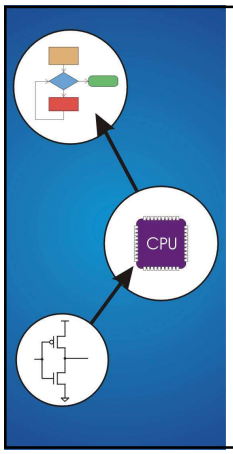


Transistors and State Machines

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

CMOS Circuit

Complementary MOS

Uses both n-type and p-type MOS transistors

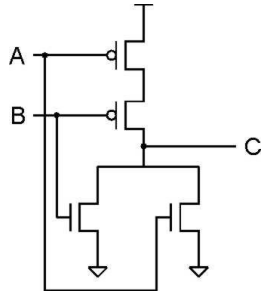
- **p-type**
  - **Attached to + voltage**
  - **Pulls output voltage UP when input is zero**
- **n-type**
  - **Attached to GND**
  - **Pulls output voltage DOWN when input is one**

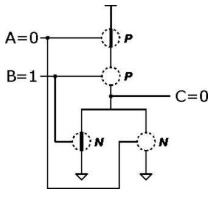
For all inputs, make sure that output is either connected to GND or to +, but not both!

3-2

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

NOR Gate



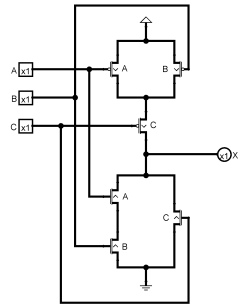


A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

Note: Serial structure on top, parallel on bottom.

3-3

Arbitrary Boolean Expression

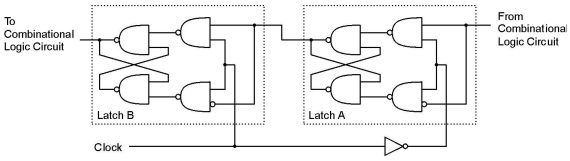


A	B	C	Out
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

NOT(C AND (A OR B))

Storage: Master-Slave Flipflop

A pair of gated D-latches,
to isolate *next* state from *current* state.



During 1st phase (clock=1), previously-computed state becomes *current* state and is sent to the logic circuit.

During 2nd phase (clock=0), *next* state, computed by logic circuit, is stored in Latch A.

Analyzing a FSM: Logic Circuit to State Diagram

1. **Describe combinational circuit outputs using Boolean algebra.**
2. **Create the state table (truth table) for all possible input/state combinations.**
   1. **Inputs: Input, Present State**
   2. **Outputs: Next State, Outputs (if different from State)**
3. **Produce a state diagram that satisfies the state table.**

Designing a FSM: Specification to Circuit

- Create a State Diagram from the specifications.
  - May need to clarify specifications
- Determine the number of flip-flops needed by assigning each state a unique binary combination.
- Create the State Table (truth table) for all possible input/state combinations.
  - Inputs: Input, Present State
  - Outputs: Next State, Outputs (if different from State)
- Create the combinational circuit from State Table
- Complete the circuit with by adding flip-flops to compinational circuit
- Simulate and verify the design.

Mealy vs Moore state machines

Moore: Outputs are only based on current state

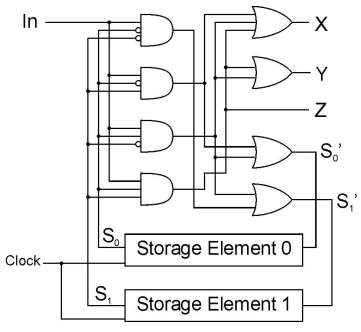
- Each state labeled with an output
- Outputs change only at clock edge following input change
- Potentially simpler to conceptualize
- Simpler to interconnect with other state machines
- Every Moore machine convertible to a Mealy machine

Mealy: Outputs are based on current state and inputs

- Each arc/transition labeled with a output
- Tend to have fewer states
- Outputs shown on transition arcs in state diagrams
- Output changes in the same cycle as input is received

https://en.wikipedia.org/wiki/Mealy_machine https://en.wikipedia.org/wiki/Moore_machine

Traffic Sign: Mealy or Moore?



Mealy: Output depends on State and Input.

Traffic Sign Truth Tables: Moore

Outputs
(depend only on state: S_1S_0)

Lights 1 and 2
Lights 3 and 4
Light 5

S_1	S_0	X	Y	Z
0	0	0	0	0
0	1	1	0	0
1	0	1	1	0
1	1	1	1	1

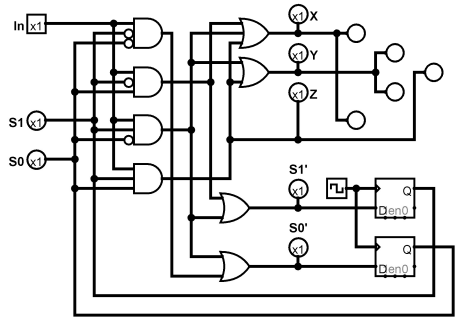
Next State: S_1' ,S_0'
(depend on state and input)

Switch

In	S_1	S_0	S_1'	S_0'
0	X	X	0	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	0

Whenever In=0, next state is 00.

Mealy sign arrow implementation in Logisim



Traffic Sign Truth Tables: Mealy

Next State: S_1' , S_0' , Outputs X, Y, Z
(depend on state and input)

Switch

Lights 1 and 2
Lights 3 and 4
Light 5

In	S_1	S_0	S_1'	S_0'	X	Y	Z
0	X	X	0	0	0	0	0
1	0	0	0	1	1	0	0
1	0	1	1	0	1	1	0
1	1	0	1	1	1	1	1
1	1	1	0	0	0	0	0

Whenever In=0, next state is 00.

Moore sign arrow implementation in Logisim