Embedded System Design

Dr. Hamid Vakilzadian
Guest Professor
Department of Informatics
University of Hamburg

Course Outline

- Review of FSM Design
- Review of ROM/PAL implementation
- Datapath Design
- Controller Design
- VHDL Description
- Design Using Altera FPGA (UP1 System)
- Codesign and virtual design
- Hardware/Software Partitioning

Review of FSM Design Process

Combinational Logic CKTs

Output(s) of these circuits that is a function of the present inputs.

Sequential CKTs

Output(s) of these circuits is a function of the current inputs and their past history. Memory elements are needed to store the past history. These circuits have feedbacks.

Basic Design Approach of Sequential CKTs (FSM)

Six Step Process

1. **Understand the statement of the Specification**

2. **Obtain an abstract representation of the FSM**

   - **Using State Diagram or**
   - **Using ASM Chart**
     and develop a Transition Table

3. **Perform a state minimization**

4. **Make state assignment**

5. **Choose FF types to implement FSM state register**

6. **Implement the FSM**

Basic Design Approach

Example: **Vending Machine FSM**

General Machine Concept:

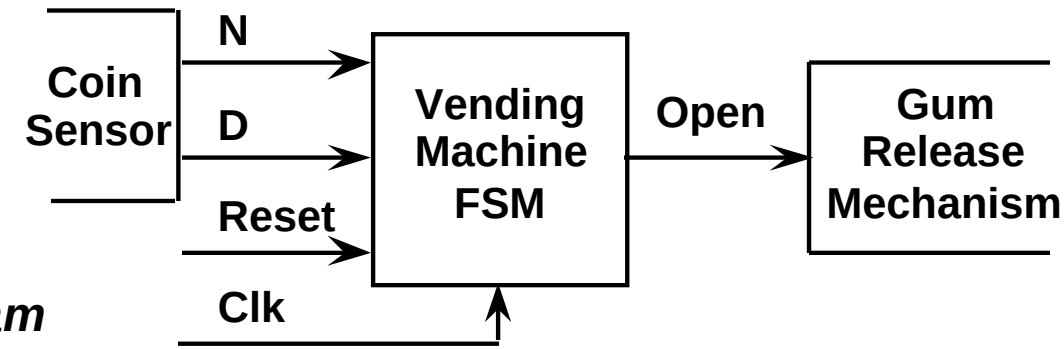
- **Deliver a package of gum after 15 cents deposited**
- **Single coin slot for dimes, nickels**
- **No change**

Step 1. *Understand the problem:*

Specification may not be complete:

Ex: What happens if penny is inserted or what happens after gum is delivered? Make reasonable assumptions!

Draw a picture!



Block Diagram

Vending Machine Example

Step 2. ***Map into more suitable abstract representation***

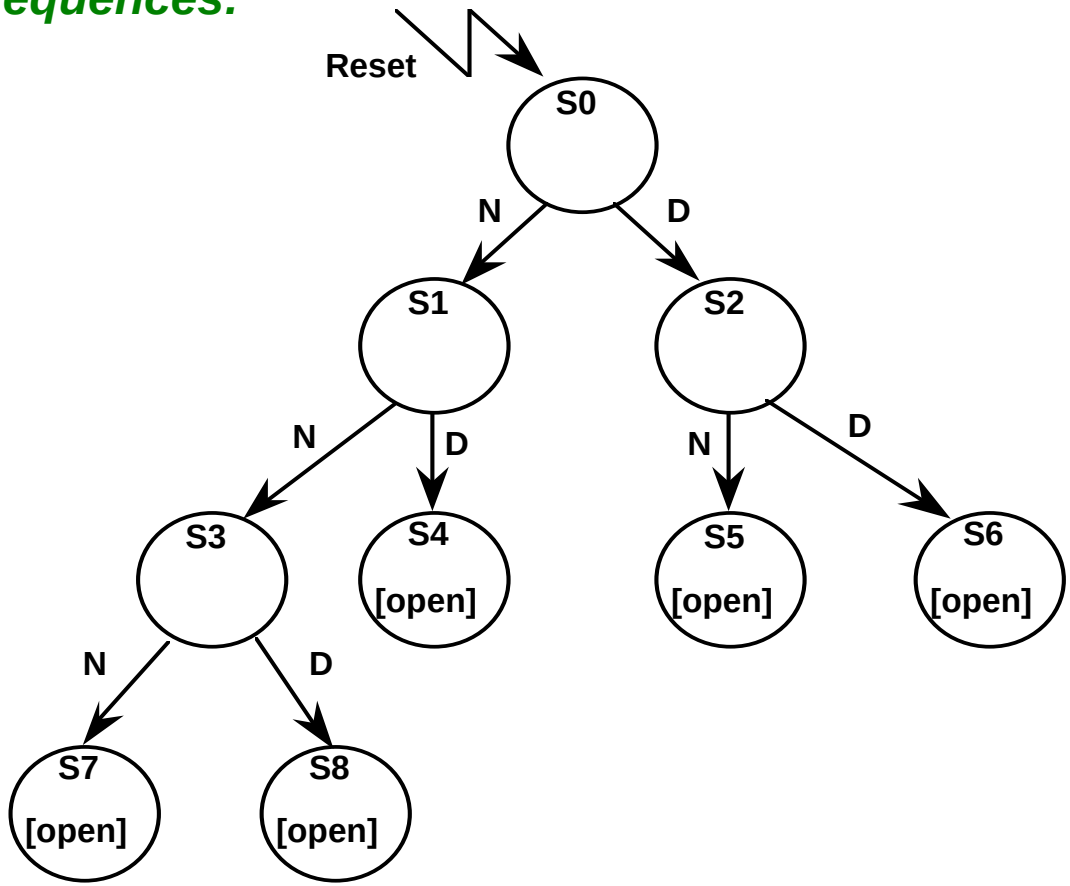
Tabulate typical input sequences:

a. three nickels
b. nickel, dime
c. dime, nickel
d. two dimes
e. two nickels, dime

Draw state diagram:

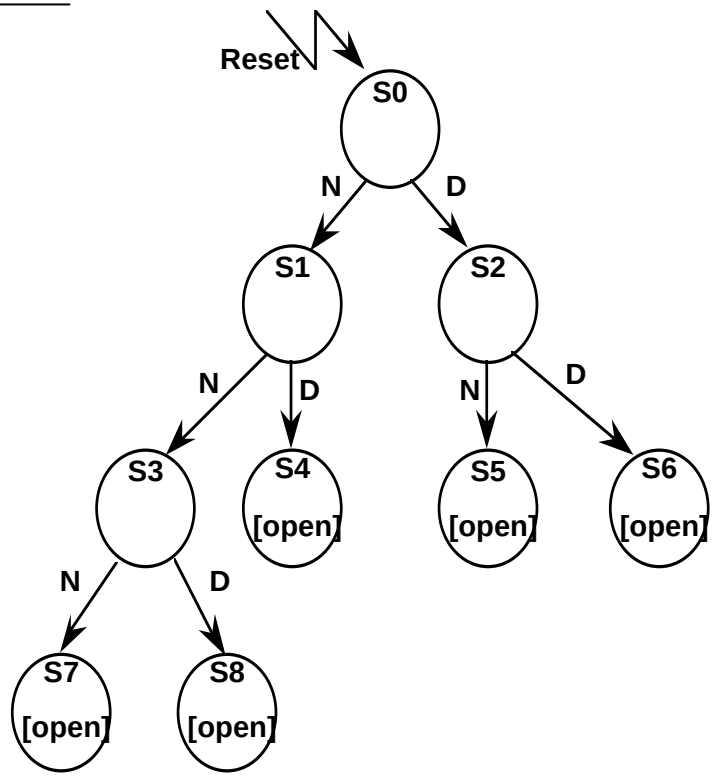
Inputs: N, D, reset

Output: open

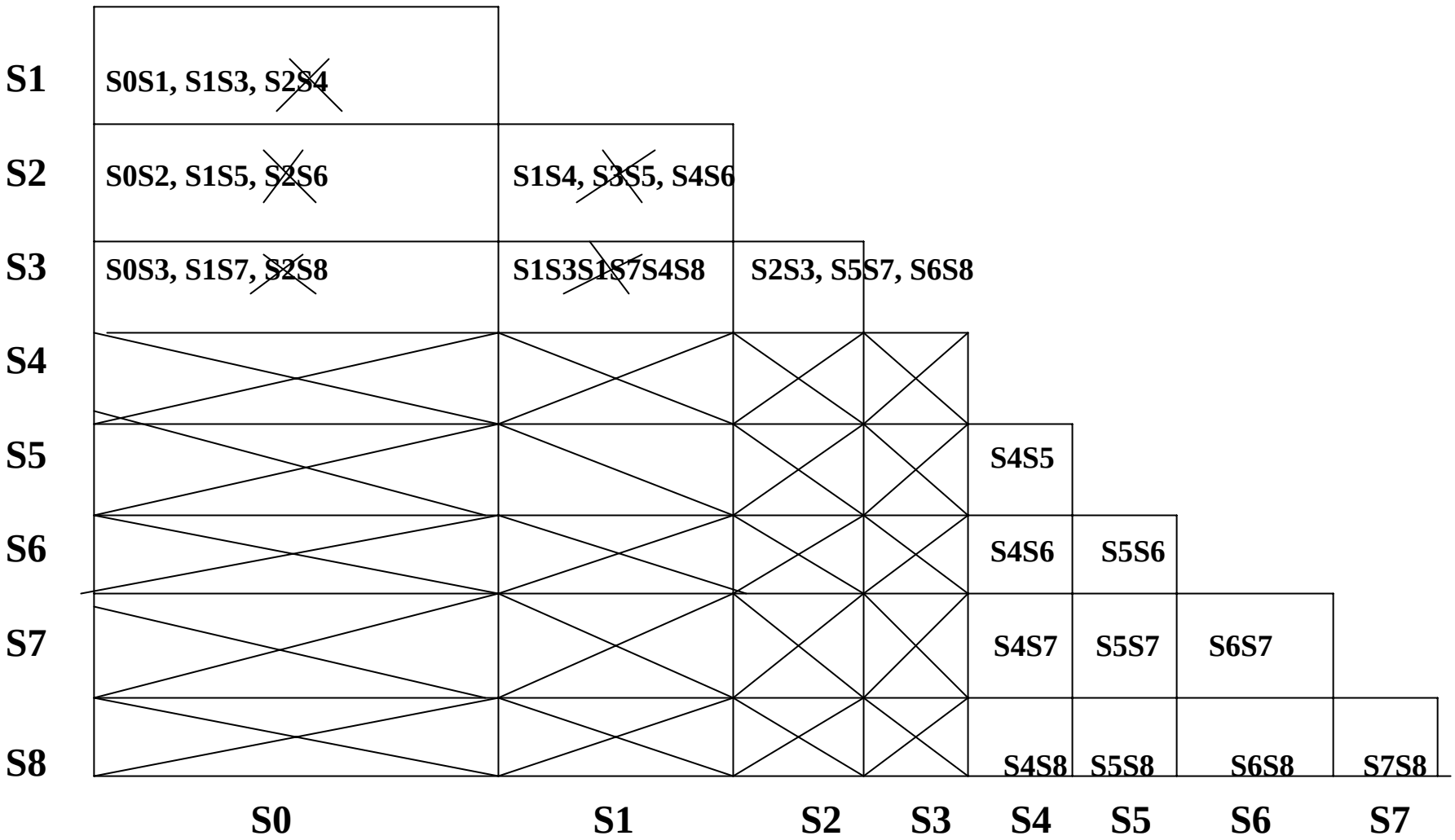


State Diagram Representation

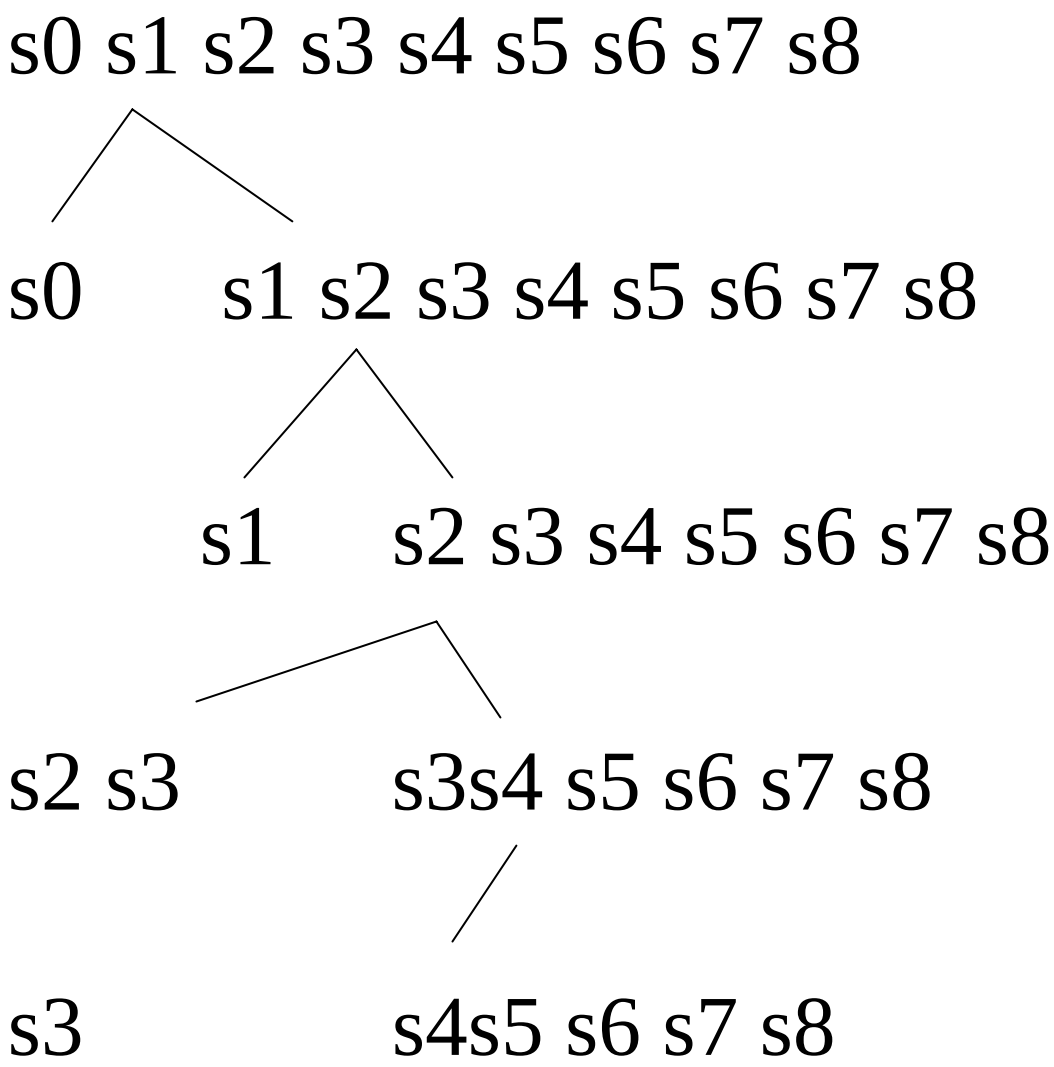
Present State	DN 00	01	10	11	Output
S0	S0	S1	S2	X	0
S1	S1	S3	S4	X	0
S2	S2	S5	S6	X	0
S3	S3	S7	S8	X	0
S4	S4	X	X	X	1
S5	S5	X	X	X	1
S6	S6	X	X	X	1
S7	S7	X	X	X	1
S8	S8	X	X	X	1



State Transition Table



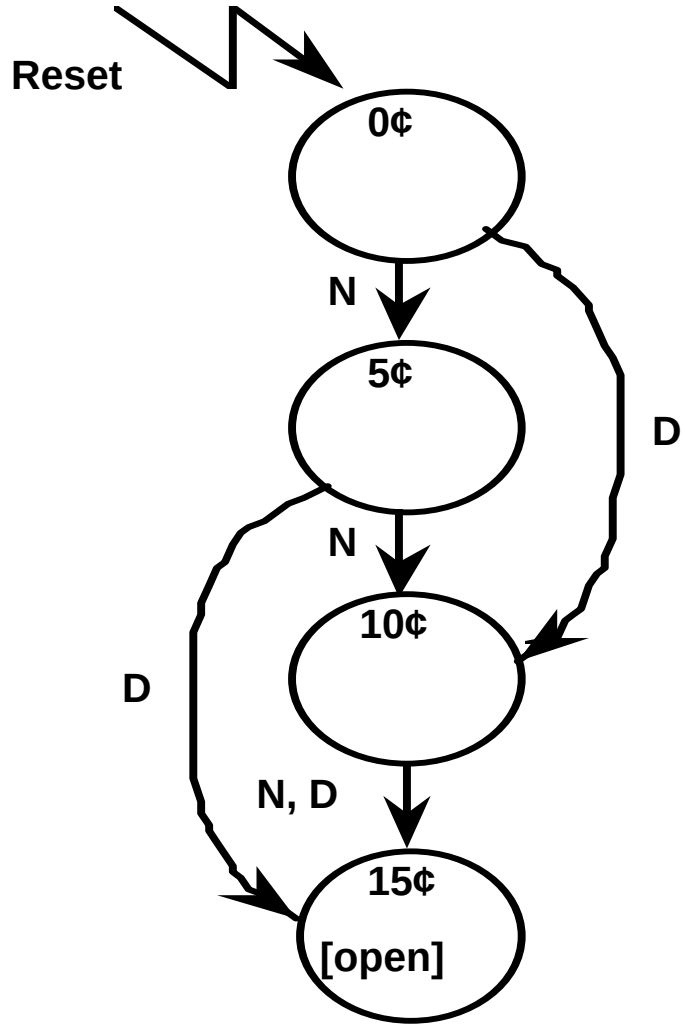
State Minimization



States needed: A=s0, B=s1, C=s2s3, D=s4s5s6s7s8

Vending Machine Example

Step 3: State Minimization



State Assignment

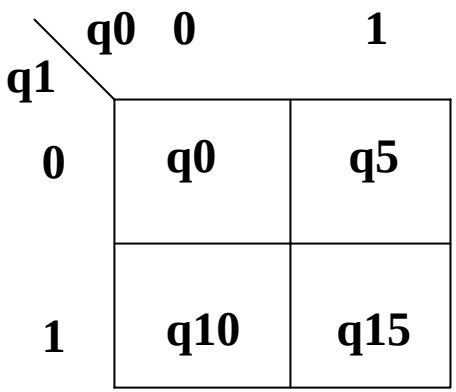
- **Rule 0** (Fundamental Mode CKTs Only!)
  - States which are connected on the flow graph must be adjacent or at least effectively adjacent.

- **Rule 1**
  - Under a given input condition, states which go to the same next state should be adjacent. Priority should be given to states having common next states for more number of inputs.

State Assignment

- **Rule 2**
  - For each state, those states which are the “next state” entries for adjacent inputs should be adjacent
- **Rule 3**
  - States with the same output class should be adjacent

State	00	01	11	10	Rule II
q0	q0	q5	x	q10	q0q5, q0q10
q5	q5	q10	x	q15	q5q10, q5q15
q10	q10	q15	x	q15	q10q15, q10q15
q15	q15	x	x	x	
Rule I				q5q10	



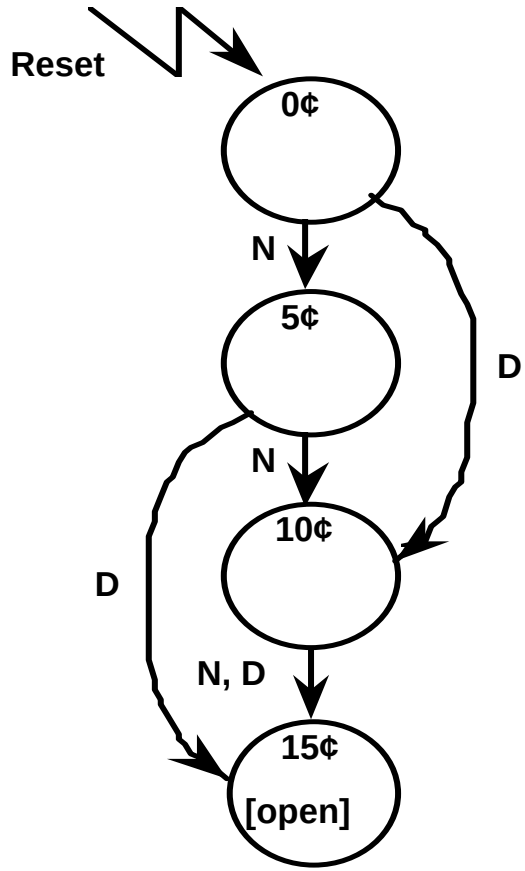
State Encoding

q1q0	
00	0 Cent
01	5 Cents
10	10 Cents
11	15 Cents

Vending Machine Example

Step 4: State 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



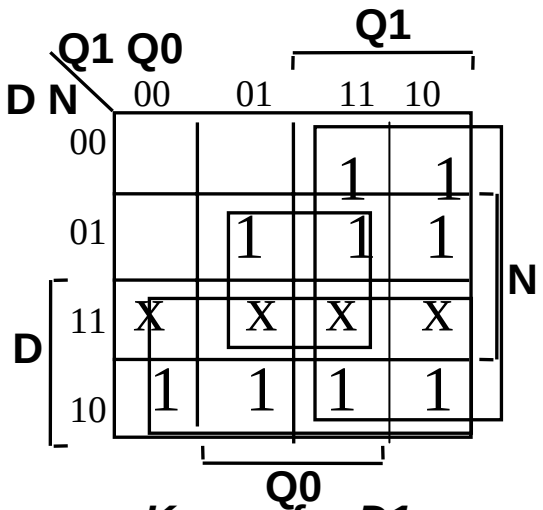
Vending Machine Example

Step 5. ***Choosing FF for Implementation***

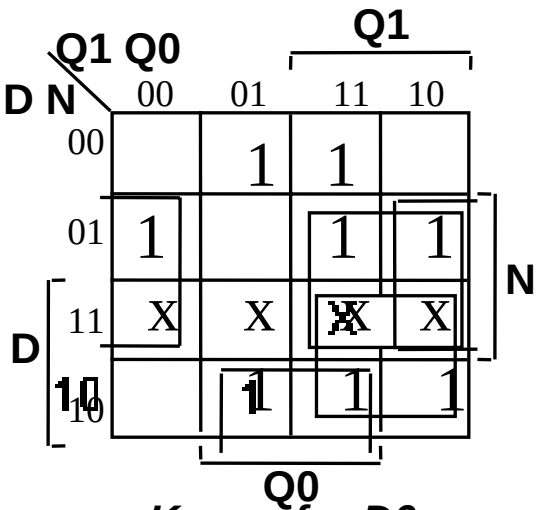
D, J-K FF, T ?

Present State Q_1	Present State Q_0	Inputs D	Inputs N	Next State D_1	Next State D_0	J_1	K_1	J_0	K_0
0	0	0	0	0	0	0	X	0	X
		0	1	0	1	0	X	1	X
		1	0	1	0	1	X	0	X
		1	1	X	X	X	X	X	X
0	1	0	0	0	1	0	X	X	0
		0	1	1	0	1	X	X	1
		1	0	1	1	1	X	X	0
		1	1	X	X	X	X	X	X
1	0	0	0	1	0	X	0	0	X
		0	1	1	1	X	0	1	X
		1	0	1	1	X	0	1	X
		1	1	X	X	X	X	X	X
1	1	0	0	1	1	X	0	X	0
		0	1	1	1	X	0	X	0
		1	0	1	1	X	0	X	0
		1	1	X	X	X	X	X	X

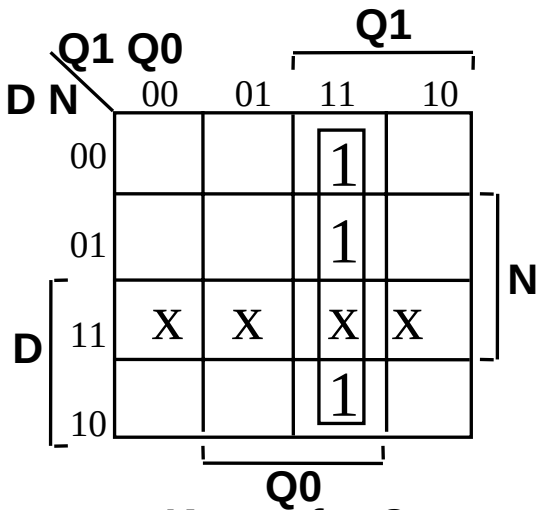
Remapped encoded state transition table



K-map for D1



K-map for D0

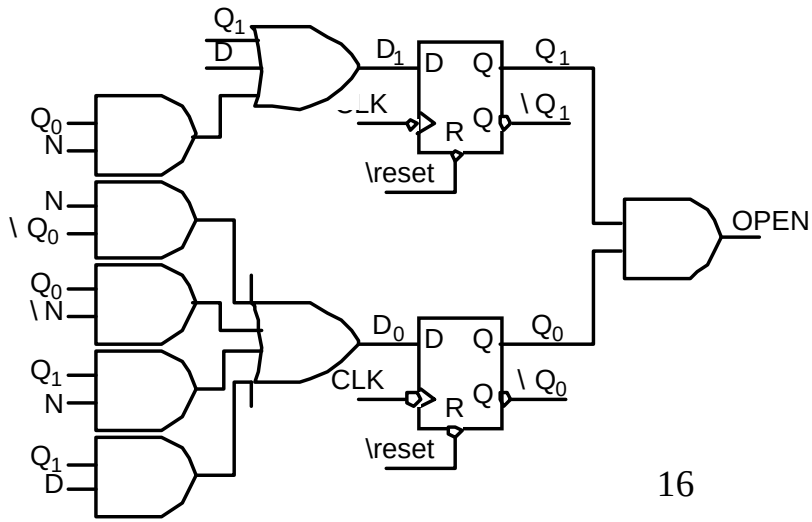


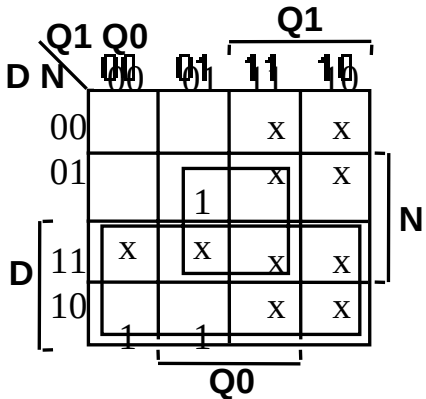
K-map for Open

D1 = Q1 + D + Q0 N

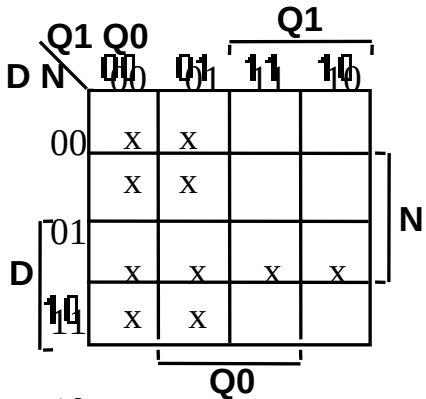
D0 = N Q0 + Q0 N + Q1 N + Q1 D

OPEN = Q1 Q0





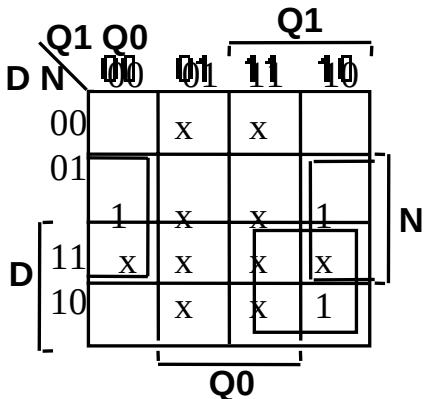
K-map for J1



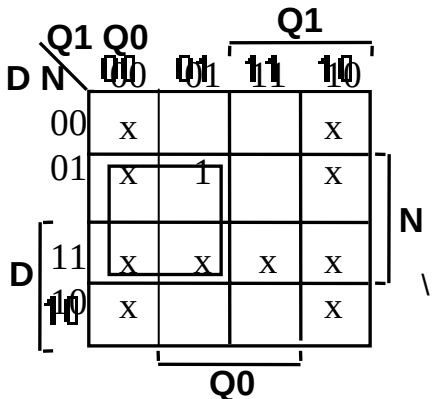
K-map for K1

J1 = D + Q0 N

K1 = 0



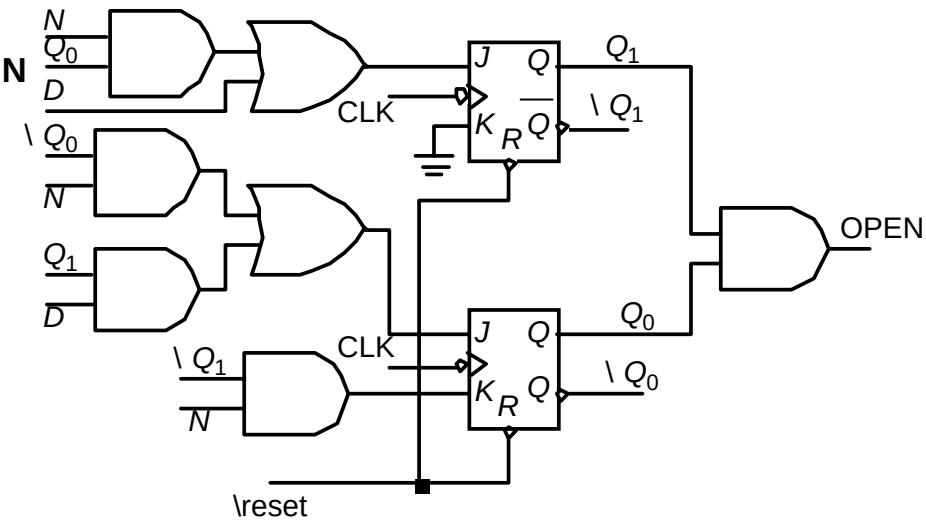
K-map for J0



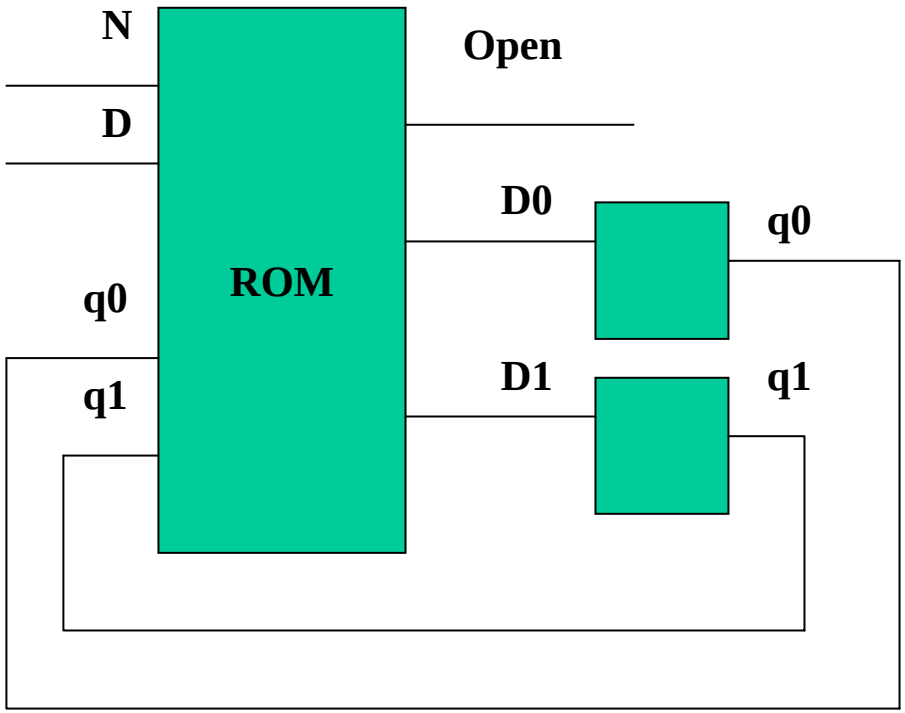
K-map for K0

J0 = Q0 N + Q1 D

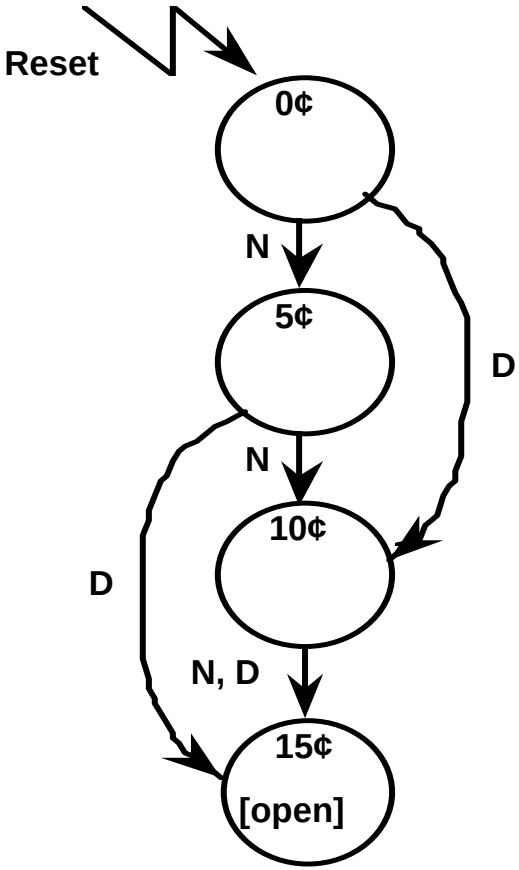
K0 = Q1 N



ROM Implementation



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



State Encoding

q1q0	
00	0 Cent
01	5 Cents
10	10 Cents
11	15 Cents

Row	D_1	D_0	Open
0	0	0	0
1	0	1	0
2	1	0	0
3	0	0	0
4	0	1	0
5	1	0	0
6	1	1	0
7	0	0	0
8	1	0	0
9	1	1	0
10	1	1	0
11	0	0	0
12	1	1	1
13	1	1	1
14	1	1	1
15	0	0	0

ROM Contents, **ROM Size: 16 rows and 3 columns**

ROM Realization doesn't need state assignment!

Module: Vending_machine;
INPUTS: D, N, Reset.
Outputs: Open

1. =>(D'N', D'N, DN', DN)/(1, 2, 3, 1) – Reset State
2. =>(D'N', D'N, DN', DN)/(2, 3, 4, 1) _ q5 State
3. =>(D'N', D'N, DN', DN)/(3, 4, 4, 1) _ q10 state
4. Open = 1 =>(1) _ q15 State

Endsequence
Controlreset(1).
End

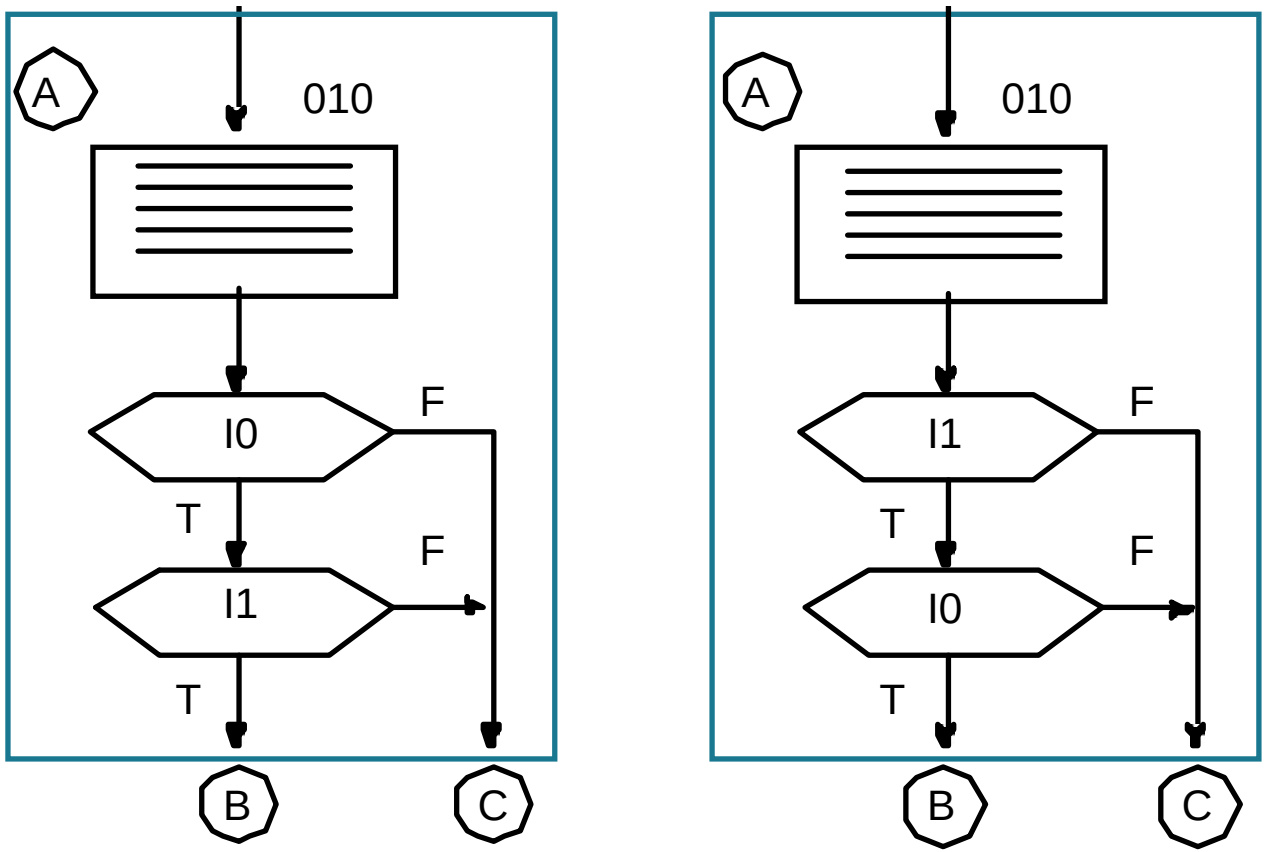
ASM Notation

Condition Boxes:

Ordering has no effect on final outcome

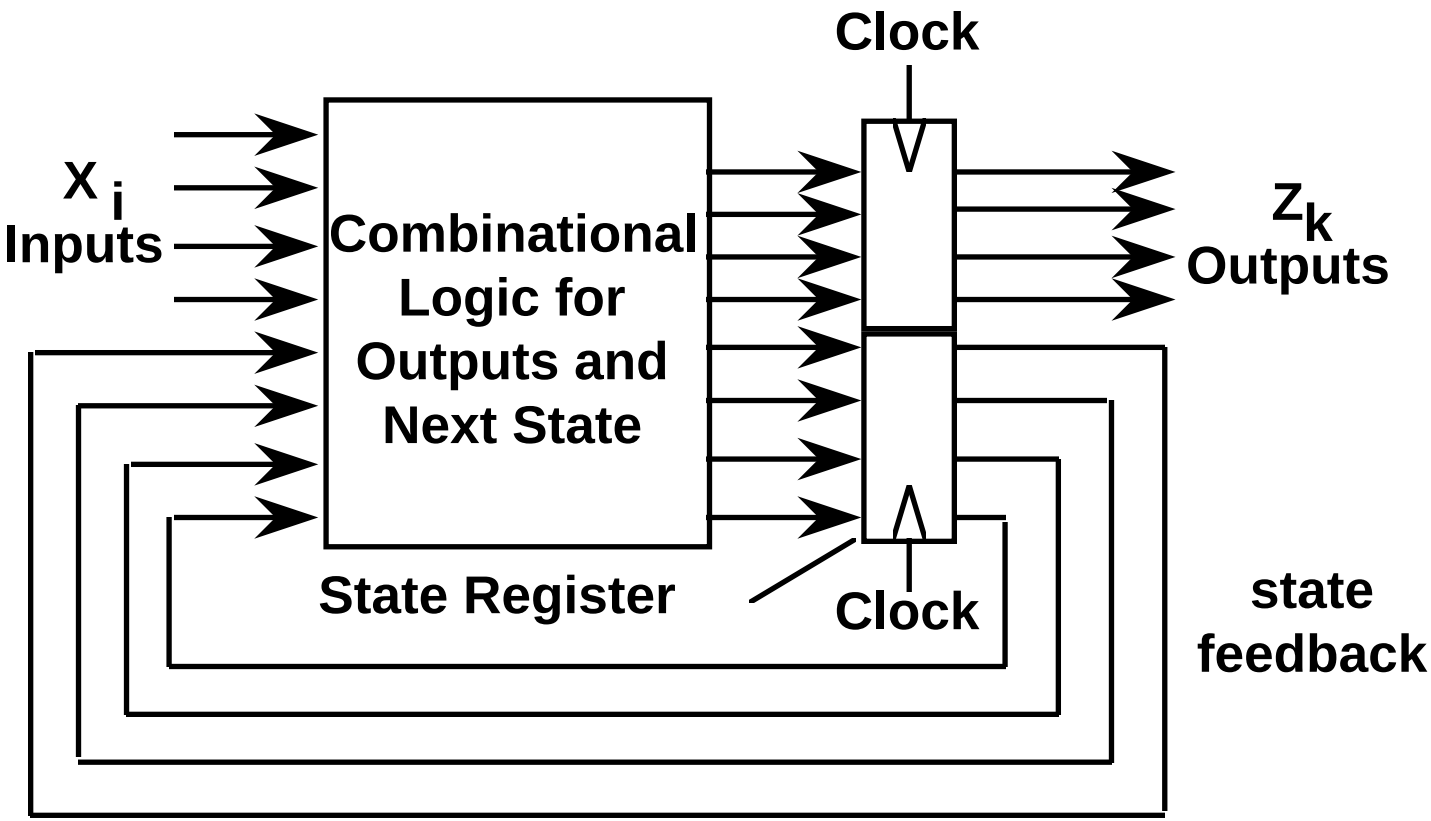
Equivalent ASM charts:

A exits to B on (I0 • I1) else exit to C



Moore and Mealy Machines

Synchronous Mealy Machine



latched state AND outputs

avoids glitchy outputs!