Lab #7

**Objective:**

To gain experience in using VHDL to design circuits, such as designing decoders and multiplexors, as well as implementing functions using decoders.

**Design:**

Formulas:

Seven Segment:

4:16 Decoder with Enable:

Decoder Functions:

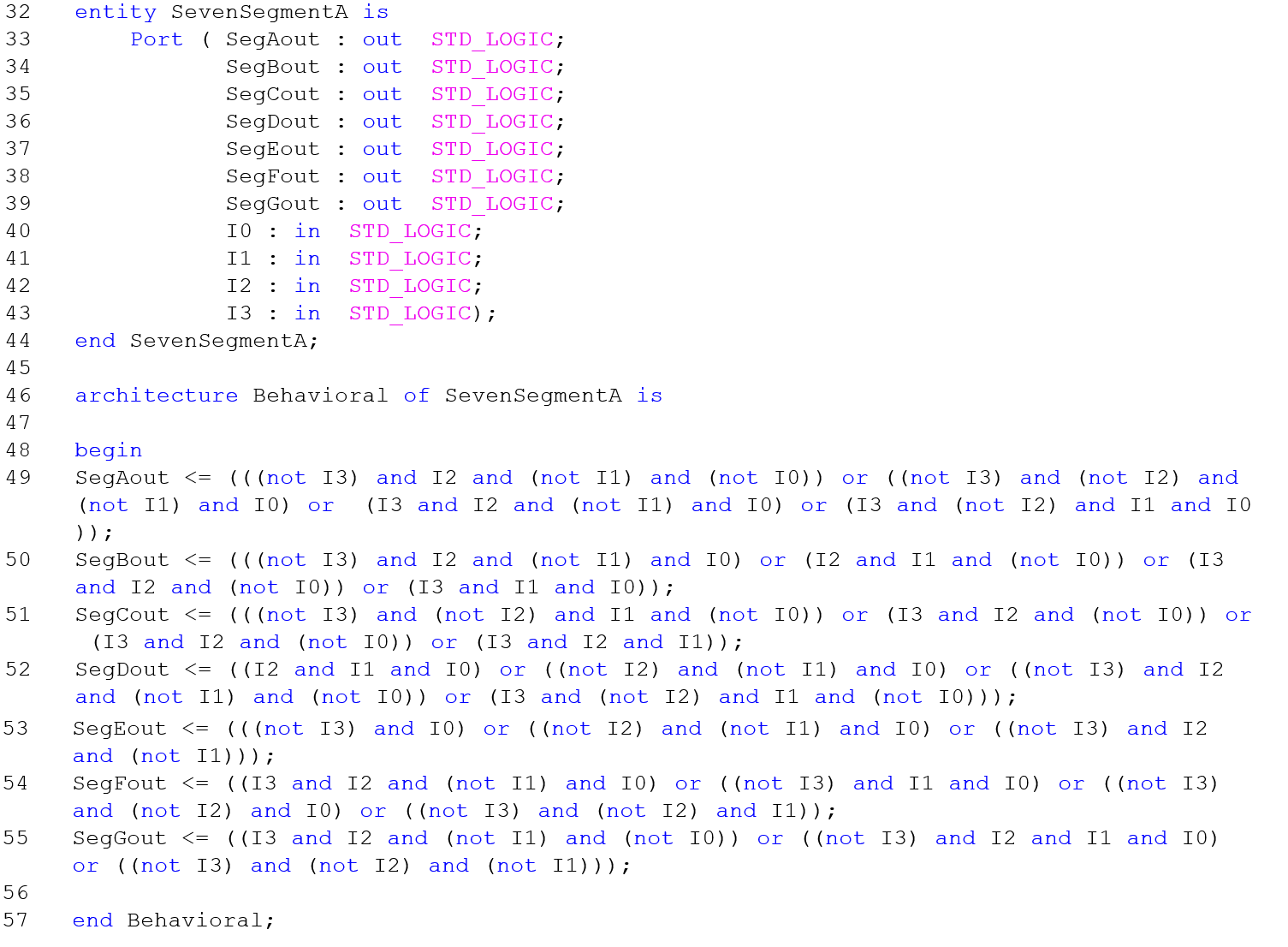


Figure 1 - VHDL for Seven Segment Display

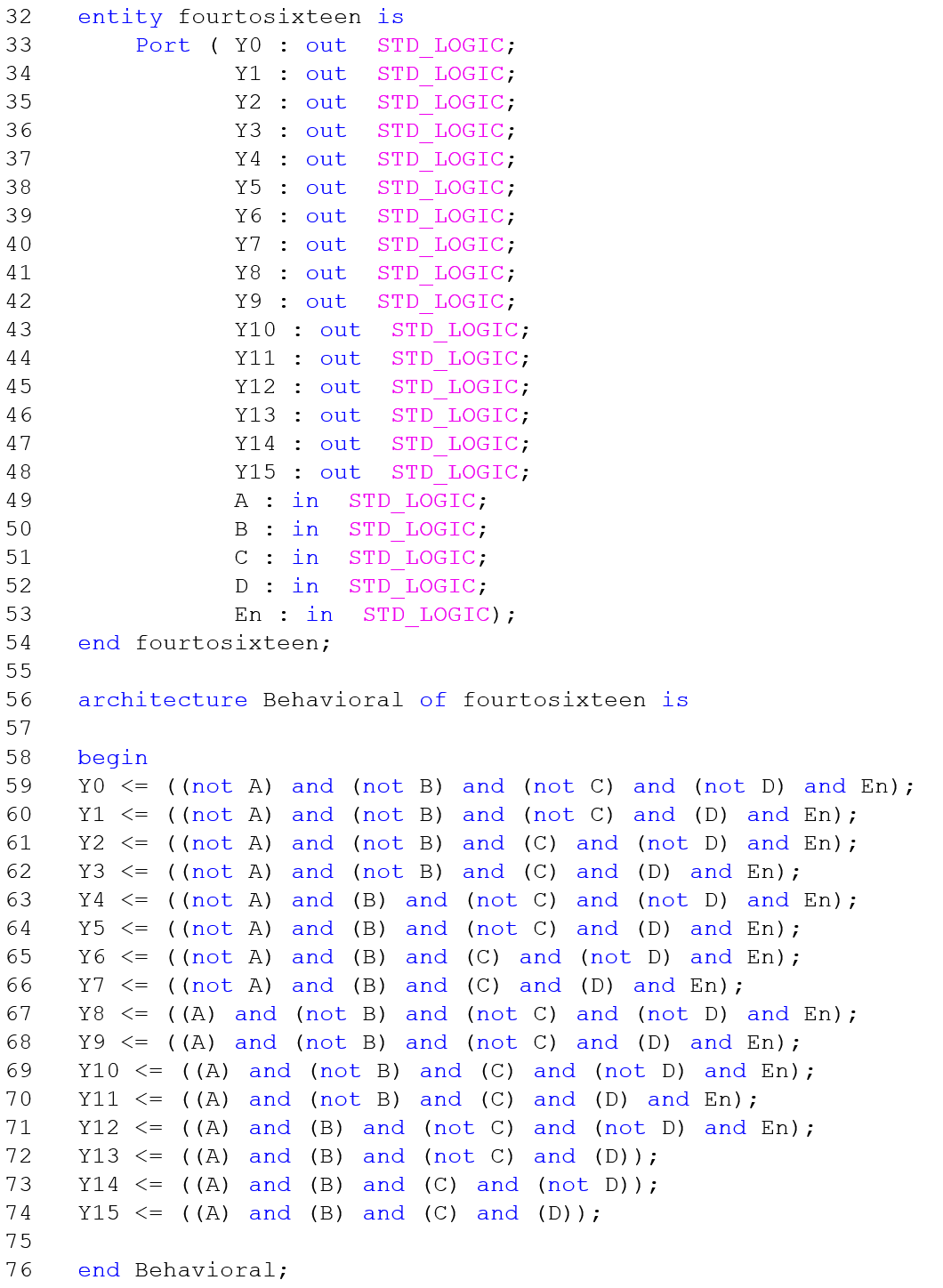


Figure 2 - VHDL for 4:16 Decoder

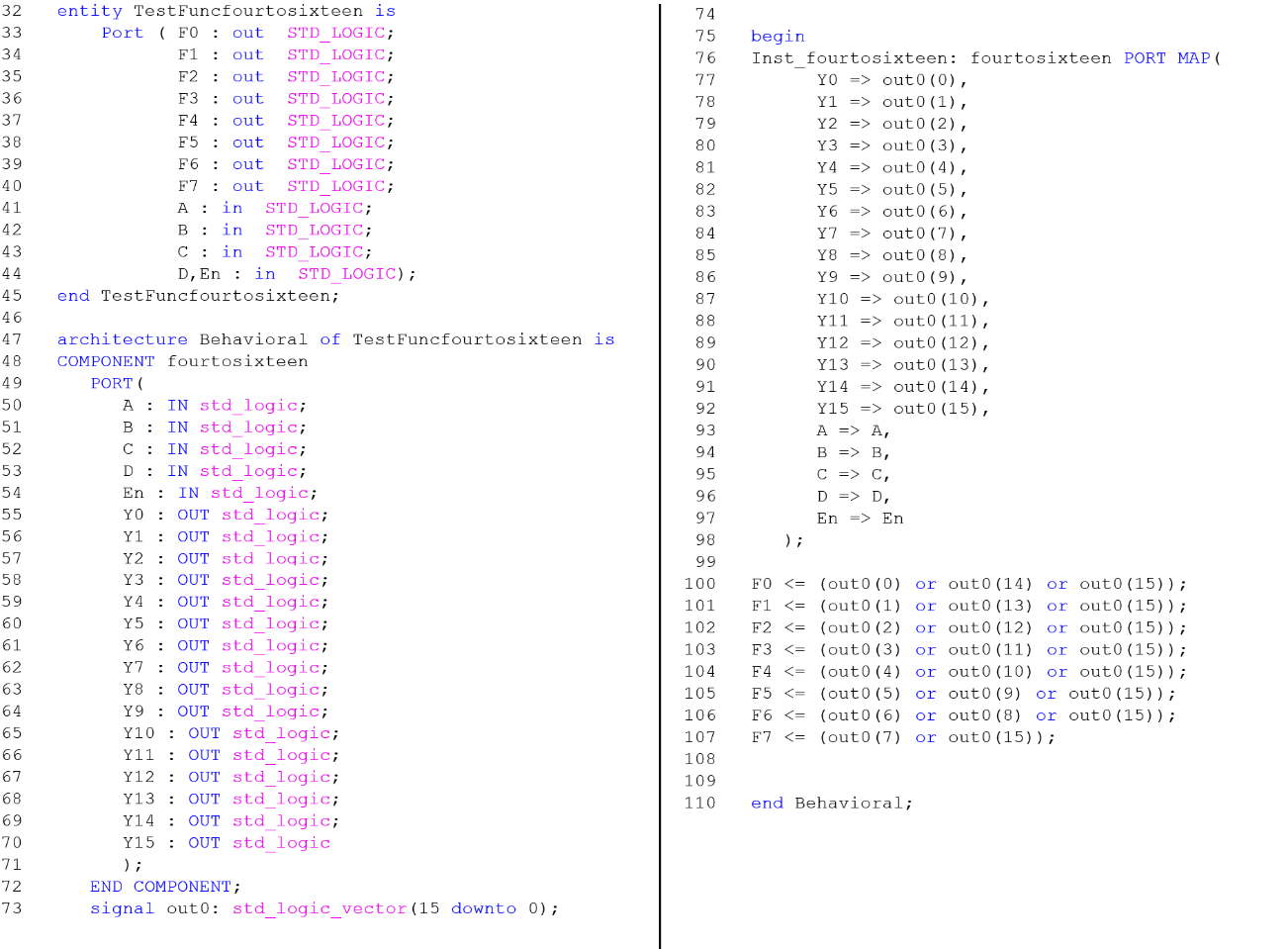


Figure 3 - VHDL for 4:16 Decoder to Implement Functions

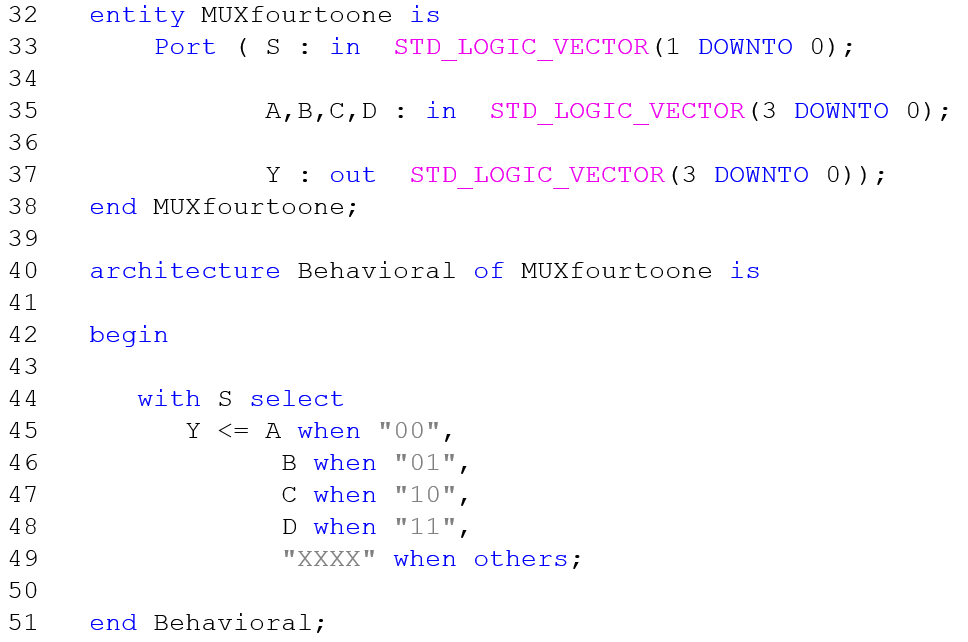


Figure 4 - VHDL for 4set:1set Multiplexer



Table 1 - Seven Segment Display Truth Table



Table 2 - 4:16 Decoder with Enable and Functions Truth Table



Table 3 - 4set:1set Multiplexer Test Vector Truth Table

**Procedure:**

**Part 1:**

* Make a Seven Segment Display circuit using VHDL, using equations from Lab 5
  + Make a new VHDL Module
    - Project -> New Source
    - Select VHDL Module
    - Enter an appropriate file name
    - Add an output for each segment A through G
    - Add four inputs
  + Add the equations for each segment between begin and end behavioral in the following general format:
    - SegmentName <= (formula);
* Synthesize the file and run a clock driven behavioral simulation, using 16ns for the most significant bit
* Download the design to the FPGA board and test all inputs

**Part 2**

* Make a 4:16 decoder with enable in VHDL
  + Make a new VHDL Module
    - Project -> New Source
    - Select VHDL Module
    - Enter an appropriate file name
    - Add and label 16 outputs (0-15)
    - Add and label 5 inputs (0-3 and Enable)
  + Add the equations between begin and end behavioral in the following general format:
    - OutputName <= (formula);
* Synthesize the file and run a clock driven behavioral simulation, using 16ns for the most significant bit
* Implement the given functions in a new VHDL file, using the 4:16 decoder as a component
  + Create a new VHDL file as before, with 5 inputs and 7 outputs
  + To make a component
    - Select 4:16 file
    - Expand Design Utilities
    - Double click VHDL Instantiation Template
  + To implement the template in another VHDL File
    - Copy and paste the component section between the Architecture and Begin statement
    - Copy and paste the component section between the Begin and End Behavioral statement
  + Define a signal vector to connect between the decoder and the function outputs
    - Declare the signal between end component and begin in the general following format:
      * signal signalName: std\_logic\_vector(15 downto 0);
  + Define the functions just before end Behavioral;
    - Implement the given functions in the general following format:
      * functionName0 <= (signalName(0) or signalName(14) or signalName(15));
* Synthesize the file and run a clock driven behavioral simulation, using 16ns for the most significant bit
* Download to the FPGA and test for all combinations

**Part 3**

* Write a VHDL code to implement a 4set:1set multiplexer
  + Make a new VHDL Module
    - Project -> New Source
    - Select VHDL Module
    - Enter an appropriate file name
    - Add input vectors
      * 1 downto 0 for select lines
      * 3 downto 0 for the four sets of inputs (e.g. A,B,C,D)
    - Add output vector
      * 3 downto 0
  + Add the equations between begin and end behavioral using with…select
* Copy and modify .do file Lab 6
  + Use find/replace to correct the file and variable names
* Synthesize and run a behavioral simulation using the .do file

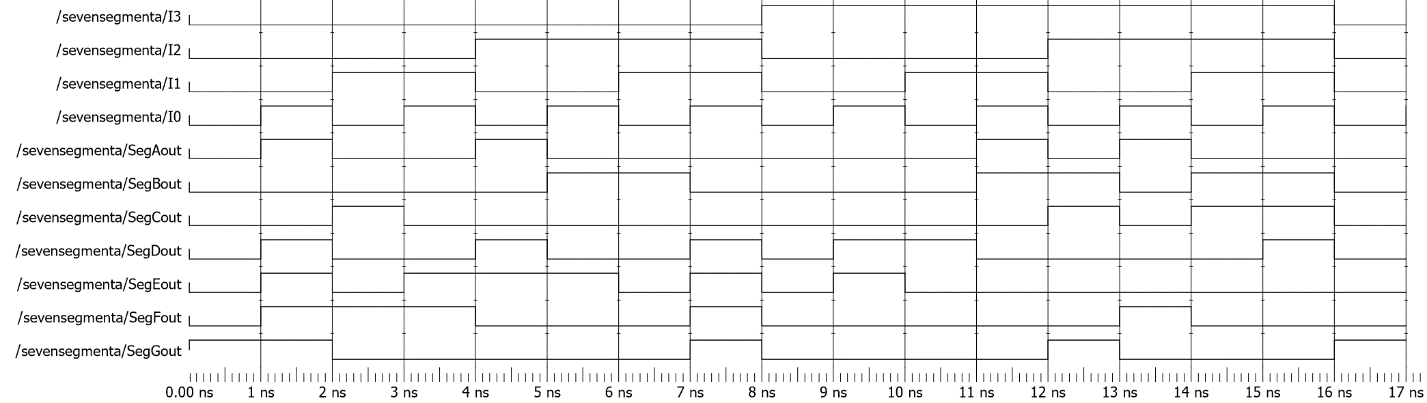
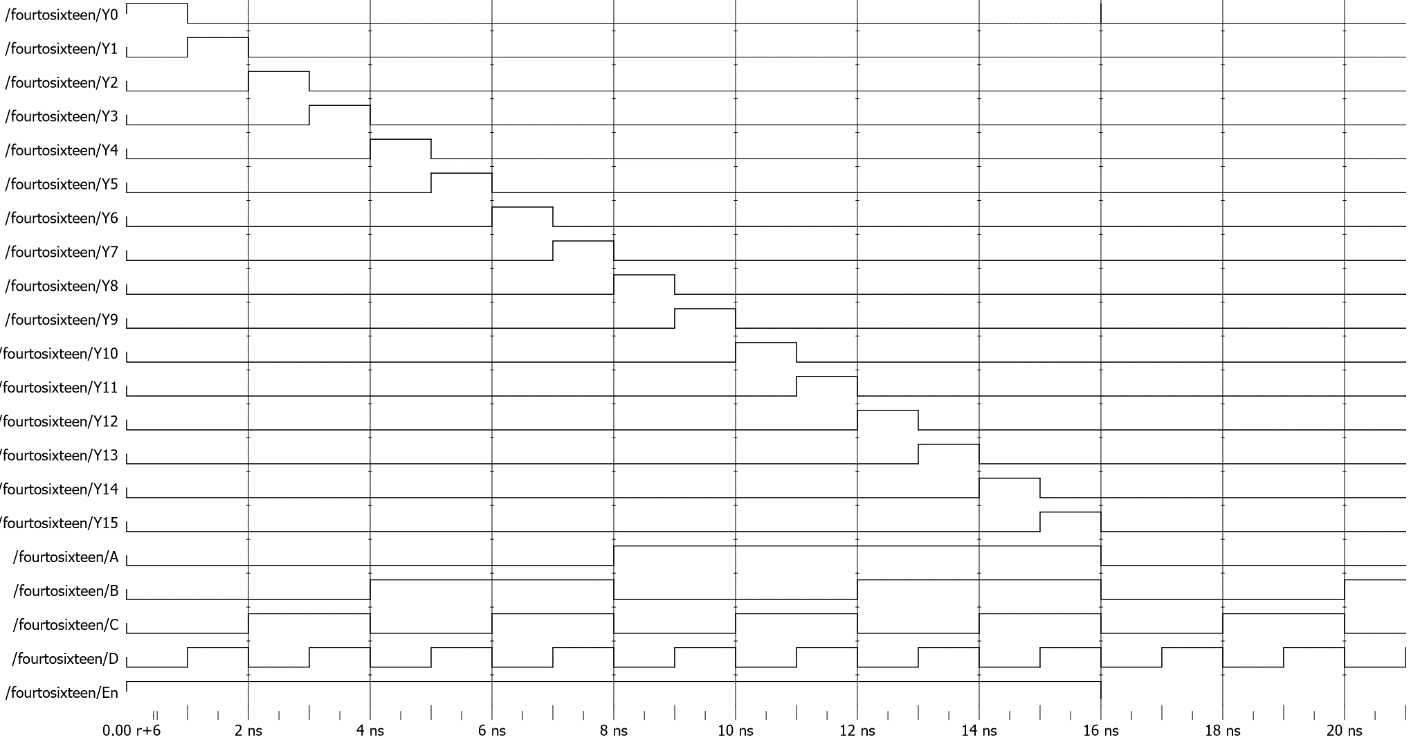
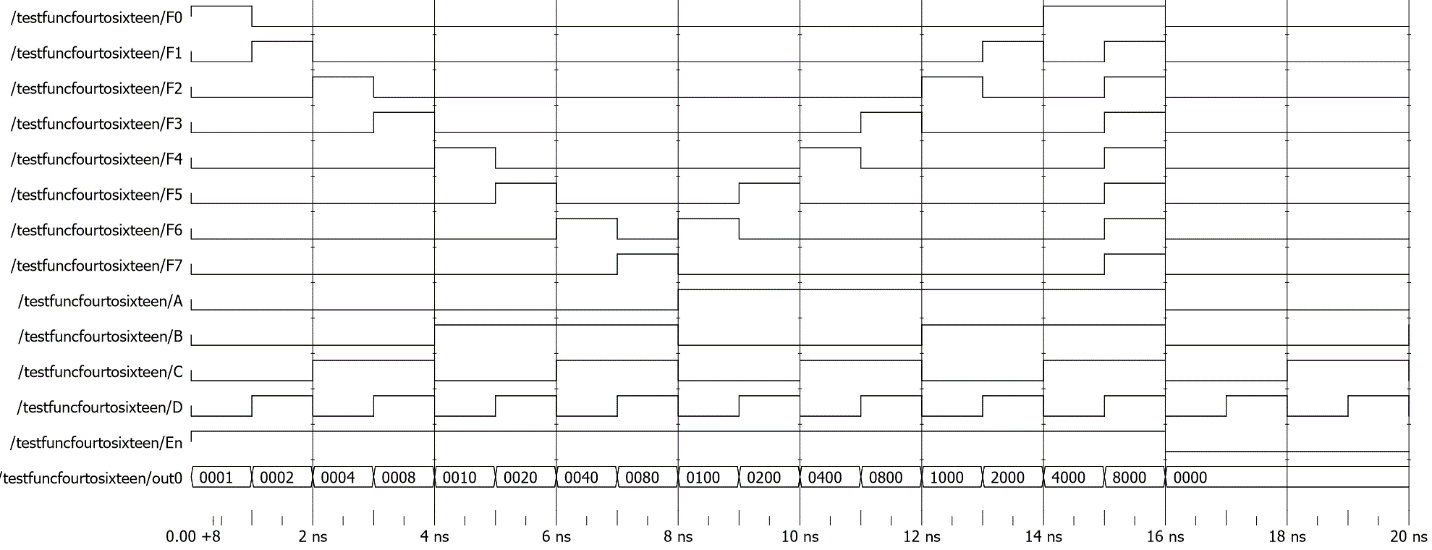
**Data:**

Figure 7 - Decoder Function Behavior Simulation

Figure 6 - 4:16 Decoder Behavioral Simulation

Figure 5 - Seven Segment Behavioral Simulation

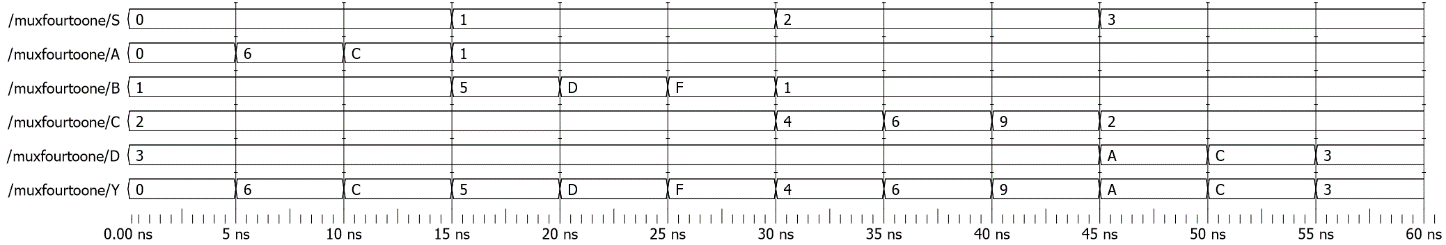


Figure 8 - 4set:1set Multiplexer Behavioral Simulation

**Data Analysis:**

The behavioral simulations matched the truth tables for all of the test values.

**Conclusion:**

This experiment was very useful introduction to VHDL, especially using circuits that we had made in previous experiments. Instead of clicking and dragging multiple gates and connecting input and output lines, the entirety of the circuits could be written out in a few lines. The most obvious example is the 4set:1set multiplexer only requiring 6 lines of code (excluding variable declarations).

I did make a mistake in declaring the inputs and outputs for the decoder by declaring all of the inputs and outputs individually, which made connecting the signals and implementing the functions messy. In the future, when there are multiple inputs and outputs like that I can use vectors to make the file a little cleaner, like how they are implemented in the multiplexer.

One mistake I noticed after the fact was that I left the enable input out of the functions for the last 3 functions of the 4:16 decoder. It did not skew my tested results as they generally were tested with enable on.