Cover Page

Name: Ethan Lingad

Student ID: 862268541

Course Title: CS120A

Lab Section: 22

Lab Title: Lab 03. Combinatorial

Logic Using EDA Playground

Overview:

What I did in this lab was create the BCD-to-7seg decoder. Given some four bit input, we were expected to give a 7 bit output in order to activate the 7 segment display. The conclusion I got from doing the lab was we are able to create a decoder using Verilog. We are able to replicate creating a decoder and that we have to take into account all the possible combinations when creating the decoder using a truth table.

Analysis:

1. First copy and paste the code for the design.sv
2. Then we fill in the rest of the code that we had to provide for making different possible combinations for the given inputs
3. Then we switch over over to creating a test bench
4. The code was provided then we were required to fill in the rest of the code for the multiple tests for given inputs and expected outputs

I used this table but I switched the 0s and 1s.

	x_3	x_2	x_1	x_0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
	1	0	1	0	x	x	x	x	x	x	x
	1	0	1	1	x	x	x	x	x	x	x
	1	1	0	0	x	x	x	x	x	x	x
	1	1	0	1	x	x	x	x	x	x	x
	1	1	1	0	x	x	x	x	x	x	x
	1	1	1	1	x	x	x	x	x	x	x

I also used this table.

Specification

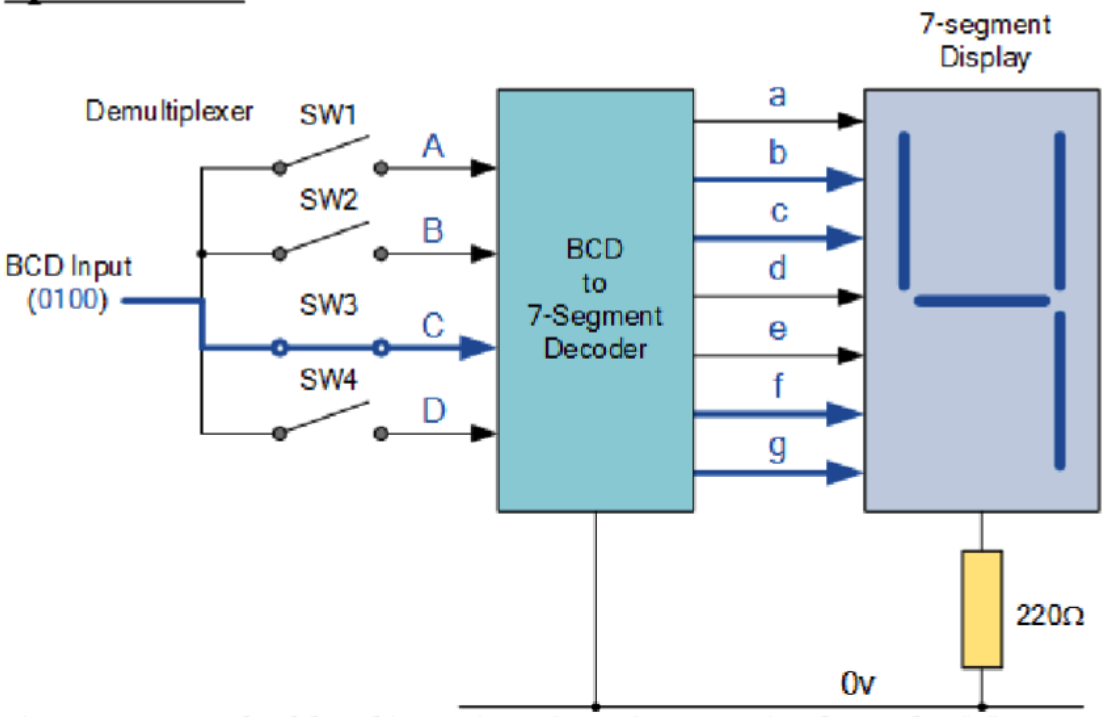


Figure 2: An example of the 4-bit BCD input (0100) representing the number "4".

Records:

Design.sv

```
// Code your design here
`timescale 1ns / 1ps
module bcd_to_7led_bh (
input wire sw0 ,
  // Switches
input wire sw1 ,
input wire sw2 ,
input wire sw3 ,
output reg a ,
 // LED segments
output reg b ,
output reg c ,
output reg d ,
output reg e ,
```

```
output reg f ,
output reg g
);
// Internal wire
wire [3:0] bundle ;
assign bundle = {sw3,sw2,sw1,sw0 } ;
always @(*) begin
// Setting the segments signals (Initialize all to off/1)
a = 1'b1 ;
b = 1'b1 ;
c = 1'b1 ;
d = 1'b1 ;
e = 1'b1 ;
f = 1'b1 ;
g = 1'b1 ;
case ( bundle )
4'b0000 : begin // 0
a = 1'b0 ;
b = 1'b0 ;
c = 1'b0 ;
d = 1'b0 ;
e = 1'b0 ;
f = 1'b0 ;
g = 1'b1 ; //(Don't need to explicitly state that g is off here since
// it is initialized to off already, but it doesn't hurt)
end
4'b0001: begin
a = 1'b1 ;
b = 1'b0 ;
c = 1'b0 ;
d = 1'b1 ;
e = 1'b1 ;
f = 1'b1 ;
g = 1'b1 ;
end

4'b0010: begin
a = 1'b0 ;
b = 1'b0 ;
```

```
c = 1'b1 ;
d = 1'b0 ;
e = 1'b0 ;
f = 1'b1 ;
g = 1'b0 ;
end

4'b0011: begin
a = 1'b0 ;
b = 1'b0 ;
c = 1'b0 ;
d = 1'b0 ;
e = 1'b1 ;
f = 1'b1 ;
g = 1'b0 ;
end

4'b0100: begin
a = 1'b1 ;
b = 1'b0 ;
c = 1'b0 ;
d = 1'b1 ;
e = 1'b1 ;
f = 1'b0 ;
g = 1'b0 ;
end
4'b0101: begin
a = 1'b0 ;
b = 1'b1 ;
c = 1'b0 ;
d = 1'b0 ;
e = 1'b1 ;
f = 1'b0 ;
g = 1'b0 ;
end
4'b0110: begin
a = 1'b0 ;
b = 1'b1 ;
c = 1'b0 ;
```

```
d = 1'b0 ;
e = 1'b0 ;
f = 1'b0 ;
g = 1'b0 ;
end
4'b0111: begin
a = 1'b0 ;
b = 1'b0 ;
c = 1'b0 ;
d = 1'b1 ;
e = 1'b1 ;
f = 1'b1 ;
g = 1'b1 ;
end

4'b1000: begin
a = 1'b0 ;
b = 1'b0 ;
c = 1'b0 ;
d = 1'b0 ;
e = 1'b0 ;
f = 1'b0 ;
g = 1'b0 ;
end
4'b1001: begin
a = 1'b0 ;
b = 1'b0 ;
c = 1'b0 ;
d = 1'b0 ;
e = 1'b1 ;
f = 1'b0 ;
g = 1'b0 ;
end
// Your code goes here for the other 8 numbers (1-9)


endcase
end
endmodule
```

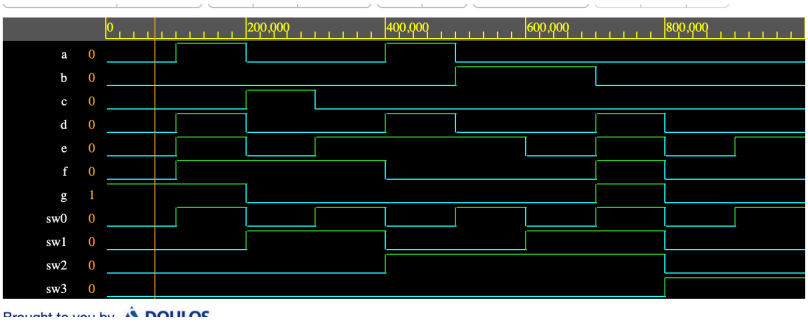
Testbench.sv

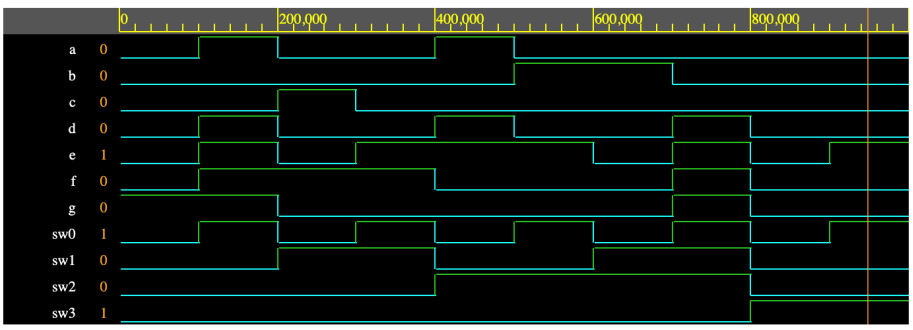
```
// Code your testbench here
// or browse Examples
`timescale 1ns / 1ps
// http://www.electronics-tutorials.ws/combination/comb_6.html
module bcdtoled_tb;
// Inputs
reg sw0;
reg sw1;
reg sw2;
reg sw3;
// Outputs
wire a;
wire b;
wire c;
wire d;
wire e;
wire f;
wire g;
// Instantiate the Unit Under Test (UUT)
// bcdto7led_st uut (
bcd_to_7led_bh uut (
.sw0(sw0),
.sw1(sw1),
.sw2(sw2),
.sw3(sw3),
.a(a),
.b(b),
.c(c),
.d(d),
.e(e),
.f(f),
.g(g)
);
initial begin
$dumpfile("dump.vcd"); $dumpvars;
// Initialize Inputs
sw3 = 0; sw2 = 0; sw1 = 0; sw0 = 0;
```

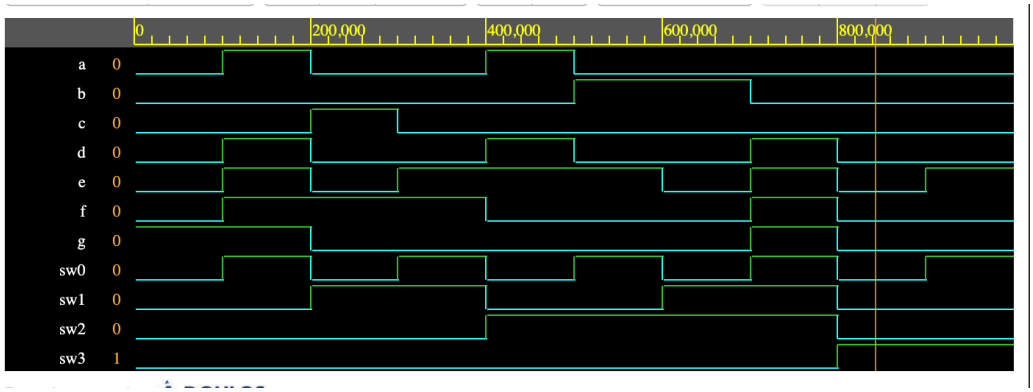
```
#100;
$display("TC10 ");
if ( {a,b,c,d,e,f,g} != 7'b0000001 ) $display ("Result is wrong %b",
{a,b,c,d,e,f,g});
// Your test cases go here (9 left)
sw3 = 0; sw2 = 0; sw1 = 0; sw0 = 1;
#100;
  $display("TC11 ");
                                                  if ( {a,b,c,d,e,f,g}
!= 7'b1001111 ) $display ("Result is wrong %b", {a,b,c,d,e,f,g});
                        sw3 = 0; sw2 = 0; sw1 = 1; sw0 = 0;
#100;
  $display("TC12 ");
  if ( {a,b,c,d,e,f,g} != 7'b0010010 ) $display ("Result is wrong
%b", {a,b,c,d,e,f,g});

sw3 = 0; sw2 = 0; sw1 = 1; sw0 = 1;
#100;
  $display("TC13 ");
  if ( {a,b,c,d,e,f,g} != 7'b0000110 ) $display ("Result is wrong
%b", {a,b,c,d,e,f,g});

sw3 = 0; sw2 = 1; sw1 = 0; sw0 = 0;
#100;
  $display("TC14 ");
  if ( {a,b,c,d,e,f,g} != 7'b1001100 ) $display ("Result is wrong
%b", {a,b,c,d,e,f,g});

sw3 = 0; sw2 = 1; sw1 = 0; sw0 = 1;
#100;
  $display("TC15 ");
  if ( {a,b,c,d,e,f,g} != 7'b0100100 ) $display ("Result is wrong
%b", {a,b,c,d,e,f,g});

sw3 = 0; sw2 = 1; sw1 = 1; sw0 = 0;
#100;
  $display("TC16 ");
  if ( {a,b,c,d,e,f,g} != 7'b0100000 ) $display ("Result is wrong
%b", {a,b,c,d,e,f,g});
```

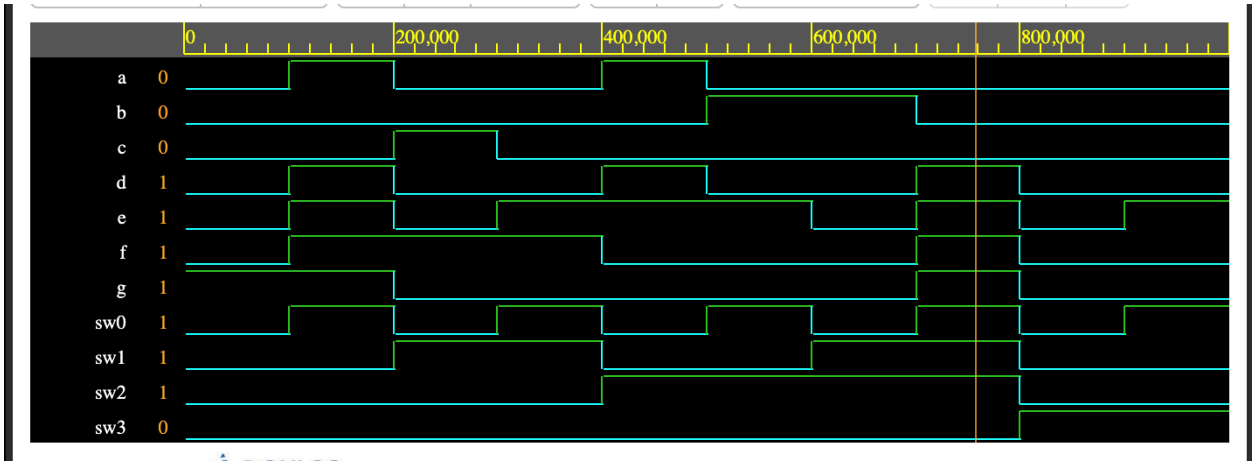
```
sw3 = 0; sw2 = 1; sw1 = 1; sw0 = 1;
#100;
  $display("TC17 ");
  if ( {a,b,c,d,e,f,g} != 7'b0001111 ) $display ("Result is wrong
%b", {a,b,c,d,e,f,g});

sw3 = 1; sw2 = 0; sw1 = 0; sw0 = 0;
#100;
  $display("TC18 ");
  if ( {a,b,c,d,e,f,g} != 7'b0000000 ) $display ("Result is wrong
%b", {a,b,c,d,e,f,g});

sw3 = 1; sw2 = 0; sw1 = 0; sw0 = 1;
#100;
  $display("TC19 ");
  if ( {a,b,c,d,e,f,g} != 7'b0000100 ) $display ("Result is wrong
%b", {a,b,c,d,e,f,g});
end
endmodule
```

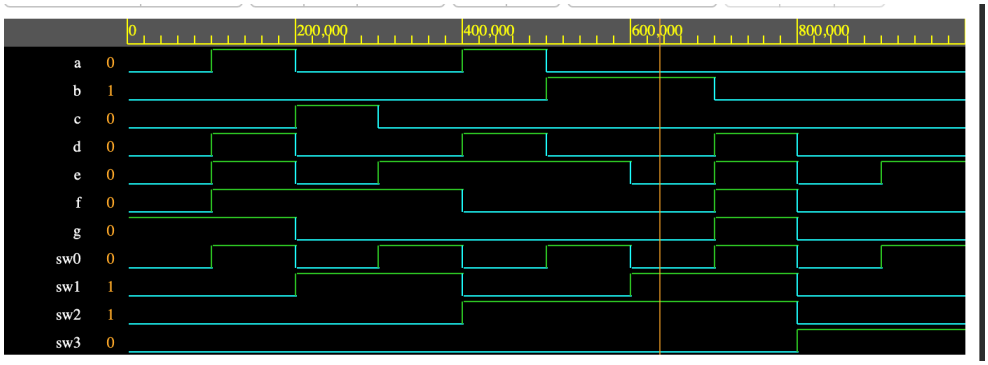
Waveforms for Each Input and Output:

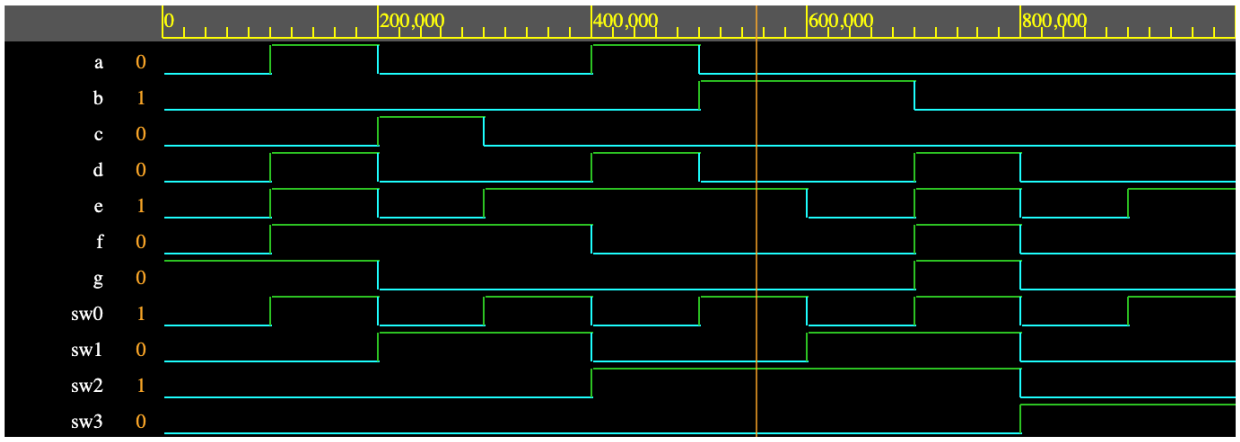


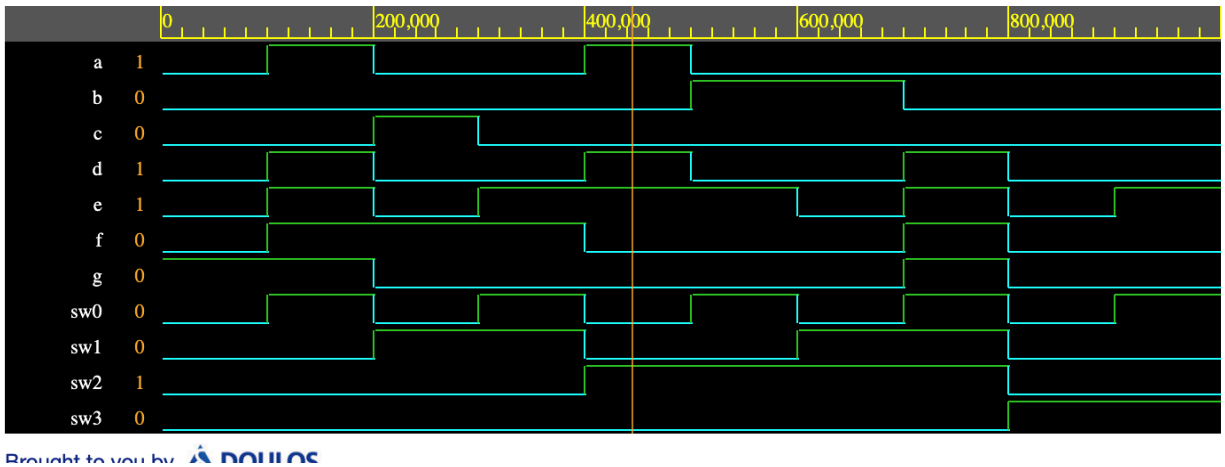


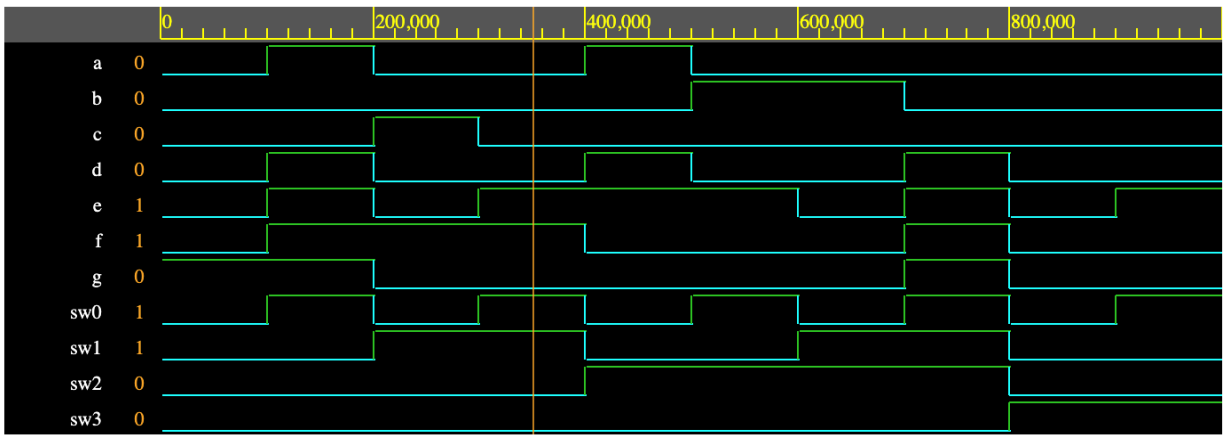

0
200,000
400,000
600,000
800,000
a 0
b 0
c 0
d 0
e 0
f 0
g 0
sw0 0
sw1 0
sw2 0
sw3 1

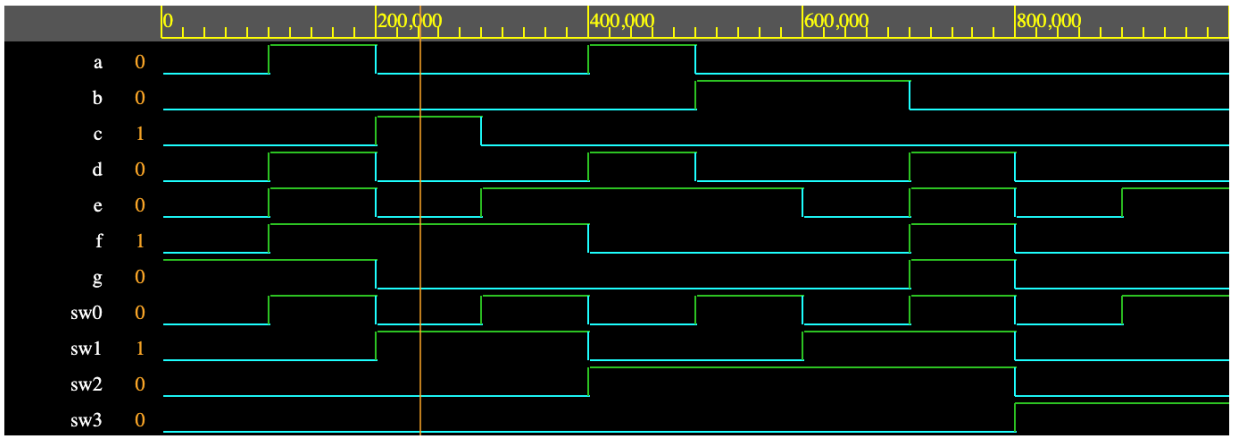

0
200,000
400,000
600,000
800,000
a 0
b 0
c 0
d 1
e 1
f 1
g 1
sw0 1
sw1 1
sw2 1
sw3 0

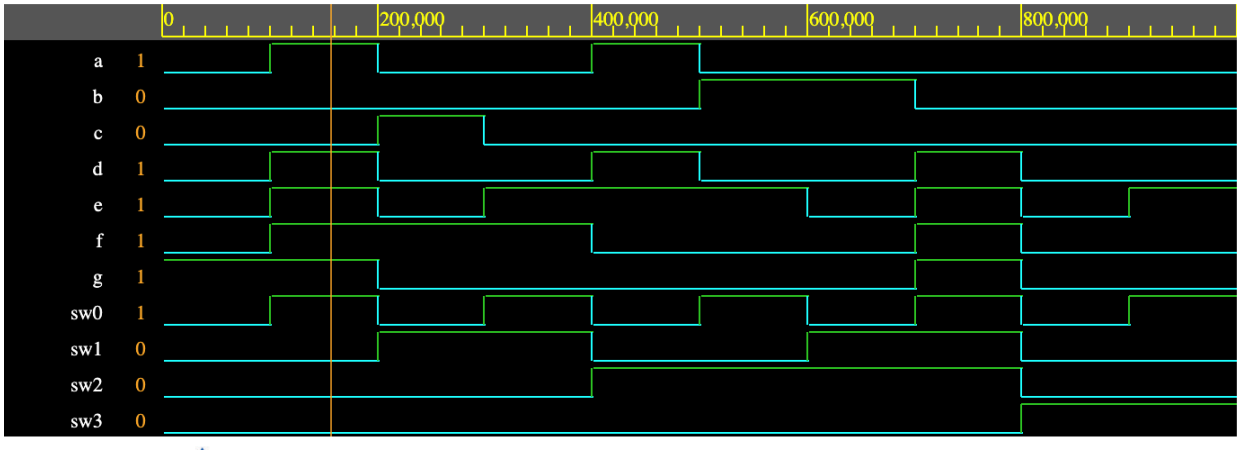

0
200,000
400,000
600,000
800,000
a 0
b 1
c 0
d 0
e 0
f 0
g 0
sw0 0
sw1 1
sw2 1
sw3 0











Discussion:

The system does work, since for each input, the output gives the expected result that we want. For example when sw3 = 0, sw2 = 0, sw1 = 0, and sw0 = 1, then the expected output should be a = 1, b = 0, c = 0, d = 1, e = 1, f =1, and g= 1, which is does give the correct output. I did not stumble upon any problems when implementing the code. I think the way the code runs is sufficient enough according to prior knowledge. I don't think there can be any other way to optimize the code unless stated otherwise.

Conclusion:

The purpose of this lab was to learn how to implement a decoder in a broader scope, but more specifically a BCD-to7seg decoder. We have to learn how to be able to construct a decoder and

to understand how to be able to implement it in Verilog. By using the truth table for the expected inputs and outputs, it guided me to be able to create the decoder in Verilog.

Questions:

There were no specific questions in the lab.