Chapter 3
Digital Logic Structures

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

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

CS270 - Fall Semester 2014

Computing Layers

Problems
Algorithms
Language
Instruction Set Architecture
Microarchitecture
Circuits
Devices

CS270 - Fall Semester 2014

Combinatorial Logic

Cascading set of logic gates

What is the truth table?

Truth Table (from circuit)

Truth table for circuit on previous slide

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>W</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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

CS270 - Fall Semester 2014
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

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.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Sout</th>
<th>Cout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Four-bit Adder

Logical Completeness

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?

Programmable Logic Array

1. Front end is a input decode
2. Back end selects outputs
3. Not necessarily minimal circuit!
4. Logic arrays are prebuilt
Looking Ahead: C Structures

- Useful for data structures

```c
struct student
{
    char *lastName;
    char *firstName;
    Date birthDate;
};
```

```c
struct student s;
s.lastname = (char *)malloc(80);
strcpy(s.lastname, "Smith");
```

Looking Ahead: Dynamic Memory

- Static versus dynamic memory allocation:

```c
// 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:

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