# Week 2 Lab B: Arithmetic and Logic

## Objectives

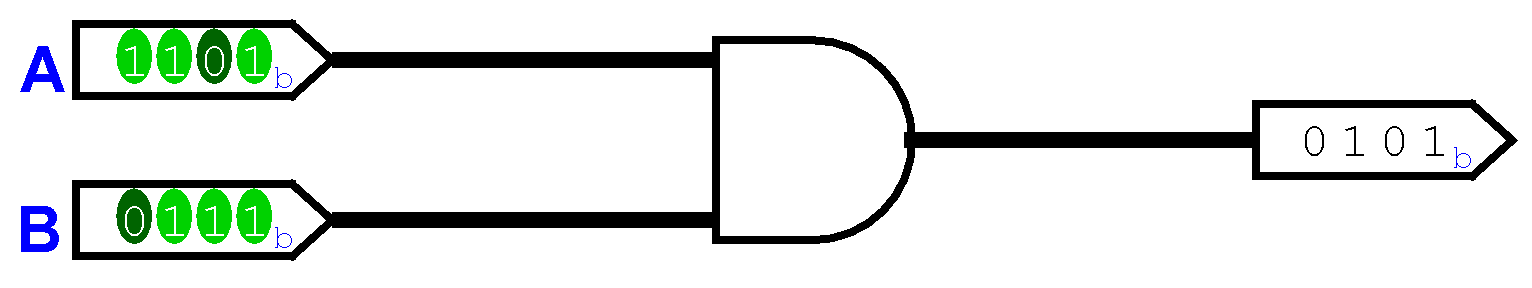
Develop understanding and experience of:

1. Using built-in components that handle multiple bits at a time in Logisim Evolution

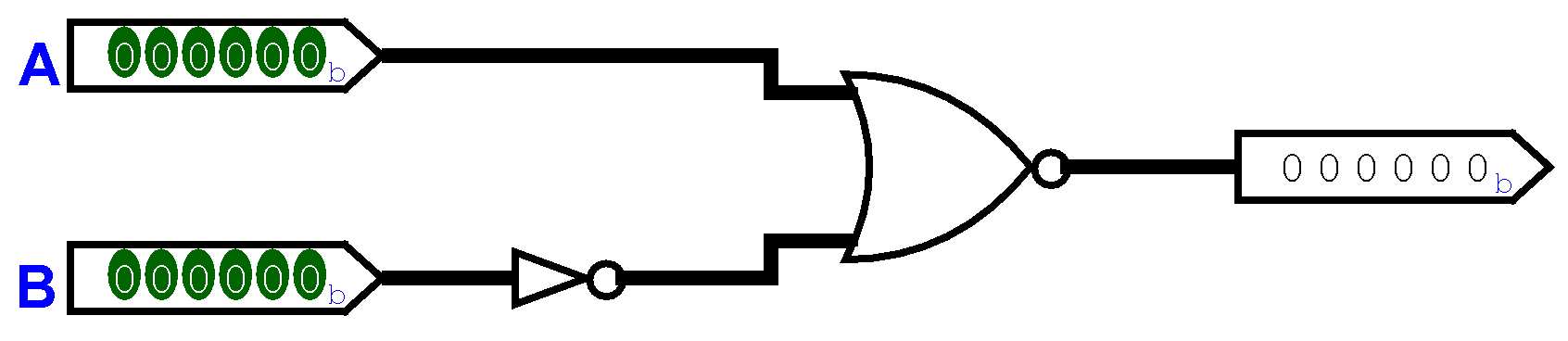
## Bitwise Logic Operations

When we work with basic logic gates and multiple bits, they work ***bitwise*** on each bit in the signal in turn.

1. The figure below shows a bitwise AND operation. The output is 1 where both the inputs are 1.



1. What would the output be if the AND gate was changed to an OR gate? 1111
2. What would the output be if the AND gate was changed to an XOR gate? 1010
3. The following circuit outputs 0 when the inputs are both zero.



What would the output be in the following situations (you may need to make rough notes for working out):

1. Binary values: A is 011001 and B is 001101

011001

110010 NOR

000100

1. Decimal values: A is 7 and B is 16: In binary A is 000111 and B is 010000

000111

101111 NOR

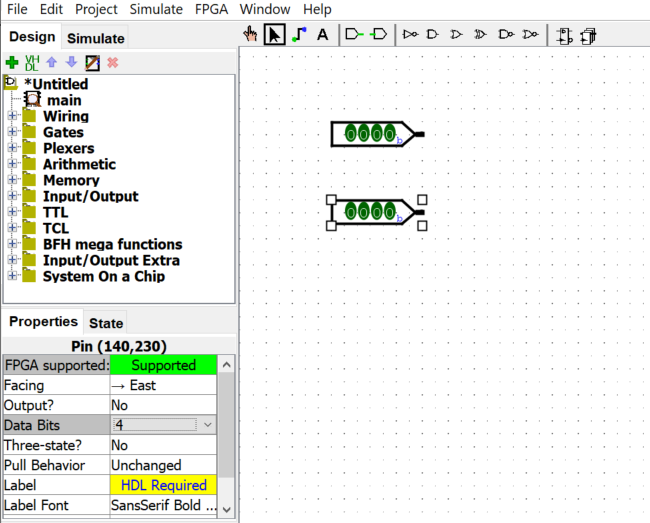
010000

1. Hexadecimal values: A is 0xA and B is 0xF: A 001010 and B is

001010

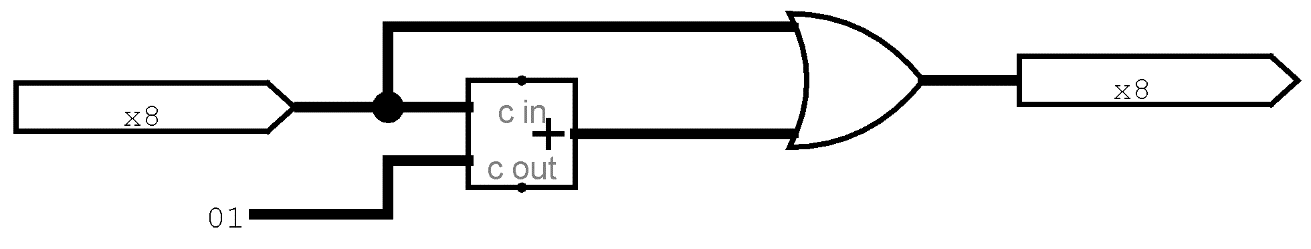
110000 NOR

000101



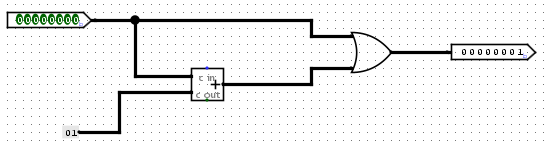
In the Figure above, I have set the Data Bits property to 4 for the input pins. I changed the Data bits attribute value for **all** the components in the figures above.

1. Mystery circuit A. Create the following circuit:



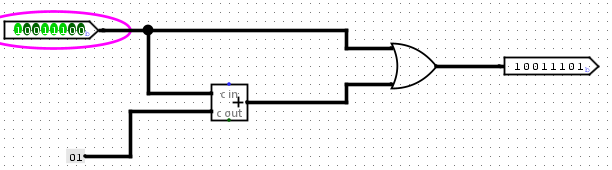
The circuit should handle 8-bit signals (indicated by the x8 in the input and output pins). The component with the + sign is an Adder (from the Arithmetic components in Logisim Evolution). The 01 on the left is an 8-bit constant (from the wiring components in Logisim evolution).

Test your circuit with different binary inputs. Paste images here that show some testing.



Summarise in words what the effect of the circuit is on the input.

If I set the constant to 1, and my value has no value (set to 8x0) the output is 1 due to the constant.



if I add value to the input, the output will add 1 to the input due to the constant.

1. Add three 8-bit numbers together.

The Adder component from Logisim Evolution takes two inputs (as seen in the previous question).

Create a circuit that adds together three 8-bit inputs to give one output. You will need two Adder components.

Paste images here that show some testing.

What limitations does your addition circuit have? Are there situations where it doesn’t give the correct result?

­

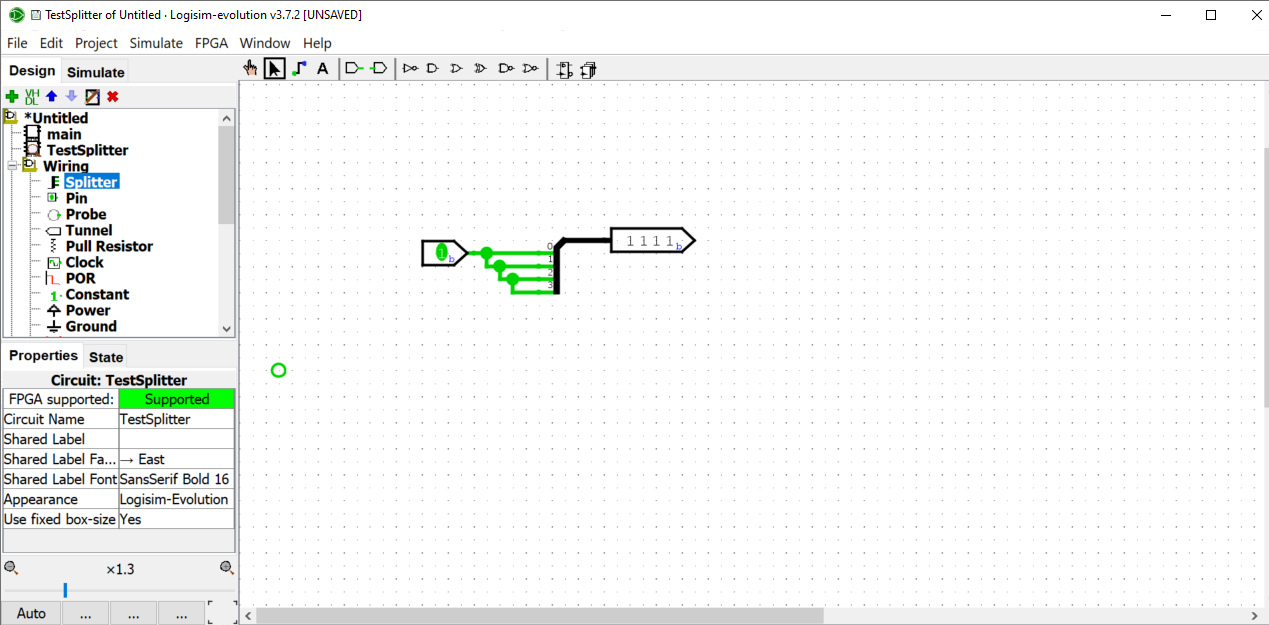
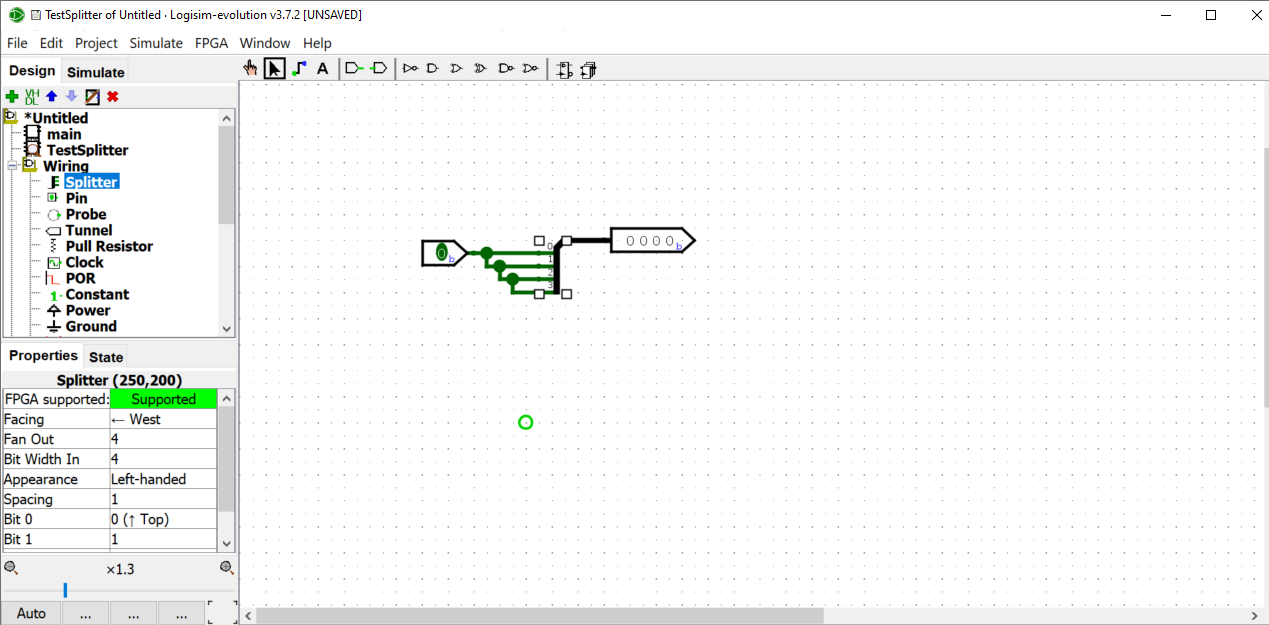
## Comparators

Computers need to have a process for comparing two values as it’s important for programs to be able to behave differently depending on values while a program is running.

When you change a gate to have 4-bit inputs, **all** the inputs need to be 4-bits so that it can do bitwise logic.

Sometimes we want to use the value from a 1-bit wire in all positions, that is using 0000 instead of 0; and 1111 instead of 1. This can be done in Logisim Evolution using a splitter (from the wiring tools) which can also join signals together.

In the examples below, I set the splitter to face East and have Fan Out and Bit Width In as 4. The output pin shows that the signal has been combined correctly (I would feed this to another component rather than just a pin).



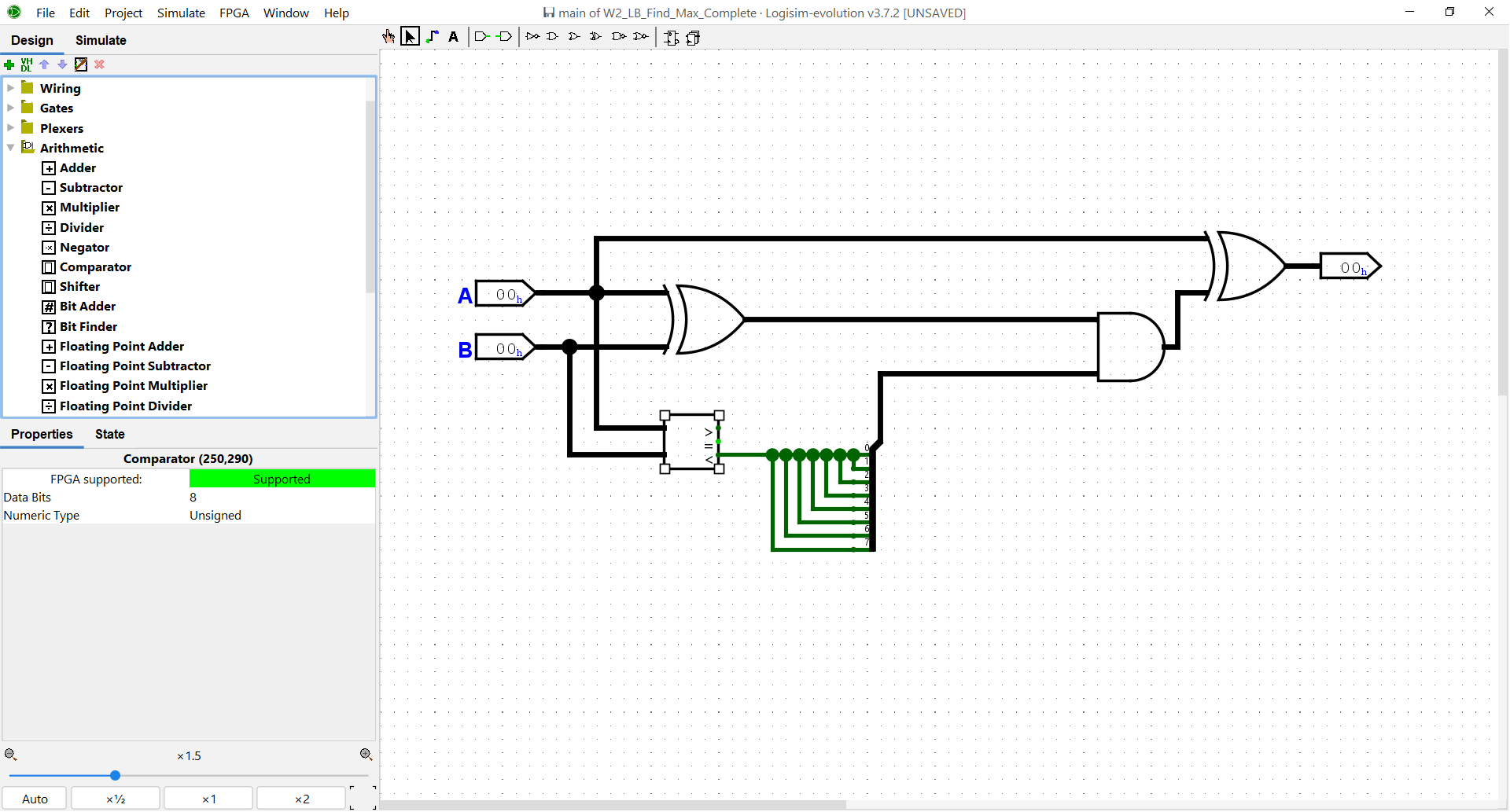
1. Create a component that checks whether a 4-bit input is equal to zero.

Use an input pin with Data Bits set to 4. Note that a 4-bit signal is zero when all the bits are zero so you will need to use a splitter to separate all the bits. You may use an LED from the Input/Output tools in Logisim evolution for the output. The output should be 1 (or the LED on) when all the bits of the input signal are zero, and the output should be 0 (or the LED off) otherwise (when any of the bits are 1).

You should use basic gates and wiring only. You may use gates with more than two inputs but must **not** use a built-in comparator component from Logisim Evolution for this task. There are multiple possible solutions to this.

Paste images here to show your circuit with various test inputs.

1. Mystery circuit B.



Download the circuit **W2\_LB\_Mystery\_B.circ** from the Starter Code on Moodle

Note that this circuit uses a built-in comparator component from Logisim Evolution and the Numeric Type property is set to Unsigned. A comparator takes two signals (both must have the same number of bits) and has three 1-bit outputs. Only one of the outputs will be 1 according to whether the first input is greater than, equal to or less than the second.

The mystery circuit handles 8-bit signals, the Data Bits property of all the components has been set to 8. The Radix property of the input and output pins has been set to Hexadecimal, so you see two hexadecimal digits.

Try the circuit with different inputs A and B. You can click on a digit from the inputs and key a valid hexadecimal digit using your keyboard. Paste some images of your testing.

Summarise in words how the output relates to the two inputs.

­

## Extension exercises

These exercises continue to work with multiple bits in creating components that would be needed for a computer. One of the exercises above used the comparator component built in to Logisim Evolution. To complete the aim of creating all required components for a simulated computer from logic gates, you should create one that works for a limited number of bits in the input.

1. Using wiring and gates, create a component which checks whether two 4-bit signals are equal to each other (do **not** use the comparator built in to Logisim Evolution). It should have a 1-bit output that is 1 if both 4-bit inputs are the same and output a 0 otherwise. Note that signals will be equal when each corresponding bit is the same in both inputs.
2. Now should build a comparator that compares two 2-bit inputs and sets one of three 1-bit outputs according to whether input A is greater than input B, input A is equal to input B or input A is less than input B. This is a limited version of the comparator component from Logisim Evolution.

Look at the W2 Concept Videos page on Moodle. The slides and video on adder and comparator show a 1-bit comparator and the truth table for a 2-bit comparator. Your task is to try to create a 2-bit comparator using basic logic gates.