

Adding Instruction Decoding to the Datapath

1 Objective and Overview

In this lab you will control the datapath using instructions from the lab computer. This lab mainly includes two new components. First, you should write an *instruction decoder* that translates the different bits in an instruction word to control and data signals for the datapath. Next, you will be given a debounce circuit to integrate it into your design that connects the button *BTN1* on the TUL PYNQ board to the clock so that you can single step your design.

You must complete Lab 5 before you start this experiment.

2 Decoding Instructions and Executing them on the Datapath

In this lab, you will add instruction decode logic to the datapath you completed in Lab 5. You should implement an instruction decoder as described below. The machine code for your instructions will be entered into your design using VIO. Instruction formats are described in `instruction_set.pdf` available on Canvas. Please note that the instruction decoder is a combinational logic and a clock edge is only needed for writing to the register file and writing to memory in the datapath. The second change is to add a debounce circuit that allows you to single step your datapath. This circuit will connect the *BTN1* button on the TUL PYNQ board to the clock inputs for memory and the register file. Figure 1 shows the top level design for this lab experiment.

2.1 Prelab

Download the prelab assignment from the course webpage. should include your answers to the prelab assignment.

2.2 Entering Your Design

2.2.1 Instruction Decoder

Design an instruction decoder in Verilog with the interface shown in Figure 2. Name your module `inst_decoder`. Your circuit should be combinational and should take in an instruction and generate the signals shown in the figure. Note that there is a new signal `RegDst` that comes from the decoder. This signal chooses whether the register write address from the instruction word is the `rd` field (for R type instructions) or the `rt` field (for I type instructions). The top level module will instantiate a mux based on this signal.

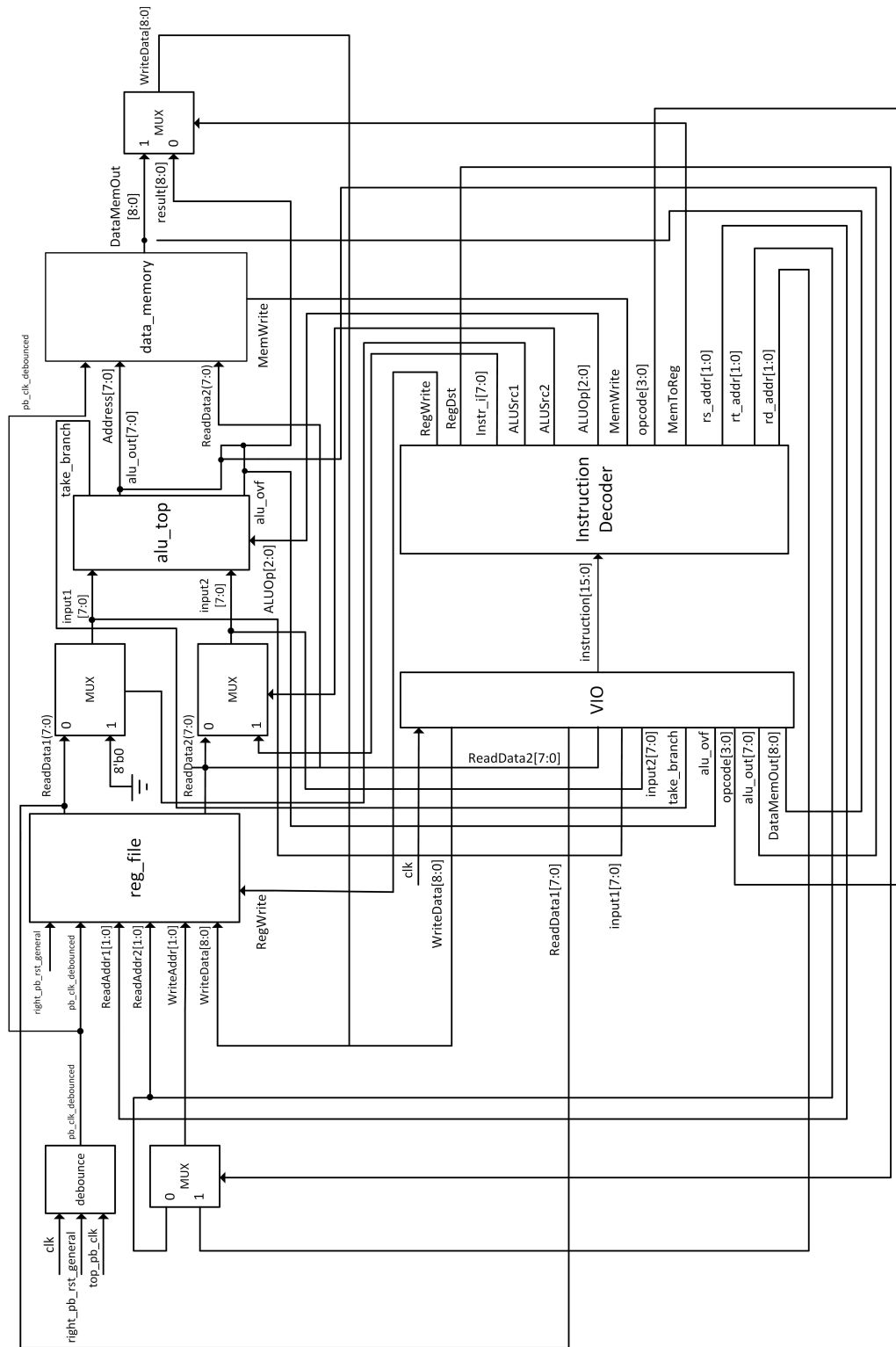


Figure 1: Datapath with Instruction Decoder Unit

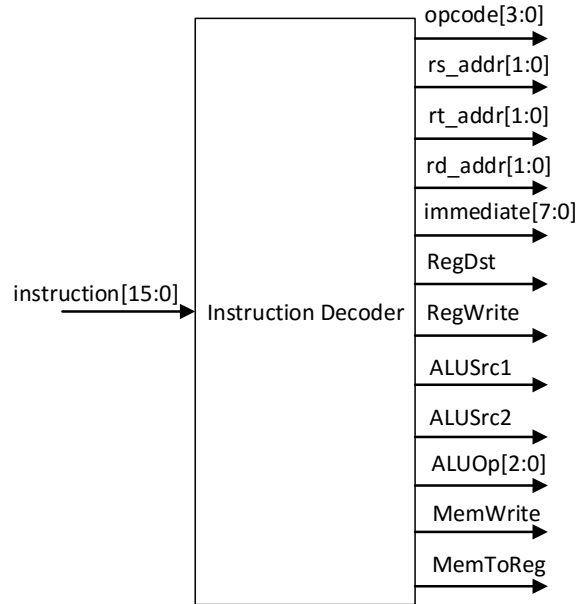


Figure 2: Instruction Decoder Interface

2.2.2 Simulate the instruction decoder

Before implementing the instruction decoder on FPGA, write a testbench to test your instruction decode circuitry. Your *test vectors* should cover all possible types of instruction (by varying opcode, following the same order as shown in Prelab Table 1).

Create a new testbench module called `inst_decoder.tb`.

Enter the test vectors that you have chosen into the testbench, save the testbench, and run it in XSIM. Make sure that you set the Simulation Run Time property to an appropriate value before running your simulation.

Save the waveform as part of the evidence that your lab works. The TA will provide specific instructions regarding what is expect to be handed in.

2.2.3 Designing the top level

Get the top level Verilog file, `pdatapath_top.v` from the course webpage. This Verilog code shows how all of your components are connected. You should have regfile, alu and instruction decode components already designed. Get the debounce circuit, `debounce.v` from the course webpage. You will need to generate data memory and VIO to add to this experiment. Note that this VIO module has several new inputs and outputs.

Add the constraint file for your design. The constraint file for this design is named: `pdatapath.xdc`

2.3 Testing in Hardware

2.3.1 Implementing the Design with Vivado Synthesizer

Click on **Generate Bitstream** under the **Program and Debug** section in the **Flow Navigator** pane to run the synthesis process. Vivado IDE will run the **Synthesis** and **Implementation** processes automatically.

If the synthesis process runs correctly, continue by programming the FPGA board. Follow the same steps you have for previous labs.

2.3.2 Programming the FPGA with Hardware Manager

Program your hardware the way you have in previous labs. Open the hardware manager, open the target device, and program the PYNQ.

Now you should see the `hw_vio_1` window open. Click on the **+** sign and add all of the probes by selecting and adding them. Now you are ready to test your design in hardware.

2.3.3 Testing Your Design

For this design, we are using one reset, named `rst_general` in Verilog, to reset the memory elements (memory and register file). On the board, it is connected to the right push button (BTN0). We have added the single step button the push button (BTN1). You need to press BTN1 whenever you want to store data in either memory or the register file.

Use VIO to test your design. Use the instructions you developed in the prelab. Keep screen shots to show that you have done this.

Follow instructions from your TA for what you need to submit or demo to show that your lab works.