cs281: Introduction to Computer Systems

Lab02 – Digital Design

IMPORTANT

You **MUST** complete the pre-lab prior to starting this lab. There will not be sufficient time in this lab to get everything done unless you have fully designed all the circuits ahead of time. Be sure to get approval of your prelab circuit diagrams from Dr. Kelley or Alistaire before starting construction of any circuits.

Overview

The purpose of this lab is to build on the introductory circuits from last week and design two important circuits this week. You will design and build a multiplexer and also design and build a 1-bit adder circuit. There are four parts to the lab.

1. Build a 1 bit adder circuit. Test with switches.
2. Build a 2 to 1 mux out of AND, OR and NOT gates. Test with switches.
3. Build a 4 to 1 mux using a 74150 mux chip. Test with switches.
4. Build a 4 to 1 mux using a 74150 mux chip. Test with Arduino program.

You are to write a lab report that is to be submitted by the Monday (in class) following the lab.

1 Part 1: 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}.

You should have completed a Logisim model of a 1-bit adder before coming to lab. Now you will build this circuit on a breadboard. Demonstrate this working to Dr. Kelley or Alistaire. Include a picture of both in your lab report.

2 A 2 to 1 Mux

In this first part of the experiment, you are to build a simple 2 to 1 mux using AND, OR and NOT gates. This is a circuit you designed in the prelab. You should have BOTH a circuit diagram (with shaped gates) and also a wiring diagram (with rectangles for ICs and pin numbers showing connections). Use the latter diagram to build your circuit.

Use the appropriate switches for the select line and two data lines. Demonstrate the proper operation of your circuit to Dr. Kelley or Alistaire.

You will deconstruct your circuit from Part 1 since it is not at all useful for the second part. Please return any AND/OR/NOT gates not found in your hardware box to their appropriate drawers.

3 A 4 to 1 Mux

In this experiment, you will obtain a 74150 chip and place it on your breadboard. Again, you should have a COMPLETE wiring diagram for this IC. Follow your wiring diagram, double checking all the connections as you go. There are a lot of pins on this chip, so you have to be careful in counting and attaching wires; it is easy to put a wire in the wrong place.

Use the appropriate switches for your two select lines. Use the other indicated switches for your four data lines. Be sure the other lines on the 74150 are wired as needed.

Demonstrate the proper operation of this circuit for Dr. Kelley or Hayley.

When you are done, remove only the wires to the switches. Keep the 74150 on the board and the other connections you had for the chip.

4 A 4 to 1 Mux with Arduino

In this section, we will use the Arduino, not manual switches, to test the operation of the mux. The Arduino will drive the inputs (data and control) for the mux, read the output of the mux, and run through a series of tests to verify the correct operation of the mux (and your circuit).

To save time, you might have one person start writing the program in this section and the other person start the wiring. Then you can each double check the other person's work.

Use the smaller Arduino wires for connecting to the Arduino so that you do not damage the pin sockets on the Arduino.

For consistency with the program below, I recommend using the following digital pins on the Arduino for the following purposes:

Pin 10 Data input A (E0 on the mux)

Pin 11 Data input B (E1 on the mux)

Pin 12 Data input C (E2 on the mux)

Pin 13 Data input D (E3 on the mux)

Pin 8 Select Line 0 (A on the mux)

Pin 9 Select Line 1 (B on the mux)

Pin 7 output data from mux (w on the mux)

GND Connect the GND (adjacent to pin 13) to ground on the breadboard

From the *Arduino's* perspective, pins 8-13 are outputs. They are data and control signals coming out of the Arduino which will serve as inputs to the mux. Pin 7 is an input for the Arduino; it is reading the output data from the mux. Be sure to understand which pins are input and output from the Arduino's perspective.

You will connect the GND pin (on the data side, not on the power side) to ground on the breadboard. This is important so that both the Arduino and the breadboard share a common sense of "ground". You will not power up the Arduino using the +5/GND pins on the power side since the Arduino will stay connected to the laptop during the whole experiment (it receives power through the USB cable).

Write the following program for your Arduino.

```
//================================
// Lab 2
// Your name(s)
// Date
//================================

const int S0[] = {0,0,1,1,0,0,1,1};
const int S1[] = {0,0,0,0,1,1,1,1};

const int A[] = {0,1,0,0,0,0,0,0};
const int B[] = {0,0,0,1,0,0,0,0};
const int C[] = {0,0,0,0,0,1,0,0};
const int D[] = {0,0,0,0,0,0,0,1};

const int Y[] = {0,1,0,1,0,1,0,1};

const int WAIT0 = 300;
const int WAIT1 = 2000;

int index = 0;
```

```
int x;  // for reading input

void setup() {
  // Serial Port setup for communication back to computer
  Serial.begin(9600);

  // data pins are outputs (for Arduino)
  pinMode(10,OUTPUT); // A
  pinMode(11,OUTPUT); // B
  pinMode(12,OUTPUT); // C
  pinMode(13,OUTPUT); // D

  // select pins are outputs (for Arduino)
  pinMode(8,OUTPUT);  // S0
  pinMode(9,OUTPUT);  // S1

  // Mux output is input for Arduino
  pinMode(7,INPUT);

}

void loop() {
  // write data inputs to MUX
  digitalWrite(10,A[index]);
  digitalWrite(11,B[index]);
  digitalWrite(12,C[index]);
  digitalWrite(13,D[index]);

  // write select line inputs to MUX
  digitalWrite(8,S0[index]);
  digitalWrite(9,S1[index]);

  delay(WAIT0);   // give time for logic signal to propagate

  // read the MUX output
  x = digitalRead(7);

  // display the results
  Serial.print(index);
  Serial.print(" x:");
  Serial.print(x,BIN);
  Serial.print(", y:");
  Serial.print(Y[index],BIN);
  Serial.print("\t ");
  if ( x != Y[index] )
  {
```

```
    Serial.print(": OK\n");
  }
  else
  {
    Serial.print(": BAD\n");
  }
  delay(WAIT1);
  index = (index+1) % 8;  // increment index
}
```

Run this program. You will have to open the ”Serial Monitor” window in the Arduino IDE. This is the magnifying-glass like icon in the upper right corner of the IDE. You should see the results of your tests scroll across this window.