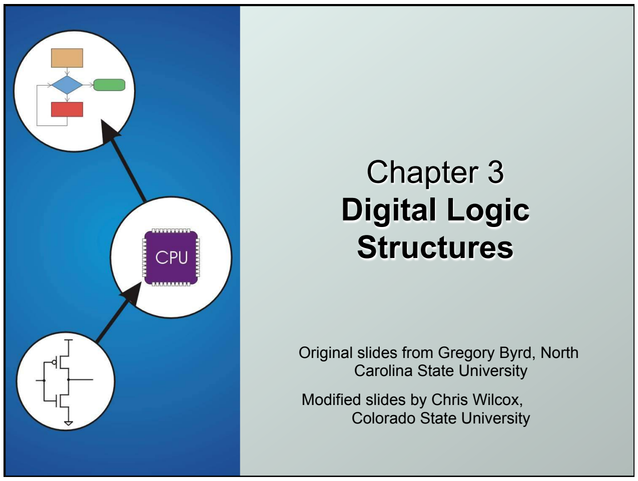
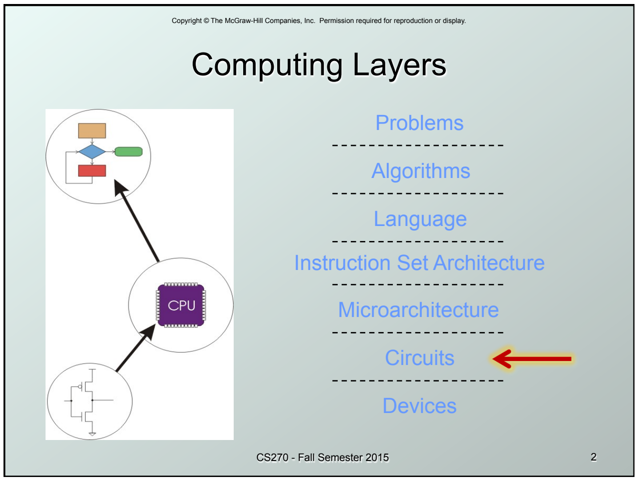


Chapter 3
Digital Logic Structures

Original slides from Gregory Byrd, North Carolina State University

Modified slides by Chris Wilcox, Colorado State University



Computing Layers

Problems

Algorithms

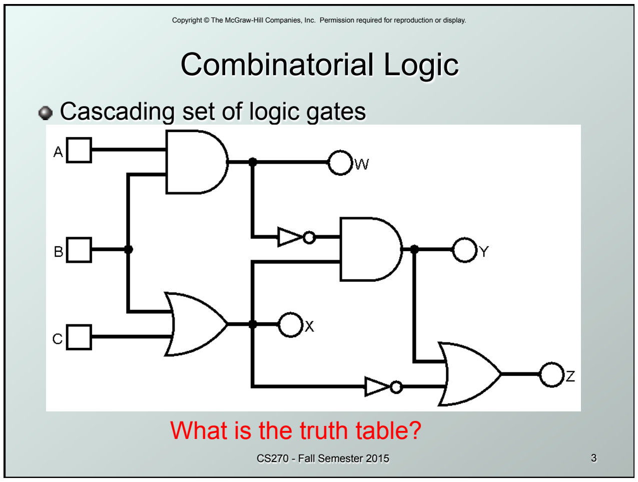
Language

Instruction Set Architecture

Microarchitecture

Circuits ←

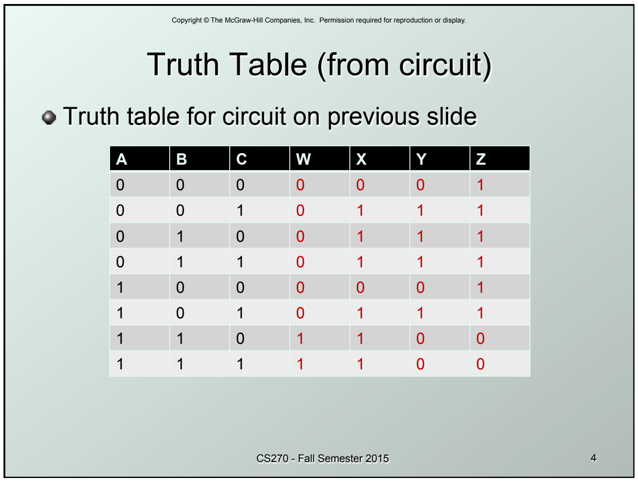
Devices



Combinatorial Logic

- Cascading set of logic gates

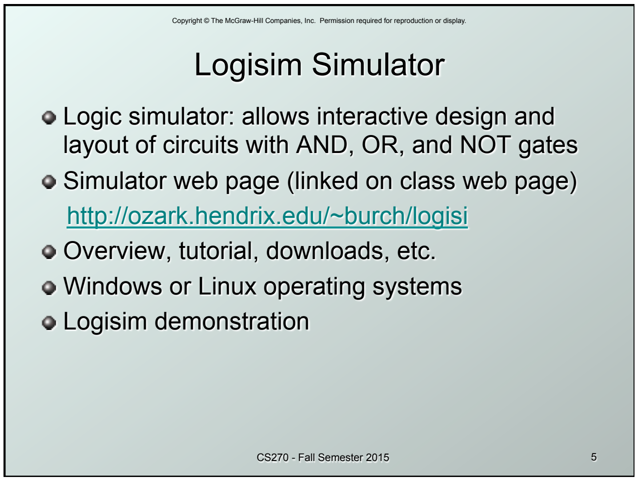
What is the truth table?



Truth Table (from circuit)

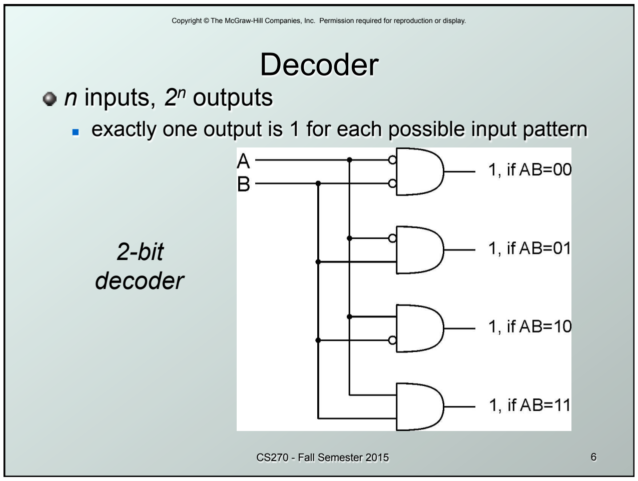
- Truth table for circuit on previous slide

A	B	C	W	X	Y	Z
0	0	0	0	0	0	1
0	0	1	0	1	1	1
0	1	0	0	1	1	1
0	1	1	0	1	1	1
1	0	0	0	0	0	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	1	1	0	0



Logisim Simulator

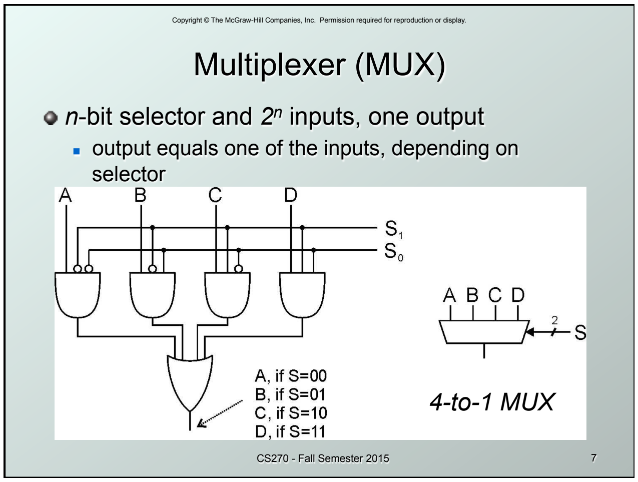
- Logic simulator: allows interactive design and layout of circuits with AND, OR, and NOT gates
- Simulator web page (linked on class web page)
  http://ozark.hendrix.edu/~burch/logisi
- Overview, tutorial, downloads, etc.
- Windows or Linux operating systems
- Logisim demonstration



Decoder

- *n* inputs, 2^n outputs
  - exactly one output is 1 for each possible input pattern

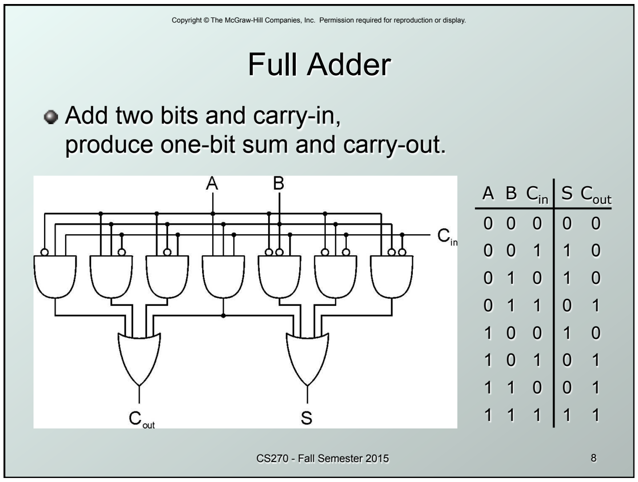
2-bit decoder



Multiplexer (MUX)

- *n*-bit selector and 2^n inputs, one output
  - output equals one of the inputs, depending on selector

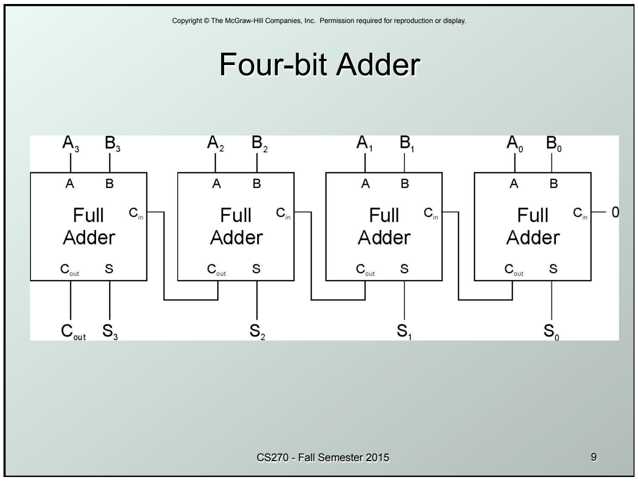
4-to-1 MUX



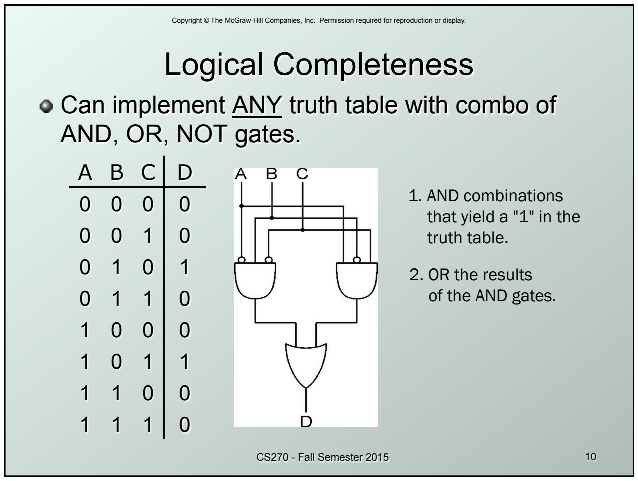
Full Adder

- Add two bits and carry-in, produce one-bit sum and carry-out.

A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Four-bit Adder

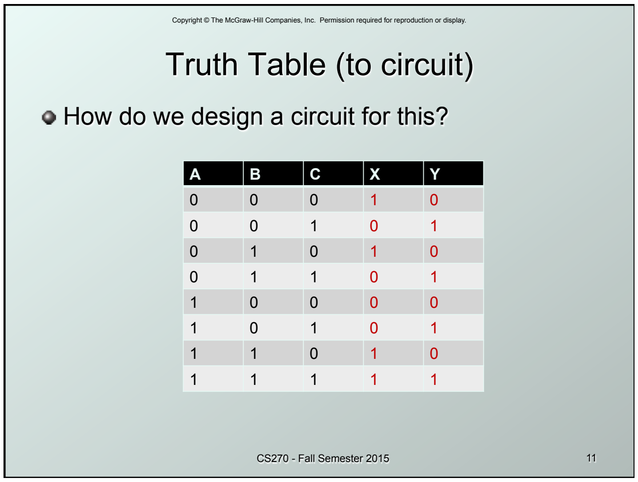


Logical Completeness

- Can implement ANY truth table with combo of AND, OR, NOT gates.

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

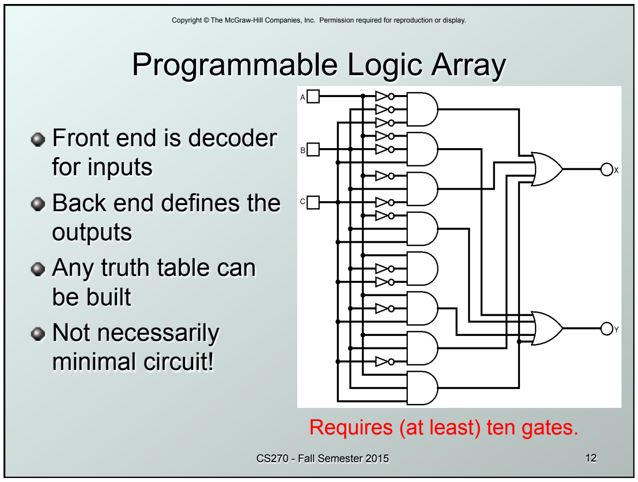
1. AND combinations that yield a "1" in the truth table.
2. OR the results of the AND gates.



Truth Table (to circuit)

- How do we design a circuit for this?

A	B	C	X	Y
0	0	0	1	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1



Programmable Logic Array

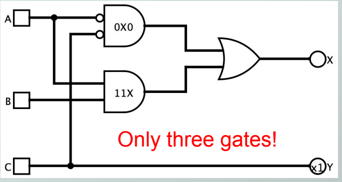
- Front end is decoder for inputs
- Back end defines the outputs
- Any truth table can be built
- Not necessarily minimal circuit!

Requires (at least) ten gates.

Circuit Minimization

- Gate array design has one unused AND gate.
- Boolean logic lets us reduce even further:
  - $X = \overline{ABC} + \overline{A}B\overline{C} + AB\overline{C} + ABC =$
    $= \overline{AC} + AB$
  - $Y = \overline{AB}C + \overline{A}BC + A\overline{B}C + ABC = C$

A	B	C	X	Y
0	0	0	1	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1



Only three gates!

Looking Ahead: C Structures

- Useful for data structures

```
 struct student
{
   char *lastName;
   char *firstName;
   Date birthDate;
   …
};
struct student s;
s.lastname = (char *)malloc(80);
strcpy(s.lastname, “Smith”);
```

Looking Ahead: Dynamic Memory

- Static versus dynamic memory allocation:

```
// static allocation
char name[80];
strcpy(name, “Smith”);
printf(“Name: %s\n”, name);
// dynamic allocation
char *name = (char *)malloc(80);
strcpy(name, “Smith”);
printf(“Name: %s\n”, name);
free(name);
```

Looking Ahead: String Tokens

- How to extract tokens from a string:

```
char *token = strtok(string, “ \t”);
while (token != null)
{
  tokens[numTokens] = (char *)
        malloc(strlen(token)+1);
  strcpy(tokens[numTokens], token);
  token = strtok(NULL, “ \t”);
  numTokens++;
}
```