CENG 351: Computer Architecture I

Lab 4: Datapath

Labs should be completed in pairs.

Processor design requires careful attention to detail. As we add functionality for executing more instructions to the datapath, it quickly becomes complicated. However, understanding the details of each component is critical for avoiding errors in design.

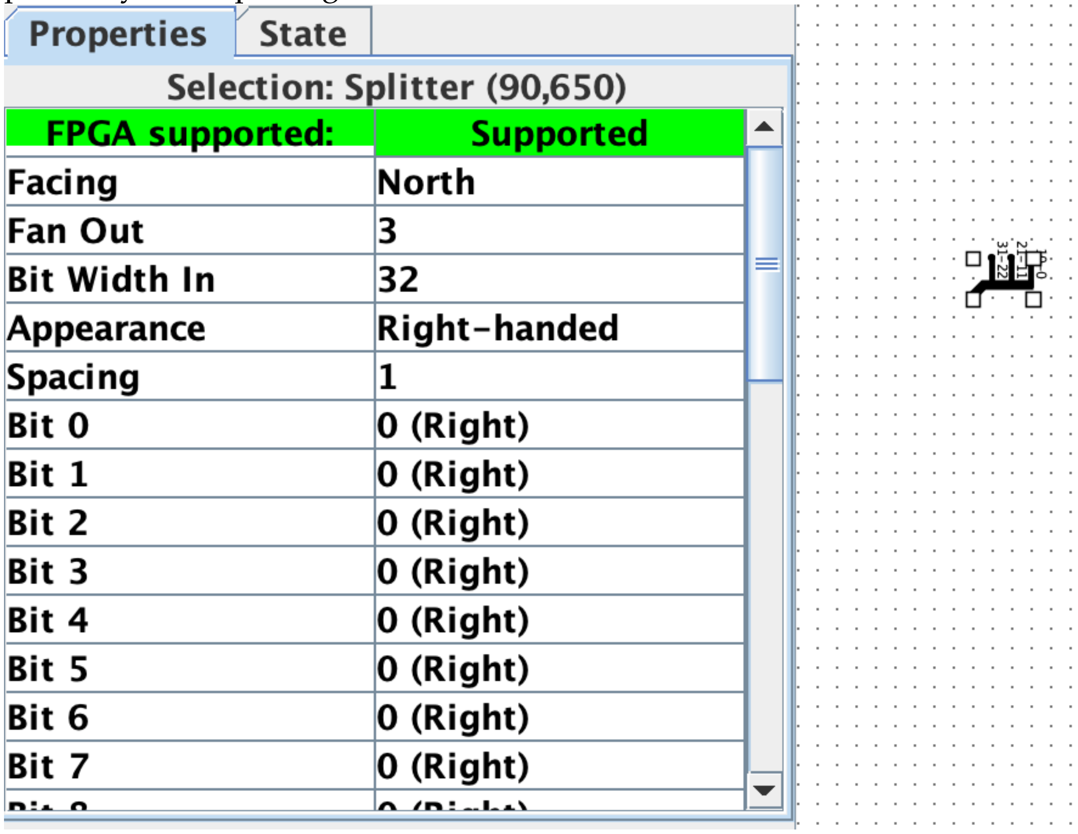
In this lab, you will construct a single-cycle datapath in Logisim using a skeleton provided. The skeleton includes the architectural state as well as a 32-bit ALU. For this lab, we will not implement the control unit (that will be in the next lab). Because there is no control unit, you will need to set the control signals manually for testing. You will test that a basic set of instructions executes correctly on your datapath.

Processor skeleton

1. Please use the provided Logisim file, "architectural_state.circ" to get started. This file contains a PC, instruction memory, register file, 32 bit ALU, and data memory. These components work the same way as the components discussed in class.
2. Familiarize yourself with these components. Note that none of them are connected, but they should have the same inputs and outputs as in the components we used to design the datapath in class. Some of the components are built-in to Logisim, and some are constructed from these built-in components. If you want to see how the component was constructed, you can double-click the name in the design hierarchy on the left-hand side of the window.
3. The instruction memory and data memory have a maximum address size of 24 bits. We don't need large memories, so we will use 16-bit addresses to keep the size manageable. **We will use bits 17-2 to access the memories** so the word addresses are multiples of 4 (as is the case for byte-addressable memory). This will require some splitters in your datapath.
4. Currently, the instruction memory and data memory are empty. We will need to populate the instruction memory with the instructions we wish to execute and the data memory with any data we wish to use, but we will save those steps for testing. Notice that the instruction memory is a ROM and the data memory is a RAM. Why is that the case?

Constructing the datapath

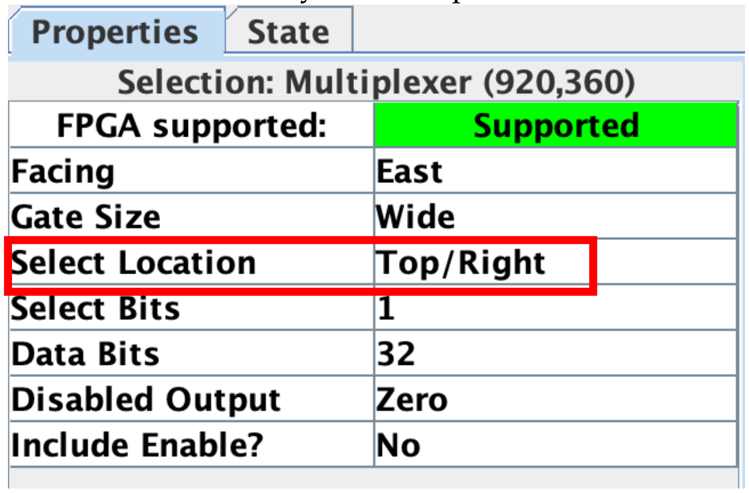
1. You will need to draw in the entire datapath. You can start one instruction at a time, or you can try to start from the complete datapath as well (given below). Note that the clock signal is already included. You will only need to add combinational logic components.
2. You will need to make use of splitters (under the wiring sub-menu) that include only a subset of the bus. When using the splitter, the "bit width in" field should contain the number of bits on the bus you want to split (probably 32). The "fan out" field contains how many pins or busses you want to split the input into. You can have any number of fan out pins as you like, and you can put whatever bits you want on each pin. At the bottom of the properties tab are dropdowns for every input bit. These dropdown menus allow you to select which fan out pin/bus you are putting the bit on.



Selection: Splitter (90,650)

FPGA supported:	Supported
Facing	North
Fan Out	3
Bit Width In	32
Appearance	Right-handed
Spacing	1
Bit 0	0 (Right)
Bit 1	0 (Right)
Bit 2	0 (Right)
Bit 3	0 (Right)
Bit 4	0 (Right)
Bit 5	0 (Right)
Bit 6	0 (Right)
Bit 7	0 (Right)

3. You will need to use several multiplexers. In our diagram, the select bit is usually on the top, but Logisim defaults to having it on the bottom. There is a property you can change to move it to the top. Also, make sure you specify the correct bit width for your multiplexer.

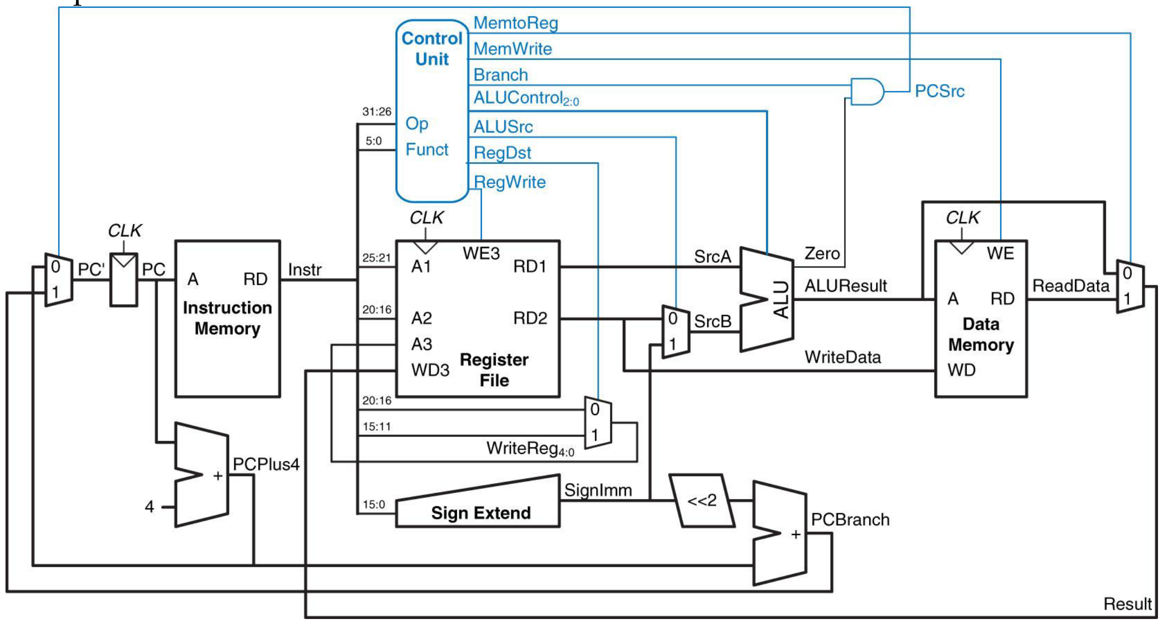


Properties | State

Selection: Multiplexer (920,360)

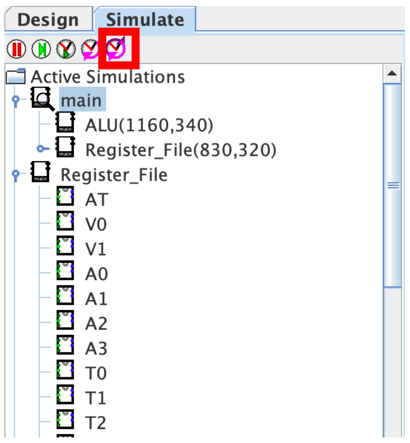
FPGA supported:	Supported
Facing	East
Gate Size	Wide
Select Location	Top/Right
Select Bits	1
Data Bits	32
Disabled Output	Zero
Include Enable?	No

4. The instructions you should implement are lw, sw, r-type, beq, and addi (j optional).
5. The control unit will not be implemented in this lab. Instead, your control signals should be (mostly) one-bit inputs that you can adjust as-needed during your simulation. **Please label your control inputs.** This can be done by double-clicking them, then typing in a name.
6. Datapath:



Simulation

1. You will need to load some sample instructions in the instruction memory in order to simulate your design.
2. An assembly program which will loop forever is given below.
   ```
   top: addi $s0, $0, 36
        sw   $s0, 4($0)
        lw   $s1, 4($0)
        addi $s1, $s1, -1
        sub  $t0, $s0, $s1
        and  $t1, $t0, $s1
        beq  $t0, $t1, top
   ```
   machine language:
   ```
   20100024
   ac100004
   8c110004
   2231ffff
   02114022
   01114824
   1109fff9
   ```
3. You can load this assembly program into instruction memory in two ways. Right-click the instruction memory and select "edit contents" to manually type it in. Or, you can right-click and select "load image…" to load the hex file attached to the lab on Canvas.
4. Once you have loaded the simulation, you will step through the assembly one clock cycle at a time. Before you step forward, make sure to set the control signals to the appropriate values and verify the proper operation of the datapath. You can use the finger-pointer tool to click on wires and see their values.
5. After each instruction is verified, click on the "Tick clock one full cycle" button. You should see the program counter increment and the instruction memory move on to the next instruction. The simulation should be un-paused, as shown in the screenshot below. In other words, as soon as you click "Tick clock one full cycle", all signals should update.



6. Repeat this process for all instructions, taking screenshots to verify the operation along the way. At the end, the assembly program should repeat. You should also be able to see a value stored in the data memory. To observe values stored in the register file, you can use the drop-down menu on the left to look at individual registers. You can also click on the register file, and then click on the magnifying glass to observe the underlying circuit.

Compile a short report that contains screenshots of the entire datapath showing the correct operation for each instruction. You should also show evidence that the instruction completed correctly (correct data in register, showing the contents of busses before writing, etc.) There should be at least 5 screenshots, but you can include more if breaking up the screenshot makes your report easier to read. Describe any problems you had during the lab and what you did to troubleshoot. Describe each partner's contribution to the lab. Include answers to the lab questions below

Lab questions:

1. Include answers to the question posed in step 4 of the processor skeleton section. Why did we use ROM for instruction memory and RAM for data memory?
2. For instructions that write to memory or the register file, when does the processor actually write those values to memory? Why?

Submission

Please submit a digital copy of your lab report. If taking screenshots, please save them into the lab report document instead of as separate files. The lab should be submitted through canvas and only one report needs to be submitted per group. In the lab report, please indicate all lab partners and their contributions.

Due Date

This assignment is due on the date shown on Canvas by 11:59pm. Submit via Canvas in PDF format.

<u>Grading</u>

Grading will be based on correctness and completeness of the solution (i.e. show all your work).