CMSC 313
COMPUTER ORGANIZATION
&
ASSEMBLY LANGUAGE
PROGRAMMING

LECTURE 24, FALL 2012

TOPICS TODAY

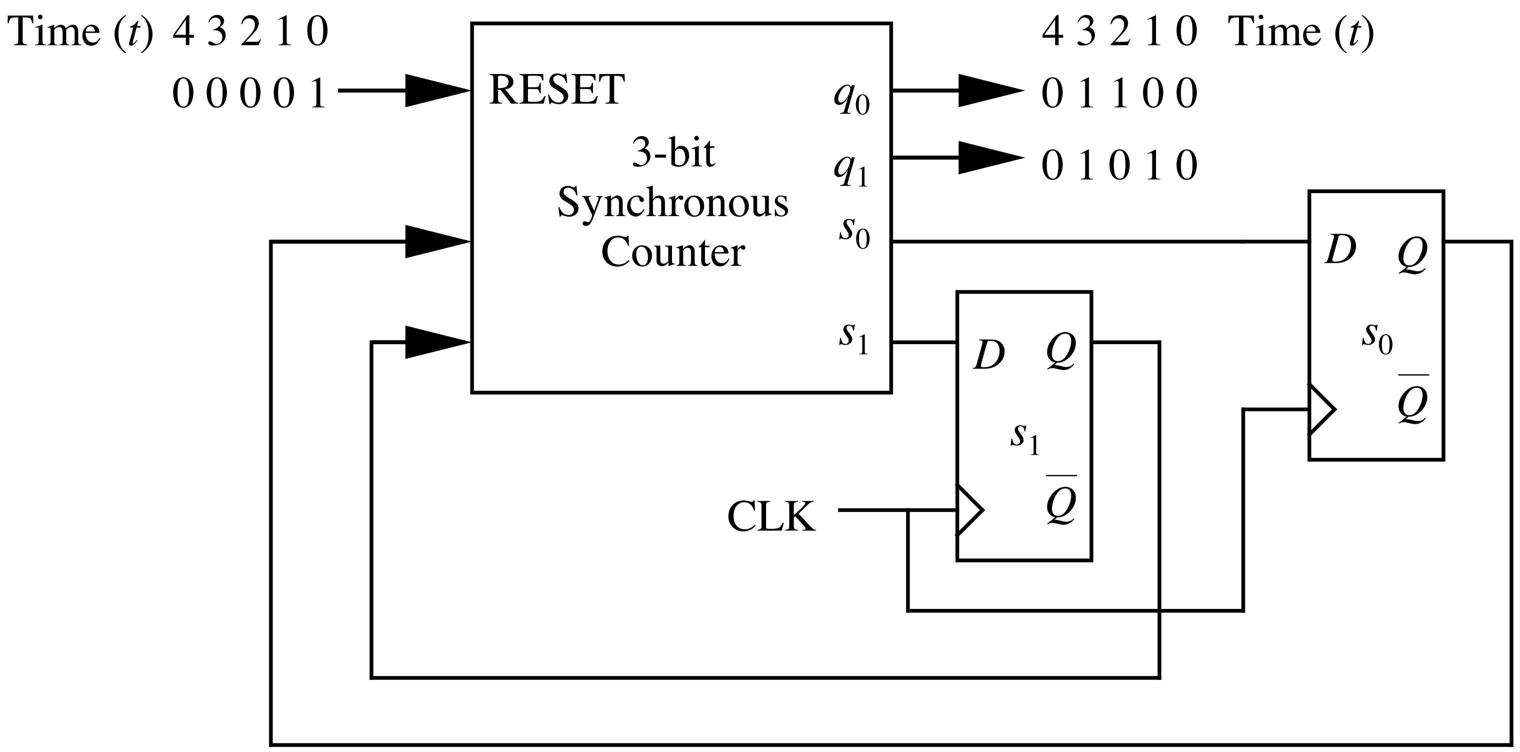
- **Finite State Machines**
- **Example: Mod-4 Counter**
- **Example: Vending Machine**

FINITE STATE MACHINES

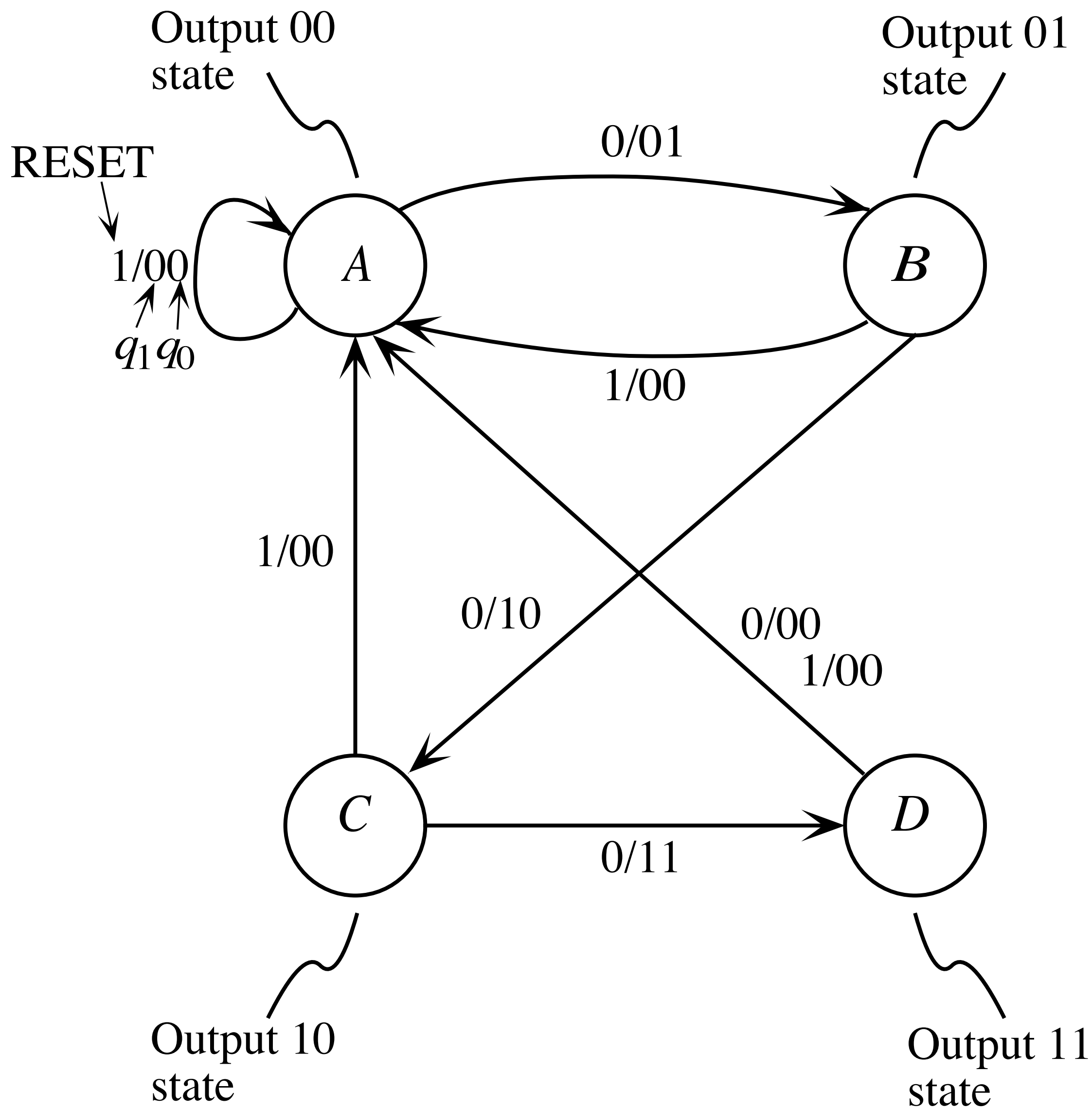
EXAMPLE:
MOD 4 COUNTER

Example: Modulo-4 Counter

- Counter has a clock input (CLK) and a RESET input.
- Counter has two output lines, which take on values of 00, 01, 10, and 11 on subsequent clock cycles.



State Transition Diagram for Mod-4 Counter



State Table for Mod-4 Counter

Present state \ Input	*RESET*	
	0	1
A	*B*/01	*A*/00
B	*C*/10	*A*/00
C	*D*/11	*A*/00
D	*A*/00	*A*/00

Next state Output

State Assignment for Mod-4 Counter

Present state (S_t) \ Input	*RESET* 0	*RESET* 1
A:00	01/01	00/00
B:01	10/10	00/00
C:10	11/11	00/00
D:11	00/00	00/00

Truth Table for Mod-4 Counter

RESET $r(t)$	$s_1(t)$	$s_0(t)$	$s_1 s_0(t+1)$	$q_1 q_0(t+1)$
0	0	0	01	01
0	0	1	10	10
0	1	0	11	11
0	1	1	00	00
1	0	0	00	00
1	0	1	00	00
1	1	0	00	00
1	1	1	00	00

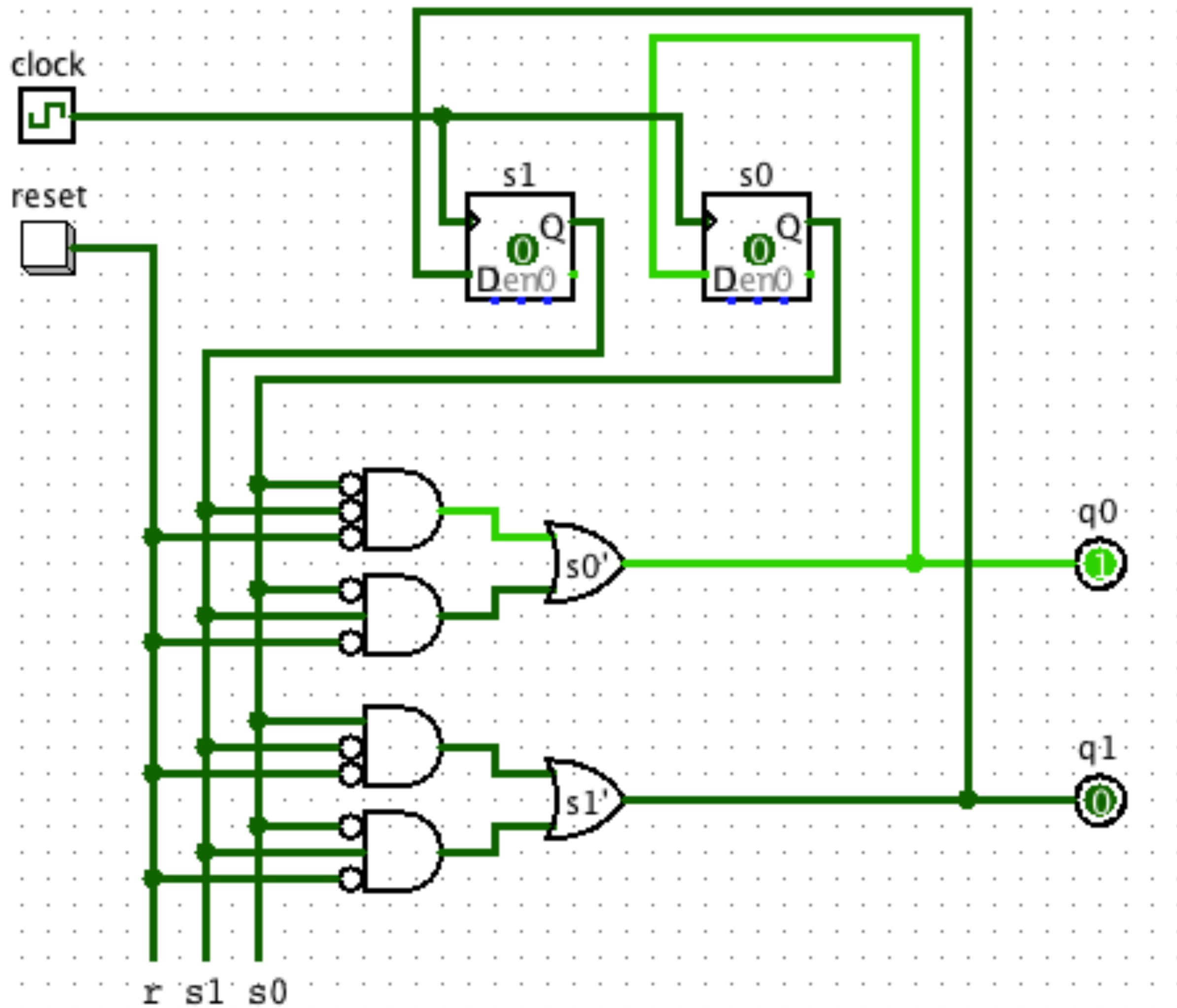
$$s_0(t+1) = \overline{r(t)}\,\overline{s_1(t)}\,\overline{s_0(t)} + \overline{r(t)}\,s_1(t)\,\overline{s_0(t)}$$

$$s_1(t+1) = \overline{r(t)}\,\overline{s_1(t)}\,s_0(t) + \overline{r(t)}\,s_1(t)\,\overline{s_0(t)}$$

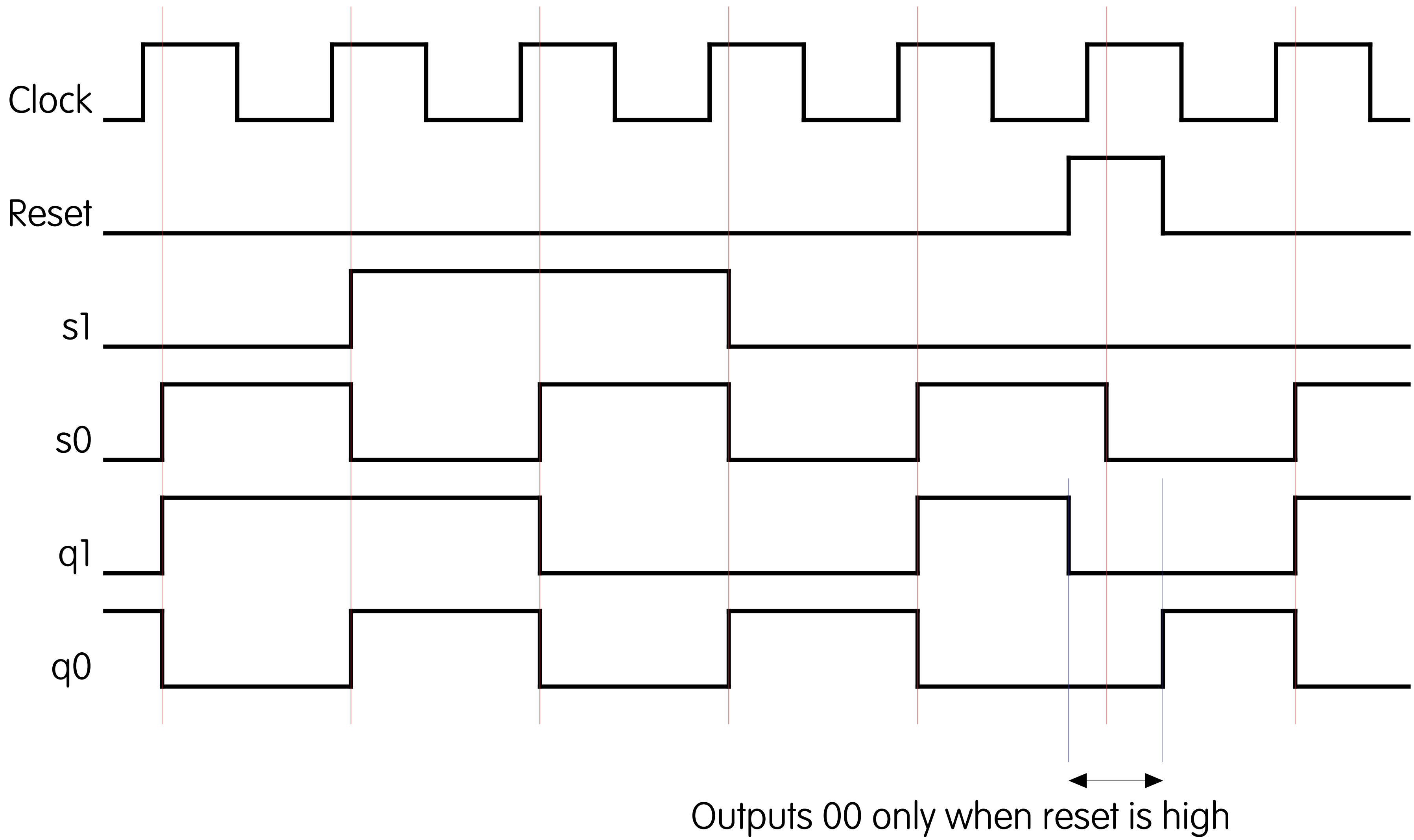
$$q_0(t+1) = \overline{r(t)}\,\overline{s_1(t)}\,\overline{s_0(t)} + \overline{r(t)}\,s_1(t)\,\overline{s_0(t)}$$

$$q_1(t+1) = \overline{r(t)}\,\overline{s_1(t)}\,s_0(t) + \overline{r(t)}\,s_1(t)\,\overline{s_0(t)}$$

Mod 4 Counter

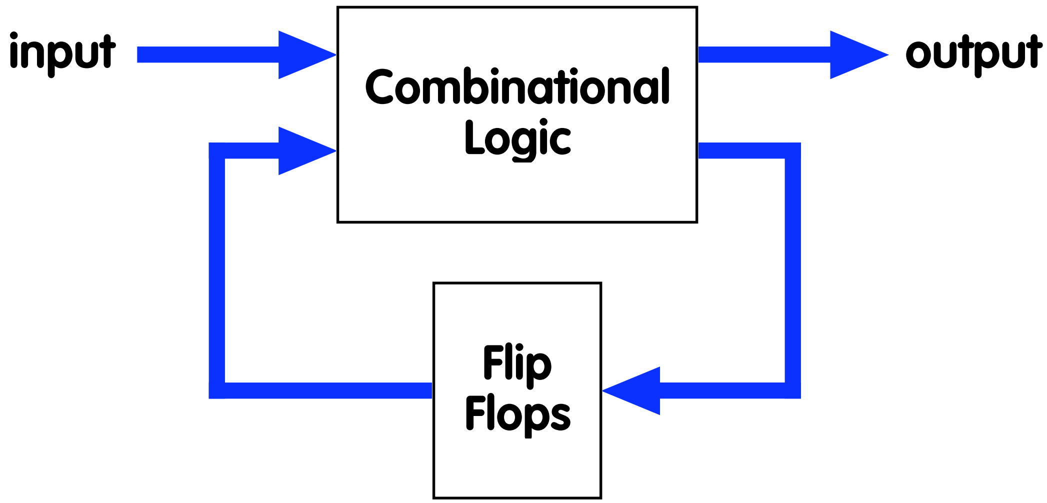


Mod 4 Counter Timing

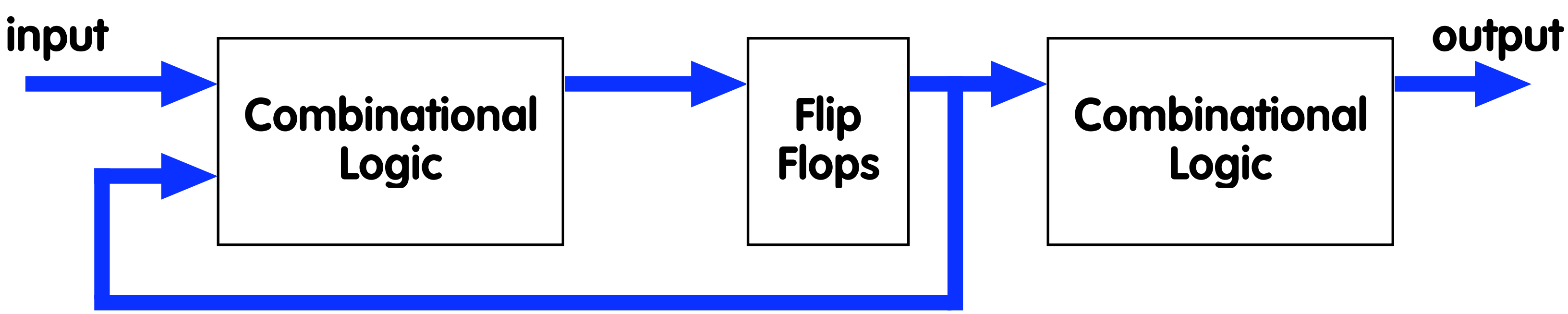


Mealy vs Moore Finite State Machines

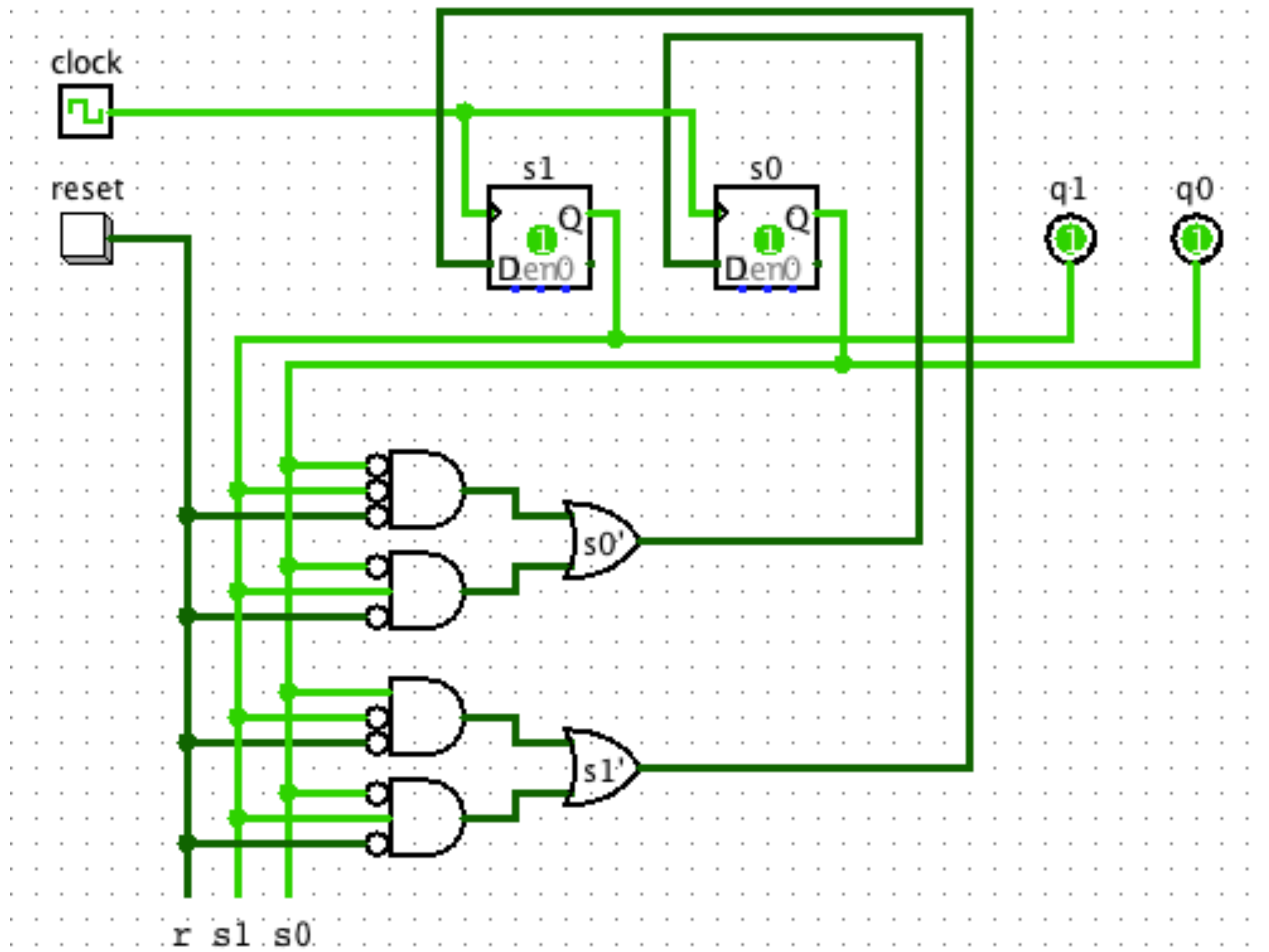
- Mealy: output depends on input and state bits



- Moore: output depends only on state bits



Mod 4 Counter (Moore)

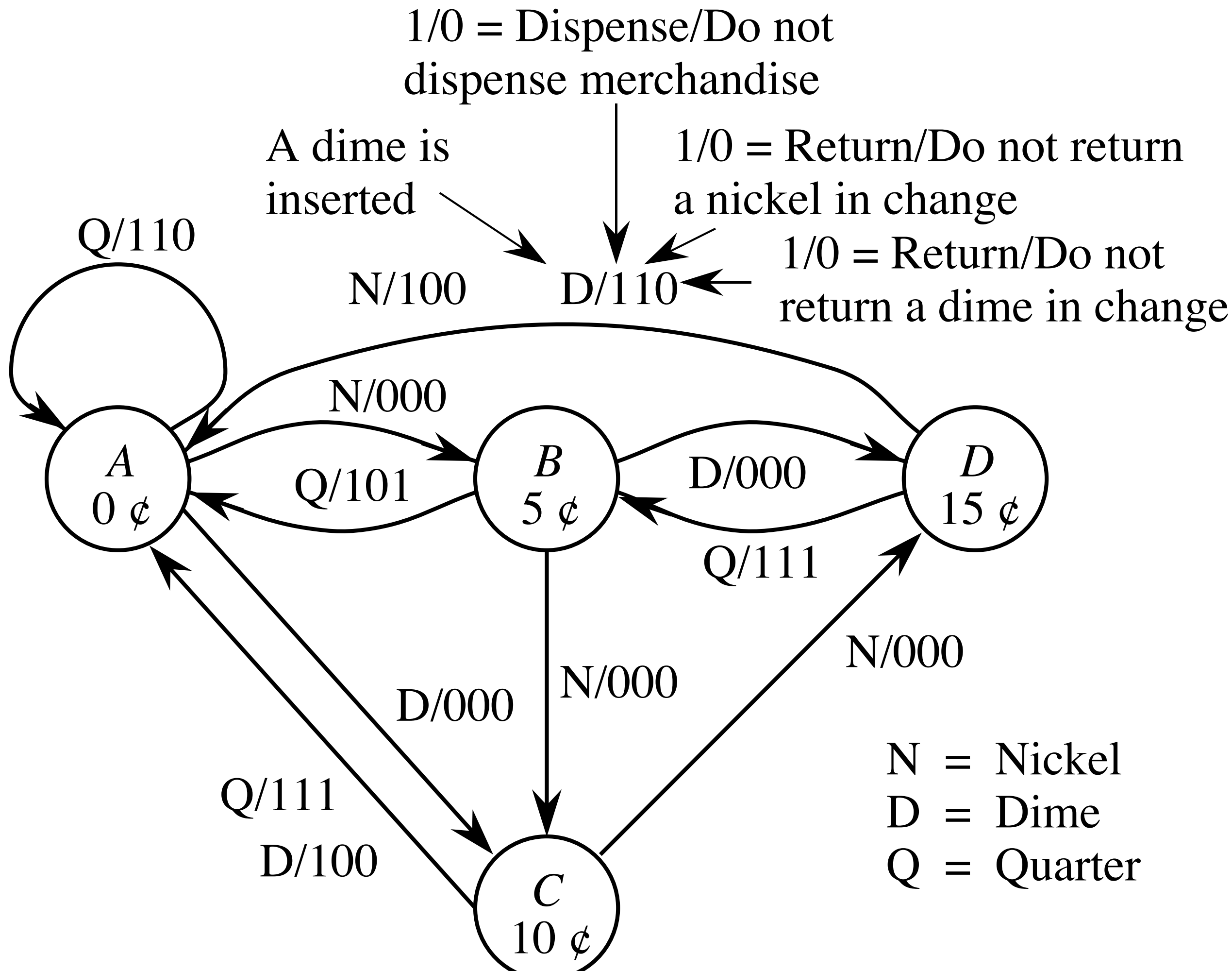


EXAMPLE:
VENDING MACHINE

Example: A Vending Machine Controller

- Example: Design a finite state machine for a vending machine controller that accepts nickels (5 cents each), dimes (10 cents each), and quarters (25 cents each). When the value of the money inserted equals or exceeds twenty cents, the machine vends the item and returns change if any, and waits for next transaction.
- Implement with PLA and D flip-flops.

Vending Machine State Transition Diagram



Vending Machine State Table and State Assignment

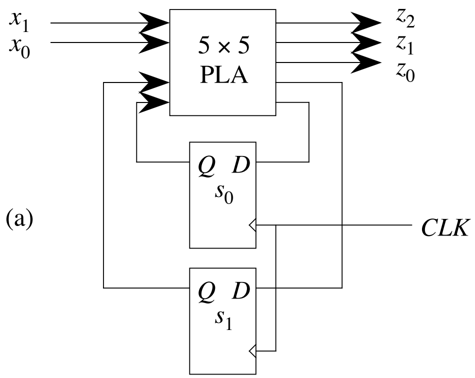
P.S. \ Input	N 00	D 01	Q 10
A	*B*/000	*C*/000	*A*/110
B	*C*/000	*D*/000	*A*/101
C	*D*/000	*A*/100	*A*/111
D	*A*/100	*A*/110	*B*/111

(a)

P.S. \ Input	N x_1x_0 00	D x_1x_0 01	Q x_1x_0 10
s_1s_0	s_1s_0 / $z_2z_1z_0$		
A:00	01/000	10/000	00/110
B:01	10/000	11/000	00/101
C:10	11/000	00/100	00/111
D:11	00/100	00/110	01/111

(b)

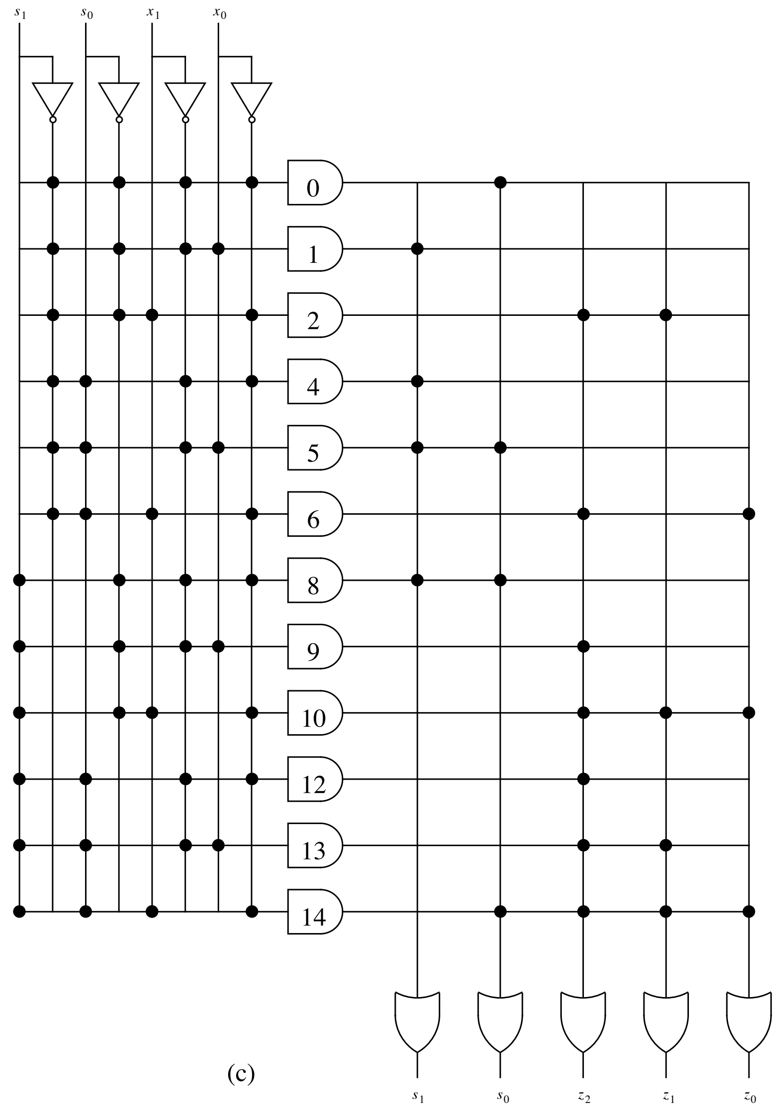
PLA Vending Machine Controller



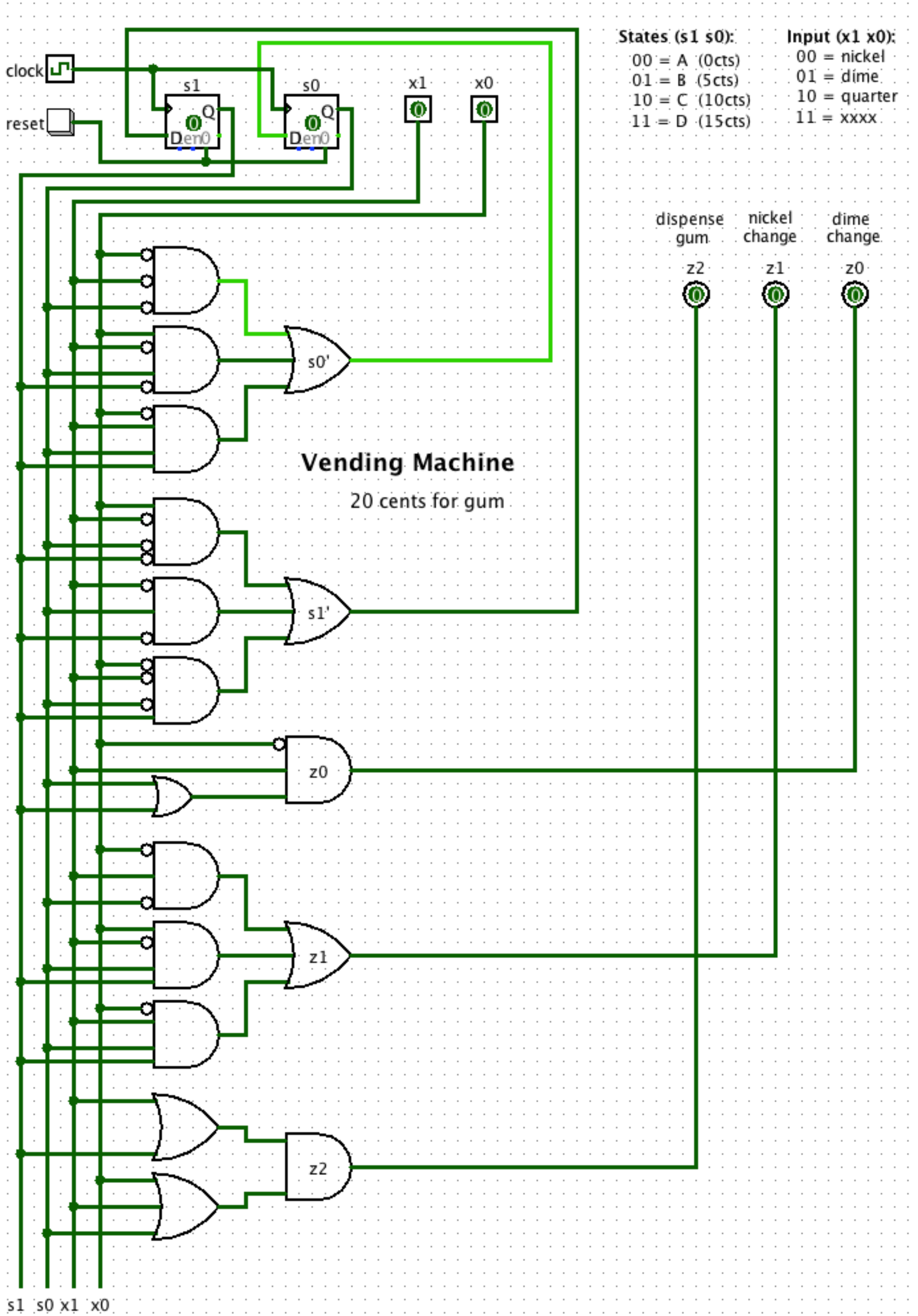
(a)

Base 10 equivalent	Present state s_1	Present state s_0	Coin x_1	Coin x_0	Next state s_1	Next state s_0	Dispense z_2	Return nickel z_1	Return dime z_0
0	0	0	0	0	0	1	0	0	0
1	0	0	0	1	1	0	0	0	0
2	0	0	1	0	0	0	1	1	0
3	0	0	1	1	d	d	d	d	d
4	0	1	0	0	1	0	0	0	0
5	0	1	0	1	1	1	0	0	0
6	0	1	1	0	0	0	1	0	1
7	0	1	1	1	d	d	d	d	d
8	1	0	0	0	1	1	0	0	0
9	1	0	0	1	0	0	1	0	0
10	1	0	1	0	0	0	1	1	1
11	1	0	1	1	d	d	d	d	d
12	1	1	0	0	0	0	1	0	0
13	1	1	0	1	0	0	1	1	0
14	1	1	1	0	0	1	1	1	1
15	1	1	1	1	d	d	d	d	d

(b)



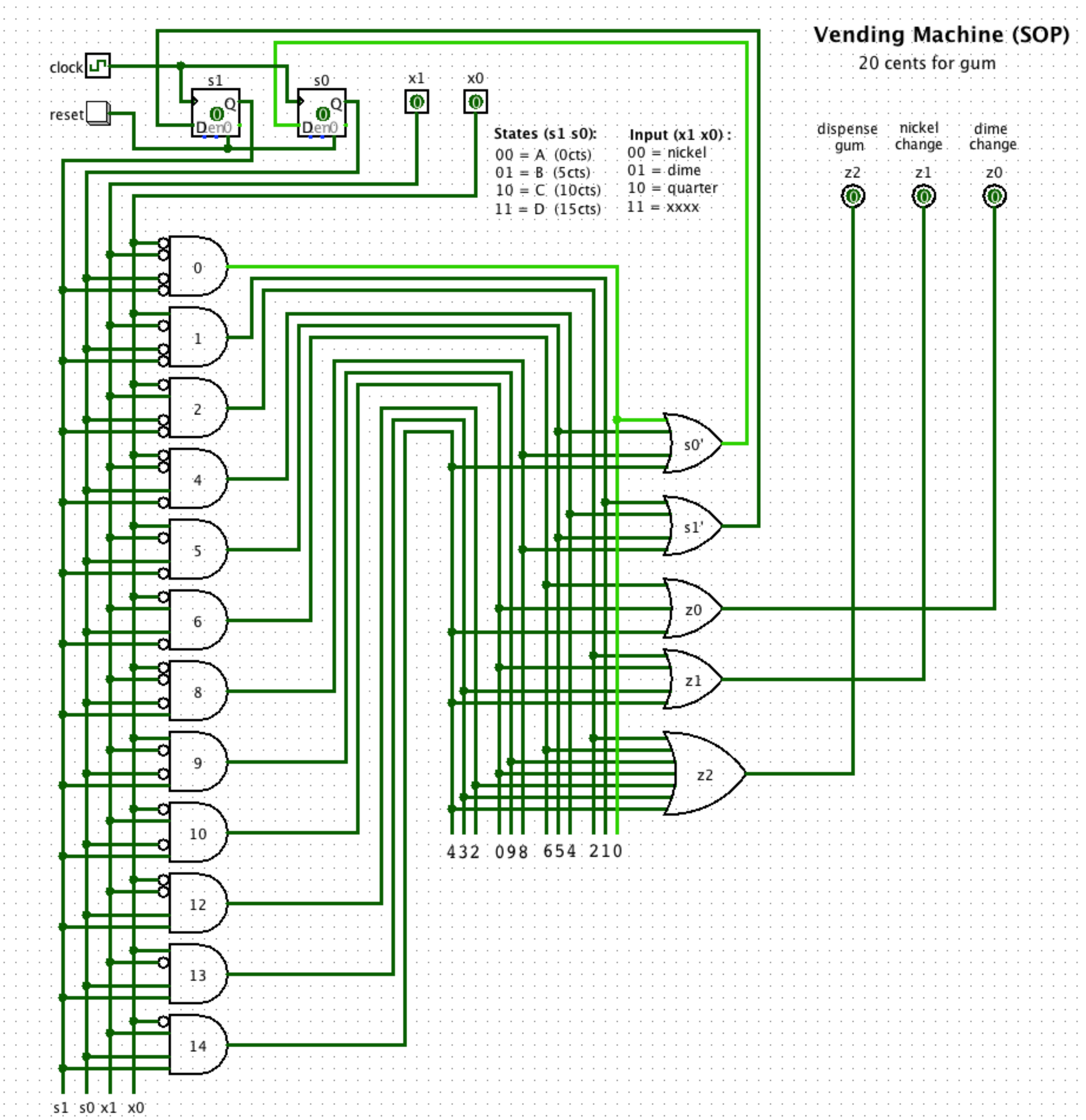
(c)



States (s1 s0):
00 = A (0cts)
01 = B (5cts)
10 = C (10cts)
11 = D (15cts)
Input (x1 x0):
00 = nickel
01 = dime
10 = quarter
11 = xxxx
clock
reset
s1
s0
x1
x0
dispense gum
nickel change
dime change
z2
z1
z0
s0'
s1'
Vending Machine
20 cents for gum
s1 s0 x1 x0

Vending Machine (SOP)

20 cents for gum



NEXT TIME

- **Finite State Machine Simplification**
- **2-bit CPU**