Christopher Badolato

Experiment 5 Decoder and Demultiplexer

EEE 3342 LAB 0014

10/28/2018

**Objective:**

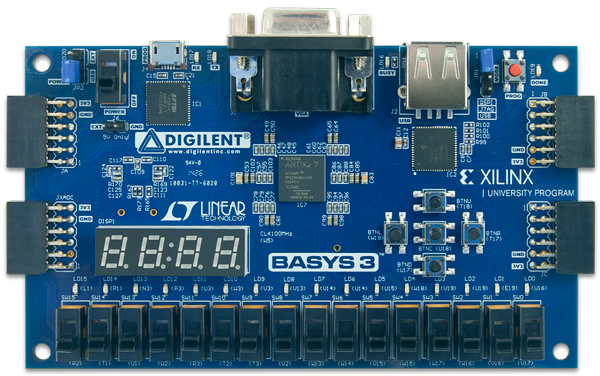
* We will design a 3 to 8 decoder.
* We will design the decoder with two, 2to4 decoders with an enable signal to choose which decoder we will use.

Apparatus List:

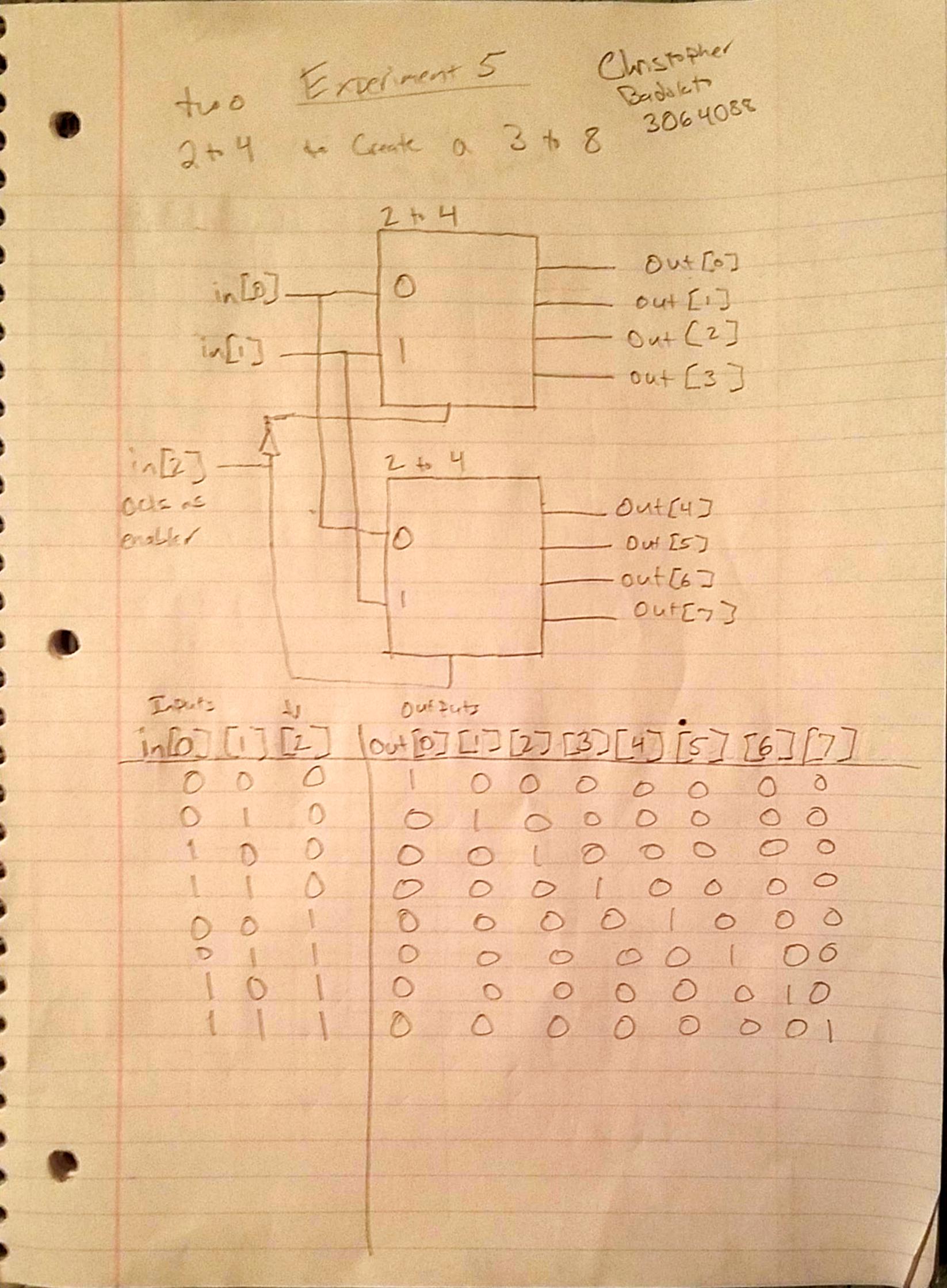
Xilinix’s FPGA VIVADO HLX edition

VIVDAO 2017.4

Basby Atrix-7



Procedure and/or Design Methodology:

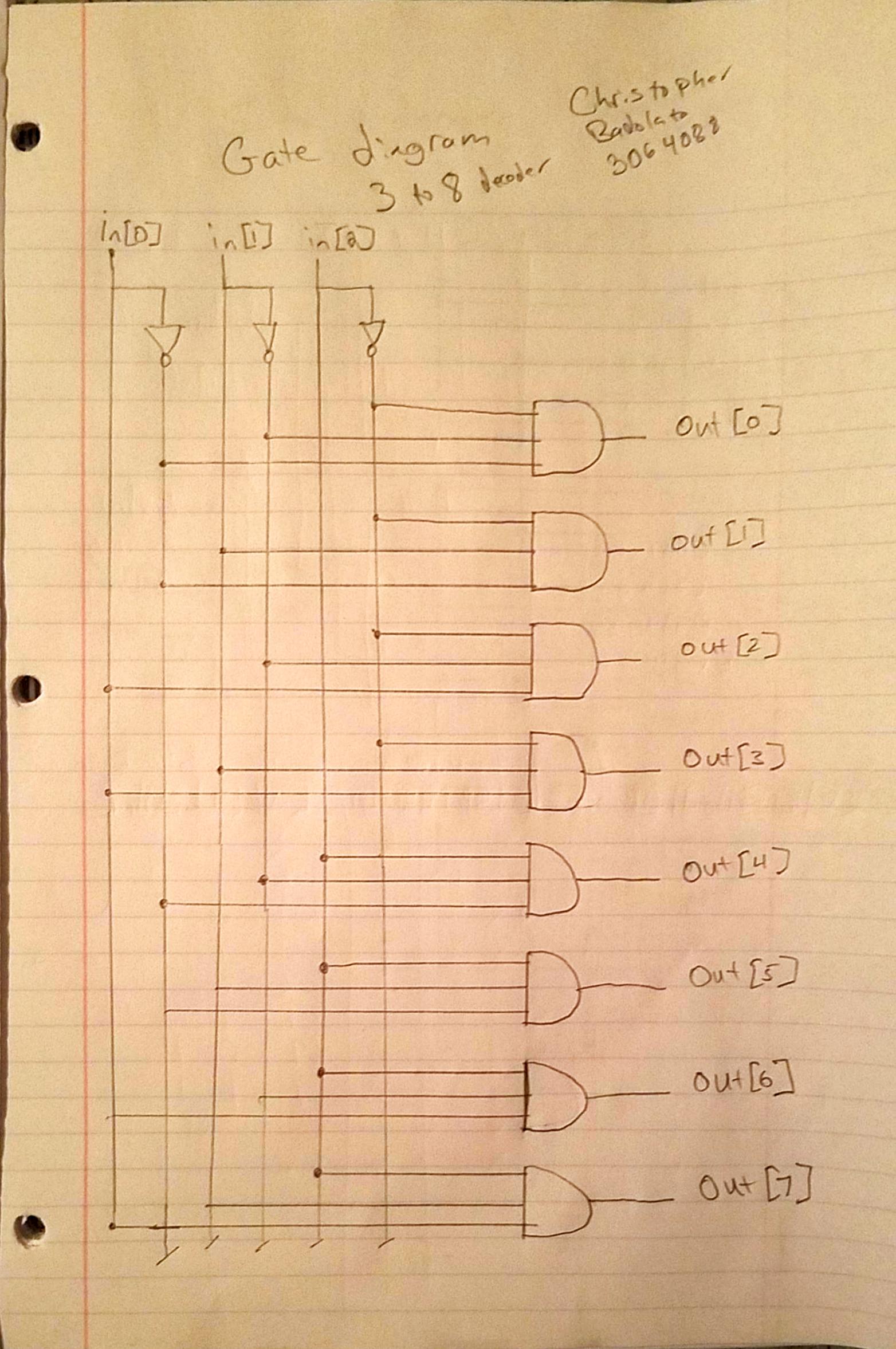
Block Diagram:

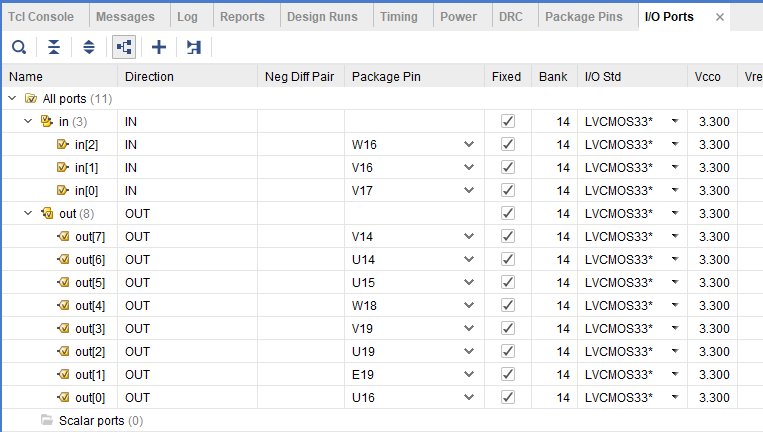
Design Specification Plan:

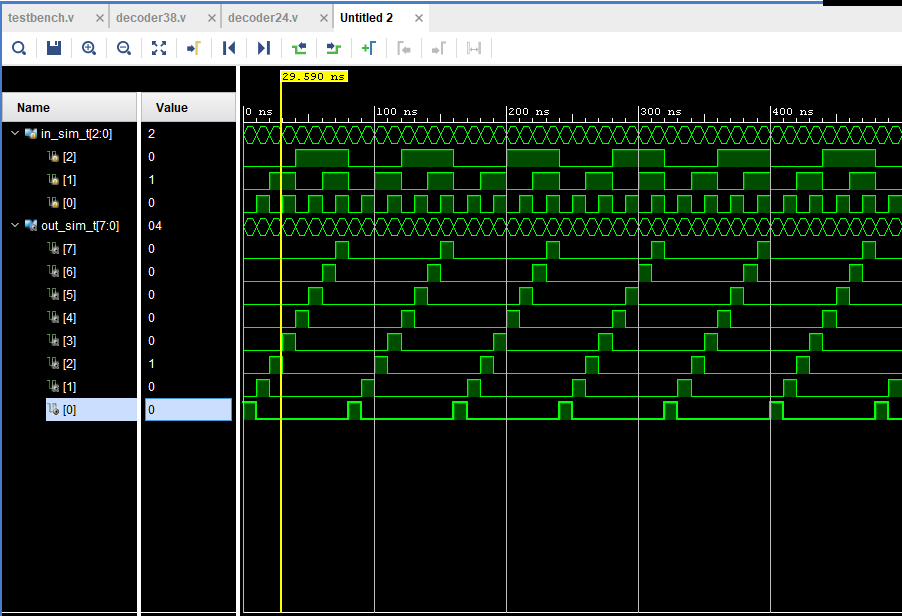
decoder24.v decoder38.v

|  |  |
| --- | --- |
| `timescale 1ns / 1ps  module decoder24(en, in, out);  input en;  input [1:0] in;  output reg [3:0] out;    always@(en or out or in)  begin  if(en == 1)  case(in)  2'b00: out = 4'b0001;  2'b01: out = 4'b0010;  2'b10: out = 4'b0100;  2'b11: out = 4'b1000;  endcase  else if(en == 0)  begin  out = 4'b0000;  end  end  endmodule | `timescale 1ns / 1ps  module decoder38(in,out);  input[2:0] in;  output[7:0] out;  wire wire1;  decoder24 decoder1(in[2], in[1:0], out[7:4]);  decoder24 decoder0(wire1, in[1:0], out[3:0]);  not inverter0(wire1,in[2]);  endmodule |
| Testbench.v |  |
| `timescale 1ns / 1ps  module testbench();  reg [3:0]in\_sim\_t;    wire [7:0]out\_sim\_t;    decoder38 UUT(  .out(out\_sim\_t),  .in(in\_sim\_t)  );    initial begin  in\_sim\_t[0] = 1'b0;  in\_sim\_t[1] = 1'b0;  in\_sim\_t[2] = 1'b0;  end    always#10 in\_sim\_t[0] = ~in\_sim\_t[0];  always#20 in\_sim\_t[1] = ~in\_sim\_t[1];  always#40 in\_sim\_t[2] = ~in\_sim\_t[2];    endmodule |  |

Detailed Schematic Diagram:







Test Plan:

To test, we must first load our three files to vivado. Our first file, decoder24.v, is two 2 to 4 decoders each with an enable signal line (our third input). The file, decoder38.v calls our decoder24.v creating a 3 to 8 decoder.

So, Our value in[0] is our least significant bit, In[2] is our enable signal choosing which decoder to select. based on our value of in[0] and in[1] we will choose within the 2 to 4 decoder our output.

If we follow our truth table with our switches. V17 being in[0], V16 being in[1], W16 being in[2] we can follow our truth table from digits 0 to 3 in binary while turning on our in[0] and in[1] and leaving our in[2] off, we can go through our first set of 4 decoded values. Then we can enable our in[2], going through 0 to 3 in binary with our in[0] and in[1] inputs do get our final values 4 to 7.

Results:

The results will be a 3 to 8 decoder, created from two 2 to 4 decoders.

Conclusion

Our binary values will match the index of our output array.

Questions:

1. If the decoder has active low outputs (74155) instead of active high outputs as the case for D3\_8E 3-to-8 decoder, why can a NAND gate be used to logically OR its outputs?

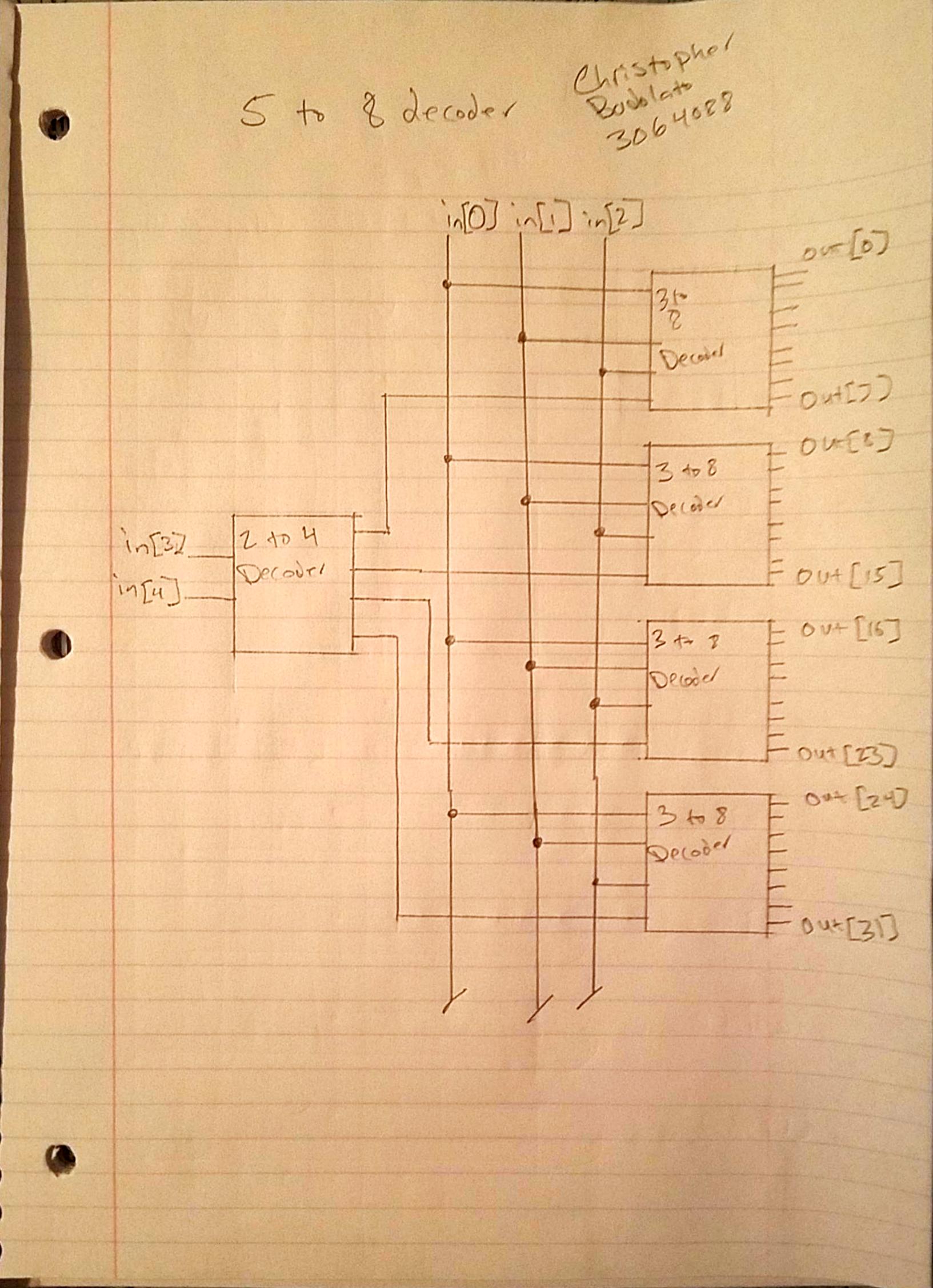
NAND gates can be used because of deMorgans law. providing active high inputs is the only case where we can active low outputs. Thus we can use NAND to logically OR outputs.

2. Which MSI function – multiplexer or decoder would best implement multiple output functions (i.e. many functions of the same input variables)? Why?

A decoder would be better to implement multiple output functions because we can get selected minterms with a multiplexer, we get the unreduced for of the function.

1. What are the advantages of using an FPGA over MSI devices or SSI devices?
   1. An FPGA is used over and MSI or SSI because they are cheaper, the boarders are smaller and require less power.

4. In this experiment, the enable line is used to obtain a 3-to-8 decoder. Using the D3\_8E 3-to-8 decoders, how would you obtain/implement a 5-to-32 decoder? Draw a block diagram schematic of a 5-to-32 decoder.



5. Write the Test Plan of how this experiment should be tested.

Test plan above.

6. Write the Design Specification Plan to verify that the all the requirements have been met.

Design Specification is above.