CS447 Lab #6 (Week 7)

Introduction

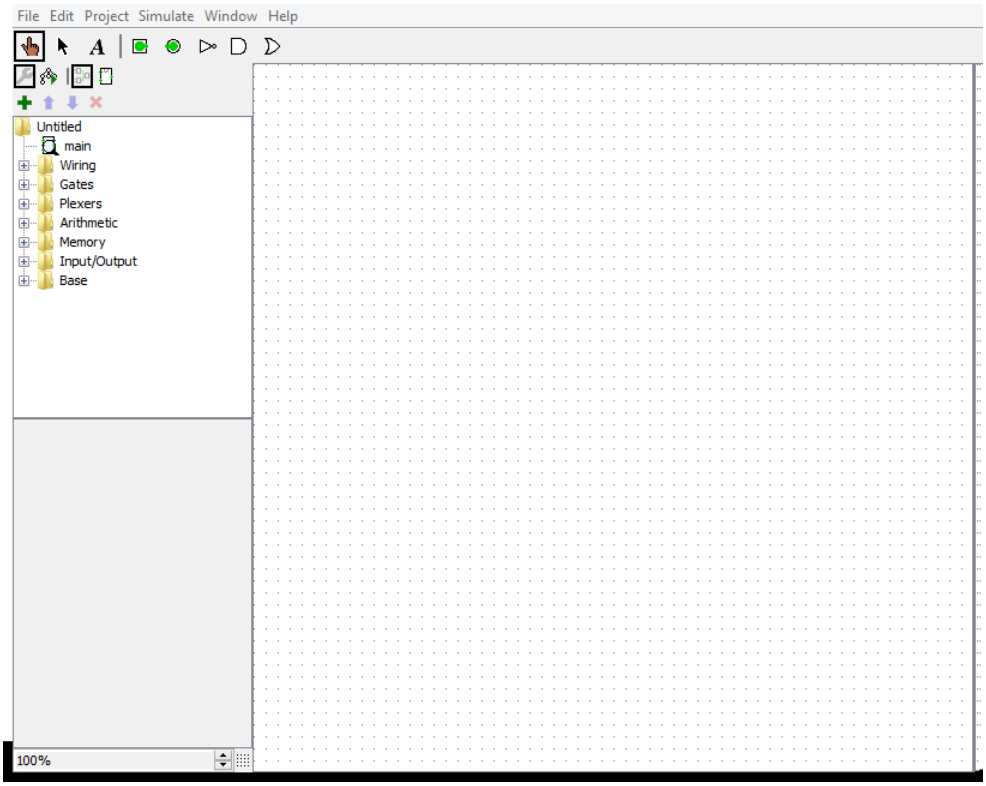
For this lab, you will learn the basics of Logisim, which is the software that we will use to simulate digital logic circuits. Specifically, you will assemble a basic sign-extension circuit.

Note: This Lab must be submitted online by 11:59 pm on Tuesday, October 18th. Please see the instructions on the final page.

Objective: Use the Logisim simulator to create and manipulate a sign-extension circuit.

The main Logisim website is located here: http://www.cburch.com/logisim/index.html and is also linked to the course website. To download Logisim, click on "Download", and then follow the link to Logisim's SourceForge.net page.

When you run Logisim, you should see a screen that looks like this (all screenshots are from Windows 8.1):



The dotted window, referred to as the "Canvas", is where you can create a circuit. The window of folders, referred to as the "Explorer Pane", is where you will find a list of the current set of

circuits available in your program (right now, we are just working with a default circuit called "main"), and a series of folders containing the various components that you might use in a circuit.

Review of sign extension

For this lab you will build a sign extension circuit. Recall that MIPS frequently relies upon sign extension to widen signed binary values of limited bit width (for example, 8 or 16 bits) to the corresponding 32-bit representation. A good example of sign extension can be observed when using the **load byte** (**lb**) instruction in MARS. Consider a scenario where we place the hexadecimal value 0xff in memory:

```
.data
number: .byte 0xff
```

In isolation, the byte "ff" could represent either 255_{10} or -1_{10}, depending on whether we interpret it as a signed or unsigned value. The following instructions load this byte of memory using either the **lb** instruction, which assumes a signed value, or the **lbu** instruction, which assumes an unsigned value:

```
.text
la $s0, number
lb $s1, 0($s0)
lbu $s2, 0($s0)
```

After running these instructions, the value **0xffffffff** (-1_{10}) would appear in **$s1**, and **0x000000ff** (255_{10}) would appear in **$s2**. In other words, **$s2** contains a zero-extended version of the original byte retrieved from memory, and **$s1** contains the sign-extended version.

Re-creation of sign extension in a Logisim circuit

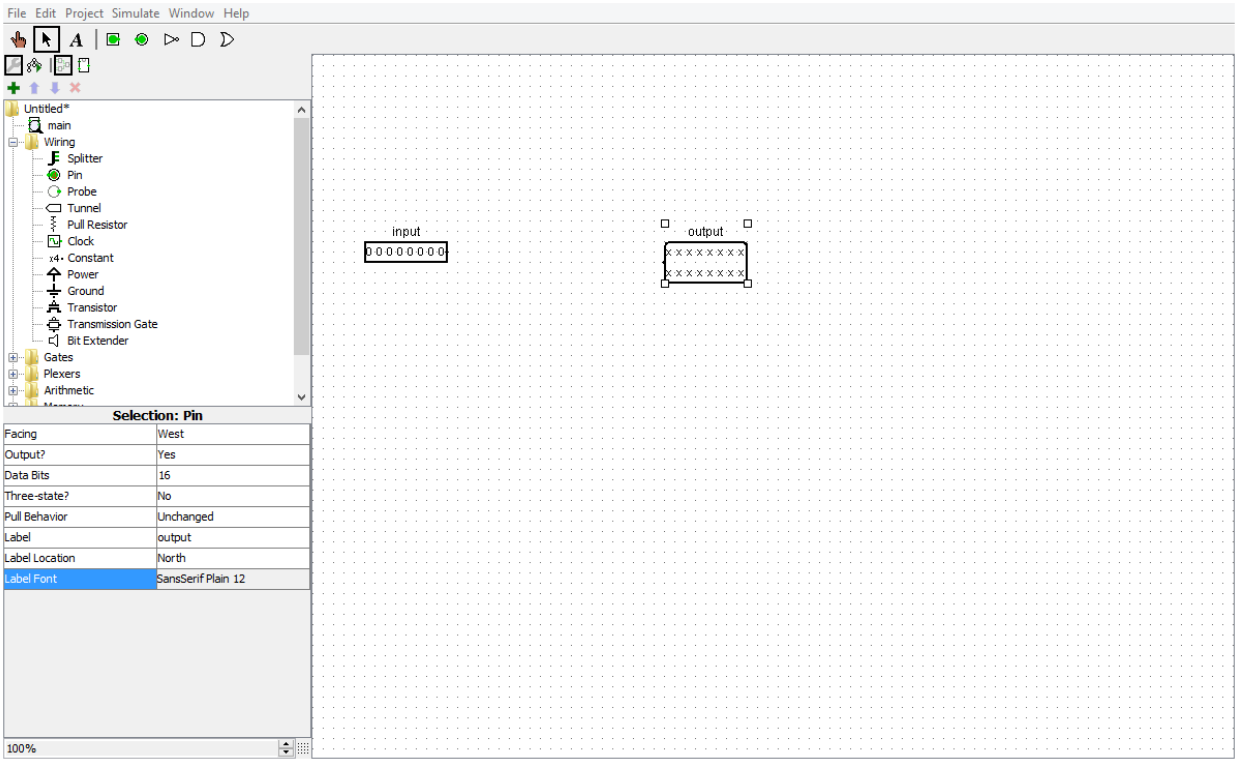
For the very basic sign extension circuit that we'll create in Logisim, we will just place the binary inputs and outputs into simple interactive components that we can view and manipulate (instead of into registers or memory, although Logisim can also simulate these elements). Instead of using the 8-to-32 bit example from above, we'll set up a more modest circuit that expands from 8 to 16 bits. Here's how to set up the input and output:

1. For the 8-bit input: In the Explorer Pane, expand the **Wiring** folder, and click on **Pin**.
    a. Once **Pin** is highlighted, you can release the mouse button, and move the cursor over to the Canvas. You will see a set of crosshairs next to a small rectangle.
    b. The rectangle can be positioned anywhere on the canvas by clicking the mouse button once.
    c. Once the rectangle is placed, you are always free to reposition it or delete it, if you want (using mostly intuitive mouse/key actions).

d. **Pins** may be used to either accept inputs or display outputs. Pins are assigned the Output function by default. To switch to an 8-bit Input:
    i. Select the pins on the Canvas by clicking them with the Select tool (arrow icon in the toolbar).
    ii. In the Attribute Table (under the Explorer Pane), you should see the header "Selection: Pin". Click on the field by **Output**, and switch the setting to **No**.
    iii. Add a label (for example, "Input"), to the blank box by **Label**.
    iv. For **Data Bits**, choose the value 8.
    v. By default, the **Facing** attribute is set to **West**. This means that a wire can be hooked up to the input pins through a small point that is present on the pins' "West" (that is, left) side. Sometimes it is helpful to move this connection point to a different side, and the **Facing** attribute allows you to do this.

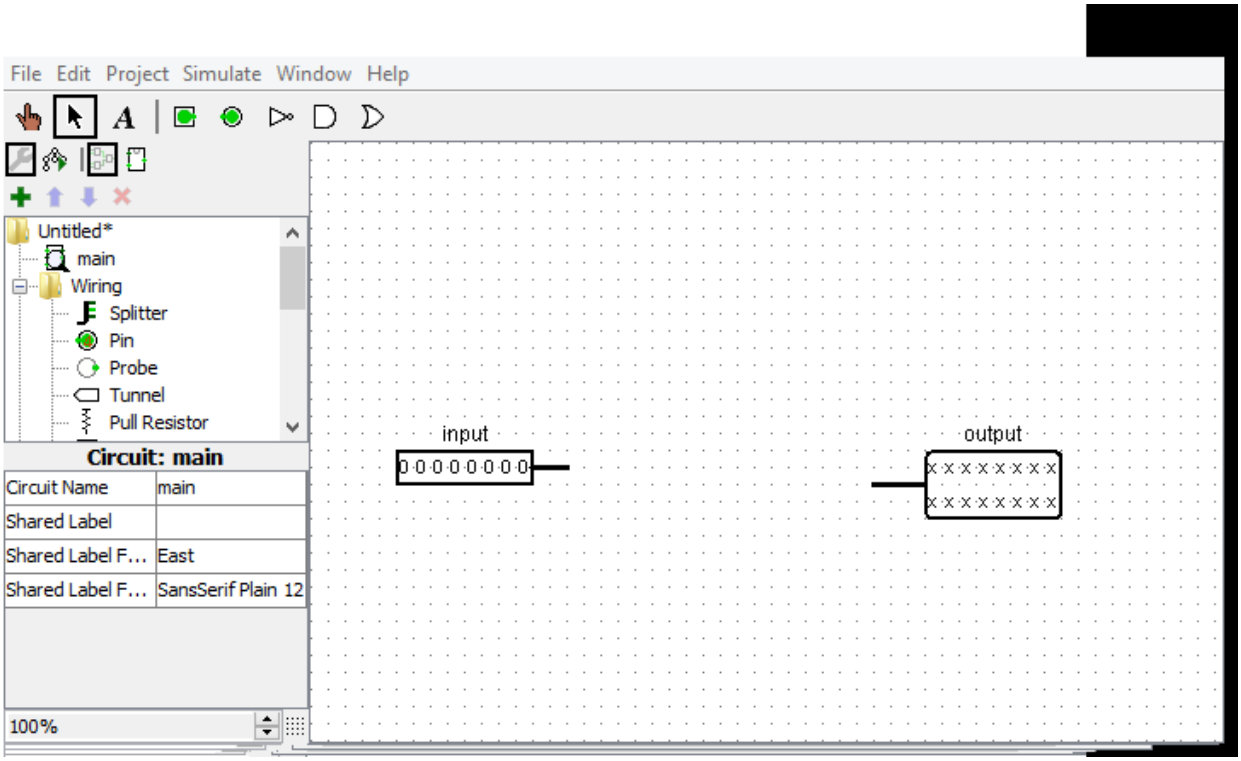
2. <u>For the 16-bit output</u>:
    a. Repeat steps a-c from above. Remember that Output is the default setting.
    b. Give the pins a label (for example, "Output").
    c. For **Data Bits**, choose the value 16.

At this point, the Canvas might look something like this:



Note that you can change the contents of the pins using the **Poke** tool (see the pointing hand icon). Click on the **Poke** tool, and then trying flipping some bits in the **Input** pins. This can be done just by clicking on the individual bits.

Now we need to hook up some wires to the Input and Output pins. You can create the beginnings of a wire by hovering over the small black dot on the side of the rectangle representing the pins. When your mouse cursor hovers over this, you should notice a green circle. Hold the mouse button down and drag outward to start creating a wire.

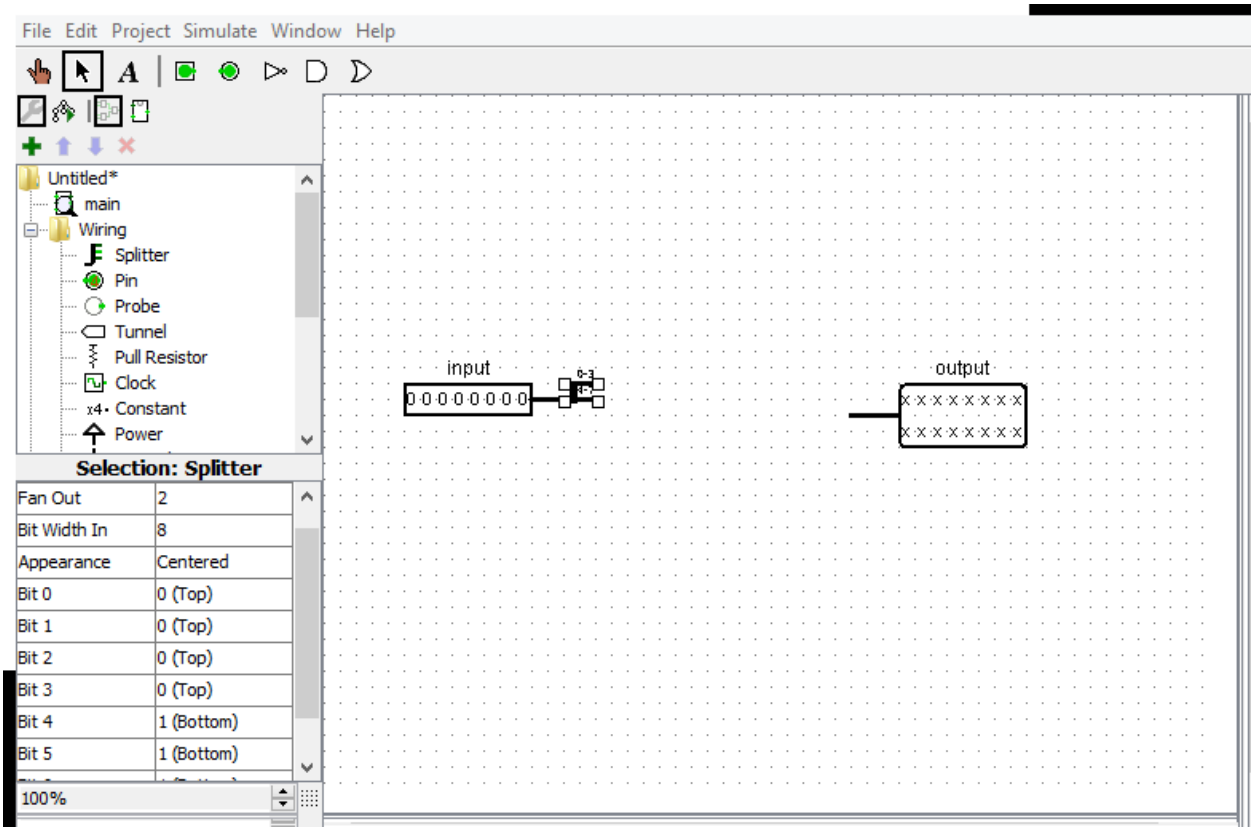


Next, we need to set up a circuit that will (1) copy the 7 least significant bits of the input to the 7 least significant bits of the output, and (2) copy the most significant bit of the input to the most 9 most significant bits of the output. To accomplish this, we need to use the **Splitter** tool (see the first icon under **Wiring** in the image above). Splitters are useful when we want to handle different input bits in different ways, and when we want to output bits that we combine from multiple sources.

First, we need to create a splitter to divide up the Input bits:

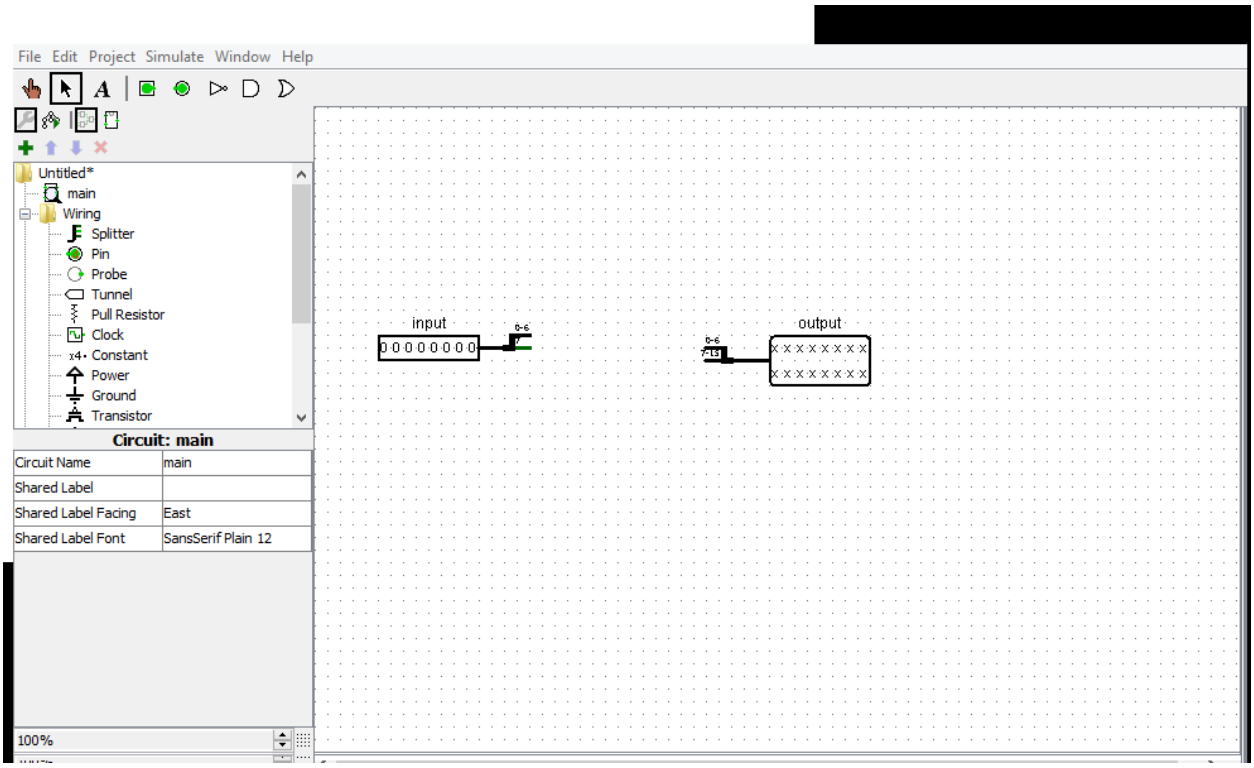
1. Create a new **Splitter** on the Canvas by using the same steps you used for the pins.
2. Select your new **Splitter** with the Select tool. In the Attribute Table:
    a. The **Bit Width** should be changed to the number of bits in our Input (8).
    b. Attach the Splitter to the wire that you created for the input pins. This can be done by simply dragging the splitter next to the wire. You want the side with a single prong (instead of the "fan out" side) to be hooked up to the Input. To confirm that

the Splitter is truly attached, try moving it back and forth; you should see the wire expanding or contracting accordingly.

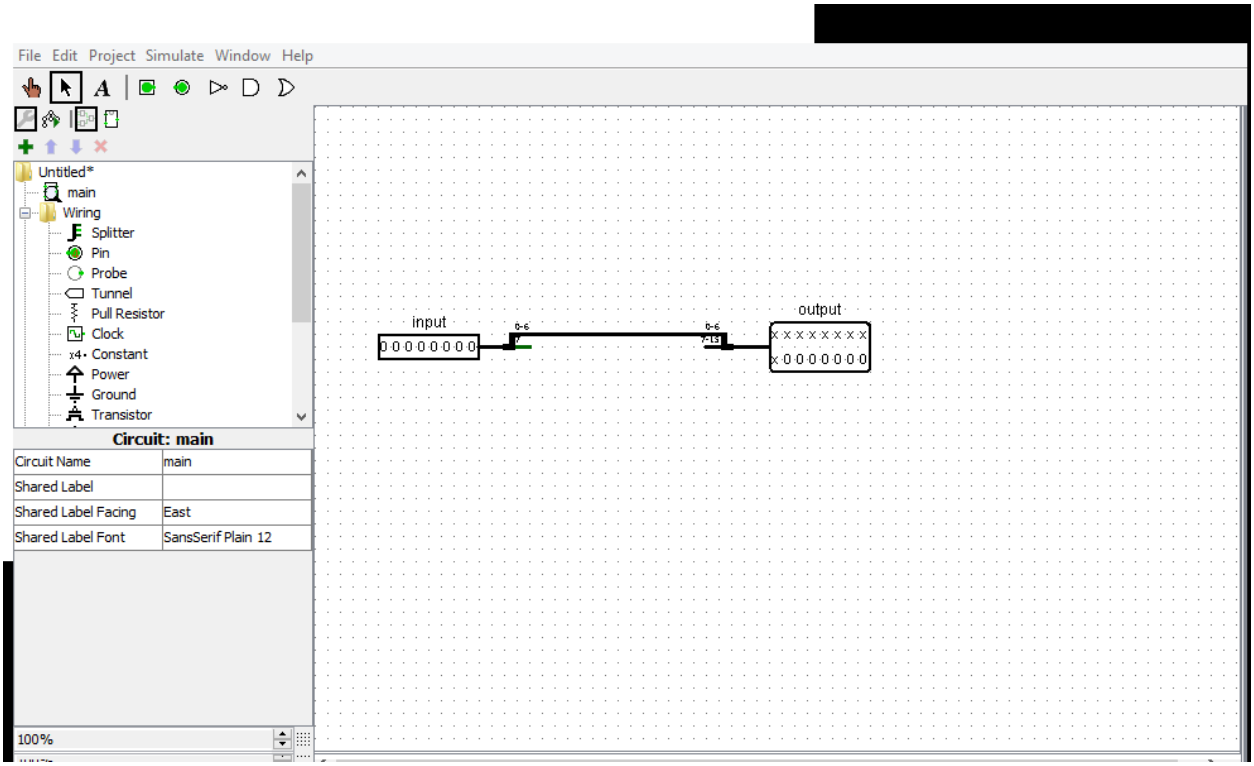


3. In the Attribute table, the entries **Bit 0…Bit 7** let us decide how we want to divide the different bits of the input amongst the two different output wires. In this case, we want to treat bits 0-6 and bit 7 differently.
   - Assign bits 0 through 6 to the **0 (Top)** output of the splitter; bit 7 can remain assigned to **1 (Bottom)**. Now we can send these bits through two different paths in our circuit.

4. For the output, we will need to combine the results of these two different paths. The top path will copy the 7 least significant bits of the input, and the bottom path will send a 9-bit-long copy of the most significant bit of the input. To the output pins, we can hook up a splitter that will accept these two different sets of bits.
   a. Create another **Splitter**. You may find it helpful to change the facing direction to be opposite that of the **Splitter** heading out of the input pins.
   b. Hook up the **Splitter** to the Output, again using the side with only a single prong.
   c. Set **Bit Width In** to the number of bits in the Output (16).
   d. In the Attribute Table, the fields for **Bit 0….15** are set, by default, to be equally split between the top and bottom splitter outputs. However, we only want bits 0-6 to go through the top route, and bits 7-15 should go through the bottom route. So **Bit 7** should be changed to **1 (Bottom).**

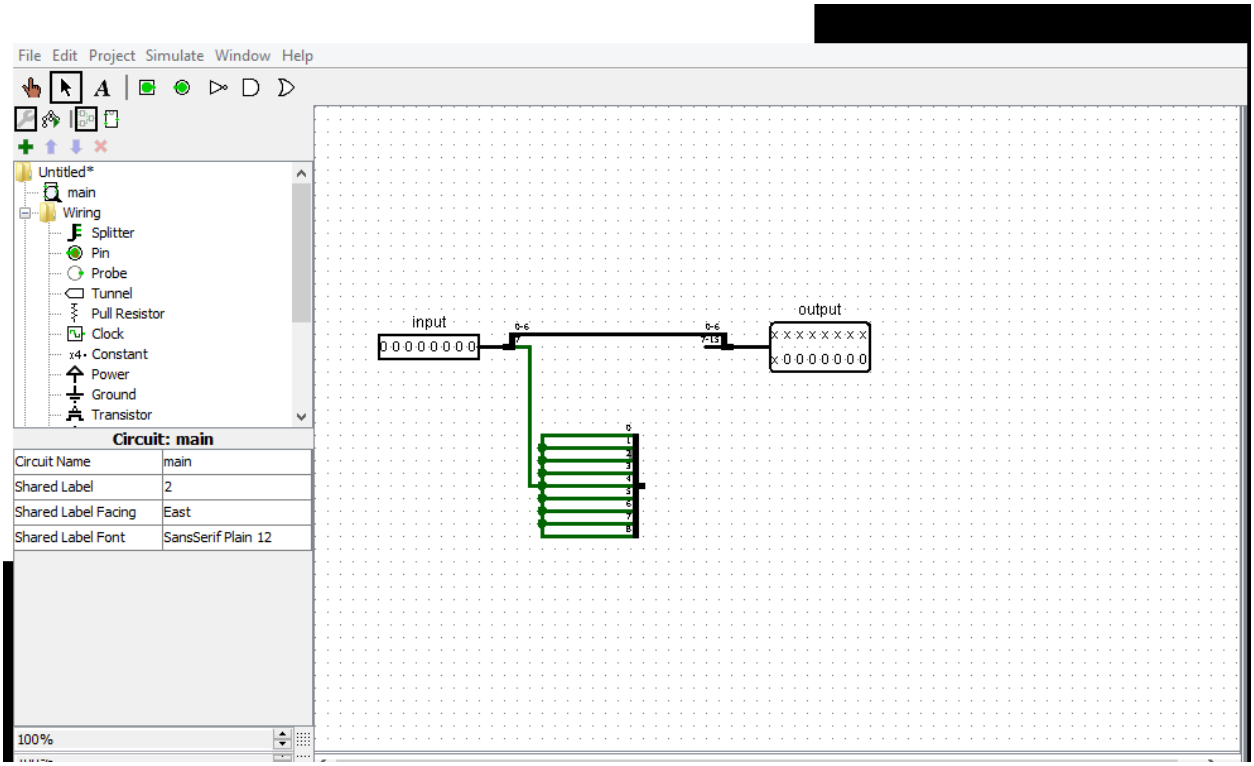


Now, we can build the path that copies bits 0-6 from the input to the output. This can be achieved by drawing a wire to connect the top output of the Input splitter to the top input of the Output splitter:

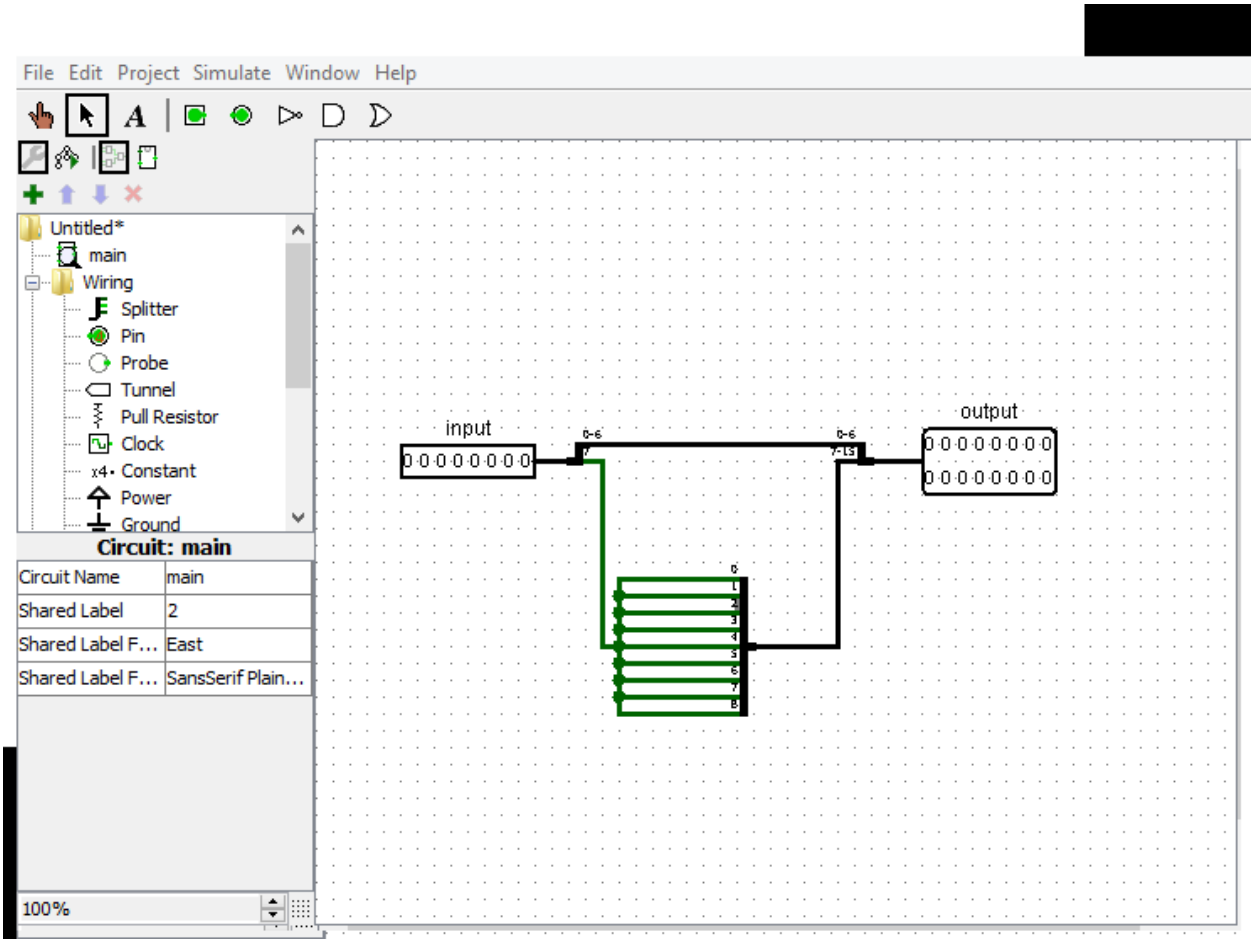


For the path that performs the sign extension, we can use a third splitter to combine the original Bit 7, along with 8 copies of it, into a single concatenated input that we will feed into the bottom prong of the output splitter:

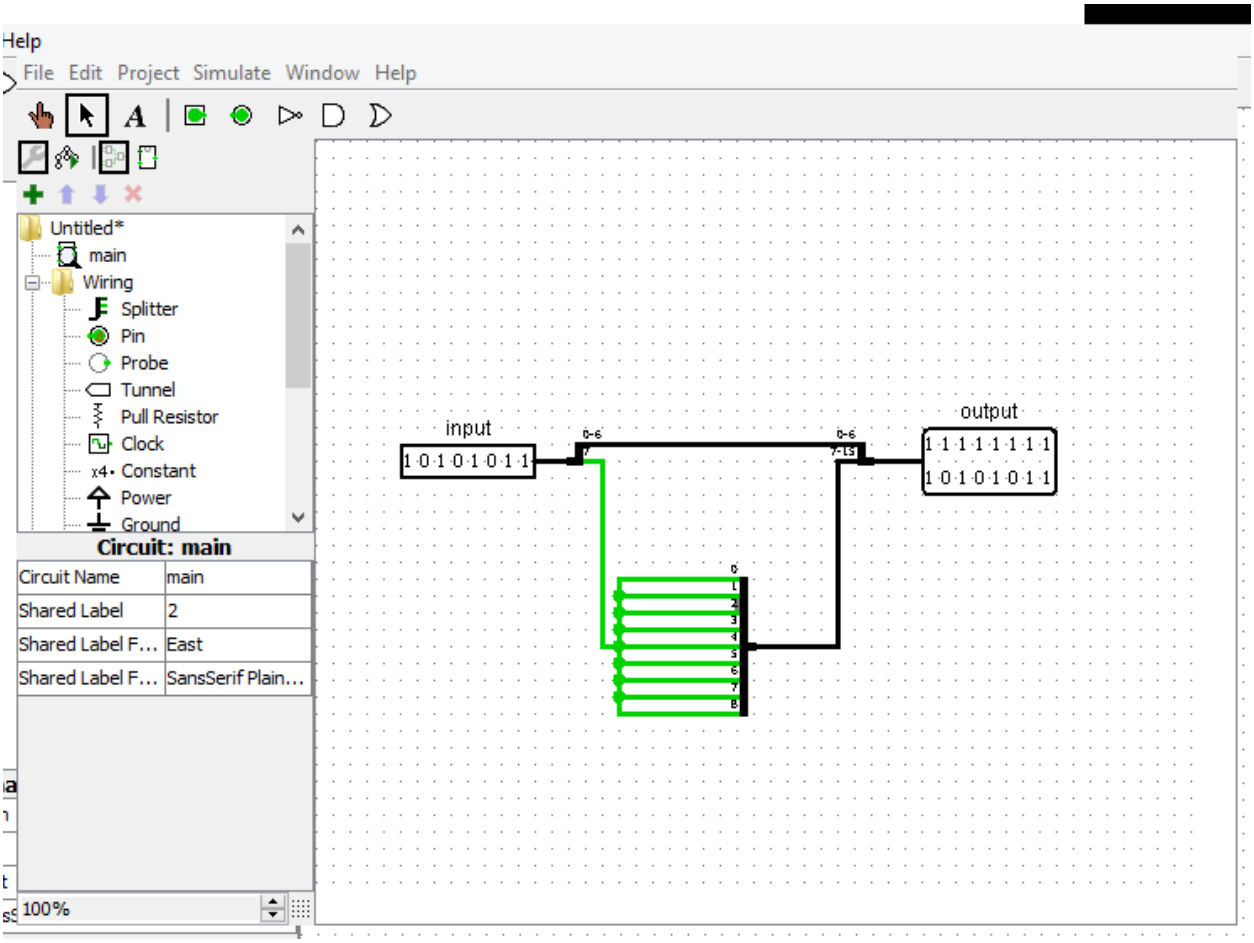
1. Create a new splitter with the “Fan Out” side facing the input pins.
2. Change **Bit Width In** to 9.
3. Change **Fan Out** to 9.
4. Next, hook up the wire emerging from the bottom prong of the Input splitter to all 9 prongs of the 9-bit splitter, as shown below.
    a. To make new paths that diverge from an original wire path, click and hold the mouse button at a point on the wire, and then drag out the new branch of the wire. (You can also drag this new segment back to its point of origin in order to delete it).
    b. Each of these new paths will carry the same value as was on the original single wire – effectively resulting in 9 copies of the bit.



5. Finally, connect the prong on the opposite side of the 9-bit splitter to the bottom prong of the Output splitter. This will send bits 0 to 8 of the 9-bit splitter to bits 7 to 15 of the Output.



At this point, you should now have a functional sign extender. You can test it out by using the poke tool on the Input pins. So long as the MSB remains as 0, you should only see the least 7 significant bits copied to the output. If you set the MSB of the input to 1, then you will notice that the bottom row of the output still matches the input, and the top row should be set to all 1s. As shown on the next page, the wires leading into the 9-bit circuit will turn light green, indicating that these wires are carrying a value of 1.



You may have noticed that Logisim includes a Bit Extender tool under Wiring. If you would also like to try to use this, that is fine, so long as you can also demonstrate a circuit that uses the splitter approach from above.

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_lab6.circ** and submit the file via the appropriate link in Courseweb (see Course Documents/Week 7 Lab). The lab must be submitted by **Tuesday, October 18th**, by **11:59 pm**.