METU NORTHERN CYPRUS CAMPUS

EEE 445 COMPUTER ARCHITECTURE
CNG 331 COMPUTER ORGANISATION

TERM PROJECT: PART #1

FALL 2023

PART #1: BUILDING AN 8-BIT ALU

This part will participate in 5% of the overall course grade (out of 30% which is the percentage of the overall term project). This five percent will divide as follows:

- **20% for functionality and test of the 8-bit full adder.**
- **20% for functionality and testing of each ALU operation.**

Remember: Your labeling (circuits and inputs/outputs labeling) must match the description in this part. Otherwise, you will lose points. In addition, your mux selection (ALU controller) MUST match the table listed in this part.

Blue font color represents the steps you need to finish in this part.

Red font color represents rules, important information and regulations.

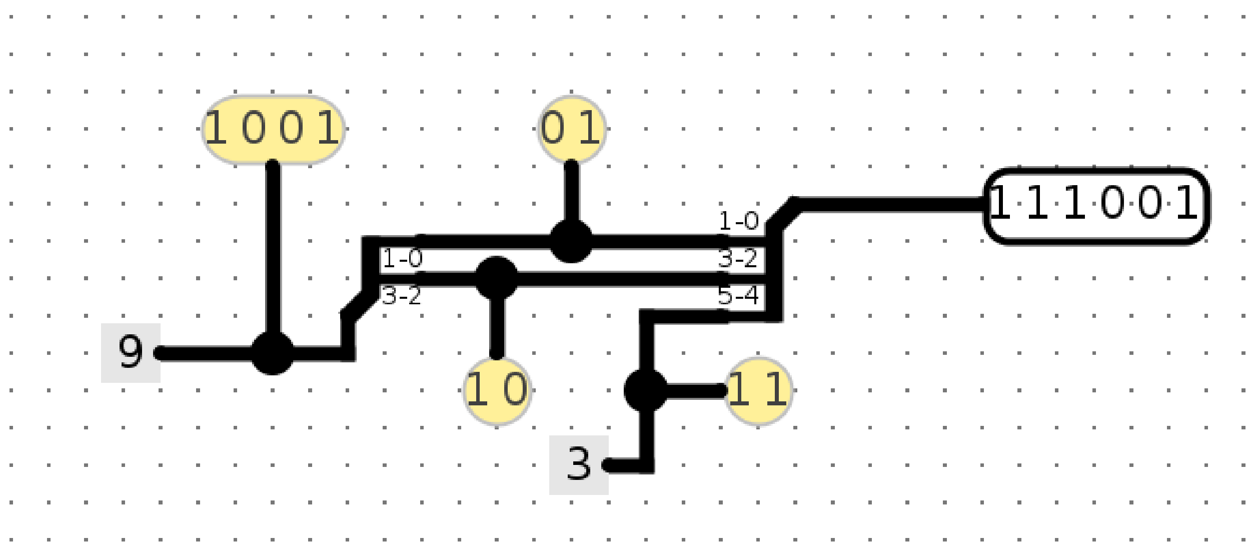
1.1 OBJECTIVE

This part consists of two sections. In the first section, you will learn how to use further essential parts of Logisim; in particular, you will use splitters to split a multi-bit value into component parts and combine component parts into a multi-bit value. In the second section, you will design an 8-bit basic ALU. In the end, you will attach your ALU to a simple system consisting of a file register, controller, and memory.

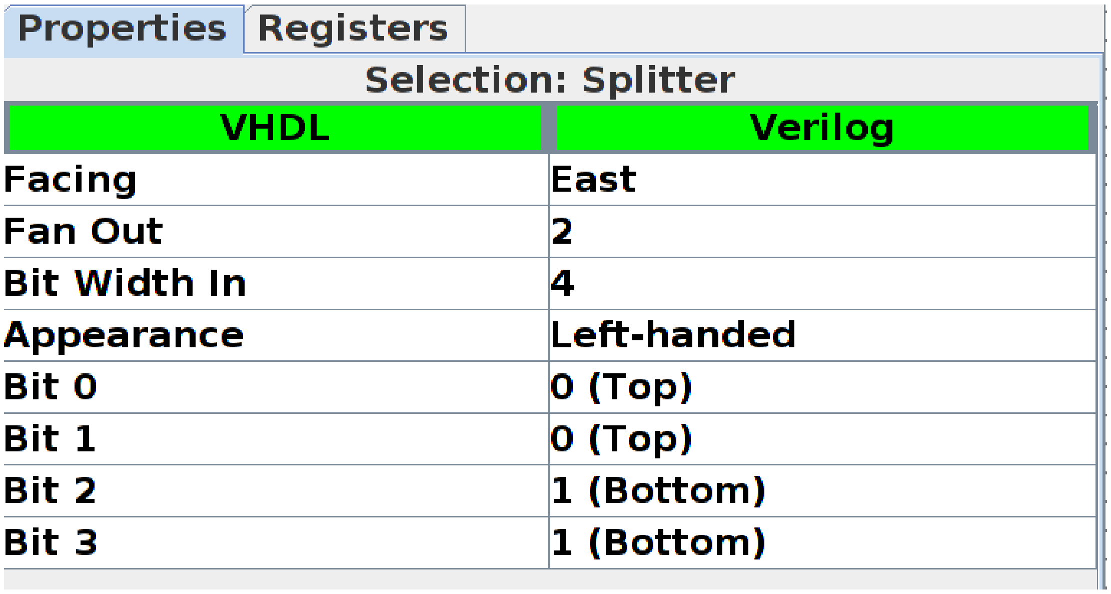
1.2 ADVANCED LOGISIM FEATURES

1.2.1 Splitters

A multi-bit value can be divided up into smaller pieces using splitters, or (despite their name) several values of one or more bits can be combined into a single value. In this case, the 4-bit binary number 1001 is divided into 10 and 01, and then it is added back to 11 to form the final 6-bit number 111001:



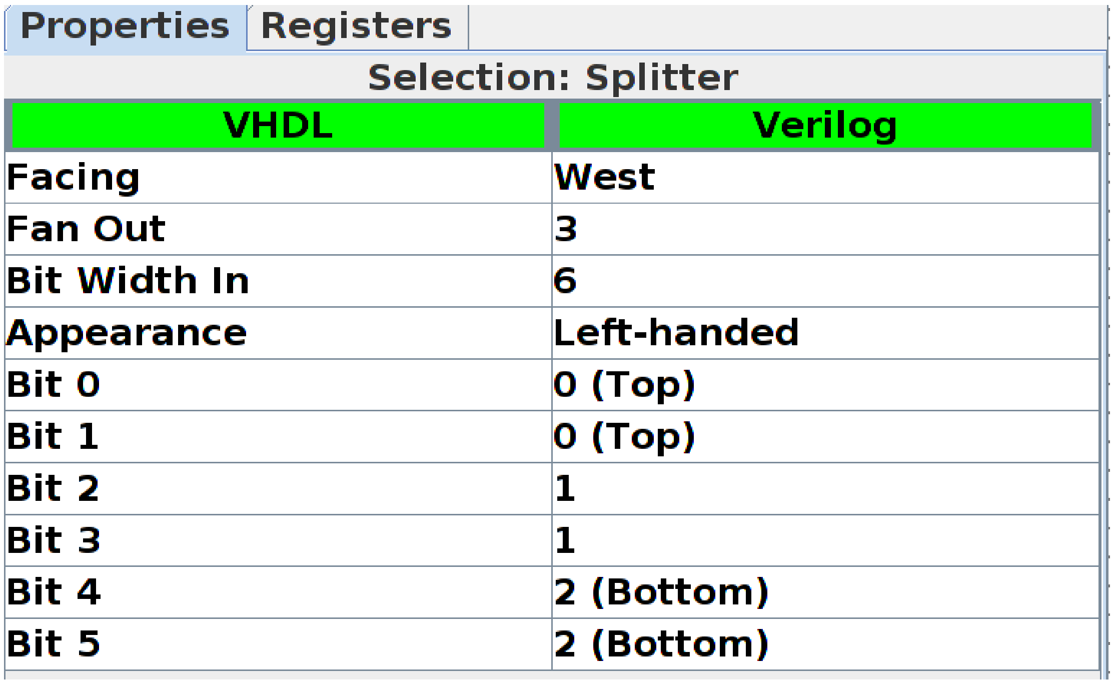
To access a splitter's menu on the sidebar(properties), click on it. You can choose how many arms your splitter should have from this menu, as well as how many bits each arm should hold. The left splitter's menu for the circuit above looks like this:



Selection: Splitter

VHDL	Verilog
Facing	East
Fan Out	2
Bit Width In	4
Appearance	Left-handed
Bit 0	0 (Top)
Bit 1	0 (Top)
Bit 2	1 (Bottom)
Bit 3	1 (Bottom)

While the menu for the right splitter looks like this:



Selection: Splitter

VHDL	Verilog
Facing	West
Fan Out	3
Bit Width In	6
Appearance	Left-handed
Bit 0	0 (Top)
Bit 1	0 (Top)
Bit 2	1
Bit 3	1
Bit 4	2 (Bottom)
Bit 5	2 (Bottom)

Notice that there’s an option called facing. You can use this to rotate your splitter. Above, see that the splitter on the right is facing West while the splitter on the left is facing East.

To understand this tool, you will demonstrate a circuit that uses splitters to manipulate 8-bit number.

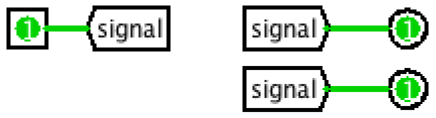
1. Create a new subcircuit.
2. Add an 8-bit input pin and label it.
3. Add 1-bit and 8-bit output pins labeled “O1” and “O2”.
4. From the wiring library, choose splitter.
5. Change the “Bit Width In” property to 8 (bus width) and the “Fan Out” property to 3 (number of branches).
6. Now, select which bits to send out to which part of your fan. The least significant bit is bit 0 and the most significant bit is bit 7. Bit 0 should come out on fan arm 0, bits 1, 2, 3, 4, 5 and 6 should come out on fan arm 1, and bit 7 should come out on fan arm 2. FYI: the None option means that the selected bit will not come out on ANY of the fan arms.
7. Place the splitter after configuring it to your main circuit.
8. Route In1 to the splitter. Attach a 2-input AND gate to fan arms 0 and 2 and route the output of the AND gate to O1.
9. Now, interpret the input as a “sign and magnitude” number. Place logic gates and other circuits to make O2 to be the negative “sign and magnitude” value of the input. Sign and magnitude is an alternate way of representing signed values - like 2’s Complement, but simpler! The combinational logic should be straight-forward.
10. We will need another splitter to recombine the fans into a single 8-bit bus. Place another splitter with the proper properties (Bit Width In: 8, Fan Out: 3, correct fan widths). Play with the Facing and Appearance properties to make your final circuit as clean-looking as possible.
11. Take a screenshot of your schematic (circuit you build through steps 1-10) and add it to your report with an explanation and a test value (using poke tool).

Next, you will be asked to Implement a circuit to rotate 8-bit value using the knowledge you have gained so far. (Remember you must use splitters, and you can’t use the already built rotate or shift circuits that Logisim provides)

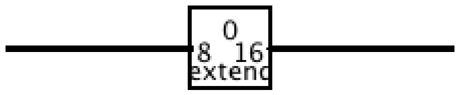
1.2.1.1 Implement two subcircuits, one to rotate right an 8-bit input value and a second to rotate an 8-bit input value left. Hint: you will need two inputs for each subcircuit, one is the input to be rotated, and the second input will hold the rotation amount. Call the first input A and the second Input B. What is the size of input B? And why? Answer this question and add screenshots of your subcircuits with an explanation to your report. Name the subcircuit “rotr” and “rotl” and upload them to ODTUClass Part #1 submission folder as (.circ).

1.2.2 Tunnels and Extenders

A Tunnel allows you to connect two points with an invisible wire. They can be used as follow:



An extender is used to change the width of wire. The following example shows an implementation of extending 8-bit wire into 16-bit wire.



You can find “Bit Extender” tool from wiring library. And from properties you can change input and output widths.

1.3 BUILDING AN ALU

You will start by building subcircuits such as an 8-bit full adder and sign extension, then put everything together to build a simple ALU with four operations. Lets start with what we learn so far.

1.3.1 build an 8-bit full adder using the same method explained in the tutorial examples. Add a screenshot of your schematic and test results (USING A TEST VECTOR YOU WILL WRITE). Attach the circuit and test file to odtuclass.

Note: you MUST label inputs and outputs as shown in the figure, you will have three inputs: X,Y, and CIN, along with two outputs: SUM and COUT. In addition, you MUST label your circuit FULLADDER8. SAVE THE CIRCUIT AS: <YOUR NAME>_<YOUR ID>_FADDER8.CIRC

FOR EXAMPLE: Othman_2455921_FADDER8.circ

You must build the full adder circuit using logic gates. You can’t use the full adder circuit that Logisim provides.

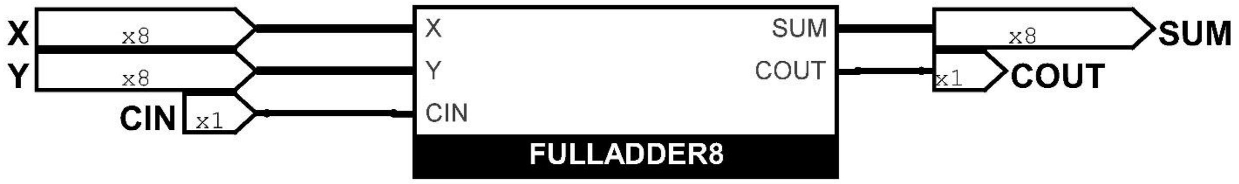


Figure 1.1: 8-bit Full Adder.

Next, you will use the following requirements you will build an 8-bit ALU:

1. your ALU will perform four arithmetic functions on two 8-bit inputs, resulting in one 8-bit output and five 1-bit outputs (condition code flag).
2. You will have 2-bit input to select which function to perform called opcode.
3. To implement this in your circuit you need to use a multiplexor with four inputs, one for each function and two select lines. **(In Logisim, be sure to change "Include Enable?" from "Yes" to "No").**

The function that your ALU will perform depends on the opcode is summarized in the following table.

Opcode	Function
00	Add
01	And
10	Or
11	Rotate left

Hint: you can use “rotl” subcircuit to perform rotate left function.

Your ALU will output the following flag bits:

- EQ flag: this flag will set to 1 when x is equal to y.
- OVF flag: overflow flag will set to 1 when requested operation results in signed overflow.
- CF flag: the carry flag is set to 1 when the requested operation results in unsigned overflow.
- ZF: the zero flag is set to 1 when the results equal to zero.
- SF: the sign flag is set to 1 if the result is negative (when interpreted as an 8 bit 2’s complement value)

1.3.2 Draw a circuit layout, build it using Logisim, and write a test to test all the functions. You need to add the layout along with the test results (USING A TEST VECTOR YOU WILL WRITE) to your report. In addition to that, you need to upload the .circ file along with the ALU8_test.txt to ODTUClass.

Note: you MUST label your inputs: X,Y and your outputs: RESULT,ZF,SF,CF,OVF,EQ and label your circuit: ALU8. ALSO, YOU MUST SAVE YOUR CIRCUIT AS: <NAME>_<ID>_ALU8.CIRC

FOR EXAMPLE: Othman_2455921)_ALU8.circ

AN EXAMPLE OF A TEST VECTOR WILL BE UPLOADED TO ODTUCLASS.

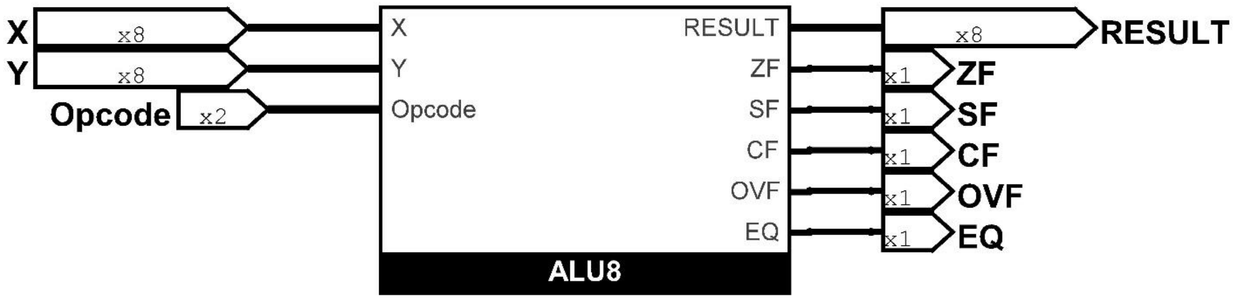


Figure 1.2: 8-bit ALU.

TIPS:

- I suggest that you **test your ALU circuit often as you go**; add a little functionality, stop and test that functionality by trying out different input values using the poke tool, once that part works, then add a little more functionality …
- If you have an 8-bit number and you want to access just the bits individually, you can use the Splitter located under Wiring. If you set "Fan Out" to 8 and "Bit Width In" to 8, you'll be able to turn any 8-bit input into 8 1-bit outputs. You can also do the reverse by simply connecting 8 1-bit inputs to one side of the splitter and getting 1 8-bit output back out.
- To implement the memory and register file, you need to use the tools the Memory folder library provides.

1.4 Submission

YOU WILL SUBMIT A ZIP FILE CONTAINS: 8-BIT ADDER AND ALU CIRCUITS ALONG WITH YOUR REPORT TO ODTUCLASS BEFORE SATURDAY 5 SEPTEMPER 21:00.

THERE WILL BE TWO ASSIGNMENT SUBMISSIONS ON ODTUCLASS. ONE FOR THE REPORT AND ONE FOR THE CIRCUITS ZIP FILE.

Please save your .circ files as follow: <your name>_<your ID>_<circuit name>.circ

For example: Othman_245592_ALU8.circ

Remember you will have the following circuits inside your zip file:

- 8-bit adder
- 8-bit ALU
- Rotate left
- Rotate right

You will have two test vector files inside your zip file for ALU and FULL ADDER circuits.

THIS PROJECT IS INDIVIDUAL WORK, AND ANY CHEATING WON'T BE TOLERATED AND WILL BE PENALIZED!

ANY LATE SUBMISSION WILL BE PENALIZED 15% OFF FOR EACH DAY AFTER THE DEADLINE. SUBMISSIONS LATER THAN TWO DAYS WON’T BE ACCEPTED.