ECE 443

MIPS-16, Part 5: Designing a 16-bit RISC style microprocessor

Lab #8: The Complete Data Path

Lab assigned: **April 11, 2023**
Report due **April 18, 2023**

Introduction

The goal of this laboratory is to consolidate the microprocessor datapath from the components that have been previously designed in laboratories 4-7. The microprocessor is a 16-bit machine that is based on the 32-bit Reduced Instruction Set Computer (RISC) machine that is presented in our textbook.

Equipment:

1. Intel Quartus Prime software.

Procedure:

1. Carry out attached worksheet and exercises.

Worksheet:

1. Review Figure 4.17 from Patterson and Hennessey and confirm that all the components you need have been designed. You will need to design only two new components – **PC** (**Program Counter**) **register** and **CONTROL** module:
    a. The **PC** component should have three control signals (CLK, EN [enable], and RST [to reset the value of PC to 0]), a 16-bit input and a 16-bit output.
    b. Design a VHDL implementation of **CONTROL**. The code should look at the operation code within each instruction and generate the appropriate signals. One way to do this is to create a process that will be triggered by instruction changes (i.e., the instruction should be in the sensitivity list of the process), and have a big SWITCH/CASE statement inside the process that sets all signals based on instruction fields. Make a table of control signals with names, description, and instructions in which they are active. Follow the style of Table 1.

2. Copy all the VHDL files you designed previously into the folder for this Lab. Include all the VHDL files in the project. Don't forget to select the correct Cyclon chip for the board you are using.

3. The handout for this lab includes three BDF files depicting a hierarchical schematic diagram of one MIPS-16 implementation. [***This is just a suggestion****. You are free to do it any other way you find more practical or better suited to your design.*] Following those templates (or your own organization of components) **and the datapath given in Figure 4.17**, create your own BDF files - one for the top level and several for lower levels. For example, in my organization:

- The top level (control) contains CONTROL and Data Memory
    a. The first level-two component (m16dp1) contains PCREG, Instruction Memory, INCTWO, ADD16, PCJMP, sign extension module, and some multiplexers;
    b. The second one (m16dp2) contains Register File, ALU16, and accompanying multiplexers

This is not necessarily the best organization, and you would want to come up with a better (more logical) one. You might also have to adjust some connections or to change signal names, to make it work with the files you designed.

Before coming to lab, create a sketch drawing of your proposed hierarchical organization. Make sure all the bus sizes are correct. Discuss it with your TA or instructor before you proceed.

4. If you haven't already, change your register file so that register 0 ($0) always reads zero. Read step 6, think about your processor implementation, and if you have to, come back to this step and set one of the remaining register to a constant.

5. Create a symbol for each VHDL file that will be used in schematic diagrams (i.e., in Block Diagram – .BDF files) that employ that component. Make sure you saved and compiled all the changes. Open each file and select *File ⇨ Create/Update ⇨ Create Symbol Files for Current File*. Insert components into your BDF (schematic capture) files and connect them to create schematic diagrams you sketched before.

6. Put some data at a couple of addresses in the data memory (hardcoded in the DATA_MEM). Write an assembly language program that demonstrates the functionality of your solution. For example, include the following actions in your program: and the instruction sequence in the instruction memory (INS_MEM) that executes the following sequence of actions;
    a. Moving data from data memory into registers.
    b. performing arithmetic operations on registers and placing results into another register.
    c. Storing data from one of the registers into data memory (DAT_MEM) at a specific location.
    d. Read the data from R1 so that it shows up on output (notice how I made the output of Data Memory an external bus).
    e. Add a couple more instructions to demonstrate other fascinating abilities of your microprocessor ☺

Given the load/store architecture of the machine, the task may be carried out by instructions in a different order than that listed above. Some of these tasks may require several instructions. Also, there may be some extra data (constants) that should be loaded into registers and/or memory to carry out the task. Make a table of the actions, instructions and machine code. Follow the style of Table 2. Separate the different fields in the instruction machine code using colons (:).

7. Compile the complete design with this program for a timing simulation. This requires several steps. Note the following;
    a. Make sure the symbols are up-to-date when you compile the BDF file. If you changed something, **recreate** (just like you did in Step 4) and **update** (right click ⇨ *Update Symbol or Block*) the symbol
    b. Make sure to select "Timing simulation" (not Functional)
    c. Make sure to select a **Cyclone IV family and your device (chip)**.

8. Conduct a simulation of the design for the selected **Cyclone IV** device. Show the appropriate input and output signals.

9. Record the resources used (i.e. number of logic cells and percent utilized) for the entire design (not for each component individually). Determine the maximum propagation delay and calculate the maximum clock frequency for this device.

10. Can your design fit into a smaller device? Select AUTO and let Quartus find a device for you. Record the specific device and the device resources that are used for the entire design in this case. Repeat your calculations for the maximum clock frequency (if different). Open *Chip Planner* and *Technology Map Viewer* and try to appreciate the complexity of the project you just completed. **Congratulations!**

Requirements:

1. The report should include:
    - Annotated simulation file(s) verifying that the machine works as designed.
    - Completed Tables 1 and 2.
    - Printouts of your BDF files.
    - Answers and calculations.

2. **Upload all relevant *.bdf, *.vhdl and *.vwf files on Blackboard**

Tables:

TABLE 1. BUS CONTROL SIGNALS FOR MIPS-16		
CONTROL SIGNAL	**DESCRIPTION**	**ASSERTED (OR VALUE) IN:**
NAME	*Selects one of …*	*LW*

TABLE 2. PROGRAM		
ACTION	**INSTRUCTION**	**MACHINE CODE**
R7 $\leftarrow$ 0x????	add \$6, \$2, \$2 lw \$7, 0x04(\$6)	xxxx:xxx:xxx:xxxxxx