EEL-4713 Assignment #1
Fall 2012
Assigned: 8/30/2012
Due: 9/6/2012 @ 11:55pm Via Sakai
Developed by Scott Arnold and Ann Gordon-Ross

Overview:
The EEL-4713 assignments this semester will have 3 main sections: setup, laboratory, and deliverables. Setup (section 1) provides information on how to setup software tools and environments, when applicable. Laboratory (section 2) covers practical aspects of this class, including the design of a VHDL-based MIPS microprocessor, writing code for your processor, etc. Deliverables (section 3) covers the desired format of the report for the given and lab and a detailed list of deliverables including a point breakdown.

In this assignment, you will perform the following setup tasks:

1.1: Install Quartus II on your laptop
1.2: Go through the Quartus II/VHDL tutorial
1.3: Install Modelsim Student PE-edition
1.4: Go through Modelsim tutorial
2.1: Build and simulate muxes
2.2: Build and simulate registers
2.3: Build and simulate extenders
3: Review deliverables and write report

Section 1: Setup

1.1: Install Quartus II on your laptop (5%)
- Go to www.altera.com, select "download" and "Quartus II Web Edition." Follow the instructions to download and install the software.
- Mention in the appropriate section of your report the version of Quartus you are using.

1.2: Go through the Quartus II/VHDL tutorial (10%)
- Once Quartus II is installed and you are able to execute it, select "Tutorial" from the "Help" menu and follow the tutorial.
- In your report, comment on the usefulness of this tutorial and, if appropriate, make recommendations for how it could be improved.

1.3: Install Modelsim Student PE-edition (5%)
- Go to http://model.com/content/modelsim-pe-student-edition-hdl-simulation, select the "Downloads" tab, and follow the instructions to install the software.
- Mention in the appropriate section of your report the version of Modelsim you are using.

1.4: Go through the Modelsim tutorial (10%)
- Once the Modelsim software is correctly installed, follow either or both of the following tutorials to become acquainted with the software.
- The following link is for the tutorial given in Dr. Lam's Digital Design Course: *http://www.hlam.ece.ufl.edu/EEL4712/Labs/Lab0/ModelsimQuickStart.pdf*
  - Advantages: lots of pretty pictures and simple instructions.
  - Disadvantages: Very basic and does not cover as many features of the tools.
- The following link is for a tutorial provided by Mentor Graphics: *http://www.hlam.ece.ufl.edu/EEL4712/Labs/Lab0/modelsim_tut.pdf*
  - Advantages: Very detailed and comes from the company that makes the tool.
  - Disadvantages: Lengthy.
- In your report, comment on the usefulness of this tutorial and, if appropriate, make recommendations on how it could be improved.

Section 2: Laboratory

For the timing portions in this lab you will be using the **Cyclone IV Gx - EP4CGX150DF31I7** device, which will be large enough to run your simulations on and have plenty of I/O ports.

2.1: Build and Simulate Muxes (20%)

There will be situations in the design process where you will need to select between two 1-bit signals. In order to provide this functionality, you will design a 1-bit multiplexer (name your VHDL entity "mux1") that will select between two 1-bit input signals, "in0" and "in1", depending on the value of a third 1-bit select input signal. This third signal, "Sel", should cause an output signal, "O", to take the value of "in0" when Sel=0 or the value of "in1" when Sel=1. Compile, simulate, and test your design. Turn in the VHDL entity and architecture and a screen capture of a simulation trace showing the multiplexer's output for all possible input combinations.

Question 2.1.1: What is the worst-case propagation time from input to output of your 1-bit multiplexer?

If you look at the ***BASIC INSTRUCTION FORMATS*** section of your green reference sheet you will see that the values rs, rt, and rd each corresponding to 1 of 32 distinct registers and are represented by 5-bit numbers. During the instruction decode process, it is sometimes required to select between these 5-bit values. For this reason, you will extend your 1-bit multiplexer design to create a new component "mux5" that will select between two 5-bit input signals, "in0" and "in1", depending on the value of a third 1-bit select input signal. This third signal, "Sel", should cause an output signal, "O", to take the value of "in0" when Sel=0 or the value of "in1" when Sel=1. Compile, simulate, and test your design. Turn in the VHDL entity and architecture and a screen capture of a simulation trace showing the multiplexer's output for two distinct input words and both sel=0, sel=1.

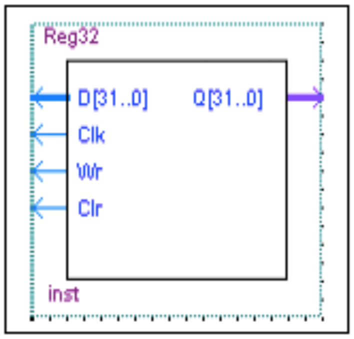
Question 2.1.2: Why are the values rs, rt, and rd represented with 5-bit numbers?
Question 2.1.3: What is the worst-case propagation time from input to output of your 5-bit multiplexer?

The MIPS processor you are designing is a 32-bit processor. For this reason there will be many situations where you will need to select between two 32-bit numbers. Extend your 1-bit multiplexer design to create a new component "mux32" that will select between two 32-bit input signals, "in0" and "in1", depending on the value of a third 1-bit select input signal. This third signal, "Sel", should cause an output signal, "O", to take the value of "in0" when Sel=0 or the value of "in1" when Sel=1. Compile, simulate, and test your design. Turn in the VHDL entity and architecture, and a screen capture of a simulation trace showing the multiplexer's output for two distinct input words and both sel=0, sel=1.

Question 2.1.4: What is the worst-case propagation time from input to output of your 32-bit multiplexer?
Question 2.1.5: Explain the differences, if any, between the worst-case propagation times for your 3 multiplexer designs?

2.2: Build and Simulate Registers (20%)

For a 32-bit processor there will be situations in the design process where you will need to store a 32-bit number without storing it in memory. For this task you will need a 32-bit register, "reg32", that on the rising edge of the clock signal, "Clk", latches a 32-bit input word, "D", if the write enable signal, "wr", is asserted and if "wr" is deasserted, the register's output word "Q" is unchanged. Additionally, include an asynchronous, active-low reset signal "clr", that immediately causes the "D" output to go low. Compile, simulate, and test your design. Turn in the VHDL entity and architecture and a screen capture of a simulation trace showing the following sequence of events: the register is reset; the word "0x1234ABCD" gets written to the register in the first cycle after the reset and remains stored in the register for 10 cycles; the word "0xABCD1234" then gets written to the register and remains stored in the register for an additional 2 cycles. Below is block diagram that might help:



Question 2.2.1: What is the fastest clock rate that your 32-bit register can operate at correctly (remember that clock rate is not the same as worst-case propagation delay)?

2.3: Build and Simulate Extenders (20%)

The immediate field of the Immediate instruction is only 16-bits wide. If you want to use the immediate value of an Immediate instruction as the input to a component that only accepts 32-bit input signals you will need to extend the 16 bits to 32 bits either by zero or sign extension. In zero extension, you concatenate 16 zeros with the input signal to form a 32-bit output signal with zeros as the 16 most significant bits. Sign extension is essentially the same thing except the 16 most significant bits of the output word take on the value of the most significant bit (bit 15) in the input signal.

Design and simulate a 16-bit to 32-bit zero extender called “zeroext”. Your input signal should be called “in0” and your output signal should be called “out0”. Compile, simulate, and test your design. Turn in the VHDL entity, architecture and a screen capture of a simulation trace showing the zero-extender’s output for input words “0x7FFF” and “0xFFFF”.

Design and simulate a 16-bit to 32-bit sign extender called “signext”. Your input signal should be called “in0” and your output signal should be called “out0”. Compile, simulate, and test your design. Turn in the VHDL entity, architecture and a screen capture of a simulation trace showing the sign-extender’s output for input words “0x7FFF” and “0xFFFF”.

Combine your “zeroext” and “signext” components into a single component called “extender” that will perform a zero extension on “in0” when a new 1-bit input signal called “Sel” is equal to ‘0’ and a sign extension when “Sel” is equal to ‘1’. Compile, simulate, and test your design. Turn in the VHDL entity, architecture, and a screen capture of a simulation trace showing the sign-extender’s output for input words “0x7FFF” and “0xFFFF” when “Sel” is both ‘0’ and ‘1’.

Section 3: Report (10%)
All lab assignment reports must adhere to the following format:

1. The report should contain the following sections
   - Cover Page – with your name, date, and the assignment number.
   - Introduction – briefly comment on the assignment and what it accomplishes.
   - Installation and tutorials – briefly comment on the tools used in this lab and the tutorials used to prepare you for the work.
   - Design and Testing:
     - Description of the components and their function. (If relevant, provide boolean descriptions.)
     - Well commented VHDL code (if longer than 2 pages please attach in the appendix)
     - Tests that were performed to verify that the components work correctly. Provide functional simulation waveforms and make sure to describe the test they represent. In other words, ANOTATE!

    - Label answers to questions asked through-out the lab document (e.g., Question 2.2.1 in the lab document should be Answer 2.2.1 in your report)
  - Appendix
2. The report needs to be typed, annotations should not be scans of hand-drawn annotations, and illegible annotations are almost as bad as no annotations
3. Any waveform that is not labeled and annotated will not be considered as valid.
4. Make sure you use the names for inputs, outputs, and entities as outlined in the assignment. This will be essential for connecting these components together in later assignments.

Note: Since all reports will be submitted electronically, annotating your waveforms can be challenging and/or time consuming especially if you're going to use a program like Paint, Photoshop, or OneNote. I've found an open source screenshot and annotation utility called Greenshot http://greenshot.org that can make the process quick and easy.

Rubric:

Section	**Deliverables**	**Percentage of Total grade**
1.1	Include the version of Quartus being used in this lab	5%
1.2	Comment on the usefulness of the tutorial (feedback – no wrong answer)	10%
1.3	Include version of the Modelsim being used in this lab	5%
1.4	Comment on the usefulness of the tutorial you used (feedback – no wrong answer)	10%
2.1	VHDL (5%), Answers to Questions (5%), Annotated Simulations (10%)	20%
2.2	VHDL (5%), Answers to Question (1%), Annotated Simulations (14%)	20%
2.3	VHDL (5%), Annotated Simulations (15%)	20%
Report	Report follows the format given in section 3 (5%). Report is neat and professional (5%)	10%