

## Chapter 3 Digital Logic Structures

Original slides from Gregory Byrd, North Carolina State University  
Modified slides by Chris Wilcox, Colorado State University

The diagram on the left shows a hierarchy of abstraction. At the bottom is a logic gate symbol. An arrow points up to a CPU chip icon. Another arrow points up to a block diagram of a system with various components.

## Computing Layers

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

Problems  
-----  
Algorithms  
-----  
Language  
-----  
Instruction Set Architecture  
-----  
Microarchitecture  
-----  
Circuits ←  
-----  
Devices

CS270 - Spring Semester 2016 2

The diagram on the left is identical to the one on slide 1, showing a logic gate, a CPU, and a system block diagram.

## Combinational Logic

- Cascading set of logic gates

The circuit diagram shows three inputs: A, B, and C. Input A goes to an AND gate and an OR gate. Input B goes to the same AND gate and the same OR gate. Input C goes to the OR gate. The AND gate output is W. The OR gate output is X. W goes to an AND gate along with the inverted output of the OR gate (X-bar). The output of this second AND gate is Y. Y goes to an OR gate along with the inverted output of the OR gate (X-bar). The output of this third OR gate is Z.

What is the truth table?

CS270 - Spring Semester 2016 3

## 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

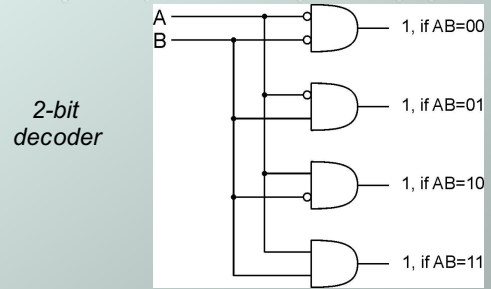
CS270 - Spring Semester 2016 4

## 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/logisim>
- Overview, tutorial, downloads, etc.
- Windows or Linux operating systems
- Logisim demonstration

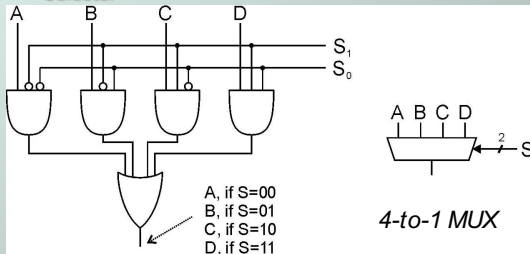
## Decoder

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



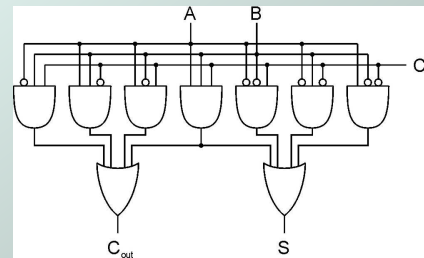
## Multiplexer (MUX)

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



## Full Adder

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



A	B	C <sub>in</sub>	S	C <sub>out</sub>
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

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

### Four-bit Adder

CS270 - Spring Semester 2016 9

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

### 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

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

CS270 - Spring Semester 2016 10

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

### 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

CS270 - Spring Semester 2016 11

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

### Programmable Logic Array

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

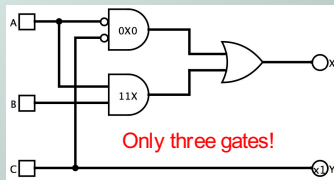
Requires (at least) ten gates.

CS270 - Spring Semester 2016 12

## Circuit Minimization

- Gate array design has one unused AND gate.
- Boolean logic lets us reduce even further:

$$\begin{aligned}
 X &= \overline{A}BC + \overline{A}B\overline{C} + A\overline{B}C + ABC = \\
 &= \overline{A}C + AB \\
 Y &= \overline{A}BC + \overline{A}B\overline{C} + A\overline{B}C + ABC = C
 \end{aligned}$$



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

CS270 - Spring Semester 2016

13

## 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");
  
```

CS270 - Spring Semester 2016

14

## 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);
  
```

CS270 - Spring Semester 2016

15

## 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++;
}
  
```

CS270 - Spring Semester 2016

16