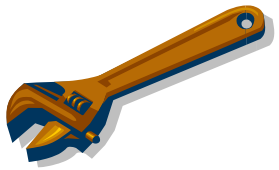
Finite State Machine Design Procedure

Overview

- Design of systems that input flip flops and combinational logic
- Specifications start with a word description
- Create a state table to indicate next states
- Convert next states and outputs to output and flip flop input equations
  - Reduce logic expressions using truth tables
- Draw resulting circuits.



Lots of opportunities for interesting design

Concept of the State Machine

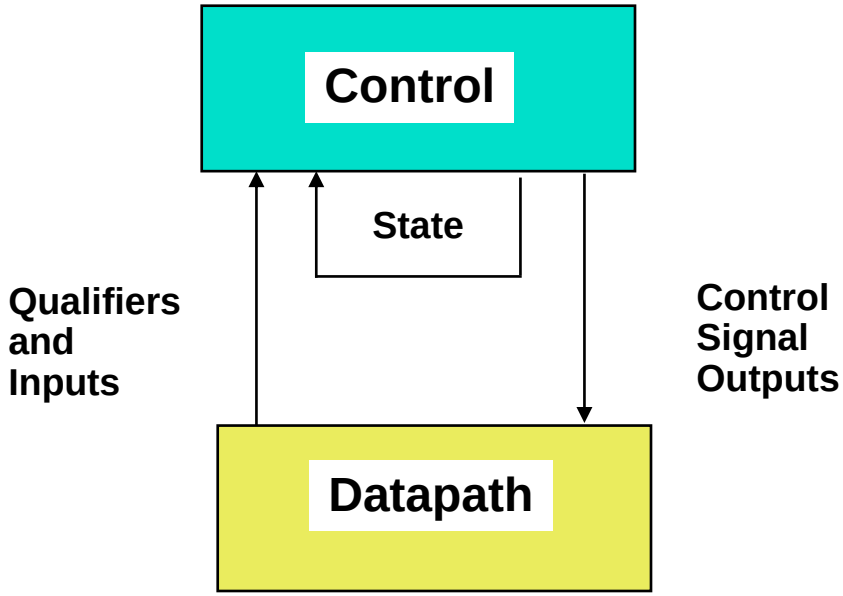
Computer Hardware = Datapath + Control

Registers
Combinational Functional Units (e.g., ALU)
Busses

Qualifiers

Control

FSM generating sequences of control signals
Instructs datapath what to do next



Designing Finite State Machines

- Specify the problem with words
  - *(e.g. Design a circuit that detects three consecutive 1 inputs)*
- Assign binary values to states
- Develop a state table
- Use K-maps to simplify expressions
  - Flip flop input equations and output equations
- Create appropriate logic diagram
  - Should include combinational logic and flip flops

Example: Detect 3 Consecutive 1 inputs

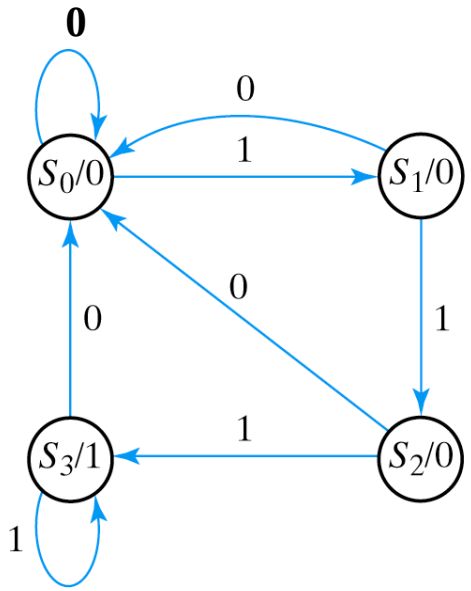


Fig. 5-24 State Diagram for Sequence Detector

- State S_0 : zero 1s detected
- State S_1 : one 1 detected
- State S_2 : two 1s detected
- State S_3 : three 1s detected

- Note that each state has 2 output arrows
- Two bits needed to encode state

State Table for Sequence Detector

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

- **Sequence of outputs, inputs, and flip flop states enumerated in state table**
- **Present state indicates current value of flip flops**
- **Next state indicates state after next rising clock edge**
- **Output is output value on current clock edge**

- **$S_0 = 00$**
- **$S_1 = 01$**
- **$S_2 = 10$**
- **$S_3 = 11$**

Finding Expressions for Next State and Output Value

- Create K-map directly from state table (3 columns = 3 K-maps)
- Minimize K-maps to find SOP representations
- Separate circuit for each next state and output value

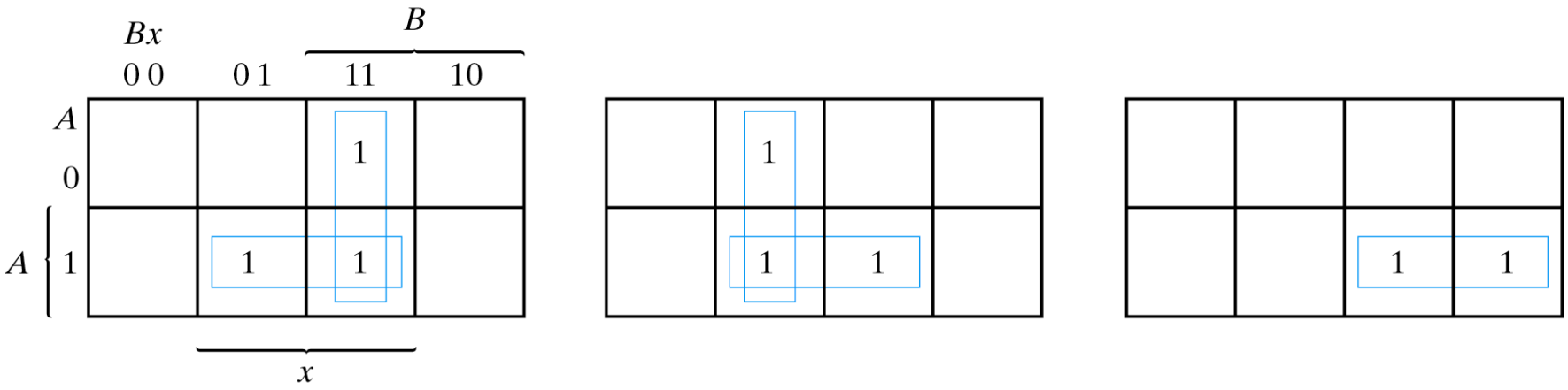


Fig. 5-25 Maps for Sequence Detector

Circuit for Consecutive 1s Detector

- Note location of state flip flops
- Output value (y) is function of state
- This is a Moore machine.

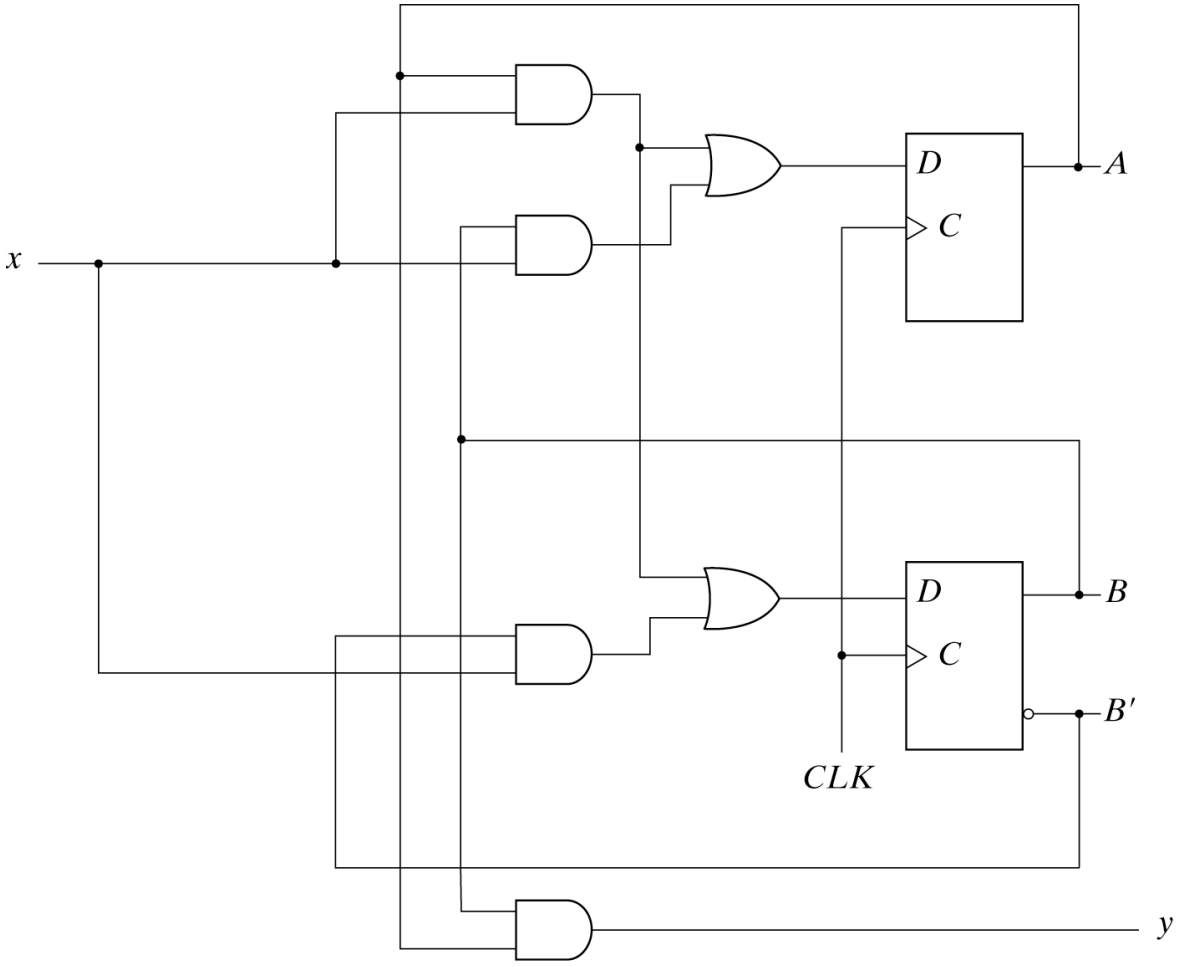
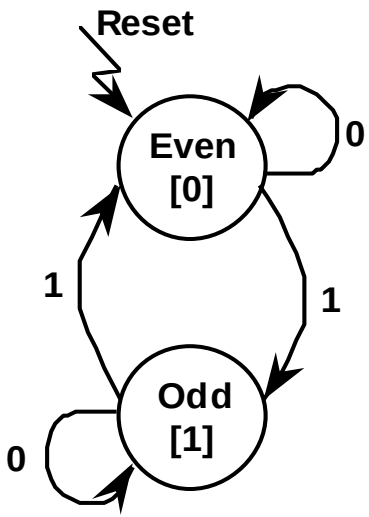


Fig. 5-26 Logic Diagram of Sequence Detector

Concept of the State Machine

Example: Odd Parity Checker

Assert output whenever input bit stream has odd # of 1's



State Diagram

Present State	Input	Next State	Output
Even	0	Even	0
Even	1	Odd	0
Odd	0	Odd	1
Odd	1	Even	1

Symbolic State Transition Table

Present State	Input	Next State	Output
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1

Encoded State Transition Table

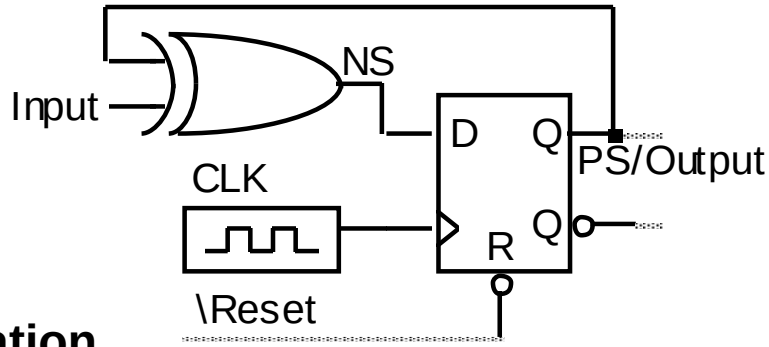
- **Note: Present state and output are the same value**
  - **Moore machine**

Concept of the State Machine

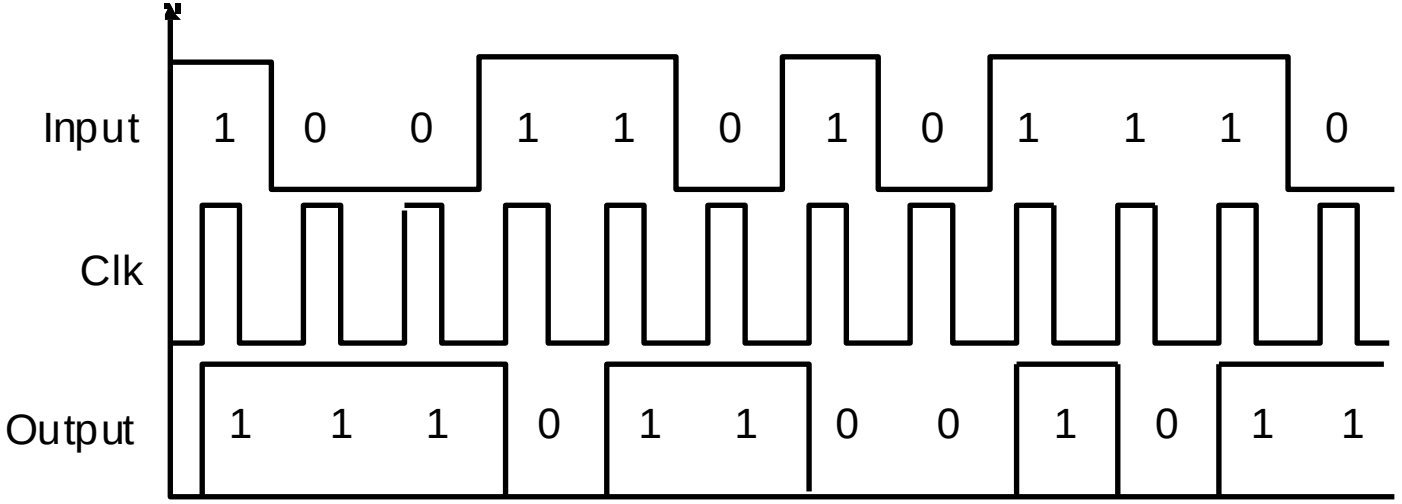
Example: Odd Parity Checker

Next State/Output Functions

NS = PS xor PI; OUT = PS



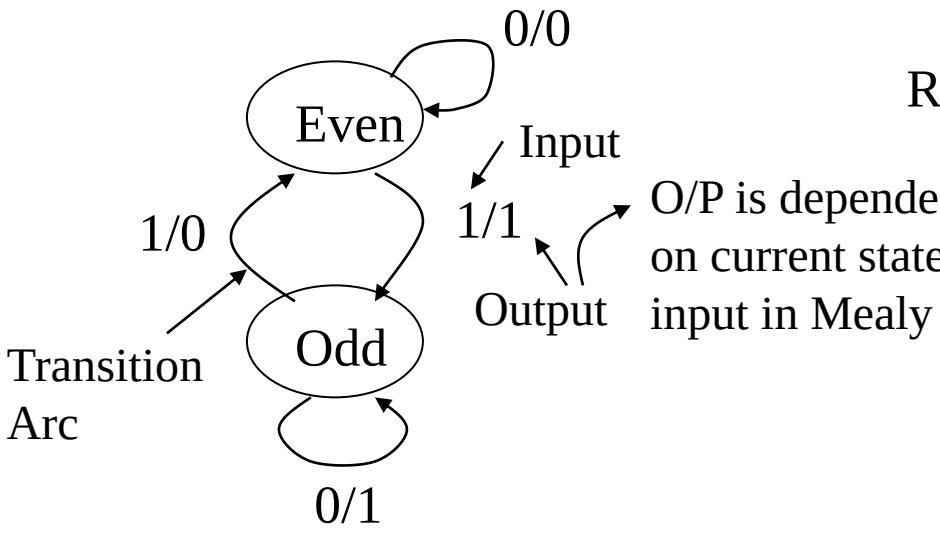
D FF Implementation



Timing Behavior: Input 1 0 0 1 1 0 1 0 1 1 1 0

Mealy and Moore Machines

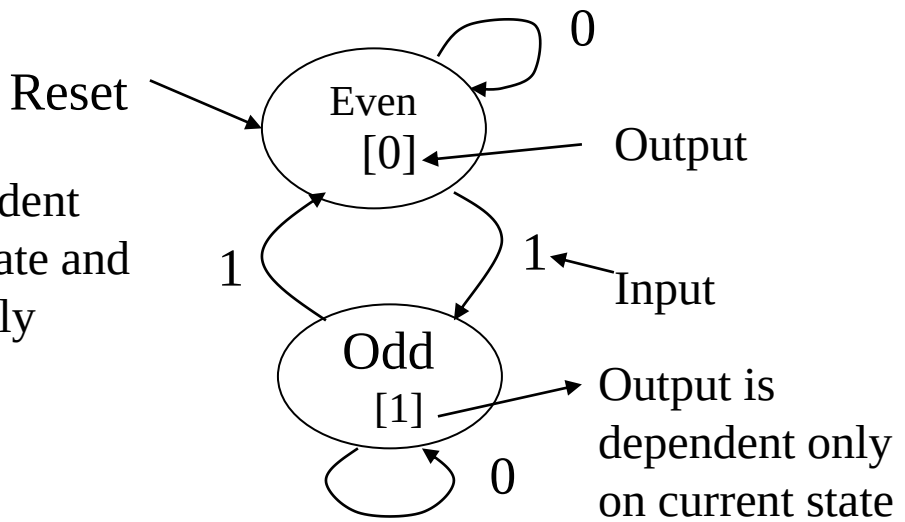
Solution 1: (Mealy)



Mealy Machine: Output is associated with the state transition

- Appears before the state transition is completed (by the next clock pulse).

Solution 2: (Moore)



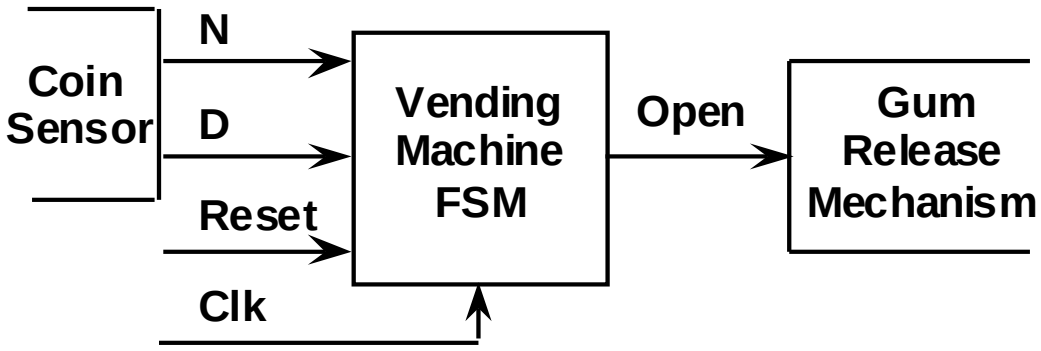
Moore Machine: Output is associated with the state

-Appears after the state transition takes place.

Vending Machine FSM

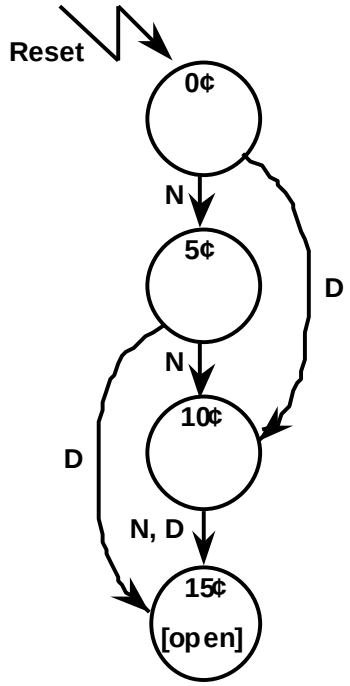
Step 1. Specify the problem

- Deliver package of gum after 15 cents deposited
- Single coin slot for dimes, nickels
- No change
- Design the FSM using combinational logic and flip flops



Vending Machine FSM

State Diagram



Reuse states whenever possible

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

Symbolic State Table

Vending Machine FSM

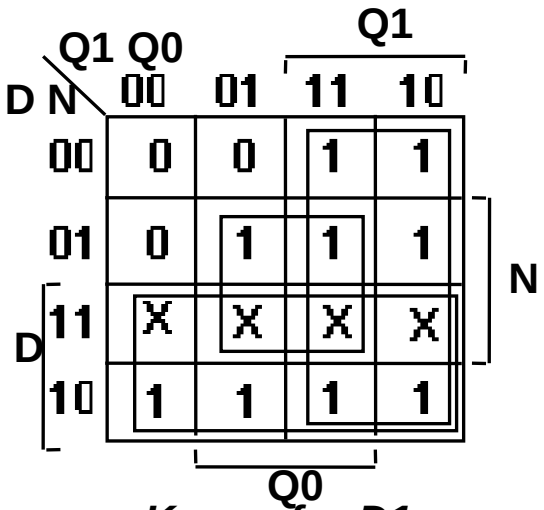
State Encoding

How many flip-flops are needed?

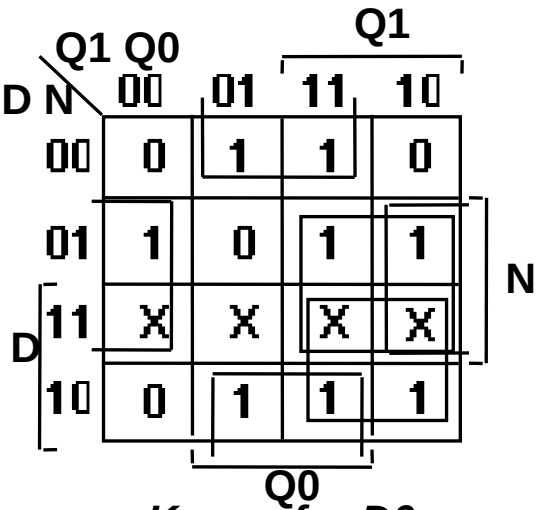
Present State Q_1	Present State Q_0	Inputs D	Inputs N	Next State D_1	Next State D_0	Output Open
0	0	0	0	0	0	0
		0	1	0	1	0
		1	0	1	0	0
		1	1	X	X	X
0	1	0	0	0	1	0
		0	1	1	0	0
		1	0	1	1	0
		1	1	X	X	X
1	0	0	0	1	0	0
		0	1	1	1	0
		1	0	1	1	0
		1	1	X	X	X
1	1	0	0	1	1	1
		0	1	1	1	1
		1	0	1	1	1
		1	1	X	X	X

Vending Machine FSM

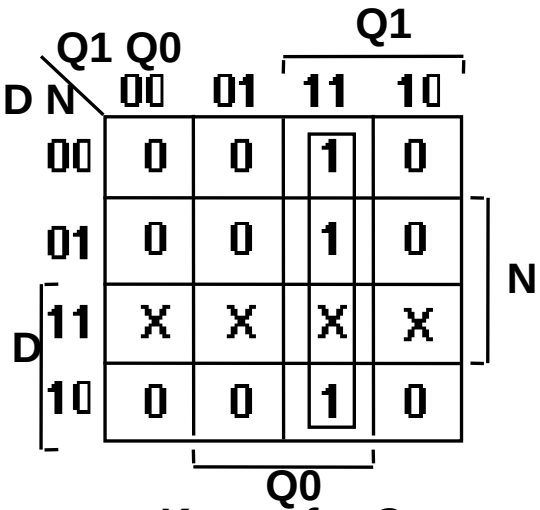
Determine F/F implementation



K-map for D1

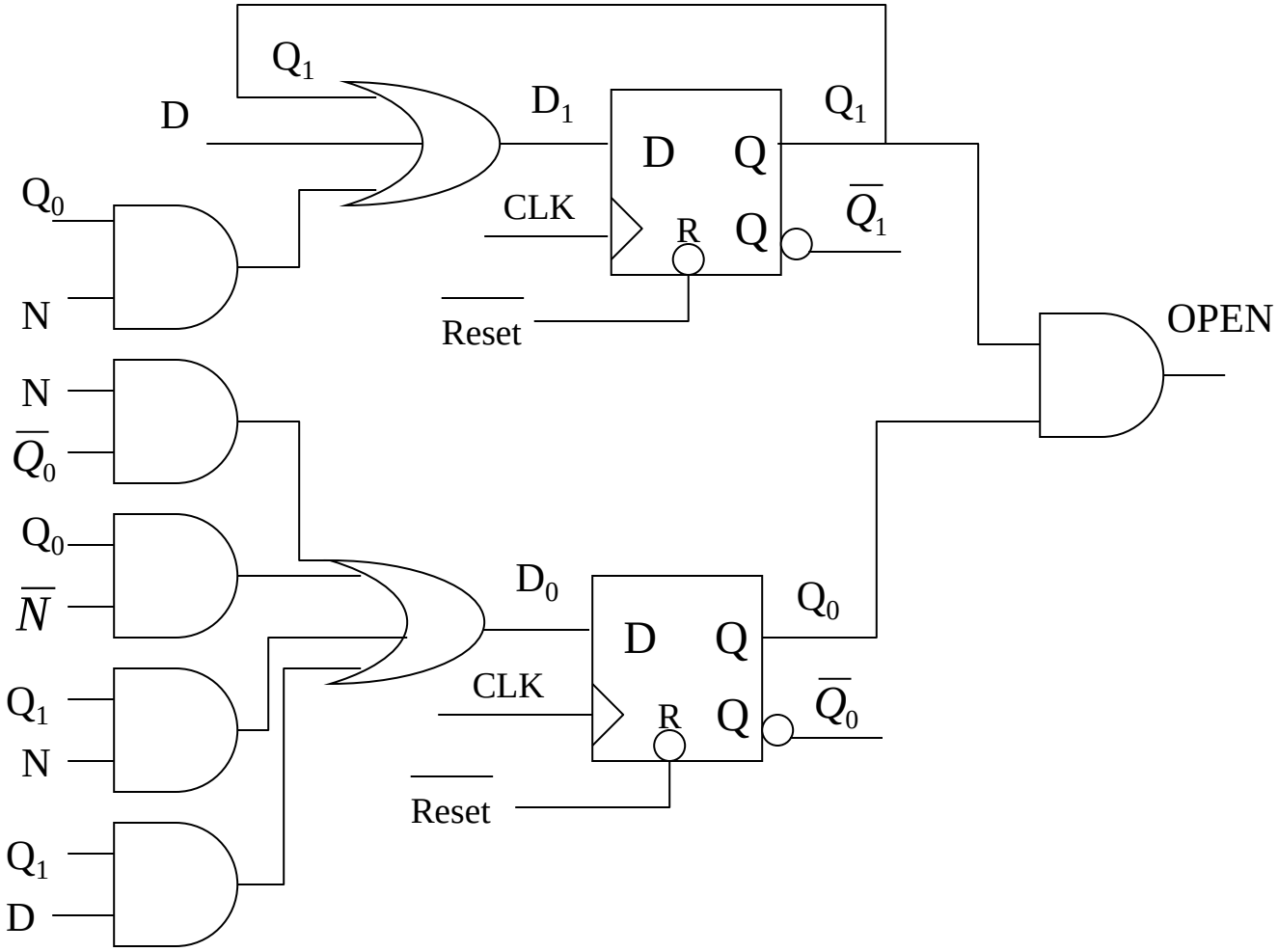


K-map for D0



K-map for Open

Minimized Implementation



Vending machine FSM implementation based on D flip-flops(Moore).

Summary

- Finite state machines form the basis of many digital systems
- Designs often start from clear specifications
- Develop state diagram and state table
- Optimize using combinational design techniques
- Mealy or Moore implementations possible
  - Can model approach using HDL.