CSE140 L

Instructor: Thomas Y. P. Lee
January 18, 2006

CSE140L Course Info

- Lectures
  - Wednesday 10:00-11:20AM, HSS1330

- Lab 1 Assignment Begins
  - TA’s JinHua Liu (jhliu@cs.ucsd.edu)
  - Contact TAs if you’re still looking for a lab partner.
  - Lab group list is maintained on the webboard.
  - TA office hour will be posted on the class webpage.

- Office Hours
  - E-mail: yunpaam_lee@yahoo.com (may change later)
  - Office Hours: appointment only

- Lab Hours Start from next week
  - When: check website for detail schedule
  - Where: B230, B240, B250, B260, B270
  - You may work on your own PC. VHDL or Verilog both accepted.

- Course Policy
  - Honesty is strictly enforced. You may only discuss project with your lab partner.
Number System

- Three common schemes: sign-magnitude, one’s complement, two’s complement
- Sign-magnitude: MSB=0 for positive, 1 for negative
  - Range: $-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$
  - Two representations for zero: 0000.. & 1000..
  - Simple multiplication but complicated addition/subtraction
- One’s complement: If N is positive, then the negative is $\overline{N}$
  - Example: 0111 = 7, 1000 = -7
  - Range: $-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$
  - Two representation for zero: 0000.. & 1111..
  - Subtraction implemented as addition followed by one’s complement
Two’s Complement Representation

- Two’s complement = bitwise complement + 1
  - \(0111 \rightarrow 1000 + 1 = 1001 = -7\)
  - \(1001 \rightarrow 0110 + 1 = 0111 = 7\)
- Asymmetric range: \(-2^{N-1}\) to \(+2^{N-1}-1\)
- Only one representation for zero
- Simple addition and subtraction
- Many advantages, most common representation

\[
\begin{array}{cccccccc}
4 & 0100 & -4 & 1100 & 4 & 0100 & -4 & 1100 \\
+3 & 0011 & +(-3) & 1101 & -3 & 1101 & +3 & 0011 \\
7 & 0111 & -7 & 11001 & 1 & 10001 & -1 & 1111 \\
\end{array}
\]

[Katz93, chapter 5]

What is Overflow Condition in Addition/Subtraction?

Add two positive numbers to get a negative number or two negative numbers to get a positive number.

If carry in to sign equals carry out then can ignore carry out, otherwise have overflow.
1 Bit- Half Adder

- 1-Bit Half Adder - Addend, augend as inputs, S and Co as outputs
- Truth Table is showing below

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>


Full Adder

- Full Adder – Addend, augend, Ci as inputs, S and Co as outputs
- Truth Table is showing below

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Sum of Product Canonical Form

\[ S = A \oplus B \oplus C_i + A B C_i + A B C_i + A B C_i \]

\[ C_o = A B + C_i (A+B) \]
Block Diagram of Full Adder

- 1-Bit Full Adder –Addend, augend, Ci as inputs, S and Co as outputs
  - Truth Table is showing below


Verilog Example of Full Adder

- Module define the input, output signals
- Assign statement contains logic operation of a full adder

module full_adder (a, b, cin, sum, cout);
  input a, b, cin;
  output sum, cout;

  assign sum = a ^ b ^ cin;
  assign cout = (a & b) | (cin & (a | b));
endmodule

Note: Operators (&,|,…) will be explained in the next session
Also see Verilog References and VHDL references
**Ripple Carry Adder**

- Worst case propagation delay is related to number of bits
  \[ t_{\text{adder}} = (N-1) \times t_{\text{carry}} + t_{\text{sum}} \]

---

**2’s Complement Subtractor**

Subtracting a number is the same as adding the bitwise complement of number then adding 1, combination of addition & subtraction is shown below:

Overflow occurs if carry in to sign bit differs from final carry out
Comparator

- 2-Bit Comparator
- Compares two 2-bit unsigned numbers, output the relationship
  - LE = 1 if A < B, LE = 0 otherwise
  - EQ = 1 if A = B, EQ = 0 otherwise

2-Bit Comparator

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>LE</th>
<th>EQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>10</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>01</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Comparator - Example of Combinational Circuits

- Output is a function of current input
- Example - Thermostat

Temp Sensor

A Comparator

Preset Value

A >= B

Heater Turn On
Multiplexer

- Multiplexing is converting several inputs to single output according to control (select) lines
- 2-to-1 Multiplexer
  - Select one of the input signals according to the control signals, truth table is shown below

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Sel</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

4 Inputs Multiplexer Logic Diagram

- 4-to-1 Multiplexer: 2 Select control lines

An Example of Verilog of 2-to-1 Multiplexer

- Verilog designs consist of interconnected modules
- A module can be an element or lower level design block
- `?:` means conditional operator

```verilog
module mux_2_to_1(a, b, out, outbar, sel);
    // This is 2:1 multiplexer
    input a, b, sel;
    output out, outbar;
    assign out = sel ? a : b;
    assign outbar = ~out;
endmodule
```

Declare and name a module; list its ports. Don’t forget that semicolon.
Comment starts with `//`
Verilog skips from `//` to end of the line
Specify each port as input, output, or inout
Express the module’s behavior. Each statement executes in parallel; order does not matter.
Conclude the module code.
2-to-4 Demultiplexer (Decoder)

- n-Bit binary number decode to $2^n$ output lines, 3-to-8 decoder, 4-to-16 decoder, useful in memory decoding

http://www.play-hookey.com/digital/decoder_demux_four.html

HDL and Design Process

- Hardware Description Language (HDL) is a convenient, device-independent representation of digital logic
- The most common HDL – Verilog and VHDL
- HDL description is compiled into NETLIST
- Synthesis optimizes the logic
- Mapping the target NETLIST to a specific hardware platform

```
Verilog
input a, b;
output sum;
assign sum = (1'b0, a) + (1'b0, b);
```

Compilation and Synthesis

```
Netlist
# g1 "and" n1 n2 n5
# g2 "and" n3 n4 n6
# g3 "or" n5 n6 n7
```

Mapping

FPGA
PAL
ASIC (Custom ICs)
Synthesis and FPGA’s Macros

- FPGA Build-in components are called “Macro”
- FPGA is like an electronic breadboard that is wired together by an Automated Synthesis Tool

Place and Route in FPGA

- **Optimal choose the FPGA macros that can efficiently implement various parts of the HDL code**

```
... always @ (posedge clk) begin
    count <= count + 1;
end
...
```

“*This section of code looks like a counter. My FPGA has some of those...*”

- **Place-and Route: consider area and/or speed factor, choose the needed macros by location and route the interconnect**

```
```

“*This design only uses 10% of the FPGA. Let’s use the macros in one corner to minimize the distance between blocks.*”
Floorplaner

- Floorplanner is an interactive graphical tool that allows edit/view location contraints in design.

- Floorplanning is an optional methodology to improve automatic place and route. Particularly useful on structured designs and data path logic.

Time Delay in Digital Circuits

\[ v_{\text{out}}(t) = (1 - e^{-t/\tau}) V \]

\[ t_p = \ln(2) \tau = 0.69 RC \]
Worst Case Time Delay in Adder

![4-bit adder diagram]

worst case delay: B0 to Co

QuartusII Design Flow

QuartusII Tutorial

- Create New Design Entry – Verilog, VHDL or Schmetic Diagram

Create Project of Adder/Subtractor

- Creating and Compiling Projects
- Setting Design Constraints using the assignment editor
- Analyzing clock and I/O timing
- Adding timing constraints to achieve better design performance
- Simulating design using QuartusII Simulator
- Configuring Altera FPGA device