Experiment 5

Decoder and Demultiplexer

Digital Systems

EEE3342C-20FALL 0011

Spenser Tacinelli
PID: sp498810

Pre-Laboratory Assignment:

1. Read this experiment to become familiar with this experiment.

2. Draft the Design Specification Plan.

3. Draft the Test Plan for the experiment.

Objective:

To learn to design and implement decoders and how to select one output at a time. A 2to4 and 3to8 decoders will be implemented.

Equipment:

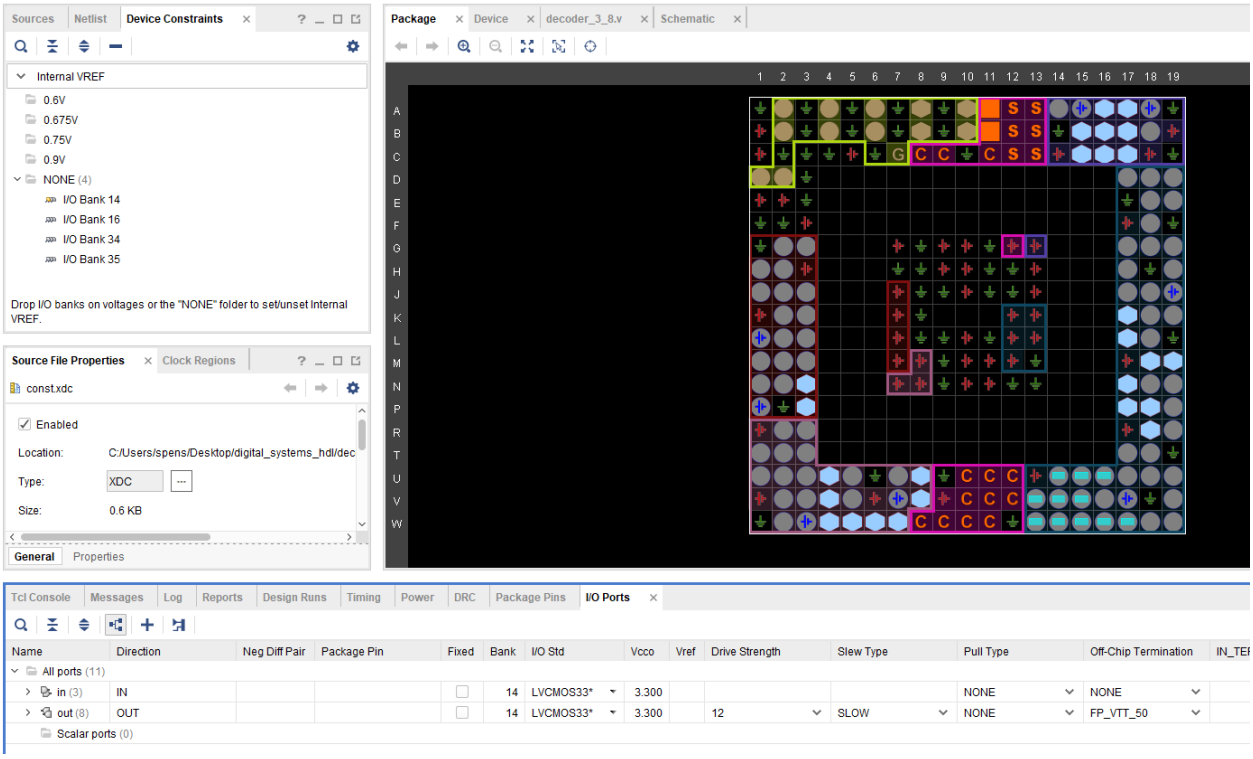
The Xilinx's FPGA VIVADO HLx Editions design tools are available in the laboratory. These tools can also be downloaded from Xilinx's web site at www.xilinx.com. The WebPack version of this tool that we use for the laboratory experiments are located under the support download section at the related website (https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloa dNav/vivado-design-tools/2017-4.html). Please take note that the file download size exceeds 1 GB and also during the installation process updates may have to be installed. It can take you up to a few hours to download and install the software on your computer. The user does not need to have the BASYS development board interface to the computer to design and simulate an FPGA.

Design Steps:

I created the project (decoder_proj) and designed the 2-to-4 decider with its input and outputs. Then I designed the 3-to-8 decoder and used to 2-to-4 decoder in the 3-to-8 decoder.

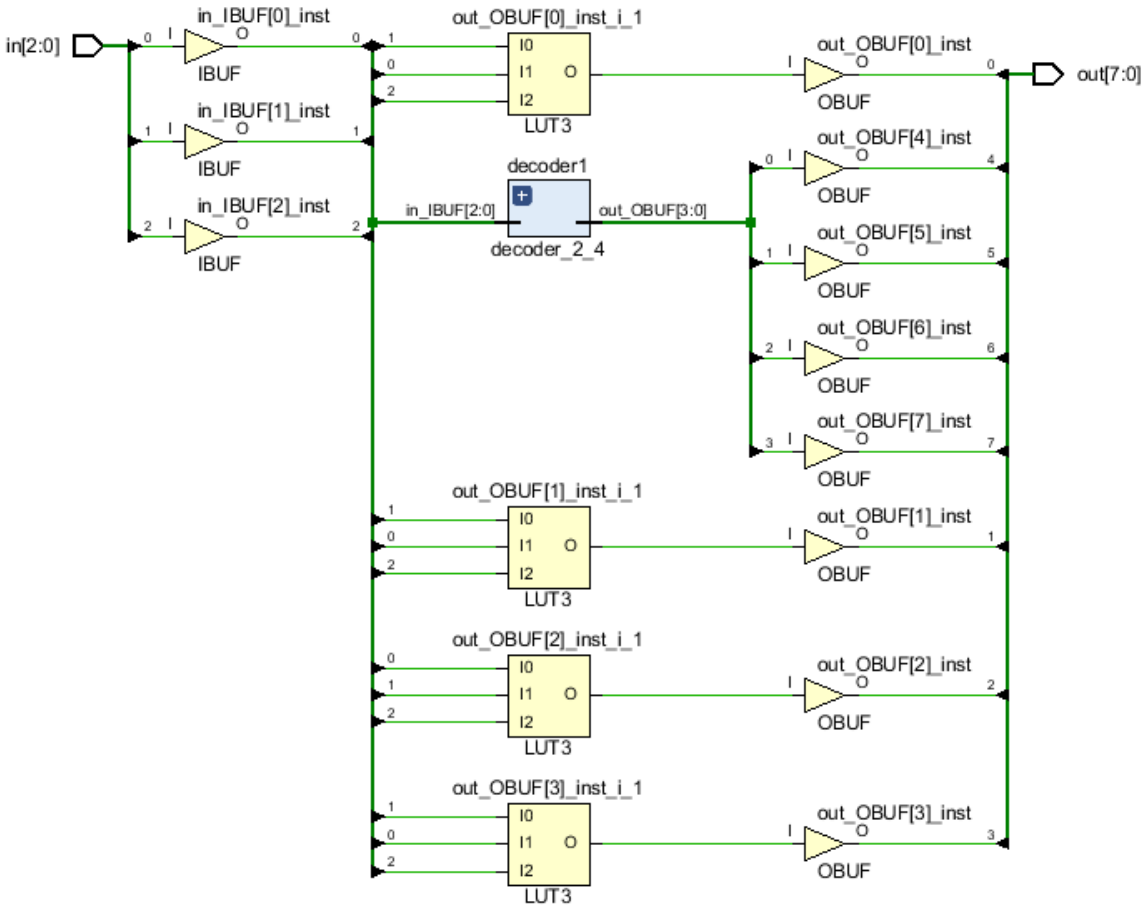
```
module decoder_2_4(en, in, out);
    input en; // Enable signal
    input [1:0] in;
    output reg[3:0] out; // Declare 'out' as a register
    always@(en or in) // Fill the sensitivity list begin
     if(en==1)
     begin
     case(in)
     2'b00: out = 4'b0001;
     2'b01: out = 4'b0010;// Fill this line
     2'b10: out = 4'b0100; // Fill this line
     2'b11: out = 4'b1000; // Fill this line
     endcase
     end
     else
     begin
     out = 4'b0000; // What should 'out' be?
     end
endmodule
module decoder_3_8 (
    input [2:0] in, // 3-bit input
    output [7:0] out // 8-bit output
    //wire wire1; // Declaring a wire
    );
    //module mydecoder24vlog (en, in, out);
     decoder_2_4 decoder0 (
     .en(~in[2]),
     .in(in[1:0]),
     .out(out[3:0])
     );

    decoder_2_4 decoder1 (
    .en(in[2]),
    .in(in[1:0]),
    .out(out[7:4])
    );
endmodule
```

I then ran synthesis and implementation, to implement I/O Planning and Schematics. I then changed the I/O std to LVCMOS33 in the I/O Planning.



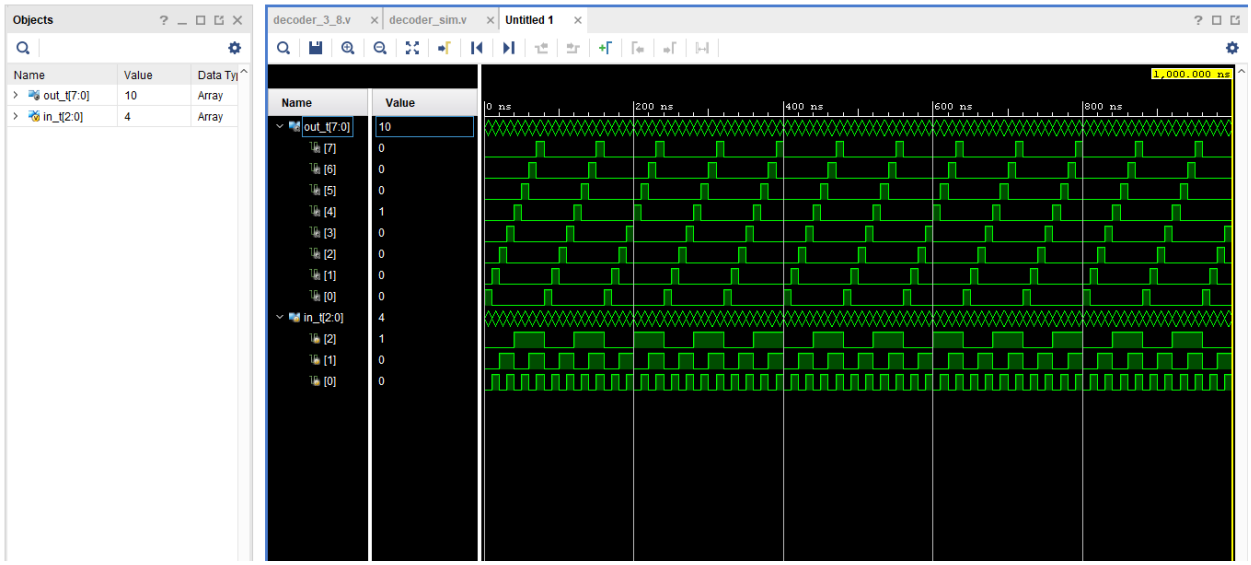
Then I generated the schematic of the 3-to-8 decoder.

I then programmed the logic for the simulation.


in[2:0]
in_IBUF[0]_inst
IBUF
in_IBUF[1]_inst
IBUF
in_IBUF[2]_inst
IBUF
out_OBUF[0]_inst_i_1
LUT3
decoder1
in_IBUF[2:0]
out_OBUF[3:0]
decoder_2_4
out_OBUF[1]_inst_i_1
LUT3
out_OBUF[2]_inst_i_1
LUT3
out_OBUF[3]_inst_i_1
LUT3
out_OBUF[0]_inst
OBUF
out_OBUF[4]_inst
OBUF
out_OBUF[5]_inst
OBUF
out_OBUF[6]_inst
OBUF
out_OBUF[7]_inst
OBUF
out_OBUF[1]_inst
OBUF
out_OBUF[2]_inst
OBUF
out_OBUF[3]_inst
OBUF
out[7:0]

```
module decoder_sim(

    );

        wire [7:0] out_t;
        reg [2:0] in_t;

        decoder_3_8 UUT(
        .out(out_t),
        .in(in_t)

        );

        initial begin
            in_t = 3'b000;
        end

        always #10 in_t[0] = ~in_t[0];
        always #20 in_t[1] = ~in_t[1];
        always #40 in_t[2] = ~in_t[2];


endmodule
```

I then ran a behavioral simulation.

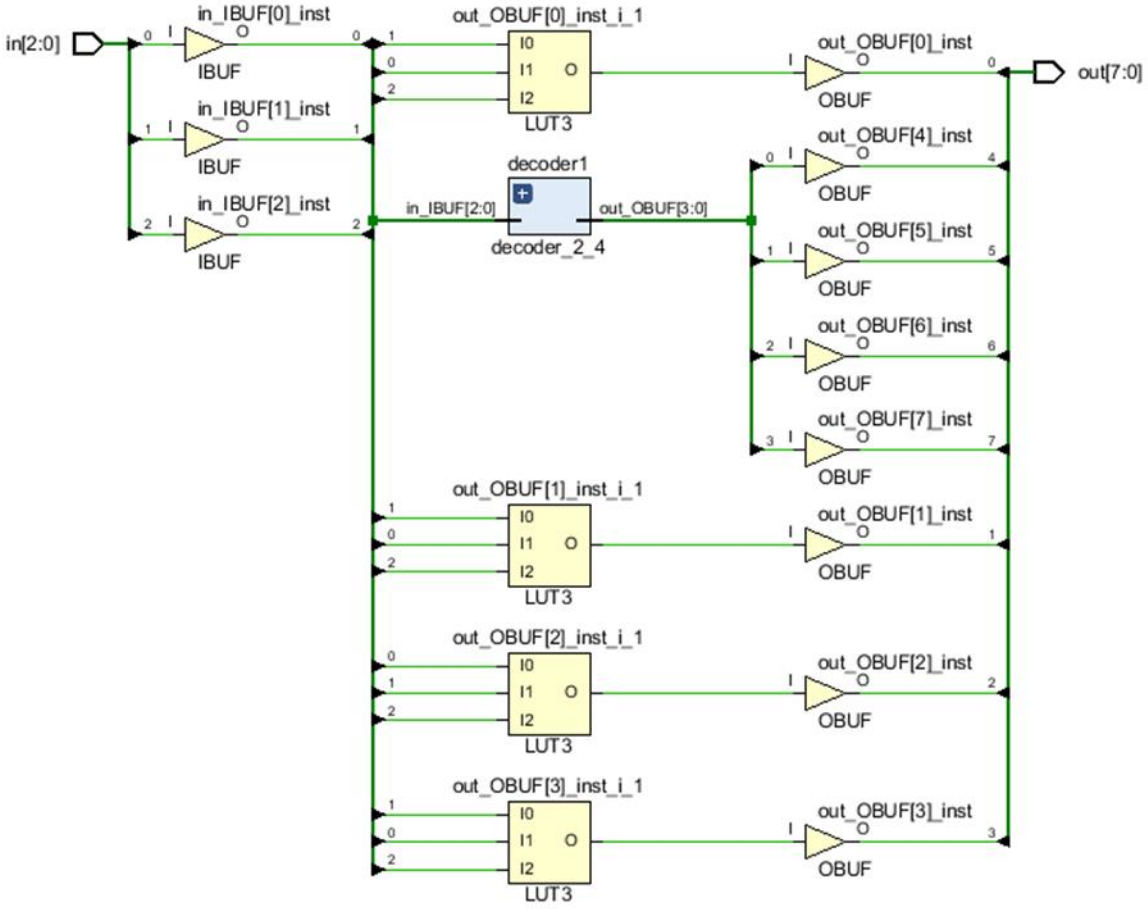


(behavioral)

3-to-8 Decoder circuit Schematic

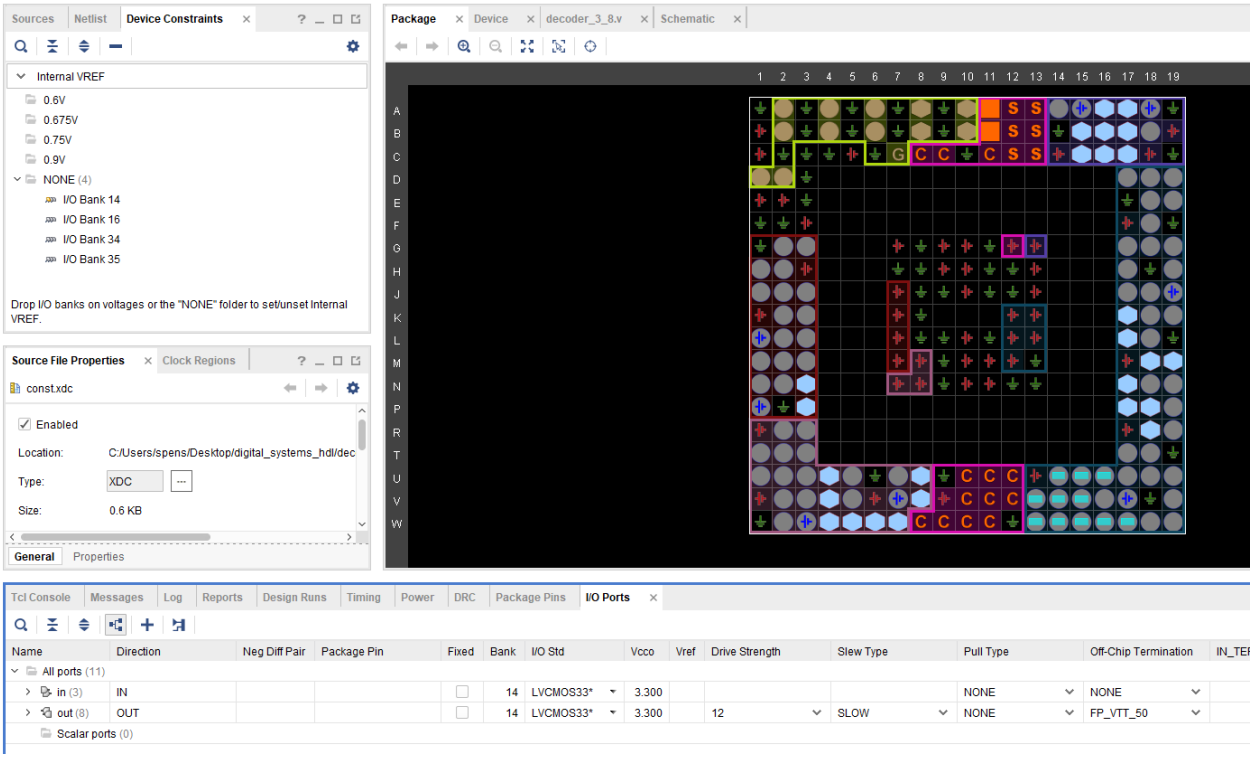
Logic Diagram:

The resulting schematic diagram for the 3-to-8 decoder.



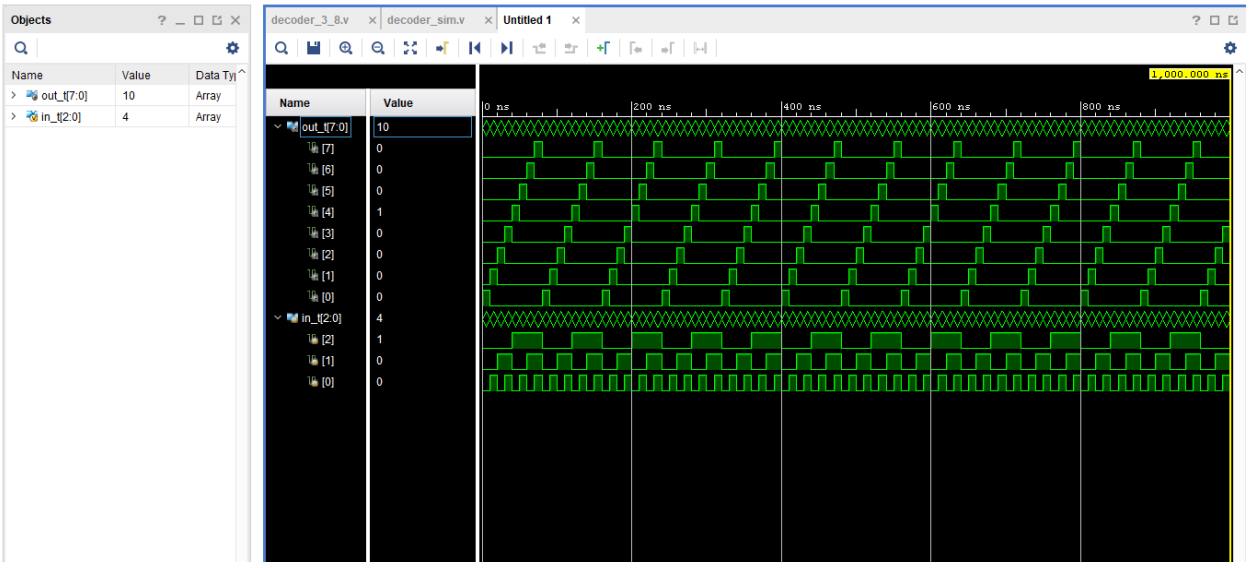
Design Specification Plan:

This is the design specification for a3-to-8 decoder. It has one input (in[2:0]) and one output (out[7:0]).



Results Statement:

Here is the behavioral simulation results for the multiplexer.

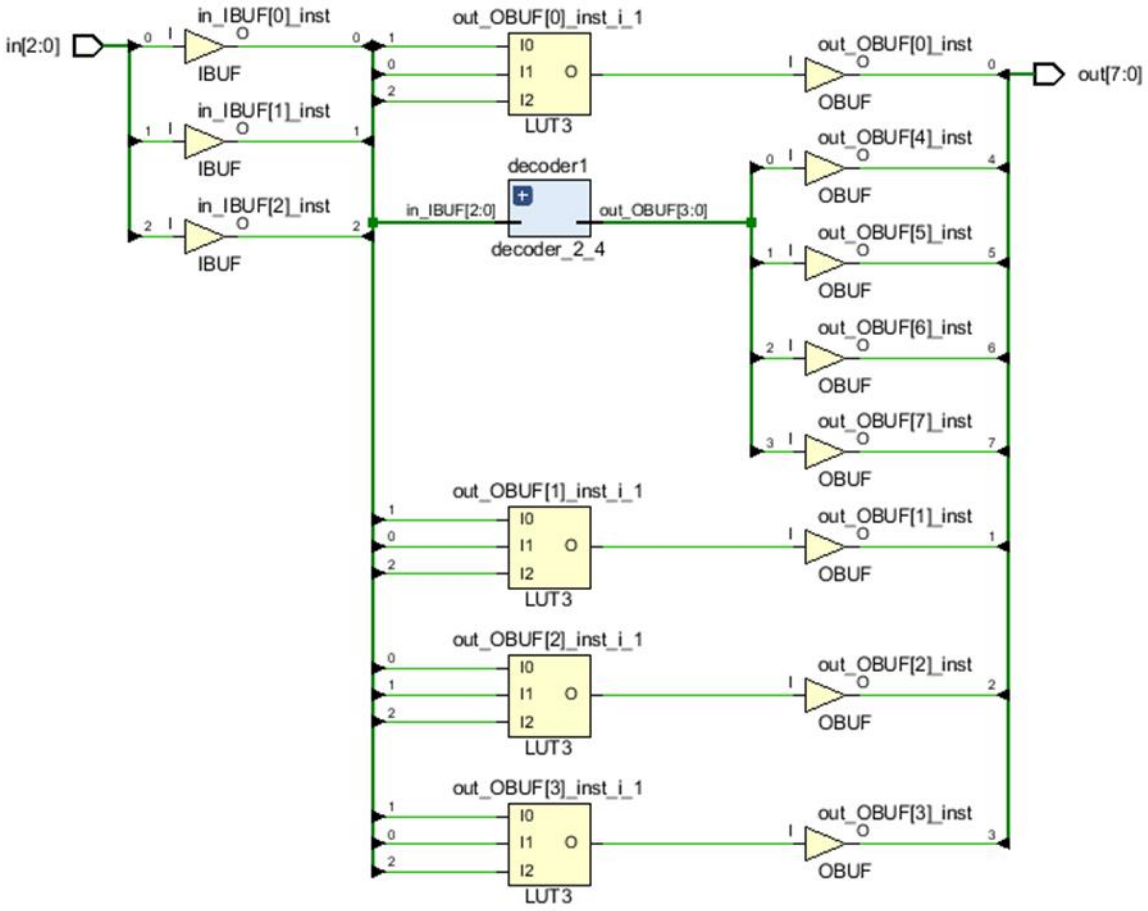


In2	In1	In0	Out7	Out6	Out5	Out4	Out3	Out2	Out1	Out0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

3-to-8 Decoder circuit Verilog

Logic Diagram

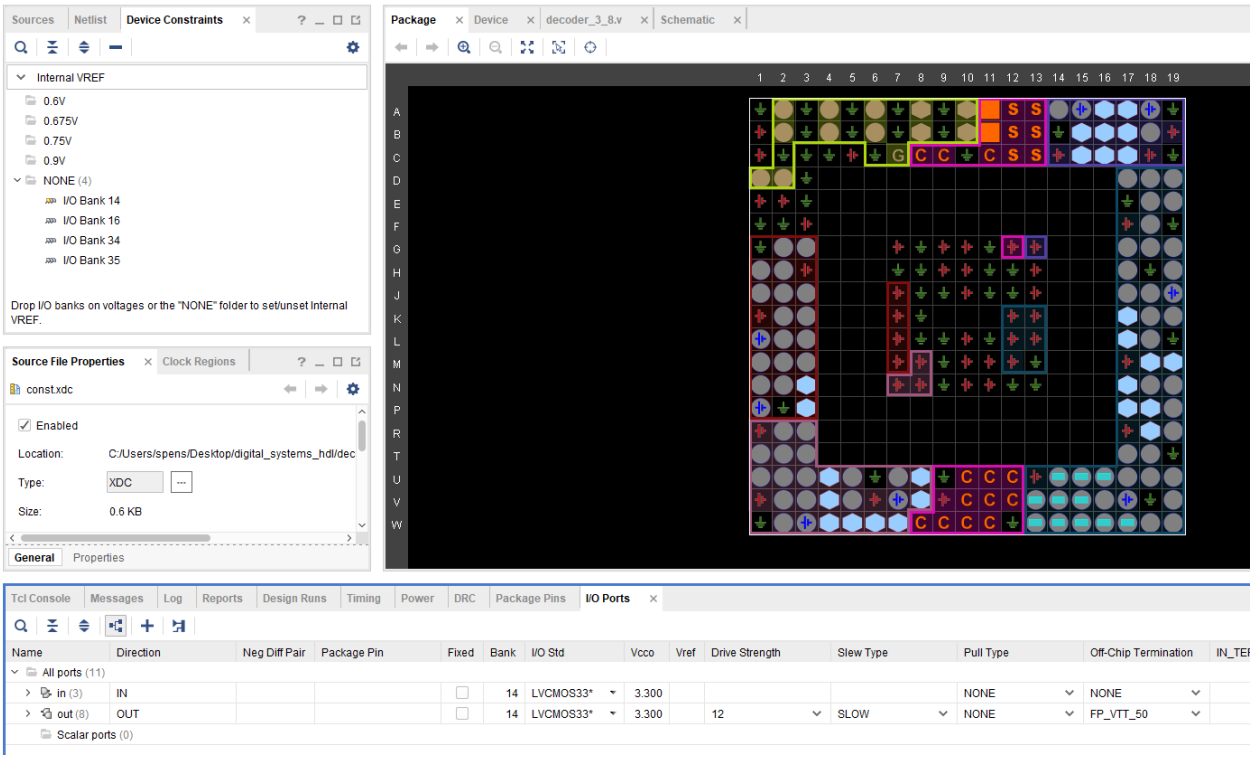
The resulting schematic diagram for the 3-to-8 decoder.



Design Specification Plan

This is the design specification for a multiplexer. It has two inputs (in[2:0]) and one output (out[7:0]).

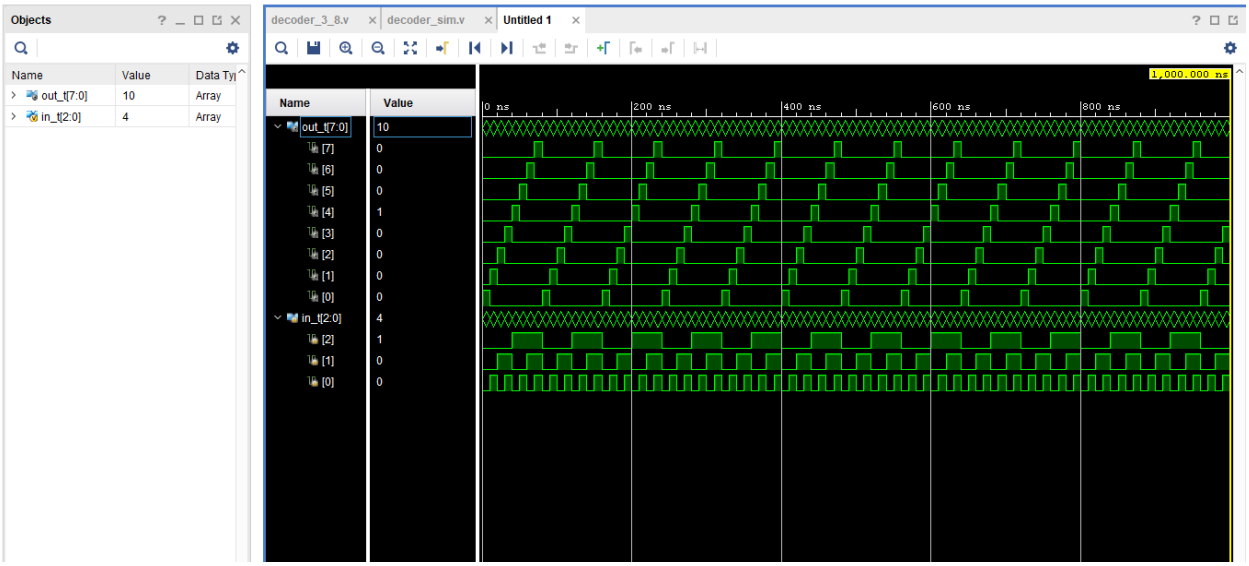
```
module decoder_sim(

    );

        wire [7:0] out_t;
        reg [2:0] in_t;

        decoder_3_8 UUT(
        .out(out_t),
        .in(in_t)

        );

        initial begin
            in_t = 3'b000;
        end

        always #10 in_t[0] = ~in_t[0];
        always #20 in_t[1] = ~in_t[1];
        always #40 in_t[2] = ~in_t[2];


endmodule
```



Results Statement

Explanation of pictures and tables.

Here is the behavioral simulation results.



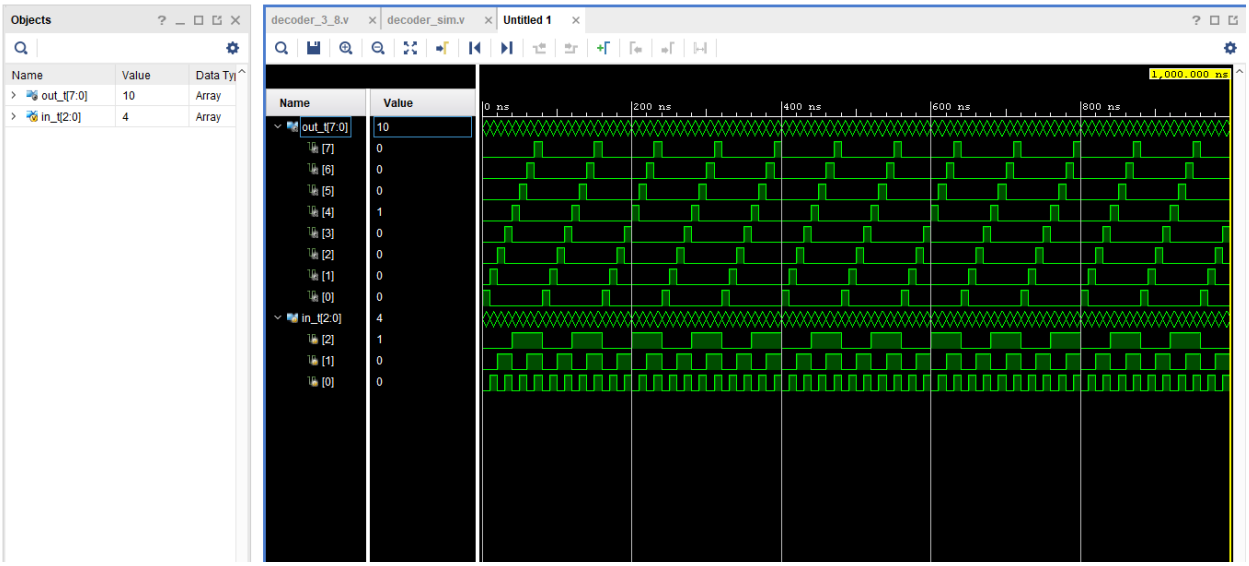
In2	In1	In0	Out7	Out6	Out5	Out4	Out3	Out2	Out1	Out0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Test Plan

The design was tested through a behavioral simulation that looped through every possible input for each selector option paired with each input option, which would give every output.

Burglar alarm controller circuit

Here is the behavioral simulation results.



Conclusion

All of my results came out as expected and agreed with each other, including the simulation. By analyzing the simulation and the 2-to-4 decoders that were used to build the 3-to-8 decoders, with some trial and error with the programming logic, it really helped with understanding how the 3-to-8 decoder works.

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

A NAND gate can be used to “logically OR” the output due to the fact that a negated OR the same as a NAND.

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

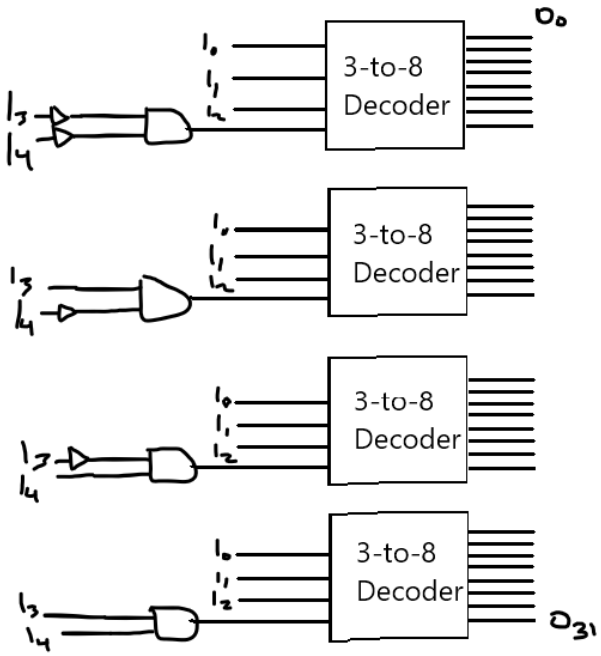
A decoder would be the best choice the implement multiple output functions since a decoder can do minterm expansion of a function.

3. What are the advantages of using an FPGA over MSI devices or SSI devices?

Cheap, physically small, and very low power usage.

4. In this experiment, we used the enable line in the 2-to-4 decoder to build a 3-to-8 decoder from two 2-to-4 decoders. What needs to be added to our 3-to-8 decoder in order to be able to build a 4-to-16 decoder using two 3-to-8 decoders? What would we need to add to said 4-to-16 decoder in order to build a 5-to-32 decoder using two 4-to-16 decoder? Draw a schematic of a 5-to-32 decoder using four 3-to-8 decoders.

4 3-to-8 decoders are needed for a 5-to-32 decoder.



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

6. Write the Product Specification Plan to verify all the requirements have been meet.