Victor Yuan

October 13, 2016

Lab 7

Introduction:

This lab was a continuation of Lab 6 but instead of looking of looking just at demultiplexer, this Lab examined how a multiplexer, and how its output leads into a demultiplexer.

Team Member Responsibilities:

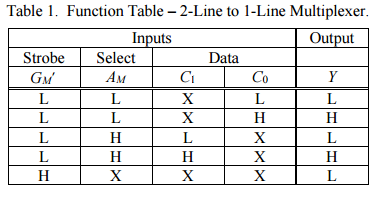
This lab was done alone.

Materials:

ELENCO Analog kit, CMOS Logic gates, assorted wires, Verilog simulation program through Putty.

Procedure:

First convert the following table into a Karnaugh map.



The map should look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Gm’ Am  C1,C0 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 0 | 0 | 0 |
| 11 | 1 | 1 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 |

From the map the Boolean expression (Gm’)’ Am’ C0 + (Gm’)’ Am C1 is derived, converting this AND-OR network to a NAND-NAND network will make the hardware implementation portion of this lab easier. But for now convert the 3 input and gate into two input only.

Simulate the expression in Verilog. Your code should look like this:

module MAIN;

reg a,b,c,d;

wire x;

not U1b(e, a);

not U2b(f, b);

and U3b(h,b,d);

and U4b(g,f,c);

and U5b(i,e,g);

and U6b(j,e,h);

or U7b(x,i,j);

initial

begin

$monitor($time, "- a=%b b=%b c=%b d=%b e=%b f=%b g=%b h=%b i=%b j=%b x=%b",a,b,c,d,e,f,g,h,i,j,x);

#00 a=0; b=0; c=0; d=0;

#30 a=0; b=0; c=0; d=1;

#30 a=0; b=0; c=1; d=0;

#30 a=0; b=0; c=1; d=1;

#30 a=0; b=1; c=0; d=0;

#30 a=0; b=1; c=0; d=1;

#30 a=0; b=1; c=1; d=0;

#30 a=0; b=1; c=1; d=1;

#30 a=1; b=0; c=0; d=0;

#30 a=1; b=0; c=0; d=1;

#30 a=1; b=0; c=1; d=0;

#30 a=1; b=0; c=1; d=1;

#30 a=1; b=1; c=0; d=0;

#30 a=1; b=1; c=0; d=1;

#30 a=1; b=1; c=1; d=0;

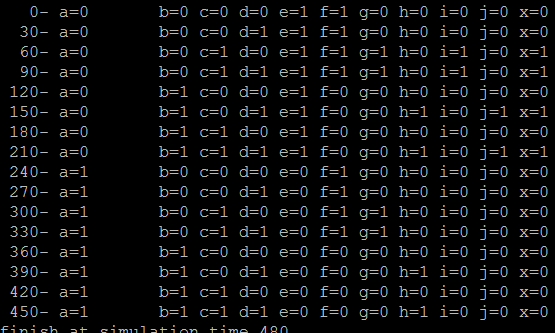
#30 a=1; b=1; c=1; d=1;

#30 $finish;

end

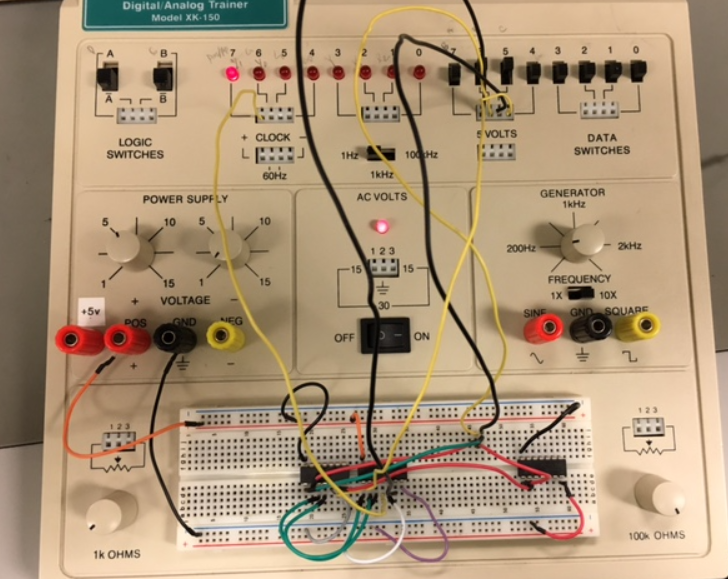
endmodule

The display result should look like this:



X is the output and a,b,c,d stand for Gm’ , Am, C0, C1.

Now implement this into hardware using the ELENCO kit and whichever chips you need according to your design.



Afterwards connect the output of your code to the input of code from lab 6.

module MAIN;

reg a,b,c,d,A,B;

wire x;

wire Y;

wire Z;

//lab 7

not U1b(e, a);

not U2b(f, b);

and U3b(h,b,d);

and U4b(g,f,c);

and U5b(i,e,g);

and U6b(j,e,h);

or U7b(x,i,j);

//lab 6

not U1a(D,A);

not U2a(E,B);

and U3a(F,B,x);

and U4a(G,E,x);

and U5a(Y,D,G);

and U6a(Z,D,F);

initial

begin

$monitor($time, "- A=%b B=%b a=%b b=%b c=%b d=%b x=%b Y=%b Z=%b" ,A,B,a,b,c,d,x,Y,Z);

#00 A=0;B=1;a=0;b=0;c=0;d=0;

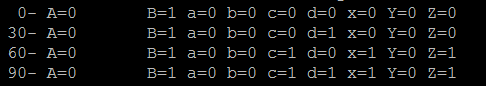
#30 A=0;B=1;a=0;b=0;c=0;d=1;

#30 A=0;B=1;a=0;b=0;c=1;d=0;

#30 A=0;B=1;a=0;b=0;c=1;d=1;

#30 $finish;

And the output should look like:



X is the out put from lab 7 that goes into lab 6 Y and Z is the output from lab 6

The last step is to implement this new combination into hardware.

Questions:

I. Experiment:

1. How do the Verilog simulation truth-table and hardware implementation truth table for the 2:1 multiplexer compare?

They are the same therefore the simulation was implemented correctly.

2. Does your design correctly implement the function table for a 2:1 multiplexer given in Table 1?

Yes my design was correctly implemented into the function table.

II. System Integration:

1. How do the Verilog simulation and hardware implementation for the overall simple digital communication system shown in Figure 1 compare?

For the values shown the matched as for the other possibilities that are not shown, they should match as well.

1. How did you determine the test vectors used to test your overall design?

The four test vectors was copied from the board but they were selected for the purpose of showing where the output would change.

Conclusions:

1. Discuss how you could expand your design to include more information sources and more destination sinks.

More complex multiplexer and demultiplexer combinations could always be added by simply adding more logic gates into the design combining output of one multiplexer to input of multiple multiplexers. Connecting the output of multiple demultiplexer into one demultiplexer.