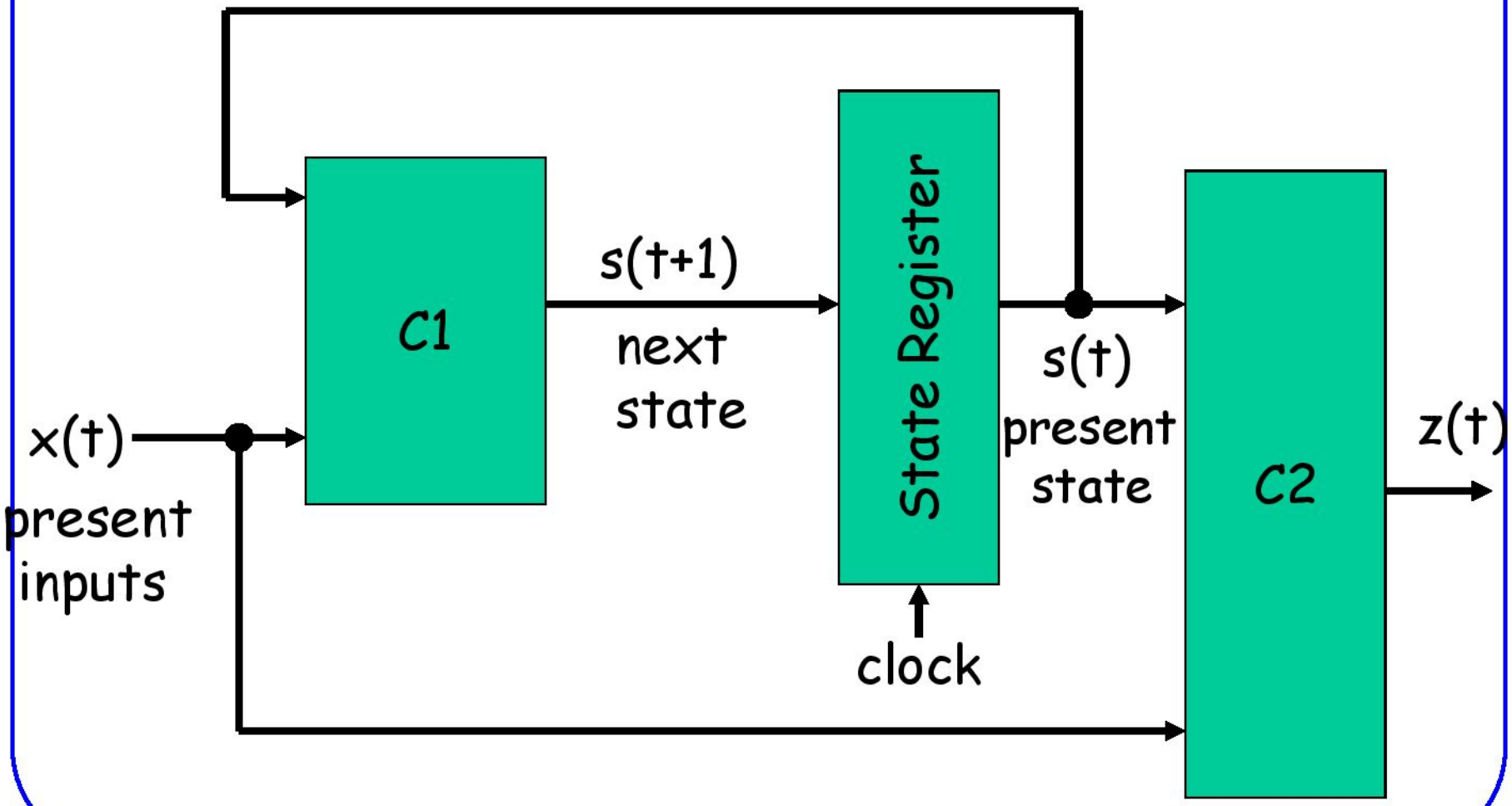
Sequential Circuit Design

Design Procedure

1. Specification
2. Formulation
   - Obtain a state diagram or state table and determine the type (Mealy or Moore)
3. State Assignment and Minimization
   - Assign binary codes to the states and reduce them if possible
4. Flip-Flop Input Equation Determination
   - Select flip-flop types and derive flip-flop equations from next state entries in the table
5. Output Equation Determination
   - Derive output equations from output entries in the table
6. Optimization
   - Optimize the equations
7. Technology Mapping
   - Find circuit from equations and map to flip-flops and gate technology
8. Verification
   - Verify correctness of final design

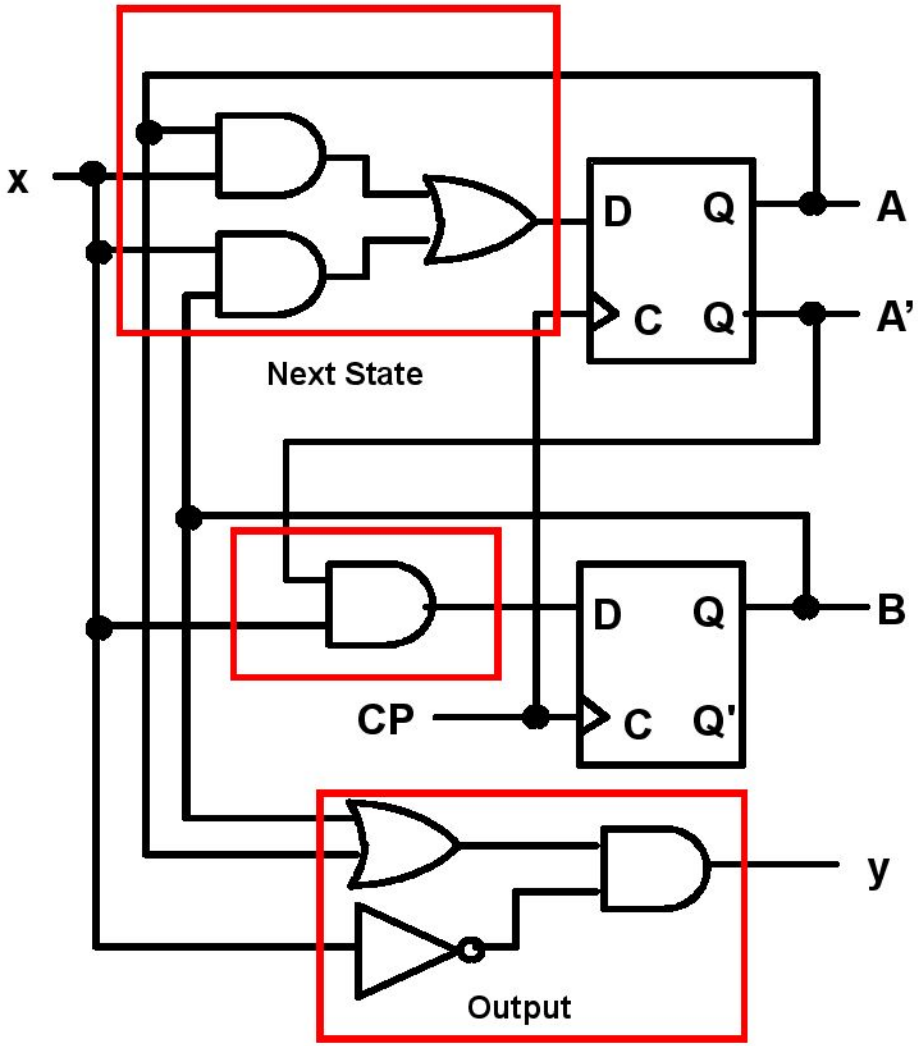
Typical Sequential Circuit

Mealy Machine



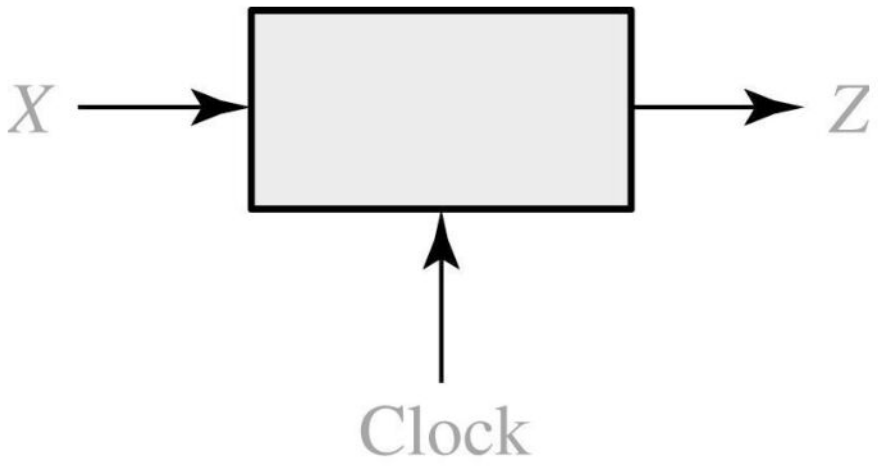
Typical Sequential Circuit

Example



Sequence Detector

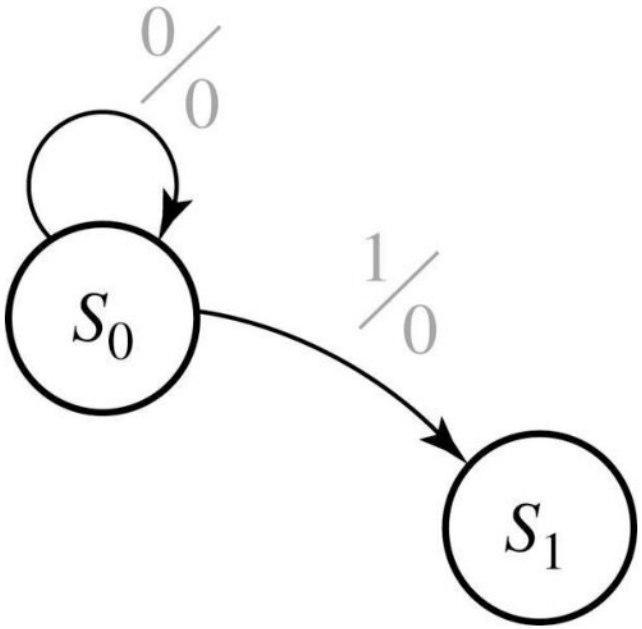
- **101 Sequence Detector**

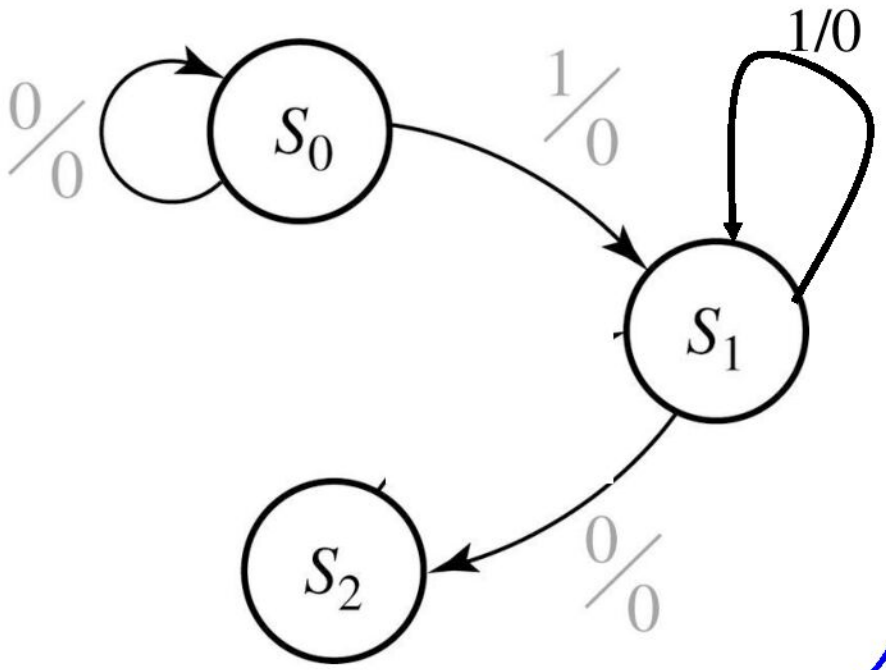


X	**=**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**
Z	**=**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**
(time:		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15)**

Design of 101 Sequence Detector

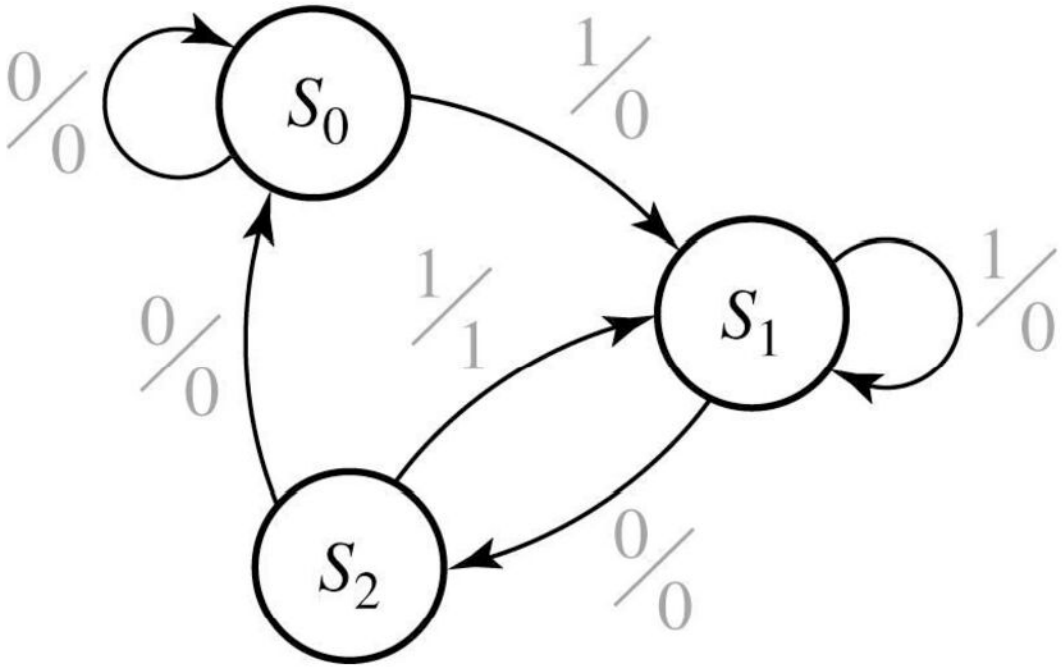
- **State Diagram:**





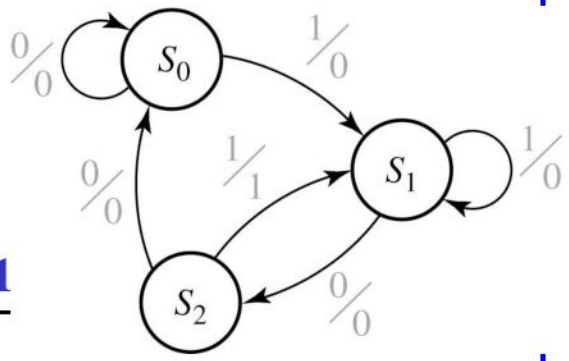
Design of 101 Sequence Detector

- **State Diagram (final):**



Design of 101 Sequence Detector

- **State Table:**



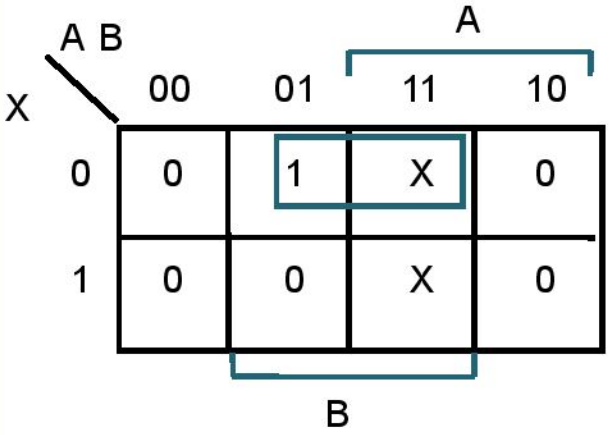
Present state	Next State		Output	
	X = 0	X = 1	X = 0	X =1
S_0	S_0	S_1	0	0
S_1	S_2	S_1	0	0
S_2	S_0	S_1	0	1

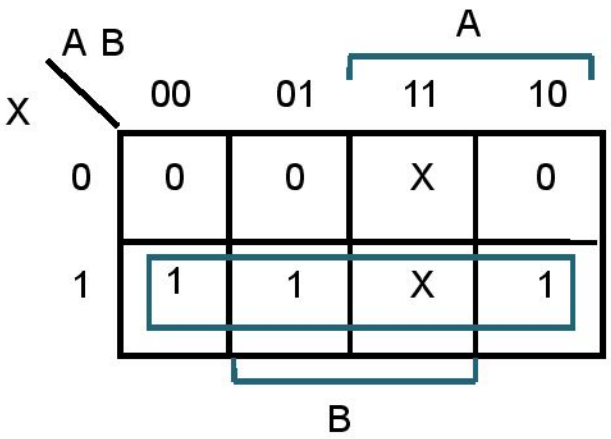
- **State Table with State Assignment:**

	D_A D_B		Z	
	A^+ B^+			
AB	X = 0	X = 1	X = 0	X =1
00	00	01	0	0
01	10	01	0	0
10	00	01	0	1

Design of Sequence Detector

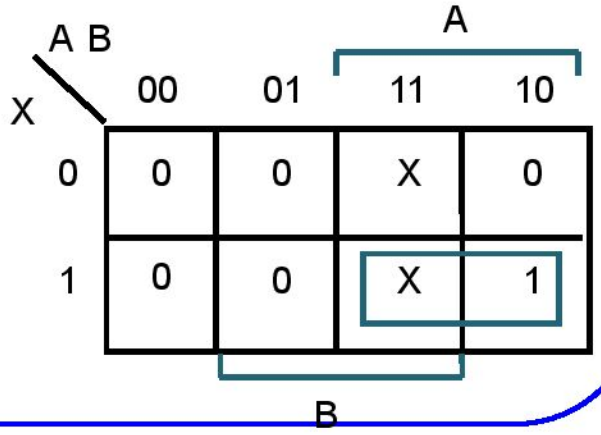
- **Derive Boolean Equations:**





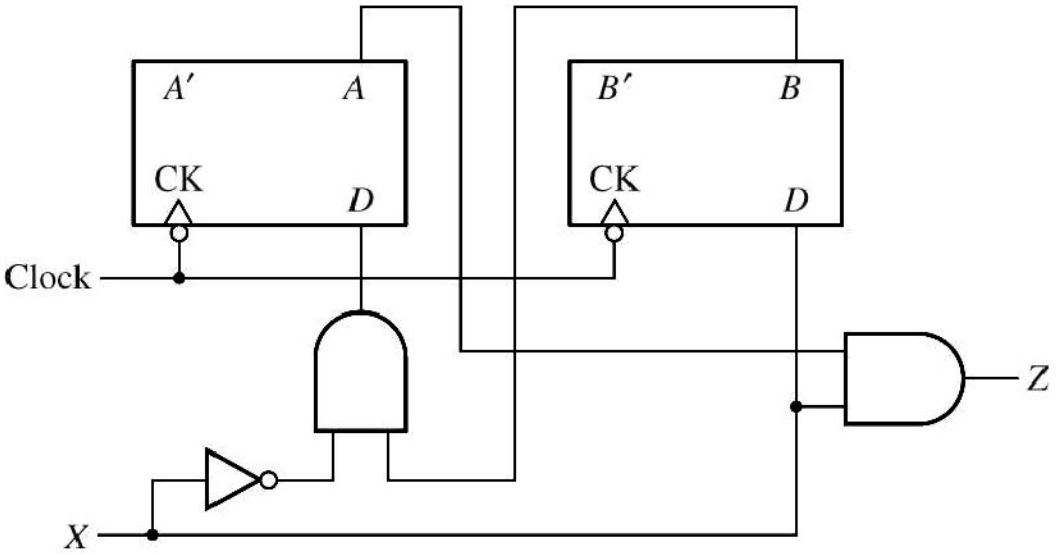
$$D_A = X'.B$$

$$D_B = X$$

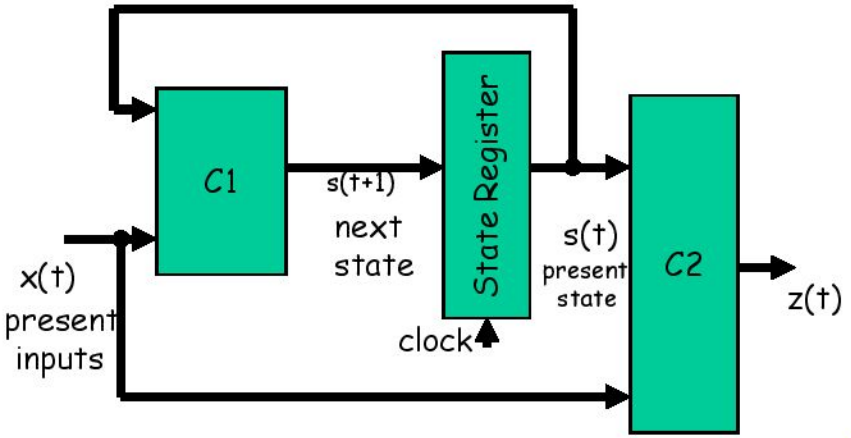


$$Z = X.A$$

Design of Sequence Detector

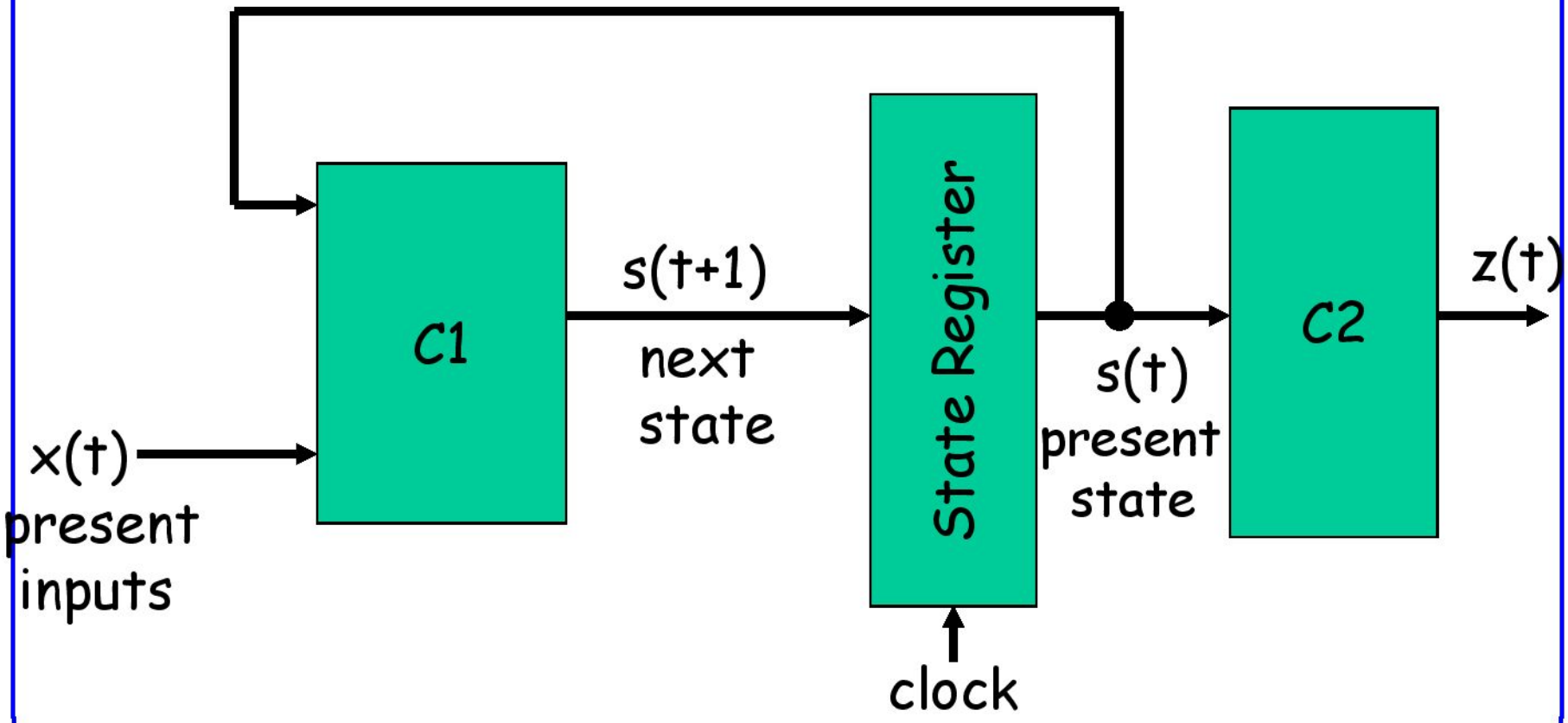


Compare with Typical Mealy Machine:



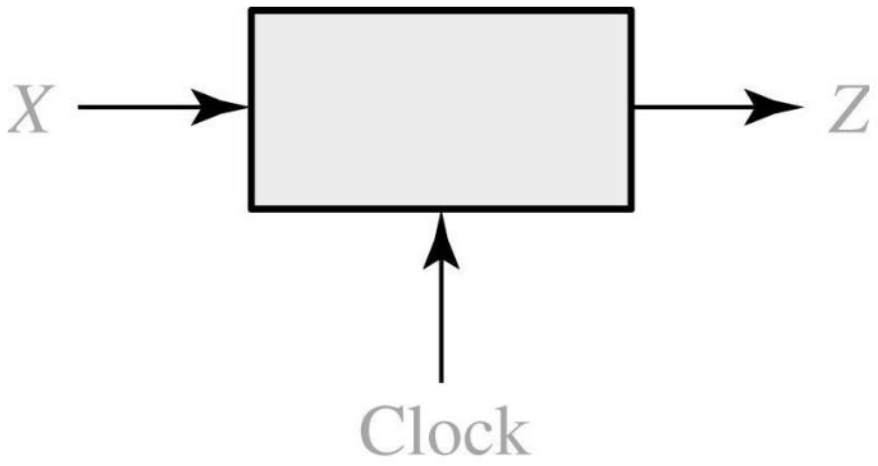
Design of Sequence Detector

- A Moore Sequence Detector:



Sequence Detector

- **101 sequence Detector**



X	**=**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**
Z	**=**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**
(time:		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15)**

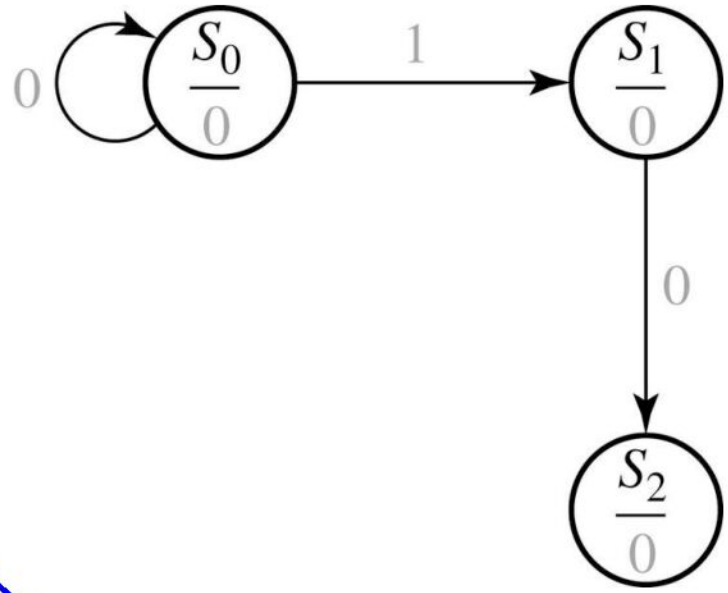
Design of a Sequence Detector

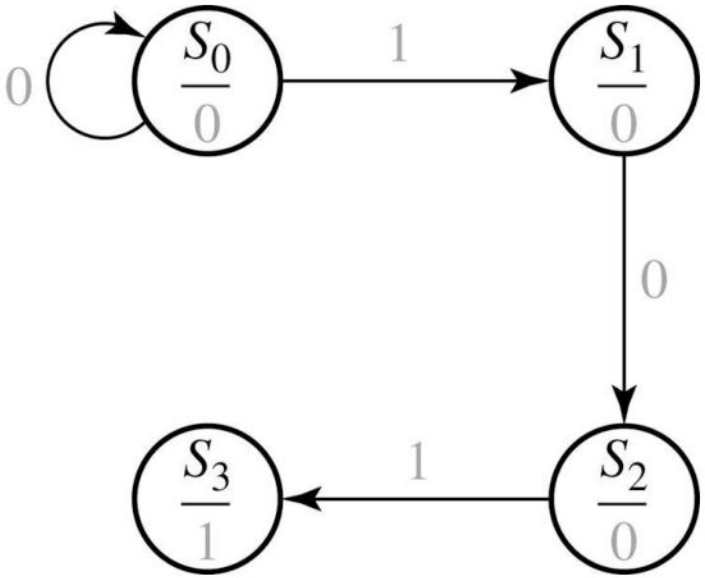
S_0: start

S_1: got 1

S_2: got 10

S_3: got 101





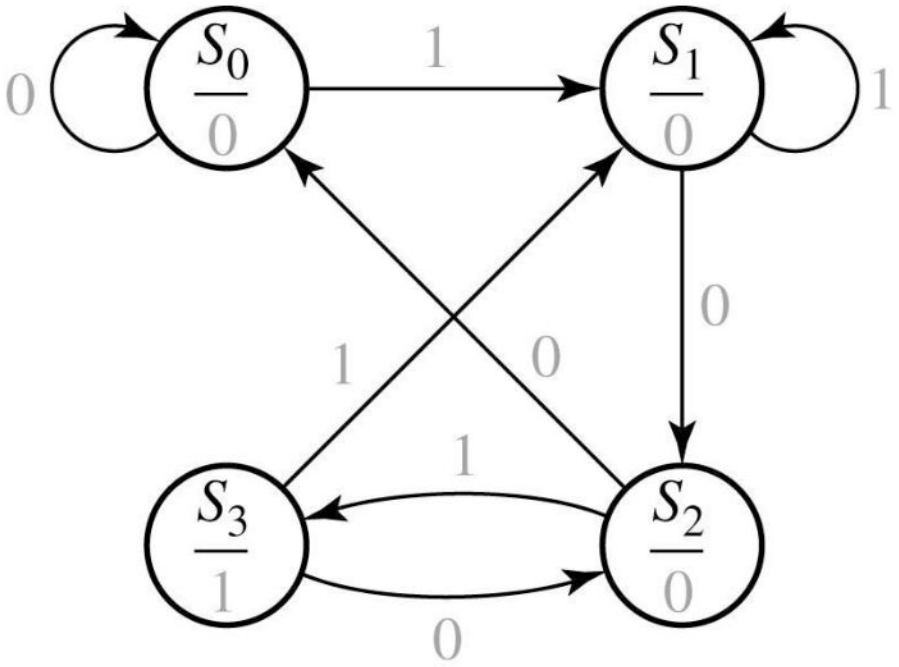
Design of a Sequence Detector

S_0: start

S_1: got 1

S_2: got 10

S_3: got 101



Design of a Sequence Detector

State Table

Present state	Next State X = 0	Next State X = 1	Output (Z)
S_0	S_0	S_1	0
S_1	S_2	S_1	0
S_2	S_0	S_3	0
S_3	S_2	S_1	1

Transition Table with State Assignment

AB	D_A D_B A^+ B^+ X = 0	D_A D_B A^+ B^+ X = 1	Z
00	00	01	0
01	11	01	0
11	00	10	0
10	11	01	1

State Diagram Development

- **To develop a sequence detector state diagram:**
  1. Construct some sample input and output sequences to make sure that you understand the problem statement.
  2. Begin in an initial state in which NONE of the initial portion of the sequence has occurred (typically "reset" state).
  3. Add a state that recognizes that the first symbol has occurred.
  4. Add states that recognize each successive symbol occurring.
  5. Each time you add an arrow to the state graph, determine it can go to one of the previously defined states or whether a new state must be added
  6. The final state represents the input sequence occurrence.
  7. Add state transition arcs which specify what happens when a symbol *not* in the proper sequence has occurred.
  8. Check your state graph for completeness and non-redundant arcs.
  9. When your state graph is complete, test it by applying the input sequences formulated in part1 and making sure the output sequences are correct

State Assignment

- Each of the *m* states must be assigned a unique code.
- Minimum number of bits (or flip-flops) required is *n* such that

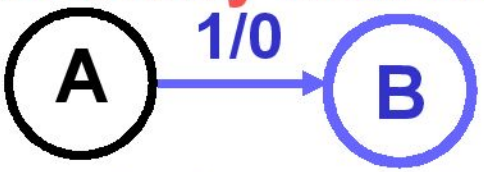
$$n \geq \lceil \log_2 m \rceil$$

- where $\lceil x \rceil$ is the smallest integer $\geq x$.
- There may be 2^n - m unused states.
- There are useful state assignments that use more than the minimum number of bits.

Sequence Detector

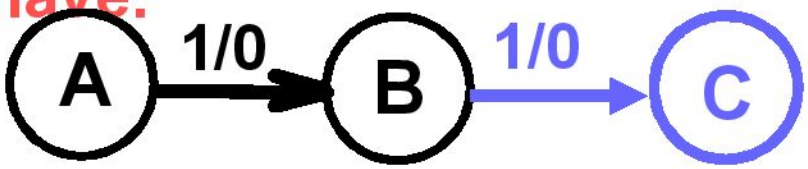
Example: sequence 1101

- **Define states for the sequence to be recognized:**
  - assuming it starts with first symbol,
  - continues through each symbol in the sequence to be recognized, and
  - uses output 1 to mean the full sequence has occurred,
  - with output 0 otherwise.
- **Starting in the initial state (Arbitrarily named "A"):**
  - Add a state that recognizes the first "1."
  - The output symbol "0" means that the full recognized sequence has not yet occurred.

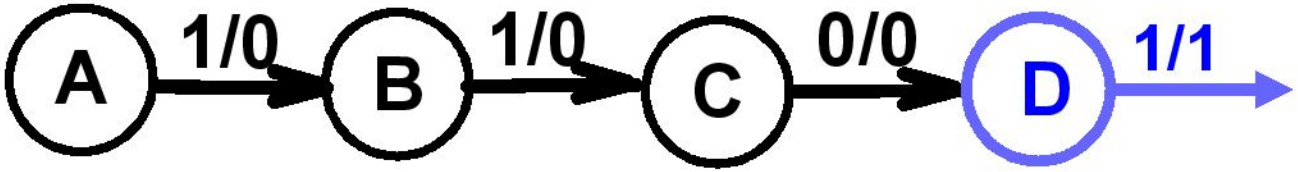


1101 Sequence Detector

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

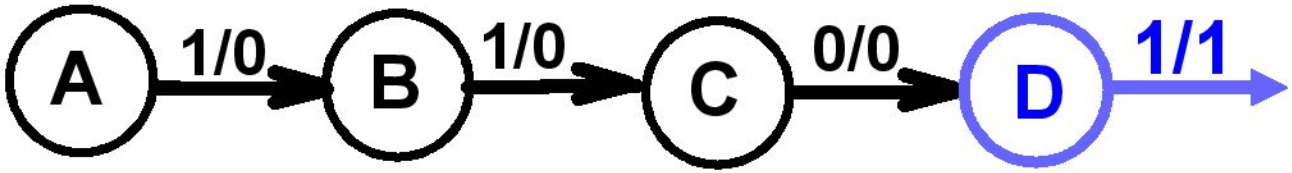


- **Finally, after 110 and a 1, we have:**

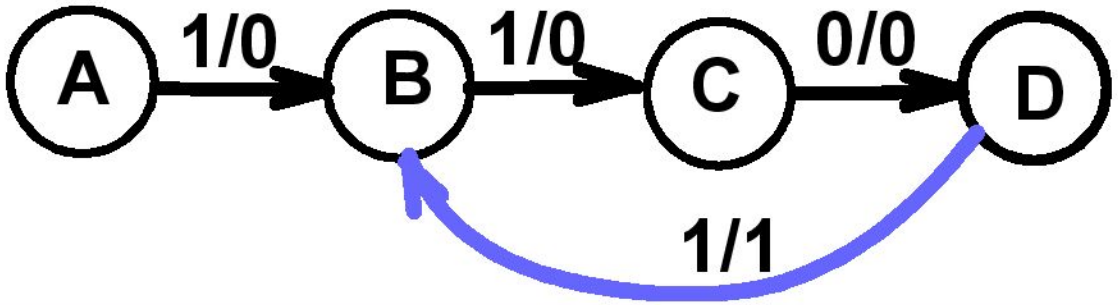


  - Output 1 on the arc from D means the sequence has been recognized
  - To what state should the arc from state D go? Remember: 1101<u>1101</u> ?
  - Note that D is the last state but the output 1 occurs for the input applied in D. This is the case when a *Mealy model* is assumed.

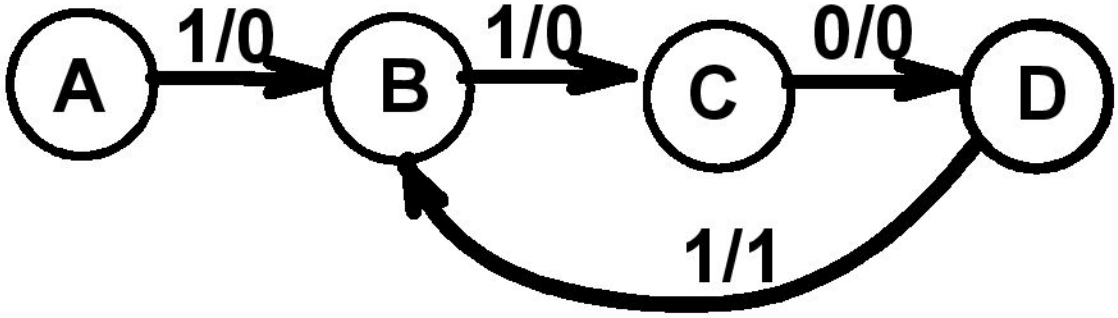
1101 Detector



- 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:



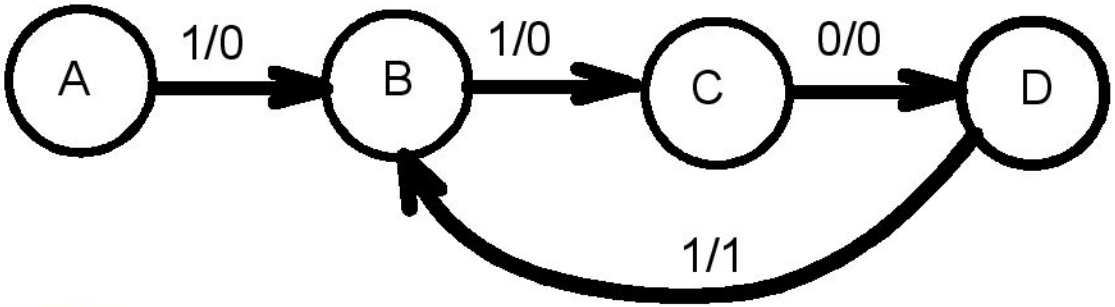
1101 Detector



- **The states have the following abstract meanings:**
  - A: No proper sub-sequence of the 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 has occurred and thus, the sequence is recognized.

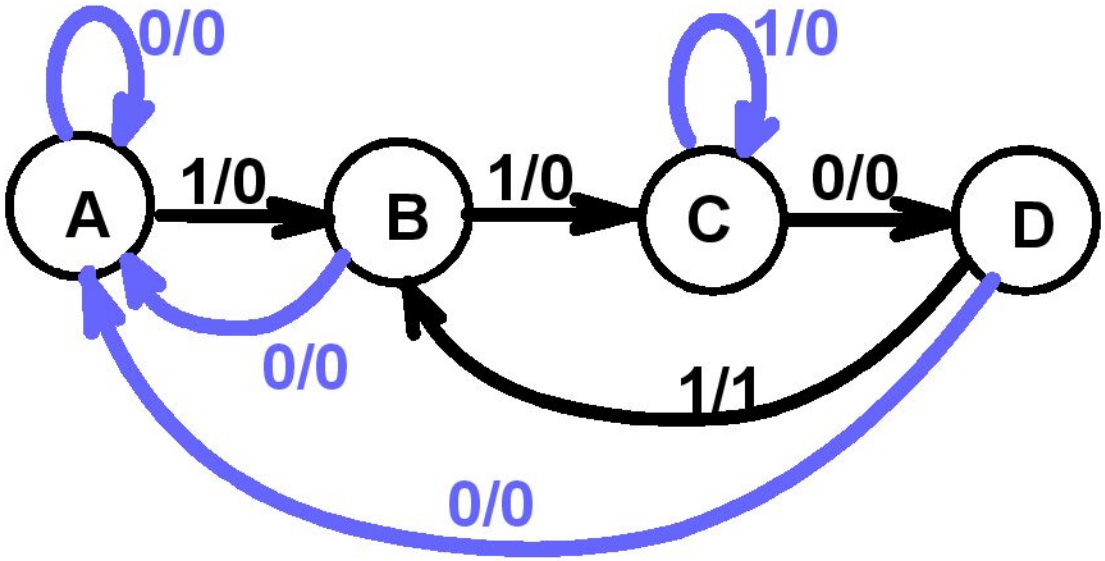
1101 Detector

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



- Answer:
- "0" arc from A
  "0" arc from B
  "1" arc from C
  "0" arc from D.

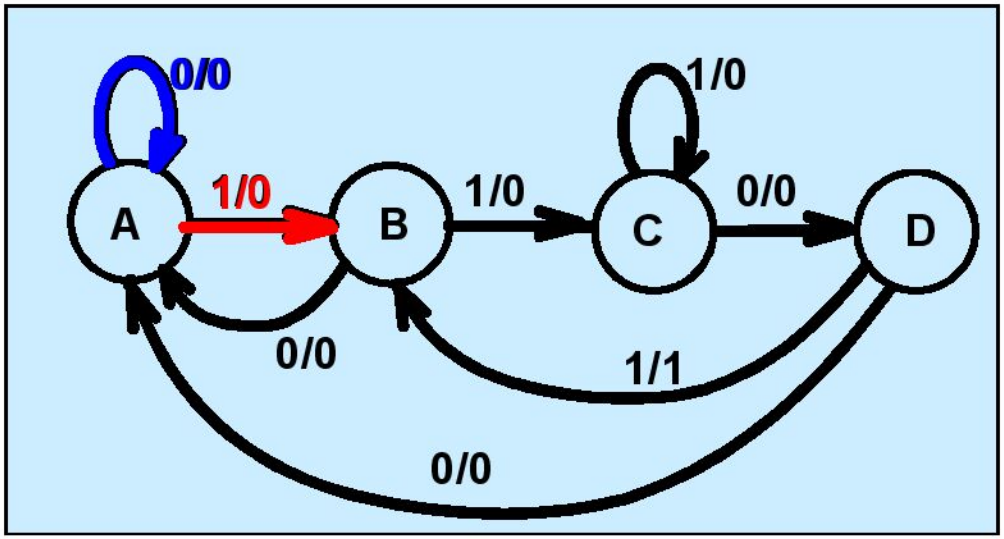
1101 Detector



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

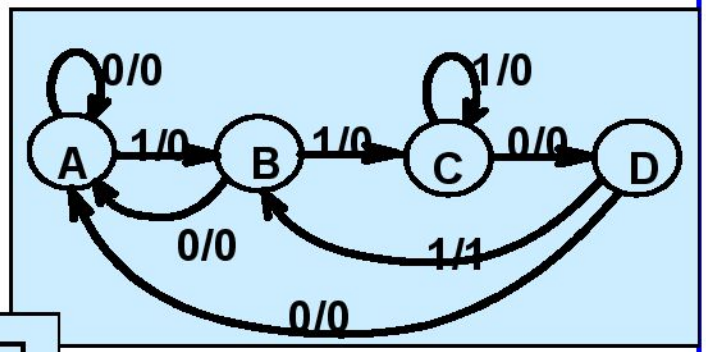
1101 Detector

- From the State Diagram, we can fill in the State Table.
- There are 4 states, one input, and one output. We will choose the form 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.



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

1101 Detector: 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

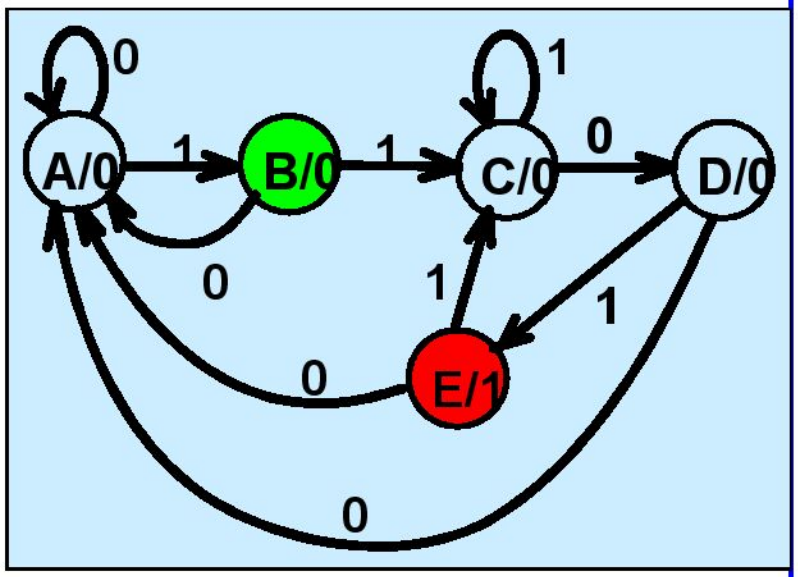
- What would the state diagram and state table look like for the Moore model?

Sequence Detector: Moore Model

- For the Moore Model, outputs are associated with states.
- We need to add a state "E" with output value 1 for the final 1 in the recognized input sequence.
  - This new state E, though similar to B, would generate an output of 1 and thus be different from B.
- The Moore model for a sequence recognizer usually has *more states* than the Mealy model.

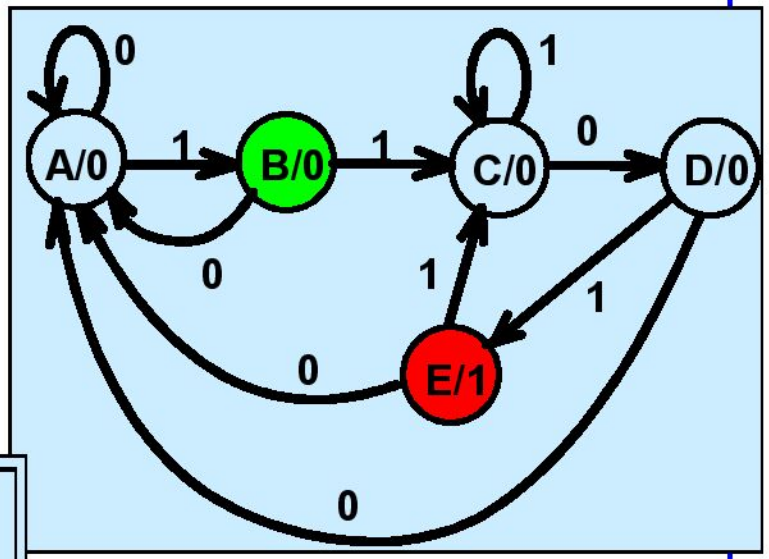
Sequence Detector: Moore Model

- We mark outputs on states for Moore model
- Arcs now show only state transitions
- Add a new state E to produce the output 1
- E produces the same behavior in the future as state B. But it gives a different output at the present time.



Sequence Detector: Moore Model

- The state table is shown below
- More state in the Moore model: "Moore is More."



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

State Assignment: Example 1

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

- How may assignments of codes with a minimum number of bits?
  - Two: A = 0, B = 1 or A = 1, B = 0
- Does it make a difference?
  - Only in variable inversion, so small, if any.

State Assignment: Example 2

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

- How may assignments of codes with a minimum number of bits?
  - $4 \times 3 \times 2 \times 1 = 24$
- Does code assignment make a difference in cost?
- Probably yes!

State Assignment: Example 2

- Assignment 1:
  - A = 0 0, B = 0 1, C = 1 0, D = 1 1
- The resulting coded state table:

Present State	Next State x = 0	Next State x = 1	Output x = 0	Output x = 1
0 0	0 0	0 1	0	0
0 1	0 0	1 0	0	0
1 0	1 1	1 0	0	0
1 1	0 0	0 1	0	1

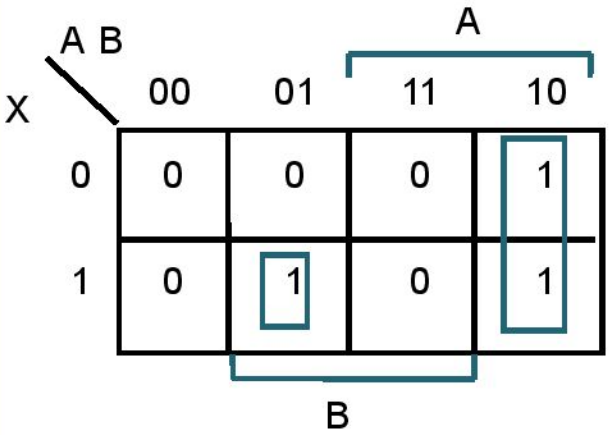
State Assignment: Example 2

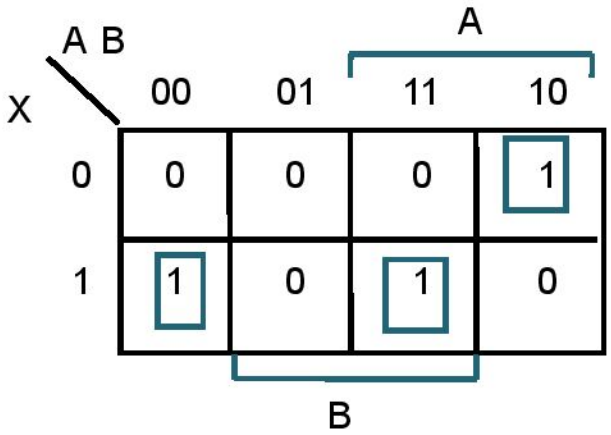
- Assignment 2:
  - A = 0 0, B = 0 1, C = 1 1, D = 1 0
- The resulting coded state table:

Present State	Next State x = 0	Next State x = 1	Output x = 0	Output x = 1
0 0	0 0	0 1	0	0
0 1	0 0	1 1	0	0
1 1	1 0	1 1	0	0
1 0	0 0	0 1	0	1

Flip-Flop Input and Output Equations: Example 2 (version 1)

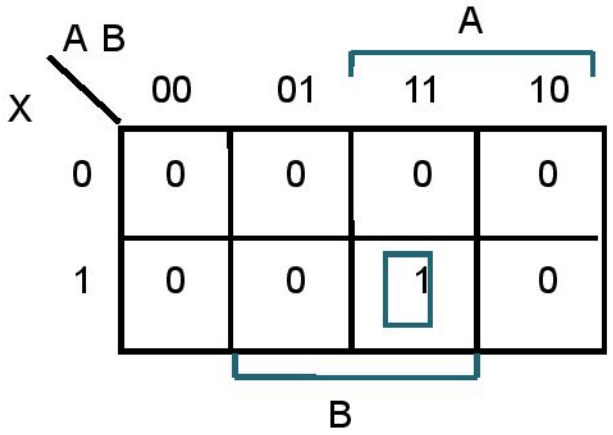
- Assume D flip-flops
- Interchange the bottom two rows of the state table, to obtain K-maps for D_A, D_B, and Z:





$$D_A = A.B' + X.A'.B \qquad D_B = X'.A.B' + X.A'.B' + X.A.B$$

Flip-Flop Input and Output Equations: Example 2 (version 1)

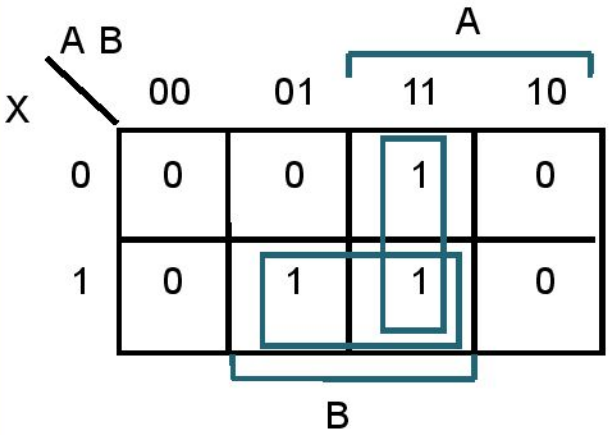


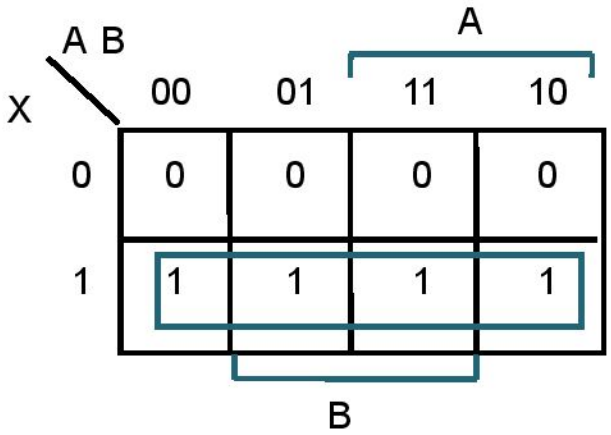
$Z = A.B.X$

Gate Input Cost = 22

Flip-Flop Input and Output Equations: Example 2 (version 2)

- Assume D flip-flops
- Interchange the bottom two rows of the state table, to obtain K-maps for D_A, D_B, and Z:

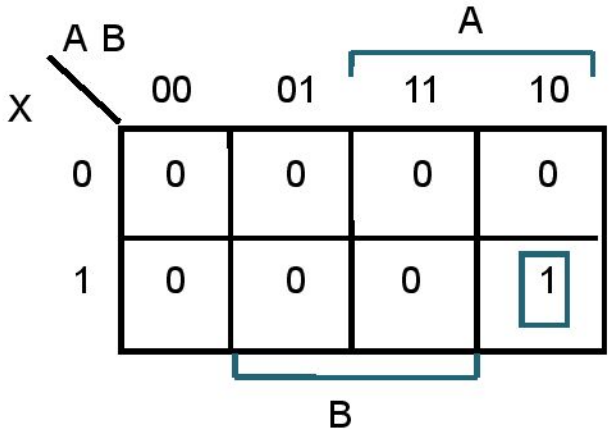




$$D_A = A.B + X.B$$

$$D_B = X$$

Flip-Flop Input and Output Equations: Example 2 (version 2)



$$Z = A.B'.X$$

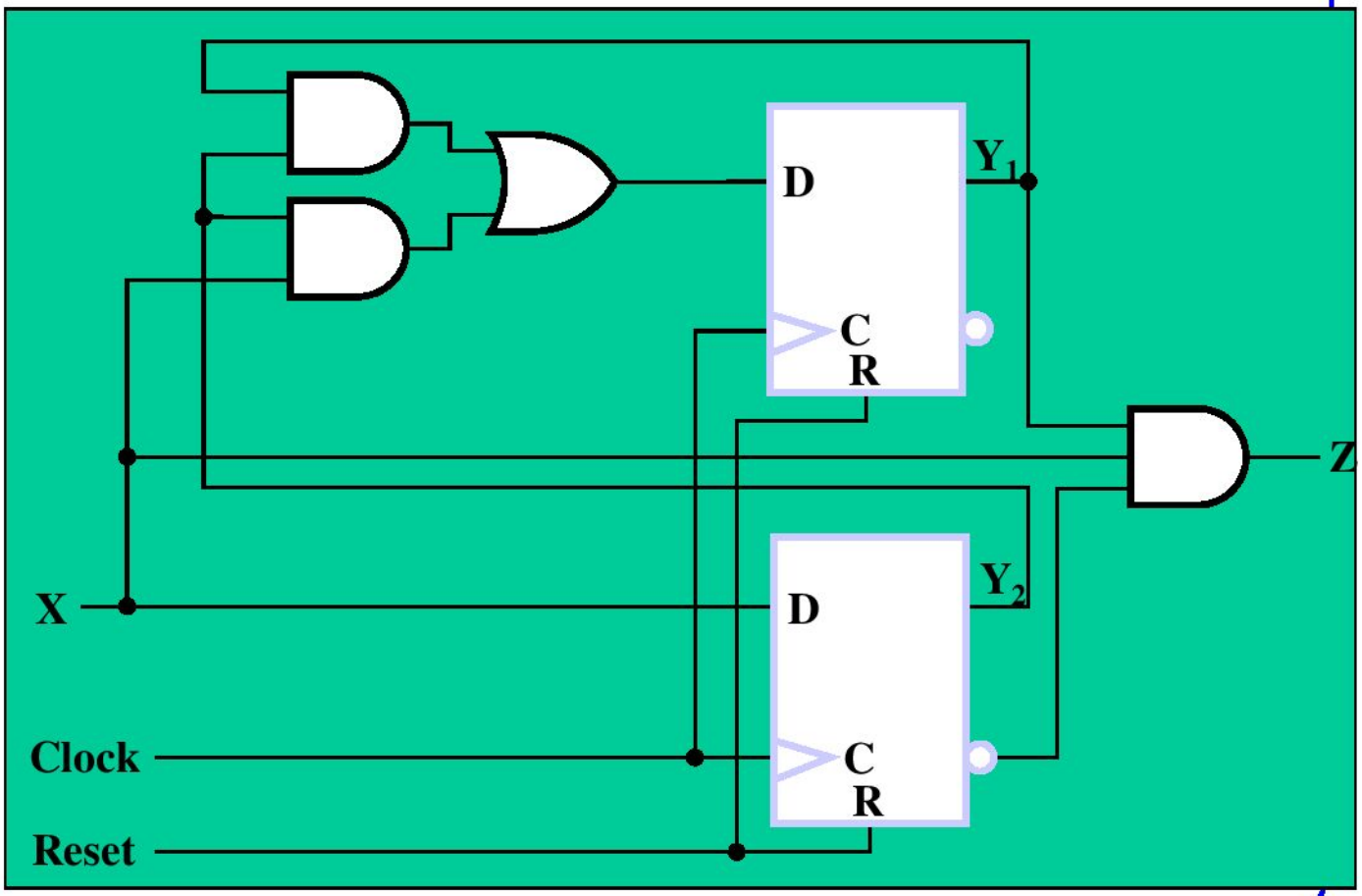
Gate Input Cost = 9

Select this state assignment!

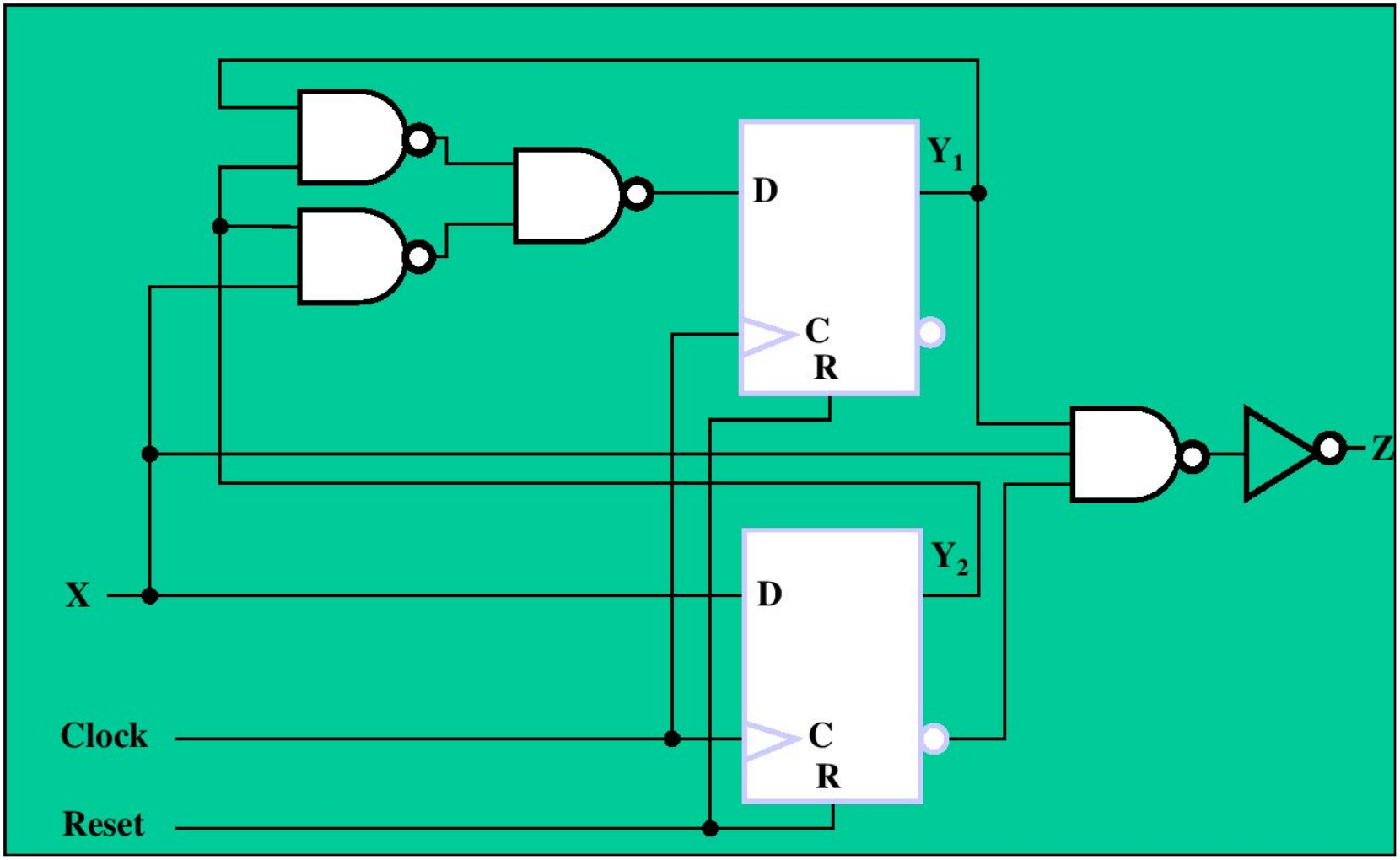
Implementation

- **Library:**
  - D Flip-flops with Reset (not inverted)
  - NAND gates with up to 4 inputs
  - Inverters

- **Initial Circuit:**



Technology Mapping

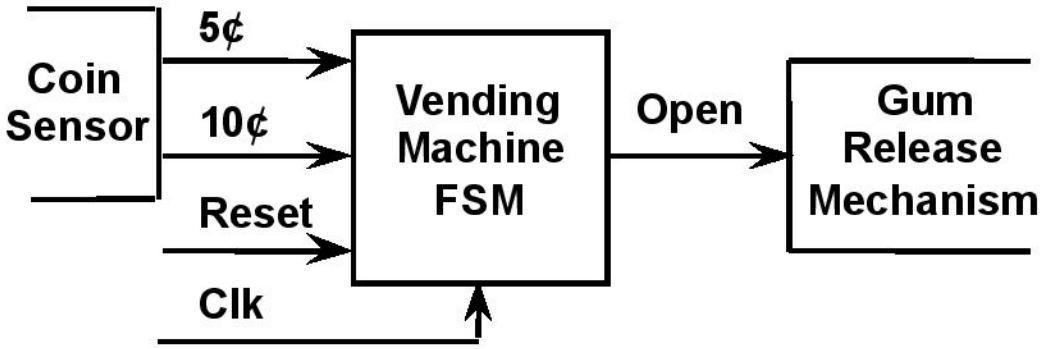


Design Example: Vending Machine

- **General Machine Concept:**
  - Deliver a package of gum after 15 cents deposited
  - Single coin slot for dimes (10¢) and nickels (5¢)
  - No change

Example: Vending Machine

- **Step 1: Understand the problem:**
  - Draw a block diagram

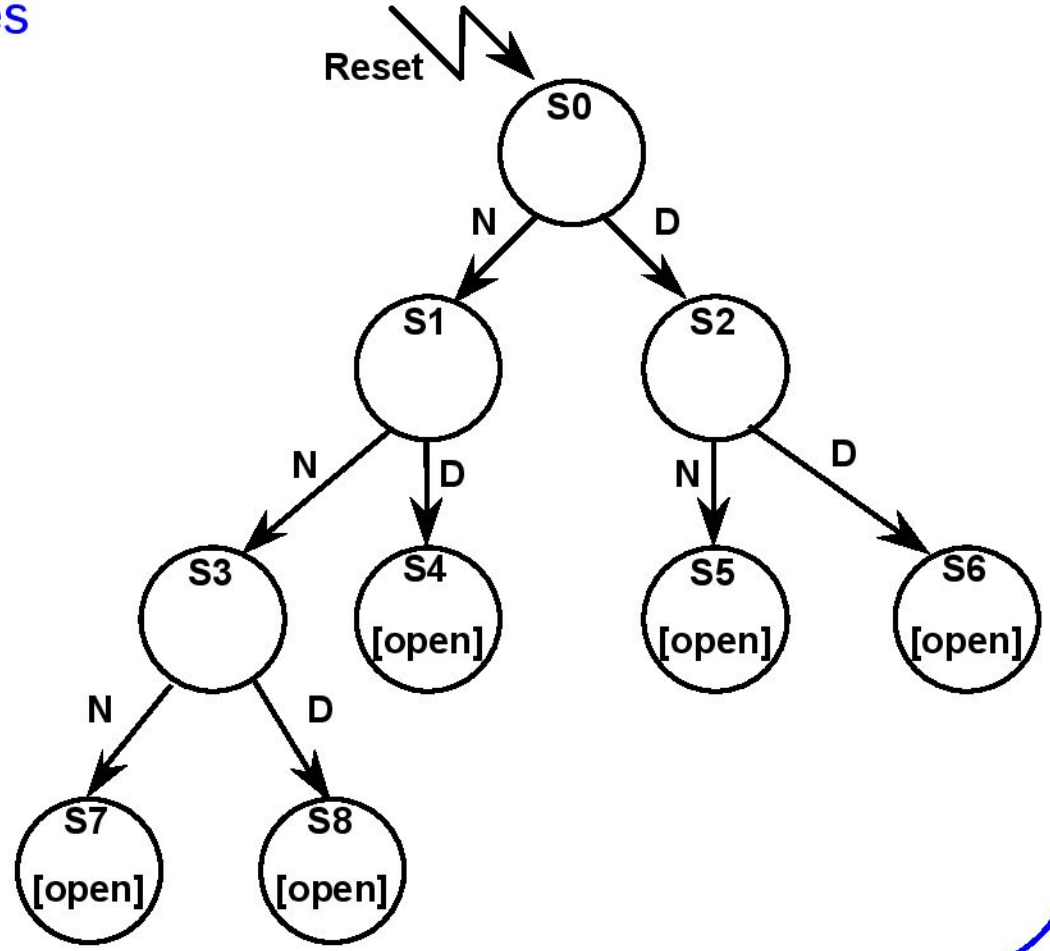


Example: Vending Machine

- **Step 2: Draw state diagram:**
  - All possible sequences
    - Inputs: N, D, reset
    - Output: open

Dime: 10¢

Nickel: 5¢



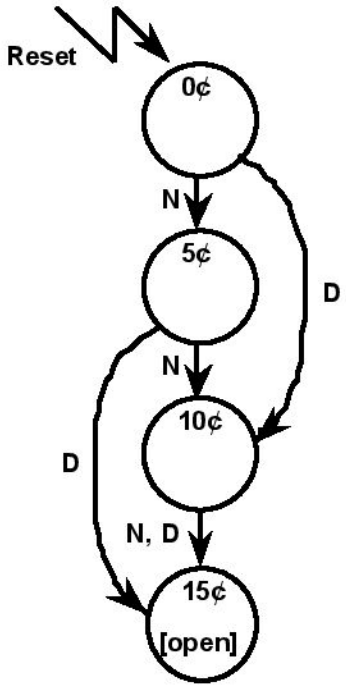
- **Notes:**
  - If neither N nor D, goes to itself.
  - Both N and D is not possible.

Example: Vending Machine

- **Step 3: State minimization:**
  - Reuse states whenever possible

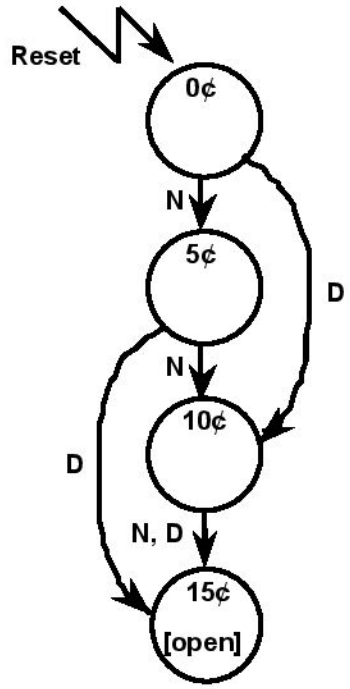
Dime: 10¢

Nickel: 5¢



Example: Vending Machine

- **Step 4: Symbolic state table:**



Present State	Inputs D	Inputs N	Next State	Output Open
0¢	0	0	0¢	0
	0	1	5¢	0
	1	0	10¢	0
	1	1	X	X
5¢	0	0	5¢	0
	0	1	10¢	0
	1	0	15¢	0
	1	1	X	X
10¢	0	0	10¢	0
	0	1	15¢	0
	1	0	15¢	0
	1	1	X	X
15¢	X	X	15¢	1

From 15¢ state, you may want to go to reset state

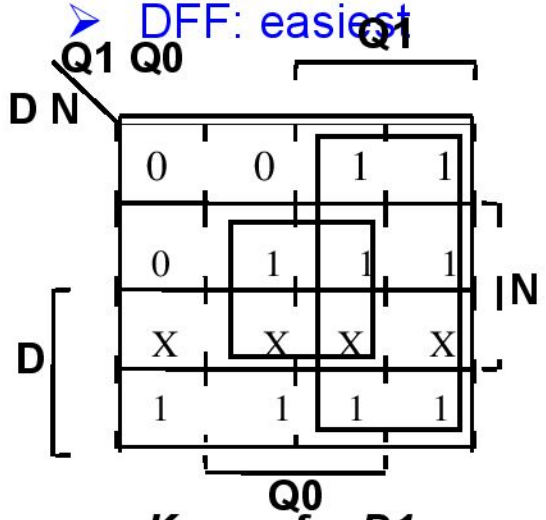
Example: Vending Machine

- **Step 5: State assignment/encoding:**

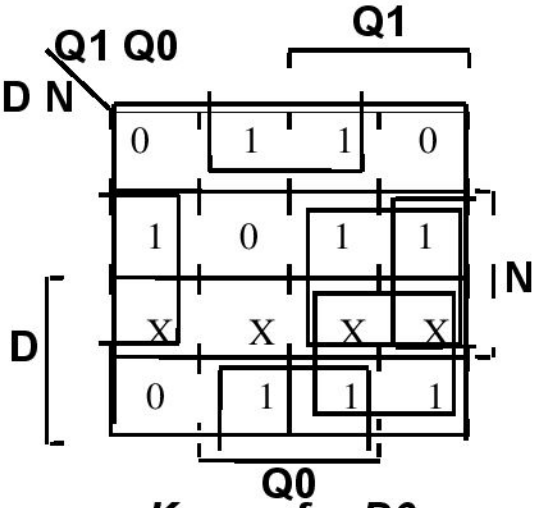
Present State Q_1	Q_0	Inputs D	N	Next State D_1	D_0	Output Open
0	0	0	0	0	0	0
		0	1	0	1	0
		1	0	1	0	0
		1	1	X	X	X
0	1	0	0	0	1	0
		0	1	1	0	0
		1	0	1	1	0
		1	1	X	X	X
1	0	0	0	1	0	0
		0	1	1	1	0
		1	0	1	1	0
		1	1	X	X	X
1	1	0	0	1	1	1
		0	1	1	1	1
		1	0	1	1	1
		1	1	X	X	X

Example: Vending Machine

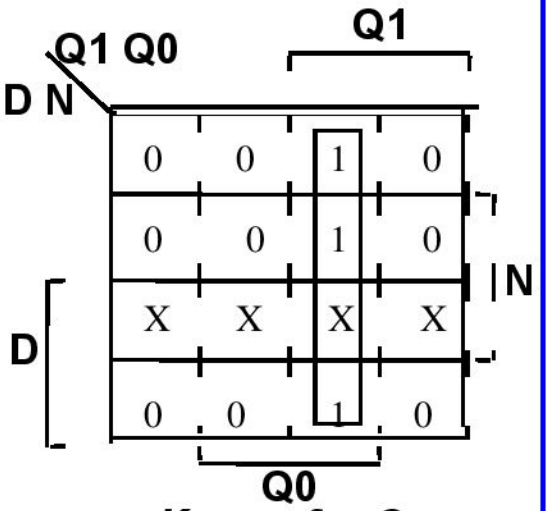
- **Step 6: Choose FF for implementation:**
  - DFF: easiest



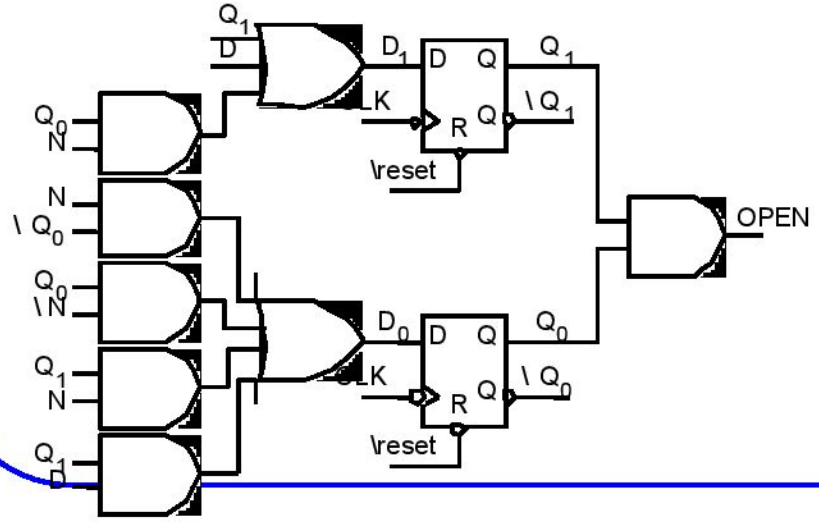
K-map for D1



K-map for D0



K-map for Open



D1 = Q1 + D + Q0 N

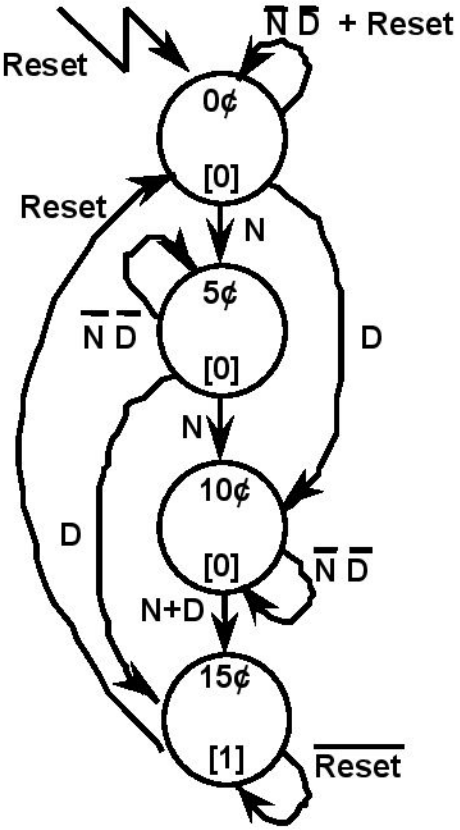
D0 = N Q0' + Q0 N' + Q1 N + Q1 D

OPEN = Q1 Q0

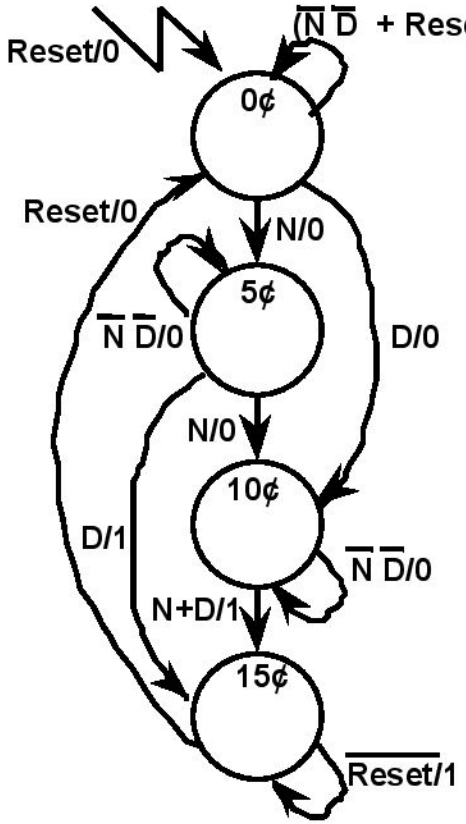
8 Gates

Equivalence of Moore and Mealy Machines

Moore Machine



Outputs are associated with the states



Mealy Machine

Outputs are associated with the transitions

Using Other FFs for Design

- **Characteristic Table:**
  - Defines the next state of the flip-flop in terms of flip-flop inputs and current state.
    - Used in Circuit Analysis
- **Excitation Table:**
  - Defines the flip-flop input variable values as function of the current state and next state.
    - Used in Circuit Design

SR FF Tables

- **Characteristic Table:**

S	R	Q(t+1)	Operation
0	0	*Q(t)*	No change
0	1	0	Reset
1	0	1	Set
1	1	?	Undefined

- **Excitation Table:**

Q(t) →	Q(t+ 1)	S	R	Operation
0	0	0	X	No change/Reset
0	1	1	0	Set
1	0	0	1	Reset
1	1	X	0	No change/Set

DFF Tables

- **Characteristic Table:**

D	Q(t+1)	Operation
0	0	Reset
1	1	Set

- **Excitation Table:**

Q(t) →	Q(t+1)	D	Operation
X	0	0	Reset
X	1	1	Set

JK FF Tables

- **Characteristic Table:**

J	K	Q(t+1)	Operation
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q}(t)$	Complement

- **Excitation Table:**

Q(t)	Q(t+1)	J	K	Operation
0	0	0	X	No change/Reset
0	1	1	X	Set/Toggle
1	0	X	1	Reset/Toggle
1	1	X	0	No Change/Set

T FF Tables

- **Characteristic Table:**

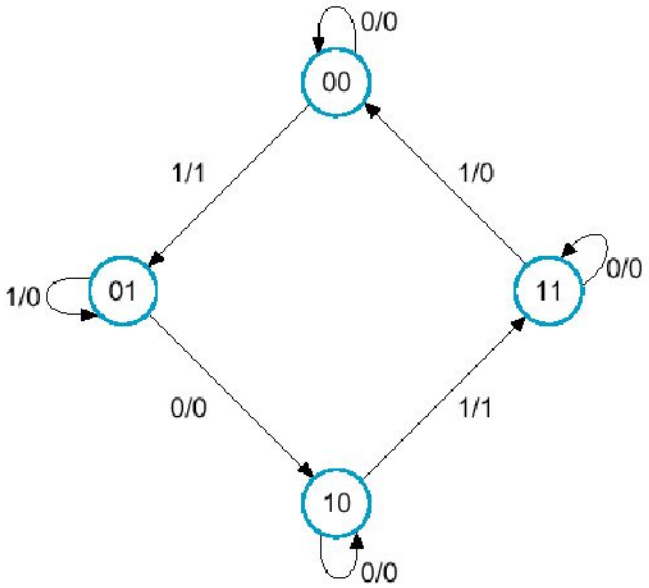
T	Q(t+1)	Operation
0	$Q(t)$	No change
1	$\overline{Q}(t)$	Complement

- **Excitation Table:**

Q(t)	Q(t+1)	T	Operation
0	0	0	No change
0	1	1	Toggle
1	0	1	Toggle
1	1	0	No Change

Example

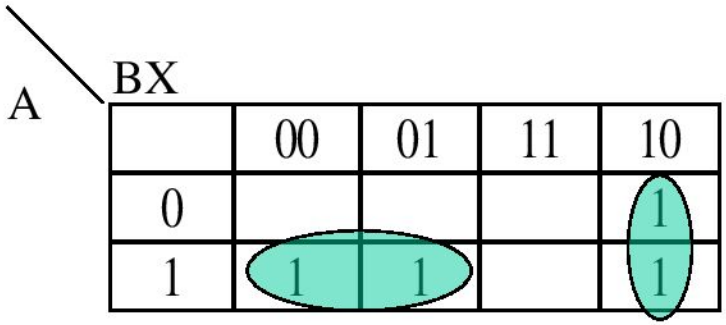
- **Design by DFF**



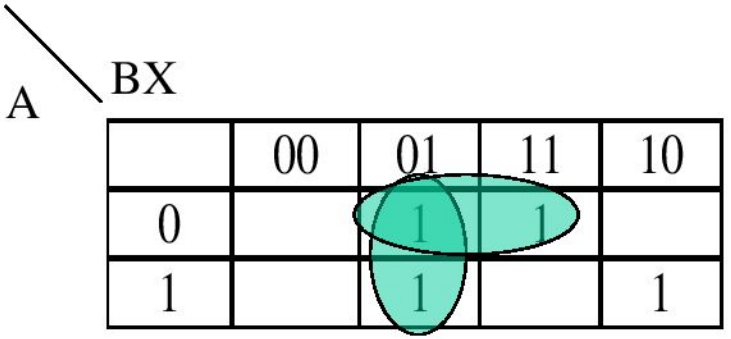
Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	0	0	0

Example

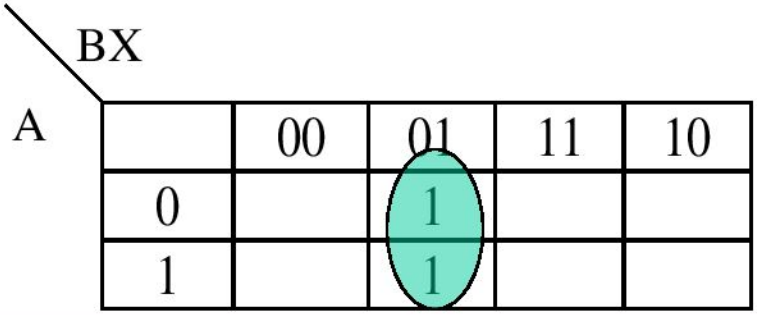
$$A(t+1) = D_A(A,B,X) = \Sigma\, m(2,4,5,6)$$
$$B(t+1) = D_B(A,B,X) = \Sigma\, m(1,3,5,6)$$
$$Y(A,B,X) = \Sigma\, m(1,5)$$



$$D_A = A\overline{B} + B\overline{X}$$



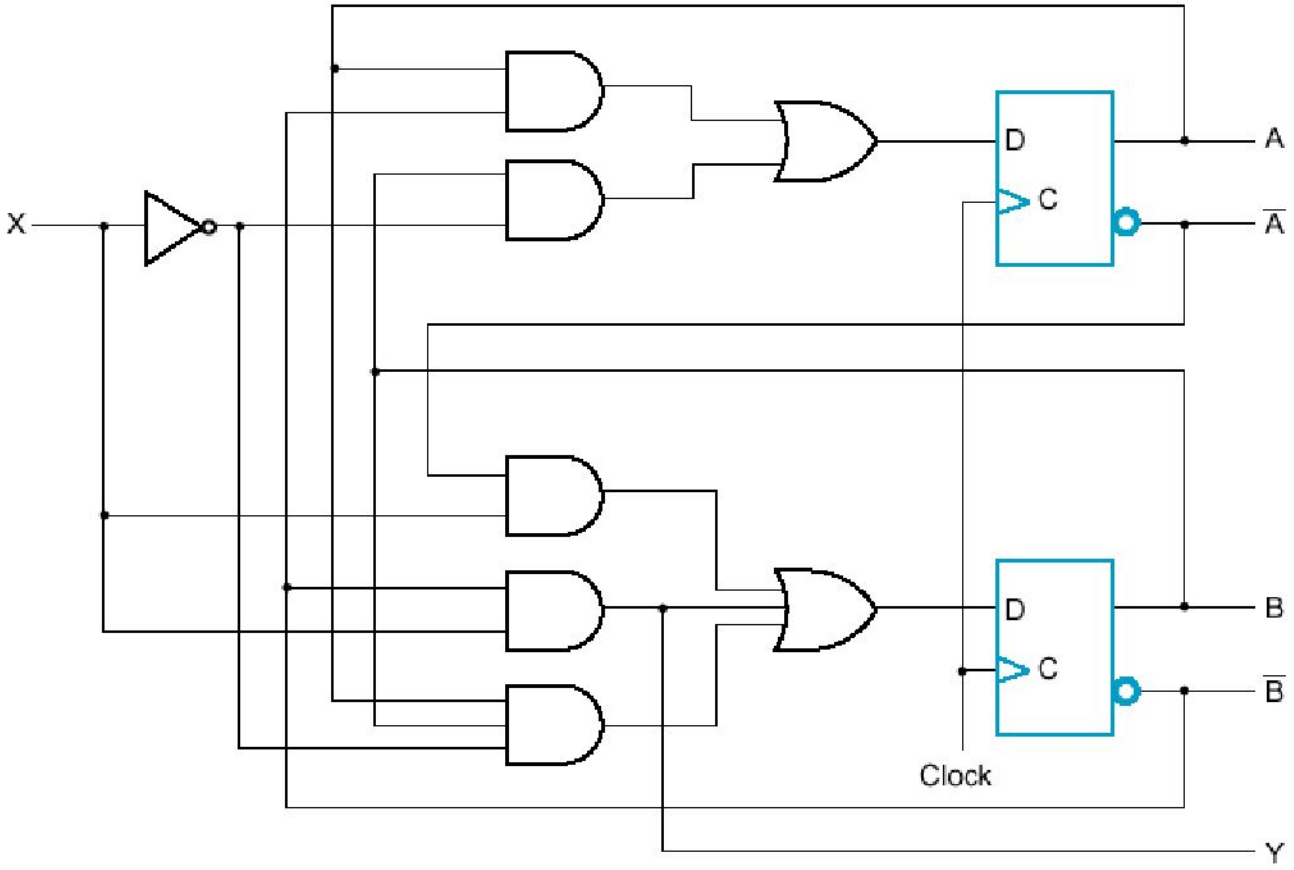
$$D_B = \overline{A}X + \overline{B}X + AB\overline{X}$$



$$Y = \overline{B}X$$

Example

- Logic Diagram for Circuit with D Flip-Flops



Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	0	0	0

Q(t) → Q(t+1)		J	K	Operation
0	0	0	X	No change/reset
0	1	1	X	Set/Toggle
1	0	X	1	Reset/Toggle
1	1	X	0	No change/set

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

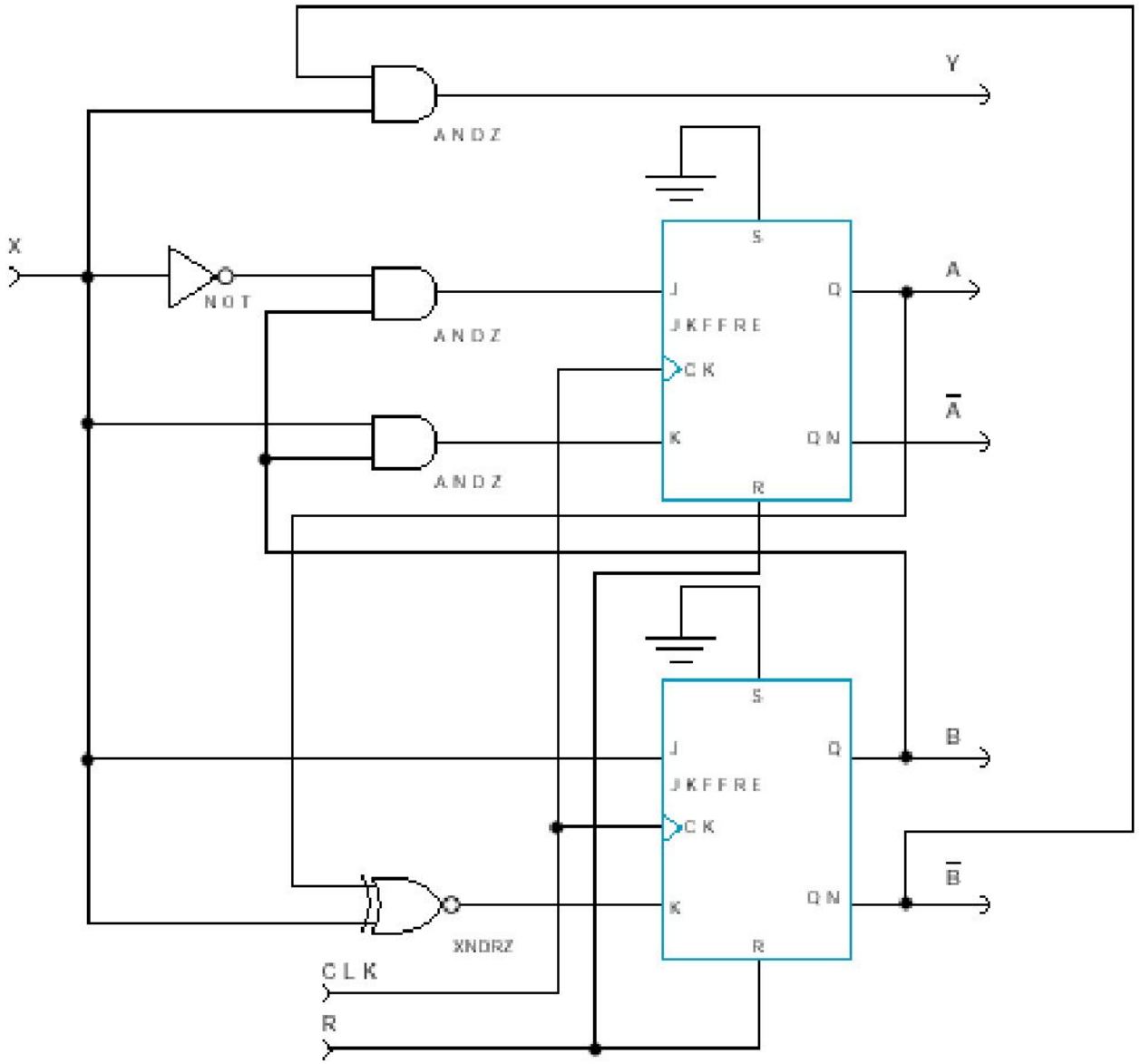
- Don't cares
  - lead to simpler combinational circuit

Example: Boolean Equations

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

$$J_A = B\overline{X} \qquad J_B = X$$

$$K_A = BX \qquad K_B = AX + \overline{A}\,\overline{X}$$


Y
ANDZ
X
NOT
ANDZ
S
J
Q
JKFFRE
CK
A
K
QN
R
ANDZ
S
J
Q
JKFFRE
CK
B
K
QN
R
XNORZ
CLK
R