ENGR 9865 Advanced Digital Systems

Mini-project / Labs Requirements Issued date: January 8, 2016

Summary: This project involves design, VHDL implementation, simulation, and testing of a 32-bit multiplier that handles signed operands; in addition to the HDL simulation of both the data path and the control unit, one of the components should be realized on a PLD. Students will work in teams of two.

Description of the device: Multiplication of signed binary numbers can be efficiently carried out using Booth's recording algorithm. The multiplier will be provided with a pair of 32-bit signed binary numbers and a 'start' signal. The output of the device should consist of a 64-bit signed binary number and a 'done' signal.

Design: The paper design should be completed with block diagrams, schematics, ASM charts, state diagrams, excitation tables, and other design details. Other necessary analysis and justification information must be provided.

PLD realization of the controller: The control unit of the above design should be implemented using a 22V10 PLD device. One flip-flop per state design must be employed. Simulation and test results must be shown to the TA before burning the PLD; proper initialization must be employed. The device should be tested using a logic analyzer.

Structural design and testing: This is the bottom-up gate-level design of the multiplier device. Use *bit* data type for all signals unless otherwise specified.

High-level design and testing: This step includes top-down design of the multiplier device.

Fulfilling the requirements described below would fetch you 80% of the total marks. Out of this, 40% of the total marks will be for correctness, 10% for the final report, and 30% for testing – thoroughness of testbench routines, mnemonic program style, annotated waveform and list files, etc. Efforts in terms of introduction of additional useful features (for example, an 'incorrect result' signal, where applicable), bus data type for wire interconnections, file input/output for testbench files, test data generation using high level languages such as C++/C and Java and final multiplication result verification using high level languages, testability, modularity (employ subprograms, overloading, etc.), clarity and expandability, and readability will be worth up to an additional 20%.

Thursday January 21, 2016 (Week of January 18-22)

1. Implement simple gates (NOT, AND, OR, XOR, NAND, NOR) using dataflow architecture and bit data type. Use a propagation delay of 1 nanosecond for the NOT gate, 2 nanoseconds for the 2-input gates and 3 nanoseconds for the 3-input AND, OR, NAND and NOR gates. Write testbenches to thoroughly test all the devices. Keep the tested devices in a package for later use.
2. Implement a full adder, a 2×4 decoder and a 4-to-1 multiplexor by wiring appropriate gates. Write testbenches.
3. Submit the annotated code, waveform results, and the detailed design at the end of the second lab session.

Thursday January 28, 2016 (Week of January 25-29)

1. Pre-lab requirement: Complete the final design of the multiplier. During the lab, each group will be asked to demonstrate to TA their design methodology and how their design will work. TA will check each group's work and provide feedback. At this time, you must remember that you are going to implement the whole multiplier at gate level, and so be thorough about the details.
2. Implement a positive-edge triggered D flip-flop with an asynchronous clear input using dataflow architecture. Dataflow architecture uses guarded assignments, and no sequential statements
3. Overload AND operator to work in qit data type and test.
4. Using structural level VHDL code, implement a positive-edge triggered D flip-flop with asynchronous clear input using simple gates. While testing this, determine the setup and hold time requirements as well as propagation delays.
5. Submit your annotated code and waveform files.

Thursday February 4, 2016 (Week of February 1- February 5)

1. Write VHDL code with structural level architectures for a binary counter, a shift register, 32-bit adder/subtractor, and other blocks needed for a 32-bit multiplier. Use **rit** data type ('0', '1', and 'Z') for necessary signals. Implement the data path of the multiplier. Interface this device to a data bus.
2. Implement the control unit using dataflow architecture. Use one hot design.
3. Submit source code, list and annotated waveform files. You must include clear diagrams showing your datapath and control unit design.

Thursday February 11, 2016 (Week of February 8-12)

1. Combine the datapath and the control unit and test the final assembly.
2. Prepare file input/output for testbench files.
3. Get ready high level language programs for test data generation, and final results verification.
4. Demonstrate successful functioning of your device to the TA/Instructor.
5. Submit source code and annotated waveform files.

As you can clearly notice, there is not much slack in the schedule. You will be well advised to stay ahead of schedule to avoid last minute complications.