# Week 4 Lab B: Computer building blocks

## Objectives

Develop understanding and experience of:

1. Decoder
2. Bit shifting

### Binary number revision

Write down the binary equivalent of all the decimal numbers from 0 to 7.

1. 0
2. 1
3. 10
4. 11
5. 100
6. 101
7. 110
8. 111

How many bits do we need to write any of the numbers from 0 to 7 in binary? 3

### Decoder

* A decoder will change a binary number to pass a signal to one (and only one) of its outputs by decoding that binary signal
* For example, if we have a decoder that takes in a 3-bit signal then we know that we can represent the decimal number 0 to 7 using those three binary digits. Therefore, the decoder would need 8 different outputs (one for each number from 0 to 7).

Diagram, schematic

Description automatically generated

Diagram, schematic

Description automatically generatedDiagram, schematic

Description automatically generatedWhich of the LEDs will be lit in the following two examples

Diagram, schematic

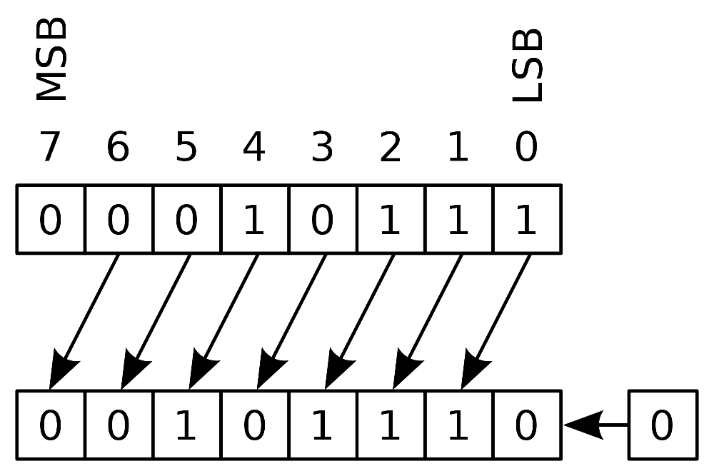
Description automatically generated

### Bit shifter

Shift components are often built into ALUs.

A logical shift moves all the bits a specified number of positions (left or right). The bits that drop off are lost. Zeros are inserted to fill the new places.

Left shift by 1 bit



Right shift by 1 bit

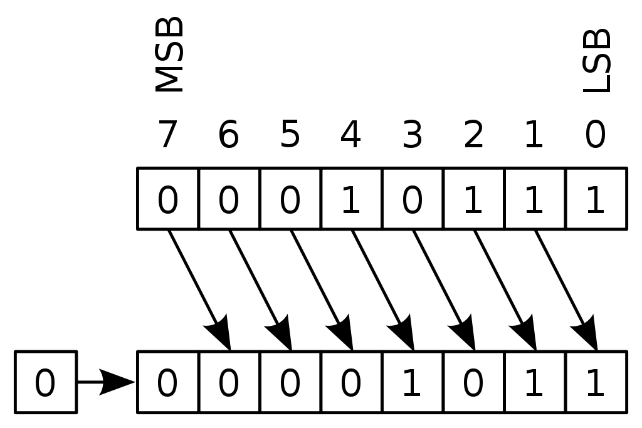


Image credit en:User:Cburnett, CC BY-SA 3.0 <http://creativecommons.org/licenses/by-sa/3.0/>, via Wikimedia Commons

Decimal 5 in binary is 00000101.

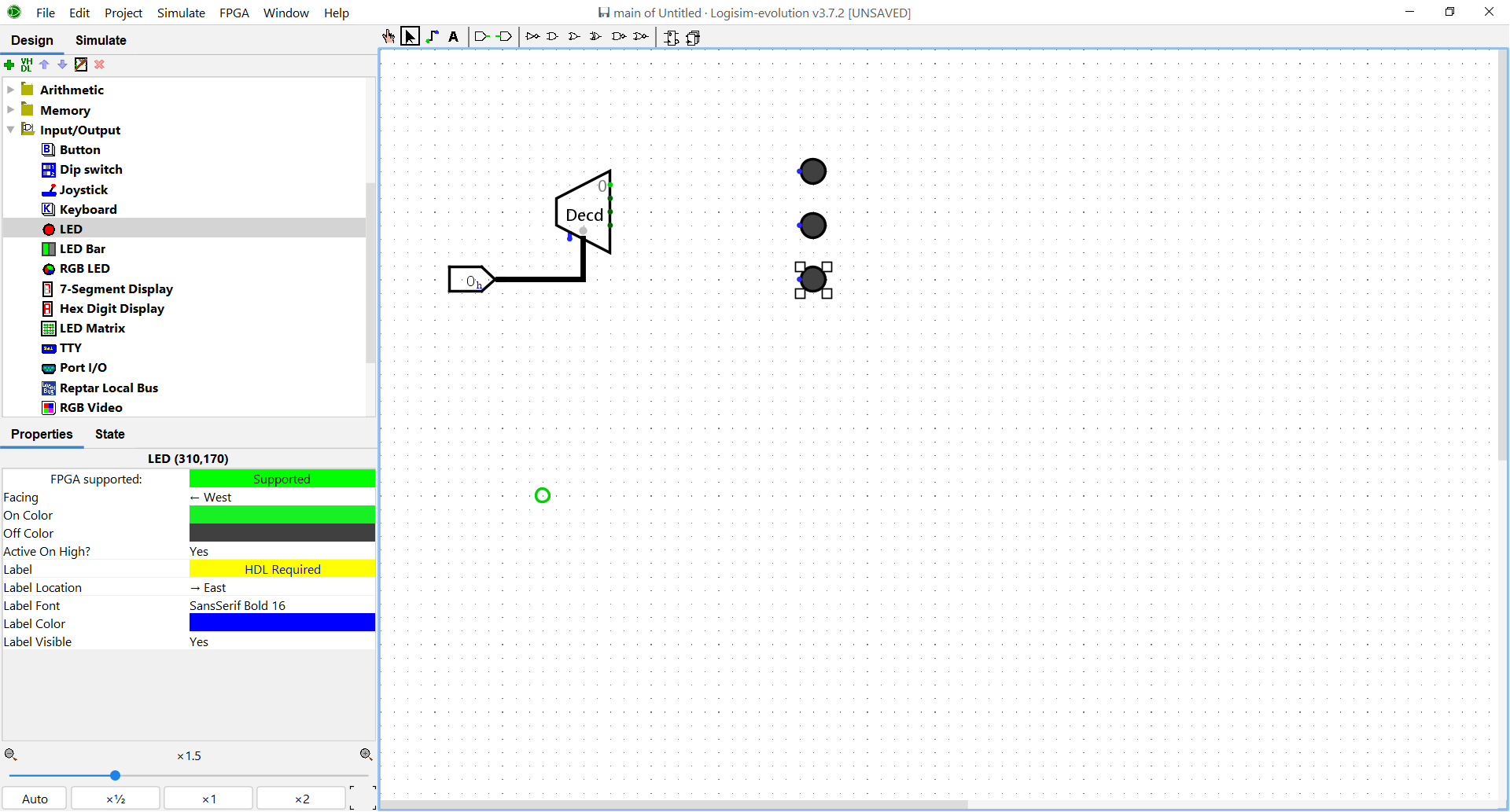
1. What will it become (in binary) after a logical left shift of 1 bit? 00001010
2. What is the decimal equivalent of that answer? 10
3. What has happened to the value? It has doubled
4. What effect would a logical right shift have? Halve the number.
5. What about a right-shift by 2 bits? Divide by 4.
6. Decimal 11 is binary 00001011. What do we get (in decimal) if we do a logical right shift by 2 bits? In binary we get 00000010 which is 2 in decimal.

### Use a Decoder

For this task you will use the decoder component from Logisim Evolution to create a traffic light simulation that has four stages as given in the table below.

|  |  |
| --- | --- |
| Stage | Colour |
| 0 | Red |
| 1 | Red and Amber |
| 2 | Green |
| 3 | Amber |

You might want to start as in the image below but will need some additional OR gates.



I have used LEDs from the Input/Output tools in Logisim Evolution and have changed the “On Color” property to match traffic lights (the green one is highlighted in the image above). The input pin has 2 bits and I have set the “Radix” property for the input pin to be Hexadecimal. When in simulation mode, I can click on the pin to cycle through the possible states to test it.

A diagram of a diagram

Description automatically generated

The logic for this is exactly the same as you could use in programming. Notice that when we ask when the red light needs to be on, we say when the stage is 0 and 1. When we convert it to a program we say “**If** the stage is 0 **or** 1 **then** switch on the red light”.

Here is a Processing (Java) program to do the same. Note the use of || for OR.

int y = 25;

int x = 50;

int lightSize = 40;

int stage = 0;

void setup()

{

size(200, 200);

frameRate(2);

}

void draw()

{

background(85);

fill(0);

ellipse(x, y, lightSize, lightSize);

ellipse(x, y + lightSize \* 1.2, lightSize, lightSize);

ellipse(x, y + lightSize \* 2.4, lightSize, lightSize);

if (stage == 0 || stage == 1)

{

fill(255, 0, 0);

ellipse(x, y, lightSize \* 0.8, lightSize \* 0.8);

}

if (stage == 1 || stage == 3)

{

fill(255, 200, 0);

ellipse(x, y + lightSize \* 1.2, lightSize \* 0.8, lightSize \* 0.8);

}

if (stage == 2)

{

fill(0, 255, 0);

ellipse(x, y + lightSize \* 2.4, lightSize \* 0.8, lightSize \* 0.8);

}

stage++;

if (stage == 4)

{

stage = 0;

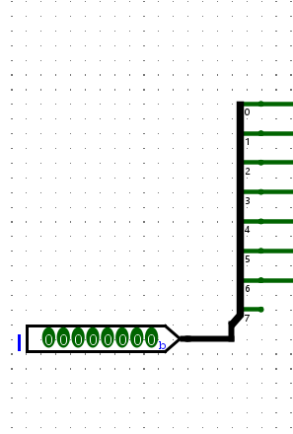
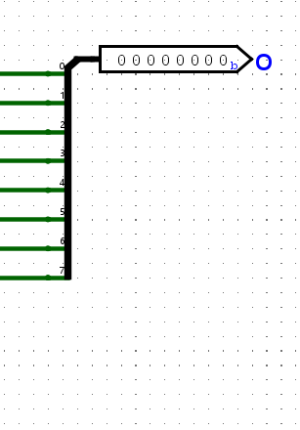
}

}

### Build a Bit Shifter

1. Build a left shift by 1 bit

Start with a 8-bit input pin and a 8-bit output pin. Use splitters to separate and join the wires between the two pins. Connect the wires to form a left shift by one bit. You will need to introduce a 1-bit constant zero. Give the circuit a suitable name and label the pins. You might want to start as shown below, you should adjust the properties of the splitters to make it as easy as possible to join the wires, I have changed the spacing.



Test your left shift and add an image to this document.

1. Build right shift by 1 bit

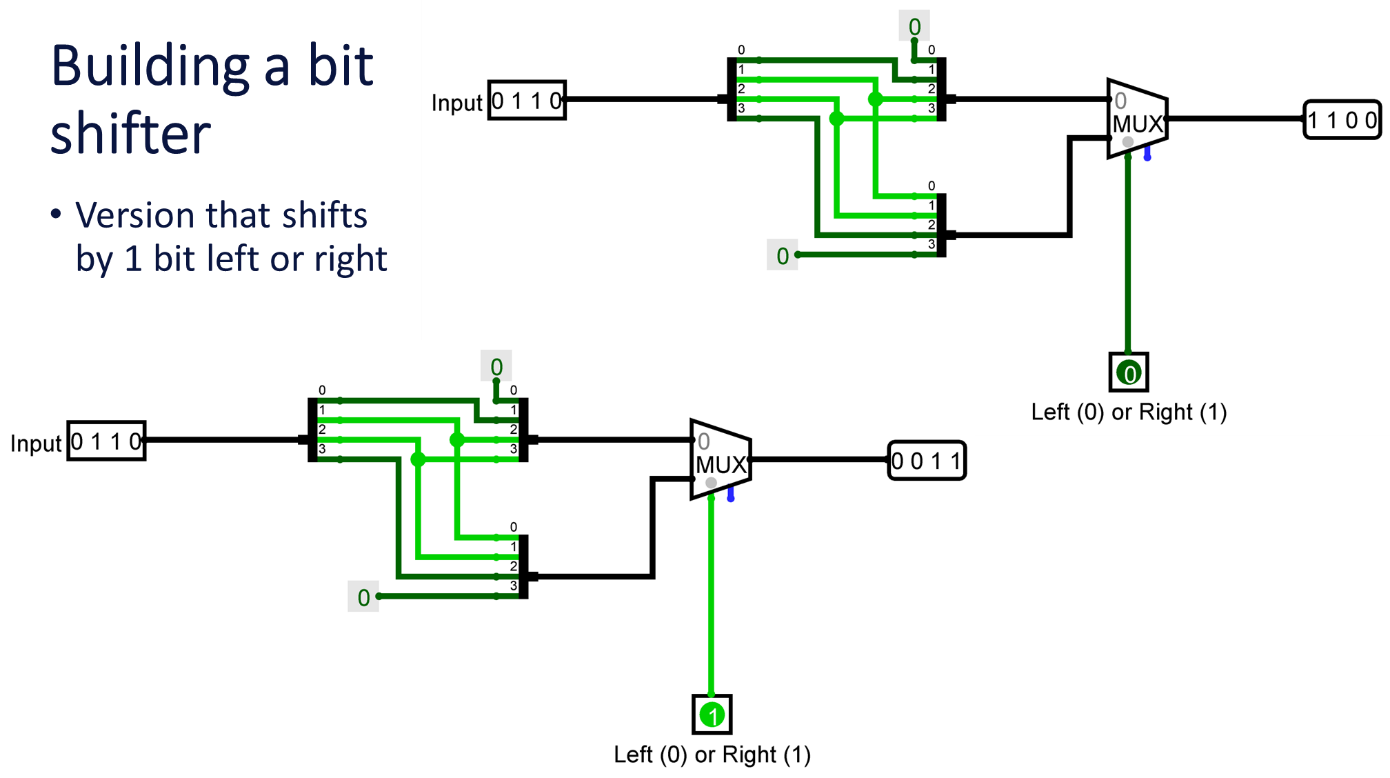
Create a new circuit inside the same Logisim Evolution project. Do the same process, but this time shift the input right by 1 bit.

Test your right shift and add an image to this document.

1. Create a shifter that gives the choice of shifting 1 bit to the left or the right.

Create another new circuit in the same Logisim Evolution project so that you can use your two shifters as sub-circuits. Add an 8-bit input pin, an 8-bit output pin and a 1-bit input pin to indicate whether to shift to the left (0) or the right (1). Add sub-circuits for both the left and right shift and pass the 8-bit input to both of these. Use a multiplexer to select which gets passed to the output.

Test your shifter and add images to this document.



My flexible bit shifter only caters for 4-bit inputs, but the same principle and use of multiplexer would apply to 8 bits.

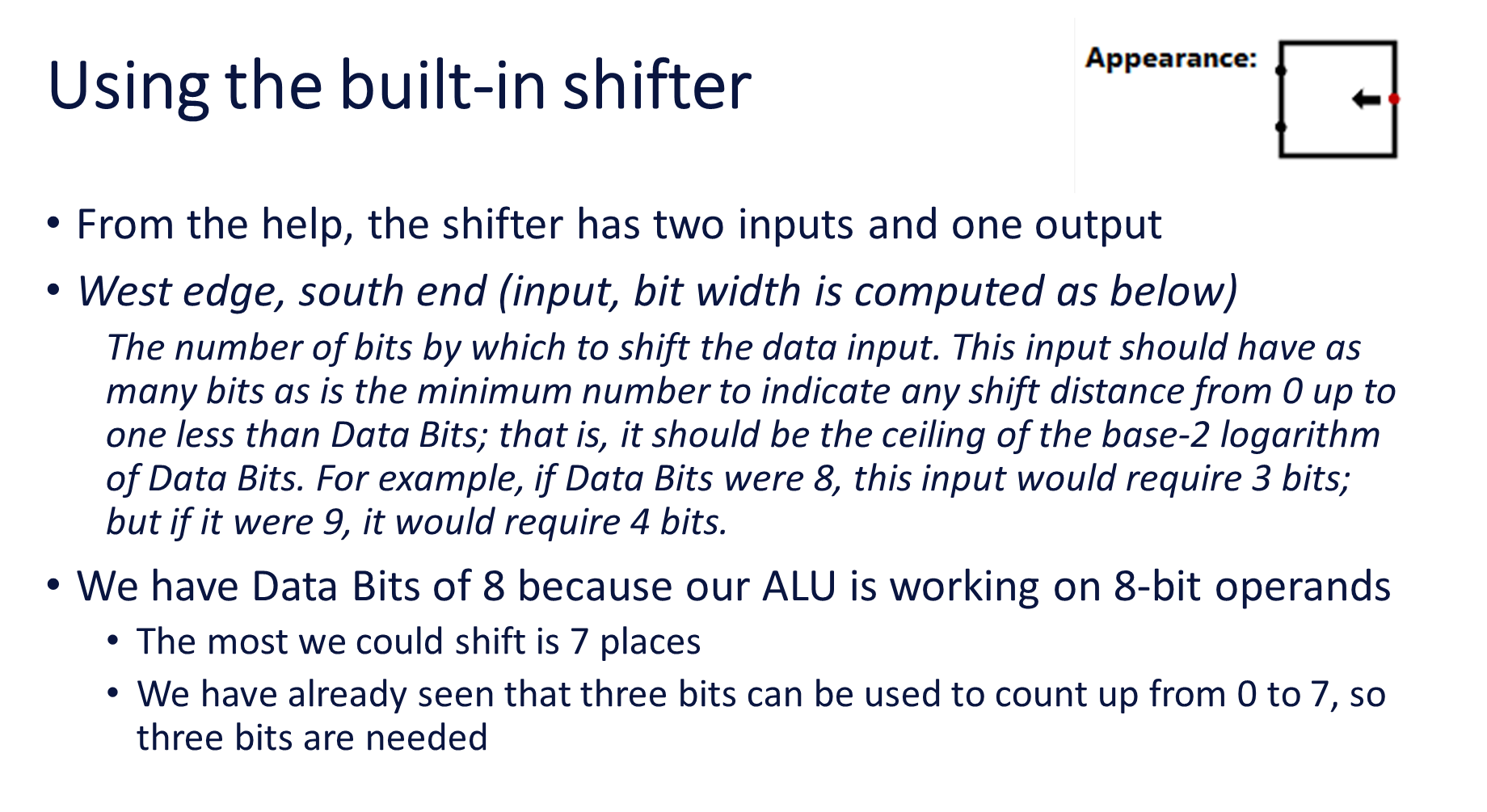
## Extension Exercises

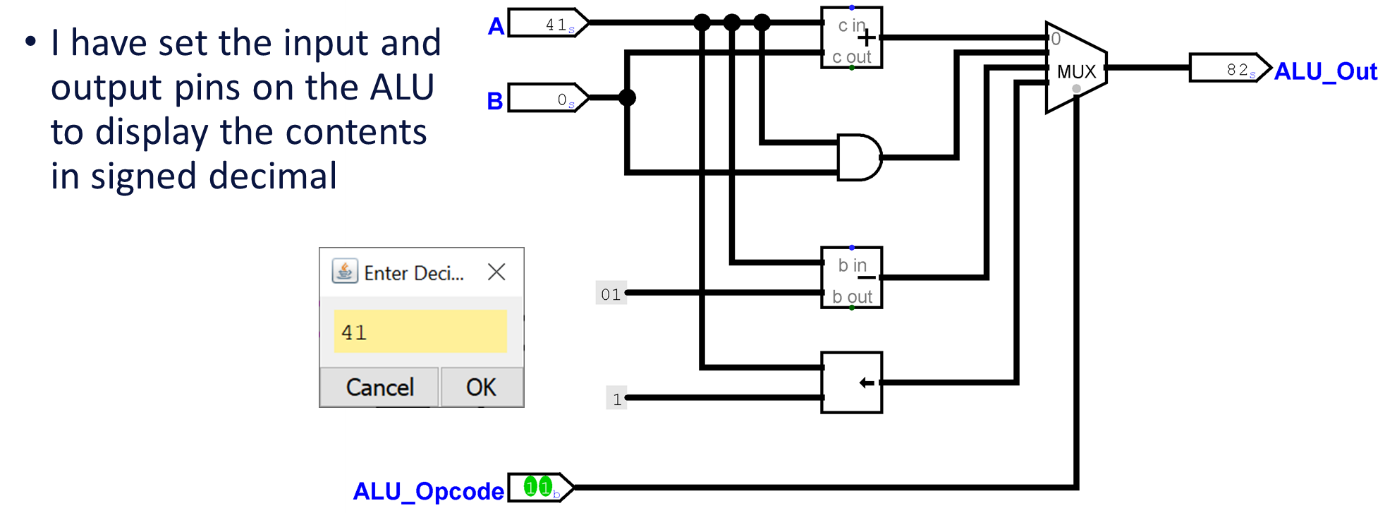
### Extend the ALU

You should have some experience with shift operations from the main part of this lab, so now we can add the shift feature for operation 3 so that the full set of operations is as shown in the table below. You may use the built-in Shifter from the Arithmetic components in Logisim Evolution but will need to use the Logisim documentation to find out how to use it.

|  |  |
| --- | --- |
| ALU OP value | Output |
| 0 | The result of adding A and B together |
| 1 | The result of a bitwise A AND B |
| 2 | The result of subtracting 1 from A |
| 3 | The result of shifting the bits in A left by 1-bit so that the rightmost bit will be filled with a zero. |

Make sure that you test your ALU as we will be using this ALU in week 6 as part of a simulation that can carry out instructions automatically, a simple computer.





### Building components

To complete the aim of creating all required components for a simulated computer from logic gates, you should try to create some decoders from basic gates and wires.

1. Build a **2-bit** decoder from basic logic gates in Logisim Evolution. Use the images from the slides to help you get started but note that the slides show a 3-bit decoder, so yours will be less complicated. When you test your circuit, make sure you test all possible combinations of input.

A diagram of a circuit

Description automatically generated

1. Make a new circuit and copy in the decoder you created above. The task is to convert it into a demultiplexer. Your demultiplexer should have an **8-bit** **data** signal that goes to one of four 8-bit outputs according to the input on a **2-bit select** signal. The outputs can be pins or LED bars, look at the input format property if using an LED bar. You should use AND gates with more than two inputs but will have to consider how to extend the select signal to 8 bits based on last week’s lab to pass through the AND gates. Note that pins in Logisim Evolution have a property called Radix which you can use to show the information in hexadecimal if you prefer. You should discuss your overall approach in small groups. Think about what combinations to use to be sure you have tested the circuit.

A diagram of a machine

Description automatically generated

1. If you have the flexible shifter working from the main part of the lab, try to create a combinatorial circuit that works as a flexible shifter that has an additional input for how many bits to shift by. This is a more challenging question particularly as we haven’t covered any mechanisms for signals changing over time.

A diagram of a machine

Description automatically generated