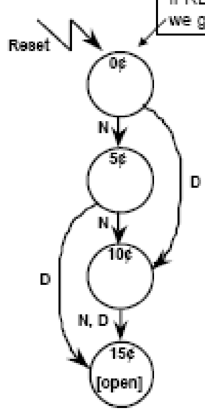
Building a State Machine

Given the following fully specified state machine, we will come up with the logic to drive it. The vending machine below has four states. One for each possible status – 0, 5, 10, and 15 cents. Once fifteen cents is reached, the machine opens. Also, the machine assumes that the machine cannot be given a nickel and a dime at the same time.

Vending Machine Example

Digital Logic Design
Switching Logic

If RESET is asserted, we go here



VEND the item once at least $0.15 has been inserted using any combination of nickels and dimes

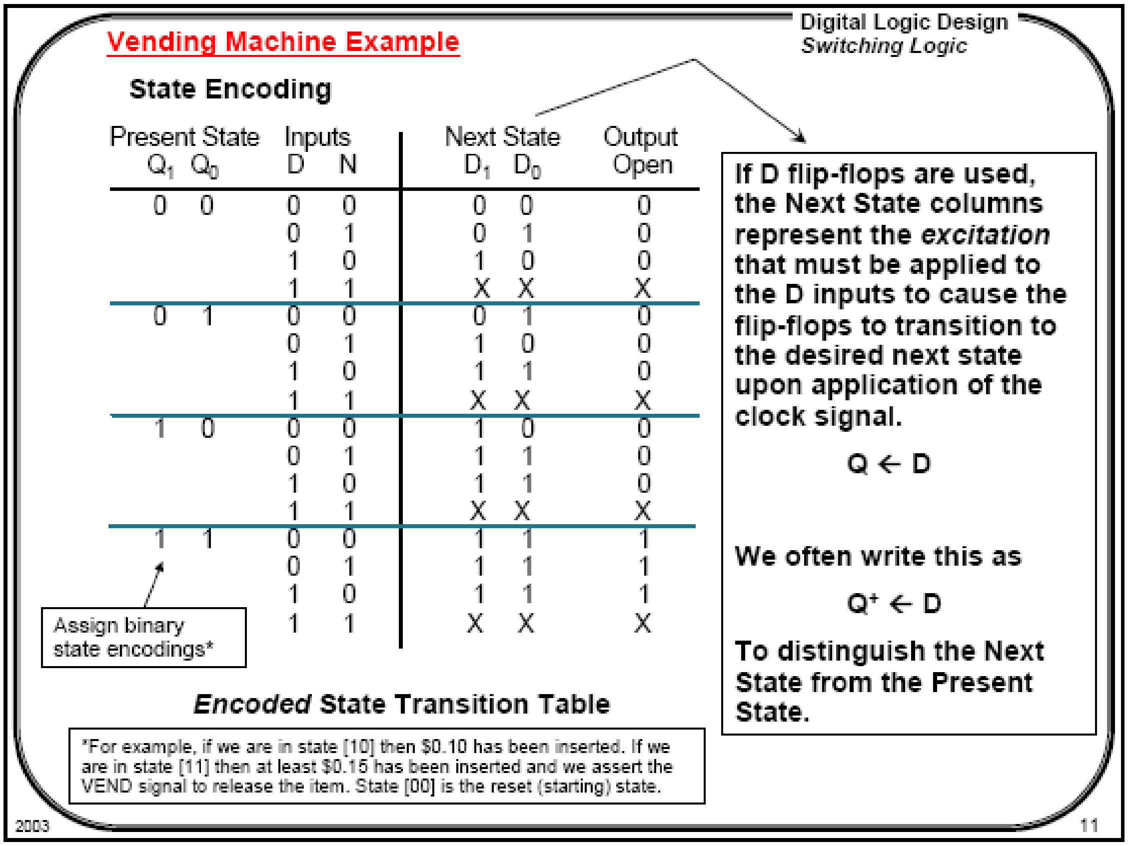
Present State	Inputs D	Inputs N	Next State	Output Open
0¢	0	0	0¢	0
	0	1	5¢	0
	1	0	10¢	0
	1	1	X	X
5¢	0	0	5¢	0
	0	1	10¢	0
	1	0	15¢	0
	1	1	X	X
10¢	0	0	10¢	0
	0	1	15¢	0
	1	0	15¢	0
	1	1	X	X
15¢	X	X	15¢	1

Symbolic *State Table* or Symbolic *State Transition Table*

The State Transition Table is a tabular version of the State Diagram. We begin by using descriptive names (mnemonics, etc.) for the states to aid in understanding. Finally, we *encode* the state transition table by assigning binary values to the states (next page), replacing the names we started with. The binary values are the actual Q values for the FFs in the circuit. The Q's define the states. We assign don't cares (X) for input combinations that cannot occur (i.e., we cannot insert both a nickel and dime at the same time).

2003 10

Next each state is given a unique binary number and placed back in the table.



Vending Machine Example

Digital Logic Design
Switching Logic

State Encoding

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

Assign binary state encodings*

If D flip-flops are used, the Next State columns represent the *excitation* that must be applied to the D inputs to cause the flip-flops to transition to the desired next state upon application of the clock signal.

$Q \leftarrow D$

We often write this as

$Q^{+} \leftarrow D$

To distinguish the Next State from the Present State.

***Encoded* State Transition Table**

*For example, if we are in state [10] then $0.10 has been inserted. If we are in state [11] then at least $0.15 has been inserted and we assert the VEND signal to release the item. State [00] is the reset (starting) state.

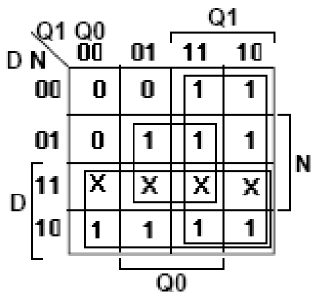
2003 11

From here, we must figure out feedback logic for how the next state will be decided. This is done with k-maps. In essence, we have four inputs that determine the next state - D, N, Q1, and Q0. The outputs for the four inputs are D1 and D0.

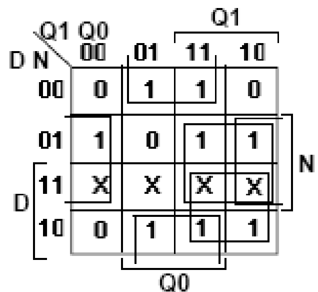
The k-maps are worked out on the next page. Also, the output variable **open** is worked out using the present state as the input. You may notice quickly, however, that **open** is logic '1' only when the present state is "11" or Q1Q0.

Vending Machine Example

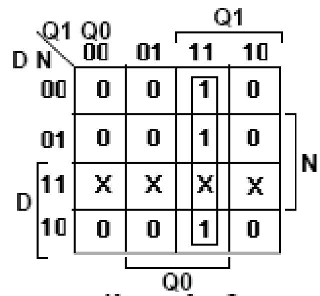
Determine minimized excitation equations based on desired transitions



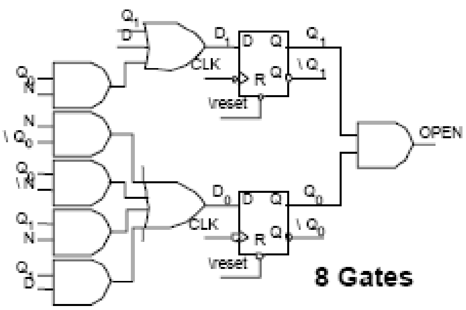
K-map for D1



K-map for D0



K-map for Open



D1 = Q1 + D + Q0 N

D0 = N $\overline{\text{Q0}}$ + Q0 $\overline{\text{N}}$ + Q1 N + Q1 D

OPEN = Q1 Q0

The present state (Q1 and Q0) and the nickel (N) and dime (D) signals are the inputs. We must design logic that takes these inputs and computes the excitation logic for the two D FF inputs (D1 and D0) necessary to produce the next state values (Q1+ and Q0+). The K-maps are used to determine the minimized logic for this operation. The circuit will now move to the correct next state given the present state (Q1 and Q0) and inputs N and D.