ES 4 Lab 2: Binary Addition Using Multiplexers

Prelab due 24 hours before your lab session, Week of Feb 22-25
Lab report due at your lab time, March 1-5

1 Introduction

In this lab, you'll build another useful logic circuit: an "adder" which adds two 3-bit binary numbers and produces a 4-bit result[1]. Unlike the previous lab where we used an assortment of logic gates (AND, OR, XOR, etc.), you'll build this circuit entirely with multiplexers. This can simplify the process of building the circuit, but it also provides a conceptual bridge to the way we'll be using our FPGAs in the future. To oversimplify, an FPGA is a chip which contains thousands of configurable multiplexers linked together, which can be programmed to compute nearly any logic function.

After successfully completing this lab, you should be able to:

- Use multiplexers to implement combinational logic.
- Build something big enough to push the limits of your patience with icestudio.
- (Optional) Use the Digital Discovery's pattern generator and logic analyzer to automate testing of a digital circuit.
- (Optional) Measure timing characteristics of combinational logic.

2 Prelab

In this prelab, you will design the circuit which compute the sum of two binary numbers, $R = M + N$. You can use $M_2 \dots M_0$ to represent the values of the individual bits. M_2 is the most significant bit (i.e., $M = 4M_2 + 2M_1 + M_0$).

$$\begin{array}{cccc} & M_2 & M_1 & M_0 \\ + & N_2 & N_1 & N_0 \\ \hline R_3 & R_2 & R_1 & R_0 \end{array}$$

> **P1**: Write the truth table for the least significant bit of the addition (i.e., write R_0 as a function of M_0 and N_0).

> **P2**: Write the truth table for the carry out of the least significant bit (i.e., find the carry from the one's place to the two's place as a function of M_0 and N_0).

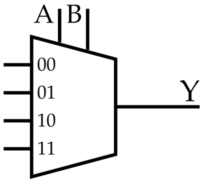
[1] A 16-bit adder would be more useful, but building it by hand would be awfully tedious.

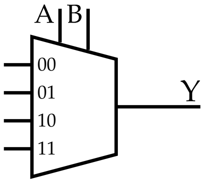
P3: Write the truth table for the second bit of the addition (i.e., write R_1 as a function of M_1, N_1, and C_{out0}, the carry from the least significant bit).

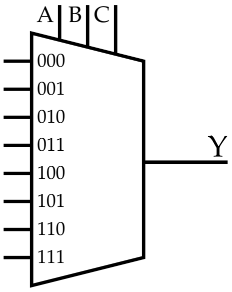
P4: Write the truth table for the carry out of the second bit of the addition (i.e., the carry from the two's place to the four's place as a function of M_1, N_1, and the carry in).

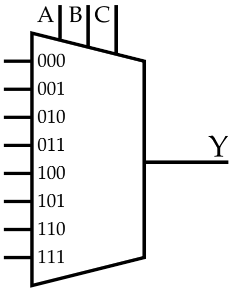
We won't ask you to write down the truth table for the third bit (R_2) and the final carry (which becomes R_3), but you should be able to quickly figure it out from what you've done above.

P5: Draw a logic diagram showing how you will wire up 8-input multiplexers to implement each of the four truth tables in P1-P4.









3 In lab

> **L1**: Start by building the least significant bit of your adder in icestudio, and test it on your FPGA.

If you installed icestudio yourself (rather than using the pre-built VM), you'll need to download and install our custom blocks library from the course website.

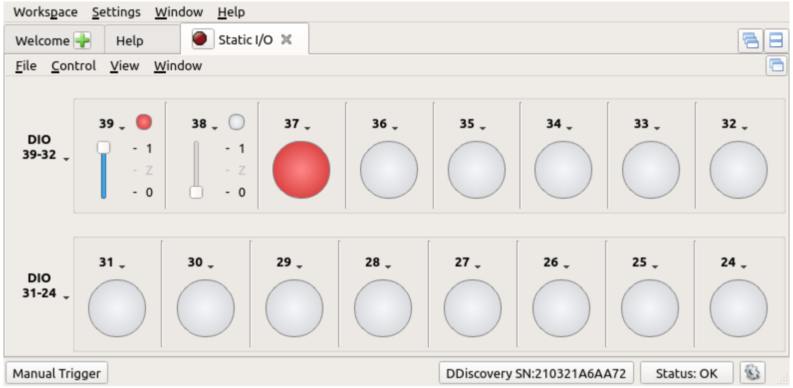
You can use switches and LEDs as the inputs and outputs of your adder as you did for lab 1, and test it by manually toggling the switches and checking the output.

However, if you're in lab or otherwise have access to a Digital Discovery (an Analog Discovery will work too), you can use a more powerful approach: a *pattern generator* and a *logic analyzer*. A pattern generator is the digital equivalent of a waveform generator: it generates patterns of bits as a function of time. A logic analyzer is the digital equivalent of an oscilloscope: it displays the logic level (high or low) as a function of time.

The Digital Discovery is sort of a digital-logic swiss army knife: it combines a pattern generator and logic analyzer, as well as some other tools. We'll start by using its "Static I/O," i.e., outputs that you can control manually and inputs that appear visually on or off.

Use two wires from the Digital Discovery to drive the inputs, and a third to monitor the output. Connect one of the black wires to ground anywhere on your circuit. *Note that only pins 24-39 can be outputs.*

1. Use the USB cable to connect the Digital Discovery to the computer and launch the Waveforms software. Waveforms is installed on the lab computers, and is included with the ES 4 virtual machine image. Search for "Waveforms" or click the icon.
2. Open the "Static I/O" pane from the left side of the start window.
3. Using the drop-down dialogs on each port number, set your two inputs to be push-pull switches, and the output to be an "LED".



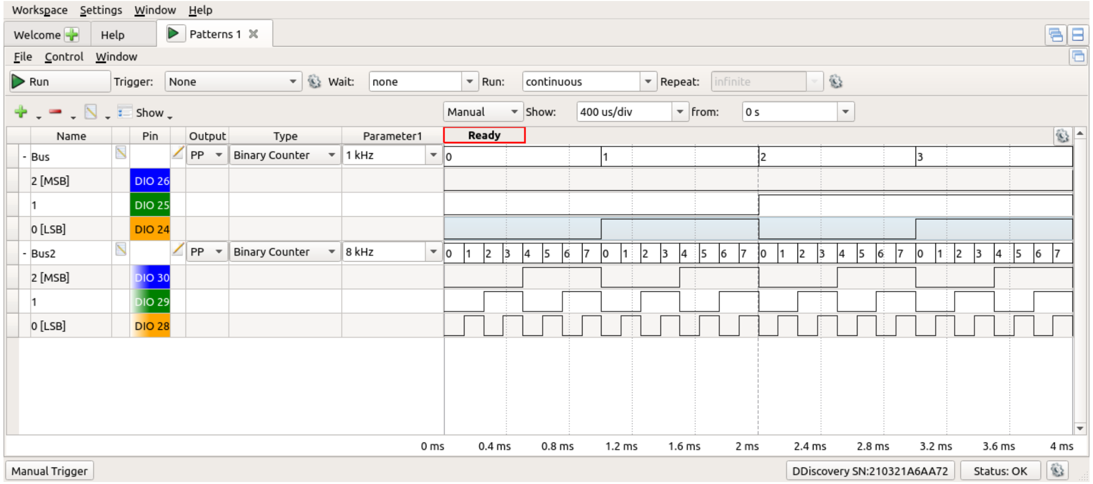
4. Toggle the switches to control the inputs, and visually check the output.

> **L2**: Finish building your adder, testing intermediate pieces as you build.
>
> Ideally we would test the adder with every possible combination of all 6 input bits, which would be $2^6 = 64$ tests. This will get really tedious by hand, so if you're using the Digital Discovery, I suggest you automate this as described below. If you just have LEDs and switches, perform and document enough tests to convince your skeptical professor (or boss) that it's working. Think carefully about what tests you could do to prove that each output and carry bit is working 100% correctly, without testing it exhaustively.

Optional: Automating testing with the Analog/Digital Discovery

You can add a “bus” (a collection of signals), and then drive that bus with a binary counter. As the counter counts up in binary, it steps through every possible combination of the three bits (just like we do when we write a truth table).

If we set up both M and N as 3-bit buses driven by counters, and make one of the counters run 8× faster than the other, then we will end up creating every possible input combination, as shown below:



Trigger: None Wait: none Run: continuous Repeat: infinite

Manual Show: 400 us/div from: 0 s

Name	Pin	Output	Type	Parameter1
- Bus		PP	Binary Counter	1 kHz
2 [MSB]	DIO 26			
1	DIO 25			
0 [LSB]	DIO 24			
- Bus2		PP	Binary Counter	8 kHz
2 [MSB]	DIO 30			
1	DIO 29			
0 [LSB]	DIO 28			

To do this with Waveforms:

1. Close the static I/O tab (since it will conflict with the pattern generator)
2. Open the pattern generator pane and create two buses, one for each input (A, B).
3. Set the type to “binary counter”. Set one frequency to be 8× the other (the dropdown doesn’t have multiples of 8, but you can type a numbers in).
4. Click “Run” to start the waveform generator, and open a new “Logic Analyzer” tab.
5. Add both your inputs and your outputs as buses here, so you can see both the decimal and binary values of the inputs and outputs.
6. Click “single” or “record” to capture the results, and check that the addition is working properly. Grab a screenshot for your lab report!

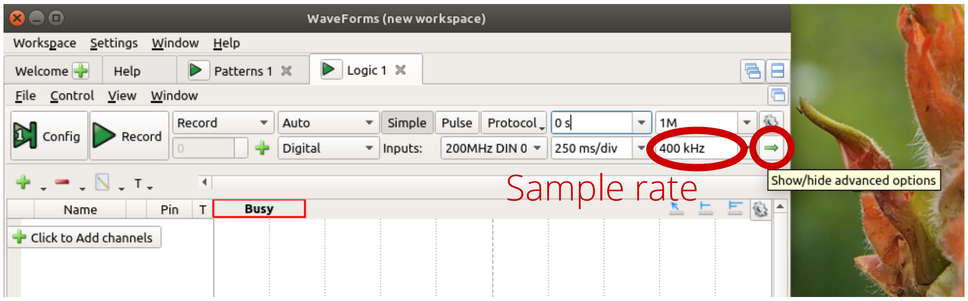
Bonus: This approach still relies on manually inspecting the output to confirm that it is correct. However, the Digital Discovery has a scripting interface which you can use to drive the inputs and read the resulting outputs, so you can automate all of the checks. If you’d like to do this, there is a starter script in the Lab 2 documents folder on the course website. *Make sure to include your completed script in your lab report!*

Optional: Measuring gate delays

How long does it take your adder to reach a stable (and correct) result after the input changes? Document your process for figuring this out, and be sure to capture some screenshots or other supporting data for your lab report. *There are multiple paths, so you’ll need to figure out what changes to the inputs trigger the worst-case propagation delay.*

This happens really fast (less than 1 microsecond), so you will need to be careful that the Digital Discovery is sampling (taking measurements) fast enough. You can control the sample rate if you set it to "Record" mode, and click the little green arrow on the right to expand the options.

The Analog Discovery has a bandwith of only 30 MHz, which isn't fast enough to capture the gate delay through the FPGA. Similarly, our benchtop oscilloscopes have a 100 MHz bandwidth, which is close but may not be reliable.



4 What to turn in

Your lab report should contain:

- Standard "front matter" (see the lab reports handout).
- A photograph of your completed circuit.
- A screenshot of your completed icestudio design.
- Test results of some kind, enough to convice us that your circuit works for all possible inputs.
- Optional: Your plan for determining how fast the adder's output changes when the input changes, your results, and screenshots from Waveforms or pictures of the oscilloscope to support your conclusion.
- A description of problems you encountered, your debugging process for identifying them, and your solutions.
- Answers to the following questions:
  - What was the most valuable thing you learned, and why?
  - What skills or concepts are you still struggling with? What will you do to learn or practice these?
  - How long did it take you to complete the lab? This will help calibrate the workload for future iterations of the course.