LAB 1:

Familiarization with Synopsys VCS

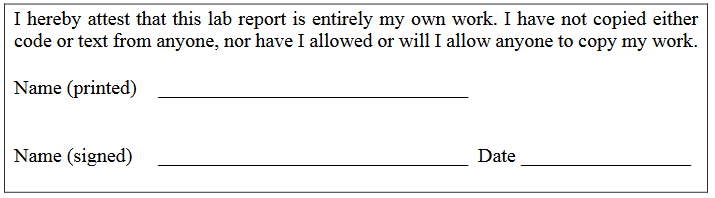
Professor Ronald Mehler

ECE 526L

Spring 2018

Garen Nikoyan

2/1/2018



**Objective:**

The purpose of this week’s lab is to become familiar with Synopsys VCS and to exhaustively test 2 inputs 1 output multiplexer. Because the test must include X and Z states, there should be a total of 64 different combinations of inputs to exhaustively test the module.

**Methodology:**

For this lab, the code for the MUX and a sample test bench was given. The sample test bench was then modified to not only test a handful of input combinations, but test all input combinations. This was done by locking one, and then two, of the inputs in one state, and varying the last one. The SEL input was the main input which was locked, and then the B input. SEL was set to 0, B to 0, and A was changed through all four different states. Then B to 1 while SEL stayed 0, and again A was changed, and so on, until all 64 combinations had been tested.

**Analysis:**

Below, the waveforms from running the simulation can be found. The output of the MUX is as expected, when the inputs are 0s or 1s. When the selected line (A if SEL=0, B if SEL=1) is X (unknown,) or Z (high impedance, floating,) the output is given as unknown. It is also interesting that when SEL is either X or Z, that the output is only 0 when both A and B are 0, and is X for the remaining values of A and B.

By modifying the $monitor system task, the gate level logic can be seen in the log. These can also be seen in the waveforms in figures 5 and 6. By looking at the log and waveforms, we can conclude that the Verilog primitives do work as real gates would work. The only major difference is that real gates would have a time delay, where Verilog primitives are almost instantaneous.

Just to see an example of the primitives working as real gates, at 170ns, A=1, B=0, SEL=1. The gates would be A1 = A & SEL\_N = 0, B1 = B & SEL = 0, and lastly, OUT = A1 | B1 = 0.

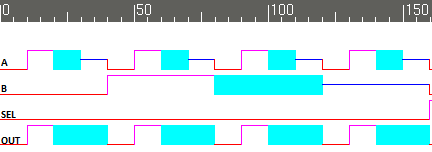


Figure 1. SEL= 0, with A and B being varied from 0, 1, X and Z, respectively.

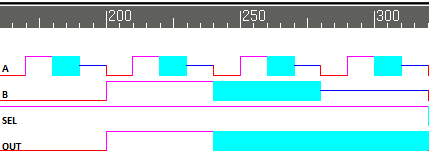


Figure 2. SEL= 1, with A and B being varied from 0, 1, X and Z, respectively.

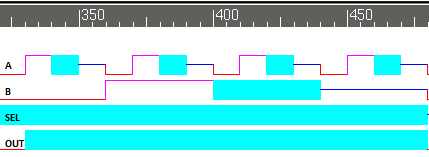


Figure 3. SEL= 1’bx, with A and B being varied from 0, 1, X and Z, respectively.

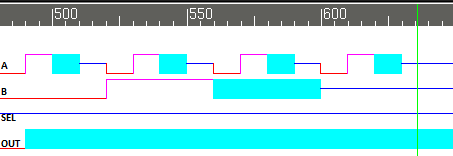


Figure 4. SEL= 1’bz, with A and B being varied from 0, 1, X and Z, respectively.

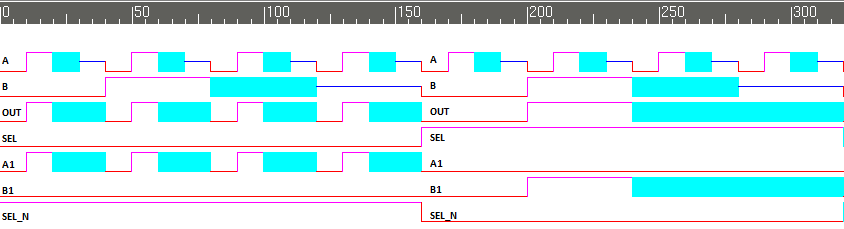


Figure 5. SEL = 0 and SEL = 1 cycles including gate level internal nodes

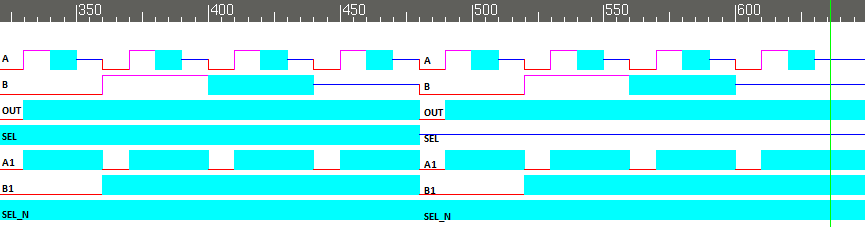


Figure 6. SEL = 1’bx and SEL = 1’bz cycles including gate level internal nodes

Module:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* ECE526L Experiment #1                Garen Nikoyan, Spring 2018 \*\*\*

\*\*\* Familiarization with Linux and the Synopsys VCS Simulator \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Filename: MUX2\_1.v       Created by: Garen Nikoyan, 1/31/2018 \*\*\*

\*\*\* -Revision History                           \*\*\*

\*\*\* 1/31/2018: Copied from ECE526L Lab Manual \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* This module models a 2:1 Multiplexer                 \*\*\*

\*\*\*                                     \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

`timescale 1 ns / 1 ns

module MUX2\_1(OUT, A, B, SEL);

// Port Declarations

input A, B, SEL;

output OUT;

// Internal variable declarations

wire A1, B1, SEL\_N;

// The netlist

not (SEL\_N, SEL);

and (A1, A, SEL\_N);

and (B1, B, SEL);

or (OUT, A1, B1);

endmodule

Testbench:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* ECE526L Experiment #1                Garen Nikoyan, Spring 2018 \*\*\*

\*\*\* Familiarization with Linux and the Synopsys VCS Simulator \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Filename: TB\_MUX2\_1\_V2 Created by: Garen Nikoyan, 1/31/2018 \*\*\*

\*\*\* -Revision History                           \*\*\*

\*\*\* 1/31/2018: Modified code copied from the lab manual \*\*\*

\*\*\* to test exhaustively \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* This module exhaustively tests a 2:1 Multiplexer module \*\*\*

\*\*\*                                         \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

`timescale 1 ns / 1 ns

module TB\_MUX2\_1();

reg A, B, SEL;

wire OUT, A1, B1, SEL\_N;

MUX2\_1 UUT(OUT, A, B, SEL); // UUT = unit under test

initial

$monitorb ("%d out = %b a = %b b = %b sel = %b a1 = %b b1 = %b sel\_n = %b", $time, OUT, A, B, SEL, UUT.A1, UUT.B1, UUT.SEL\_N ); // sets which signals will be displayed in the log

initial begin

$vcdpluson;            // enables graphical viewer

// Tests with SEL = 0

// B = 0

A = 0; B = 0; SEL = 0;

#10 A = 1; B = 0; SEL = 0;

#10 A = 1'bx; B = 0; SEL = 0;

#10 A = 1'bz; B = 0; SEL = 0;

// B = 1

#10 A = 0; B = 1; SEL = 0;

#10 A = 1; B = 1; SEL = 0;

#10 A = 1'bx; B = 1; SEL = 0;

#10 A = 1'bz; B = 1; SEL = 0;

// B = 1'bx

#10 A = 0; B = 1'bx; SEL = 0;

#10 A = 1; B = 1'bx; SEL = 0;

#10 A = 1'bx; B = 1'bx; SEL = 0;

#10 A = 1'bz; B = 1'bx; SEL = 0;

// B = 1'bz

#10 A = 0; B = 1'bz; SEL = 0;

#10 A = 1; B = 1'bz; SEL = 0;

#10 A = 1'bx; B = 1'bz; SEL = 0;

#10 A = 1'bz; B = 1'bz; SEL = 0;

// Tests with SEL = 1

// B = 0

#10 A = 0; B = 0; SEL = 1;

#10 A = 1; B = 0; SEL = 1;

#10 A = 1'bx; B = 0; SEL = 1;

#10 A = 1'bz; B = 0; SEL = 1;

// B = 1

#10 A = 0; B = 1; SEL = 1;

#10 A = 1; B = 1; SEL = 1;

#10 A = 1'bx; B = 1; SEL = 1;

#10 A = 1'bz; B = 1; SEL = 1;

// B = 1'bx

#10 A = 0; B = 1'bx; SEL = 1;

#10 A = 1; B = 1'bx; SEL = 1;

#10 A = 1'bx; B = 1'bx; SEL = 1;

#10 A = 1'bz; B = 1'bx; SEL = 1;

// B = 1'bz

#10 A = 0; B = 1'bz; SEL = 1;

#10 A = 1; B = 1'bz; SEL = 1;

#10 A = 1'bx; B = 1'bz; SEL = 1;

#10 A = 1'bz; B = 1'bz; SEL = 1;

// Tests with SEL = 1'bx (x = unknown logic value)

// B = 0

#10 A = 0; B = 0; SEL = 1'bx;

#10 A = 1; B = 0; SEL = 1'bx;

#10 A = 1'bx; B = 0; SEL = 1'bx;

#10 A = 1'bz; B = 0; SEL = 1'bx;

// B = 1

#10 A = 0; B = 1; SEL = 1'bx;

#10 A = 1; B = 1; SEL = 1'bx;

#10 A = 1'bx; B = 1; SEL = 1'bx;

#10 A = 1'bz; B = 1; SEL = 1'bx;

// B = 1'bx

#10 A = 0; B = 1'bx; SEL = 1'bx;

#10 A = 1; B = 1'bx; SEL = 1'bx;

#10 A = 1'bx; B = 1'bx; SEL = 1'bx;

#10 A = 1'bz; B = 1'bx; SEL = 1'bx;

// B = 1'bz

#10 A = 0; B = 1'bz; SEL = 1'bx;

#10 A = 1; B = 1'bz; SEL = 1'bx;

#10 A = 1'bx; B = 1'bz; SEL = 1'bx;

#10 A = 1'bz; B = 1'bz; SEL = 1'bx;

// Tests with SEL = 1'bz (z = high impedance, floating logic value)

// B = 0

#10 A = 0; B = 0; SEL = 1'bz;

#10 A = 1; B = 0; SEL = 1'bz;

#10 A = 1'bx; B = 0; SEL = 1'bz;

#10 A = 1'bz; B = 0; SEL = 1'bz;

// B = 1

#10 A = 0; B = 1; SEL = 1'bz;

#10 A = 1; B = 1; SEL = 1'bz;

#10 A = 1'bx; B = 1; SEL = 1'bz;

#10 A = 1'bz; B = 1; SEL = 1'bz;

// B = 1'bx

#10 A = 0; B = 1'bx; SEL = 1'bz;

#10 A = 1; B = 1'bx; SEL = 1'bz;

#10 A = 1'bx; B = 1'bx; SEL = 1'bz;

#10 A = 1'bz; B = 1'bx; SEL = 1'bz;

// B = 1'bz

#10 A = 0; B = 1'bz; SEL = 1'bz;

#10 A = 1; B = 1'bz; SEL = 1'bz;

#10 A = 1'bx; B = 1'bz; SEL = 1'bz;

#10 A = 1'bz; B = 1'bz; SEL = 1'bz;

#20 $finish;

end

endmodule

Log:

Chronologic VCS simulator copyright 1991-2017

Contains Synopsys proprietary information.

Compiler version M-2017.03-SP1\_Full64; Runtime version M-2017.03-SP1\_Full64; Feb 1 20:08 2018

VCD+ Writer M-2017.03-SP1\_Full64 Copyright (c) 1991-2017 by Synopsys Inc.

0 out = 0 a = 0 b = 0 sel = 0 a1 = 0 b1 = 0 sel\_n = 1

10 out = 1 a = 1 b = 0 sel = 0 a1 = 1 b1 = 0 sel\_n = 1

20 out = x a = x b = 0 sel = 0 a1 = x b1 = 0 sel\_n = 1

30 out = x a = z b = 0 sel = 0 a1 = x b1 = 0 sel\_n = 1

40 out = 0 a = 0 b = 1 sel = 0 a1 = 0 b1 = 0 sel\_n = 1

50 out = 1 a = 1 b = 1 sel = 0 a1 = 1 b1 = 0 sel\_n = 1

60 out = x a = x b = 1 sel = 0 a1 = x b1 = 0 sel\_n = 1

70 out = x a = z b = 1 sel = 0 a1 = x b1 = 0 sel\_n = 1

80 out = 0 a = 0 b = x sel = 0 a1 = 0 b1 = 0 sel\_n = 1

90 out = 1 a = 1 b = x sel = 0 a1 = 1 b1 = 0 sel\_n = 1

100 out = x a = x b = x sel = 0 a1 = x b1 = 0 sel\_n = 1

110 out = x a = z b = x sel = 0 a1 = x b1 = 0 sel\_n = 1

120 out = 0 a = 0 b = z sel = 0 a1 = 0 b1 = 0 sel\_n = 1

130 out = 1 a = 1 b = z sel = 0 a1 = 1 b1 = 0 sel\_n = 1

140 out = x a = x b = z sel = 0 a1 = x b1 = 0 sel\_n = 1

150 out = x a = z b = z sel = 0 a1 = x b1 = 0 sel\_n = 1

160 out = 0 a = 0 b = 0 sel = 1 a1 = 0 b1 = 0 sel\_n = 0

170 out = 0 a = 1 b = 0 sel = 1 a1 = 0 b1 = 0 sel\_n = 0

180 out = 0 a = x b = 0 sel = 1 a1 = 0 b1 = 0 sel\_n = 0

190 out = 0 a = z b = 0 sel = 1 a1 = 0 b1 = 0 sel\_n = 0

200 out = 1 a = 0 b = 1 sel = 1 a1 = 0 b1 = 1 sel\_n = 0

210 out = 1 a = 1 b = 1 sel = 1 a1 = 0 b1 = 1 sel\_n = 0

220 out = 1 a = x b = 1 sel = 1 a1 = 0 b1 = 1 sel\_n = 0

230 out = 1 a = z b = 1 sel = 1 a1 = 0 b1 = 1 sel\_n = 0

240 out = x a = 0 b = x sel = 1 a1 = 0 b1 = x sel\_n = 0

250 out = x a = 1 b = x sel = 1 a1 = 0 b1 = x sel\_n = 0

260 out = x a = x b = x sel = 1 a1 = 0 b1 = x sel\_n = 0

270 out = x a = z b = x sel = 1 a1 = 0 b1 = x sel\_n = 0

280 out = x a = 0 b = z sel = 1 a1 = 0 b1 = x sel\_n = 0

290 out = x a = 1 b = z sel = 1 a1 = 0 b1 = x sel\_n = 0

300 out = x a = x b = z sel = 1 a1 = 0 b1 = x sel\_n = 0

310 out = x a = z b = z sel = 1 a1 = 0 b1 = x sel\_n = 0

320 out = 0 a = 0 b = 0 sel = x a1 = 0 b1 = 0 sel\_n = x

330 out = x a = 1 b = 0 sel = x a1 = x b1 = 0 sel\_n = x

340 out = x a = x b = 0 sel = x a1 = x b1 = 0 sel\_n = x

350 out = x a = z b = 0 sel = x a1 = x b1 = 0 sel\_n = x

360 out = x a = 0 b = 1 sel = x a1 = 0 b1 = x sel\_n = x

370 out = x a = 1 b = 1 sel = x a1 = x b1 = x sel\_n = x

380 out = x a = x b = 1 sel = x a1 = x b1 = x sel\_n = x

390 out = x a = z b = 1 sel = x a1 = x b1 = x sel\_n = x

400 out = x a = 0 b = x sel = x a1 = 0 b1 = x sel\_n = x

410 out = x a = 1 b = x sel = x a1 = x b1 = x sel\_n = x

420 out = x a = x b = x sel = x a1 = x b1 = x sel\_n = x

430 out = x a = z b = x sel = x a1 = x b1 = x sel\_n = x

440 out = x a = 0 b = z sel = x a1 = 0 b1 = x sel\_n = x

450 out = x a = 1 b = z sel = x a1 = x b1 = x sel\_n = x

460 out = x a = x b = z sel = x a1 = x b1 = x sel\_n = x

470 out = x a = z b = z sel = x a1 = x b1 = x sel\_n = x

480 out = 0 a = 0 b = 0 sel = z a1 = 0 b1 = 0 sel\_n = x

490 out = x a = 1 b = 0 sel = z a1 = x b1 = 0 sel\_n = x

500 out = x a = x b = 0 sel = z a1 = x b1 = 0 sel\_n = x

510 out = x a = z b = 0 sel = z a1 = x b1 = 0 sel\_n = x

520 out = x a = 0 b = 1 sel = z a1 = 0 b1 = x sel\_n = x

530 out = x a = 1 b = 1 sel = z a1 = x b1 = x sel\_n = x

540 out = x a = x b = 1 sel = z a1 = x b1 = x sel\_n = x

550 out = x a = z b = 1 sel = z a1 = x b1 = x sel\_n = x

560 out = x a = 0 b = x sel = z a1 = 0 b1 = x sel\_n = x

570 out = x a = 1 b = x sel = z a1 = x b1 = x sel\_n = x

580 out = x a = x b = x sel = z a1 = x b1 = x sel\_n = x

590 out = x a = z b = x sel = z a1 = x b1 = x sel\_n = x

600 out = x a = 0 b = z sel = z a1 = 0 b1 = x sel\_n = x

610 out = x a = 1 b = z sel = z a1 = x b1 = x sel\_n = x

620 out = x a = x b = z sel = z a1 = x b1 = x sel\_n = x

630 out = x a = z b = z sel = z a1 = x b1 = x sel\_n = x

$finish called from file "TB\_MUX2\_1\_V2.v", line 112.

$finish at simulation time 650

V C S S i m u l a t i o n R e p o r t

Time: 650 ns

CPU Time: 0.230 seconds; Data structure size: 0.0Mb

Thu Feb 1 20:08:45 2018

**Additional Question:**

On pages 9 and 10 of the lab manual, there are two examples of behavioral code for a 2:1

Multiplexer. For the code on page 9, I am not sure if it would work properly. In line 4, it uses a ternary operator in an else statement which I do not fully understand so I do not know for sure if the outputs would be similar. Looking at the code on page 10, it was easy to follow, with nothing I have not seen before, and with following the code, I would say it would perform identically to the gate level design, as the code we used in this lab did.

Besides these points, the code on page 10 is very well written and very well documented. The code on page 9 reuses x and z even though those are the variables used for unknown and floating. It also not documented at all, so the reader would have no way of knowing what variable is what without going through the code.

**Conclusion:**

The main takeaway from this week’s lab is to become familiar with the software and programs that will be used throughout the semester, as well as to lay an early foundation in Verilog and to adhere to the *sine qua non.*