Computer Science 210

Computer Systems 1

Lecture 9

Finite State Machines

Credits: Slides prepared by Gregory T. Byrd, North Carolina State University

Finite State Machine

A description of a system with the following components:

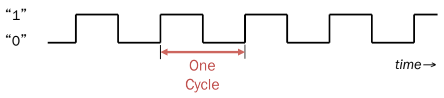
1. A finite number of states
2. A finite number of external inputs
3. A finite number of external outputs
4. An explicit specification of all state transitions
5. An explicit specification of what determines each external output value

Often described by a state diagram.
- Inputs trigger state transitions.
- Outputs are associated with each state (or with each transition).

CS210 2

The Clock

Frequently, a clock circuit triggers transition from one state to the next.



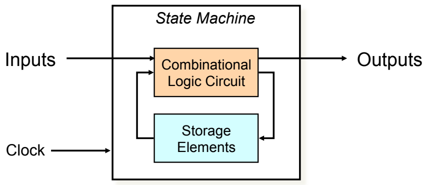
At the beginning of each clock cycle, state machine makes a transition, based on the current state and the external inputs

CS210 3

Implementing a Finite State Machine

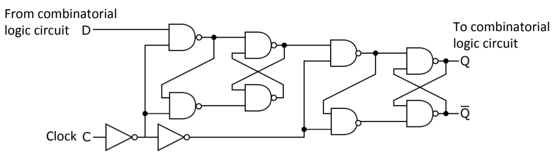
Combinational logic
- Determine outputs and next state

Storage elements
- Maintain state representation



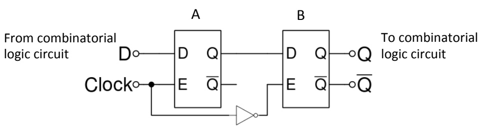
Storage: Master-Slave Flip flop

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



Storage: Master-Slave Flip flop

Master-slave edge triggered D flip-flop

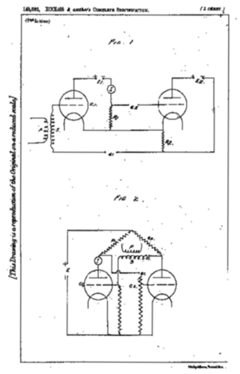


During 1st phase (clock=1), previously-computed state in A becomes *current* state in Latch B and is sent to the logic circuit.

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

Flip-Flops

- Many types
- Look at
  http://en.wikipedia.org/wiki/Flip-flop_(electronics)



Flip-flop schematics from the Eccles and Jordan patent filed 1918

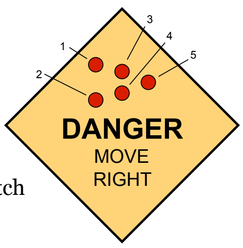
Storage

•Each master-slave flipflop stores one state bit.
•The number of storage elements (flipflops) needed is determined by the number of states
(and the representation of each state).

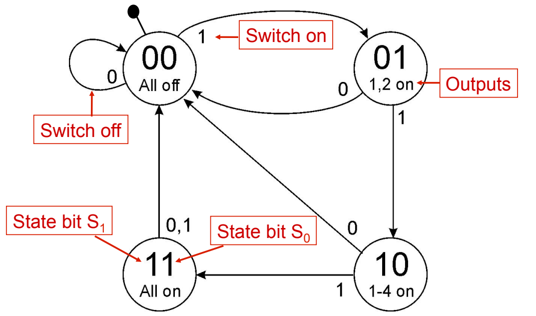
•Examples:
- Sequential lock
  - Four states – two bits
- Basketball scoreboard
  - 7 bits for each score, 5 bits for minutes, 6 bits for seconds, 1 bit for possession arrow, 1 bit for half, ...

Complete Example

•A blinking traffic sign
- No lights on
- 1 & 2 on
- 1, 2, 3, & 4 on
- 1, 2, 3, 4, & 5 on
- (repeat as long as switch is turned on)



Traffic Sign State Diagram



Transition on each clock cycle

Traffic Sign Truth Tables

Outputs
(depend only on state: S_1S_0)

Z: Lights 1 and 2; Y: Lights 3 and 4; X: Light 5

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

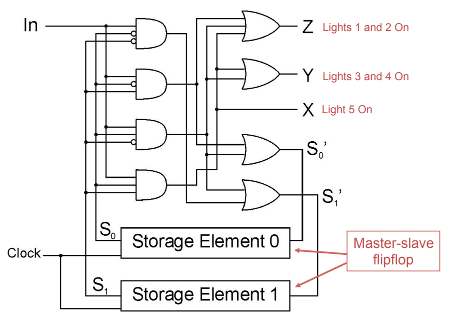
Next State: $S_1'S_0'$
(depend on state and input)

On: Switch

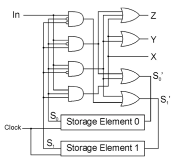
On	S_1	S_0	S_1'	S_0'
0	-	-	0	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	0

Whenever On = 0 (false), next state is 00 (off)
S_1 & S_0 are irrelevant

Traffic Sign Logic



Traffic Sign Logic



Of course our traffic sign controller has been implemented in hardware.

If wanted to change its operation we can't.

Implementing a generic controller in hardware that could be controlled by software would give us more flexibility but would be more complicated.

From Logic to Data Path

The data path of a computer is all the logic used to process information.
- See the data path of the LC-3 on next slide.

•Combinational Logic
- Decoders – convert *S* into control signals
- Multiplexers – select inputs and outputs
- ALU (Arithmetic and Logic Unit) – operations on data

•Sequential Logic
- State machine – coordinate control signals and data movement
- Registers and latches – storage elements

LC-3 Data Path