Lab 3 - Combinational Logic Circuits (III)

CDA 3201L-003
Jason Keene and Jacob Manfre
Submitted Feb 8th, 2015

Purpose and Objectives

This lab we set out to learn how to design circuits with multiplexers and comparators. We were given an expression to implement with both an 8-to-1 multiplexer and 4-to-1 multiplexers. Additionally, we were to implement a 3-bit comparator.

Component list

- Breadboard
- Wiring
- 5v power supply
- 4 x LED
- 4 x Resistor (470 Ohms 5%)
- 1 x 8-input Multiplexer (74LS151)
- 2 x Dual 4 to 1 Selector/Multiplexer (74LS153)
- 1 x Hex Inverter (74LS04)
- 1 x Quad 2-input AND Gate (74LS08)
- 2 x Triple 3-input AND Gate (74LS11)
- 1 x Quad 2-input NOR Gate (74LS02)
- 1 x Quad 2-input OR Gate (74LS32)

Design

For the 8-to-1 mux we simply pulled up the datasheet and learned how that chip worked. Implementing the provided expression simply required the inputs to be set to the select lines and have the desired output for each input set on that data line.

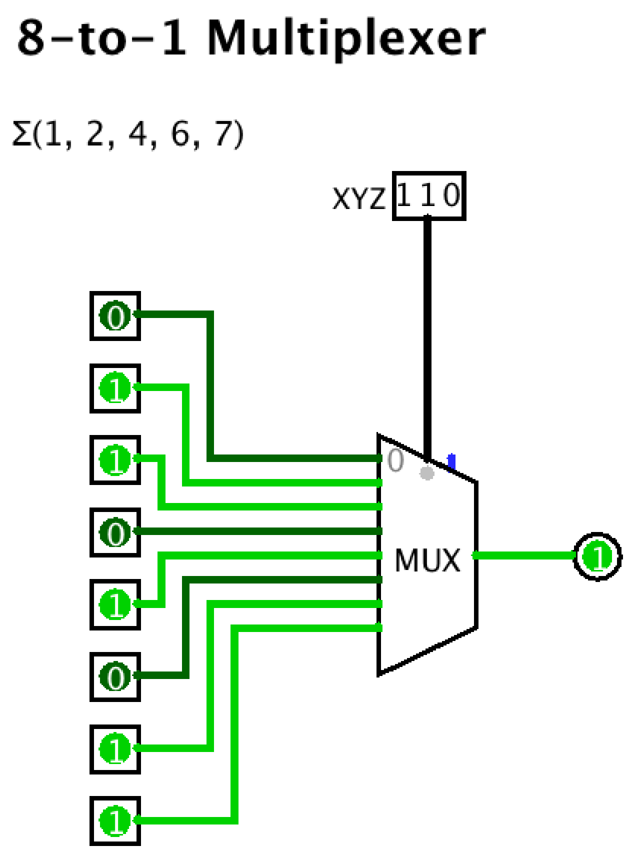
We expanded the sigma notation:

```
F(x, y, z) = Σ(1, 2, 4, 6, 7)
           = 001 + 010 + 100 + 110 + 111
           = x'y'z + x'yz' + xy'z' + xyz' + xyz
```

We then used this to generate the truth table:

x	y	z	F(x, y, z)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

We then implemented this in logisim and in hardware:

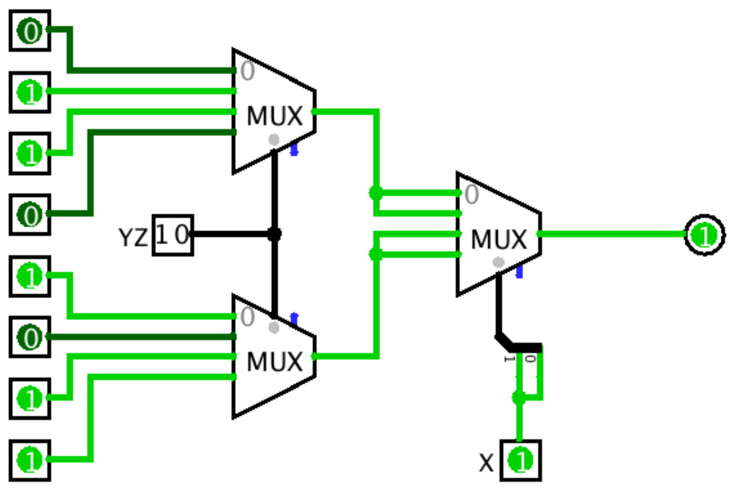


For the 4-to-1 mux we decided to use two muxs to select half of each of the data lines then have a single mux to select between each of those halves.

We implemented this in logisim and in hardware:

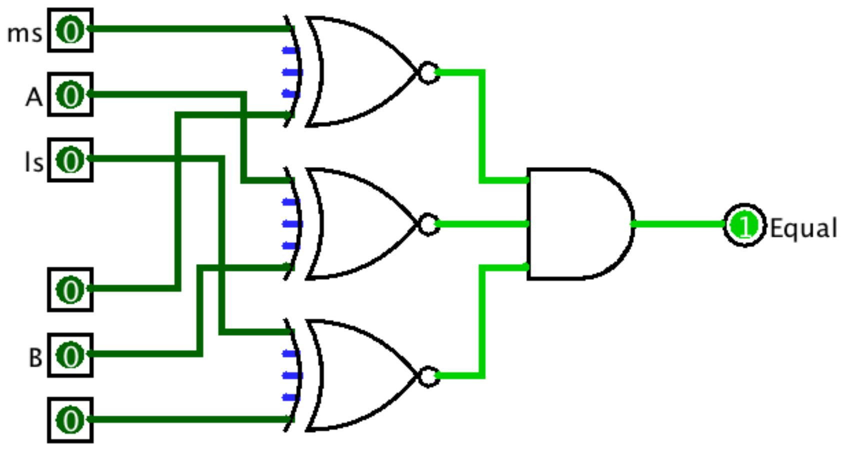
4-to-1 Multiplexer

$\Sigma(1, 2, 4, 6, 7)$



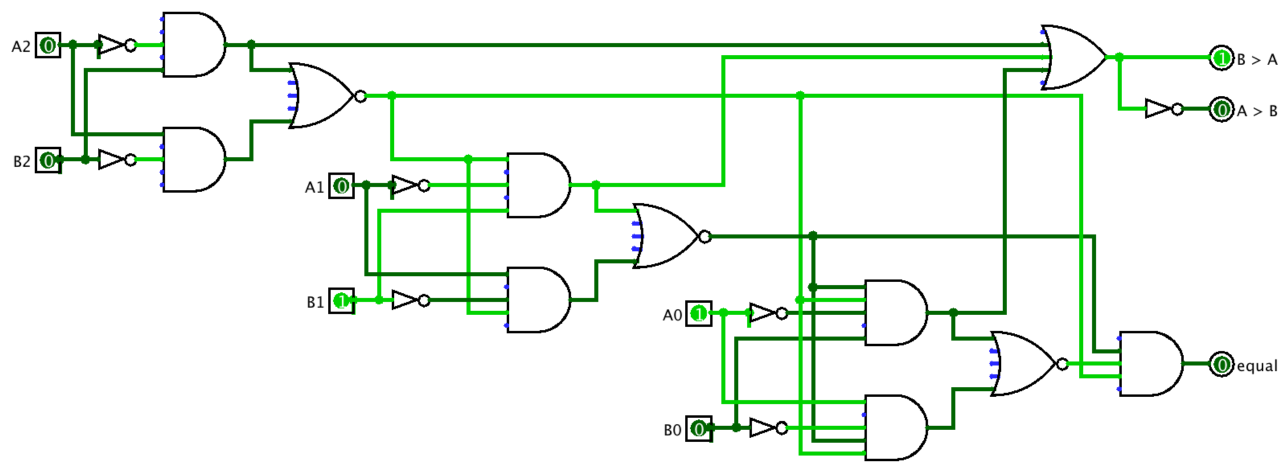
The final part of this lab gave us great difficulty. We initially took a stab at implementing a 3 bit equality checker using XNORs between each bit and ANDing them together. However we could not discover how to do the comparison using this design.

3 bit Equality



We decided to take the brute force approach and created a truth table with all 64 combinations of inputs and outputs. We then created a 3x3 k-map to get the sum of products form for this circuit. We then simplified this down to an expression that would require 22 gates. We knew this could be simplified down more but ran out of ideas for doing this using boolean algebra so we went back to the drawing board and came up with our final design. We started with defining what makes one bit greater than the other, that it is high AND the other bit is NOT high. Using this we created a 1-bit comparator. We replicated this by feeding the equality result of the previous comparison into the AND of the next comparison.

3 bit Comparator



Once we had this design we implemented it in hardware.

Test vectors and verification

The 8-to-1 mux was fairly easy to test. We tested every combination of inputs against the truth table:

x	y	z	F(x, y, z)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

We did the same for our design using the 4-to-1 mux. This proved that both circuits functioned identically.

The 3-bit comparator was a bit more difficult to test. We created a truth table with all 64 combinations of inputs:

a2	a1	a0	b2	b1	b0	eq	out
0	0	0	0	0	0	1	X
0	0	0	0	0	1	0	1
0	0	0	0	1	0	0	1
0	0	0	0	1	1	0	1
0	0	0	1	0	0	0	1
0	0	0	1	0	1	0	1
0	0	0	1	1	0	0	1
0	0	0	1	1	1	0	1
0	0	1	0	0	0	0	0
0	0	1	0	0	1	1	X
0	0	1	0	1	0	0	1
0	0	1	0	1	1	0	1
0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	1
0	0	1	1	1	0	0	1
0	0	1	1	1	1	0	1
0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0
0	1	0	0	1	0	1	X
0	1	0	0	1	1	0	1
0	1	0	1	0	0	0	1
0	1	0	1	0	1	0	1
0	1	0	1	1	0	0	1
0	1	0	1	1	1	0	1

a2	a1	a0	b2	b1	b0	eq	out
0	1	1	0	0	0	0	0
0	1	1	0	0	1	0	0
0	1	1	0	1	0	0	0
0	1	1	0	1	1	1	X
0	1	1	1	0	0	0	1
0	1	1	1	0	1	0	1
0	1	1	1	1	0	0	1
0	1	1	1	1	1	0	1
1	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	0	1	1	0	0
1	0	0	1	0	0	1	X
1	0	0	1	0	1	0	1
1	0	0	1	1	0	0	1
1	0	0	1	1	1	0	1
1	0	1	0	0	0	0	0
1	0	1	0	0	1	0	0
1	0	1	0	1	0	0	0
1	0	1	0	1	1	0	0
1	0	1	1	0	0	0	0
1	0	1	1	0	1	1	X
1	0	1	1	1	0	0	1
1	0	1	1	1	1	0	1
1	1	0	0	0	0	0	0

a2	a1	a0	b2	b1	b0	eq	out
1	1	0	0	0	1	0	0
1	1	0	0	1	0	0	0
1	1	0	0	1	1	0	0
1	1	0	1	0	0	0	0
1	1	0	1	0	1	0	0
1	1	0	1	1	0	1	X
1	1	0	1	1	1	0	1
1	1	1	0	0	0	0	0
1	1	1	0	0	1	0	0
1	1	1	0	1	0	0	0
1	1	1	0	1	1	0	0
1	1	1	1	0	0	0	0
1	1	1	1	0	1	0	0
1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	X

We tested our logisim and hardware implementations against certain inputs as we went along. Towards the end we tested every input against this table. We discovered a bug in our implementation where we were not feeding the first equality check into the last comparator. We chose to use the left over two input AND gates vs rewiring the entire circuit to accommodate a four input AND.

Discussion and conclusion

During this lab we demonstrated how to use a multiplexer to select different sets of data lines. We imagine that this will come in useful in sequential circuits when we introduce a clock and the concept of time as we could select different values at different times.

Additionally, we demonstrated how to compare multiple bit values to see if they are equal, greater than, or less than. Along the way we discovered the futility of using brute force methods such as truth tables and k-maps to express, simplify, and verify circuits with large bit inputs. We probably would have done better by designing and verifying a 1-bit comparator and wiring three of them up and verifying that wiring vs the approach we took.