CprE 281: Digital Logic

Instructor: Alexander Stoytchev

http://www.ece.iastate.edu/~alexs/classes/
Intro to Verilog
Administrative Stuff

- HW3 is due on Monday Sep 14 @ 4p
Administrative Stuff

• HW4 is out

• It is due on Monday Sep 21 @ 4pm.

• Please write clearly on the first page (in BLOCK CAPITAL letters) the following three things:
  - Your First and Last Name
  - Your Student ID Number
  - Your Lab Section Letter

• Also, please
  - Staple your pages
  - Use Letter-sized sheets
Administrative Stuff

• Midterm Exam #1

• When: Friday Sep 25.

• Where: This classroom

• What: Chapter 1 and Chapter 2 plus number systems

• The exam will be open book and open notes (you can bring up to 3 pages of handwritten notes).

• More details to follow.
Quick Review
NAND followed by NOT = AND

\[
x_1 \cdot x_2 \quad \overline{x_1 \cdot x_2} \quad x_1 \cdot x_2
\]

<table>
<thead>
<tr>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
DeMorgan’s Theorem

15a. \( x \cdot y = x + y \)
DeMorgan’s Theorem

15a. \( x \cdot y = \overline{x + y} \)

\[ \begin{array}{ll}
\text{AND gate:} & x \cdot y \quad \text{=}
\end{array} \]

\[ \begin{array}{ll}
\text{OR gate:} & x + y \quad \text{=}
\end{array} \]
Sum-Of-Products

\[ x_1 \cdot x_2 \cdot x_3 \cdot x_4 \]
Sum-Of-Products

\[ x_1 \times x_2 \times x_3 \times x_4 \text{ AND OR } \]

AND

\[ x_1, x_2 \]

AND

\[ x_3, x_4 \]

OR
Sum-Of-Products

\[ x_1 \cdot x_2 + x_3 \cdot x_4 \]
Sum-Of-Products

\[ x_1 \cdot x_2 + x_3 \cdot x_4 \]
Sum-Of-Products

$\sum \prod x_1 x_2 x_3 x_4$
Sum-Of-Products

\[ x_1 \cdot x_2 + x_3 \cdot x_4 \]
Sum-Of-Products

\[ x_1 \cdot x_2 + x_3 \cdot x_4 \]
Sum-Of-Products
2-1 Multiplexer (Definition)

• Has two inputs: \( x_1 \) and \( x_2 \)

• Also has another input line \( s \)

• If \( s=0 \), then the output is equal to \( x_1 \)

• If \( s=1 \), then the output is equal to \( x_2 \)
Graphical Symbol for a 2-1 Multiplexer
Let’s Derive the SOP form

<table>
<thead>
<tr>
<th>$s , x_1 , x_2$</th>
<th>$f(s, x_1, x_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>1</td>
</tr>
<tr>
<td>011</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>0</td>
</tr>
<tr>
<td>111</td>
<td>1</td>
</tr>
</tbody>
</table>

$f(s, x_1, x_2) = \bar{s} \, x_1 \, x_2 + \bar{s} \, x_1 \, x_2 + s \, \bar{x_1} \, x_2 + s \, x_1 \, x_2$
Let’s simplify this expression

\[ f(s, x_1, x_2) = \overline{s} x_1 \overline{x}_2 + \overline{s} x_1 x_2 + s \overline{x}_1 x_2 + s x_1 x_2 \]

\[ f(s, x_1, x_2) = \overline{s} x_1 (\overline{x}_2 + x_2) + s (\overline{x}_1 + x_1)x_2 \]

\[ f(s, x_1, x_2) = \overline{s} x_1 + s x_2 \]
Circuit for 2-1 Multiplexerer

(b) Circuit

(c) Graphical symbol

[ Figure 2.33b-c from the textbook ]
Analogy: Railroad Switch

http://en.wikipedia.org/wiki/Railroad_switch]
Analogy: Railroad Switch

http://en.wikipedia.org/wiki/Railroad_switch]
Analogy: Railroad Switch

This is not a perfect analogy because the trains can go in either direction, while the multiplexer would only allow them to go from top to bottom.

http://en.wikipedia.org/wiki/Railroad_switch
More Compact Truth-Table Representation

(a) Truth table

<table>
<thead>
<tr>
<th>s x₁ x₂</th>
<th>f(s, x₁, x₂)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>1</td>
</tr>
<tr>
<td>0 1 1</td>
<td>1</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1</td>
</tr>
</tbody>
</table>

[ Figure 2.33 from the textbook ]
4-1 Multiplexer (Definition)

- Has four inputs: $w_0, w_1, w_2, w_3$
- Also has two select lines: $s_1$ and $s_0$
- If $s_1=0$ and $s_0=0$, then the output $f$ is equal to $w_0$
- If $s_1=0$ and $s_0=1$, then the output $f$ is equal to $w_1$
- If $s_1=1$ and $s_0=0$, then the output $f$ is equal to $w_2$
- If $s_1=1$ and $s_0=1$, then the output $f$ is equal to $w_3$

We’ll talk more about this when we get to chapter 4, but here is a quick preview.
Graphical Symbol and Truth Table

(a) Graphic symbol

(b) Truth table

<table>
<thead>
<tr>
<th>$s_1$</th>
<th>$s_0$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$w_0$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$w_1$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$w_2$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$w_3$</td>
</tr>
</tbody>
</table>

[ Figure 4.2a-b from the textbook ]
The long-form truth table

<table>
<thead>
<tr>
<th>S₁ S₀</th>
<th>I₃ I₂ I₁ I₀</th>
<th>F</th>
<th>S₁ S₀</th>
<th>I₃ I₂ I₁ I₀</th>
<th>F</th>
<th>S₁ S₀</th>
<th>I₃ I₂ I₁ I₀</th>
<th>F</th>
<th>S₁ S₀</th>
<th>I₃ I₂ I₁ I₀</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0 0 0 0 0</td>
<td>0</td>
<td>0 1</td>
<td>0 0 0 1 0</td>
<td>0</td>
<td>1 0</td>
<td>0 0 0 0 0</td>
<td>0</td>
<td>1 1</td>
<td>0 0 0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>1</td>
<td></td>
<td>0 1 0 0</td>
<td>0 1 0 0 0</td>
<td>0</td>
<td>1 0 1 1</td>
<td>0 1 0 1 1</td>
<td>0</td>
<td>1 1 0 0</td>
<td>0 1 0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>1</td>
<td></td>
<td>1 0 0 0</td>
<td>1 0 0 0 1</td>
<td>0</td>
<td>1 0 1 1</td>
<td>1 0 1 1 1</td>
<td>0</td>
<td>1 1 0 1</td>
<td>1 0 1 1 1</td>
<td>0</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>0</td>
<td></td>
<td>1 1 0 0</td>
<td>1 1 0 1 1</td>
<td>0</td>
<td>1 1 1 1</td>
<td>1 1 1 1 1</td>
<td>0</td>
<td>1 1 1 1</td>
<td>1 1 1 1 1</td>
<td>0</td>
</tr>
</tbody>
</table>

[http://www.absoluteastronomy.com/topics/Multiplexer]
4-1 Multiplexer (SOP circuit)
Using three 2-to-1 multiplexers to build one 4-to-1 multiplexer

[ Figure 4.3 from the textbook ]
Analogy: Railroad Switches

http://en.wikipedia.org/wiki/Railroad_switch]
Analogy: Railroad Switches

http://en.wikipedia.org/wiki/Railroad_switch
Analogy: Railroad Switches

these two switches are controlled together

http://en.wikipedia.org/wiki/Railroad_switch
Using three 2-to-1 multiplexers to build one 4-to-1 multiplexer
Using three 2-to-1 multiplexers to build one 4-to-1 multiplexer
That is different from the SOP form of the 4-1 multiplexer shown below, which uses less gates.
16-1 Multiplexer

[ Figure 4.4 from the textbook ]
7-Segment Display Example
Display of numbers

(a) Logic circuit and 7-segment display

<table>
<thead>
<tr>
<th>$s_1$</th>
<th>$s_0$</th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$d$</th>
<th>$e$</th>
<th>$f$</th>
<th>$g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<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>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

(b) Truth table

[ Figure 2.34 from the textbook ]
## Display of numbers

<table>
<thead>
<tr>
<th>$s_1$</th>
<th>$s_0$</th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$d$</th>
<th>$e$</th>
<th>$f$</th>
<th>$g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</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>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
### Display of numbers

<table>
<thead>
<tr>
<th></th>
<th>$s_1$</th>
<th>$s_0$</th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$d$</th>
<th>$e$</th>
<th>$f$</th>
<th>$g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</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>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>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- $a = \overline{s_0}$
- $c = \overline{s_1}$
- $e = \overline{s_0}$
- $g = s_1 \overline{s_0}$

- $b = 1$
- $d = \overline{s_0}$
- $f = \overline{s_1} \overline{s_0}$
Intro to Verilog
History

• Created in 1983/1984

• Verilog-95  (IEEE standard 1364-1995)

• Verilog 2001  (IEEE Standard 1364-2001)

• Verilog 2005  (IEEE Standard 1364-2005)

• SystemVerilog

HDL

- Hardware Description Language
- Verilog HDL
- VHDL
Verilog HDL != VHDL

- These are two different Languages!
- Verilog is closer to C
- VHDL is closer to Ada
Design conception

DESIGN ENTRY

- Schematic capture
- Verilog

Synthesis

Functional simulation

Design correct?

Yes

Physical design

Timing simulation

No

Timing requirements met?

No

Chip configuration

[ Figure 2.35 from the textbook ]
“Hello World” in Verilog

```verilog
module main;
  initial
    begin
        $display("Hello world!");
        $finish;
    end
endmodule
```

The Three Basic Logic Gates

NOT gate

AND gate

OR gate

[ Figure 2.8 from the textbook ]
How to specify a NOT gate in Verilog

\[ x \rightarrow \neg x \]

NOT gate
How to specify a NOT gate in Verilog

we’ll use the letter y for the output

\[ x \rightarrow \neg \rightarrow y \]

NOT gate
How to specify a NOT gate in Verilog

$\overline{x}$  

not (y, x)

NOT gate  

Verilog code
How to specify an AND gate in Verilog

\[ f = x_1 \cdot x_2 \]

AND gate

Verilog code

and (f, x1, x2)
How to specify an OR gate in Verilog

\[ f = x_1 + x_2 \]

OR gate

Verilog code

or (f, x1, x2)
2-1 Multiplexer
Verilog Code for a 2-1 Multiplexer

```
module example1 (x1, x2, s, f);
    input  x1, x2, s;
    output f;
    not (k, s);
    and (g, k, x1);
    and (h, s, x2);
    or (f, g, h);
endmodule
```
Verilog Code for a 2-1 Multiplexer

```verilog
module example3 (x1, x2, s, f);
    input x1, x2, s;
    output f;

    assign f = (~s & x1) | (s & x2);
endmodule
```
Verilog Code for a 2-1 Multiplexer

```verilog
// Behavioral specification
module example5 (x1, x2, s, f);
  input x1, x2, s;
  output f;
  reg f;

  always @(x1 or x2 or s)
  begin
    if (s == 0)
      f = x1;
    else
      f = x2;
  end
endmodule
```

[ Figure 2.36 from the textbook ]
Verilog Code for a 2-1 Multiplexer

// Behavioral specification
module example5 (input x1, x2, s, output reg f);

    always @(x1, x2, s)
        if (s == 0)
            f = x1;
        else
            f = x2;

endmodule
Another Example
Let’s Write the Code for This Circuit

[ Figure 2.39 from the textbook ]
Let’s Write the Code for This Circuit

module example2 (x1, x2, x3, x4, f, g, h);
input x1, x2, x3, x4;
output f, g, h;
and (z1, x1, x3);
and (z2, x2, x4);
or (g, z1, z2);
or (z3, x1, ~x3);
or (z4, ~x2, x4);
and (h, z3, z4);
or (f, g, h);
endmodule
Let’s Write the Code for This Circuit

module example4 (x1, x2, x3, x4, f, g, h);
  input x1, x2, x3, x4;
  output f, g, h;
  assign g = (x1 & x3) | (x2 & x4);
  assign h = (x1 | ~x3) & (~x2 | x4);
  assign f = g | h;
endmodule
Yet Another Example
A logic circuit with two modules

[ Figure 2.44 from the textbook ]
The adder module

\[
\begin{array}{cccc}
  a & + & b & s_1 \ s_0 \\
  0 & + & 0 & 0 \ 0 \\
  0 & + & 1 & 0 \ 1 \\
  1 & + & 0 & 1 \ 0 \\
  1 & + & 1 & 1 \ 0 \\
\end{array}
\]

(a) Evaluation of \( S = a + b \)

(b) Truth table

(c) Logic network

[ Figure 2.12 from the textbook ]
The adder module

// An adder module
module adder (a, b, s1, s0);
    input a, b;
    output s1, s0;
    assign s1 = a & b;
    assign s0 = a ^ b;
endmodule

[ Figure 2.45 from the textbook ]
The display module

<table>
<thead>
<tr>
<th></th>
<th>$s_1$</th>
<th>$s_0$</th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$d$</th>
<th>$e$</th>
<th>$f$</th>
<th>$g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
  a &= \overline{s_0} \\
  c &= \overline{s_1} \\
  e &= \overline{s_0} \\
  g &= s_1 \overline{s_0} \\
  b &= 1 \\
  d &= \overline{s_0} \\
  f &= \overline{s_1} \overline{s_0}
\end{align*}
\]
The display module

\[ a = \overline{s_0} \]

\[ b = 1 \]

\[ c = \overline{s_1} \]

\[ d = s_0 \]

\[ e = \overline{s_0} \]

\[ f = \overline{s_1 s_0} \]

\[ g = s_1 \overline{s_0} \]

\[ // A module for driving a 7-segment display \]

module display (s1, s0, a, b, c, d, e, f, g);

input s1, s0;

output a, b, c, d, e, f, g;

assign a = \sim s0;

assign b = 1;

assign c = \sim s1;

assign d = \sim s0;

assign e = \sim s0;

assign f = \sim s1 \& \sim s0;

assign g = s1 \& \sim s0;

endmodule

[ Figure 2.46 from the textbook ]
// An adder module
module adder (a, b, s1, s0)
    input a, b;
    output s1, s0;

    assign s1 = a & b;
    assign s0 = a ^ b;

endmodule

// A module for driving a 7-segment display
module display (s1, s0, a, b, c, d, e, f, g);
    input s1, s0;
    output a, b, c, d, e, f, g;

    assign a = ~s0;
    assign b = 1;
    assign c = ~s1;
    assign d = ~s0;
    assign e = ~s0;
    assign f = ~s1 & ~s0;
    assign g = s1 & ~s0;

endmodule

module adder_display (x, y, a, b, c, d, e, f, g);
    input x, y;
    output a, b, c, d, e, f, g;
    wire w1, w0;

    adder U1 (x, y, w1, w0);
    display U2 (w1, w0, a, b, c, d, e, f, g);

endmodule
Questions?
THE END