ENGR xD52: MP b000

Due September 29th 11PM EST

This lab assignment will develop one of the basic building blocks for your processor and introduce you to FPGAs.

Complete HW2 before beginning this lab. MP0 depends on building blocks created in HW2.

You may work in groups of 3 or 4. One representative from each team must submit all deliverables electronically to comparch14@gmail.com as a single compressed archive. Format the email's Header as "[MP0] Name1 Name2 Name3"

Setup

Install the Xilinx Webpack ISE 14.7. The installer is on Public, copy it to your local disk before installing. Note that if you instead download it from the internet, you may end up accidentally installing Vivado instead. Don't.

You will need to generate a node locked license (i.e. a license that's only valid on your computer; this happens during the installation).

There are not enough FPGA boards for everyone, each pair of teams will have one board between the two of you. Find another team to share an FPGA with and then check one out with a NINJA. This will be coordinated during tutorial.

Verify FPGA Tool Chain

This portion of the exercise is not collected. It is intended only to verify that your full tool chain is correctly configured, and will save you headache later.

Run Derek's Intro to FPGA tutorial: tinyurl.com/ca-fpga-13

Make sure that each of your switches and LEDs are working.

The easiest manner to do this is at the NINJA tutorial section on 9/18/14 at 7:00PM East Hall 4th Floor.

You may want to try instantiating various gates for additional practice.

4 bit Full Adder - ModelSim

In HW2 you constructed several modules in Structural Verilog and then tested them with ModelSim's simulator. Re-use the Full Adder component to create a 4 bit Full Adder.

This module should have the following definition:

```
module FullAdder4bit(sum,carryout,overflow,a,b);
output[3:0] sum;        // 2's complement sum of a and b
output carryout;        // Carry out of the summation of a and b
output overflow;        // True if the calculation resulted in an overflow
input[3:0] a;           // First operand in 2's complement format
input[3:0] b;           // Second operand in 2's complement format
// Your Code Here
endmodule
```

Your code will be verified by our own test bench, it is critical your module definition and name matches.

Within this module, you will need to instantiate four of your single bit full adders and then appropriately wire them. Reference a single bit of a bus with the bracket operator: the first (least significant) bit of "a" is a[0].

Each of the gates within your design should have a delay of 50 units of time.

Test Bench - ModelSim

Create a test bench that exercises your 4 bit full adder, and verifies proper operation of its three outputs (Sum, Carry Out, Overflow). It is probably in your best interest to write this at the same time you write the module that it tests.

An exhaustive test requires 24+4= 256 test cases. Select a subset of those test cases that provides an appropriate level of coverage. It may be helpful for you to explicitly document what each subset of test cases are testing. For example, select several test cases that test the overflow flag and preface them with *$display("Test Overflow:");*.

When a test case fails, a well-designed tester should make it easy to identify possible locations of the cause.

Each time that your test bench catches an error in your design, document it in your write-up. Include the test case, the cause of the error, and the fix that made the test case pass.

Note that your test bench will need to account for the gate delays in your design. After setting the inputs, be sure to wait sufficiently long for the result to stabilize.

Full Adder on FPGA

Load your tested 4 bit full adder design onto the FPGA board.

Your top-level module should have the following definition:

```
module mp0(input [7:0] sw, output [7:0] led);
// Input A is the first 4 switches - sw[0:3]
// Input B is the second 4 switches - sw[4:7]
// The sum is displayed on the first 4 LEDs - led[3:0]
// The carry out is the 5th LED - led[4]
// The overflow detect is shown on the last LED - led[7]
   FullAdder4bit adder(led[3:0],led[4],led[7],sw[3:0],sw[7:4]);
endmodule
```

Verify correct operation by manually inputting test cases with the switches and examining the result on the LEDs. Choose 16 test cases that provide a reasonable amount of coverage – you've already tested the design in ModelSim, so you do not need to provide the same level of coverage again.

Provide a photo of your FPGA correctly computing one of the 16 test cases you chose.

In your writeup include the full 16 test cases, why you chose them and their results.

The Write Up

Create a semi-formal lab write up. Minimally, it should include the following:

1. Waveforms showing the full adder stabilizing after changing inputs. What is the worst case delay?
2. An explanation of your test case strategy. Why did you choose the tests you did?
3. A list of test case failures and the changes to your design they inspired.
4. A summary of testing performed on the FPGA board.
5. Summary statistics of your synthesized design (Propagation Delay, Resources Used, etc)

Optionally, include additional information, such as the timing performance or design tradeoffs of your modules.

This should be a single pdf document bundled into the submitted archive.

Submission

Submit a single zip file to comparch14@gmail.com. Format the email's Header as "[MP0] Name1 Name2 Name3". It should include the following:

1. Your write-up as a pdf
2. The Verilog code for
    1. 1 bit full adder
    2. 4 bit full adder
    3. 4 bit full adder test bench
    4. The top level module for synthesis onto the FPGA
3. Associated do file

Hints / Notes

Now that we have signals that are more than 1 bit wide, it makes sense to refer to them using buses. This allows us to reference all of the related bits in a convenient manner.

If ModelSim gives really cryptic errors about missing declarations that aren't actually missing, look at the module above to see if you missed an "end".

Rubric

Adder Works	25	
Sum		15
Carry Out & Overflow		5
Gate Delays		5
Test Bench	15	
Verifies SUM		5
Verifies Carryout		5
Verifies Overflow		5
FPGA (Verified in Writeup)	20	
Verify Tool Chain		0
16 test cases are well chosen		10
Pass 16 test cases		5
Summary statistics		5
Writeup	35	
Timing Waveforms explained		5
Test Bench Explanation		20
Test Bench Failures / Mods		10
Code commented (Nice, not excessive)		5