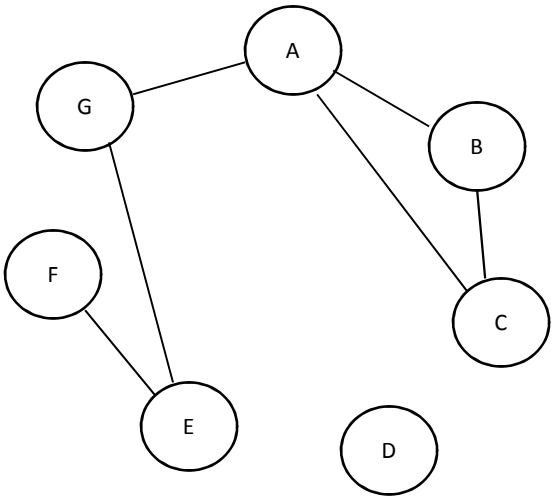
Merge Diagram



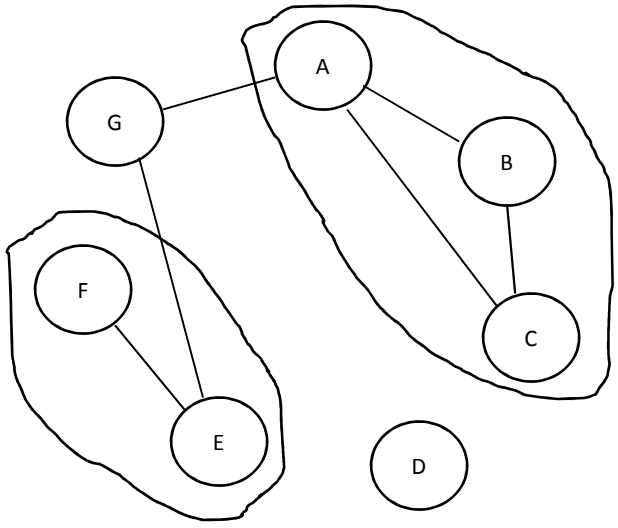
Merge Diagram



Merge Diagram



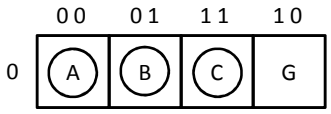
Merge Diagram



New (Final) State Table

Final State Table: Construct a state table from merged rows of the primitive state table. This state table will be a K-Map of the final logic system. Note that it is probably a good idea to try different combinations in order to find an optimal logic solution.

Final State Table



Final State Table

	00	01	11	10
0	(A)	(B)	(C)	G
0	A		F	(G)

Final State Table

	00	01	11	10
0	(A)	(B)	(C)	G
0	A		F	(G)
0				
0	A		C	(D)

Final State Table

	00	01	11	10
0	(A)	(B)	(C)	G
0	A		F	(G)
1	A	(E)	(F)	D
0	A		C	(D)

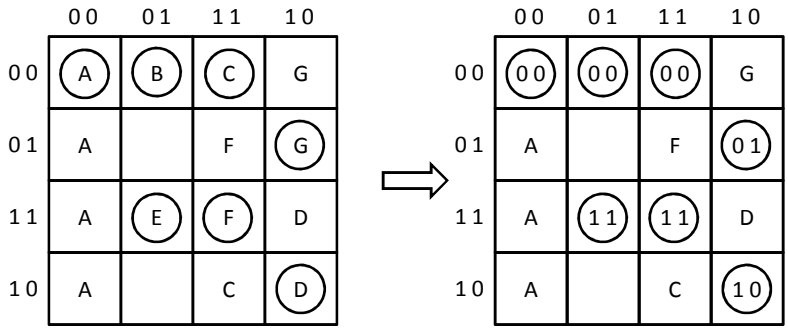
Logic Signals (Inputs) vs Feedback

Construct system with following guidelines: input combinations are listed along top of table (K-Map) in grey-code fashion. Feedback combinations are shown on side of table. Result is K-Map of system. Changes in inputs cause horizontal changes in state of system. Changes in feedback elements cause vertical changes in state of system.
Note that all adjacent minterms in final state table must be covered by a gate.

Constructing Final K-Map

Step 1 in constructing final K-Map: for all squares of the K-Map that indicate stable states, make contents equal to the feedback variables for that row.

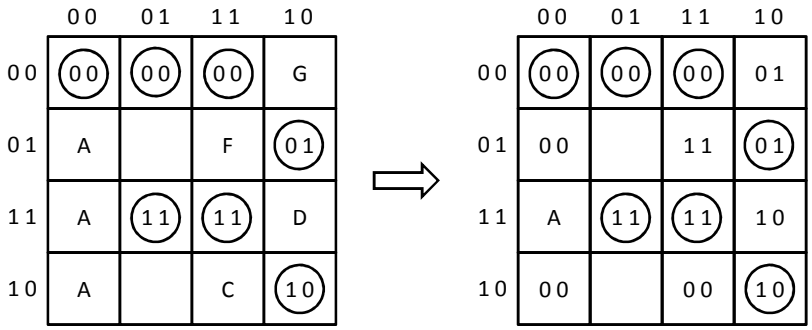
Final State Table



Constructing Final K-Map

Step 2 in constructing final K-Map: for all squares of the K-Map that identify unstable states that are logically adjacent to stable states, make feedback variables for that square (the unstable state) the same as the adjacent stable state.

Final State Table

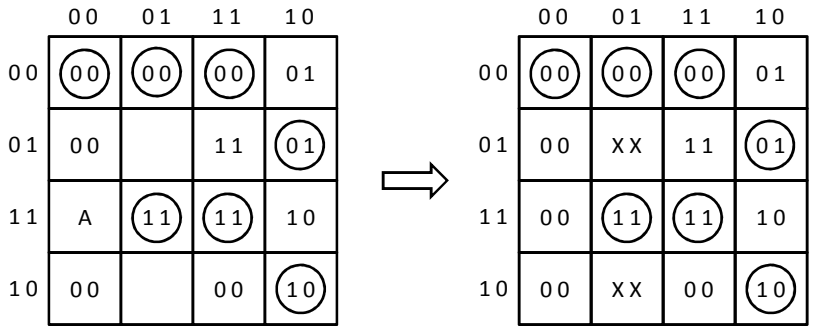


Constructing Final K-Map

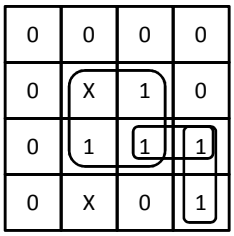
Step 3 in constructing final K-Map: for all remaining squares, make reasonable logic assumptions and fill the squares appropriately.

Step 4: split out feedback variables and generate logic equations. Be careful to make single AND-OR Implementation.

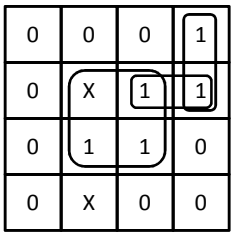
Final State Table



Final State Tables



$$f1 = CLK \bullet f0 + X \bullet \overline{CLK} \bullet f1 + X \bullet f1 \bullet f0$$

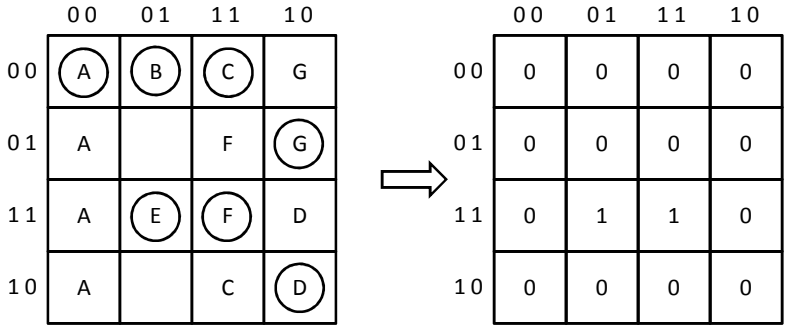


$$f0 = CLK \bullet f0 + X \bullet \overline{CLK} \bullet \overline{f1} + X \bullet \overline{f1} \bullet f0$$

Output Map

Output is asserted only in stable states corresponding to those where output is asserted.

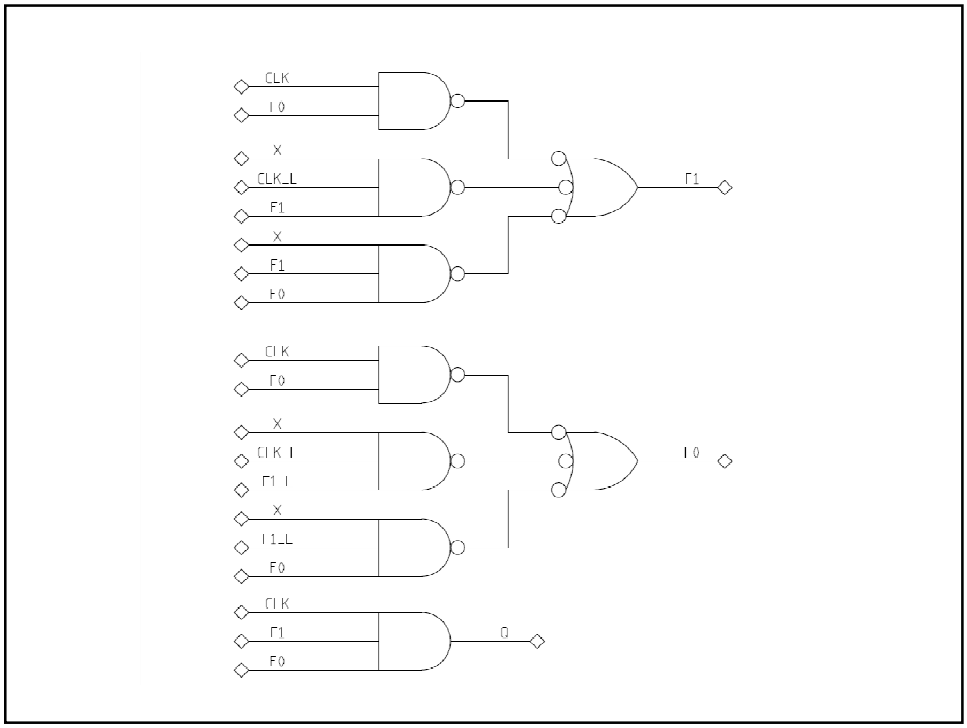
Output Map



Output Map

	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	1	1	0
10	0	0	0	0

$$Q = CLK \bullet f1 \bullet f0$$


CLK
I0
X
CLK_L
F1
X
F1
F0
F1
CLK
F0
X
CLK_L
F1_L
X
I1_L
F0
CLK
F1
F0
Q
I0