Christopher Badolato

Experiment 4 Multiplexer

EEE 3342 LAB 0014

10/14/2018

**Objective:**

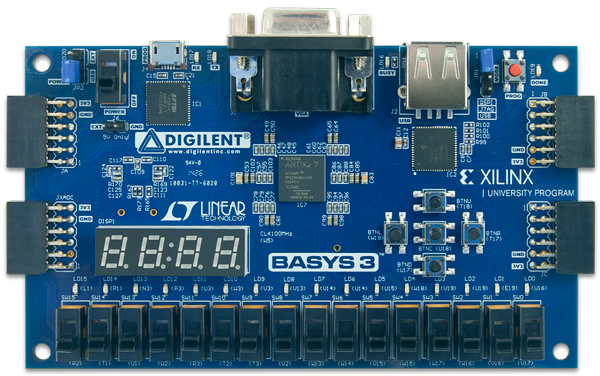
* We will design a 16 to 1 multiplexer using two 8 to 1 multiplexers and an OR gate.
* Along with the implementation of combinational logic design with procedural programming.
* minterm equation: F(w,x,y,z) = w’x’y’z’ + wxy’ + wxz + wx’y

Apparatus List:

Xilinix’s FPGA VIVADO HLX edition

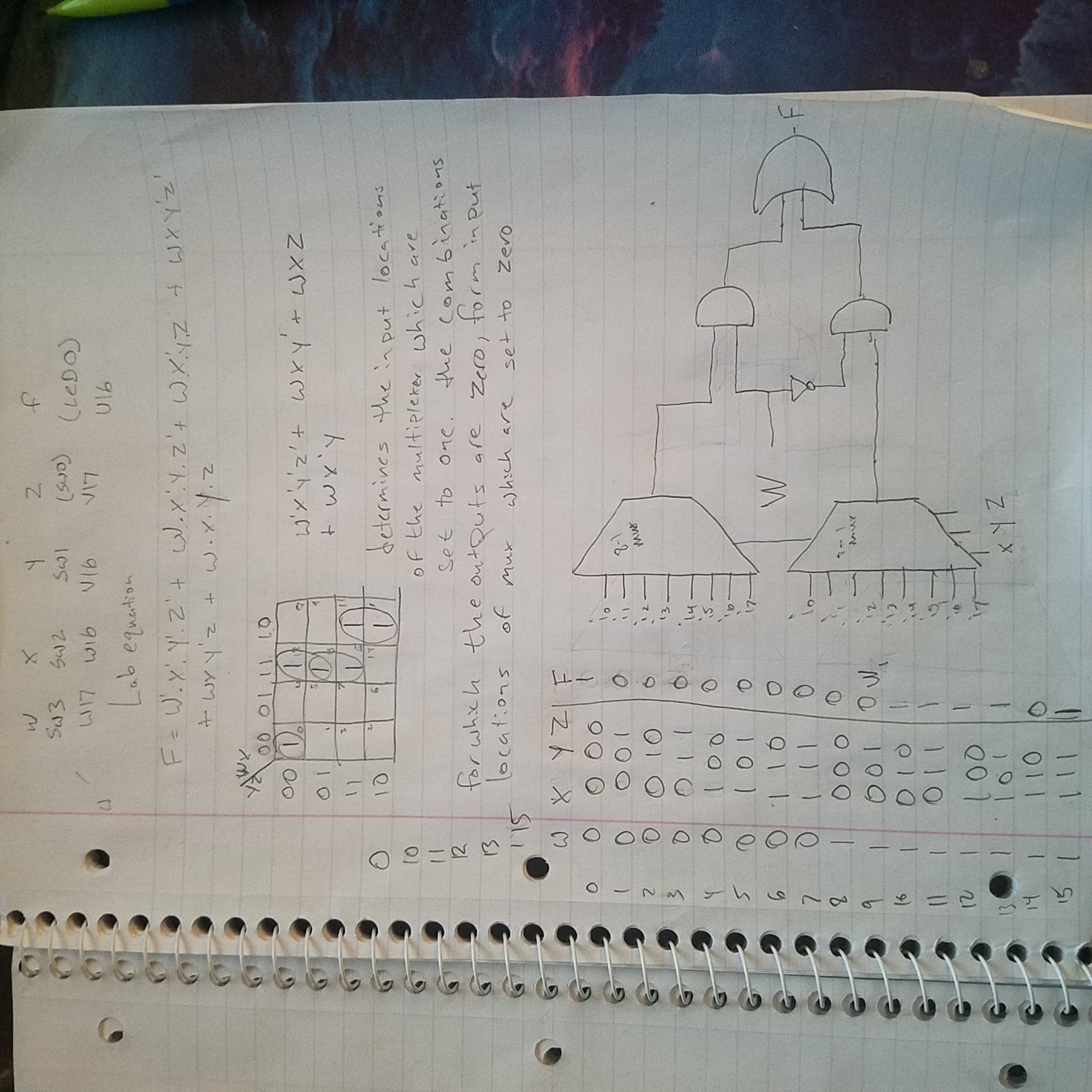
VIVDAO 2017.4

Basby Atrix-7



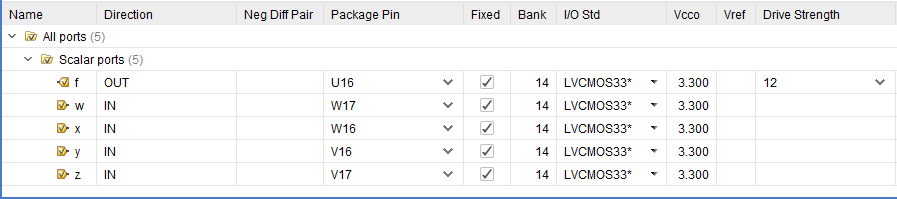
Procedure and/or Design Methodology:

Block Diagram:



Design Specification Plan:



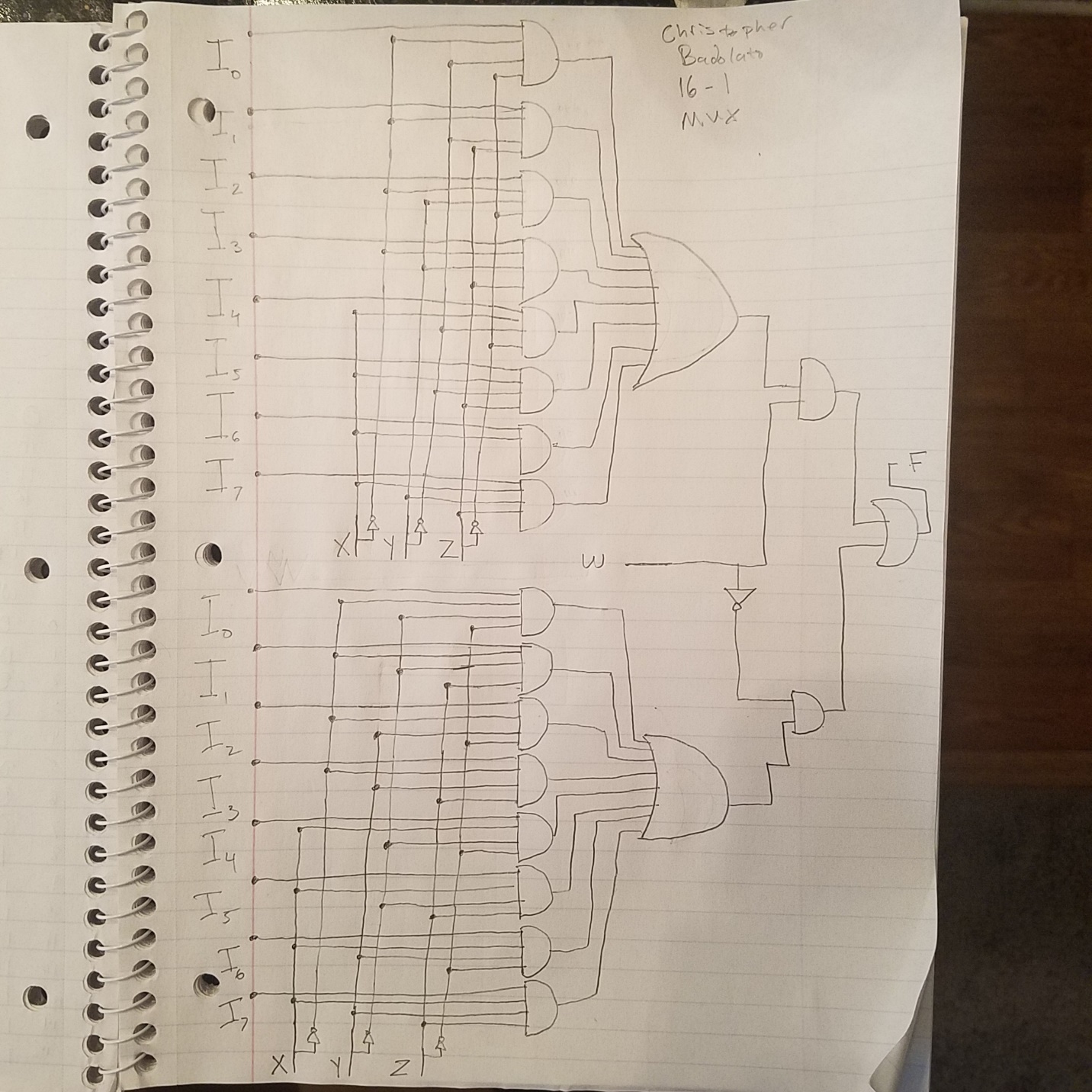


F(w,x,y,z) = w’x’y’z’ + wxy’ + wxz + wx’y

Multiplexer.v test.v

|  |  |
| --- | --- |
| `timescale 1ns / 1ps  module Multiplexer(f, w, x, y, z);  input w,x,y,z;  output f;  reg f;    always@(w or y or x or z)  begin  if(w == 0 & x == 0 & y == 0 & z ==0)  f = 1; //0  if(w == 0 & x == 0 & y == 0 & z ==1)  f = 0; //1  if(w == 0 & x == 0 & y == 1 & z ==0)  f = 0; //2  if(w == 0 & x == 0 & y == 1 & z ==1)  f = 0; //3  if(w == 0 & x == 1 & y == 0 & z ==0)  f = 0; //4  if(w == 0 & x == 1 & y == 0 & z ==1)  f = 0; //5  if(w == 0 & x == 1 & y == 1 & z ==0)  f = 0; //6  else if(w == 0 & x == 1 & y == 1 & z ==1)  f = 0; //7  else if(w == 1 & x == 0 & y == 0 & z ==0)  f = 0; //8  else if(w == 1 & x == 0 & y == 0 & z ==1)  f = 0; //9  else if(w == 1 & x == 0 & y == 1 & z ==0)  f = 1;//10  else if(w == 1 & x == 0 & y == 1 & z ==1)  f = 1;//11  else if(w == 1 & x == 1 & y == 0 & z ==0)  f = 1;//12  else if(w == 1 & x == 1 & y == 0 & z ==1)  f = 1;//13  else if(w == 1 & x == 1 & y == 1 & z ==0)  f = 0; //14  else if(w == 1 & x == 1 & y == 1 & z == 1)  f = 1;//15  end  endmodule | `timescale 1ns / 1ps  module test(  );  reg w\_Sim\_t;  reg x\_Sim\_t;  reg y\_Sim\_t;  reg z\_Sim\_t;  wire f\_Sim\_t;  Multiplexer UTT (  .w(w\_Sim\_t),  .x(x\_Sim\_t),  .y(y\_Sim\_t),  .z(z\_Sim\_t),  .f(f\_Sim\_t)  );  initial begin  w\_Sim\_t =1'b0;  x\_Sim\_t =1'b0;  y\_Sim\_t =1'b0;  z\_Sim\_t =1'b0;  end  always #10 w\_Sim\_t = ~w\_Sim\_t;  always #20 x\_Sim\_t = ~x\_Sim\_t;  always #40 y\_Sim\_t = ~y\_Sim\_t;  always #80 z\_Sim\_t = ~z\_Sim\_t;  endmodule |

Detailed Schematic Diagram:



Test Plan:

To test we will run the code with our Verilog software implementing the ports shown in the picture above. We need to write our bitstream to the board. After the board is programmed we will go through each value (0 through 15) in binary with our switches corresponding to the variable in our function F(w,x,y,z). We can simply follow along with the truth table shown in the procedure. The LED representing F will only be turned on at the minterms of our equation. In this case F(w,x,y,z) = ∑m(0,10,11,12,13,15).

Results:

The results of this experiment is a 16-multiplexer created from two 8 to 1 multiplexers and an OR gate. F(w,x,y,z) will be turned on at our minterms at

F(w,x,y,z) = ∑m(0,10,11,12,13,15)

Conclusion

We will create a 16 to 1 multiplexer by combining two 8 to 1 multiplexers. We will AND each of the outputs from the 8 to 1 multiplexer with our enabler in the case our variable w. x y and z will represent the I output of our multiplexer which is a three-bit binary number (0 through 7). Once we AND our output choice from the multiplexers to w, we will OR it with the opposite value of w, giving us our value of F(w,x,y,z), which will only be turned on at our minterms of F(w,x,y,z).

Questions:

(To be incorporated in the Conclusion section of your laboratory report.)

1. Investigate the function of a lookup table and describe how one works.

The function of a look-up table is to find values within a large set by searching through the list and comparing values to the one you are looking for.

1. Consider a 16 word by 1 bit lookup table. Give the values stored in each location 0000 binary (word zero) to 1111 binary (word fifteen) for the function F(w, x, y, z). The truth table that was generated in the pre-laboratory will help here.

