Spring 2019

California State University, Northridge

Department of Electrical & Computer Engineering

Lab Experiment 1

Familiarization with Linux and the Synopsys VCS Simulator

February 7, 2019

ECE 526L

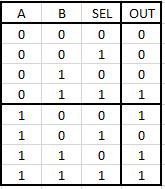
Written By: Juan Silva

**Introduction:**

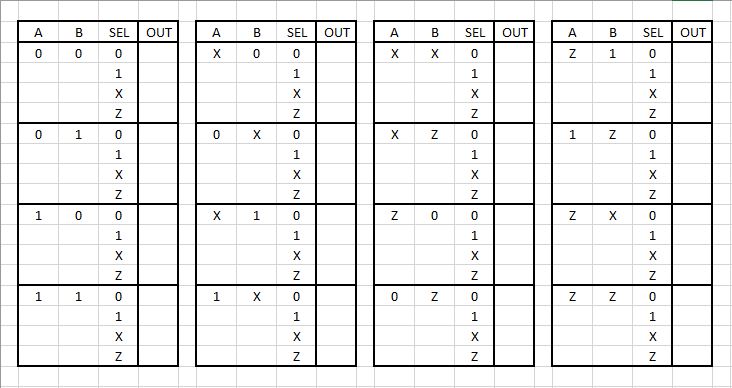
In this experiment, a Linux-based workstation is set up Verilog. A 2-input MUX is provided using Verilog built in gate-level primitives and simulated using the Verilog Compiled Simulator (VCS). A test bench is used to verify the functionality of the 2-input MUX and is modified to exhaustively test the MUX module. VC tools are used to analyze the input / output signals of the MUX module via text-based output or a waveform graph. Exhaustive testing is done by considering a quad-input system including the unknown value X and high impedance Z.

**Procedure:**

1. A workstation is set up by create a file directory system for each experiment to be organized. UNIX commands are used since the workstation uses a Linux-based operating system.
2. The code for the 2-input MUX and its testbench are written in a text editor and simulated using the VCS software. A log file is created to verify the results of the given test bench.

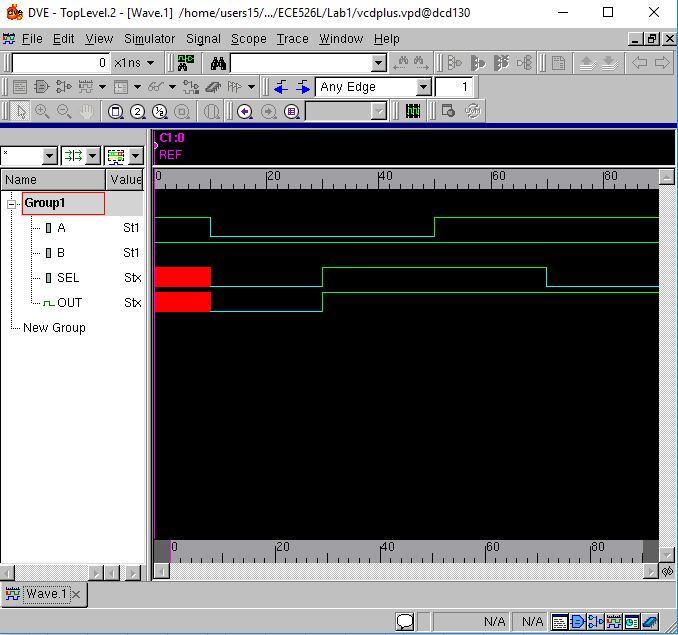
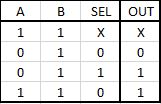


***Figure 1.1*** *- Truth Table for 2-Input MUX*

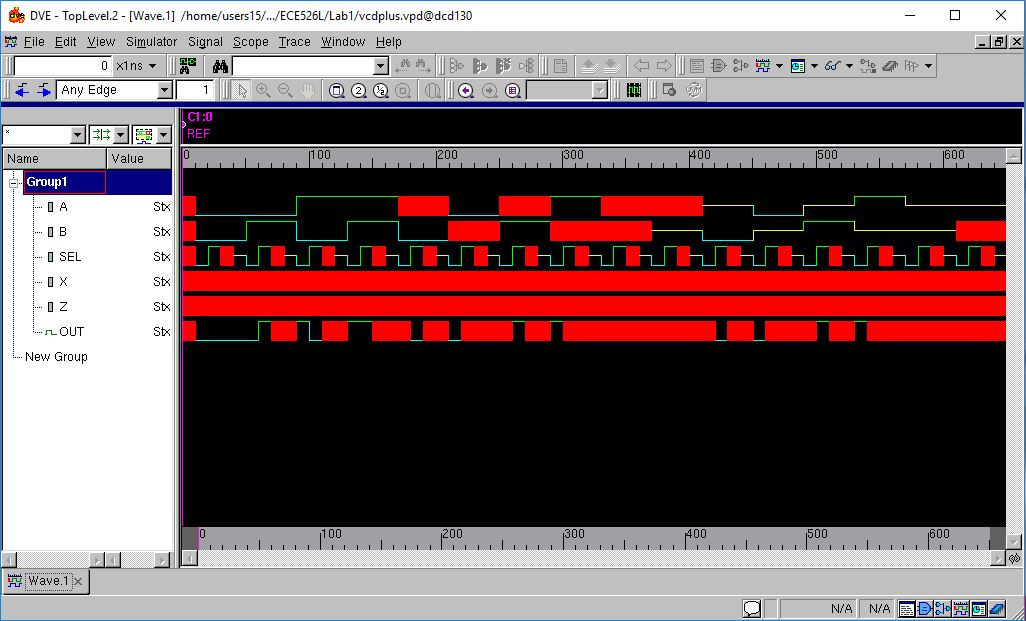
1. The testbench is modified for exhaustive testing by considering a quad-input system.   
   This was done by creating a truth table that would generate all possible combinations of a quad-input system. A log file is created to verify the results of the new test bench.
2. The truth table was created by choosing the least significant bit and generating all four cases while interchanging the higher bits. Given four inputs possible inputs in a three input MUX (A, B, and SEL), there will be a total of 64 combinations (4^3). Each interchangeable combination of A and B will have four patterns:  
     
   

***Table 1.1*** *- Quad Input Truth Table*

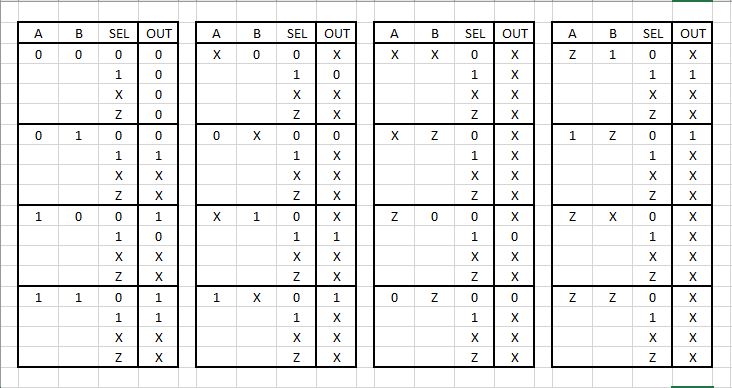
**Results:**

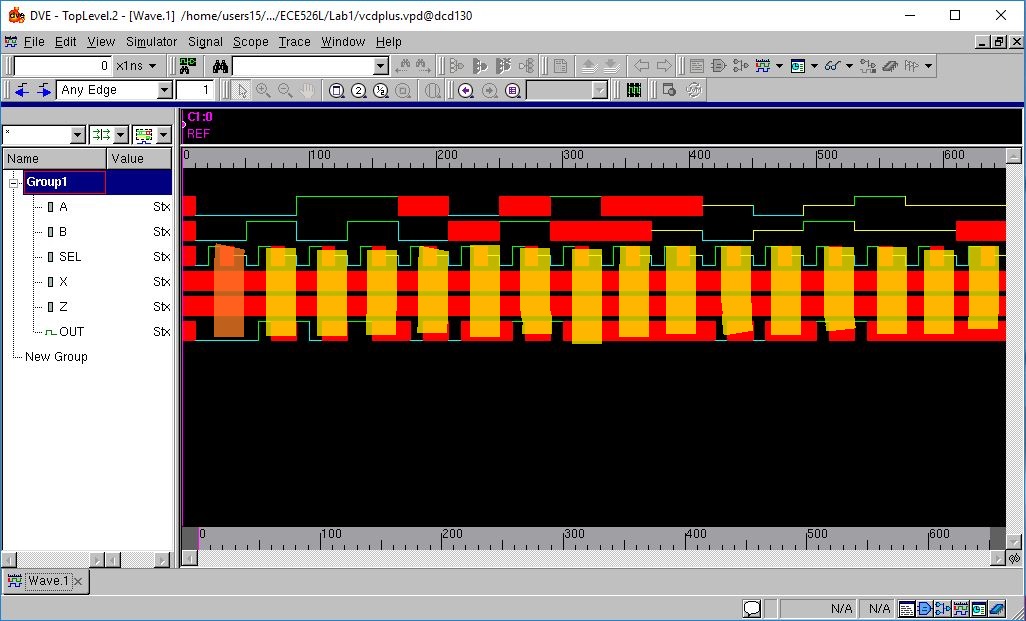
***Figure 1.2*** *- Simulated Waveform & Truth Table (Testbench 1)*



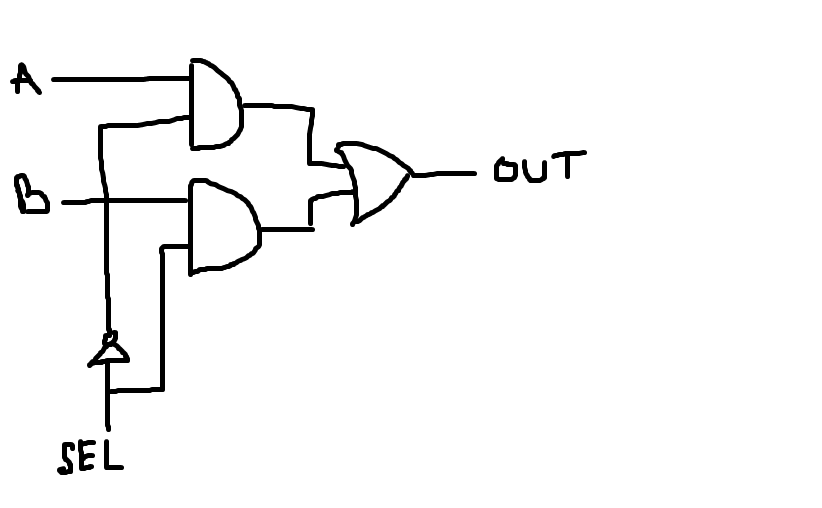
***Figure 1.3*** *- Simulated Waveform (Testbench 2)*



***Figure 1.4*** *- Quad-Input Truth Table Results (Testbench 2*)

****

***Figure 1.5*** *- MUX Waveform: Unknown and High Impedance Inputs*

**

***Figure 1.6*** *- Gate Level Logic Sketch of 2-Input MUX*

**Discussion:**

For the first testbench, the generated waveform matches the truth table in *Figure 1.1* and *Figure 1.2* (simplified) verifying the functionality of the circuit. For the second testbench that requires an exhaustive testing, the generated waveform matches the truth table in *Figure 1.4.* This truth table was created by translating the gate-level primitives from the MUX module into combinational gates shown in *Figure 1.6*. That being said, following the combinational circuit shows that a value that is unknown or high impedance will produce an unknown result. This situation occurs when the SEL signal is of value X or Z and AND’d together with input A or B. The result of the boolean expression will produce an unknown output because one of the inputs are unknown. This also applies for a high impedance value since it is considered a floating value. This pattern is highlighted in yellow in *Figure 1.5*  whenever the SEL signal is set to unknown or high impedance. This is not the case when inputs A and B are both 0 (highlighted in orange). The result from the boolean expression will remain zero.

Perhaps the purpose of the exhaustive test is to demonstrate the behavior of an input error. In other words, the inputs are not read as a zero or a one but something in between. This pattern can be seen at the SEL line in *Figure 1.3.* The unknown and high impedance of the SEL line can be regarded as values within the transition between one and zero. Thus, the output will produce an unknown value as stated above. Additional, this supports the idea that Verilog primitives do not work the same as real gates. Looking back at the waveform, any case where the inputs are unknown and produce an unknown output, it is unclear what the actual value of the output is. Since primitives in Verilog are abstract, the output will always be unknown but it is indeterminate every time whereas a real logic gate would output either a 0 or a 1.

In the lab manual, two examples of behavioral code are shown for a 2:1 MUX. These implementations will perform logically identical but the gate-level design will be different. The main difference is the use of an enable. The MUX module with an enable will ensure that any time the MUX is realized will disabled will produce an unknown result whereas the gate-level primitives assumes that the MUX will always be enabled.

**Conclusion:**

This lab serves as an introduction to using the Verilog Compiled Simulator starting with gate-level primitives. A testbench is provided to test the validity of the module using the tools provided by VCS. Exhaustive testing was used to see the impact of primary and internal inputs and its output. The output was analyzed to compare the work of a simulated gate versus a real gate. Familiarizing with the simulation tools will be useful in verify the validity of a module as well as analyze its performance.