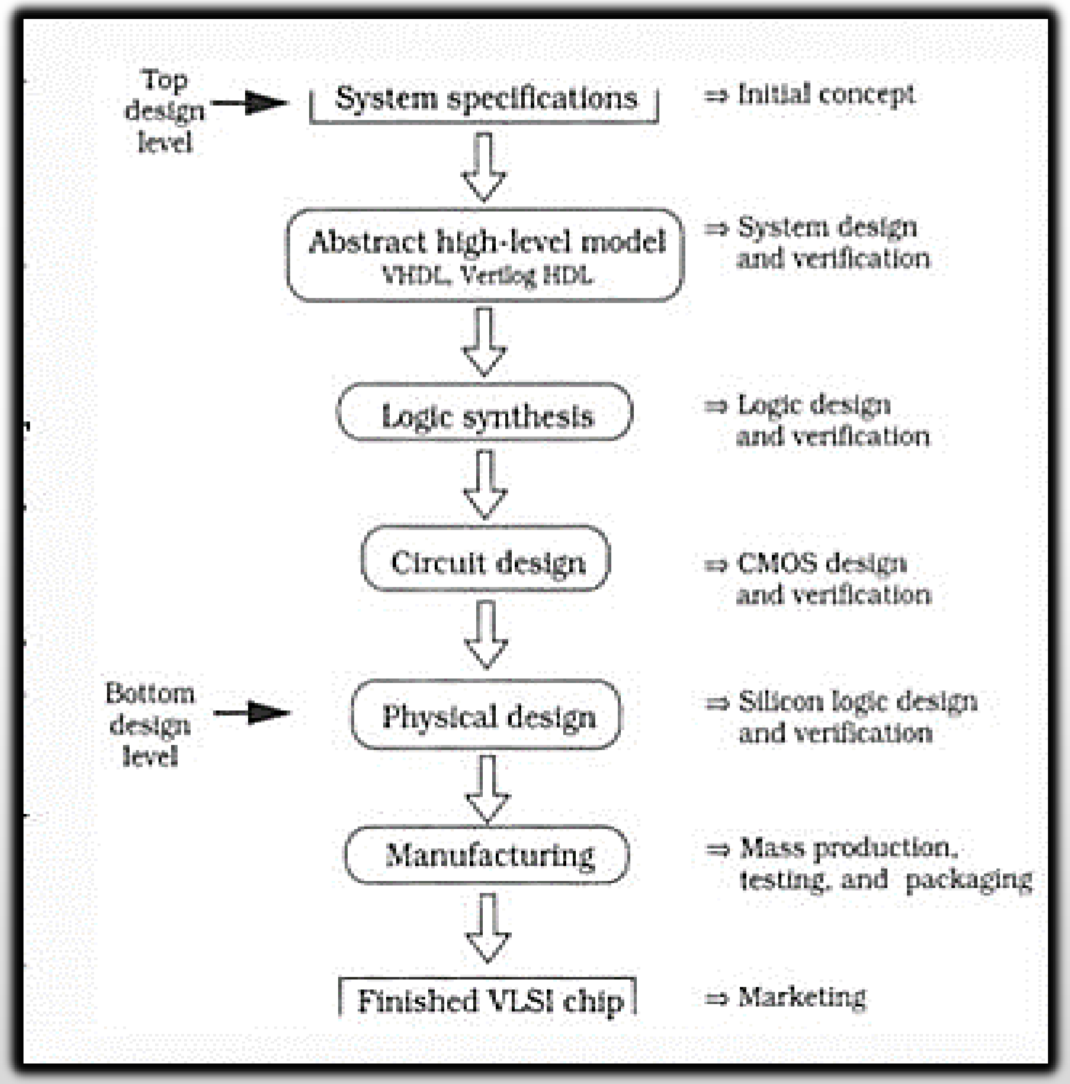
Digital System Design EC 503

Digital System Design Hierarchy

Subject Coordinator
Dr. GAURAV KUMAR BHARTI
Department of ECE

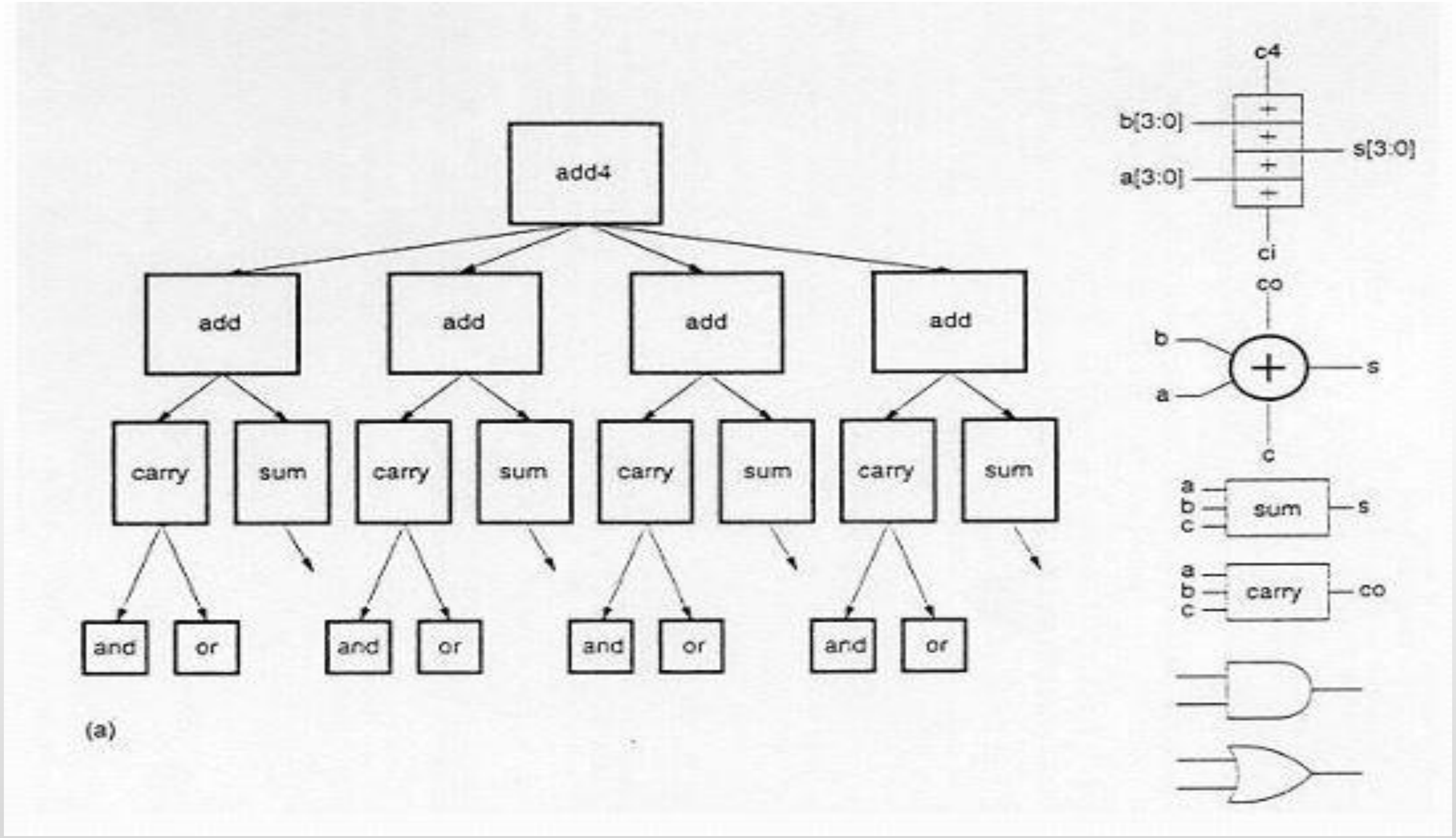
Design Hierarchy

- System specifications: is defined in both general and specific terms, such as functions, speed, size, etc.
- Abstract high-level model: contains information on the behavior of each block and the interaction among the blocks in the system
- Logic synthesis: To provide the logic design of the network by specifying the primitive gates and units needed to build each unit
- Circuit design: where transistors are used as switches and Boolean variables are treated as vary voltage signals
- Physical design: the network is built on a tiny area on a slice of silicon
- Manufacturing: a completed design process is moved on to the manufacturing line



Top design level
System specifications
⇒ Initial concept
Abstract high-level model
VHDL, Verilog HDL
⇒ System design and verification
Logic synthesis
⇒ Logic design and verification
Circuit design
⇒ CMOS design and verification
Bottom design level
Physical design
⇒ Silicon logic design and verification
Manufacturing
⇒ Mass production, testing, and packaging
Finished VLSI chip
⇒ Marketing

Example of Design Hierarchy



Hierarchical design

- **Top to down design :**
  - The initial work is quite abstract and theoretical and there is no direct connection to silicon until many steps have been completed
  - Acceptable in modern digital system design, Co-design with combining HW/SW is critical, Similar to Cell-based Design Flow
- **Bottom to up design:**
  - starts at the silicon or circuit level and builds primitive units such as logic gates, adders, and registers as the first steps
  - Acceptable for small projects , Similar to Full-custom Design Flow
- Based on divide and conquer
- Dividing a module into sub- modules and then repeating this operation on the sub-modules until the complexity of the smaller parts becomes manageable.
- In fig.,CMOS four-bit adder into its components.
- The adder can be decomposed progressively into one- bit adders, separate carry and sum circuits, and finally, into individual logic gates. At this lower level of the hierarchy, the design of a simple circuit realizing a well-defined Boolean function is much more easier to handle than at the higher levels of the hierarchy.

ASM Charts

- A State Machine or Algorithmic State Machine(ASM), which is similar to a flow-chart is used to describe the behaviour of a digital system or state machine. Algorithmic State Machine charts are also known as State Machine (SM)Chart.
- The basic difference between an ordinary flow chart and SM chart is that, certain specific rules must be followed in the construction of SM chart, but no such specific rules are to be followed in the case of flow-chart.
- An Algorithmic State Machine Chart can be constructed from the State Graph of a Digital system.
- **ASM CHART –COMPONENTS:**
  - There are three important components in an ASM Chart. They are (i)State Box(ii)Decision Box (iii)Conditional output Box.
- **State Box:** The state box contains a state name followed by a slash(/) and an optional output list. After the state assignment , a state code must be placed outside the box at the top.
- **Decision Box:** Represented by the diamond shape symbol with true and ;False branches. The condition placed in the box is a Boolean expression that is evaluated to determine which branch is true.
- **Conditional Output Box:** Has a curved ends contains a conditional output list. The conditional outputs depend on both the state of the symbol and inputs.

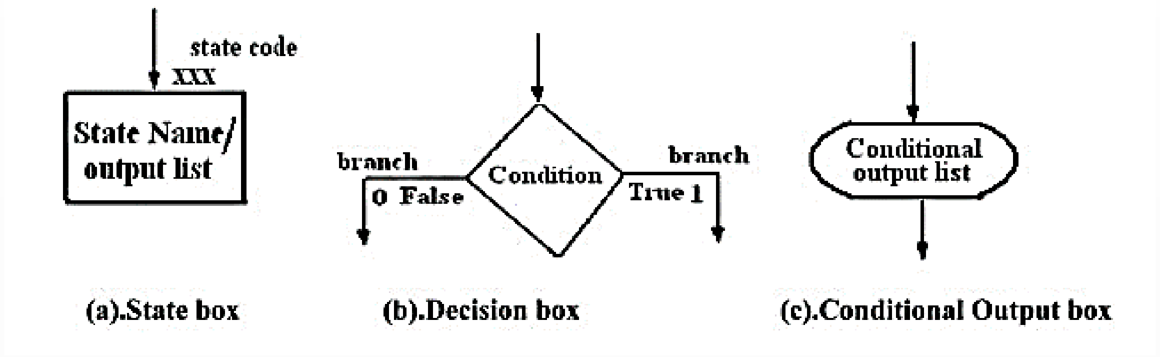


Fig1 : ASM Chart Components

- Specific Rules for constructing ASM Chart
  - For every valid combination of input variables, there must be exactly one exit path defined. This is necessary because ,each allowable input combination must lead to a single next state.
- The second rule is no internal feedback within an SM Block is allowed. This is shown in the figure 2

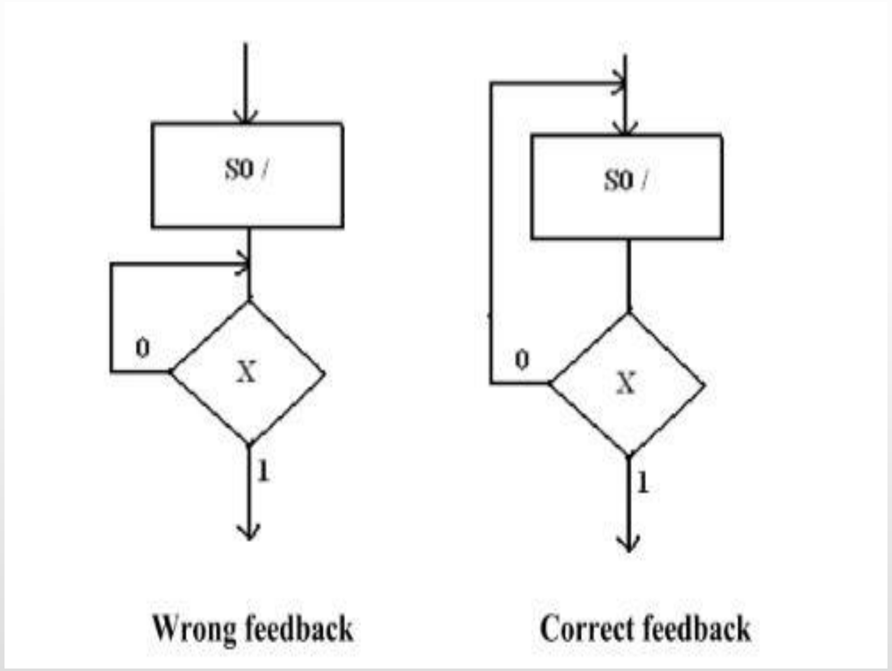


Fig 2

Conversion of State Diagram to An ASM Chart

- ASM chart can be derived derived an ASM from state diagram of machine ,but certain rules must be followed when constructing an ASM block. First for every valid combination of input, there must be exactly one exit path defined .Second ,no internal feedback within an SM block is allowed.
- **Mealy Machine.** In case of Mealy machine ,output is a function of both present state and input . For construction of ASM chart from Mealy state diagram ,we should follow the following steps.
- 1. Represent each states by state boxes.
- 2. Put input in decision box after each state box.
- 3. The Mealy output appear in conditional output boxes since they depend on both the state and input.
- 4. Mealy circuit output written only when it is equal to '1' i.e. true .
- 5. Depending on value of input connect the path to next state box.

Example

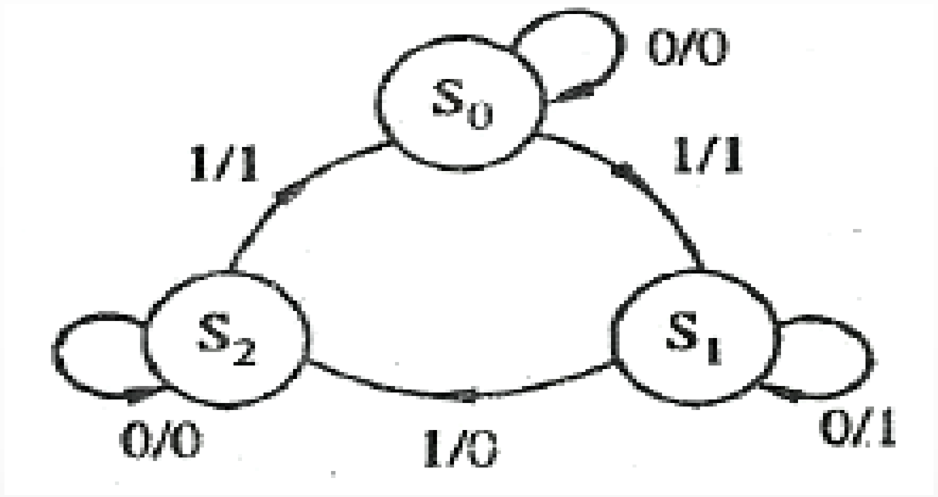


Fig : State Diagram

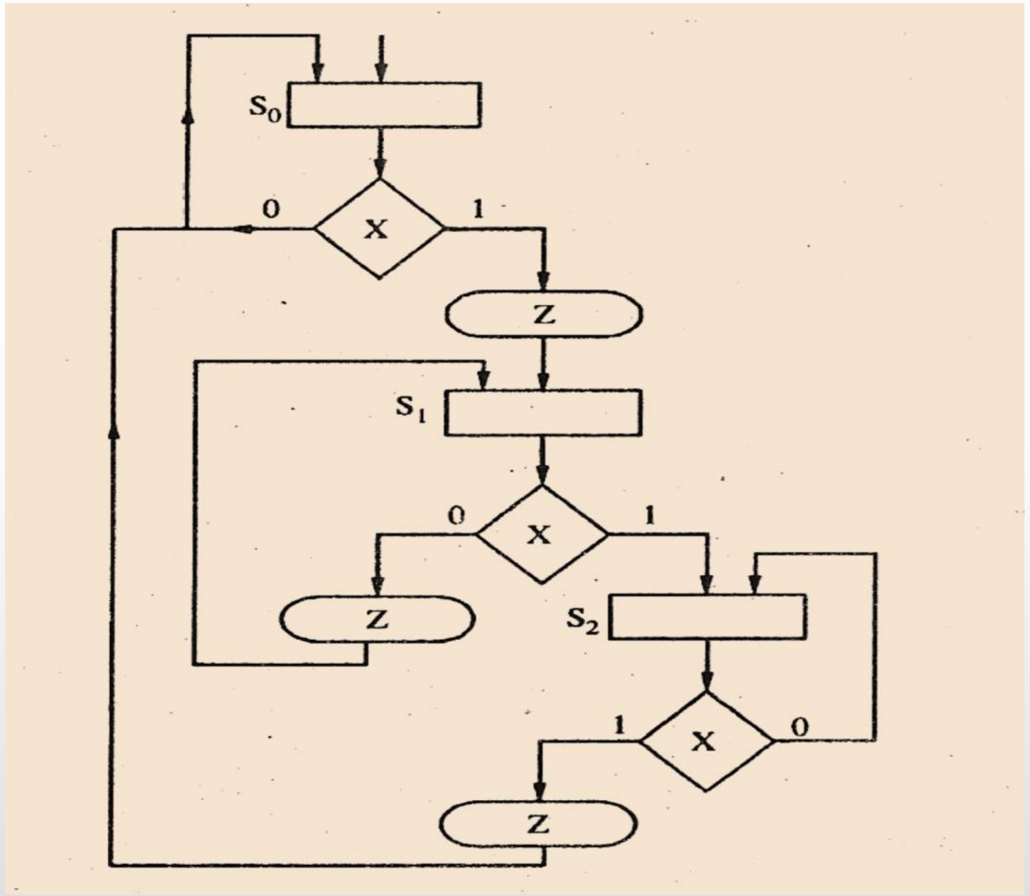


Fig : ASM Chart

- **Moore Machine .** In case of Moore machine ,output is a function of the present state only . For construction of ASM chart from Moore state diagram ,we should follow the following steps
  - 1. Represent each states by state boxes.
  - 2. The Moore output are placed in the state boxes since they do not depend on the input .
  - 3. After each state box put the input in decision box.
  - 4. Depending on value of input connect the path to next state box.
- **Example :**

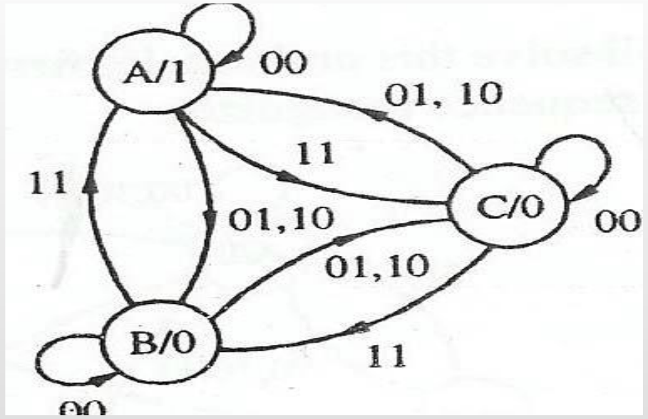


Fig: State Diagram

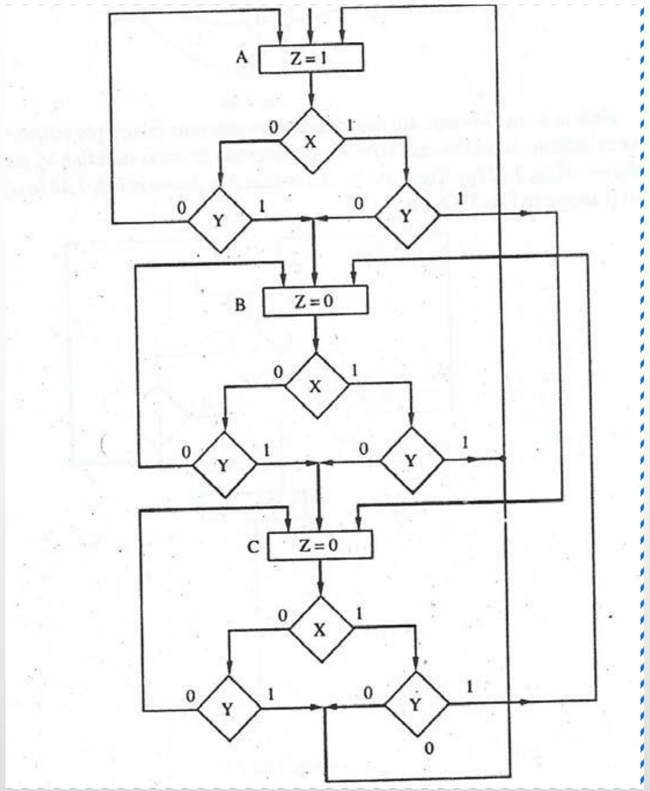
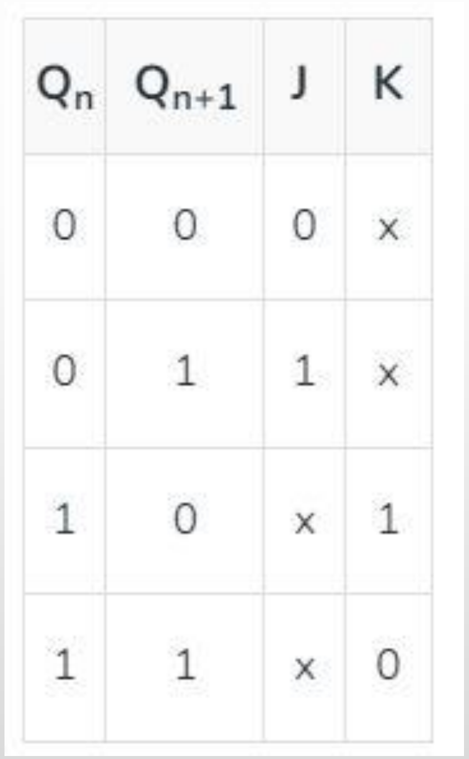


Fig: ASM Chart

Reduction of State Tables, State Assignments

- **State diagram:** The state graph or state diagram is a pictorial representation of the relationships between the present state, the input state, the next state, and the output state of a sequential circuit i.e. A state diagram is a graphical representation of a sequential circuit's behavior.
- **Example:** Consider an excitation table of J-K flip-flop



Q_n	Q_{n+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

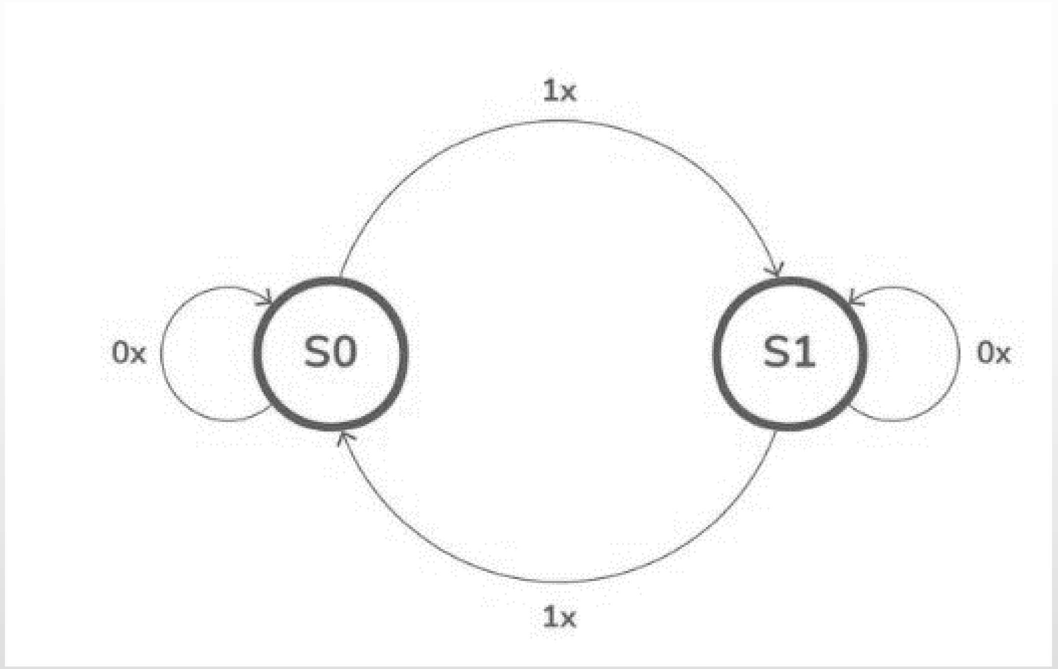
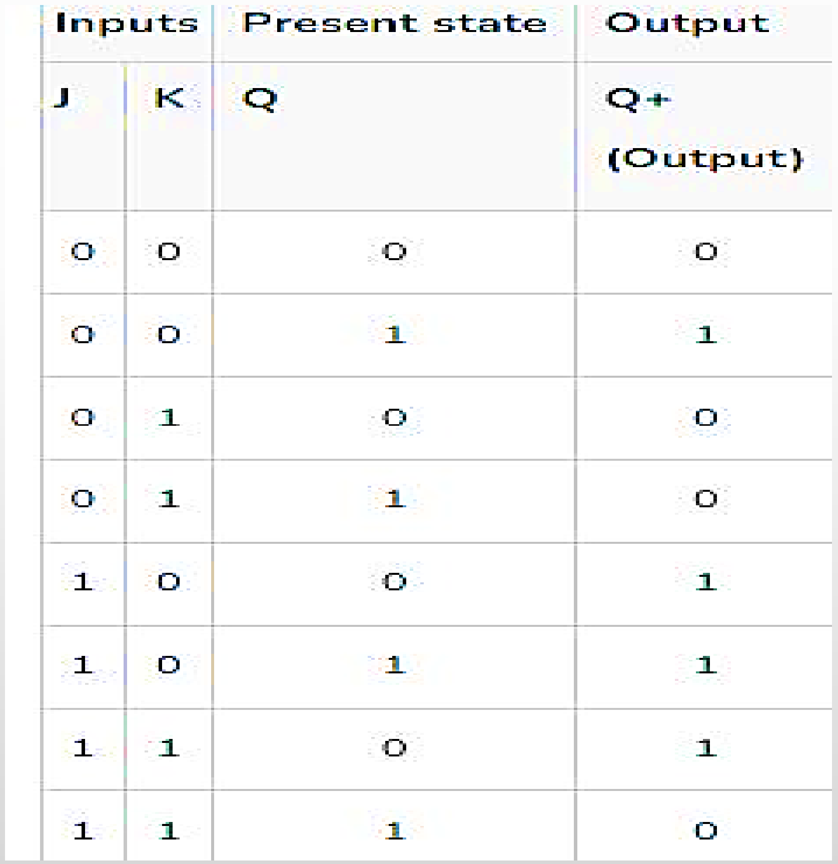


Fig: State diagram of *J-K flip-flop*

- **State table :** Even though the behavior of a sequential circuit can be conveniently described using a state diagram, for its implementation the information contained in the state diagram is to be translated into a state table. The tabular form of the state diagram is the state table. The present state, the next state, and the output are the three sections of the diagram.
- The state table of JK flip-flop is:
- **State equation:** $Q_{n+1} = Q_n$ bar J + Q_n K bar

State reduction:

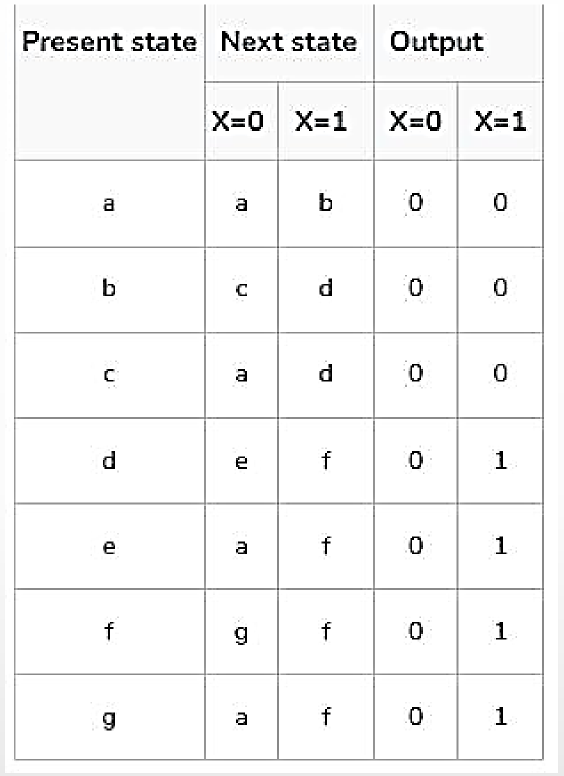
- The state reduction technique generally prevents the addition of duplicate states.
- The reduction in redundant states reduces the number of flip-flops and logic gates, reducing the cost of the final circuit.
- Two states are said to be equivalent if every possible set of inputs generates exactly the same output and the same next state.
- When two states are equal, one of them can be eliminated without changing the input-output relationship.
- The state reduction algorithm is applied in the state table to reduce equivalent states.



Inputs		Present state	Output
J	K	Q	Q+ (Output)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

- **State assignment**
- State assignment refers to the process of assigning binary values to the states of a sequential machine. The binary values should be given to the states in such a way that flip-flop input functions may be implemented with a minimum number of logic gates.
- **State assignment rules are as follows:**
- **Rule 1:** States having the same next state for a given input condition should have assignments that can be grouped into logically adjacent cells in a K-map.
- **Rule 2:** States that are the next states of a single state should have assignments that can be grouped into logically adjacent cells in a K-map.

Example



Present state	Next state		Output	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

Fig: State table

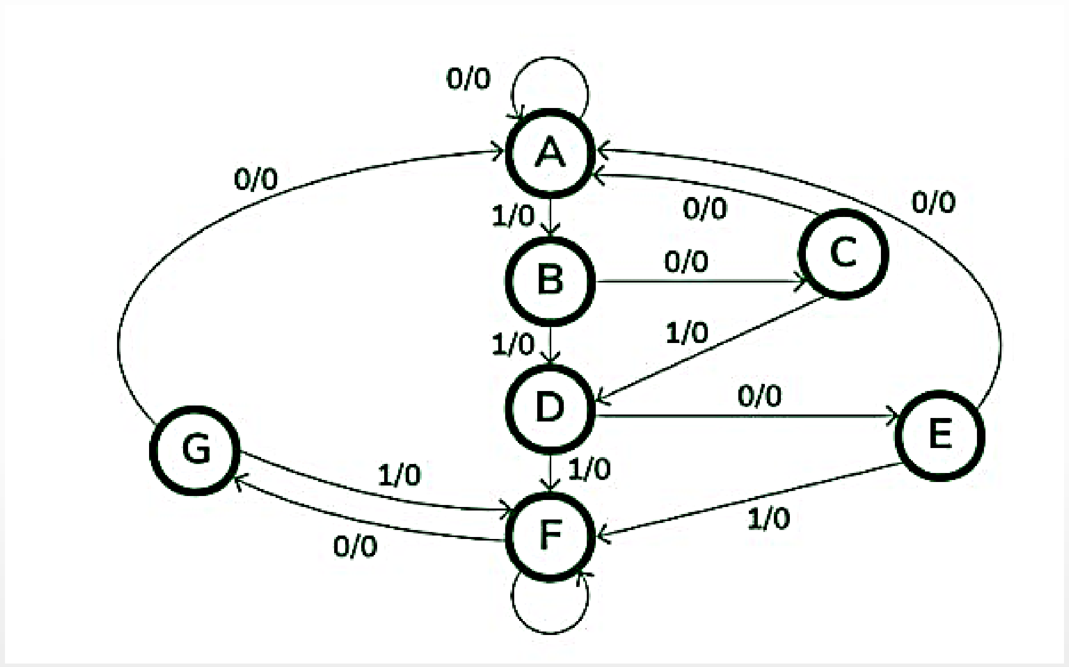
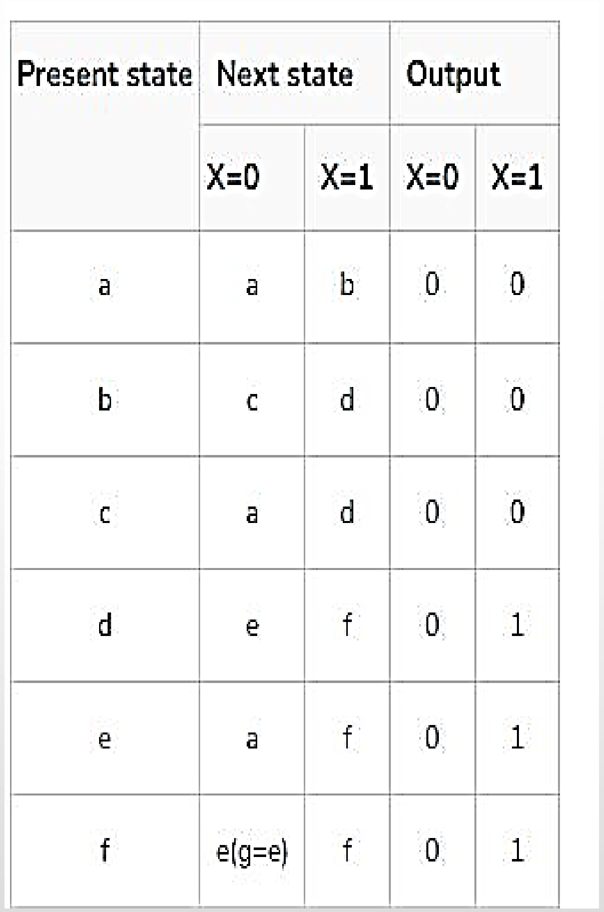


Fig: State diagram for the state table

Step1:

- First here we are supposed to identify two or more similar states in the next state and output state. In the above table if we observe states of e and g are having the same next state and output values for all combinations of input i.e. X=0 and X=1.
- So eliminate the g state in the state table and wherever g is present replace it with e. Because e and g both are the same i.e. e=g.

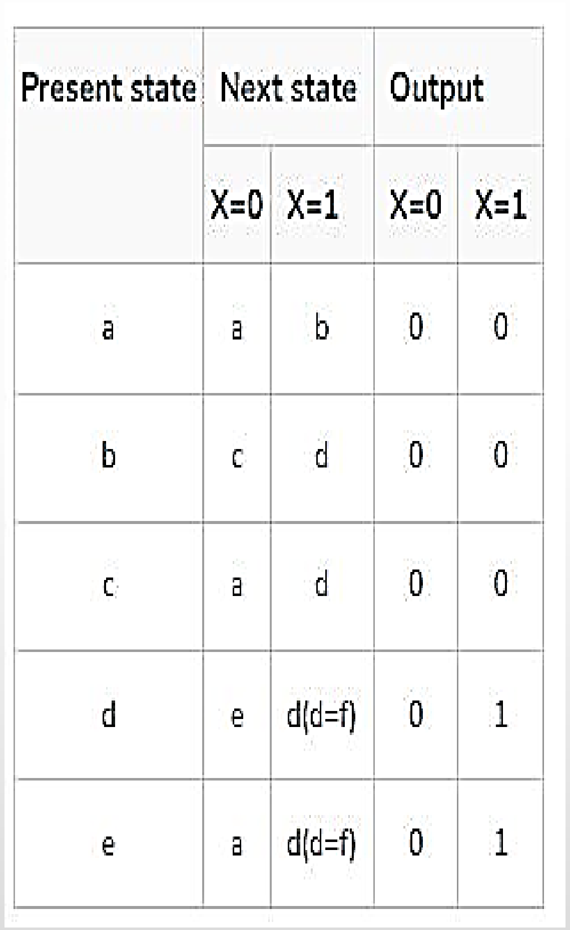


Present state	Next state		Output	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e(g=e)	f	0	1

Fig : Step 1

Step 2:

- Again check if any two states have similar values or not. If any two states have the same next state and output then eliminate one state.
- Here d and f are having the same next state value and output. So eliminate f and wherever f is present replace it with d. Because both are the same d=f



Present state	Next state		Output	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d(d=f)	0	1
e	a	d(d=f)	0	1

Fig : Step 2

- **Step 3:** Further observe if any similar states are present or not. The states c and e are having same next states but they are having different outputs. So we can not consider it a reduction state.

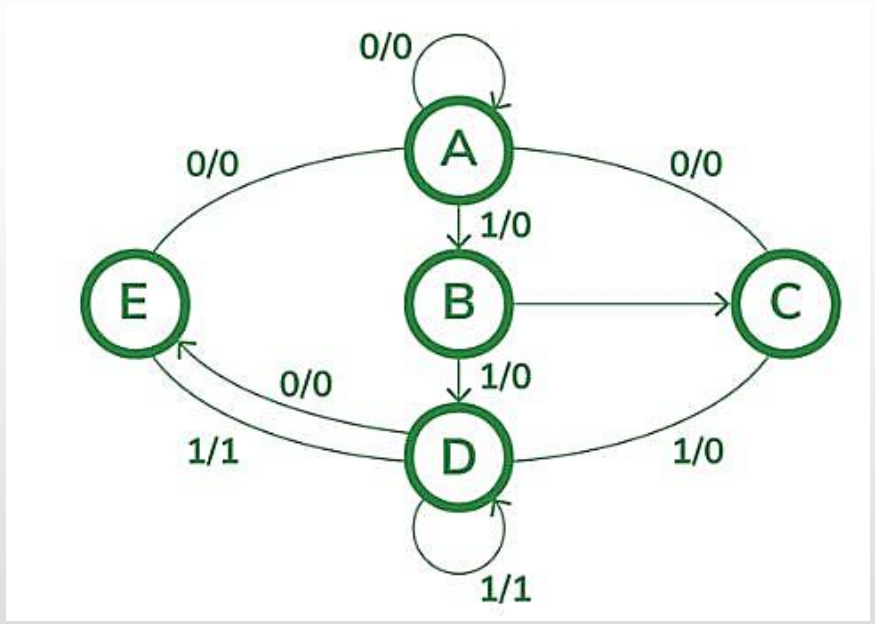
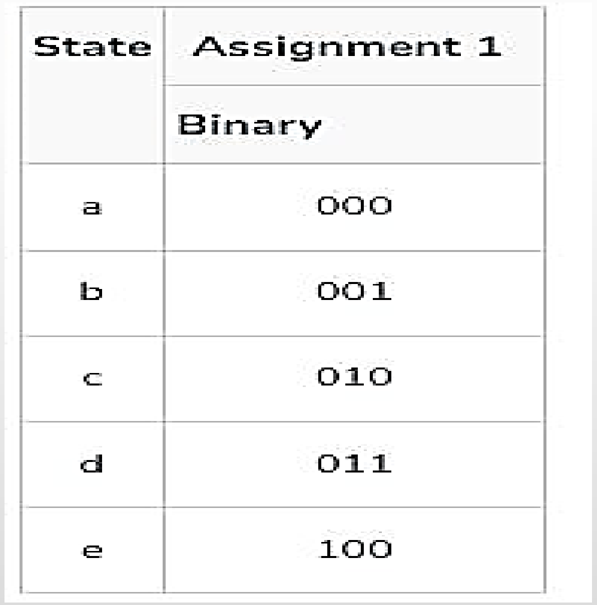


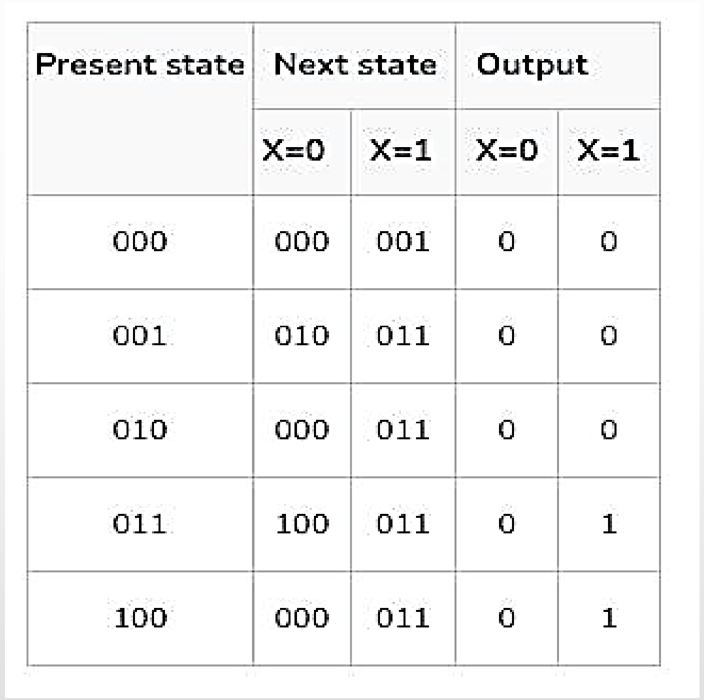
Fig: State diagram after reduction

- **Step 4:** If you observed the state table, the states are represented by using the alphabet. We can not proceed further if we are having alphabets, so, assigning binary numbers to alphabets is called a state assignment.
- To assign binary numbers to the state we have to consider the minimum number of bits.
- The codes must contain n bits for a circuit with m states, where $2^n >= m$. In this case, each state requires $2^3 >=5=>3$ bits to be represented. With three bits, there are eight possible combinations, of which five can be used to represent the states.



State	Assignment 1
	Binary
a	000
b	001
c	010
d	011
e	100

- **Step 5:** Replacing the alphabets with binary numbers



Present state	Next state		Output	
	X=0	X=1	X=0	X=1
000	000	001	0	0
001	010	011	0	0
010	000	011	0	0
011	100	011	0	1
100	000	011	0	1

Thank You