University of Florida
ECE Department

Joel D. Schipper
Summer 2007

LECTURE #16: Moore & Mealy Machines

EEL 3701: Digital Logic and Computer Systems

Based on lecture notes by Dr. Eric M. Schwartz

Sequential Design Review:

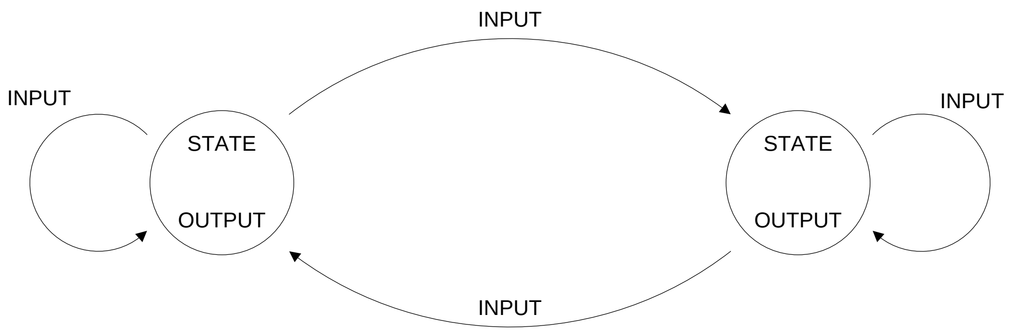
- A binary number can represent 2^n states, where n is the number of bits.
    - The number of bits required is determined by the number of states.
        - Ex. 4 states requires 2 bits ($2^2 = 4$ possible states)
        - Ex. 19 states requires 5 bits ($2^5 = 32$ possible states)
- One flip-flop is required per state bit.

Steps to Design Sequential Circuits:

1) Draw a State Diagram
2) Make a Next State Truth Table (NSTT)
3) Pick Flip-Flop type
4) Add Flip-Flop inputs to NSTT using Flip-Flop excitation equation
   (This creates an Excitation Table.)
5) Solve equations for Flip-Flop inputs (K-maps)
6) Solve equations for Flip-Flop outputs (K-maps)
7) Implement the circuit

Moore State Machines:

- Outputs determined **solely** by the current state
    - Outputs are *unconditional* (not directly dependent on input signals)

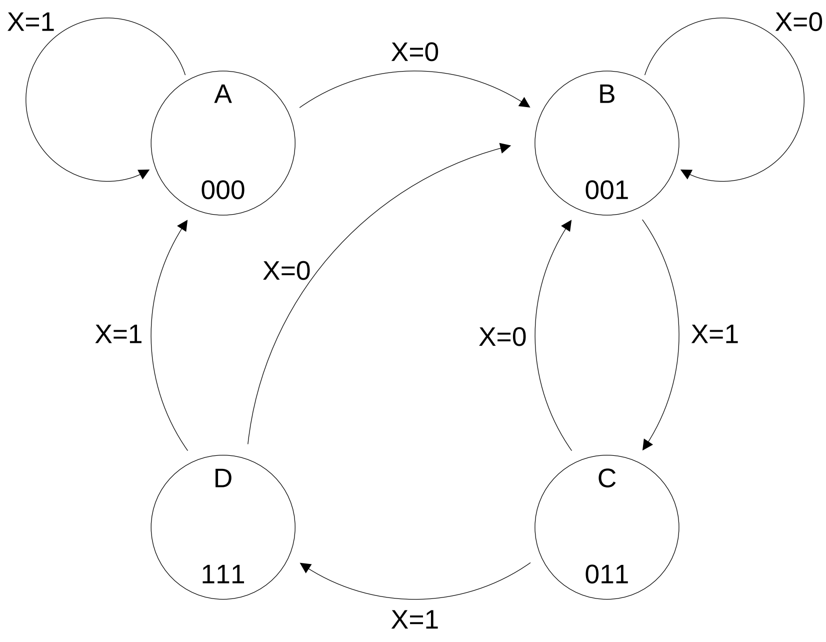


GENERIC MOORE STATE MACHINE

Note: This should look at lot like the counter designs done previously.

Example: Design a simple sequence detector for the sequence 011. Include three outputs that indicate how many bits have been received in the correct sequence. (For example, each output could be connected to an LED.)

1) Draw a State Diagram (Moore) and then assign binary State Identifiers.



STATES
A=00
B=01
C=11
D=10

MOORE SEQUENCE DETECTOR FOR 011

Note: State ‘A’ is the starting state for this diagram.

2) Make a Next State Truth Table (NSTT)

State	X	O_2	O_1	O_0	$State^+$
A	0	0	0	0	B
A	1	0	0	0	A
B	0	0	0	1	B
B	1	0	0	1	C
D	0	1	1	1	B
D	1	1	1	1	A
C	0	0	1	1	B
C	1	0	1	1	D

Q_1	Q_0	X	O_2	O_1	O_0	Q_1^+	Q_0^+
0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0
0	1	0	0	0	1	0	1
0	1	1	0	0	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	1	0	1	1	1	0

3) Pick Flip-Flop type
- Pick D Flip-Flop

4) Add Flip-Flop inputs to NSTT to make an excitation table

Q_1	Q_0	X	O_2	O_1	O_0	Q_1^+	Q_0^+	D_1	D_0
0	0	0	0	0	0	0	1	0	1
0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	1	0	1
0	1	1	0	0	1	1	1	1	1
1	0	0	1	1	1	0	1	0	1
1	0	1	1	1	1	0	0	0	0
1	1	0	0	1	1	0	1	0	1
1	1	1	0	1	1	1	0	1	0

5) Solve equations for Flip-Flop inputs (K-maps)

X\\Q_1Q_0	00	01	11	10
0	0	0	0	0
1	0	1	1	0

$$D_1 = XQ_0$$

X\\Q_1Q_0	00	01	11	10
0	1	1	1	1
1	0	1	0	0

$$D_0 = \overline{X} + \overline{Q}_1 Q_0$$

6) Solve equations for Flip-Flop outputs (K-maps)

Q_1\\Q_0	0	1
0	0	0
1	1	0

$$O_2 = Q_1\overline{Q}_0$$

Q_1\\Q_0	0	1
0	0	0
1	1	1

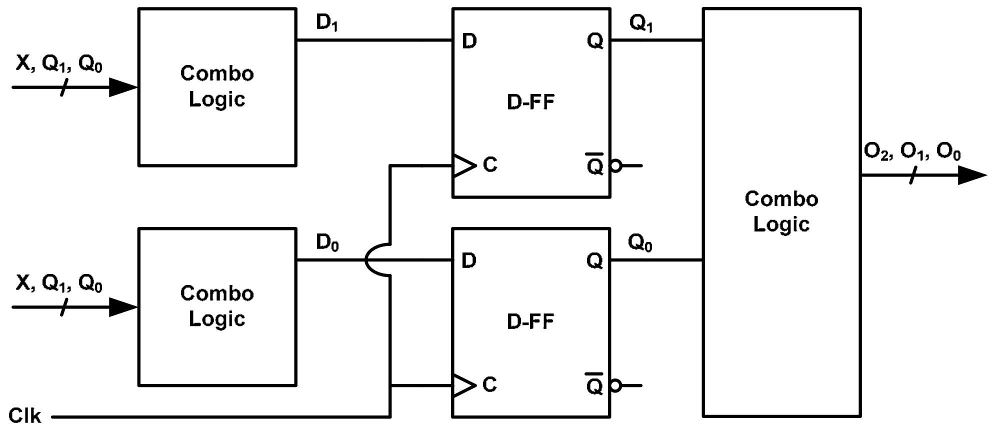
$$O_1 = Q_1$$

Q_1\\Q_0	0	1
0	0	1
1	1	1

$$O_0 = Q_1 + Q_0$$

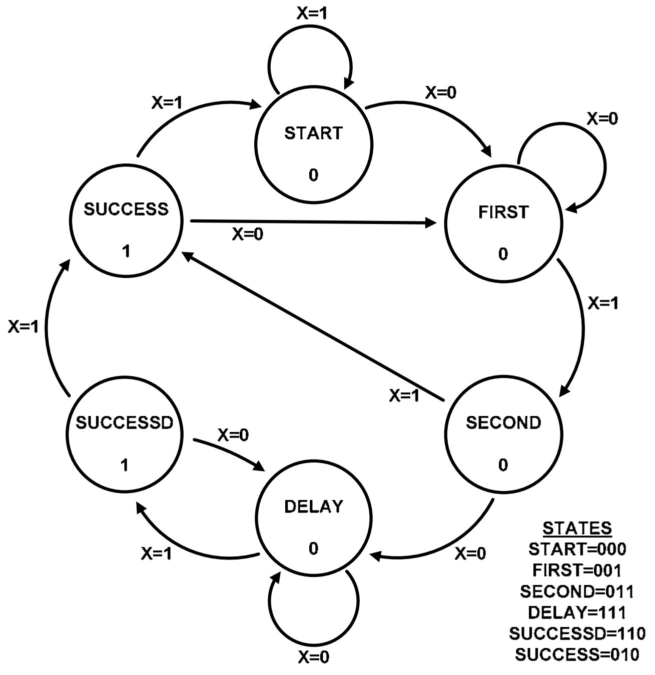
Note: Moore designs do not depend on the inputs, so X can be neglected.

7) Implement the circuit



Example: Design a sequence detector that searches for a series of binary inputs to satisfy the pattern 01[0*]1, where [0*] is any number of consecutive zeroes. The output (Z) should become true every time the sequence is found.

1) Draw a State Diagram (Moore) and then assign binary State Identifiers.



MOORE SEQUENCE DETECTOR FOR 01[0*]1
(WHERE [0*] IS ANY NUMBER OF ZEROES)

Recall: Picking state identifiers so that only one bit changes from state to state will generally help reduce the amount of hardware required for implementation. Only the transition from Success to First requires two bits to change.

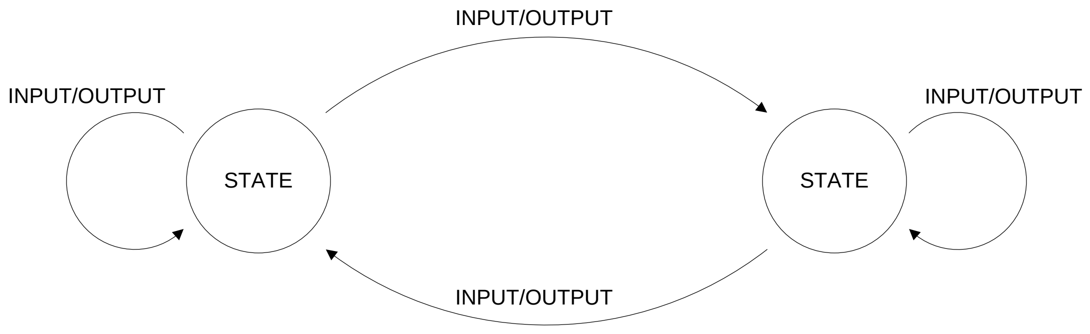
2) Make a Next State Truth Table (NSTT)

State	Q_2	Q_1	Q_0	X	Z	$State^+$	Q_2^+	Q_1^+	Q_0^+
Start	0	0	0	0	0	First	0	0	1
Start	0	0	0	1	0	Start	0	0	0
First	0	0	1	0	0	First	0	0	1
First	0	0	1	1	0	Second	0	1	1
Success	0	1	0	0	1	First	0	0	1
Success	0	1	0	1	1	Start	0	0	0
Second	0	1	1	0	0	Delay	1	1	1
Second	0	1	1	1	0	Success	0	1	0
unused	1	0	*	*	X	X	X	X	X
SuccessD	1	1	0	0	1	Delay	1	1	1
SuccessD	1	1	0	1	1	Success	0	1	0
Delay	1	1	1	0	0	Delay	1	1	1
Delay	1	1	1	1	0	SuccessD	1	1	0

3-7) Do the remainder of the design steps.

Mealy State Machines:

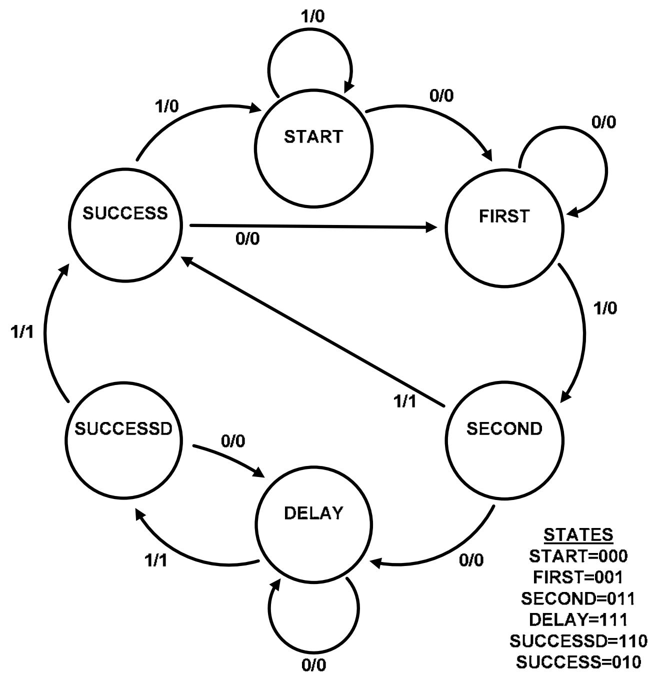
- Outputs determined by the current state **and** the current inputs.
    - Outputs are *conditional* (directly dependent on input signals)



GENERIC MEALY STATE MACHINE

Example: Design a sequence detector that searches for a series of binary inputs to satisfy the pattern 01[0*]1, where [0*] is any number of consecutive zeroes. The output (Z) should become true every time the sequence is found.

1) Draw a State Diagram (Mealy) and then assign binary State Identifiers.



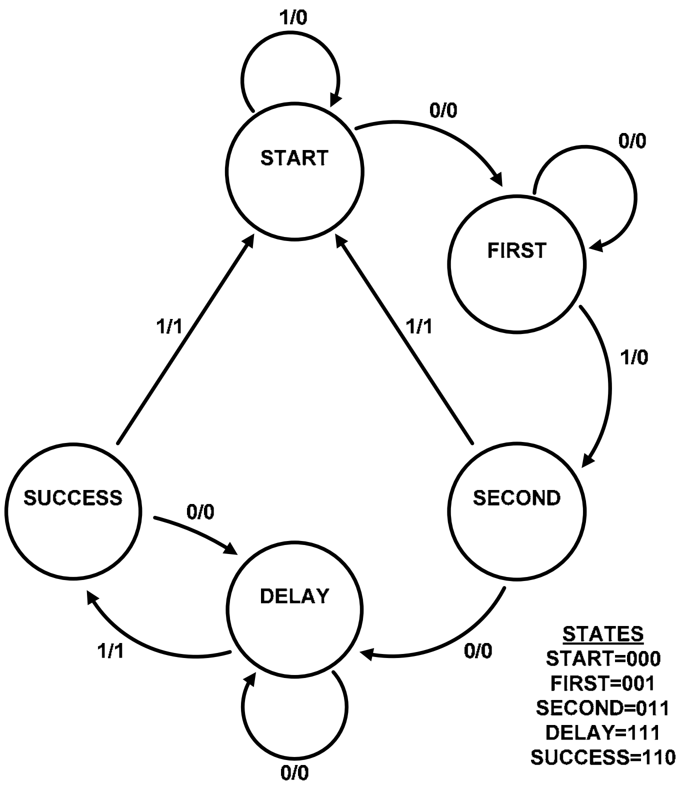
MEALY SEQUENCE DETECTOR FOR 01[0*]1
(WHERE [0*] IS ANY NUMBER OF ZEROES)
(DIRECTLY CONVERTED FROM MOORE STATE DIAGRAM)

Moore vs. Mealy Timing Comparison

Clock (CLK):	Initialize	1	2	3	4	5	6	7	8	9	A	B	C
Input (X):	_	1	0	0	1	0	0	1	0	1	1	1	0
Moore Output (Z):	0	0	0	0	0	0	0	0	1	0	1	1	0
Mealy Output (Z):	_	0	0	0	0	0	0	1	0	1	1	0	0
Current State (Q_i):	Start	Start	Start	First	First	Second	Delay	Delay	SuccD	Delay	SuccD	Succ	Start
Next State (Q_i^+):	_	Start	First	First	Second	Delay	Delay	SuccD	Delay	SuccD	Succ	Start	First

Note: The Moore Machine lags one clock cycle behind the final input in the sequence. The Mealy Machine can change asynchronously with the input.

One of the states in the previous Mealy State Diagram is unnecessary:



MEALY SEQUENCE DETECTOR FOR 01[0*]1
(WHERE [0*] IS ANY NUMBER OF ZEROES)

Note: The Mealy Machine requires one less state than the Moore Machine! This is possible because Mealy Machines make use of more information (i.e. inputs) than Moore Machines when computing the output. Having less states makes for an easier design because our truth tables, K-maps, and logic equations are generally less complex. In some cases, the reduction of states is significant because it reduces the number of flip-flops required for design implementation. In spite of the advantages of using a design with less states, **we will still use the 6-state Mealy Machine for the remainder of these notes to facilitate a direct comparison with the 6-state Moore Machine.**

2) Make a Next State Truth Table (NSTT)

State	Q_2	Q_1	Q_0	X	Z	$State^+$	Q_2^+	Q_1^+	Q_0^+
Start	0	0	0	0	0	First	0	0	1
Start	0	0	0	1	0	Start	0	0	0
First	0	0	1	0	0	First	0	0	1
First	0	0	1	1	0	Second	0	1	1
Success	0	1	0	0	0	First	0	0	1
Success	0	1	0	1	0	Start	0	0	0
Second	0	1	1	0	0	Delay	1	1	1
Second	0	1	1	1	1	Success	0	1	0
unused	1	0	*	*	X	X	X	X	X
SuccessD	1	1	0	0	0	Delay	1	1	1
SuccessD	1	1	0	1	1	Success	0	1	0
Delay	1	1	1	0	0	Delay	1	1	1
Delay	1	1	1	1	1	SuccessD	1	1	0

Note: This is identical to the Moore Machine, except for output Z.

3) Pick Flip-Flop type

Select D Flip-Flops..

4) Add Flip-Flop inputs to NSTT using Flip-Flop excitation equation

State	Q_2	Q_1	Q_0	X	Z	$State^+$	Q_2^+	Q_1^+	Q_0^+	D_2	D_1	D_0
Start	0	0	0	0	0	First	0	0	1	0	0	1
Start	0	0	0	1	0	Start	0	0	0	0	0	0
First	0	0	1	0	0	First	0	0	1	0	0	1
First	0	0	1	1	0	Second	0	1	1	0	1	1
Success	0	1	0	0	0	First	0	0	1	0	0	1
Success	0	1	0	1	0	Start	0	0	0	0	0	0
Second	0	1	1	0	0	Delay	1	1	1	1	1	1
Second	0	1	1	1	1	Success	0	1	0	0	1	0
unused	1	0	*	*	X	X	X	X	X	X	X	X
SuccessD	1	1	0	0	0	Delay	1	1	1	1	1	1
SuccessD	1	1	0	1	1	Success	0	1	0	0	1	0
Delay	1	1	1	0	0	Delay	1	1	1	1	1	1
Delay	1	1	1	1	1	SuccessD	1	1	0	1	1	0

5) Solve equations for Flip-Flop inputs (K-maps)

Q_2Q_1\$Q_0$X	00	01	11	10
00	0	0	0	0
01	0	0	0	1
11	1	0	1	1
10	X	X	X	X

$$D_2 = Q_2Q_0 + Q_2\overline{X} + Q_1Q_0\overline{X}$$

Q_2Q_1\$Q_0$X	00	01	11	10
00	0	0	1	0
01	0	0	1	1
11	1	1	1	1
10	X	X	X	X

$$D_1 = Q_2 + Q_1Q_0 + Q_0X$$

Q_2Q_1\$Q_0$X	00	01	11	10
00	1	0	1	1
01	1	0	0	1
11	1	0	0	1
10	X	X	X	X

$$D_0 = \overline{Q_0}\,\overline{X} + Q_0\overline{X} + \overline{Q_1}Q_0$$

Note: This is identical to the Moore Machine.

6) Solve equations for Flip-Flop outputs (K-maps)

Moore

$Q_2Q_1\backslash Q_0$	0	1
00	0	0
01	1	0
11	1	0
10	X	X

$$Z_{Moore} = Q_1\overline{Q_0}$$

Mealy

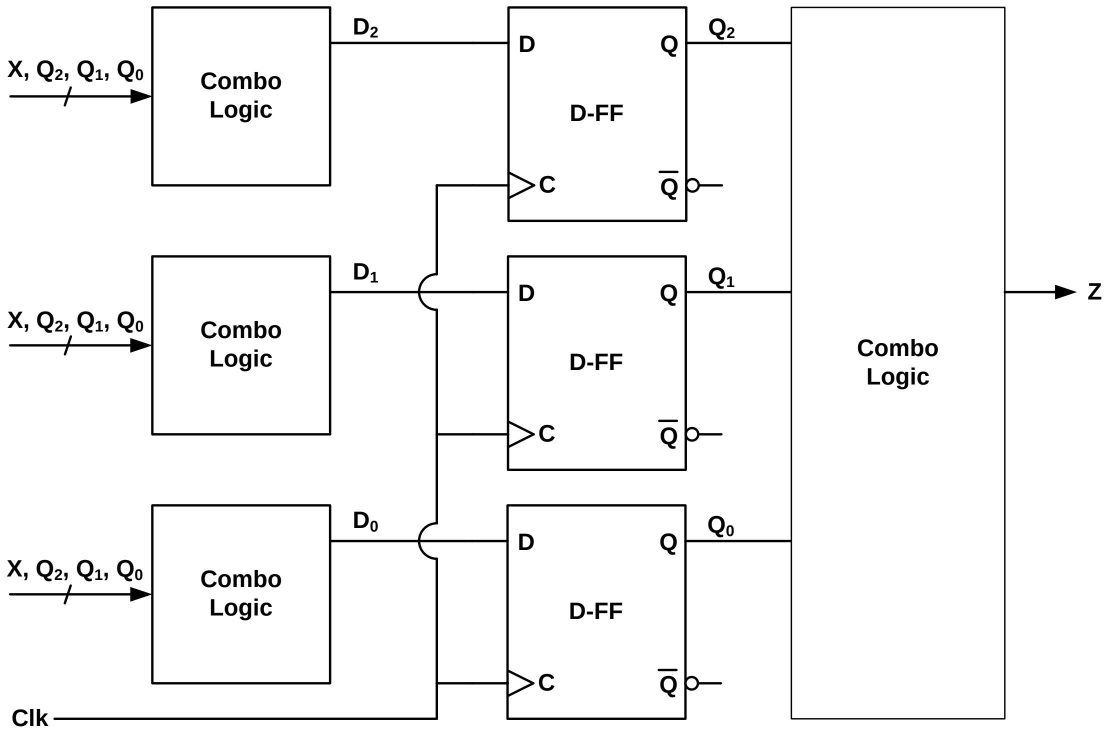
$Q_2Q_1\backslash Q_0X$	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	0	1	1	0
10	X	X	X	X

$$Z_{Mealy} = Q_2X + Q_1Q_0X$$

Recall: Moore outputs do **not** depend on the input.
- Z_{Moore} can only change when the state changes (synchronous).
- Z_{Mealy} can change asynchronously because it can change with X.

Note: The Moore and Mealy Machines solve the same problem.

7) Implement the circuit



Notes: The 3 boxes of combinational logic on the left are the same for both of the Moore and Mealy designs because the state transitions are the same. This would not have been the case had we implemented the 5-state Mealy Machine.

The larger box of combinational logic on the right is different for the Moore and Mealy designs because the output, Z, is computed differently.