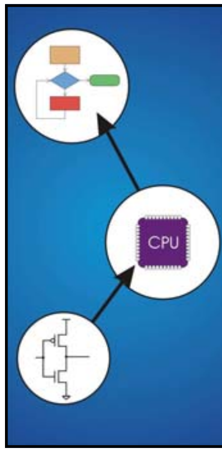
Introduction to Computer Engineering

ECE/CS 252, Fall 2010
Prof. Mikko Lipasti
Department of Electrical and Computer Engineering
University of Wisconsin – Madison



Chapter 3
Digital Logic Structures
- Part 2

Slides based on set prepared by
Gregory T. Byrd, North Carolina State University

Building Functions from Logic Gates

We've already seen how to implement truth tables using AND, OR, and NOT, etc. -- examples of *combinational logic*.

Combinational Logic Circuit
- **output depends only on the current inputs**
- **stateless**

Sequential Logic Circuit
- **output depends on the sequence of inputs (past and present)**
- **stores information (state) from past inputs**

Next we'll show how to use sequential circuits to store information.

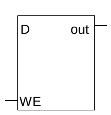
Combinational vs. Sequential

Combinational Circuit
- **always gives the same output for a given set of inputs**
  - **ex: adder always generates sum and carry, regardless of previous inputs**

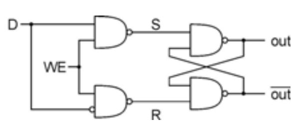
Sequential Circuit
- **stores information**
- **output depends on stored information (state) plus input**
  - **so a given input might produce different outputs, depending on the stored information**
- ***example:* ticket counter**
  - **advances when you push the button**
  - **output depends on previous state**
- **useful for building "memory" elements and "state machines"**

Gated D-Latch

Two inputs: D (data) and WE (write enable)
- **when WE = 1, latch is set to value of D**
- **when WE = 0, latch holds previous value**

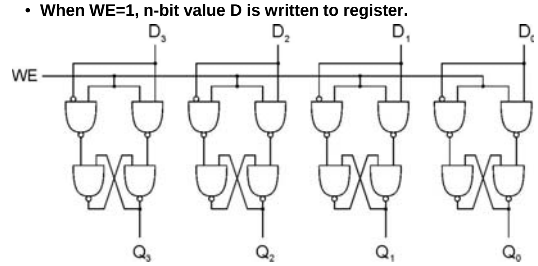


Watch online lecture to understand internals:



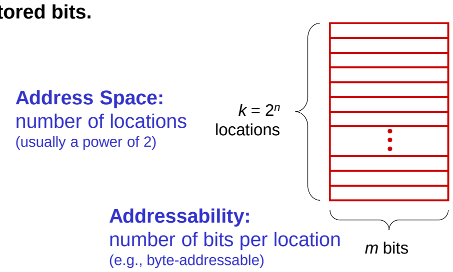
Register

A register stores a multi-bit value.
- **We use a collection of D-latches, all controlled by a common WE.**
- **When WE=1, n-bit value D is written to register.**



Memory: 2-dimensional register

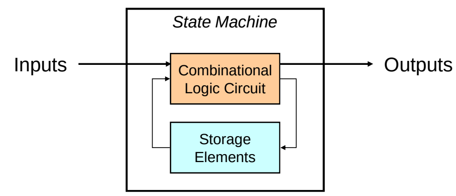
We can build a memory – a logical $k \times m$ array of stored bits.



State Machine

Important type of sequential circuit

- **Combines combinational logic with storage**
- **"Remembers" state, and changes output (and state) based on inputs and current state**



Combinational vs. Sequential

Two types of "combination" locks

Combinational
Success depends only on the values, not the order in which they are set.

Sequential
Success depends on the sequence of values (e.g, R-13, L-22, R-3).

State

The state of a system is a snapshot of all the relevant elements of the system at the moment the snapshot is taken.

Examples:

- **The state of a basketball game can be represented by the scoreboard.**
  - **Number of points, time remaining, possession, etc.**
- **The state of a tic-tac-toe game can be represented by the placement of X's and O's on the board.**

State of Sequential Lock

Our lock example has four different states, labelled A-D:

A: The lock is not open, and no relevant operations have been performed.

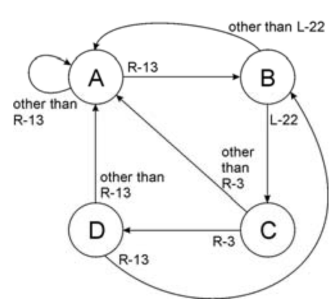
B: The lock is not open, and the user has completed the R-13 operation.

C: The lock is not open, and the user has completed R-13, followed by L-22.

D: The lock is open.

State Diagram

Shows states and actions that cause a transition between states.



Finite State Machine

A description of a system with the following components:

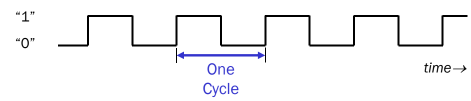
1. **A finite number of states**
2. **A finite number of external inputs**
3. **A finite number of external outputs**
4. **An explicit specification of all state transitions**
5. **An explicit specification of what causes each external output value.**

Often described by a state diagram.

- **Inputs may cause state transitions.**
- **Outputs are associated with each state (or with each transition).**

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.

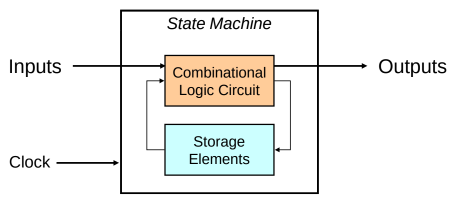
Implementing a Finite State Machine

Combinational logic

- **Determine outputs and next state.**

Storage elements

- **Maintain state representation.**



Storage

Each storage element remembers one state bit.

The number of storage elements needed is determined by the number of states (and the representation of each state).

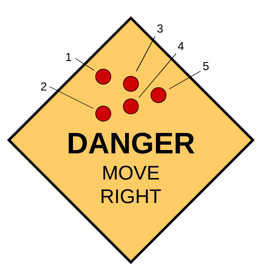
Examples:

- **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, …**

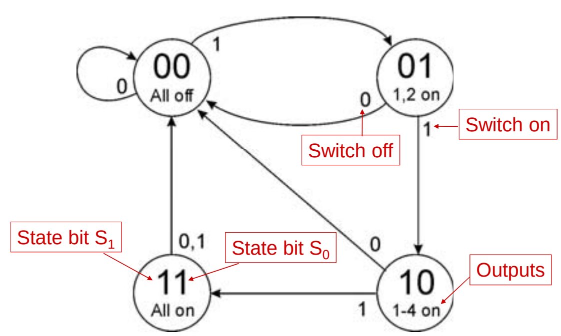
Complete Example

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)

Lights 1 and 2 (Z)
Lights 3 and 4 (Y)
Light 5 (X)

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

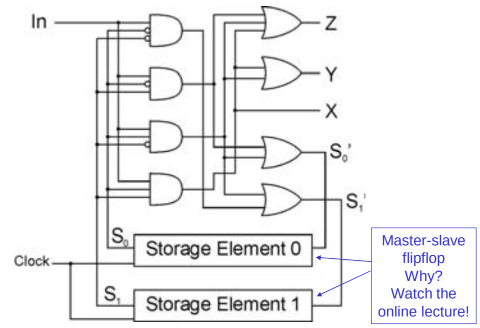
Next State: $S_1'S_0'$
(depend on state and input)

Switch (In)

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.

Traffic Sign Logic



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.**

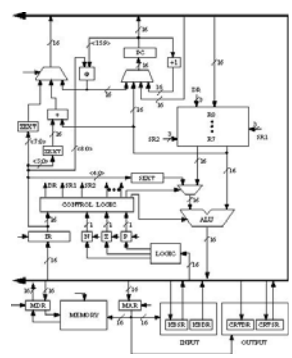
Combinational Logic

- **Decoders -- convert instructions 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



Summary

Sequential Logic Circuits

Storage/Memory

- **D Latch**
- **Register**
- **Memory**
- **Watch online lecture for more details**

Finite State Machines

- **State Diagram**
- **Output Logic**
- **Next State Logic**

LC-3 Datapath