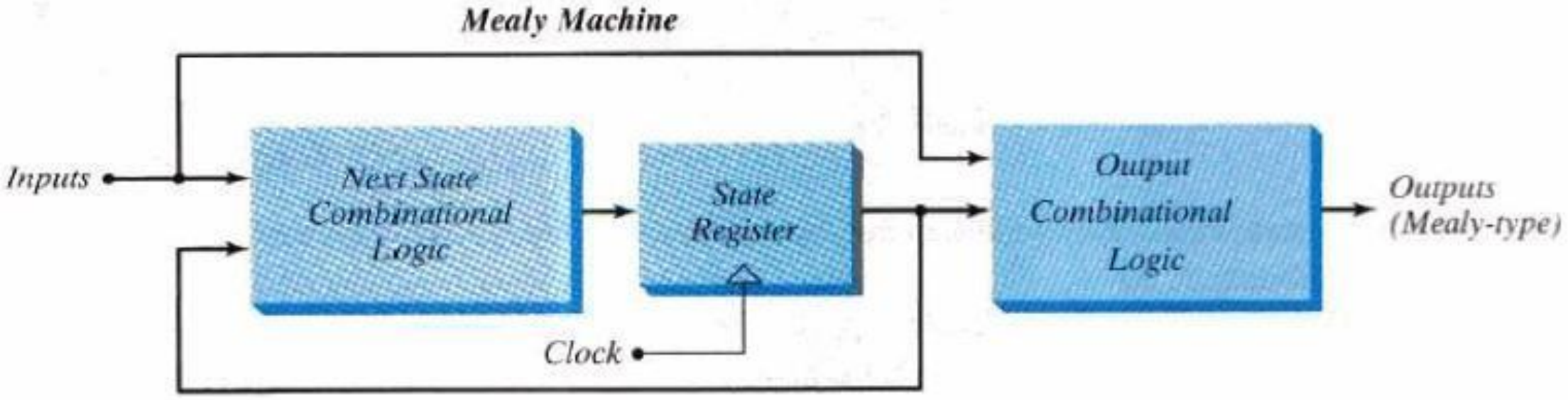
Mealy and Moore State Machines

Finite state Machines

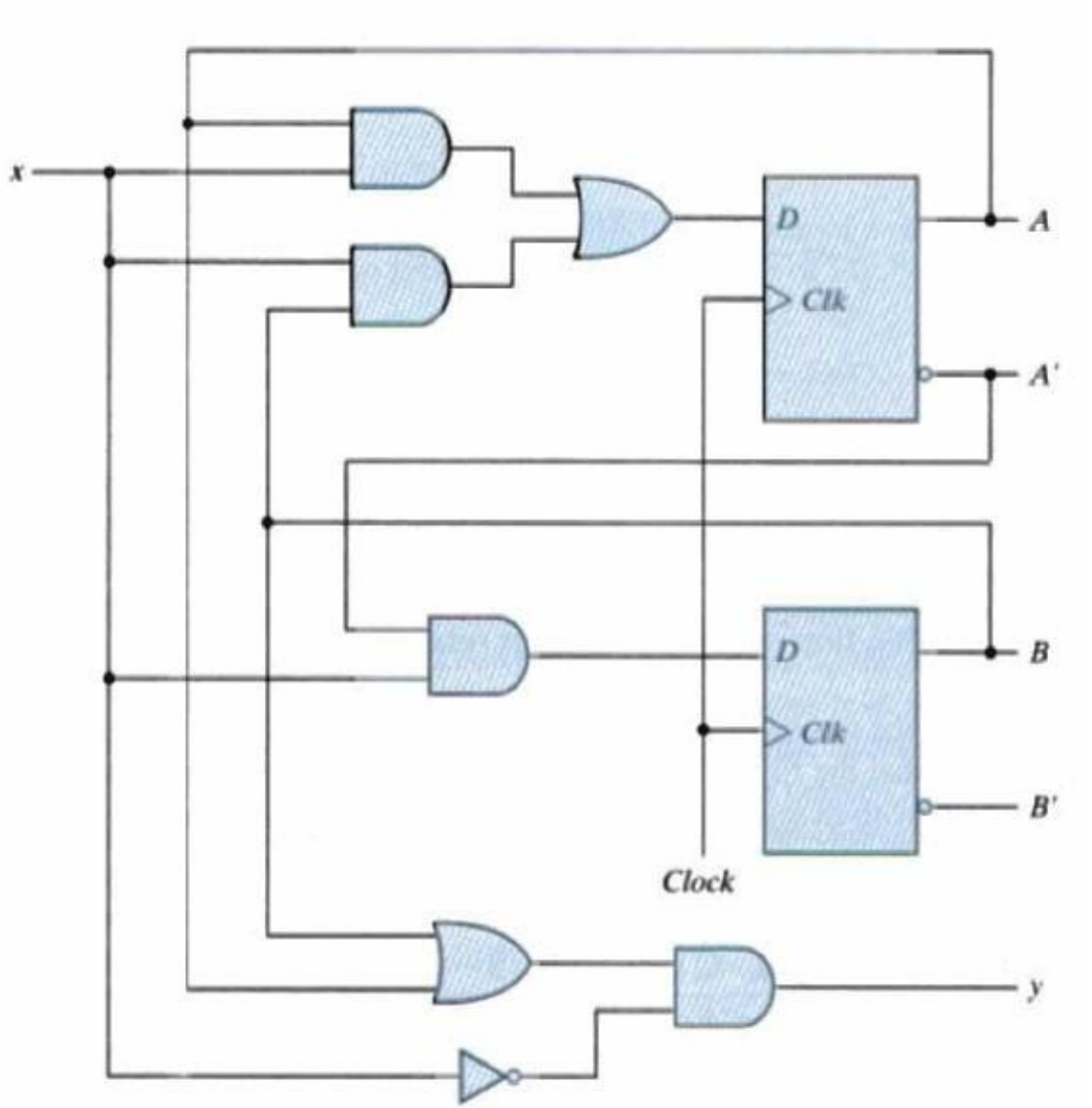
- The most general model of a sequential circuit has inputs, outputs and internal states.
- Two models are developed for representing synchronous sequential circuits and are commonly called as Finite state machines (FSM).
- The Mealy state machine and Moore state machine.
- They differ only in the way the output is generated.
- Mealy ---- Output is a function of both input and present state.
- Moore---- Output is a function of only the present state.

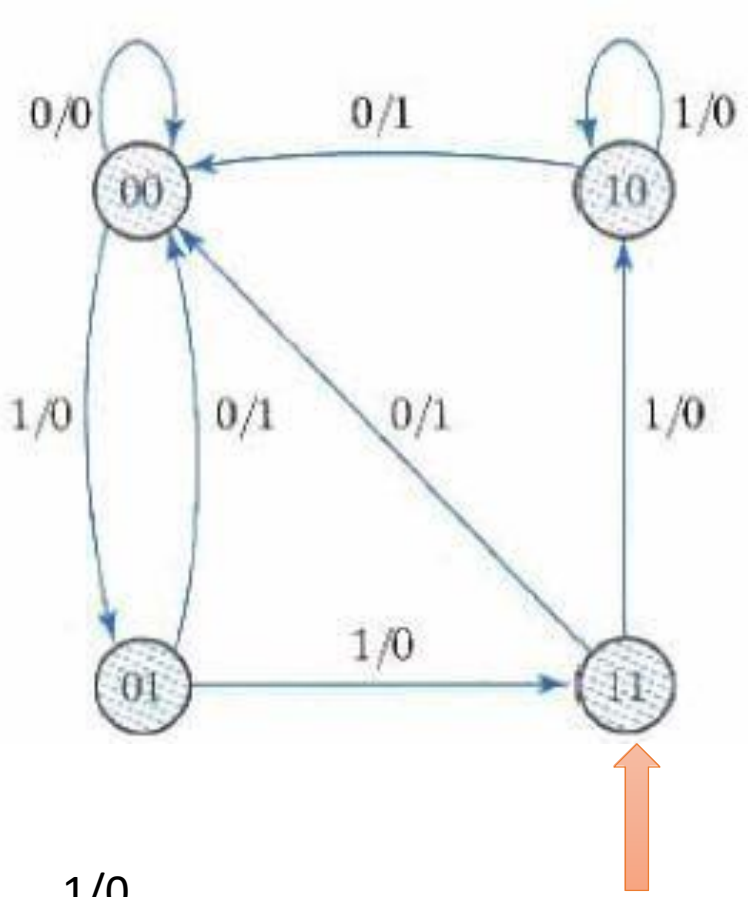
Mealy State Machine

11011



Example for Mealy model





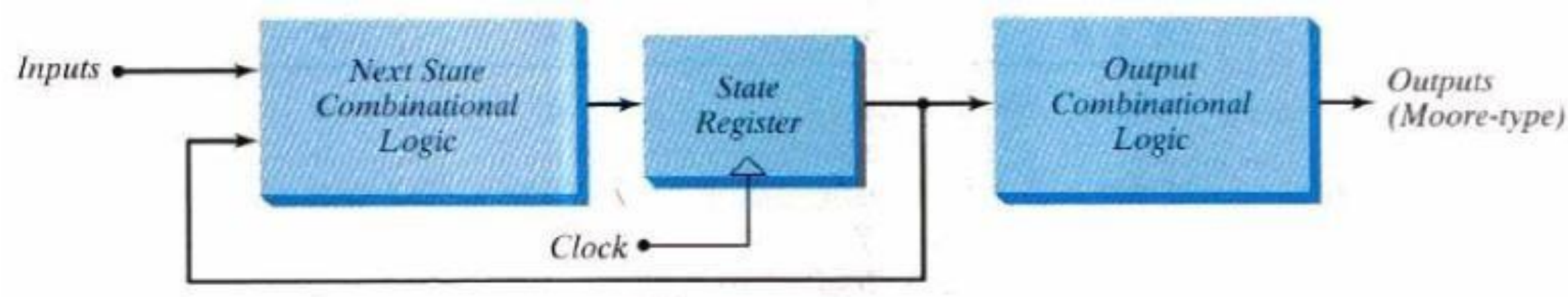
1/0

_____Inp

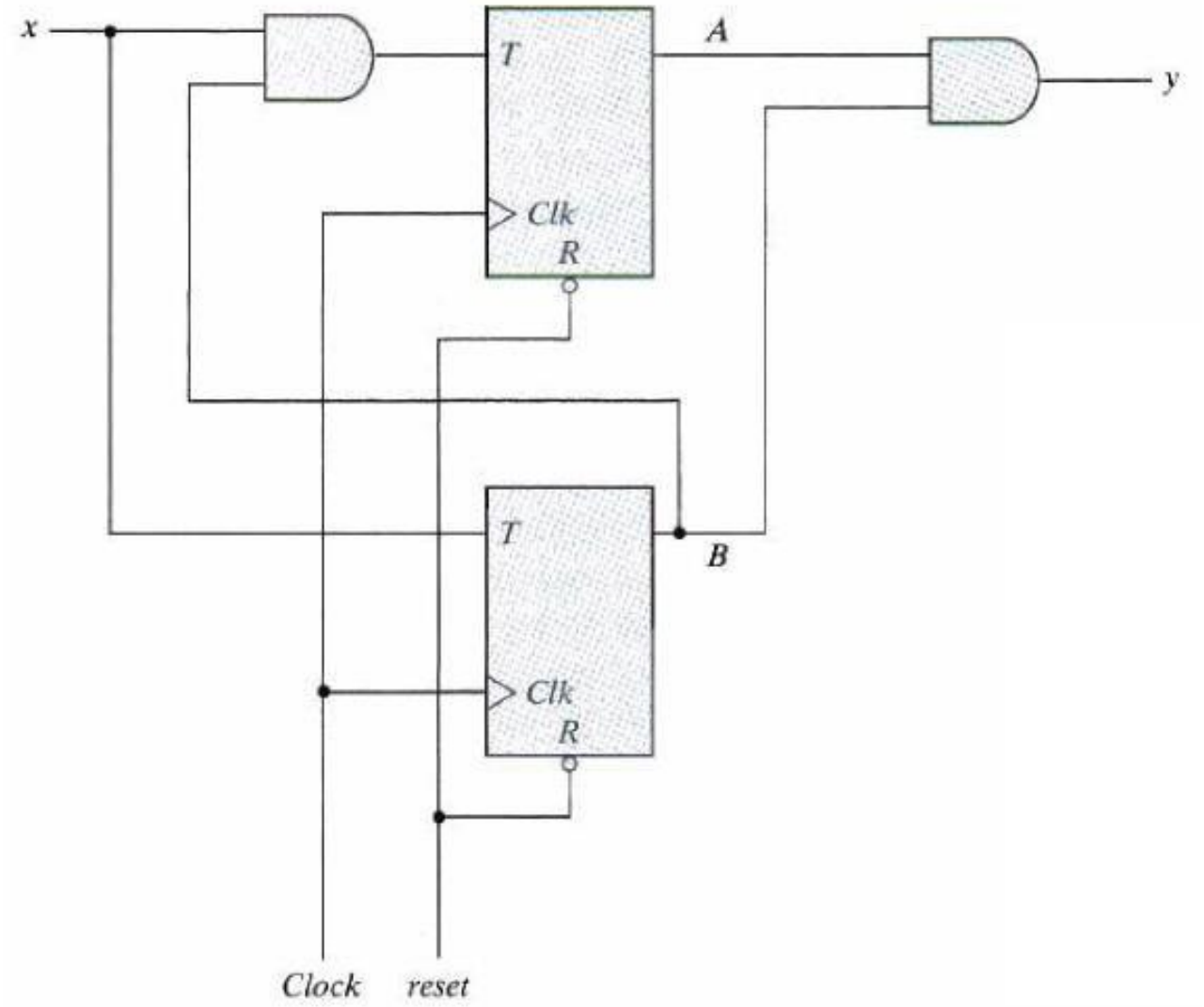
ut/Output

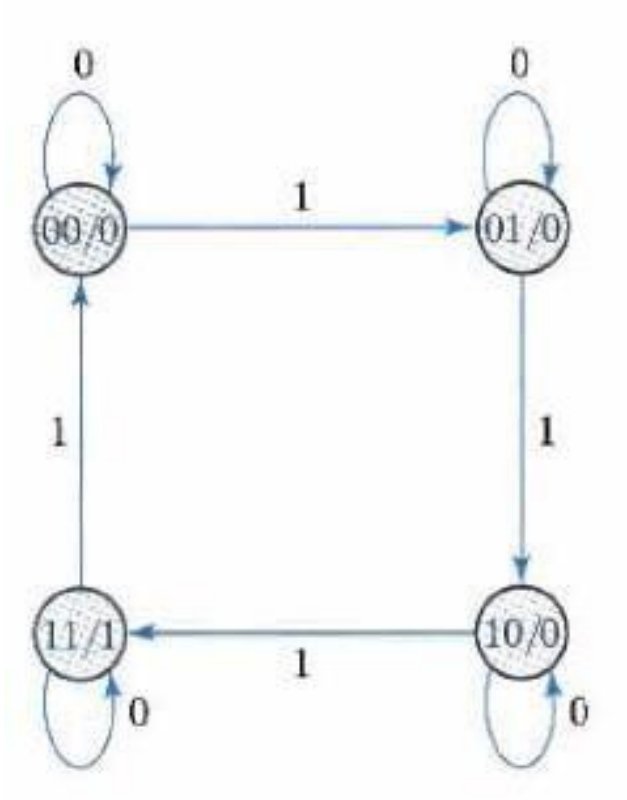
State

Moore State Machine



Example for Moore model

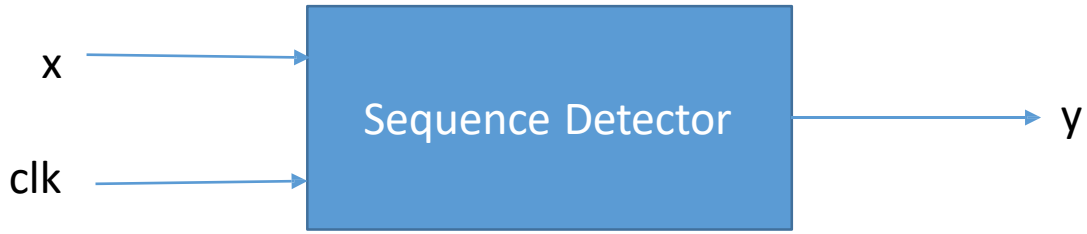




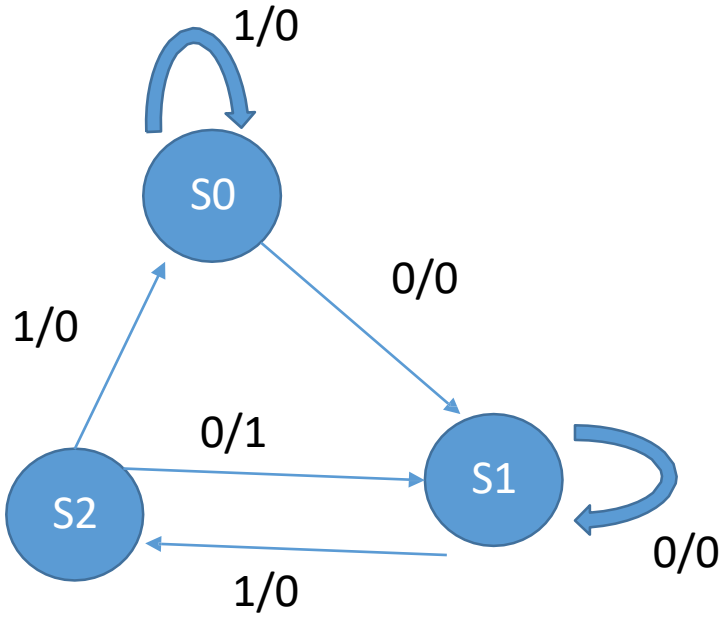
States/ output

Sequence or Pattern Detector

- The stream of bit is given as input when the clock is high and a particular sequence is detected.
- As soon as sequence is detected the output becomes high and then again becomes low.



- Sequence = 0 1 0
- Let x = 0 1 1 0 0 1 0 1 0 0.....
- Y = 0 0 0 0 0 0 1 0 1 0 (Overlapping)
- Y = 0 0 0 0 0 0 1 0 0 0 (Non overlapping)



Sequence Recognizer Procedure

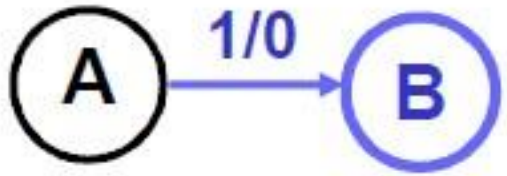
- Begin in an initial state in which NONE of the initial portion of the sequence has occurred (reset state)
- Add a state that recognizes that first symbol has occurred
- Add states that recognize each successive symbol
- The final state represents the input sequence occurence
- Add state transition arcs which specify what happens when a symbol not in the proper sequence has occurred
- Add other arcs which transition to states that represent the input subsequence that has occurred
  - The circuit must recognize the input sequence regardless of where it occurs within the overall sequence

Sequence Recognizer Example

- Example: Recognize the sequence 1101
- Thus, the sequential machine must remember that the first two one's have occurred as it receives another symbol
- Also, the sequence 1101101 contains 1101 as both an initial subsequence and a final subsequence with some overlap, i. e., 1101101 or 1101101
- The 1 in the middle, 1101101, is in both subsequences
- The sequence 1101 must be recognized each time it occurs in the input sequence

Example: Recognize 1101

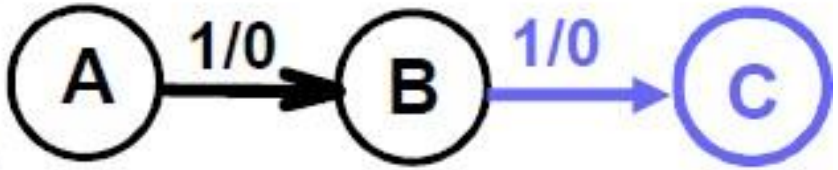
- **Define states for the sequence to be recognized:**
  - Assuming it starts with first symbol
  - Continues through each symbol in the sequence to be recognized
  - Uses output 1 to mean the full sequence has occurred
  - With output 0 otherwise
- **Start in the initial state**



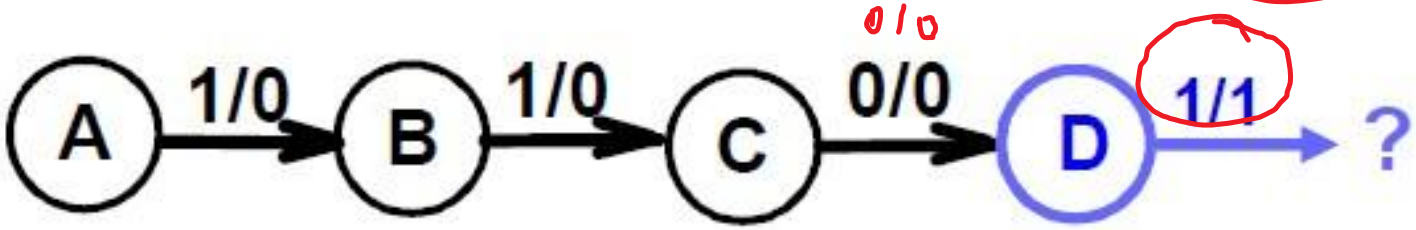
  - State ‘A’ is the initial state
  - Add a state ‘B’ that recognizes the first ‘1’
  - State ‘B’ is the state which represents the fact that the first ‘1’ in the input subsequence has occurred. The output symbol ‘0’ means that the full recognized sequence has not yet occurred

Example: Recognize 1101 (continued)

- After one more '1', we have:
  - C is the state obtained when the input sequence has two '1's.

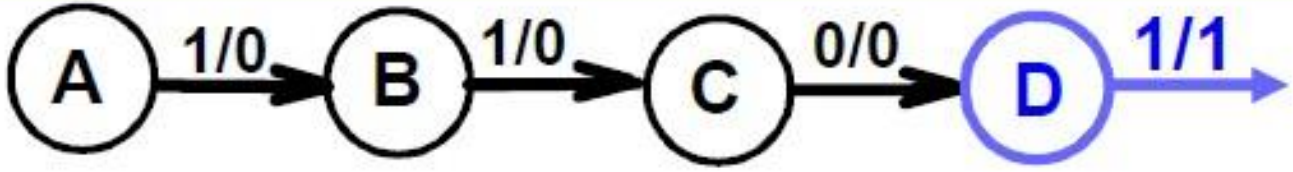


- Finally, after '110' and a '1', we have:

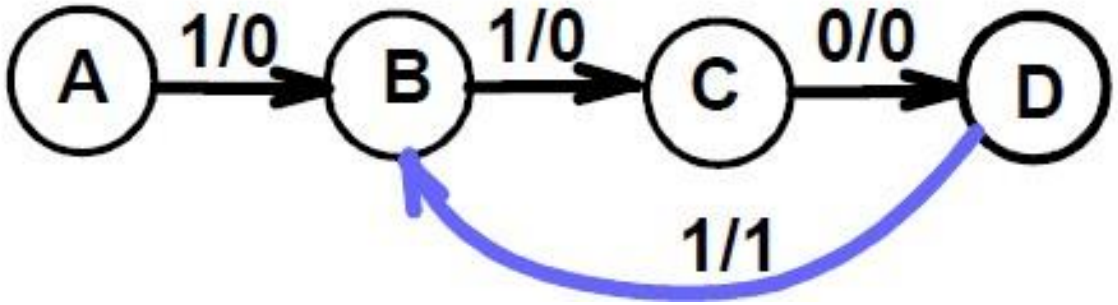


  - Transition arcs are used to denote the output function
  - Output '1' on the arc from D means the sequence is recognized
  - To what state should the arc from state D go? recall 1101101

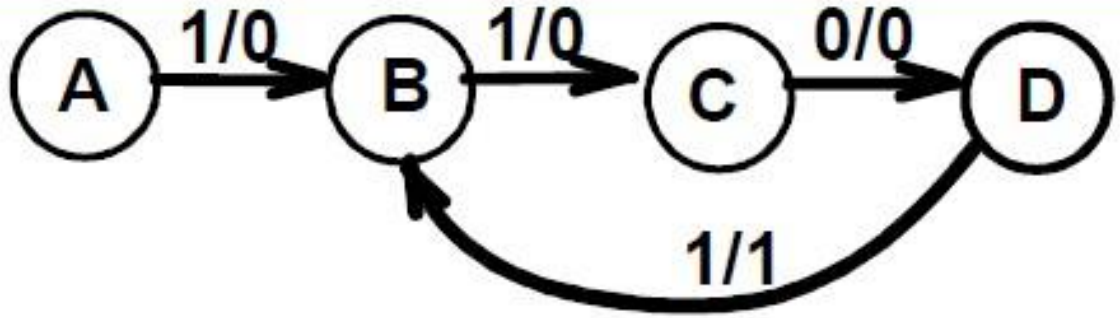
Example: Recognize 1101 (continued)



- Clearly the final '1' in the recognized sequence 1101 is a sub-sequence of 1101. It follows a '0' which is not a sub-sequence of 1101. Thus it should represent *the same state reached from the initial state after a first* '1' *is observed.* We obtain:



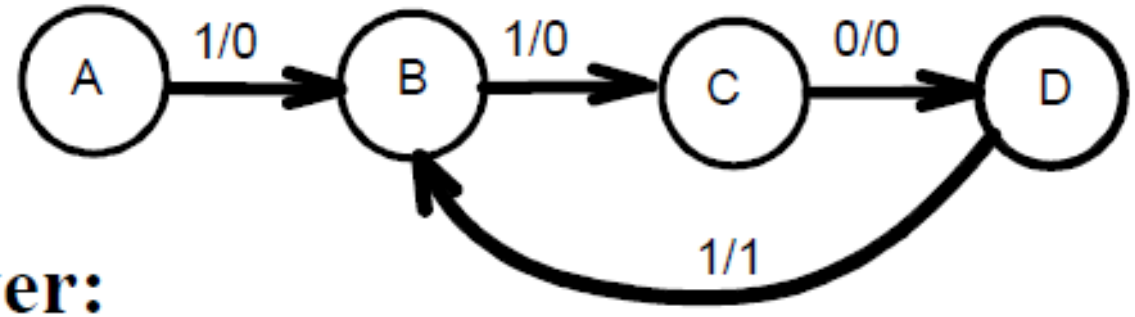
Example: Recognize 1101 (continued)



- The states have the following meanings:
  - A: Start state, no sub-sequence has occurred
  - B: The sub-sequence '1' has occurred
  - C: The sub-sequence '11' has occurred
  - D: The sub-sequence '110' has occurred
  - The 1/1 on the arc from D to B means that the last '1' in 1101 has occurred and thus, the output is '1'

Example: Recognize 1101 (continued)

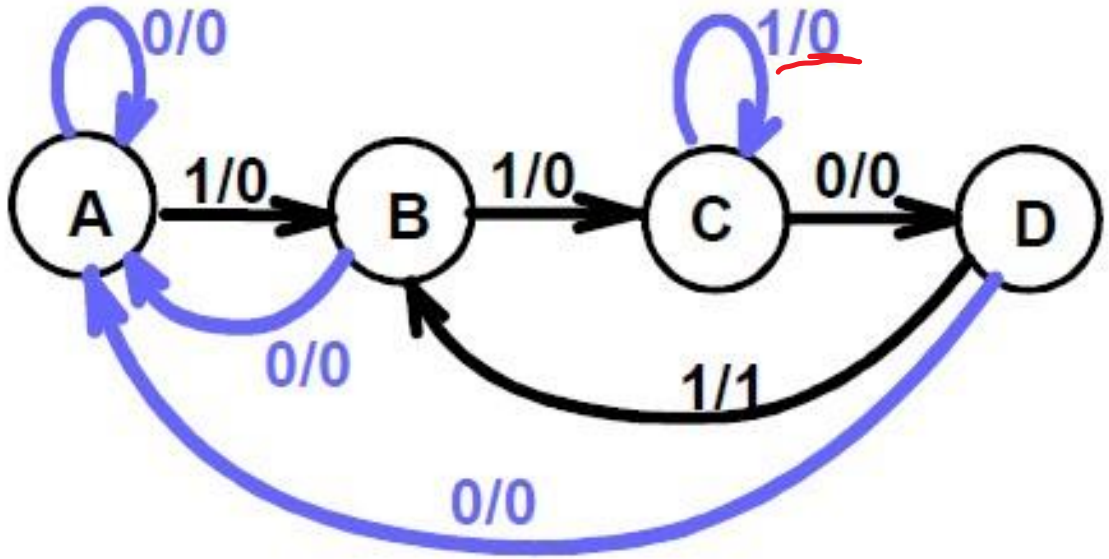
- The other arcs are added to each state for inputs are not yet listed. Which arcs are missing?



- Answer:
  - '0' arc from state A
  - '0' arc from state B
  - '1' arc from state C
  - '0' arc from state D

Example: Recognize 1101 (continued)

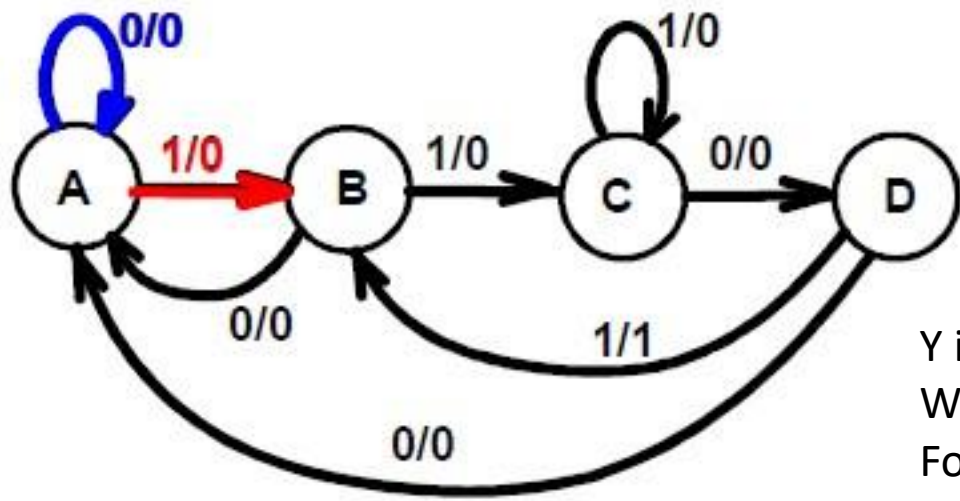
- Add the arcs for missing inputs at any state to make the state diagram complete. We get:



- The '1' arc from state C to itself implies that State C means *two or more 1's have occurred.*

Formulation: Find the State Table

- From the State Diagram, we can fill in the State Table
- There are 4 states, one input, and one output
- We will draw a table with four rows, one for each current state
- From State A, the '0' and '1' input transitions have been filled in along with the outputs



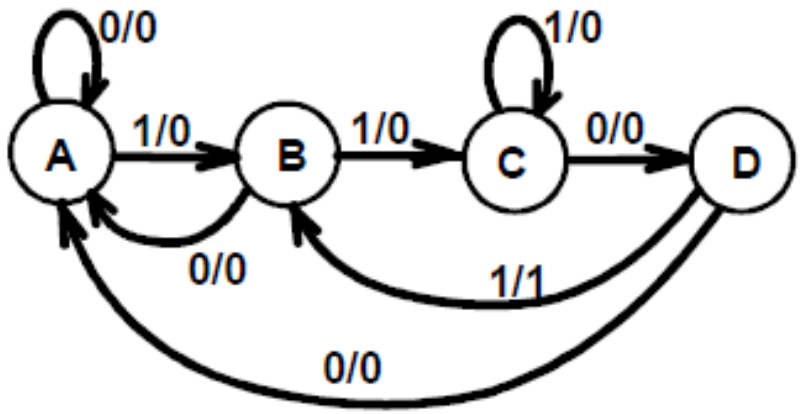
0/0 = x/y

Y is the output
We have to check the o/p
For the value x=0 or x=1

Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
A	A	B	0	0
B				
C				
D				

Formulation: Find State Table

- From the state diagram, we obtain the state table



Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

State Assignment – Example

Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

- What is the minimum number of bits to code 4 states?
  - Answer: 2 bits ($\log_2 4 = 2$). Therefore, 2 flip-flops are required
  - With 2 bits, we can have 4 codes: 00, 01, 10, and 11
- How may assignments of 2-bit codes to the 4 states?
  - Answer: $4 \times 3 \times 2 \times 1 = 24$ possible assignments
- Does code assignment make a difference in cost?
  - Answer: yes, it affects the cost of the combinational logic

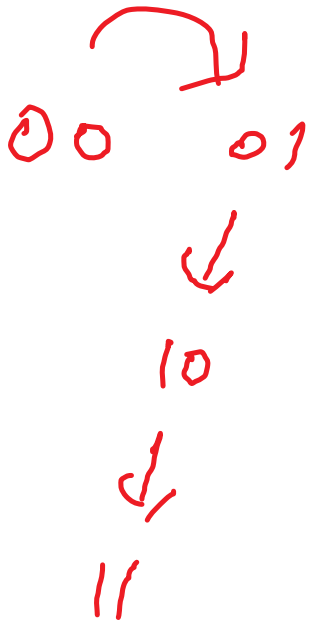
Counting Order State Assignment

- **One possible assignment is Counting Order**

  A = 00, B = 01, C = 10, D = 11

- **The resulting coded state table:**

Present State $Y_1\ Y_2$	Next State $x = 0$	Next State $x = 1$	Output Z $x = 0$	Output Z $x = 1$
A = 0 0	0 0	0 1	0	0
B = 0 1	0 0	1 0	0	0
C = 1 0	1 1	1 0	0	0
D = 1 1	0 0	0 1	0	1



Find Next State and Output K-maps

- Assume D flip-flops & counting order assignment
- Obtain the K-maps for flip-flop inputs D_1, D_2, and output Z

D_1

Y_1Y_2 \ X	0	1
00	0	0
01	0	1
11	0	0
10	1	1

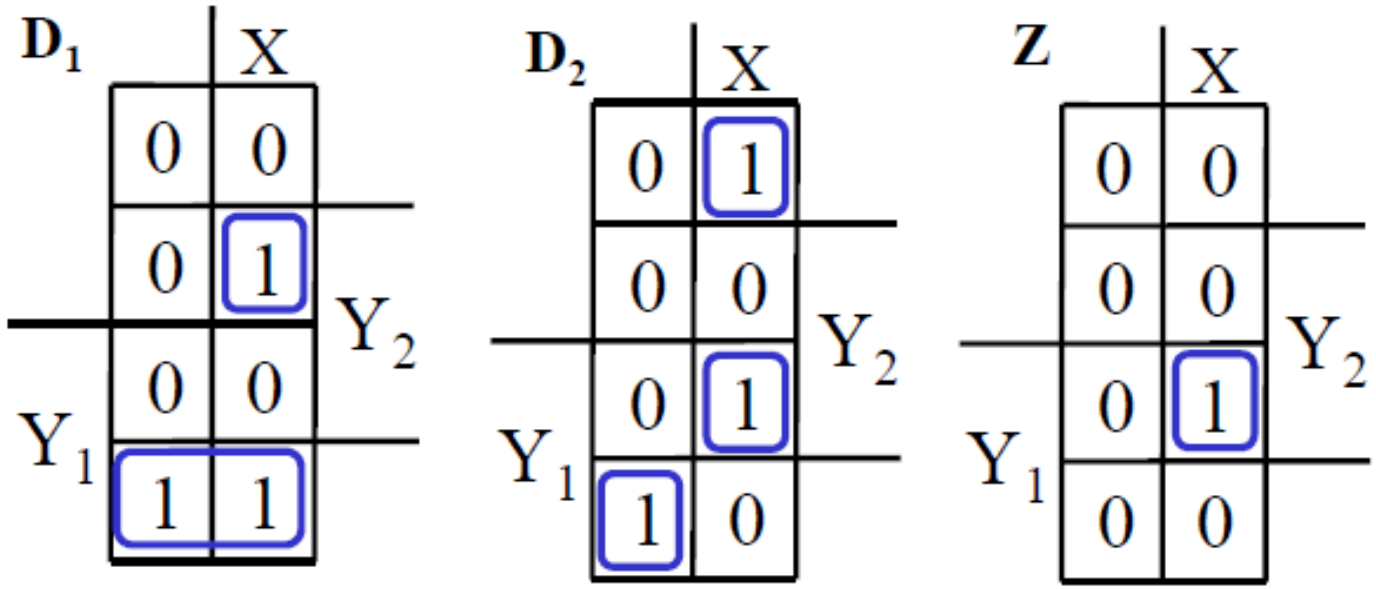
D_2

Y_1Y_2 \ X	0	1
00	0	1
01	0	0
11	0	1
10	1	0

Z

Y_1Y_2 \ X	0	1
00	0	0
01	0	0
11	0	1
10	0	0

Perform two-level optimization



D1

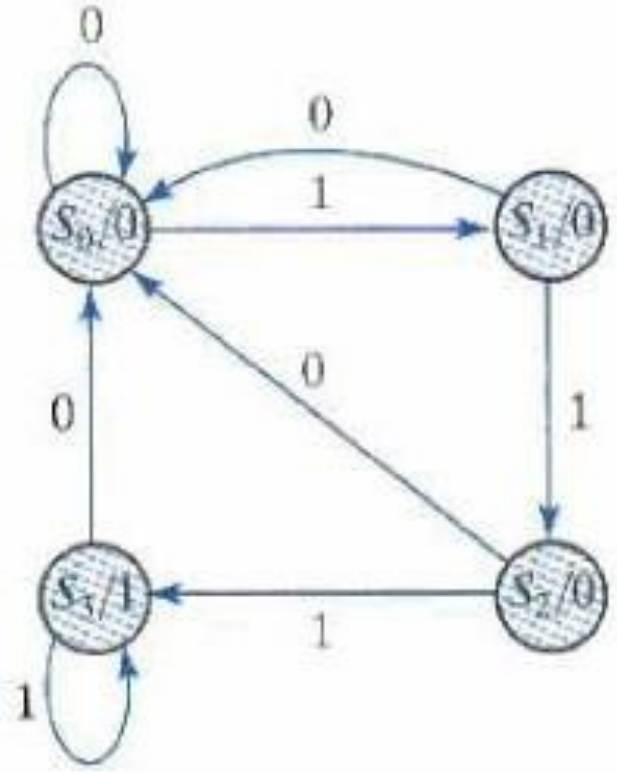
$$D_1 = Y_1\overline{Y}_2 + X\overline{Y}_1Y_2$$

$$D_2 = \overline{X}Y_1\overline{Y}_2 + X\overline{Y}_1\overline{Y}_2 + XY_1Y_2$$

$$Z = XY_1Y_2$$

- Example: Design a sequence detector to detect three or more consecutive 1's in a string of bits coming through an input line.
- X = 0 0 1 1 1 0 1 1 1 1 0 -----
- Y = 0 0 0 0 1 0 0 0 1 1 0 (Overlapping)
- Y = 0 0 0 0 1 0 0 0 1 0 0 (Non Overlapping)

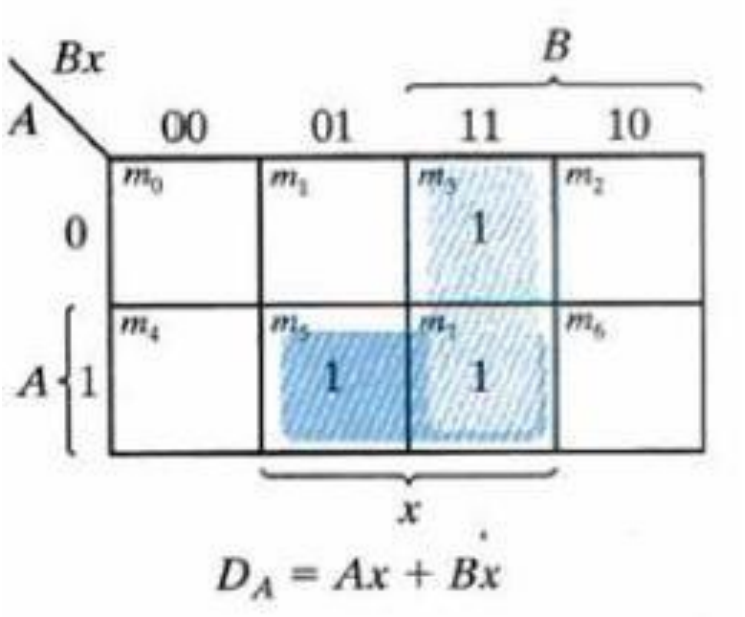
- Step 1: State diagram- Moore model
- S0 = 00
- S1 = 01
- S2 = 10
- S3 = 11

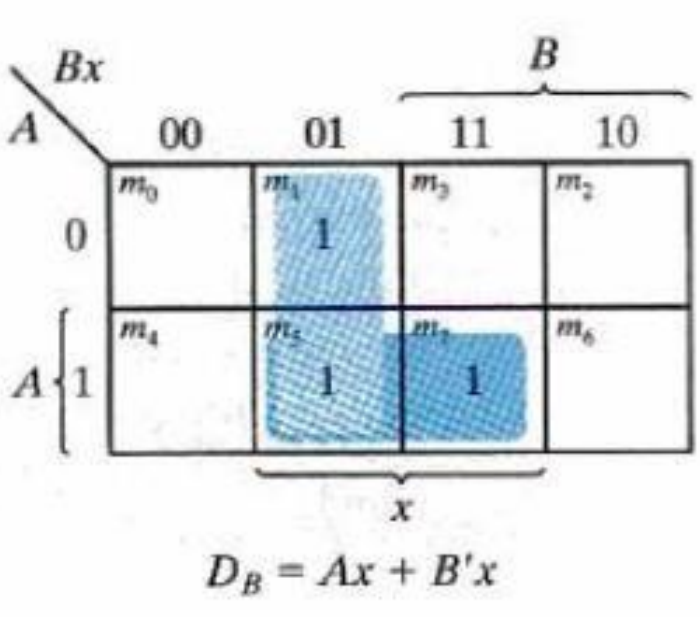


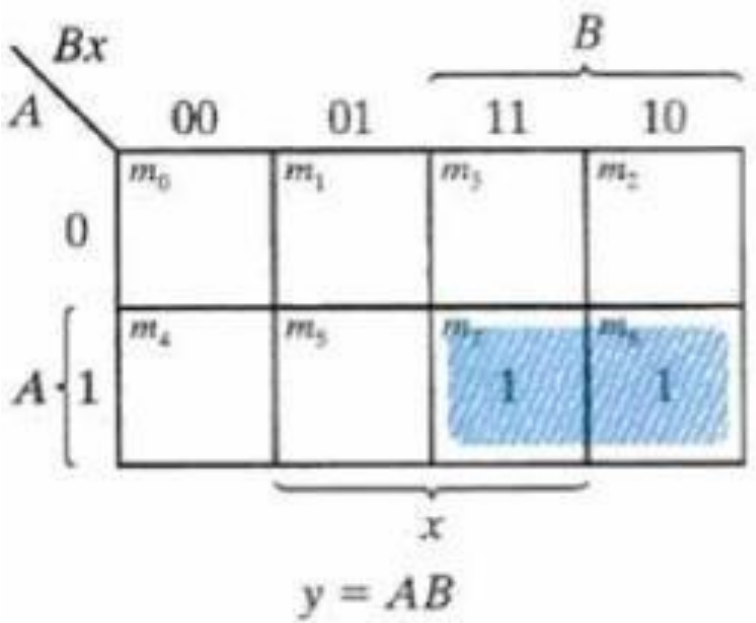
- Step 2: State table

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

- Step 3: Deriving the DFF input and output equations.
- $D_A = A^+$ and $D_B = B^+$







- Step 4: Implementation using DFF

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