ELEC-4200

Digital System Design

FROM: Jacob Howard

TO: Prof. Ujjwal Guin

DUE DATE: 1/28/21

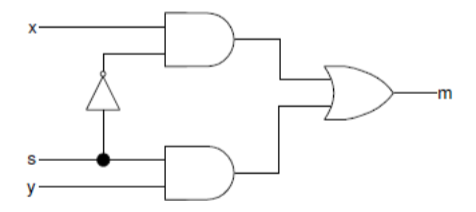
Lab 1

# **Introduction**

This Lab was our first lab requiring code before we came into class and a lab report. The goal of this lab was to familiarize ourselves with different Verilog modeling styles: gate-level, dataflow, and behavioral. We were required to use these different types of modeling to code and program our Nexys A7 boards to perform different logical circuit designs.

# **Task 1**

In Task 1, we were asked to create a 2-to-1 multiplexer using gate-level modeling. A logical circuit of a 2-to-1 multiplexer can be seen in *Figure 1* below. Once we designed a code, we were asked to download the code to the Nexys4 board and verify the functionality. The code can be seen in Code 1 below. After the TA helped fix some errors in the code, the board was functioning properly.

****

*Figure 1*

|  |
| --- |
| `timescale 1ns / 1ps  //////////////////////////////////////////////////////////////////////////////////  module Mux21Gate(  input x,  input y,  input s,  output m  );  wire notS;  wire Int1;  wire Int2;  wire Int3;    not notS (OutNotS, s);  and Int1(And1, OutNotS, x);  and Int2 (And2,s, y);  or Int3 (m, And1, And2);  endmodule |

*Code 1*

# **Task 2**

In Task 2, we were asked to create a 2-bit wide multiplexer using gate-level modeling. Once we did, we were to download the code onto the board to verify functionality. The only difference between this task and *Task 1* is that the inputs are now 2-bits wide. I verified that the board was functioning properly. The code can be seen below in *Code 2*.

|  |
| --- |
| `timescale 1ns / 1ps  //////////////////////////////////////////////////////////////////////////////////  module Mux212BitGate(  input [1:0] x,  input [1:0] y,  input s,  output [1:0] m  );  wire notS;  wire [1:0] Int1;  wire [1:0] Int3;    not notS (OutNotS, s);  and Int1[1:0] (And1, OutNotS, x);  and Int2[1:0] (And2,s, y);  or Int3[1:0] (m, And1, And2);  endmodule |

*Code 2*

# **Task 3**

In Task 3, we were asked to design a 2-bit wide 2-to-1 multiplexer using dataflow modeling. This Task is the same as Task 2 but using a different modeling style of coding. I verified that the code was functioning properly on the board in the lab. The code can be seen below in *Code 3*.

|  |
| --- |
| `timescale 1ns / 1ps  //////////////////////////////////////////////////////////////////////////////////  module Mux212BitDataFlow(  input [1:0] x,  input [1:0] y,  input s,  output [1:0] m  );    wire[1:0] m, x, y;  assign #3 m = (x & ~s) | (y & s);  endmodule |

*Code 3*

# **Task 4**

For Task 4, we were asked to create another 2-bit wide 2-to1 multiplexer, but this time using behavioral modeling style of coding. The code worked as intended and can be seen below in *Code 4*.

|  |
| --- |
| `timescale 1ns / 1ps  //////////////////////////////////////////////////////////////////////////////////  module Mux212BitBehavioral(  input [1:0] x,  input [1:0] y,  input s,  output [1:0] m  );  reg [1:0] m;  always @ (x or y or s)  begin  if (s==0)  m = x;  else  m = y;  end  endmodule |

*Code 4*

# **Task 5**

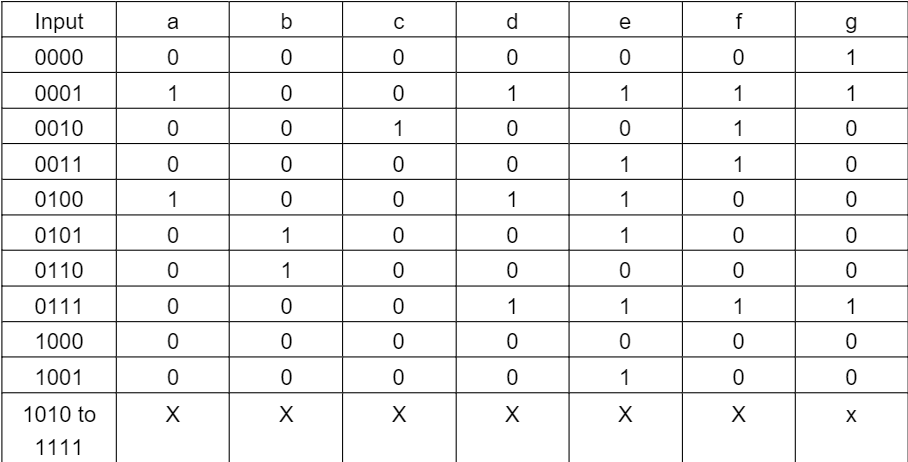
In Task 5, we were asked to model a single-bit 3-to-1 multiplexer in any type of modeling and download it to the board to verify functionality. Sadly, my code was incorrect for this part, and I ran out of time in the lab to fix the code or complete anything else. The code I wrote can be seen below in *Code 5*.

|  |
| --- |
| `timescale 1ns / 1ps  //////////////////////////////////////////////////////////////////////////////////  module Mux31(  input u,  input v,  input w,  input s0,  input s1,  output m  );  and Int1 (Out1, s0, u, v); //getting First And Gate on First Mux  and Int2 (m, Out1, w, s1); //Second and for Mux using first And Output. Makes 3-1 Mux  endmodule |

*Code 3*

# **Model a BCD to 7-Segment Decoder**

In the last task, we were asked to model a BCD to 7-Segment Decoder. This was to get us prepared for Lab 2 material. A BCD to Seven Segment decoder is a combinational logic circuit that accepts a 4-bit binary code and transmits it into a 7-bit binary code, commonly used for 7 segment number displays. The truth table for certain values of the BCD to 7-Segment Decoder can be seen below in *Figure 2.* This was the end of Lab 1.

**

*Figure 2 (x is don’t care)*