CMPE 212

Principles of Digital Design

Lecture 28

Design Examples of Finite State Machines

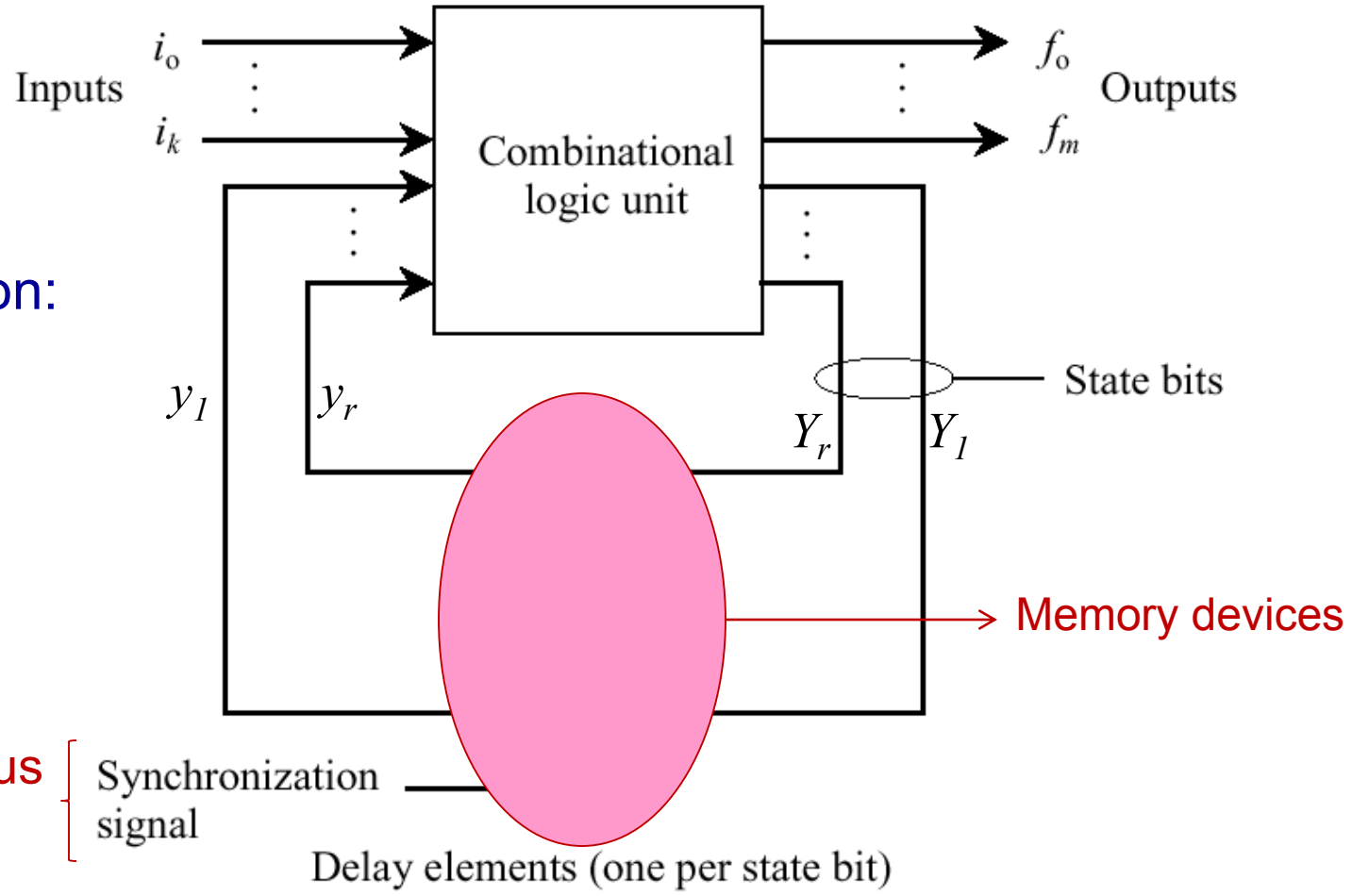
May 4, 2016

www.csee.umbc.edu/~younis/CMPE212/CMPE212.htm

Sequential Circuit Model

Possible realization:

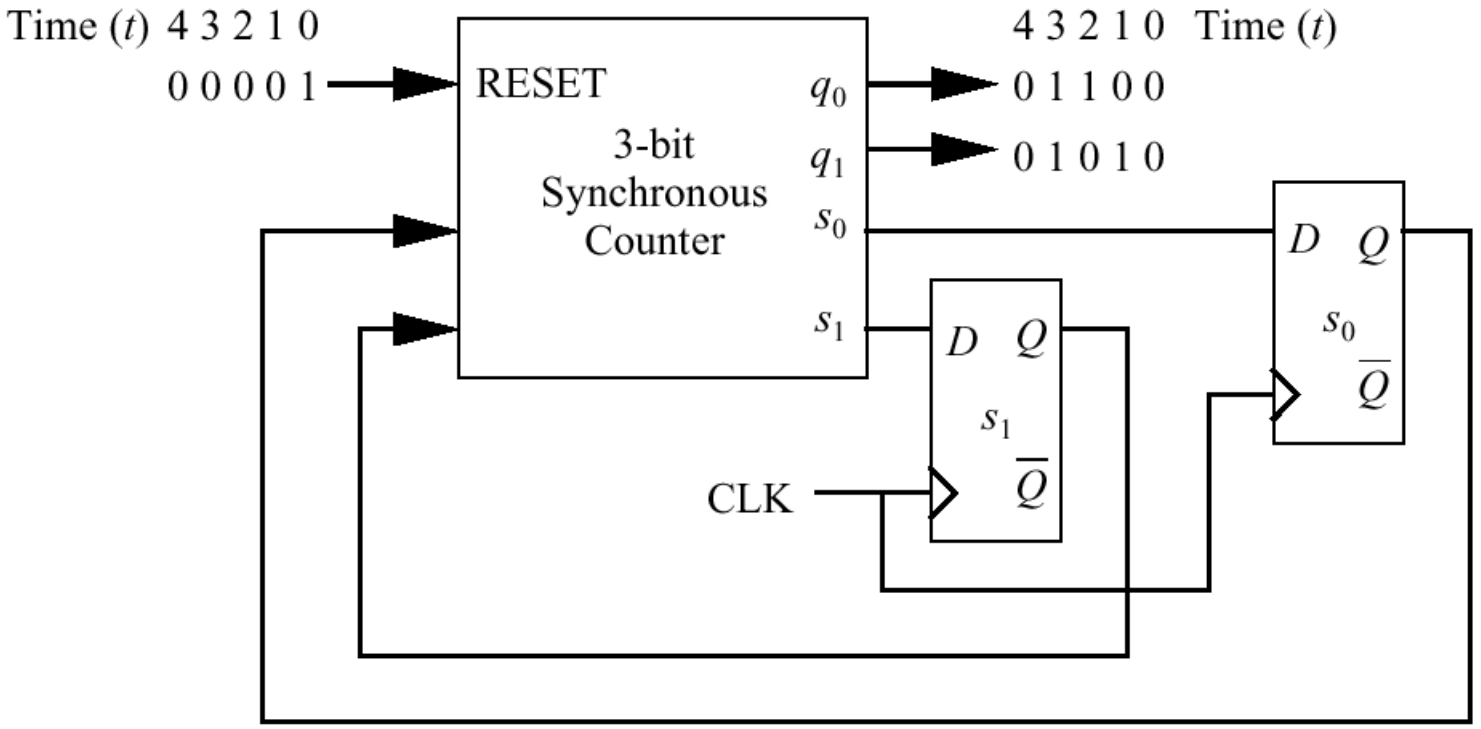
1. Mealy model
2. Moore model



- Composed of a combinational logic unit and delay elements in a feedback path, which maintains state information
- Defined by output relation to input and circuit state (values in flip-flops)

Example: FSM for Modulo-4 Counter

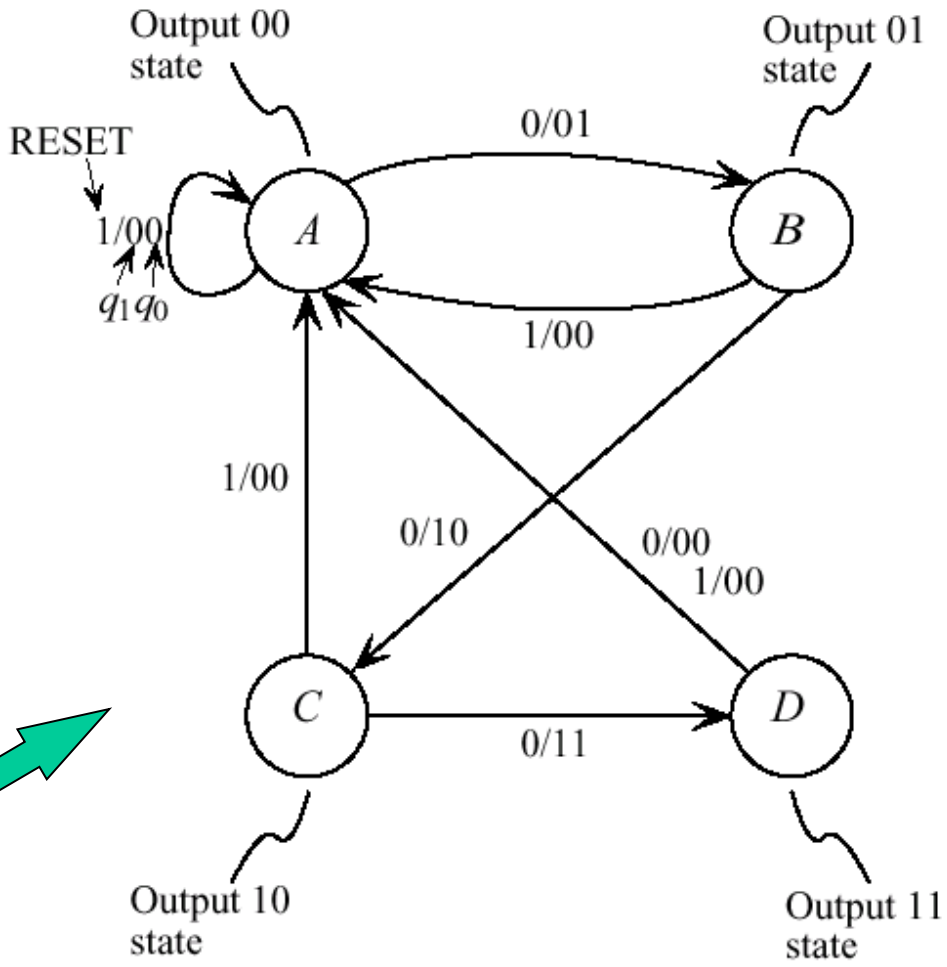
- Finite state machines use flip flops and combinational logic to implement the desired function
- A counter is a sequential circuit that tracks the number of ones or zeros in an input or the number of clock cycles
- A modulo-4 counter has two output lines, which take on values of 00, 01, 10, and 11 on subsequent clock cycles



* Slide is courtesy of M. Murdocca and V. Heuring

State Transition Diagram

- State transition diagram captures the the behavior change and output based on the current and previous inputs
- The number of bits (flip flops) needed are determined based on the number of states
- The states and transitions among states depend on the function to be implemented
- State diagrams are widely used notation for controllers



State diagram for Mod-4 Counter

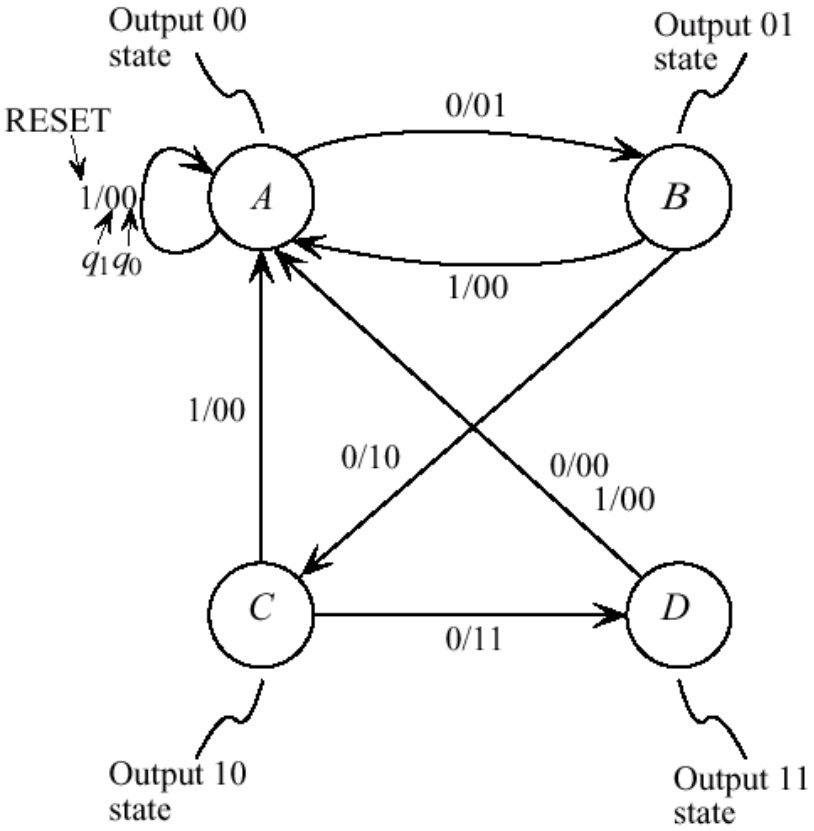
* Slide is courtesy of M. Murdocca and V. Heuring

State Table for Mod-4 Counter

- The state table is a rewrite of the state transition diagram

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

Next state Output



* Slide is courtesy of M. Murdocca and V. Heuring

State Assignment for Mod-4 Counter

- After creating the state table, states are assigned binary codes

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

Next state Output

Number of bits (flip flops) depends on number of states

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

* Slide is courtesy of M. Murdocca and V. Heuring

Truth Table 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

RESET $r(t)$	$s_1(t)$	$s_0(t)$	$s_1s_0(t+1)$	$q_1q_0(t+1)$
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

- From the state table we can extract the truth tables for the next state and output functions
- The indices for the state variables indicate timing relationships

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

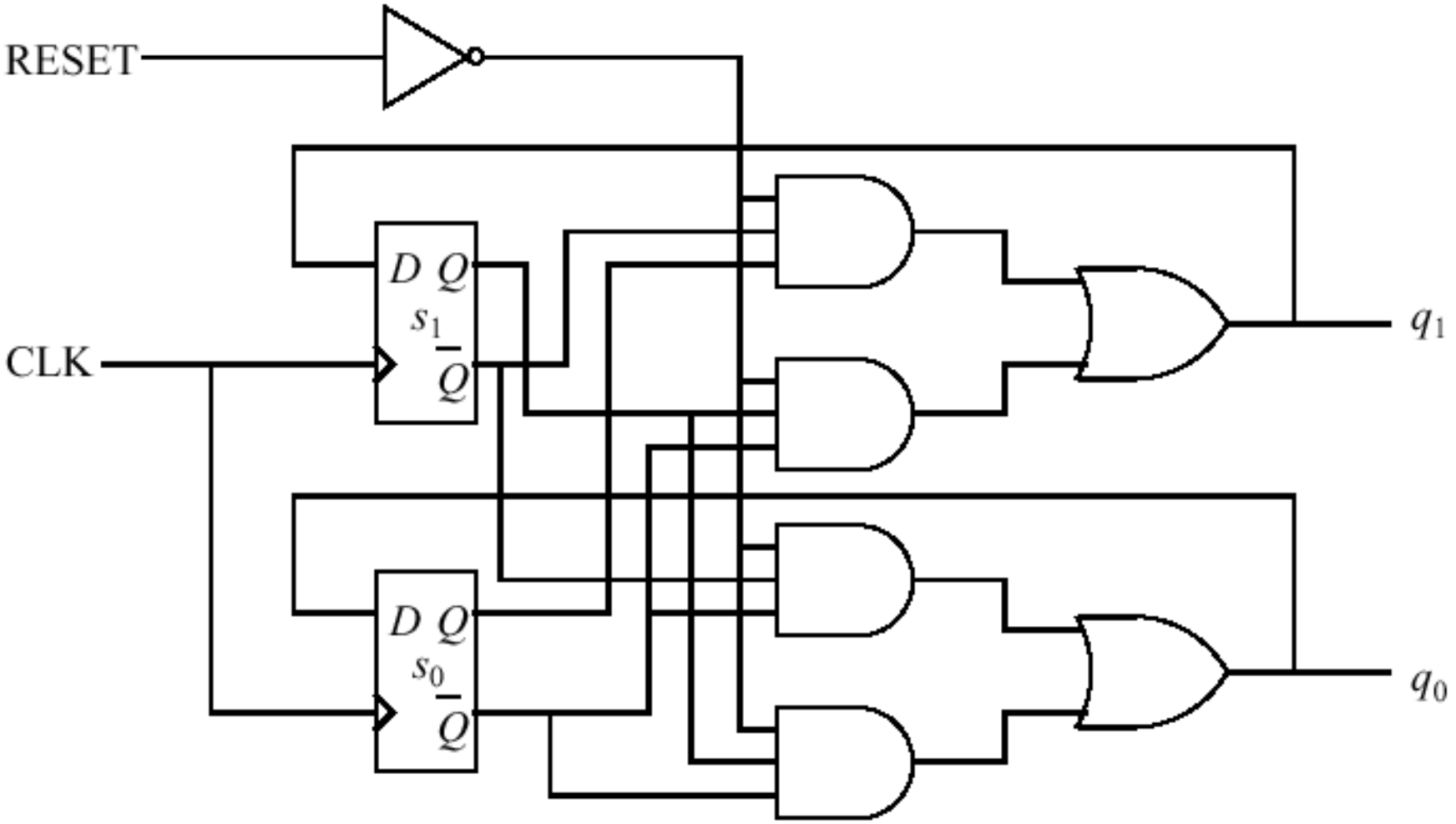
$$s_1(t+1) = \overline{r(t)}\,\overline{s_1(t)}\,s_0(t) + \overline{r(t)}\,s_1(t)\,\overline{s_0(t)}$$

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

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

* Slide is courtesy of M. Murdocca and V. Heuring

Logic Design for Mod-4 Counter



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

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

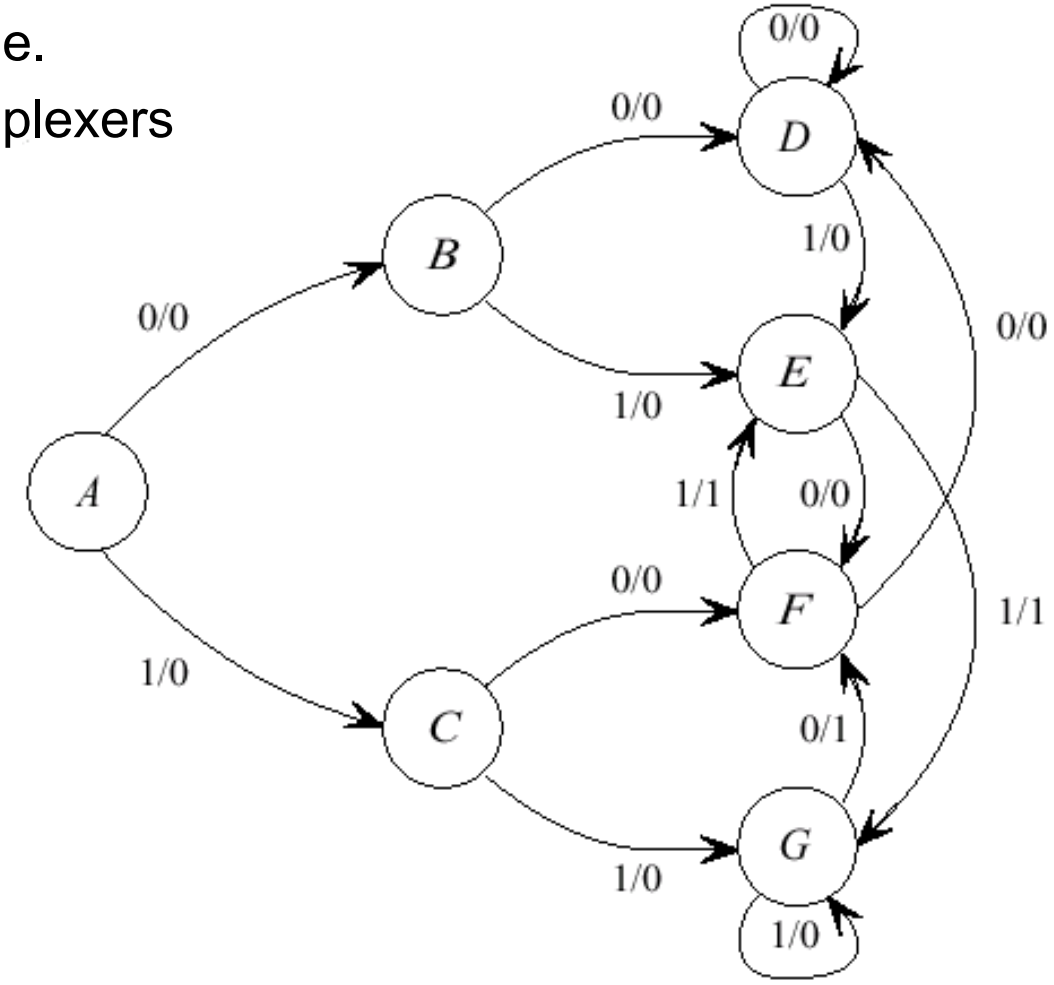
$$q_0(t+1) = \overline{r(t)}\,\overline{s_1(t)}\,\overline{s_0(t)} + \overline{r(t)}\,s_1(t)\,\overline{s_0(t)}$$

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

* Slide is courtesy of M. Murdocca and V. Heuring

Sequence Detector

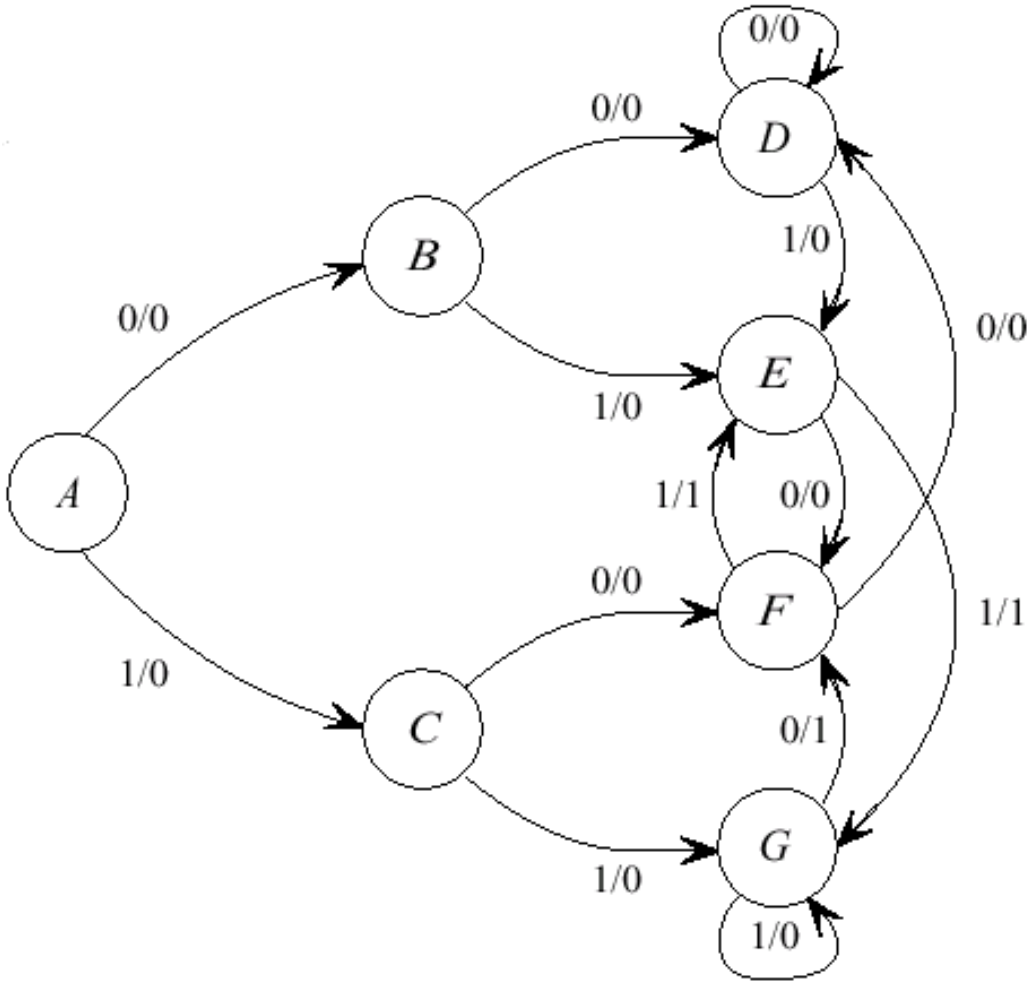
- Design a machine that outputs a 1 when exactly two of the last three inputs are 1, e.g. input sequence of 011011100 produces an output sequence of 001111010.
- Assume input is a 1-bit serial line.
- Use D flip-flops and 8-to-1 Multiplexers



➔ Start by constructing a state transition diagram

* Slide is courtesy of M. Murdocca and V. Heuring

Sequence Detector State Table



Present state \ Input	X = 0	X = 1
A	B/0	C/0
B	D/0	E/0
C	F/0	G/0
D	D/0	E/0
E	F/0	G/1
F	D/0	E/1
G	F/1	G/0

* Slide is courtesy of M. Murdocca and V. Heuring

Sequence Detector State Assignment

Present state \ Input	X = 0	X = 1
A: 000	001/0	010/0
B: 001	011/0	100/0
C: 010	101/0	110/0
D: 011	011/0	100/0
E: 100	101/0	110/1
F: 101	011/0	100/1
G: 110	101/1	110/0

Present state \ Input	X = 0	X = 1
A	B/0	C/0
B	D/0	E/0
C	F/0	G/0
D	D/0	E/0
E	F/0	G/1
F	D/0	E/1
G	F/1	G/0

Input and state at time t				Next state and output at time t+1			
s_2	s_1	s_0	x	s_2	s_1	s_0	z
0	0	0	0	0	0	1	0
0	0	0	1	0	1	0	0
0	0	1	0	0	1	1	0
0	0	1	1	1	0	0	0
0	1	0	0	1	0	1	0
0	1	0	1	1	1	0	0
0	1	1	0	0	1	1	0
0	1	1	1	1	0	0	0
1	0	0	0	1	0	1	0
1	0	0	1	1	1	0	1
1	0	1	0	0	1	1	0
1	0	1	1	1	0	0	1
1	1	0	0	1	0	1	1
1	1	0	1	1	1	0	0
1	1	1	0	d	d	d	d
1	1	1	1	d	d	d	d

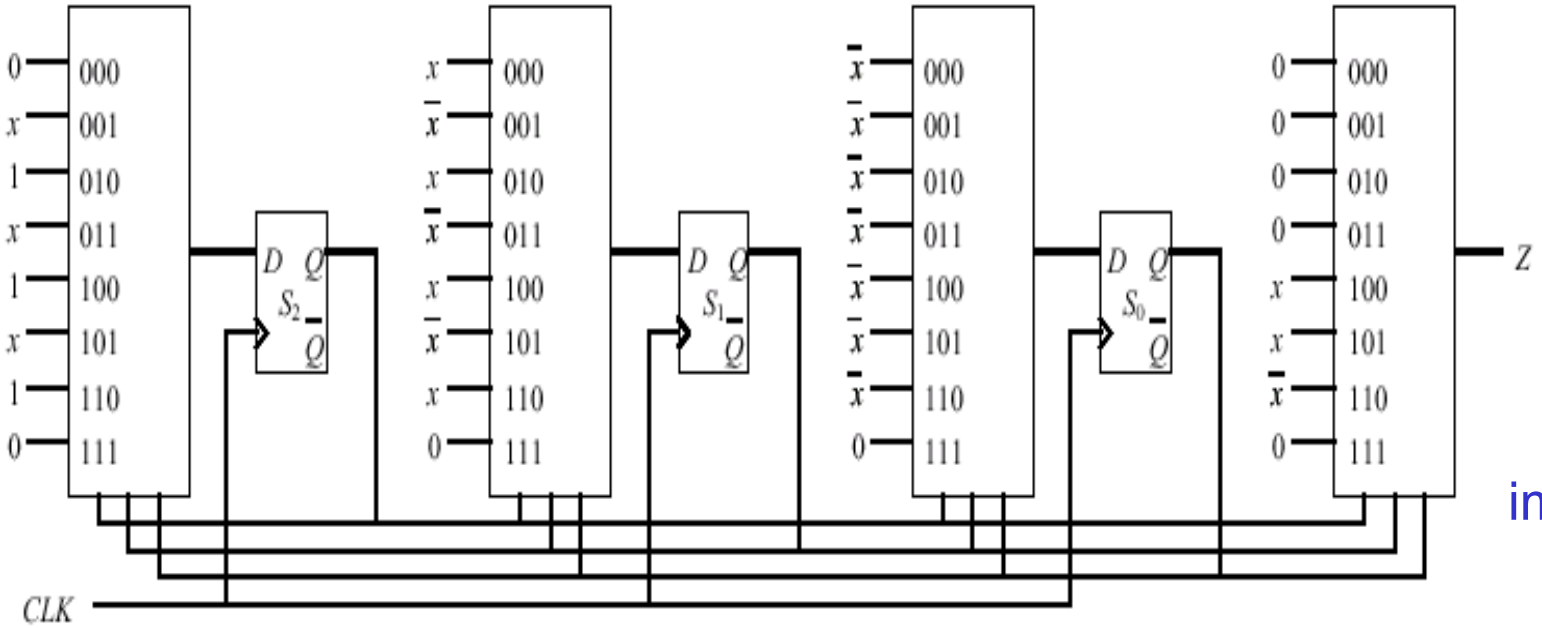
There is no state for “111”

* Slide is courtesy of M. Murdocca and V. Heuring

Sequence Detector Logic Diagram

- The “don’t care” entries have not been a factor when a MUX is used
- A gate level implementation can achieve further optimization (logic reduction)

Input and state at time t				Next state and output at time t+1			
s_2	s_1	s_0	x	s_2	s_1	s_0	z
0	0	0	0	0	0	1	0
0	0	0	1	0	1	0	0
0	0	1	0	0	1	1	0
0	0	1	1	1	0	0	0
0	1	0	0	1	0	1	0
0	1	0	1	1	1	0	0
0	1	1	0	0	1	1	0
0	1	1	1	1	0	0	0
1	0	0	0	1	0	1	0
1	0	0	1	1	1	0	1
1	0	1	0	0	1	1	0
1	0	1	1	1	0	0	1
1	1	0	0	1	0	1	1
1	1	0	1	1	1	0	0
1	1	1	0	d	d	d	d
1	1	1	1	d	d	d	d

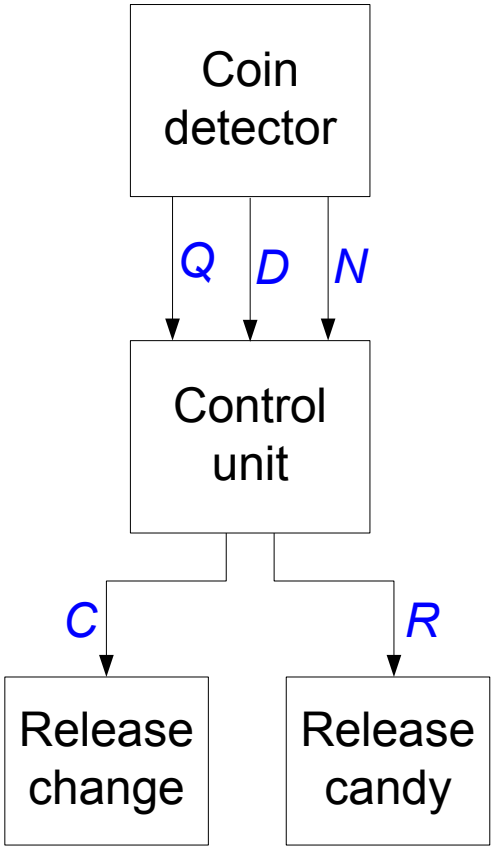


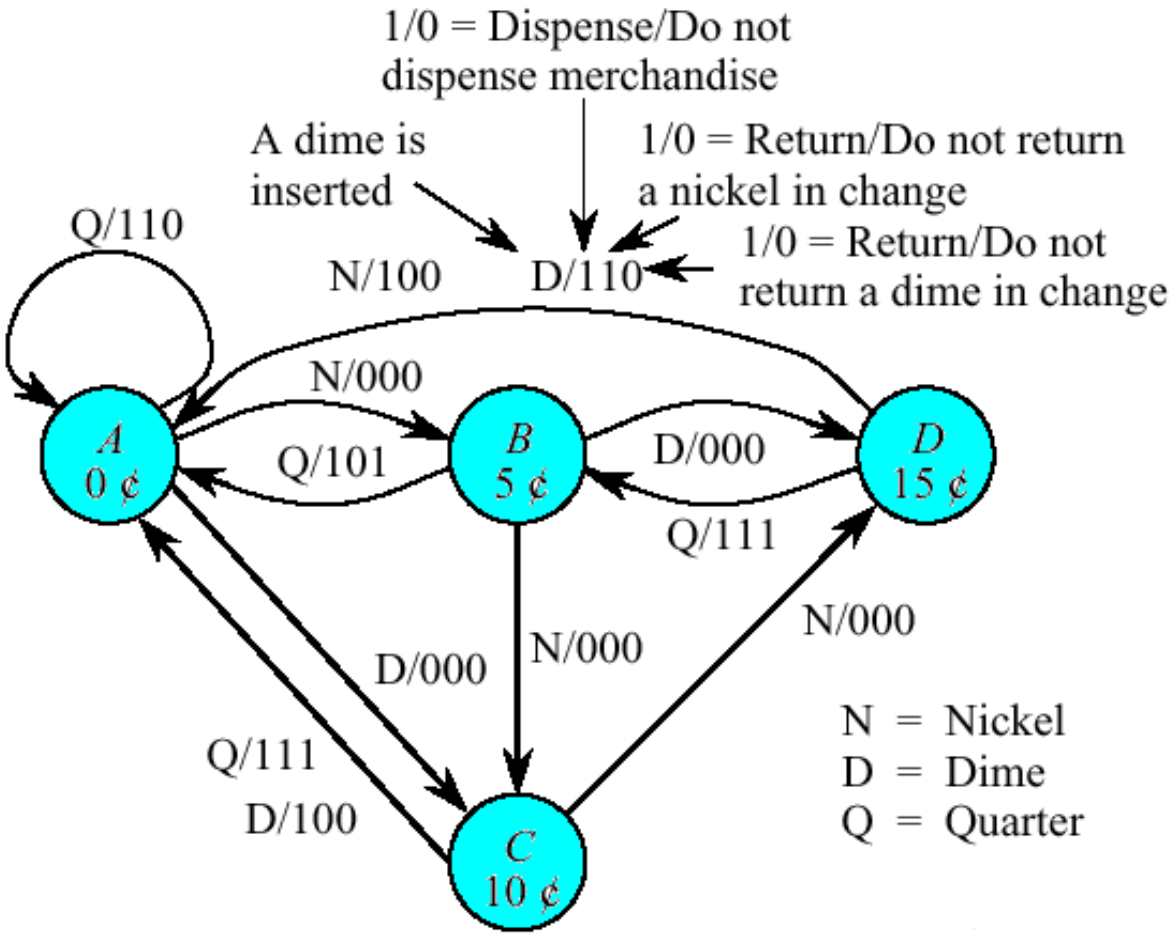
MUX-based implementation

* Slide is courtesy of M. Murdocca and V. Heuring

Vending Machine Controller

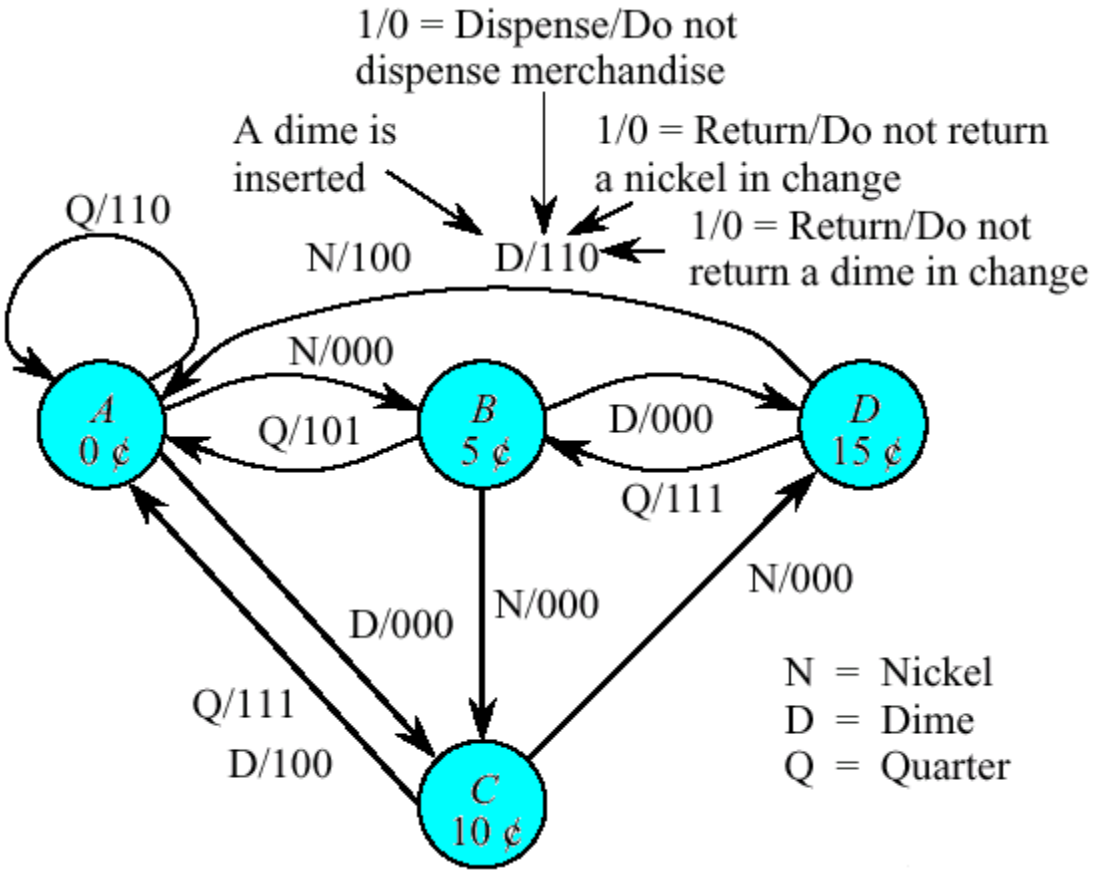
Design a finite state machine for a vending machine controller that accepts nickels, dimes, and quarters. When the value of the money inserted equals or exceeds twenty cents, the machine vends the item and returns change if any, and waits for next transaction.





* Figure is courtesy of M. Murdocca and V. Heuring

State Table and State Assignment

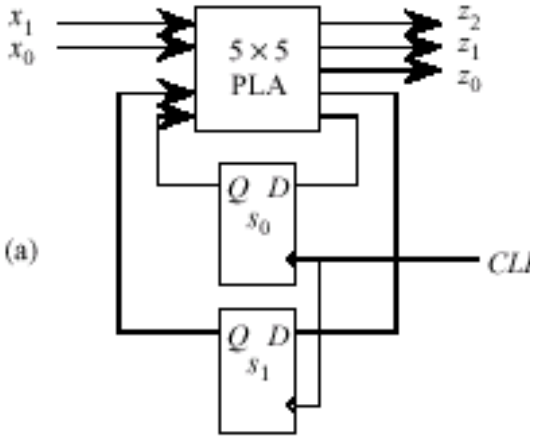


P.S. \ Input	N 00	D 01	Q 10
A	*B*/000	*C*/000	*A*/110
B	*C*/000	*D*/000	*A*/101
C	*D*/000	*A*/100	*A*/111
D	*A*/100	*A*/110	*B*/111

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

* Slide is courtesy of M. Murdocca and V. Heuring

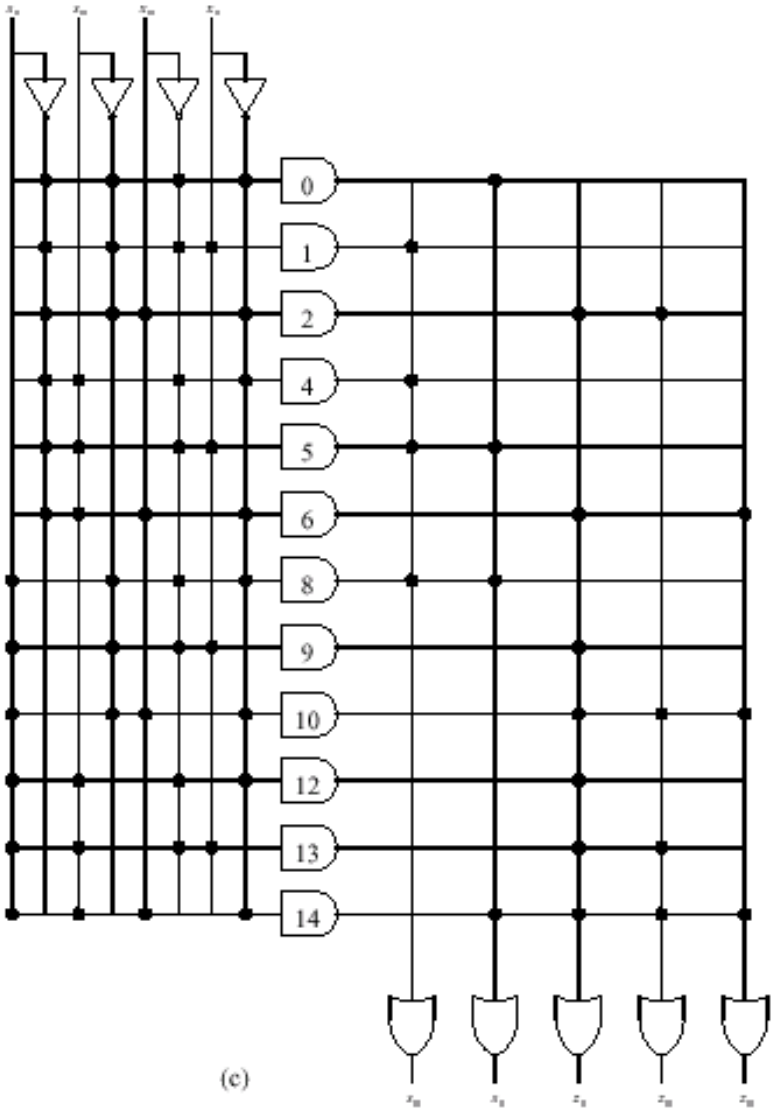
PLA Vending Machine Controller



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

(b)

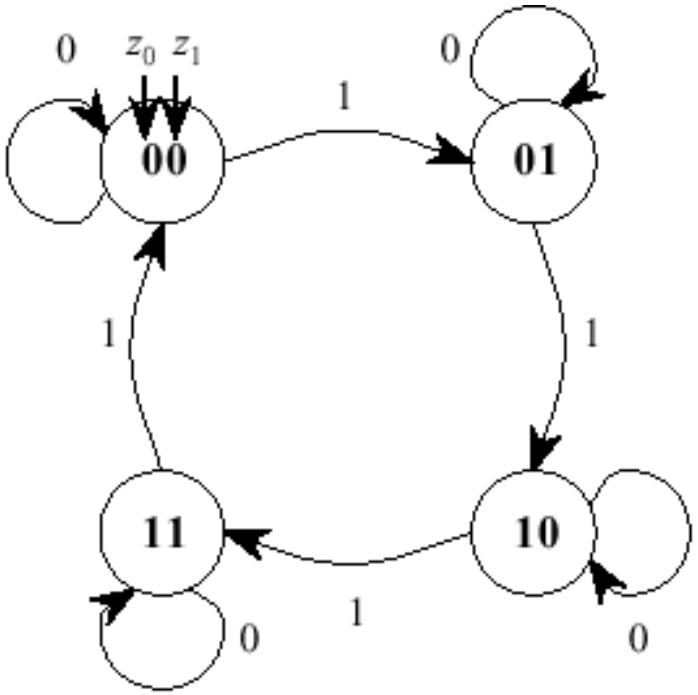
P.S. \ Input	N x_1x_0 00	D x_1x_0 01	Q x_1x_0 10
s_1s_0	s_1s_0 / $z_2z_1z_0$		
A:00	01/000	10/000	00/110
B:01	10/000	11/000	00/101
C:10	11/000	00/100	00/111
D:11	00/100	00/110	01/111

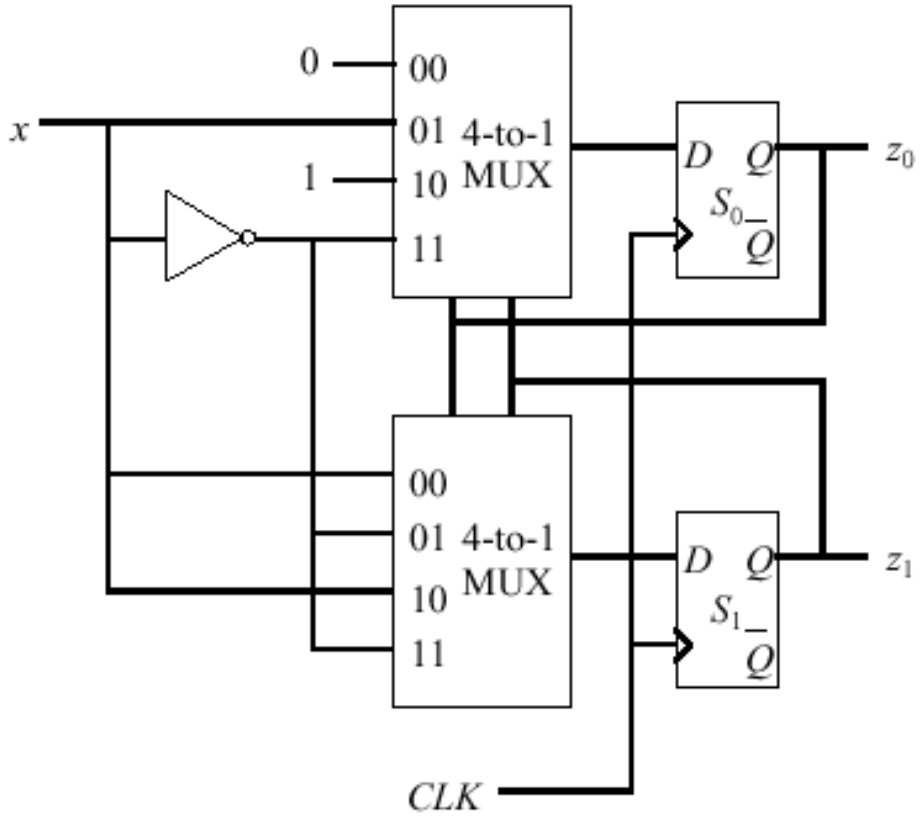


* Slide is courtesy of M. Murdocca and V. Heuring

Moore Counter

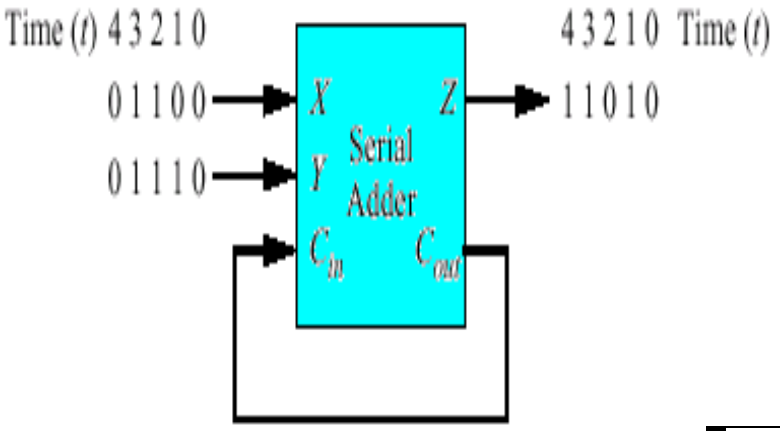
- Mealy Model: Outputs are functions of Inputs and Present State.
- Moore Model: Outputs are functions of Present State only
- Both Mealy and Moore models are used in practice
- The Mealy model is more powerful and expressive than the Moore model
- Moore model is easier to analyze since the output is synchronized with the state change

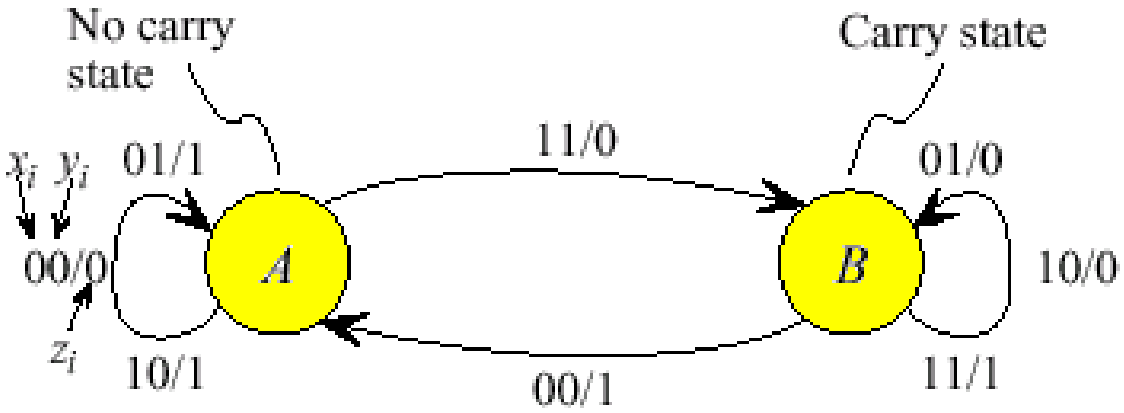




* Slide is courtesy of M. Murdocca and V. Heuring

Serial Adder





Present state \ Input	XY = 00	01	10	11
A	A/0	A/1	A/1	B/0
B	A/1	B/0	B/0	B/1

Next state Output

Present state (S_i) \ Input	XY = 00	01	10	11
A:0	0/0	0/1	0/1	1/0
B:1	0/1	1/0	1/0	1/1

State transition diagram, state table, and state assignment for a serial adder.

* Slide is courtesy of M. Murdocca and V. Heuring

Serial Adder Next-State Functions

Truth table showing next state functions for a serial adder for D, S-R, T, and J-K flip-flops

Present state (S_t) \ Input	XY = 00	01	10	11
A:0	0/0	0/1	0/1	1/0
B:1	0/1	1/0	1/0	1/1

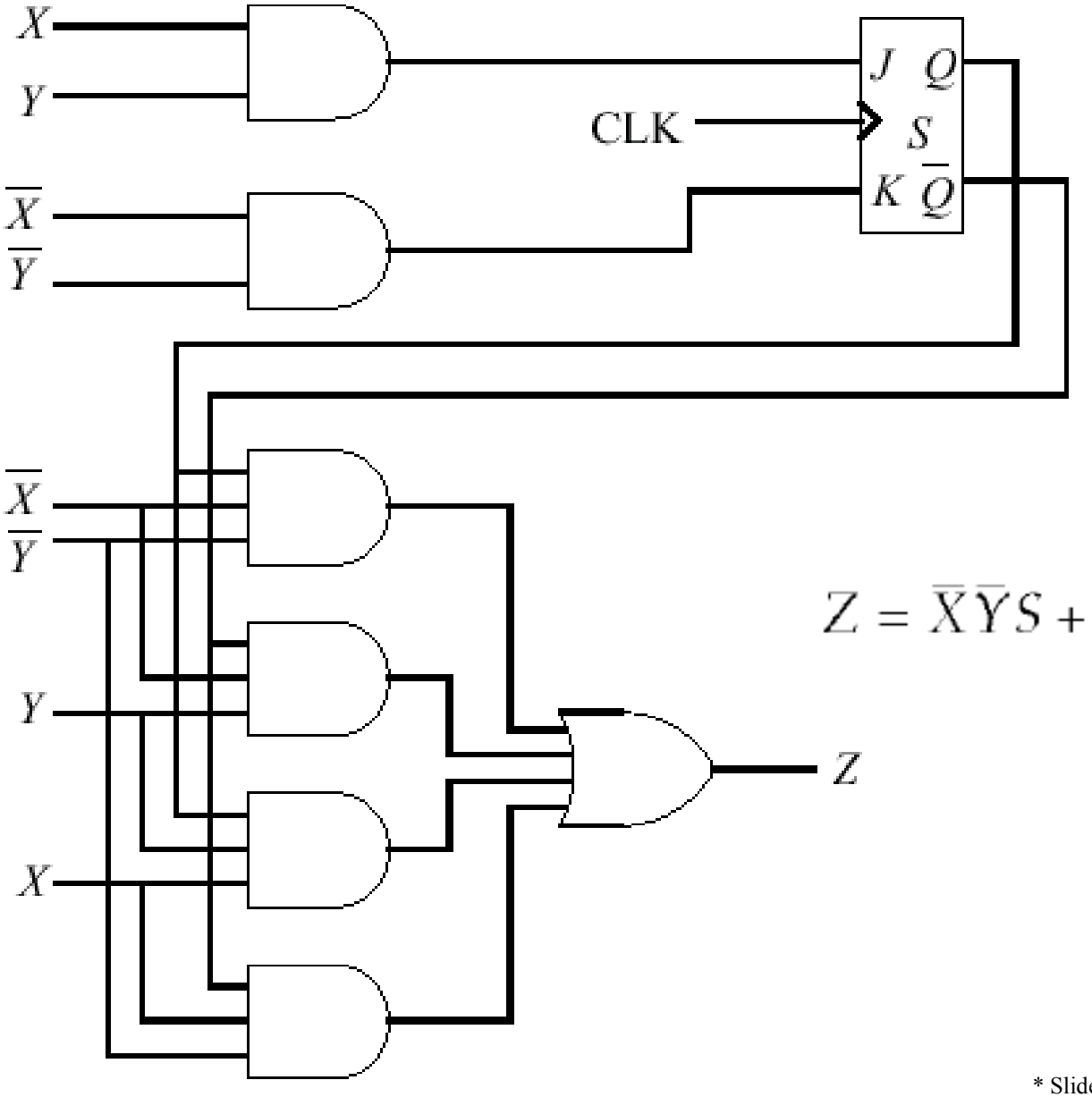
J-K flip-flop

Q_t	Q_{t+1}	J	K
0	0	0	d
0	1	1	d
1	0	d	1
1	1	d	0

Present State X	Y	S_t	(Set) D	S	(Reset) R	T	J	K	Z
0	0	0	0	0	0	0	0	d	0
0	0	1	0	0	1	1	d	1	1
0	1	0	0	0	0	0	0	d	1
0	1	1	1	0	0	0	d	0	0
1	0	0	0	0	0	0	0	d	1
1	0	1	1	0	0	0	d	0	0
1	1	0	1	1	0	1	1	d	0
1	1	1	1	0	0	0	d	0	1

* Slide is courtesy of M. Murdocca and V. Heuring

J-K Flip-Flop Serial Adder Circuit



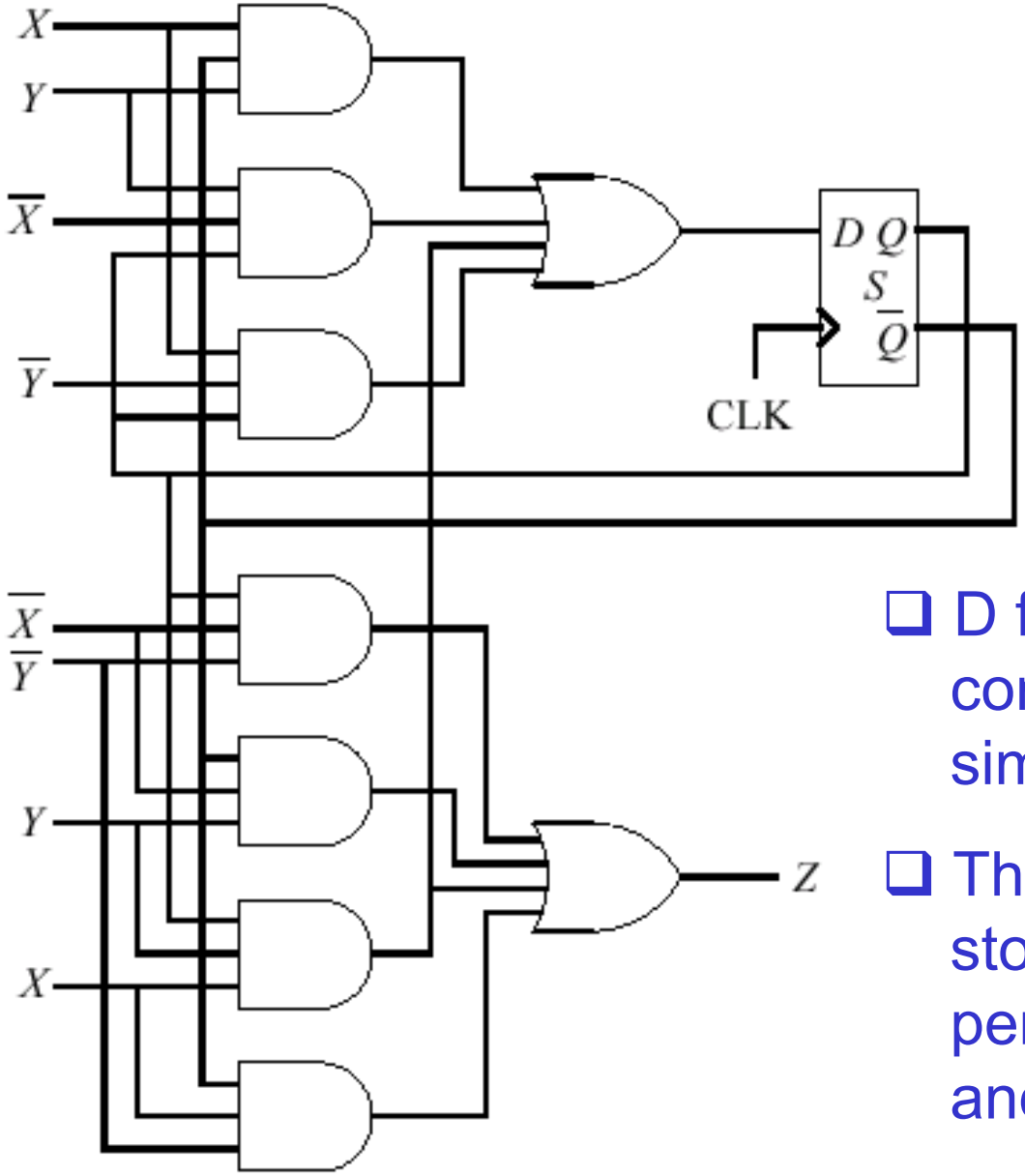
$$J = XY$$

$$K = \bar{X}\bar{Y}$$

$$Z = \bar{X}\bar{Y}S + \bar{X}Y\bar{S} + XYS + X\bar{Y}\bar{S}$$

* Slide is courtesy of M. Murdocca and V. Heuring

D Flip-Flop Serial Adder Circuit



- D flip-flops are the most commonly used and the simplest to design
- The choice of the type of storage devices depends on performance, availability and cost

* Slide is courtesy of M. Murdocca and V. Heuring