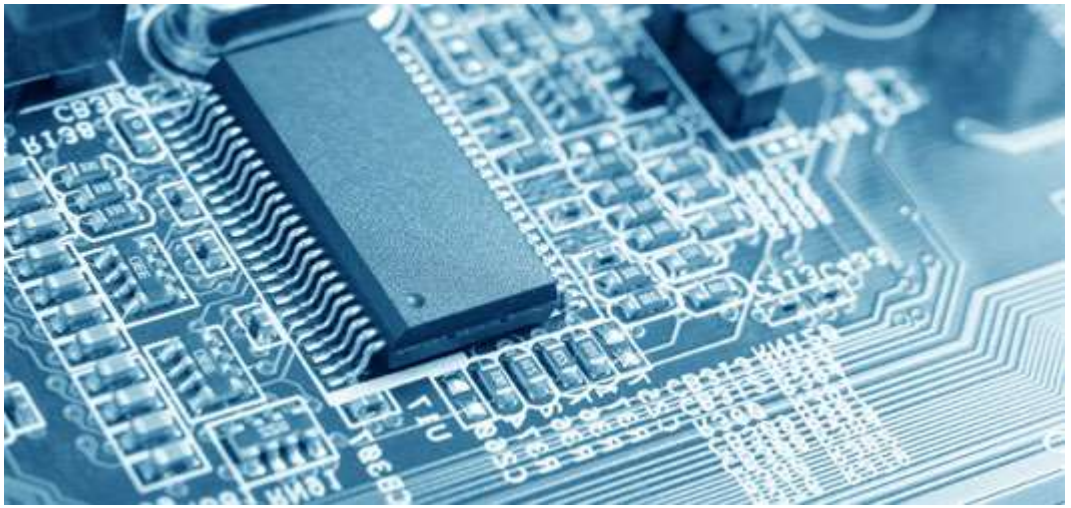
NAZARBAYEV UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY



ROBT206 – Microcontrollers with Lab

Lecture 15-16 – Sequential Circuit Design

13-15 March, 2018

Course Logistics

Important Dates and Tasks
Homework #3 due to end of March 25th (Sunday) Mano 4.3, 4.4, 5.3, 5.4, 5.7, 5.8, 5.10, 5.13, 5.15, 5.19, 5.22
Quiz #3 on 29th of March (Thursday) class time Chapter 4 and 5 – Sequential Circuits Analysis

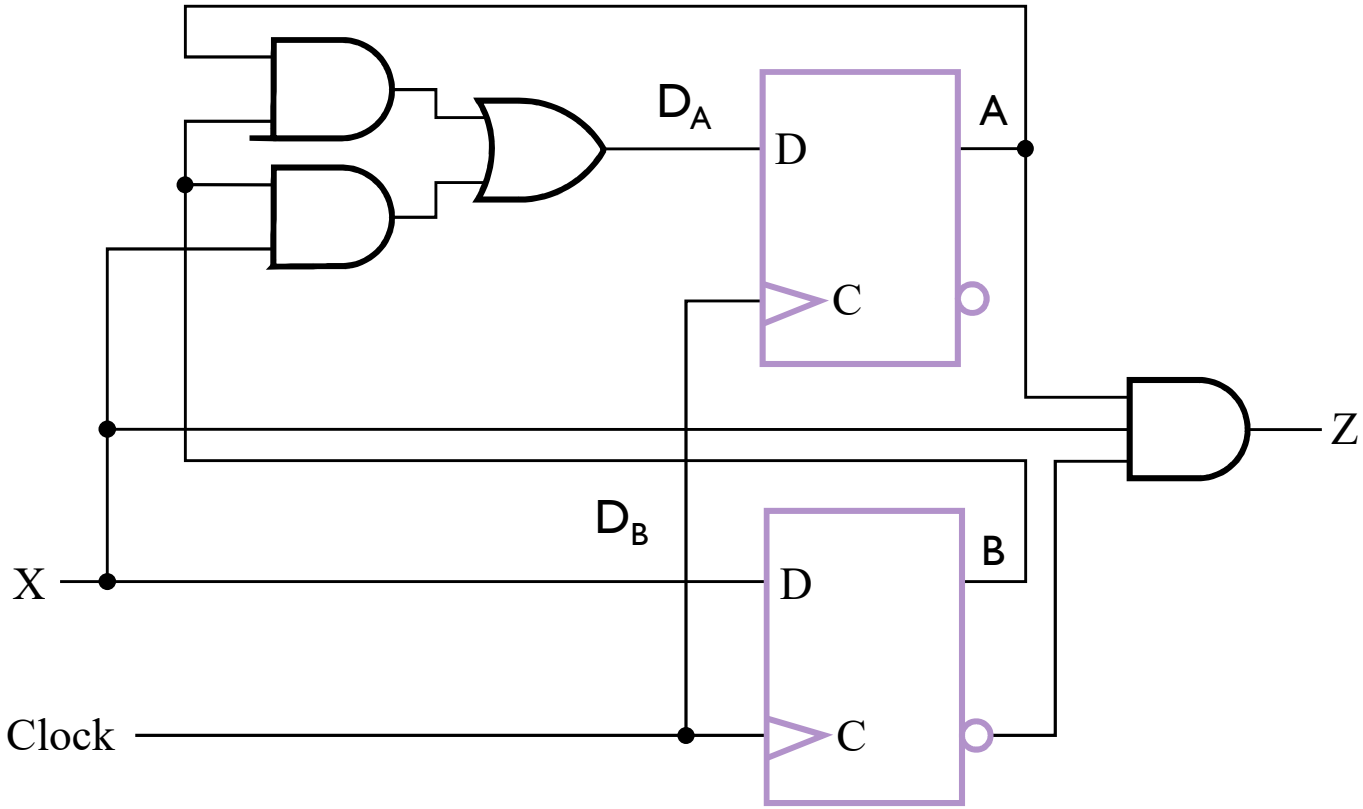
Topics

Today's Topics

Sequential Circuit Design

- Specification
- Formulation
- State Assignment
- Flip-Flop Input and Output Equation Determination
- Verification

Fill the state table, find the state and output expressions and draw the state diagram



The Design Procedure

1. **Specification**
2. **Formulation -** Obtain a state diagram or state table
3. **State Assignment -** Assign binary codes to the states
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

Specification

- Component Forms of Specification
  - Written description
  - Mathematical description
  - Hardware description language*
  - Tabular description*
  - Equation description*
  - Diagram describing operation (not just structure)*
- Relation to Formulation
  - If a specification is rigorous at the binary level (marked with * above), then all or part of formulation may be completed

Formulation: Finding a State Diagram

- A **state** is an abstraction of the history of the past applied inputs to the circuit (including power-up reset or system reset).
  - The interpretation of "past inputs" is tied to the synchronous operation of the circuit. E. g., an input value (other than an asynchronous reset) is measured only during the setup-hold time interval for an edge-triggered flip-flop.
- Examples:
  - State A represents the fact that a **1** input has occurred among the past inputs.
  - State B represents the fact that a **0** followed by a **1** have occurred as the most recent past two inputs.

Formulation: Finding a State Diagram

- In specifying a circuit, we use states to remember meaningful properties of past input sequences that are essential to predicting future output values.
- A **sequence recognizer** is a sequential circuit that produces a distinct output value whenever a prescribed pattern of input symbols occur in sequence, i.e, recognizes an input sequence occurrence.
- We will develop a procedure specific to sequence recognizers to convert a problem statement into a state diagram.
- Next, the state diagram, will be converted to a state table from which the circuit will be designed.

Sequence Recognizer Procedure

- To develop a sequence recognizer state diagram:
  - Begin in an initial state in which NONE of the initial portion of the sequence has occurred (typically "reset" state).
  - Add a state that recognizes that the first symbol has occurred.
  - Add states that recognize each successive symbol occurring.
  - The final state represents the input sequence (possibly less the final input value) occurrence.
  - Add state transition arcs which specify what happens when a symbol *not* in the proper sequence has occurred.
  - Add other arcs on non-sequence inputs which transition to states that represent the input subsequence that has occurred.
- The last step is required because the circuit must recognize the input sequence *regardless of where it occurs within the overall sequence applied since "reset."*.

State Assignment

- Each of the *m* states must be assigned a unique code
- Minimum number of bits required is *n* such that

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

where $\lceil x \rceil$ is the smallest integer $\geq x$ *(ceiling)*

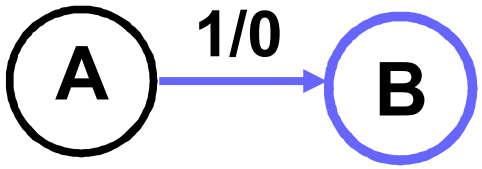
- There are useful state assignments that use more than the minimum number of bits (Then, there are 2^n - m unused states)

Sequence Recognizer Example

- Example: Recognize the sequence 1101
  - Note that the sequence 1111101 contains 1101 and "11" is a proper sub-sequence of the sequence.
- 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.
- And, 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

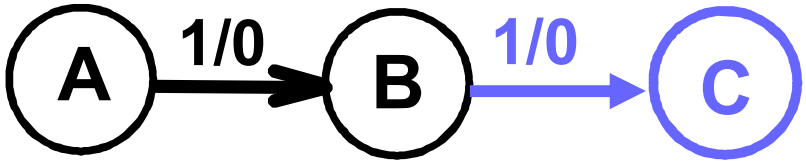
- Define states for the sequence to be recognized:
  - assuming it starts with first symbol,
  - continues through each symbol in the sequence to be recognized,
  - and uses output 1 to mean the full sequence has occurred,
  - with output 0 otherwise.
- Starting in the initial state (Arbitrarily named "A"):
  - Add a state that recognizes the first "1."



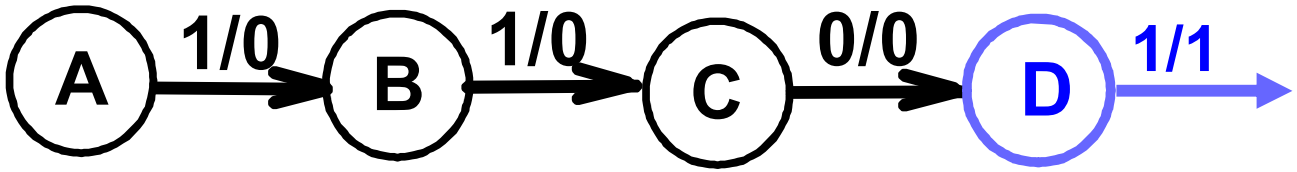
  - State "A" is the initial state, and state "B" is the state which represents the fact that the "first" one in the input subsequence has occurred. The output symbol "0" means that the full recognized sequence has not yet occurred.

Example: Recognize 1101 (continued)

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

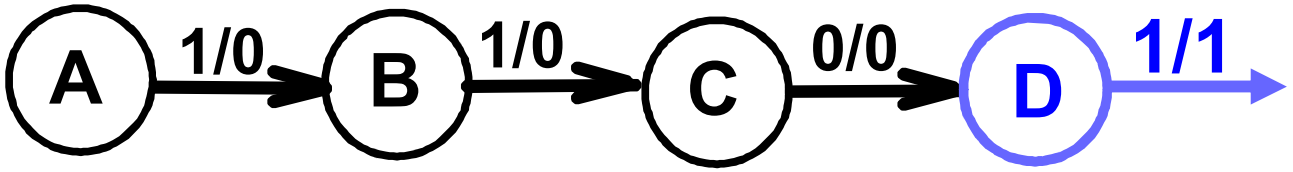


  - Finally, after 110 and a 1, we have:

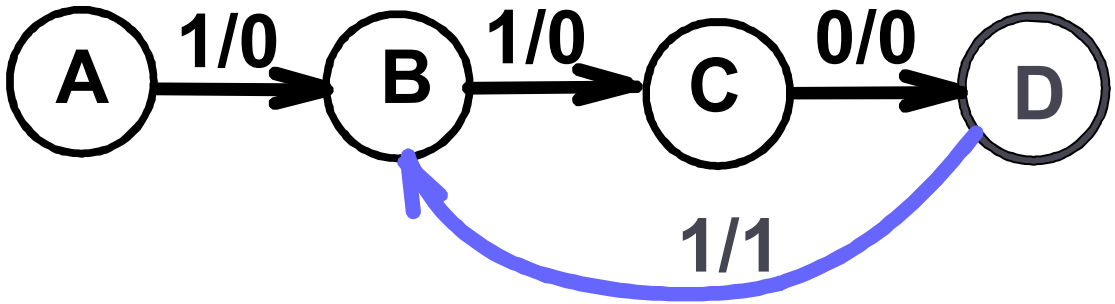


  - Transition arcs are used to denote the output function (Mealy Model)
  - Output 1 on the arc from D means the sequence has been recognized
  - To what state should the arc from state D go? Remember: 1101<u>1101</u> ?
  - Note that D is the last state but the output 1 occurs for the input applied in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

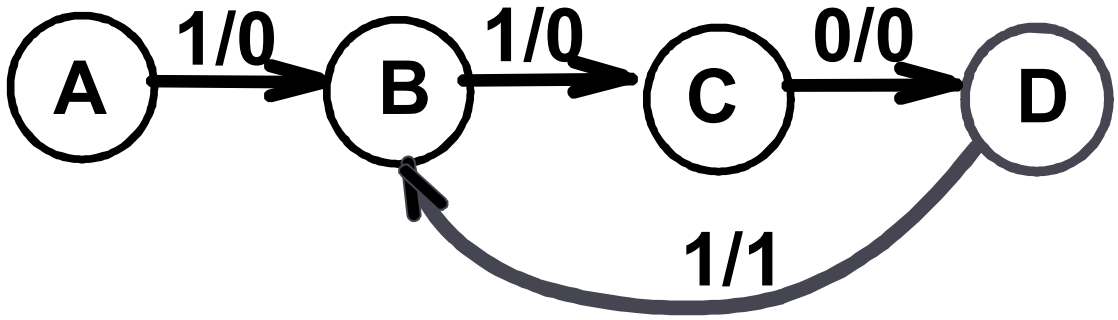
Example: Recognize 1101 (continued)



- 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:



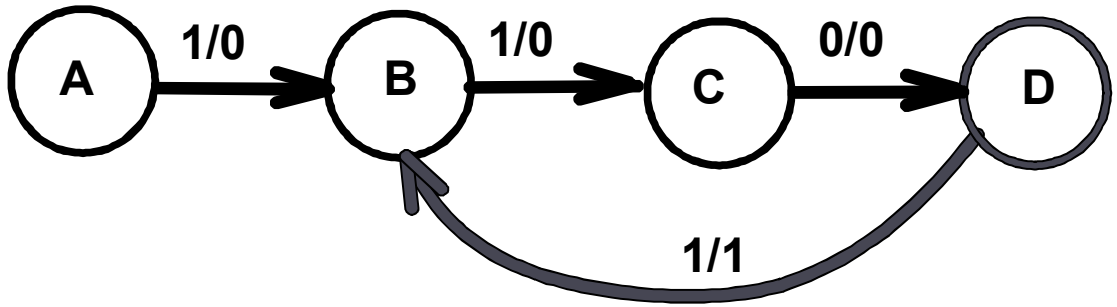
Example: Recognize 1101 (continued)



- The states have the following 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.

Example: Recognize 1101 (continued)

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



Answer:

"0" arc from A

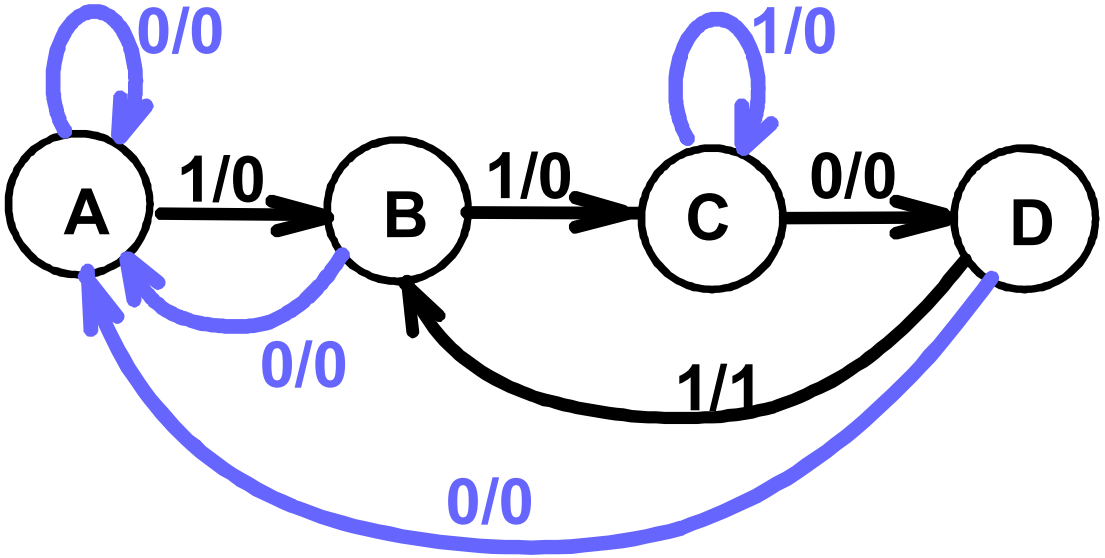
"0" arc from B

"1" arc from C

"0" arc from D.

Example: Recognize 1101 (continued)

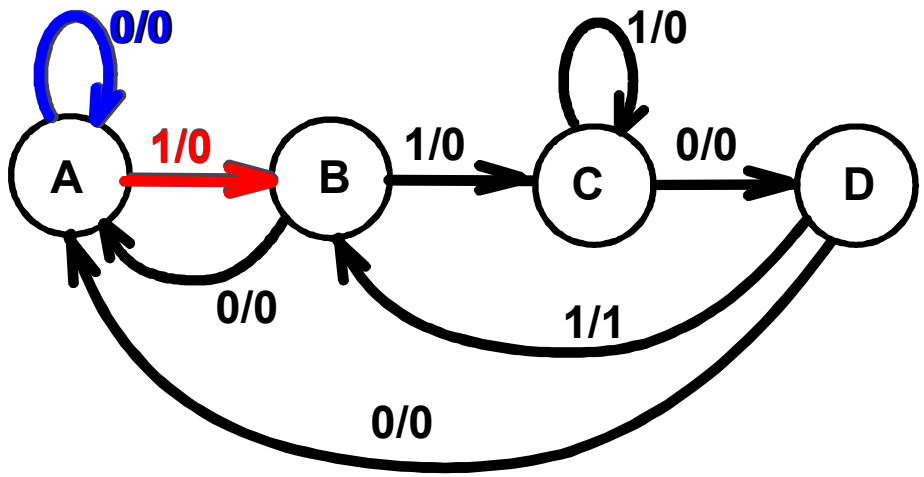
- State transition arcs must represent the fact that an input subsequence has occurred. Thus we get:



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

Formulation: Find State Table

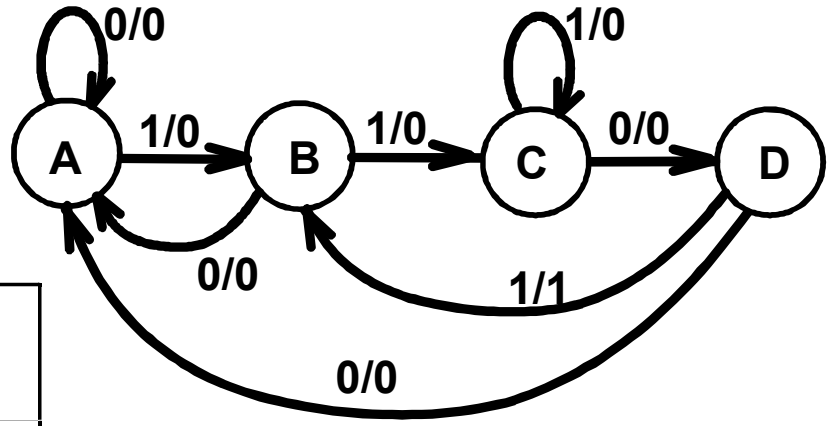
- From the State Diagram, we can fill in the State Table.
  - There are 4 states, one input, and one output. We will choose the form with four rows, one for each current state.
  - From State A, the 0 and 1 input transitions have been filled in along with the outputs.



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

Formulation: Find State Table

- From the state diagram, we complete 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

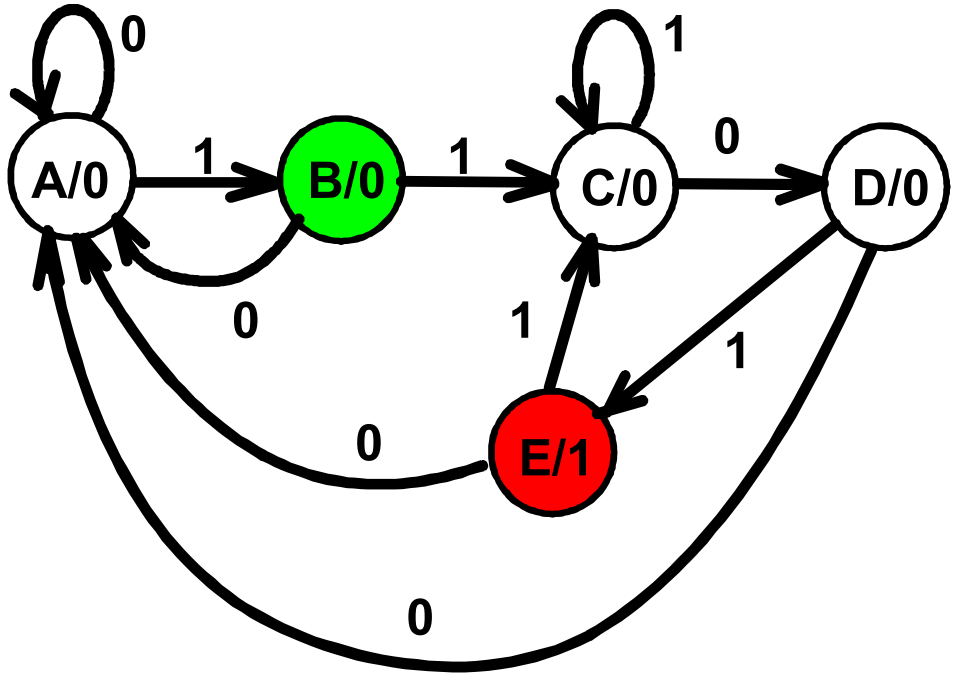
- What would the state diagram and state table look like for the Moore model?

Example: Moore Model for Sequence 1101

- 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.

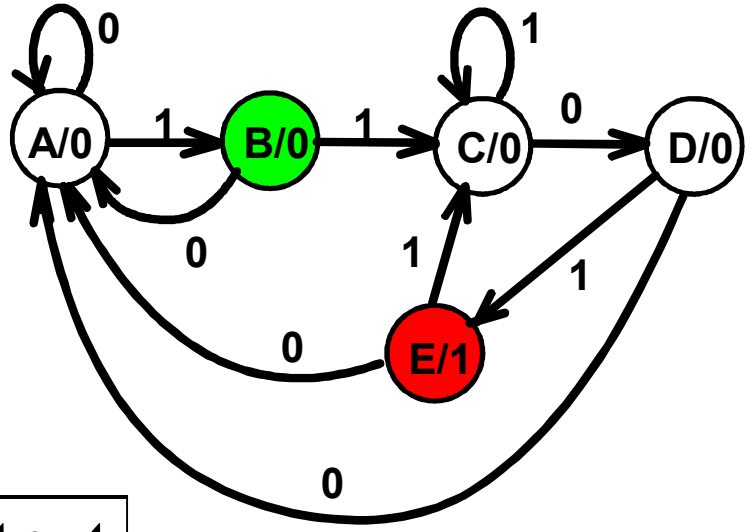
Example: Moore Model (continued)

- We mark outputs on states for Moore model
- Arcs now show only state transitions
- Add a new state E to produce the output 1



- Note that the new state, E produces the same behavior in the future as state B. But it gives a different output at the present time. Thus these states do represent a different abstraction of the input history.

Example: Moore Model (continued)



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

State Assignment – Example 2 (continued)

- Counting Order Assignment:

  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
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

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

Find Flip-Flop Input and Output Equations: Example 2 – Counting Order Assignment

- Assume D flip-flops
- 2 flip-flops are needed to represent 4 states
- Interchange the bottom two rows of the state table, to obtain K-maps for D1, D2, and Z:

D1

A \ B X	0	1
0	0	1
0	0	0
1	1	1

D2

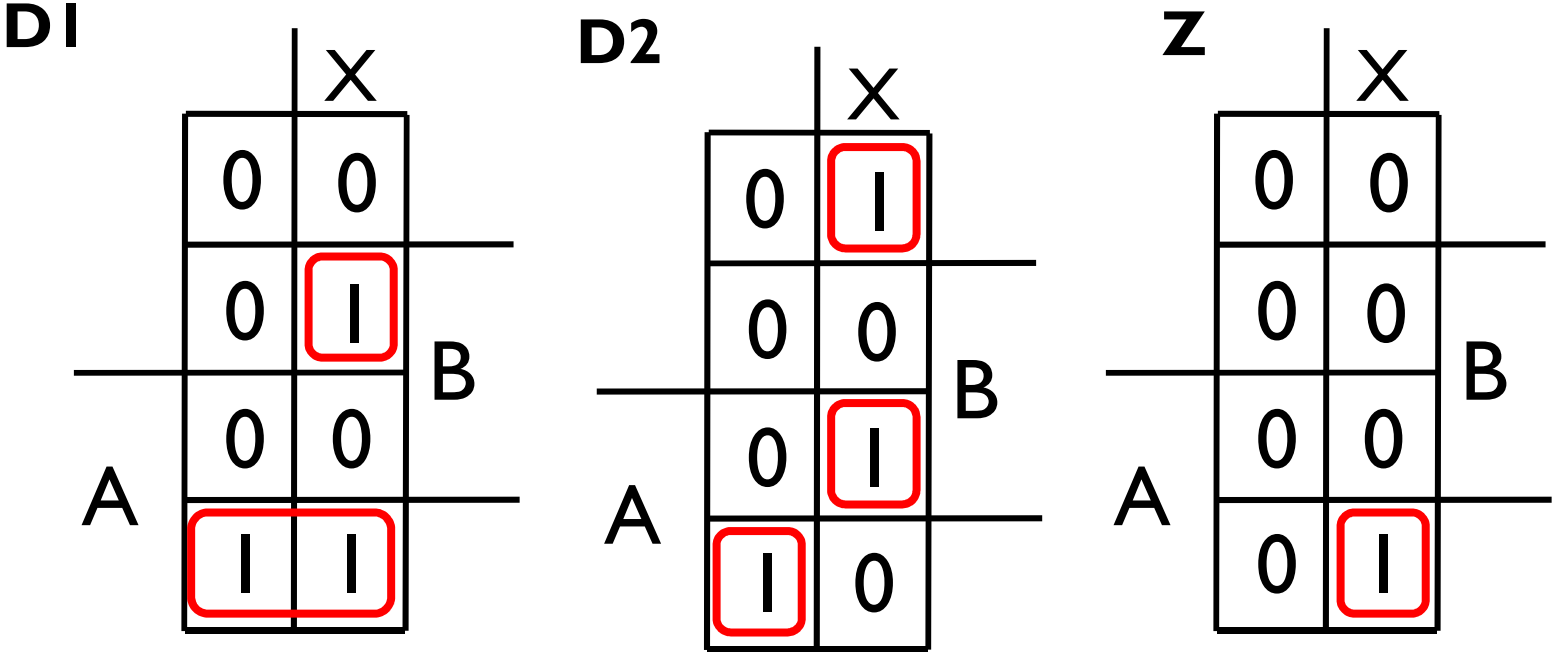
A \ B X	0	1
0	0	1
0	0	0
1	1	0

Z

A \ B X	0	1
0	0	0
0	0	0
0	0	1

Optimization: Example 2: Counting Order Assignment

- Performing two-level optimization:



$D_1 = A\bar{B} + X\bar{A}B$

$D_2 = \bar{X}A\bar{B} + X\bar{A}\bar{B} + XAB$

$Z = XA\bar{B}$ Gate Input Cost = 22

State Assignment – Example 2 (continued)

- Gray Code Assignment:

A = 0 0, B = 0 1, C = 1 1, D = 1 0

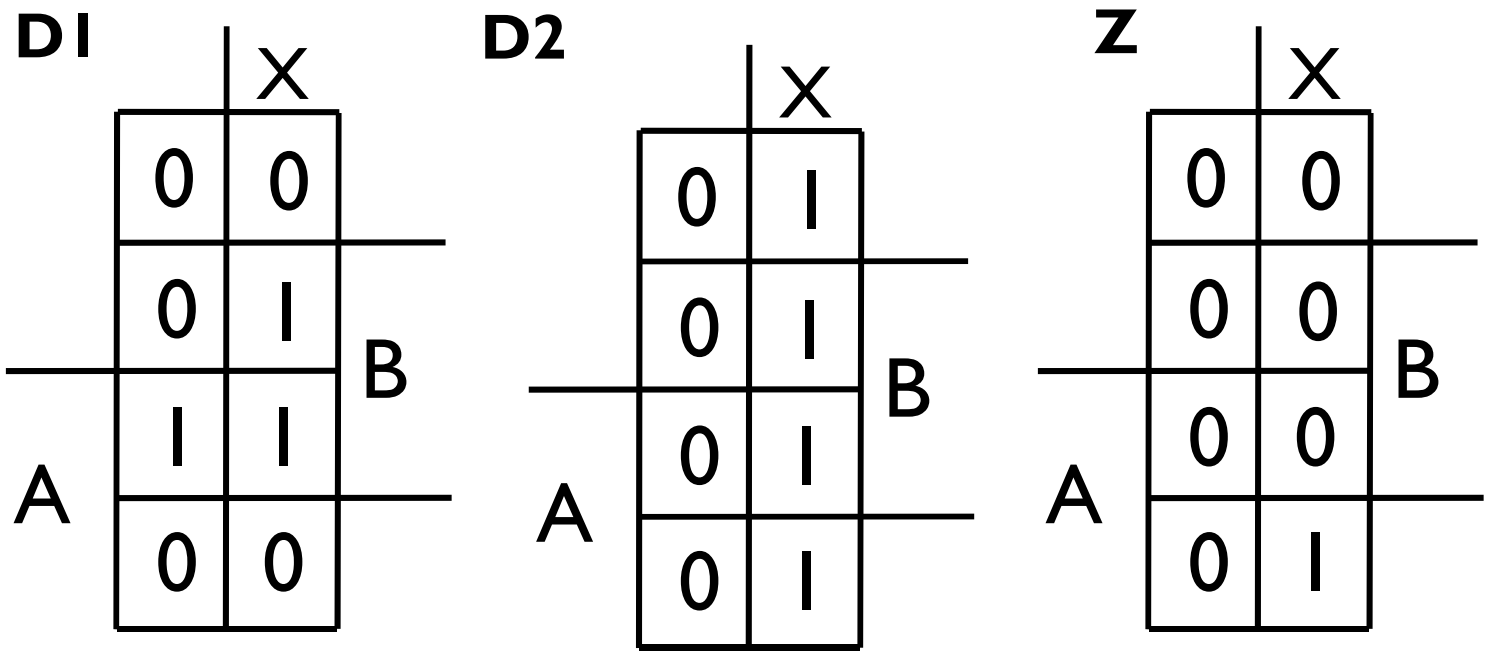
- The resulting coded state table:

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

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

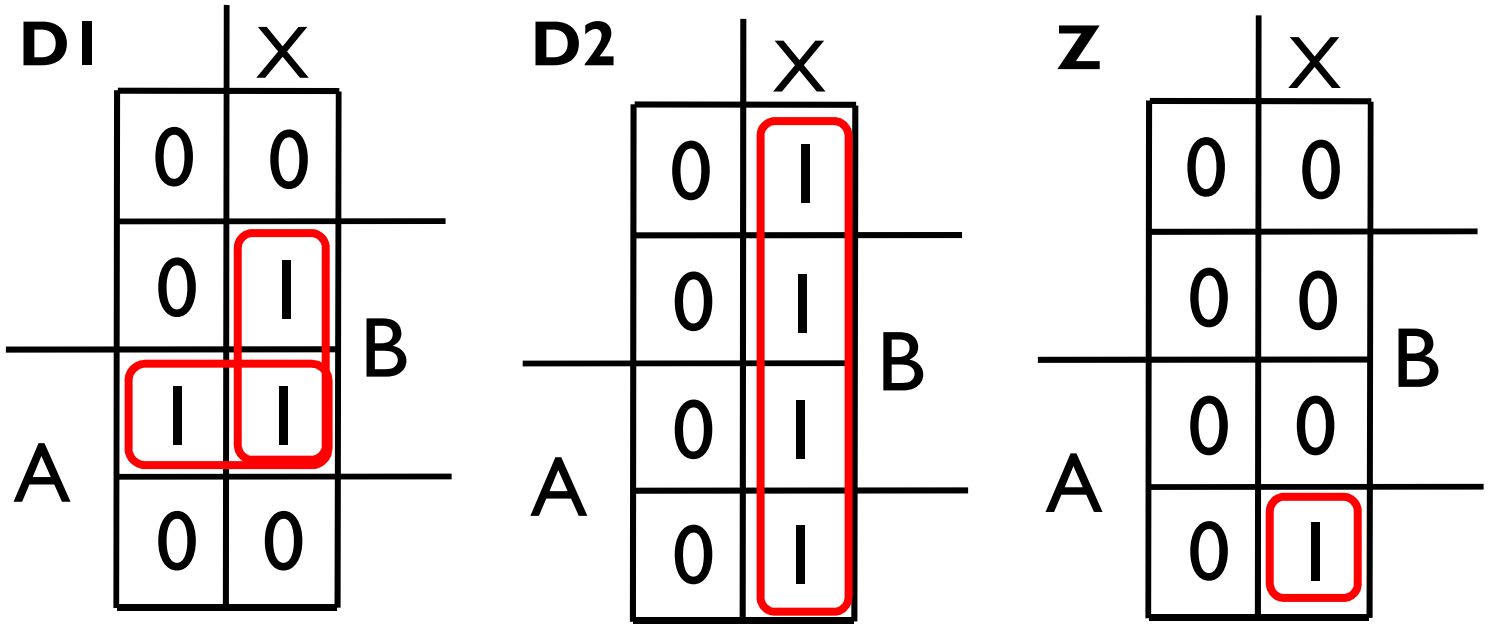
Find Flip-Flop Input and Output Equations: Example 2 – Gray Code Assignment

- Assume D flip-flops
- Obtain K-maps for D_1, D_2, and Z:



Optimization: Example 2: Assignment 2

- Performing two-level optimization:



$D_1 = AB + XB$ Gate Input Cost = 9

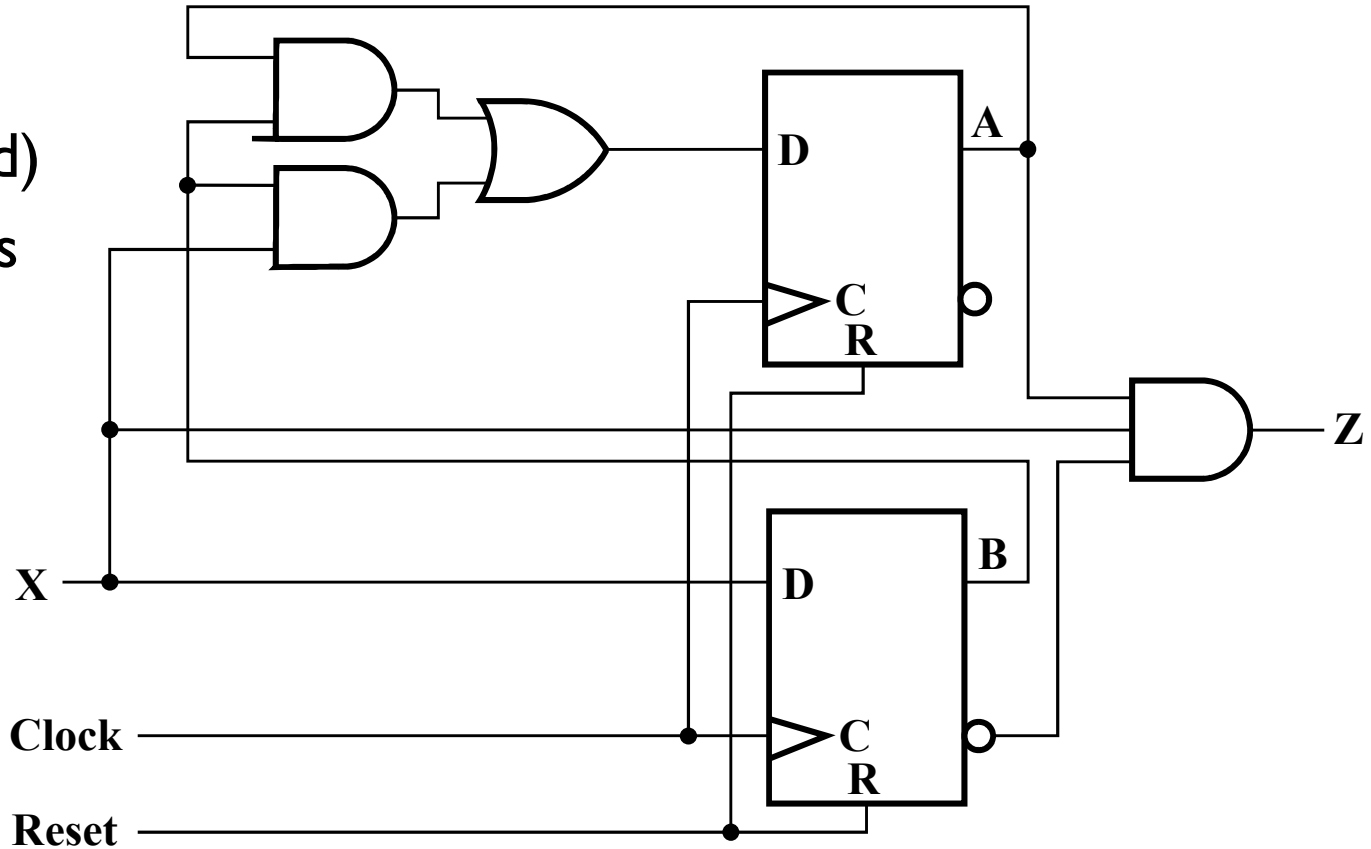
$D_2 = X$ Select this state assignment to

$Z = XA\overline{B}$ complete the design

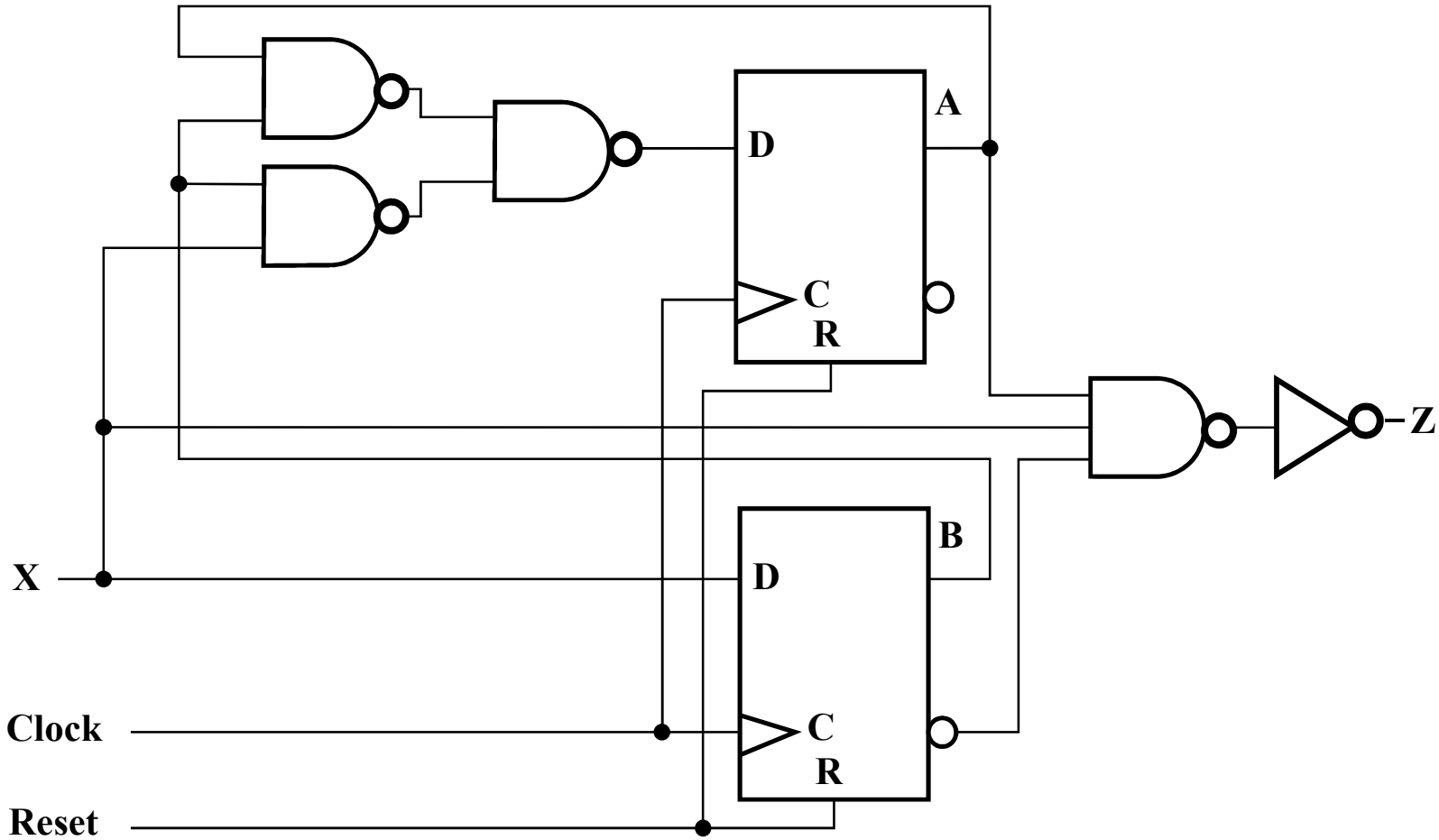
Map Technology

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

- **Initial Circuit:**



Mapped Circuit - Final Result

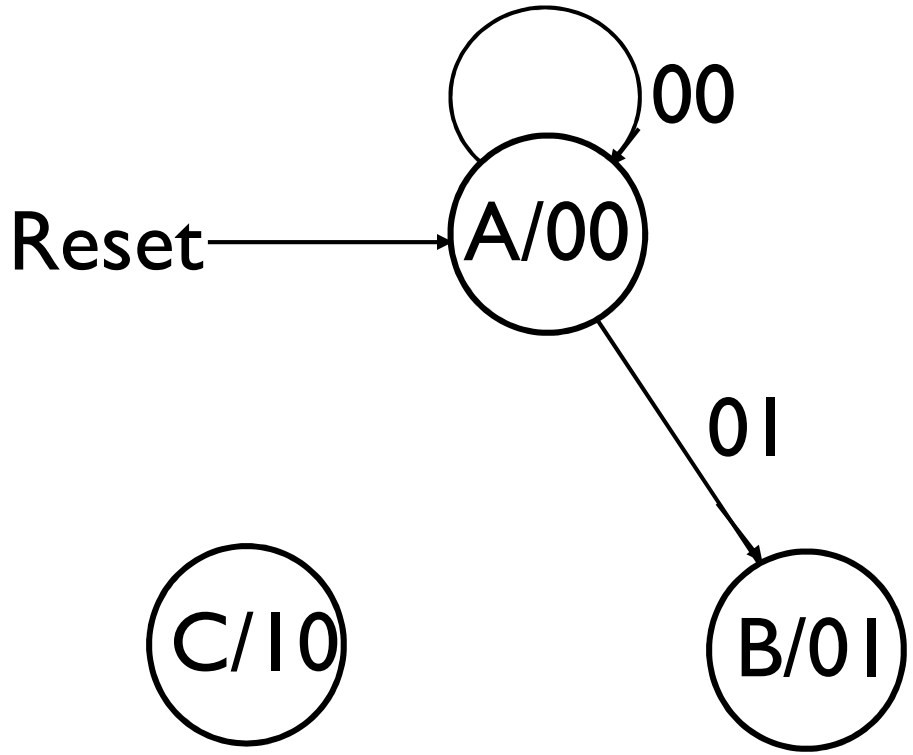


Sequential Design: Example 3

- Design a sequential modulo 3 accumulator for 2-bit operands
- Definitions:
  - Modulo *n* adder - an adder that gives the result of the addition as the remainder of the sum divided by *n*
    - Example: 2 + 2 modulo 3 = remainder of 4/3 = 1
  - Accumulator - a circuit that "accumulates" the sum of its input operands over time - it adds each input operand to the stored sum, which is initially 0.
- Stored sum: $(Y_1 Y_0)$, Input: $(X_1 X_0)$, Output: $(Z_1 Z_0)$

Example 3 (continued)

- Complete the state diagram:



Example 3 (continued)

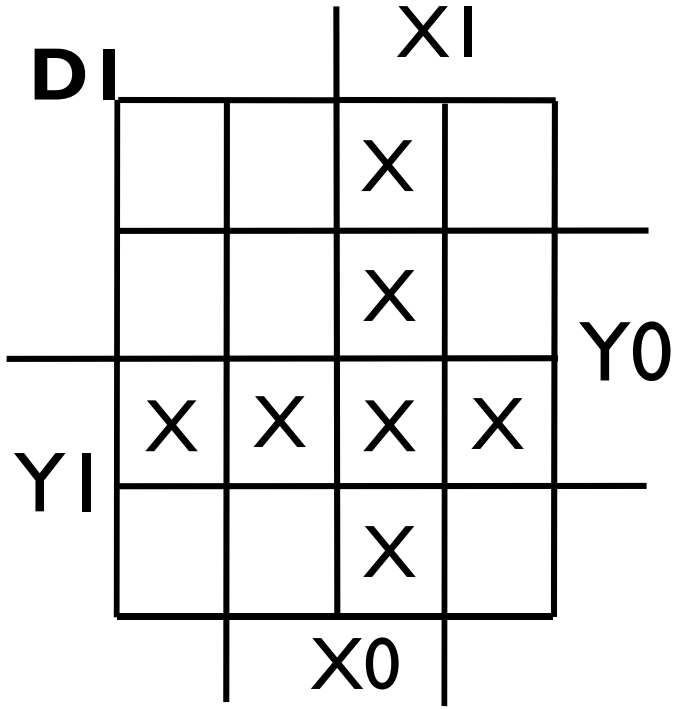
- Complete the state table

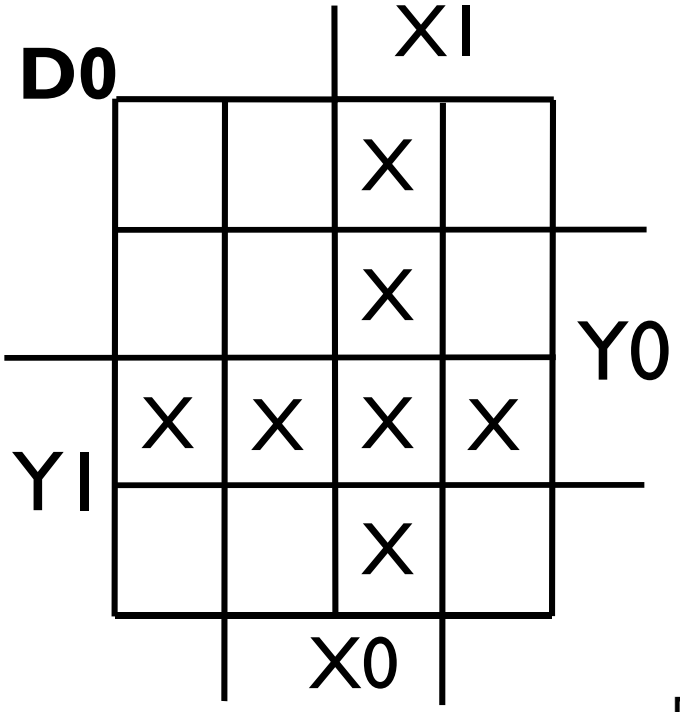
Y_1Y_0 \ X_1X_0	00	01	11	10	Z_1Z_0
	$Y_1(t+1)$, $Y_0(t+1)$	$Y_1(t+1)$, $Y_0(t+1)$	$Y_1(t+1)$, $Y_0(t+1)$	$Y_1(t+1)$, $Y_0(t+1)$	
A (00)	00		X		00
B (01)			X		01
- (11)	X	X	X	X	11
C (10)			X		10

- State Assignment: $(Y_1,Y_0) = (Z_1,Z_0)$
- Codes are in gray code order to ease use of K-maps in the next step

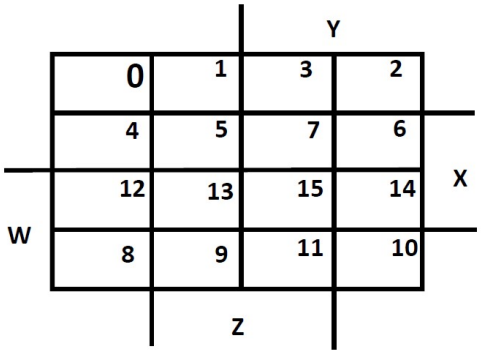
Example 3 (continued)

- Find optimized flip-flop input equations for D flip-flops

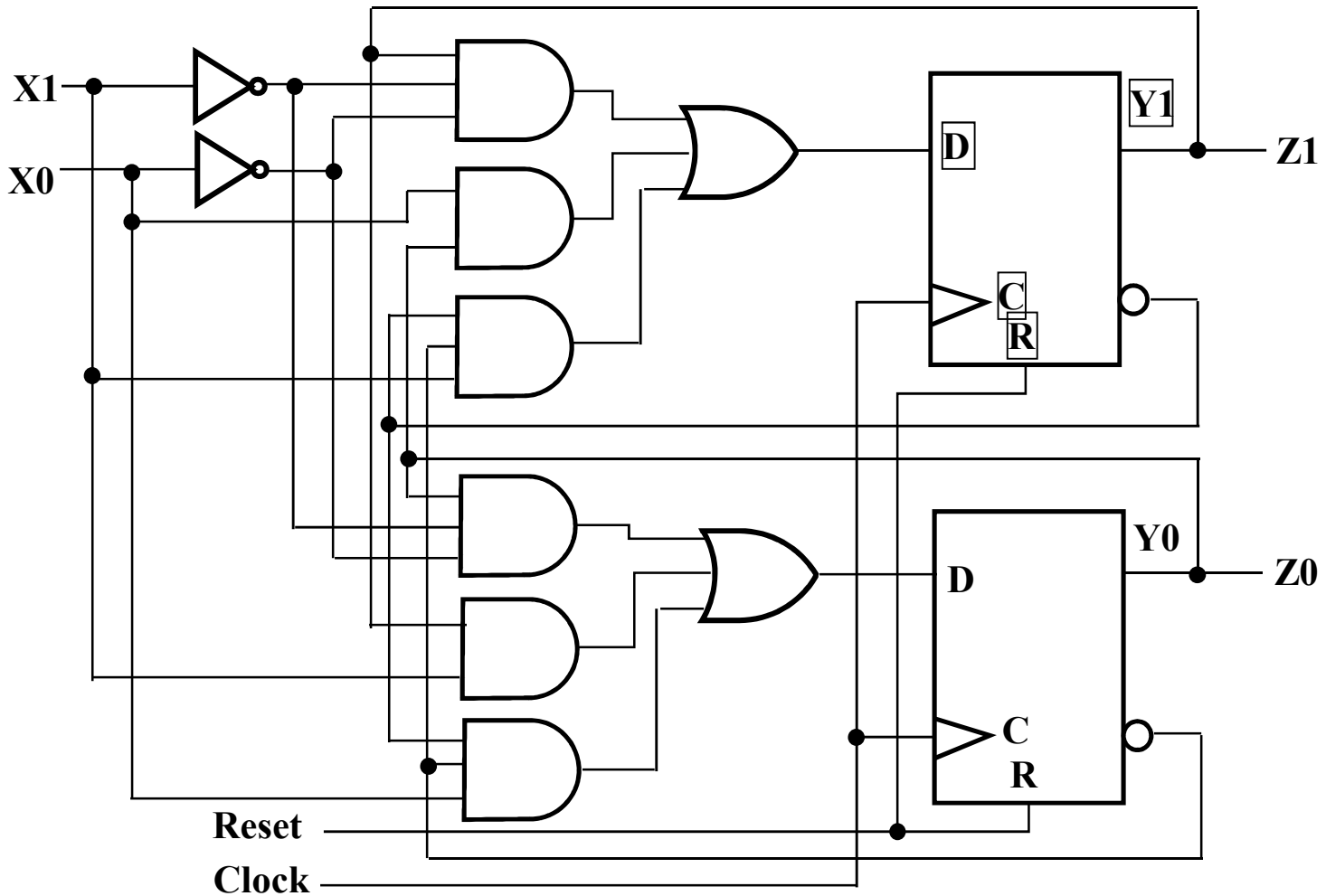




- D_1 =
- D_0 =

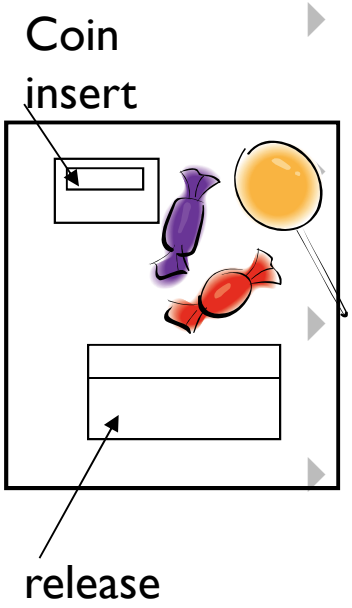


Circuit - Final Result with AND, OR, NOT



Example: Vending machine

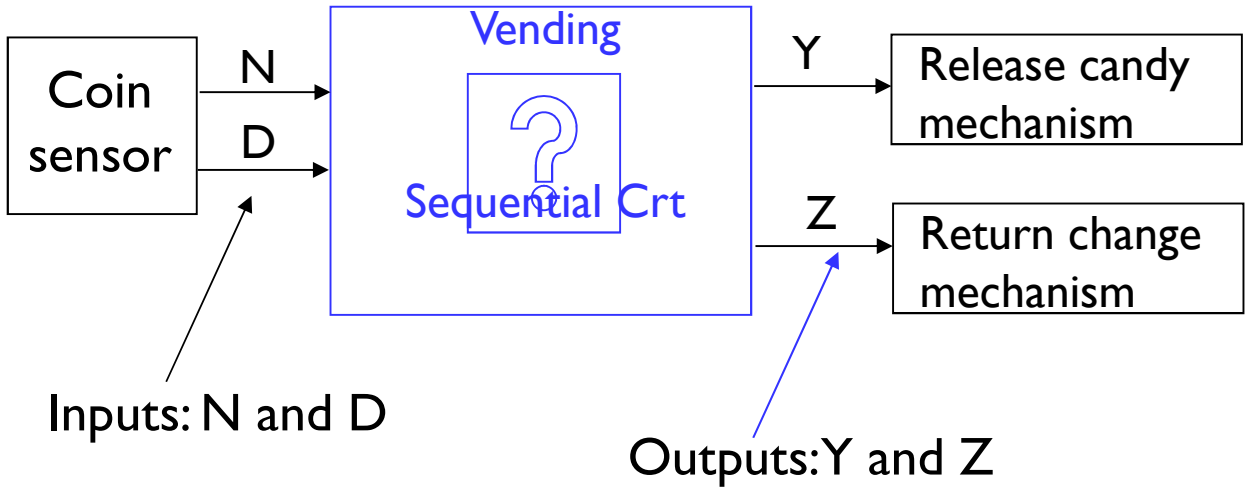
- Design the control circuit for a vending machine with the following specifications:
  - The vending machine accepts nickels (N: 5 cents) and dimes (D: 10 cents)
  - When the machine has received 15 cents it delivers a package of candy.
  - If too much money has been added, the machine returns the difference.
  - When the candy has been released, the release mechanism brings the circuit back to the original, starting state.



Design Procedure - review

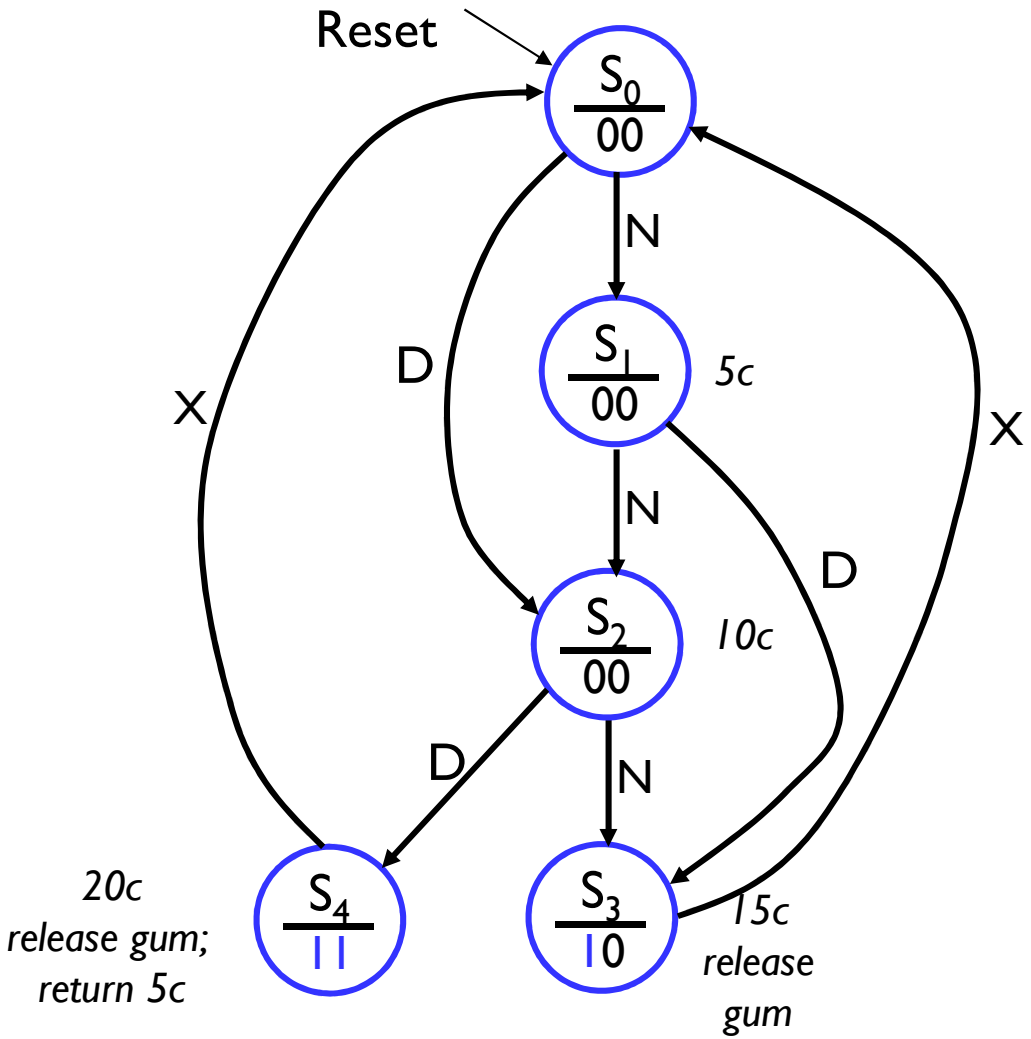
1. Understanding the problem and adding specs if needed
2. State diagram
3. State table
4. State encoding
5. Select the type of flip-flop
6. Derive the input equations to the FF; and the output equations
7. Draw the diagram
8. Verify

Step 1: Understanding the problem

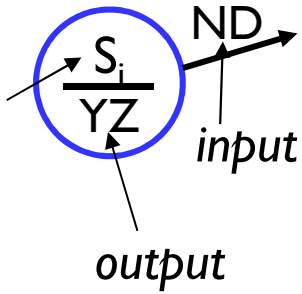


- Only one of the inputs N or D are asserted at one time (never together)
- N and D is asserted for only one clock cycle when a coin has been inserted
- Pennies are not accepted
- Z=0 (no change); Z=1 (change returned: 5 cents)

Step 2: State Diagram (Moore)

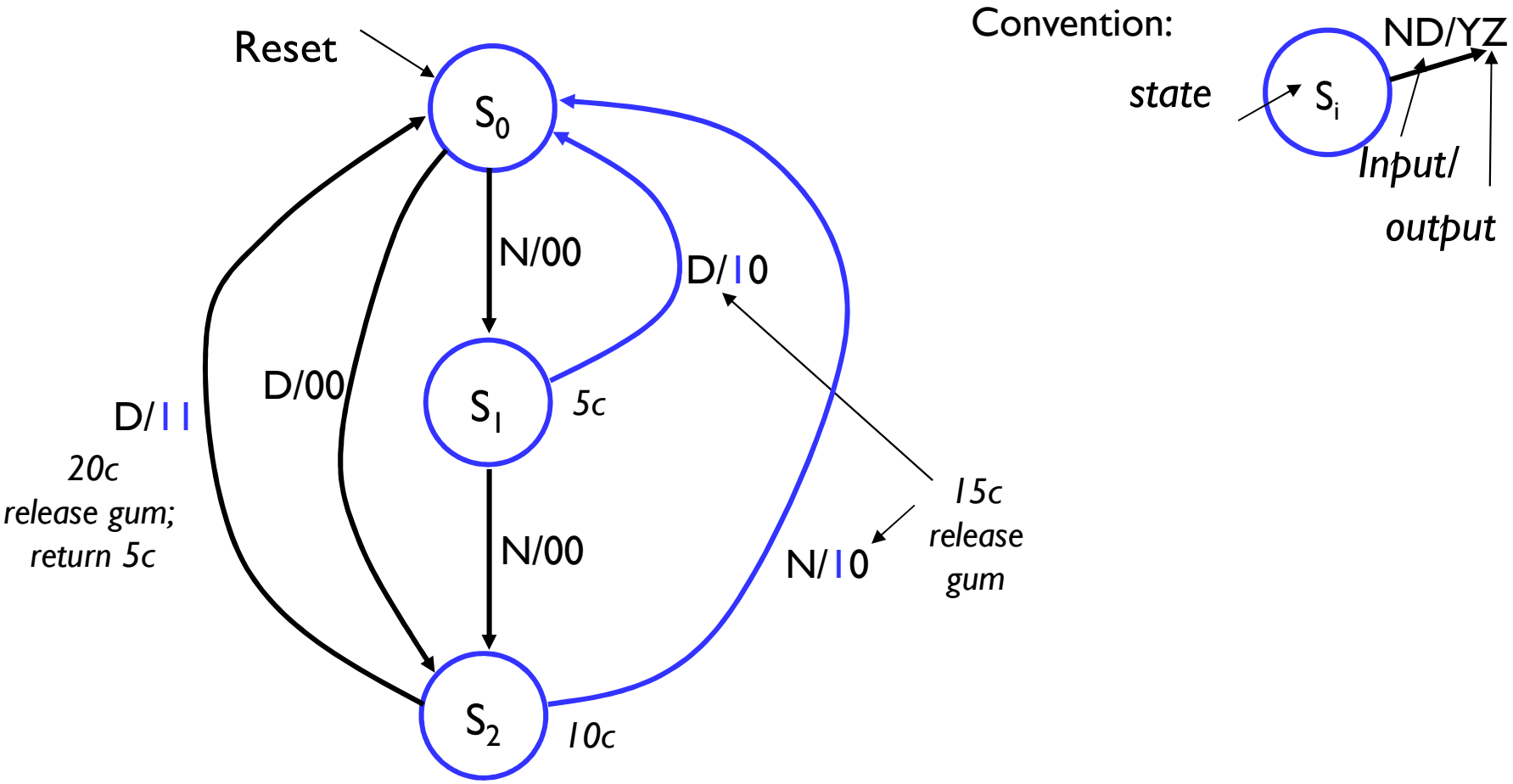


Convention:



Requires 5 states

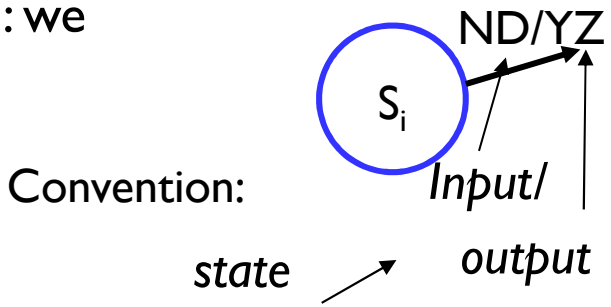
Step 2: State Diagram (Mealy)

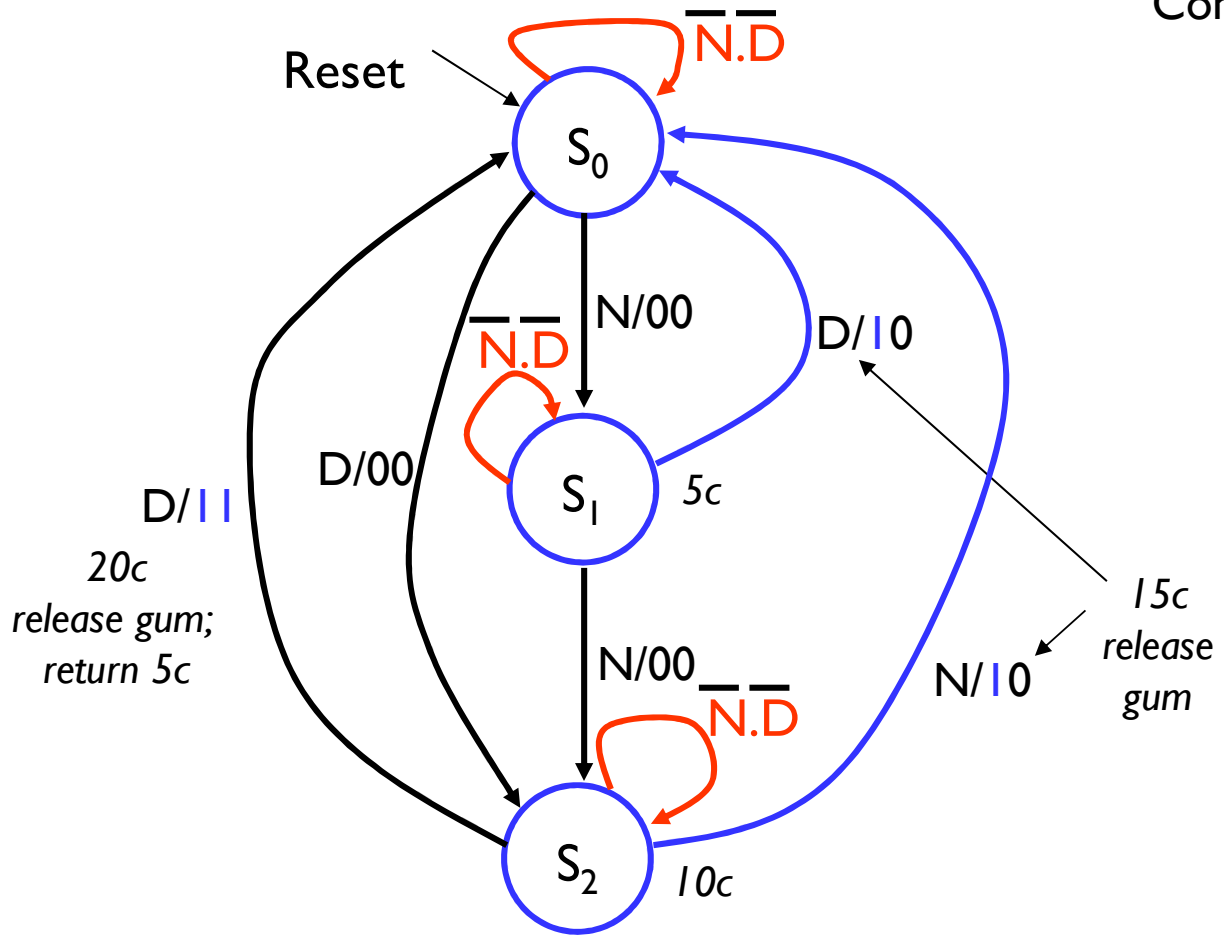


Requires 3 states

Step 2: State Diagram (Mealy)

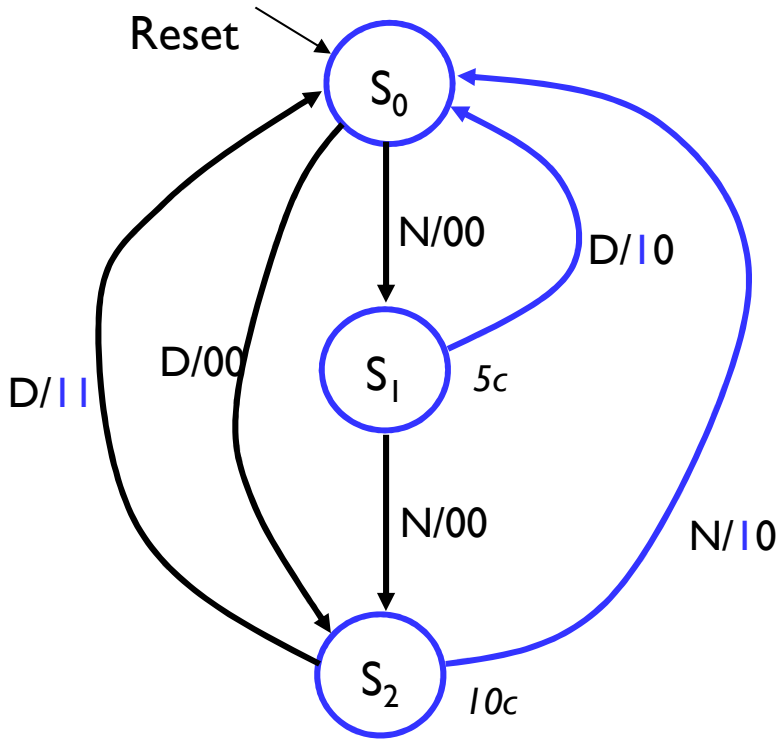
- The notation in the previous diagram was simplified: we assumed that when an input=0 there is no change.
- A more complete diagram would be:





Step 3: State table for Mealy machine

Present State	Inputs D N	Next States	Outputs Y Z
S_0	0 0	S_0	0 0
S_0	0 1	S_1	0 0
S_0	1 0	S_2	0 0
S_0	1 1	x	x x
S_1	0 0	S_1	0 0
S_1	0 1	S_2	0 0
S_1	1 0	S_0	1 0
S_1	1 1	x	x x
S_2	0 0	S_2	0 0
S_2	0 1	S_0	1 0
S_2	1 0	S_0	1 1
S_2	1 1	x	x x
S_3	0 0	x	x x
S_3	0 1	x	x x
S_3	1 0	x	x x
S_3	1 1	x	x x

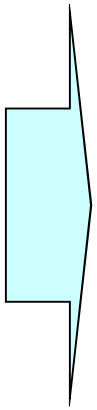


Step 4: State Encoding

- Three states requires 2 flip-flops: A and B
- Use the following encoding:
  - $S_0 = 00$
  - $S_1 = 01$
  - $S_2 = 10$
- Encoded state table

Encoded state table

Present State	Inputs D N	Next States	Outputs Y Z
S_0	0 0	S_0	0 0
S_0	0 1	S_1	0 0
S_0	1 0	S_2	0 0
S_0	1 1	x	x x
S_1	0 0	S_1	0 0
S_1	0 1	S_2	0 0
S_1	1 0	S_0	1 0
S_1	1 1	x	x x
S_2	0 0	S_2	0 0
S_2	0 1	S_0	1 0
S_2	1 0	S_0	1 1
S_2	1 1	x	x x
S_3	0 0	x	x x
S_3	0 1	x	x x
S_3	1 0	x	x x
S_3	1 1	x	x x



Present State A B	Inputs D N	Next States A^+ B^+	Outputs Y Z
0 0	0 0	0 0	0 0
0 0	0 1	0 1	0 0
0 0	1 0	1 0	0 0
0 0	1 1	x	x x
0 1	0 0	0 1	0 0
0 1	0 1	1 0	0 0
0 1	1 0	0 0	1 0
0 1	1 1	x	x x
1 0	0 0	1 0	0 0
1 0	0 1	0 0	1 0
1 0	1 0	0 0	1 1
1 0	1 1	x	x x
1 1	0 0	x	x x
1 1	0 1	x	x x
1 1	1 0	x	x x
1 1	1 1	x	x x

Step 5: Select type of flip-flop

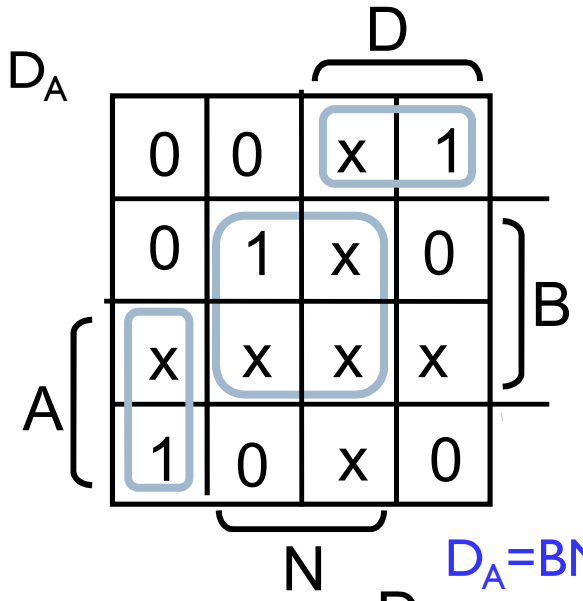
- We will use D flip-flops: A and B
- Other flip-flops are possible (see later):
  - JK flip-flop
  - SR flip-flop
  - T flip-flop

Step 6: Derive the inputs to the flip-flops and output equations

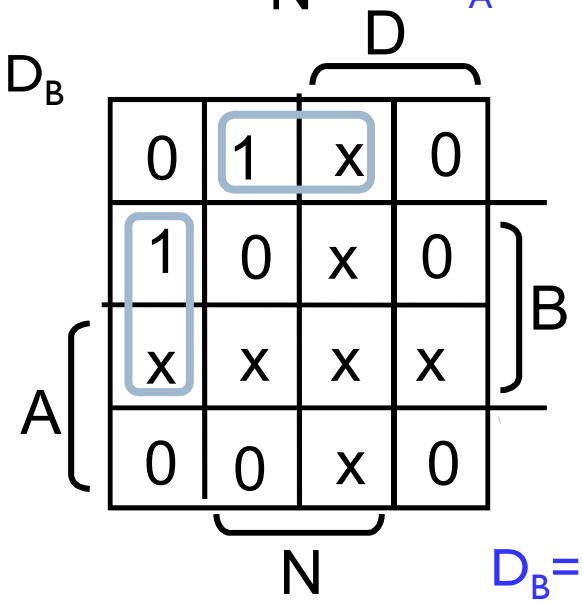
- The combinational circuits can be implemented in a variety of ways:
  - Minimized SOP
  - Decoders and OR gates
  - Multiplexers
- Let's use the minimized SOP:
  - Use K-maps for optimization

Input equations for DA and DB

Present State A B	Inputs D N	Next States A^+ B^+	Outputs Y Z
0 0	0 0	0 0	0 0
0 0	0 1	0 1	0 0
0 0	1 0	1 0	0 0
0 0	1 1	x	x x
0 1	0 0	0 1	0 0
0 1	0 1	1 0	0 0
0 1	1 0	0 0	1 0
0 1	1 1	x	x x
1 0	0 0	1 0	0 0
1 0	0 1	0 0	1 0
1 0	1 0	0 0	1 1
1 0	1 1	x	x x
1 1	0 0	x	x x
1 1	0 1	x	x x
1 1	1 0	x	x x
1 1	1 1	x	x x



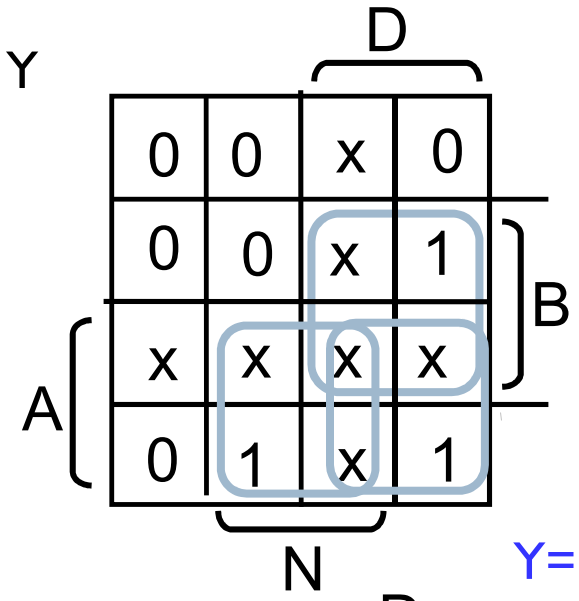
$D_A = BN + \bar{A}\bar{B}D + A\bar{N}\bar{D}$



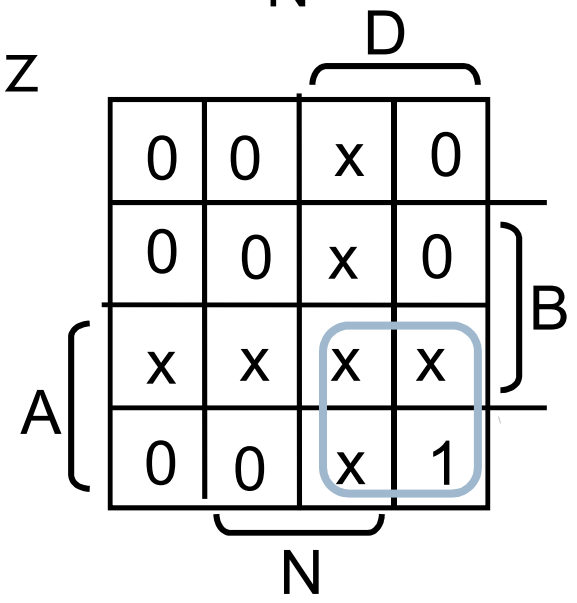
$D_B = B\bar{N}\bar{D} + \bar{A}BN$

Output equations

Present State A B	Inputs N D	Next States A^+ B^+	Outputs Y Z
0 0	0 0	0 0	0 0
0 0	0 1	0 1	0 0
0 0	1 0	1 0	0 0
0 0	1 1	x	x x
0 1	0 0	0 1	0 0
0 1	0 1	1 0	0 0
0 1	1 0	0 0	1 0
0 1	1 1	x	x x
1 0	0 0	1 0	0 0
1 0	0 1	0 0	1 0
1 0	1 0	0 0	1 1
1 0	1 1	x	x x
1 1	0 0	x	x x
1 1	0 1	x	x x
1 1	1 0	x	x x
1 1	1 1	x	x x



Y=BD+AN+AD



Z=AD

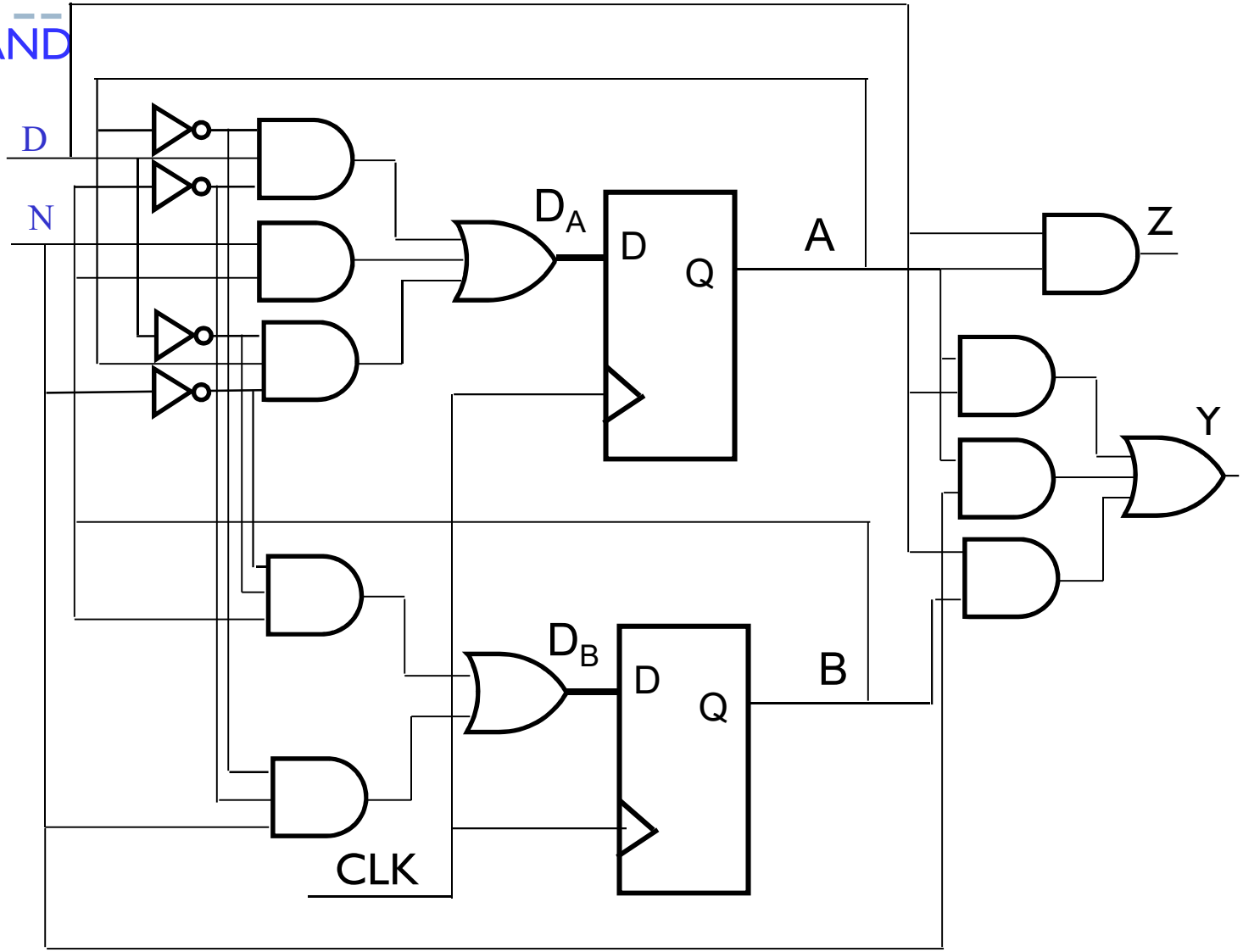
Step 7: Circuit

$D_A = BN + \bar{A}BD + A\bar{N}\bar{D}$

$D_B = B\bar{N}\bar{D} + \bar{A}BN$

$Y = BD + AN + AD$

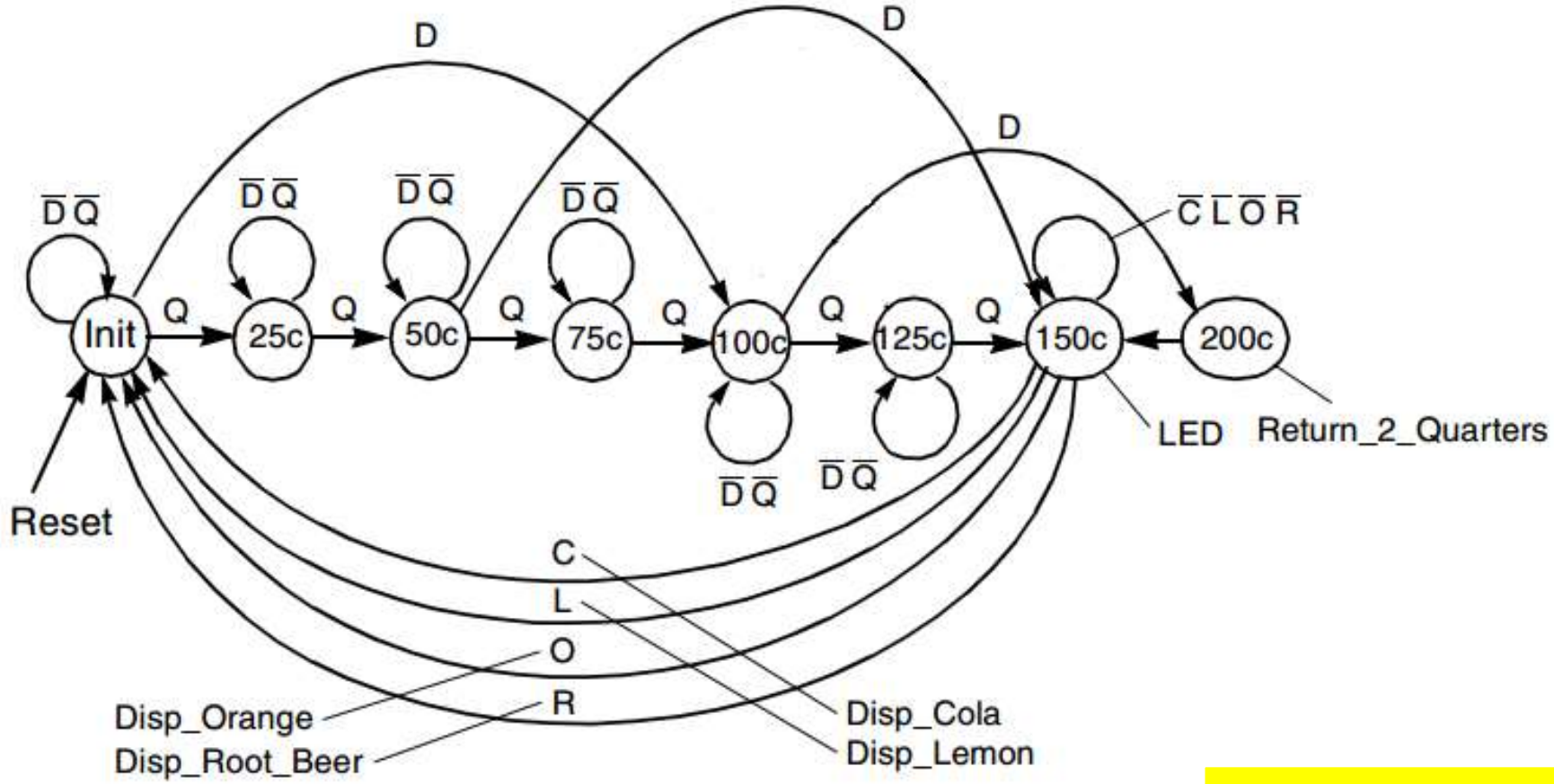
$Z = AD$



FSM Design Example: Vending Machine

- Find the state machine diagram for the following electronic vending machine specification.
- The vending machine sells soda for $1.50 per bottle.
- The machine accepts only $1 bills (D) and 25 cent coins (Q).
- When the sum of money is greater than $1.50 , i.e., two $1 bills, the machine returns change in the coin return (two quarters).
- When $1.50 has been paid, the machine lights and LED to indicate that a soda flavor may be selected.
- The choices by pushbutton are C (Cola), L (lemon soda), O (orange soda), and R (Root beer).
- When one pushbutton is pushed, the selected soda is dispensed and the machine returns to its initial state.

FSM Design Example: Vending Machine



Is this specification complete?

Any Questions?