Program #2 - Verilog!

Describe and simulate circuits using Verilog, a Hardware Description Language (HDL)

- Due: ~~Fri Jan 31, 2014~~ **Mon Feb 3, 2014**
- Worth: 10 points

Good luck!

1. Description

Let's design and simulate the ALU on page 167 in our text using Verilog.
Here's the schematic of a 1-bit slice of our ALU:

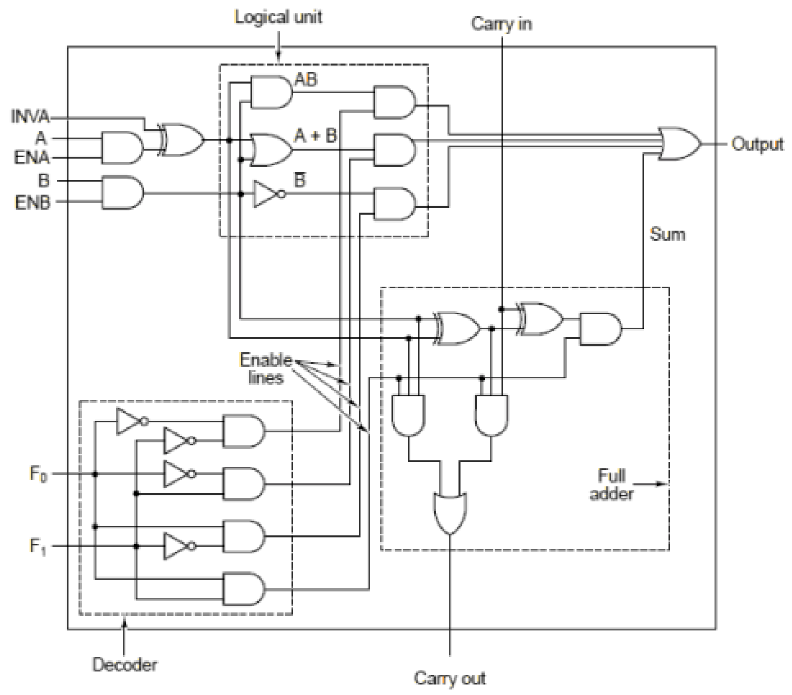
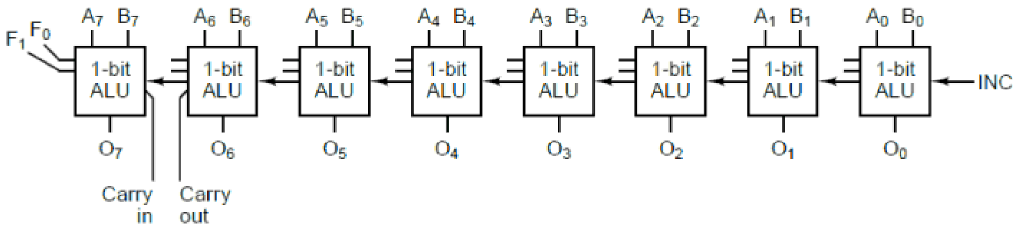


Figure 3-18. A 1-bit ALU.

And here's that slice used (8 times) to create an 8-bit ALU.



In the table below, I have broken down our task into 5 hardware components. Each requires a Verilog source description, a test bench (created by you), and a VCD output file (created by iVerilog).

Therefore, your primary deliverables for Program #2 will be a gaggle of 15 Verilog files:

ID/Description	Input Verilog src/test bench	Output VCD output file
logical_unit The Logical Unit performs AND, OR, and INVERT B operations	`logical_unit.v` `logical_unit_tb.v`	`logical_unit.vcd`
full_adder_we The Full Adder with enable (we) performs the ADD operation	`full_adder_we.v` `full_adder_we_tb.v`	`full_adder_we.vcd`
decoder The decoder uses F0, F1 inputs to select proper operation	`decoder.v` `decoder_tb.v`	`decoder.vcd`
alu_slice This is the 1-bit ALU slice from page 167 in our text	`alu_slice.v` `alu_slice_tb.v`	`alu_slice.vcd`
alu8 Our finished product - the 8 bit ALU bit built from 1-bit slices	`alu8.v` `alu8_tb.v`	`alu8.vcd`

2. Grading

Create a **program2** folder in your k: drive space. I'll look for there for the following:

- Your **README.txt** file where you describe the state of your program
- Verilog files you created: code and test bench files for each component
- Verilog output files, the VCD output file for each component

For your tests, try and be fairly comprehensive. These are small modules, and you should be able to exercise most cases in the logic.
I may ask you to print out some of your waveforms, but I'm having trouble with printing on gtkwave. Stay tuned on this front.

3. Notes

Helpful miscellany for your Program #2 efforts.

Naming Conventions

One of the first things you'll notice is that you have to name a lot of things. Here are the most common and simple Verilog naming conventions that I could find. Please use these.

- Use lower case for most names (modules, wires, variables, files, etc)... ex: alu8, decoder, test.v
- Use UPPER CASE for module parameters… ex: A, OUT1
- Use underscore ('_') to separate parts of a name… ex: CARRY_OUT, alu_slice

Comments

At the top of each Verilog file that you create, please place a block comment that includes: the file name, your name, the date and a short description of what the file does. Here's an example:

```
//
// FILE: alu_slice.v
// AUTHOR: Bill Krieger
// CREATED: Jan 16, 2014
// This is the 1-bit ALU slice from page 167 in our text.
//
module alu_slice( ...);
   ...
endmodule
```

Of course, you can also place comments inside the module to explain complex or difficult parts of your design.

Your process

Some thoughts on getting your program done.

- Do one component of your design at a time.
- Start with Verilog template files I have placed on the k: drive for Program #2. Copy template.v and template_tb.v to your folder and start editing.
- I used Notepad++ to edit my files. It's very nice and even seems to understand Verilog. It has a nice print feature.
- Write your Verilog description, and then its testbench.

The top level - alu8

I think the top-level module, `alu8`, may be the most challenging for people. Some thoughts:

- Don't start alu8 until all the submodules are designed and tested.
- Try using arrays for your data inputs and output. We'll talk about this in class, and the syntax is in our handout. The benefit of this is that you can view your output in hex or decimal and it's easier to see your machine adding two numbers.
- Testing alu8 is challenging. Try all 4 operation codes. Try some adding and subtracting.

A cautionary note - plagiarism

This assignment is different from Program #1. We will all probably similar solutions. I want to be clear about what is acceptable and what is not.
It's GREAT to get help from me: via email or in person. It's OK to talk to your peers as well.
But you know you've crossed the line and cheated when:

- You copy-paste code from someone else
- You don't understand all your code
- You change variables from someone else's existing code

Start early. Ask for help if you need it. And you'll be fine.
Thanks, Bill

More on alu8

First, let me reiterate my advice from above: don't start `alu8` until you have working designs and test benches for all the other modules.
OK. I used arrays for the `alu8` input and output ports, like this:

```
input [7:0] A;
input [7:0] B;
…
output [7:0] OUTPUT;
```

My `alu8` uses 8 `alu_slice` modules that have one-bit wide ports. To reference one bit in an array, it looks just like Java: `A[0]` or `OUTPUT[5]`. If you use arrays in `alu8`, then gtkWave will show your results in hex, which is much easier to read.