Duke ECE496 – Spring 2013 – Project Part 1: Register File

75 Points. Due electronically by 11:59pm on Friday, January 18.

The project for this course is an implementation of the Duke496-S13-32 ISA, a MIPS-like instruction set architecture that has been simplified to make it feasible for a class project. You will work in groups of two to incrementally build a processor that implements this architecture. The complete specification of the entire ISA is TBD.

VERY IMPORTANT DESIGN RULES: For all portions of this semester-long project (unless otherwise specified), you may **NOT** use Behavioral VHDL or any of the "mega- functions" provided by Quartus II. You must implement all components from the most basic structural building blocks, using either block diagrams or Structural VHDL. You may use For/Generate loops, but you may not use behavioral constructs like Process blocks, Case statements, If/When/Else statements, and loops that are not For/Generate. If you have any questions about these design rules, please post a question on the course's Piazza page. The excuse "But I thought that was allowed" will not be accepted.

Project Part 1: Register File

The register file is the collection of registers that the processor uses in its computations. The Duke496-S13-32 ISA has thirty-two 32-bit general-purpose registers, $r0-$r31, where $r0 is *always* equal to zero. At any given time, we can be reading any two registers identified by ctrl_readRegA[4:0] and ctrl_readRegB[4:0]. Reading is asynchronous (not clocked). On any rising clock edge, we can write any one register identified by ctrl_writeReg[4:0] if ctrl_writeEnable is high. Writing is synchronous (clocked) on the positive (rising) clock edge. The signal reset asynchronously resets all registers to zero when it is high, which you will do when you boot up your processor.

You will implement your register file in Structural VHDL using the Quartus II software. We will use Quartus II Web Edition 9.1 Service Pack 2, which is available for download at ftp://ftp.altera.com/outgoing/release/91sp2_quartus_free.exe. Later versions do not include a simulator.

Best practices: (1) When making registers with write-enable, never AND together clock and enable, because doing so confuses Quartus' timing analyzer; instead, use the DFFE primitive. (2) Always use Timing simulation in the simulator, found at Assignments -> Settings -> Simulation mode. (3) Use tristate buffers connected to the same bus (instead of a 32-to-1 mux) to generate the output of the register file.

You should create one VHDL (regfile.vhd) or Block-Diagram (regfile.bdf) file that has exactly the same format as the diagram in Figure 1. This top-level regfile.vhd or regfile.bdf file is likely to refer to other lower-level files (e.g., oneBitRegister, etc.). The top-level file regfile.vhd or regfile.bdf file is what you will then use later in the semester when you need a register file for your processor. Figure 1 is a screenshot of the register file component in Quartus, and it shows the signal names that you MUST use in your design to facilitate testing and grading. In general, all control signals in the project parts

will begin with “ctrl_” and all data signals will begin with “data_” in order to improve clarity through consistency, and you are encouraged (but not required) to follow this convention within your components as well.

After implementing your register file, you should test it thoroughly to verify that it works correctly. One example test waveform is provided for your register file at http://people.ee.duke.edu/~sorin/ece152/project/test_regfile.vwf. In addition, this assignment will be graded by running additional tests that are not provided, so do not assume that you can ignore bugs that do not manifest themselves on the one test that is provided.

Submitting This Assignment

To submit this assignment, create a Quartus Archive (Project -> Archive Project) named project1.qar of all the files needed to implement your design. Make sure that your top-level file is named regfile.vhd or regfile.bdf. Names of lower-level files are unrestricted, but be sure to include them along with your top-level design entity in the Quartus Archive file. Post this archive to the assignment page in Sakai. Only one submission per group should be sent.

Getting Help For Assignments

To get help with this and all other assignments in this course, please check for an answer to your question on the course’s Piazza page. If you have a new question then please post it on Piazza to receive an answer from your fellow students, the teaching assistants, and/or the professor. Private questions may be emailed directly to the teaching assistants and/or professor.

Figure 1: regfile VHDL and BDF screenshots

```
ENTITY regfile IS
    PORT (  clock, ctrl_writeEnable, ctrl_reset : IN STD_LOGIC;
            ctrl_writeReg, ctrl_readRegA, ctrl_readRegB : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
            data_writeReg   : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
            data_readRegA, data_readRegB    : OUT STD_LOGIC_VECTOR(31 DOWNTO 0) );
END regfile;
```