CPU Design Project: Part 5 – Final CPU

Report Due Friday, 4/21/2017

Now, complete the system design by adding Altera memory modules, and write a test program that contains all kinds of instructions in your ISA – including tests of different options for each instruction (ex. branches taken and not taken). Since the Altera memories were not previously used in Part 4, verify the correct execution of your test program, including any "lw" and "sw" instruction.

A 16-bit memory module must be created from Altera's Megafunction Library, as explained in 'Run time content editable memory tutorial' file (posted on course website). This will ensure that RAM blocks in the Altera FPGA are used for system memory, rather than an array of discrete flip flops. **For students who use separate instruction and data memories:** If you have written an instruction memory model for use in Modelsim/Active-HDL, you should also generate a data memory in this part, in addition to the program memory, with one for storing the test program and the other for storing data.

As explained on page 5 of the tutorial, you will need the supplied RAM_init.mif file (a "memory initialization file"). **What's written in the ".mif" file will automatically be loaded into the designated addresses of the generated Altera memory at the start of simulation**. **So any 16-bit word can be placed in any cell of the memory by modifying the RAM_init.mif file.** For example, you can modify the given RAM_init.mif file according to the binary code of your test program. You can also put some data in some cells of the memory. A .vhd file and the RAM_init.mif file will be created in your working directory.

Include the memory.vhd file created above in your datapath. Create the final **CPU component** by instantiating and connecting the memory to the top-level component created in part 4, or else include the memory within the top-level component. At the top-most level, CPU I/O ports should be limited to a ***clock***, ***reset***, and ***inr*** as the input ports, and ***outvalue*** as the output port. Submit the VHDL models of the memory and the final CPU.

Manually compile the test program into binary code, with the binary code loaded into memory via the RAM_init.mif file described above. Submit and annotate the simulation results. To minimize the length of the simulation listing for the **simulation of the test program**, you can limit the display to **one line per clock transition** (i.e., trigger only on clock signal transitions). Show a sufficient set of control signals to demonstrate correct operation of each instruction (control unit state, address bus, data bus, ALU output, register file outputs, register file input, memory control signals, etc.) On the simulation listing, **annotate** by writing the corresponding assembly language instruction next to each execute cycle and highlighting the "significant" result register or bus value. Also submit the assembly language listing for your test program.