CS447 Lab #8 (Week 9)

Introduction

For this lab, you will use Logisim to implement and use an 8-bit register. Please see Lab #6 for information on how to access Logisim.

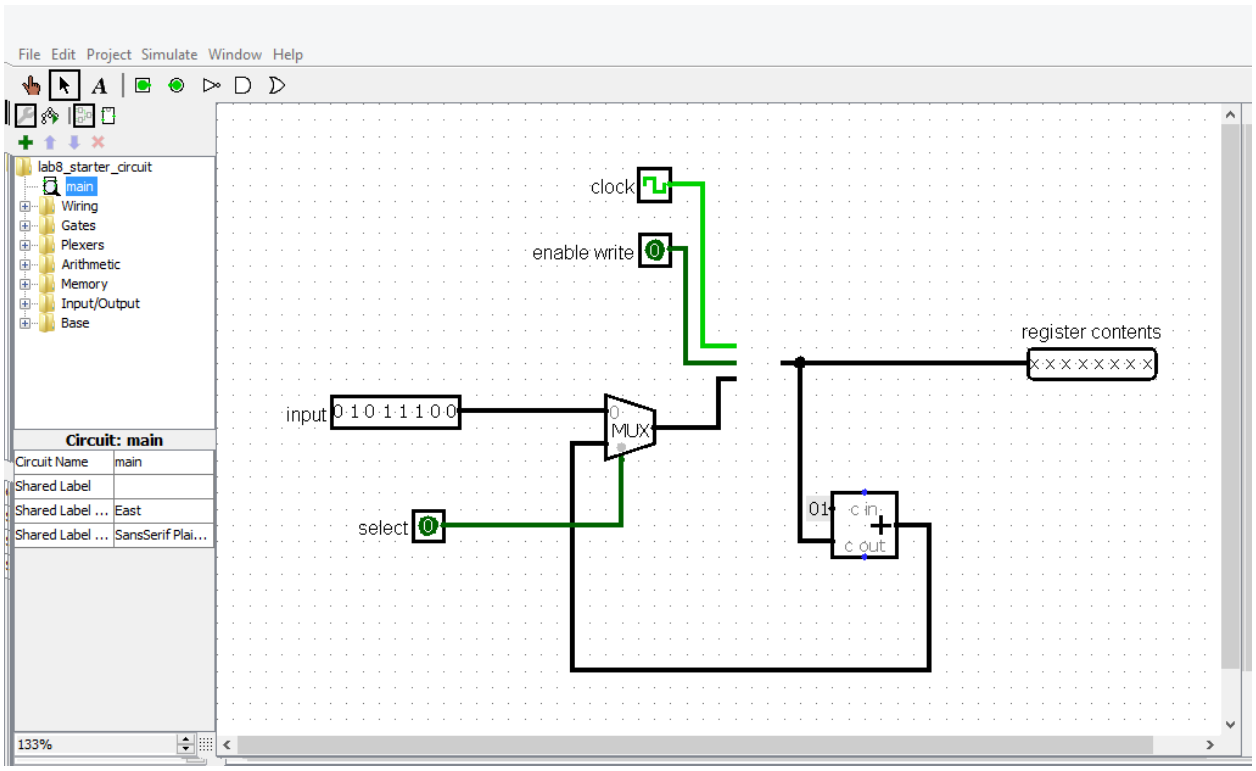
Note: This Lab must be submitted online by 11:59 pm on Thursday, November 3rd. Please see the instructions on the final page.

Objective: Implement and use an 8-bit register built from D Flip-flops.

The most recent lecture discussed sequential logic circuits, including D flip-flops. A single D flip-flop can be used to hold the value of a single bit, and an array of *n* D flip-flops can be used as the storage mechanism for an *n*-bit register. For this lab, you will first build an 8-bit register from 8 D flip-flops, and then incorporate this register into another Logisim circuit (referred to here as the "main circuit"), which uses that register.

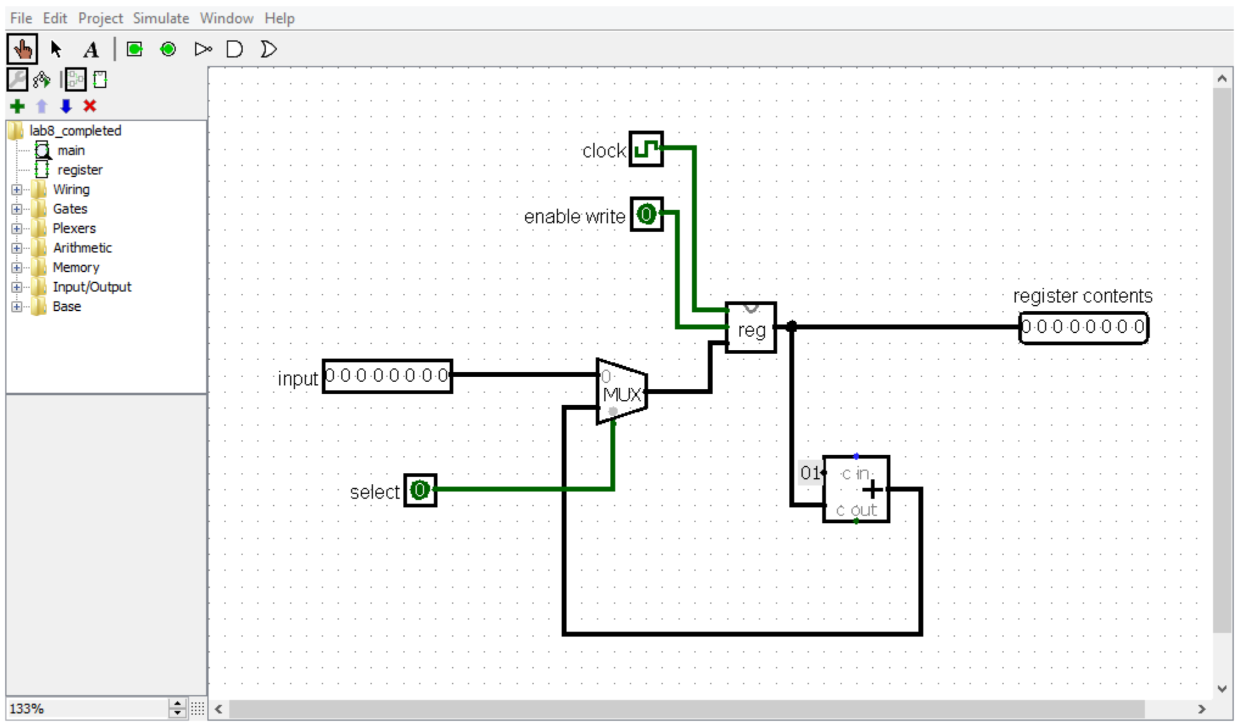
Starter code for the main circuit

The majority of the main circuit is already available in the Github repository: https://github.com/kc13/CS447/blob/master/lab8_starter_circuit.circ. You should first download and open this file in Logisim. The main canvas should look like this:



Circuit: main	
Circuit Name	main
Shared Label	
Shared Label ...	East
Shared Label ...	SansSerif Plai...

The gap in the circuit is where you will place your 8-bit register, which you will create as a separate circuit within the same .circ file. When you are done, the main circuit should look like this:



Note two changes here: (1) The placement of “reg” (the register) in the gap from above, and (2) the addition of a new circuit, “register”, under the icon representing the main circuit.

The functioning of this main circuit will be demonstrated during recitation. Although you will not need to understand it deeply in order to set up your register circuit, you should be aware of how it performs when it is functioning properly. Here is a quick summary of what each component of the circuit does:

1. The **input pins** allow you to enter an 8-bit input.
2. The “register contents” **output pins** will display the present contents of the register.
3. The **clock** provides the signal that will determine the frequency with which the register updates its state.
4. **Enable write** is a 1-bit input pin. When it is set to 1, this gives permission for the register to be updated, at the times dictated by the clock.
5. **Select** is a 1-bit input pin, which is used to govern the input that the multiplexer **Mux** selects. If select is set to 0, then the user-defined 8-bit input will be selected; if it is set to 1, the output of the **8-bit Adder** is selected.

The net result is a circuit that allows you to control the contents of the register in a variety of ways (assuming that the clock is ticking – see discussion farther below):

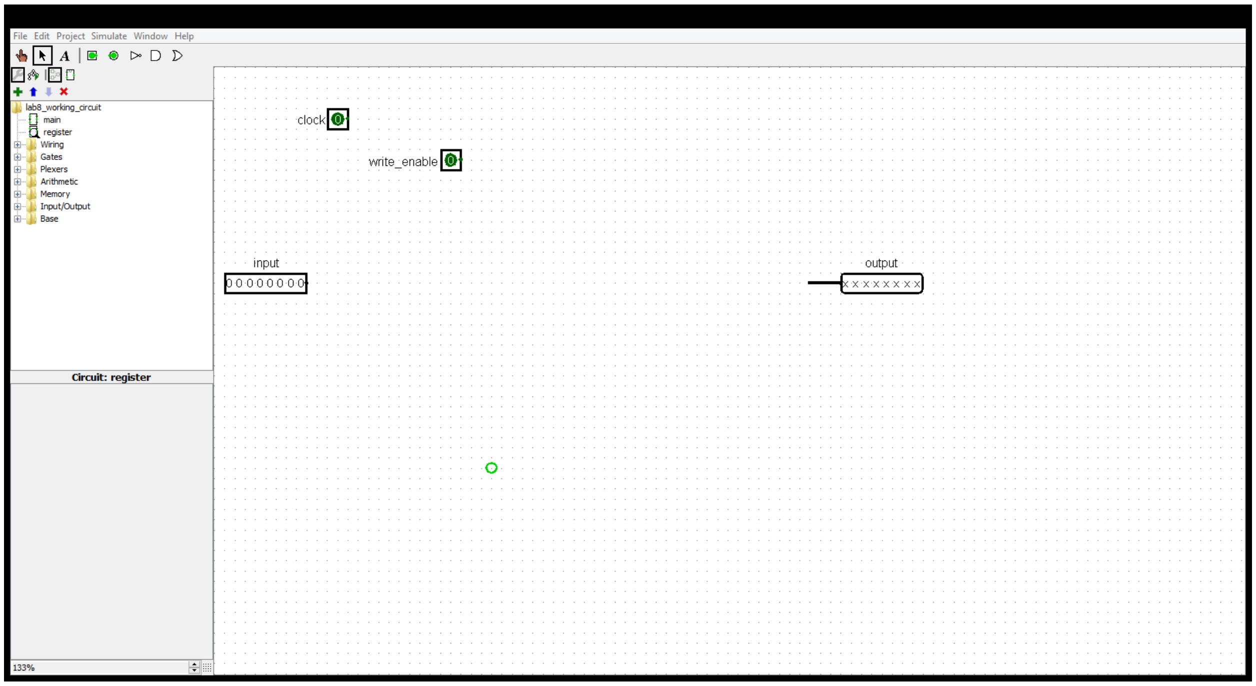
1. You can set the contents of the register to the contents of the 8-bit input, by setting **enable write to 1** and **select to 0**.
2. If you switch **select to 1** (and leave **enable write** at **1**), the register will no longer store any changes in the 8-bit input pins, but instead will accept the input from the **adder**, which acts to continuously increment the current contents of the register by 1.
3. If you switch **enable write to 0**, then no updates to the register will be made, regardless of the selection sent to the multiplexer.

Creating the 8-bit register

The following provides step-by-step instructions for creating the register circuit. As you work through this, you may find it helpful to take advantage of the ability to zoom in on the canvas (see bottom left corner), and to consult the **Library Reference** under the **Help** menu.

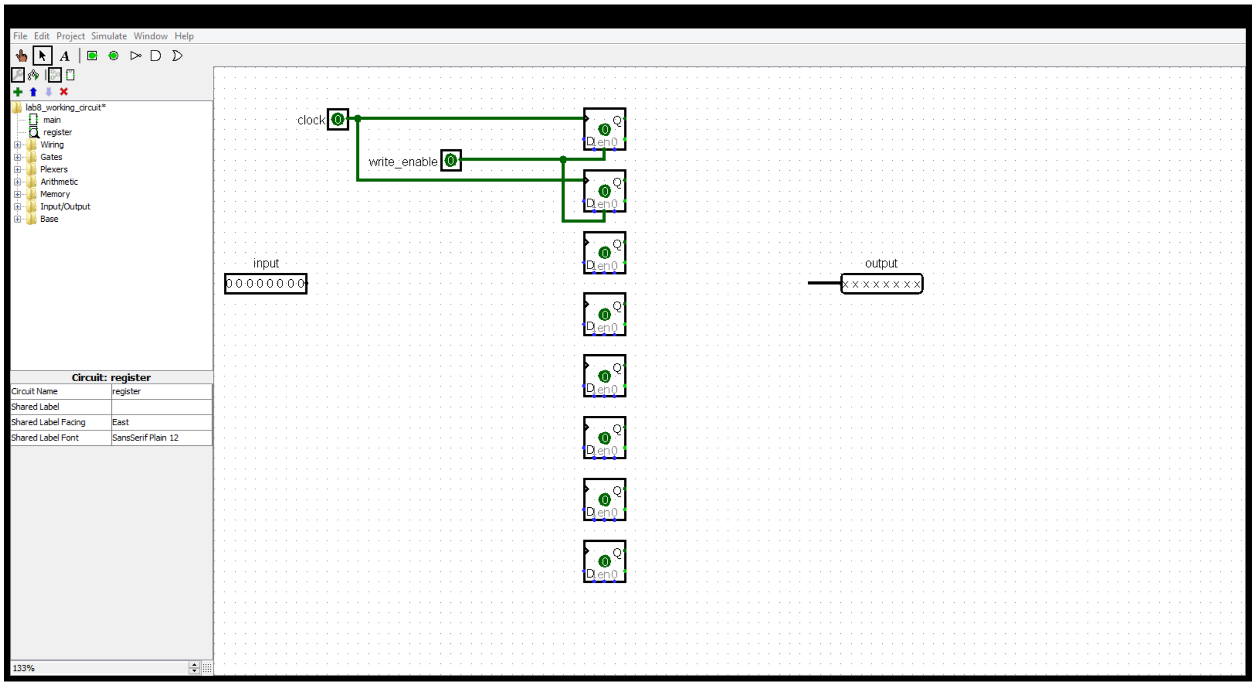
1. Create a new circuit: Under **Project**, choose **Add Circuit**. Give it a name (for example, “register”).
   a. This new circuit should show up under “main” in the explorer pane. **Double-click on the icon** to switch to the canvas for that circuit.
2. Create 8-bit data input and output pins: In the screenshot of the completed program, note that the register component accepts an 8-bit input and returns an 8-bit output. Within the register circuit itself, you will need to create pins corresponding to these inputs and outputs.
   a. In the **register** circuit, create 1 new 8-bit input, and 1 new 8-bit output.
   b. Give these pins **labels**, using the attribute panel. This will make it easier to identify where the input and output wires should go when you return to the **main** canvas.
3. Create an input pin for the clock signal: Your register will also need to accept an input from the clock, with a bit width of 1.
   - Create a **1-bit input pin**, and again, give it some descriptive **label**.
4. Create the enable write input: Create this in the same way you created the clock input.

At this point, you have created all the input and output pins that the **register** circuit will need. Your screen might look something like this:



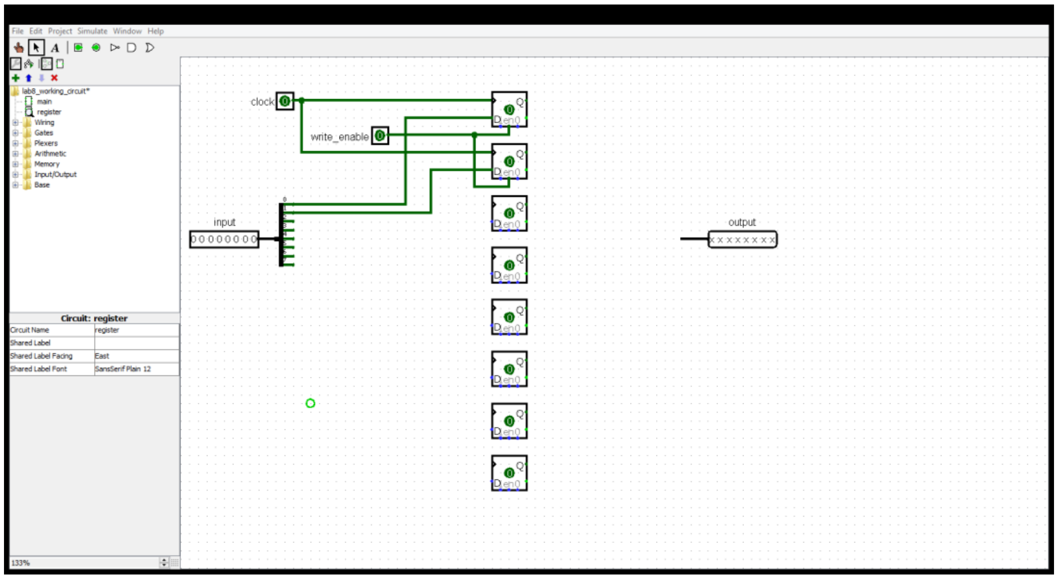
5. <u>Create the array of 8 **D flip-flops**</u>:
    a. Create a **D flip-flop** by choosing this tool from the **Memory** folder in the Explorer Pane.
        - For this lab, we will set the D flip-flops to update on the **Rising Edge** of the clock signal. You can confirm this setting by checking **Trigger** in the Attribute Panel.
    b. Create 7 more D flip-flops, identical to the first one.
6. <u>Connect the **Clock** and **Write Enable** signals to **each** of the 8 **D flip-flops**</u>**.**
    a. Connect each **D flip flop** with the **Clock** input, using wires (in combination with tunnels, if you prefer).
        - The clock signal should be input through the triangle icon on the top left ("northwest") corner of each **D flip-flop.** (If you hover your mouse over this triangle, you should see a message that states that this input site expects clock input).
    b. Connect each **D flip flop** with the **Write Enable** input.
        - This is similar to the process that you used for the clock, except in this case, the appropriate input site on the D flip-flop is in the bottom center (you should see "Enable" when you hover over it).

The following screenshot shows how your screen might appear after completing these steps for two D flip-flops:



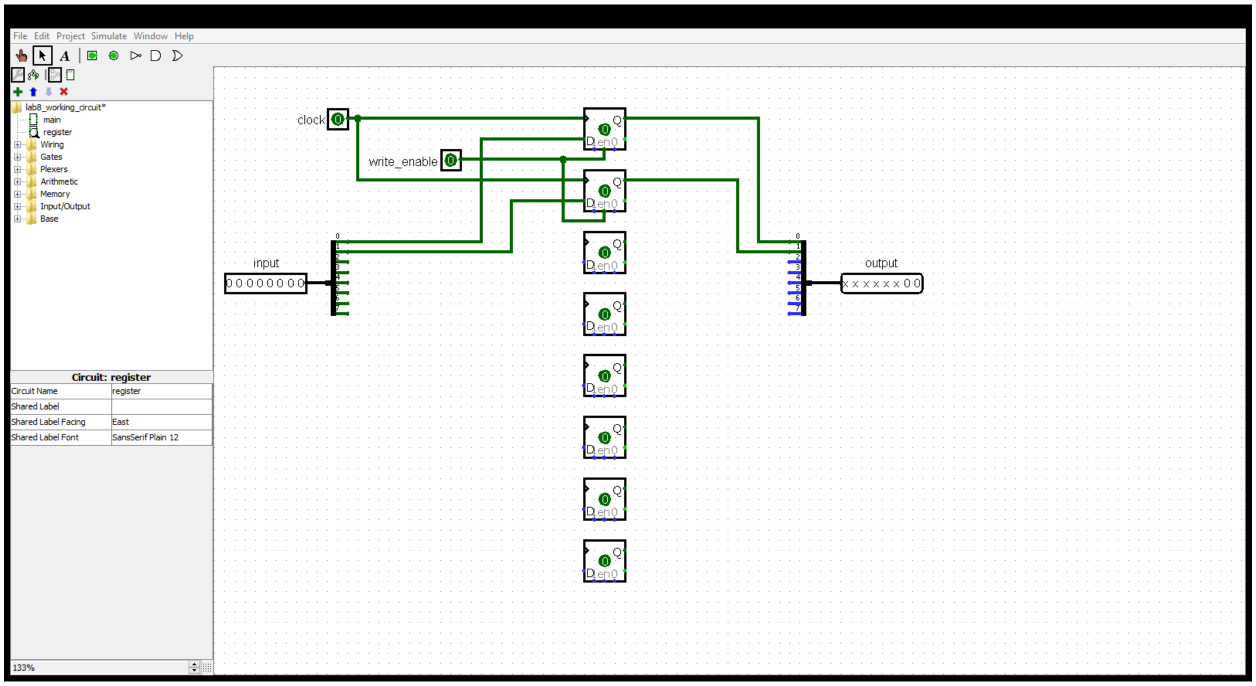
7. <u>Use a splitter and wires to send each input bit to a different D flip-flop</u>:
    a. Refer to Lab #6 for a review of how you can split a multibit input into individual bits with a splitter. The **Bit Width** and **Fan Out** should both be set to 8. The default splitting behavior – which sends each bit of the input out on a different wire – is what we want here.
    b. Once you have set up the splitter, and connected it to the 8-bit input, **send each output bit of the splitter** to a different D flip flop.
        - Each **D flip-flop** will expect to receive an input data bit at the lower point on the left side (under the clock input).

Here is how your screen might look after completing this step for two D flip-flops:



8. <u>Use a splitter and wires to send the state of each D flip-flop to the output</u>: The final goal is to report the state of each D flip-flop (the value of the bit it's holding) and send it back out to the output, so we can see these contents in our main canvas.
    a. Create wires leading out of the **Q** output site of each D flip-flop.
    b. Create a splitter that will accept the 8 different bits from the D flip-flop array, and then send this combined input to the 8-bit **Output** pin.

Here is how your screen might look after completing this step for two D flip-flops:



Circuit: register	
Circuit Name	register
Shared Label	
Shared Label Facing	East
Shared Label Font	SansSerif Plain 12

9. At this point, your Register circuit should be ready to <u>incorporate into the main circuit</u>.
    a. Return to the canvas for the main circuit.
    b. <u>Insert your new Register circuit into the gap in the main circuit</u>:
        i. You can create a new register circuit in Main just as you would create any other new circuit component: Highlight its icon in the Explorer pane, and drag the resulting object onto the canvas.
        ii. Connect the unattached wires to the appropriate input/output points on your register circuit.
            i. For guidance, consult the screenshot of the completed circuit from above, and/or hover over the various input/output sites on your new register object, and look for the labels that you assigned to the circuit's inputs and outputs.

10. <u>Test the circuit</u> from within main.
    a. The **8-bit input pins**, **write enable,** and **select** can be manipulated with the Poke Tool.
    b. **Clock ticks** can be triggered in two ways:
        - Poke tool (one tick per poke)

- Enabling automated ticks by choosing **Ticks Enabled** under the **Simulate** menu. To stop the simulation, simply go back to the menu and unselect Ticks Enabled.

Once you've confirmed that everything's working properly, your program should be ready for submission.

Submission:

If you finish during recitation, please notify the TA, so that your credit for this assignment can be confirmed right away.

Regardless of whether you finish during recitation or not, you must submit your work online. Save the file as **YourPittUserName_lab8.circ** and submit the file via the appropriate link in Courseweb (see Course Documents/Week 9 Lab). The lab must be submitted by **Thursday, November 3rd**, by **11:59 pm**.