CS450/CS650 Winter 2018 – Coding Project 3

Downloads:

1. p3.zip. Contains:
   - Makefile
   - p3_tb.sv – testbenches.
   - *.out – expected output for each benchmark program.
   - *_debug.out – debugging output for each benchmark program.
   - *.gtkw – waveform configuration file for each benchmark program.
2. p_ref.zip. Same as for p2.

There are 5 benchmark programs for p3:

- CheckVowel,
- fact,
- SimpleIf,
- SumArray, and
- Swap.

Their *.dmp and *.x files are found in p_ref.zip. You will extend your mips module from p2 so that it executes the instructions found in these five benchmark programs. (Executing the instruction in the branch-delay slot is optional for these benchmark programs because they always contain `nop` instructions.) This will require adding the following port to the mips module:

```
output logic [1:0] data_access_size
```

The values for data access size are enumerated in file params.sv. Re-use these source files from p2: memory.sv, params.sv, and regfile.sv.

(10 marks) Your submitted mips module should generate output matching the expected output for each benchmark program. There are two marks per benchmark program. Submit only your mips.sv file containing the mips module.

The testbench has three targets for each benchmark program. For the CheckVowel benchmark, for example, the targets are:

- CheckVowel – used to generate the expected output
- CheckVowel_debug – produces instruction-by-instruction output similar to that in p2. These are quite long and included as an aid to help find where errors in execution occur.
- CheckVowel_wave – dumps the signals and opens the gtkwave waveform viewer. Also a debugging aid.

Tips:

- Some benchmarks are simpler than others. Swap and SimpleIf are good starting points.
- If your control flow is not correct the testbench may go on and on. To get some meaningful output, you can execute something like “make Swap_debug | head -40” to get 40 lines of output to see where it is going wrong.

- If all you see from the *_debug outputs in a pc with no register or memory writes, you might have forgotten to copy the *.x file into the directory and the instruction memory is not getting populated with instructions.
- If your control flow is not correct and you want to view a waveform of the first *n* instructions, add this line to the tb_{Swap,SimpleIf,CheckVowel,fact,SumArray} module after it drops reset:
    - `forever #50 if($time>2000) $stop;`

      where the time value is 50 * n (in this example n is 40).