cs281: Introduction to Computer Systems

Lab02 – Digital Design (PreLab)

Overview

This prelab is intended to prepare you properly for the Thursday lab. You should complete this prelab ahead of time, so that you are ready to use the 2-hour lab period efficiently. There will not be enough time in the Thursday lab to complete all the activities unless you have completed this prelab in advance.

In this second lab, you will design an important circuit: a multiplexer. You will be required to build the multiplexer in Thursday's lab in hardware in two different ways.

Multiplexer

A multiplexer is a fancy word for ”selector” and we often abbreviate multiplexer to ”mux”. The purpose of the multiplexer is to select a signal on one of the input lines to pass through to the output. The block diagram in Figure 1 shows the general idea.

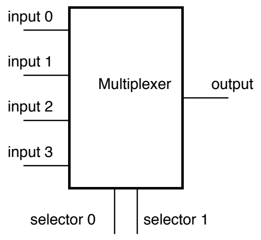

Figure 1: A 4 to 1 Multiplexer

The input lines come in from the left. In this particular multiplexer, there are four input lines labeled `input0` through `input3`. The output of the mux comes out the right side. Two selector lines are inputs in the bottom of the mux. The input lines contain data (signals that are high or low). The idea is to pass the signal from one of these lines through to the output. The selector line(s) determine *which* input line is selected to pass through.

Thus we see in the mux two different kinds of input. The input lines are *data inputs* which have external signals. The selector lines are *control input* lines which control the operation of the mux. This distinction between control and data is a recurring theme in other ICs.

Multiplexer's come in different sizes – different numbers of inputs. Generally the number of input lines is a power of 2. Mux's might have 2 inputs, 4 inputs, 8 and so on. The number of select lines is $\log_2(n)$ where n is the number of inputs. So a mux with 2 inputs, will have 1 select line. A mux with 4 inputs, will have 2 select lines, and so on. You can think of the selector lines as bits of a binary number which indicate (select) which of the inputs get passed through to the output. For example, if the two selector lines are $S_0 = 0$ and $S_1 = 1$, the binary number is `10` (the S_1 line is the higher order bit) and thus input I_2 is selected to pass through to the output.

Consider the two input mux in Figure 2. The inputs are labeled A and B, the selector line S and the output Y. We can build a truth table for the mux. The three inputs are given on the left side of the table, the output on the right.

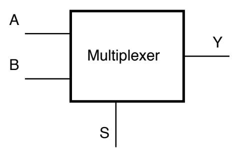

Figure 2: A 2 to 1 Multiplexer

A	B	S	Y
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

These truth tables get unwieldy quickly. A 4 to 1 mux will have 4 data lines and 2 control lines – 6 inputs for a truth table with 64 entries! Generally we write a mux truth table like the following:

S	Y
0	A
1	B

to indicate which input gets passed through by the selector lines. The logic for a mux comes straight from this truth table.

$$Y = A\bar{S} + BS$$

Therefore when S is low (off), it selects line A for the output Y. When S is high, then the output Y copies input B. We can do the same (Truth Table) for a 4 to 1 mux.

S_1	S_0	Y
0	0	A
0	1	B
1	0	C
1	1	D

where the input lines are A through D, and the boolean equation is

$$Y = A\bar{S_1}\bar{S_0} + B\bar{S_1}S_0 + CS_1\bar{S_0} + DS_1S_0$$

We can draw a logic diagram for the 4 to 1 multiplexer as shown in Figure 3.

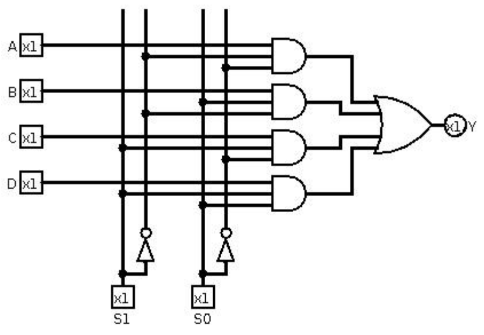

Figure 3: A 4 to 1 Multiplexer

1 Multiplexer Construction

For the Thursday lab, you will build two multiplexer circuits, one will be a 2 to 1 mux, the second will be a 4 to 1 mux. You will build the 2 to 1 Mux out of AND, OR, and NOT gates. Then you will build the 4 to 1 mux with an actual mux chip. You will test this last circuit both with manual switches and with a program driven by your Arduino.

For your research, find the gates for NOT, AND, and OR. Hint, you used two of them for the prior lab. Find the pin-out diagrams for all three chips.

1. Sketch a circuit diagram similar to Figure 3 except for a simpler 2 to 1 mux. Use AND, OR and NOT gates only. You are welcome to use *logisim* on the linux computers to draw the diagram (and test the logic!).

2. Sketch the same circuit diagram using the square IC chips with the pin numbers (show a 7404 inverter, a 7408 AND, and so on). Draw a diagram which shows how to wire up a 2 to 1 mux with two data lines and one select line. Use Switch s_0 for your select line, switch s_5 for input line A and s_6 for input line B. The output will be connected to a the logic probe (led light on the breadboard).

   The point of this sketch is to know what chips you will use and how to connect each numbered pin on each chip to achieve your 2 to 1 mux circuit. This will be your roadmap for connecting your mux when you come to the lab. If you have this all figured out in advance, then the wiring will be simple.

3. Find the IC documentation for a 74150 mux (16 to 1 mux). Look at all the pins on the chip and figure out what they do. Your goal in the Thursday lab is to use this chip to build a 4 to 1 mux. You will use the lowest four data lines only.

   Important: the 74150 does something unusual with the output. Can you see what this is? Look closely at the documentation. You have to figure out the unusual output before you build your circuit or else you will likely think your circuit is incorrect when it may be operating correctly.

   Draw a diagram showing how you will wire up each of the lines on this chip to implement a 4 to 1 mux. Use switches 4, 5, 6, 7 as your data lines. Use switches 0, 1 as your two select lines. Figure out how you need to connect the other lines on the chip. Draw a diagram that will serve as your wiring map on Thursday. Again, show **all** the wire and **all** the pin numbers so you know exactly how to wire this up on Thursday.

2 Adder Circuit

Imagine adding in binary. An example is given below.

```
 1  1
    0  1  1  0   A
  + 1  1  0  0   B
----------------
 1  0  0  1  0   C
```

Each column of the addition functions as a "one bit adder". The adder has a couple of inputs: namely bits A_i + B_i of the two operands. But you might also have to add in a "carry over" bit from the previous column; thus there is a third input. There is the obvious output sum bit, C_i, but keep in mind that your column may also generate a carry out bit that goes into the next column. Thus there are two outputs.

We refer to A and B as the two operands and the sum as C. Bit i of A is designated as A_i (where bits are numbered 0 to $n-1$). Bit 0 is the lowest order bit; bit $n-1$ is the highest order bit (most significant). Let c_{in} be the carry in bit from a previous column. Let c_{out} be the carry out bit which may go through to the next column. We can form truth tables which show the outputs for both C_i and also c_{out}.

Complete the following steps before lab on Thursday:

1. Generate a Truth Table with three inputs (A, B and c_{in}), and with two output columns (C, and c_{out}). Fill in the appropriate values for the truth table.
2. Use your truth table to construct SOP boolean equations for your two outputs.
3. *optional* Do you see ways to reduce the complexity of your boolean expression for either or both of the two equations? You are welcome to keep them in the SOP form, or to use a reduced boolean expression substitute (as long as it implements the correct function).
4. Use Logisim to build a model of your boolean expression. Does it successfully add two binary inputs, without a carry bit? With a carry bit? Bring a picture of your Logisim model to class Thursday.

These are sometimes called "cascade adders" since you can string n of them together to add two n bit binary numbers – the result cascades through the adders one at a time, from low order bits to higher order bits. The carry out of one unit becomes the carry in for the next. The first carry in is set to 0. The last carry out can be used to detect overflow (when the result does not fit in n bits).

Bring with you to class on Thursday all the diagrams requested in this prelab. Show them to either Dr. Kelley or Alistaire BEFORE beginning any wiring. Your diagrams must be correct and approved before you can begin physical construction.