Lab 5 Report

Objective:

The objective of this lab was to learn more about implementing finite state machines in Verilog. Specifically, we were tasked with expanding on our previous lab design to create a 16-bit counter that can count either up or down, with the counter value being displayed on the 7-segment display like in the last lab. Furthermore, the counter must be set to the value input on the FPGA's 8 switches at the push of a button.

Methodology:

To create this design, several sub components had to be created first. Due to how similar this lab is to the previous one, there are several components that can simply be copied over from my design for lab 4. These components are the seven segment decoder, the display control, the clock dividers, and the debouncer.

After copying all of these modules over from lab 4, I moved on to updating the previous lab's counter16 module to work according to the new requirements. This was not too difficult, as all it meant was adding three inputs: one to control the counting direction, one to tell the design when to change the value to the value input on the 8 switches, and one to feed the value input from the switches to the design. This turned out to be fairly simple to implement, as all I had to add was an if statement checking the count direction input to see what direction to increment/decrement, and an if statement checking if the counter should be changed to the value input from the switches. The reset functionality was also updated to not only reset the counter to 0, but also to reset the counting direction to count upwards.

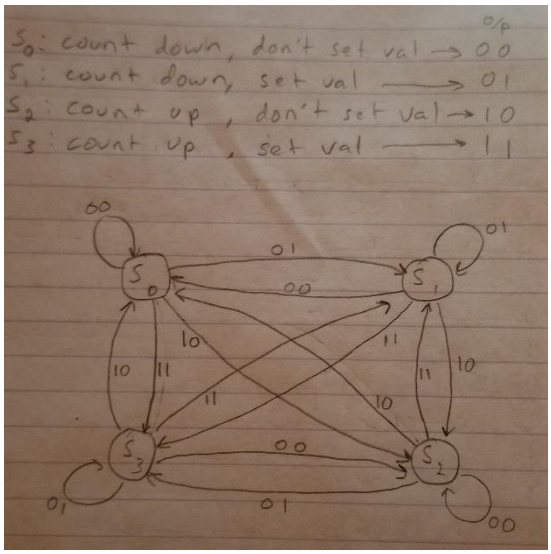
The last component that needed to be created was the actual finite state machine. The state machines requirements were that it needed to accept two inputs, direction_toggle_button and set_value_button, and have two outputs, count_direction and set_value_counter. I decided to use a moore machine, as I wanted each state to correspond with a specific state. Since there are two outputs, the state has two bits, making four states in total. These states are:

00: count down, don't set value counter
01: count down, set value counter
10: count up, don't set value counter
11: count up, set value counter

So count_direction equals 0 for counting down, 1 for counting up, and set_value_counter equals 1 when the counter should be set to the switch value input. The inputs, direction_toggle_button and set_value_button, are represented by A and B respectively, with the state machine input being in AB=00 format. So for AB=10, the count direction should be toggled, and the counter should not be set to the switch value input. Basically, whenever A=1, count_direction should be toggled from 0 to 1 or 1 to 0, and set_value_counter should always be equal to B. The outputs count_direction and set_value_counter, are represented by x and y respectively. These representations are only for the truth table and state diagrams below, which show the functionality of the state machine.

Alexander Salmi
U04201274

Current State		Input		Next State (output)	
x	y	A	B	x	y
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	1	0	1
0	1	1	0	1	0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	1	0	1



Based on the truth table above, we can simplify the outputs into the following functions:

x' = x XOR A
y' = B

After designing the state machine and the rest of the components, I moved on to combining them all into the full counter module. This was done according to the code below:

```
module Lab5(input clk, input reset, input set_value_button, input [7:0] value, input
direction_toggle_button, output [3:0] digit_select, output [6:0] seven);
        wire clk_1kHz, clk_1Hz, set_val_button_debounce, direction_toggle_button_debounce,
direction, set_val;
        wire [3:0] seven_seg_in;
        wire [15:0] counter16_out;

        clock_divider_1kHz g1(.clk(clk), .clk_1kHz(clk_1kHz));
        clock_divider_1Hz g2(.clk(clk), .clk_1Hz(clk_1Hz));
        debouncer g3(.clk(clk), .reset(reset), .button_in(direction_toggle_button),
.button_out(direction_toggle_button_debounce));
        debouncer g4(.clk(clk), .reset(reset), .button_in(set_value_button),
.button_out(set_val_button_debounce));
        control_FSM g5(.direction_toggle_button(direction_toggle_button_debounce),
.set_value_button(set_val_button_debounce), .clk(clk), .reset(reset), .count_direction(direction),
.set_value_counter(set_val));
        counter16 g6(.clk(clk_1Hz), .reset(reset), .increment(1), .count_direction(direction),
.set_value(set_val), .value(value), .count(counter16_out));
        display_control g7(.clk(clk_1kHz), .num(counter16_out), .digit_select(digit_select),
.seven(seven_seg_in));
```

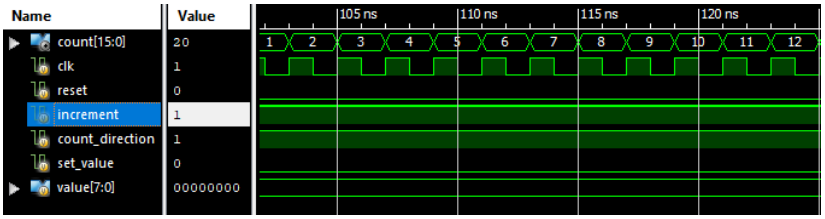
```
        seven_segment_decoder g8(.in(seven_seg_in), .out(seven));
endmodule
```

Observations:

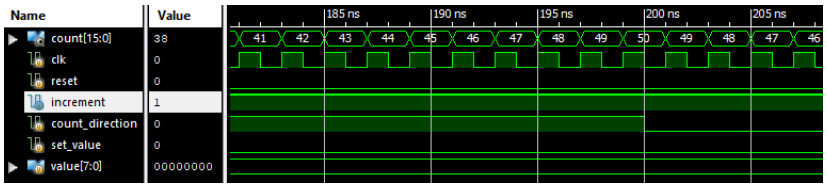
Overall, everything in this lab worked as intended. All of the modules from the previous lab work just as they did for lab 4. Their function is shown in the timing diagrams from my lab 4 report.

As for the updated counter16 module, all of its functions work correctly. This is shown in the following timing diagrams:

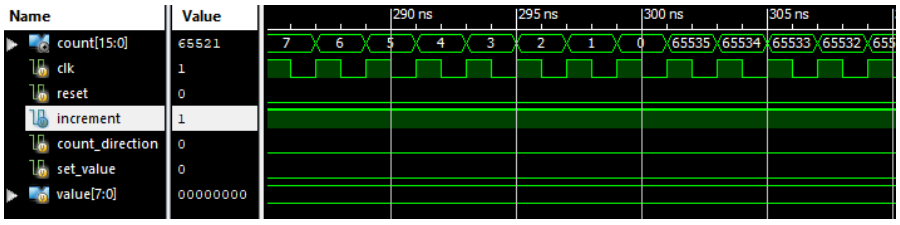
Counting:



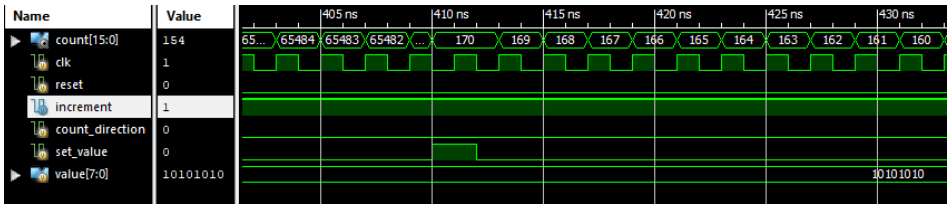
Switching direction:



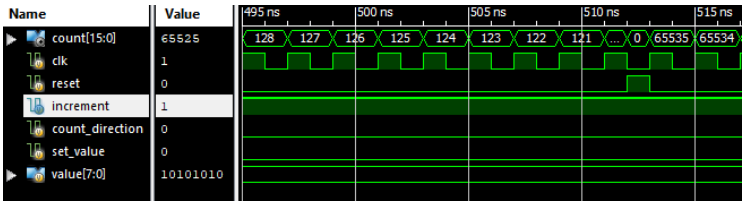
Wrapping around:



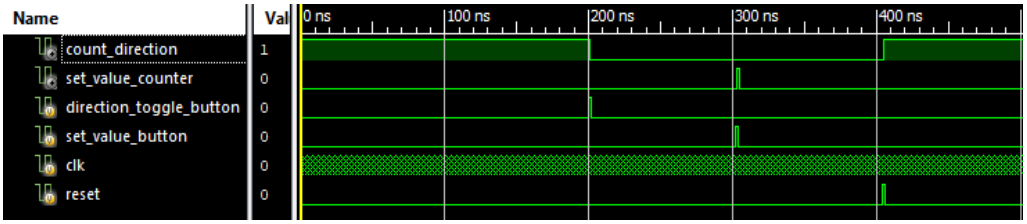
Set value:



Reset:



The control FSM also works correctly, as shown in the timing diagram below:



Conclusion:

Overall, this lab went very well. I learned how to successfully design a finite state machine in Verilog, as well as became more familiarized with designing finite state machines in general.

Code:

Counter16 Module:

```
module counter16(
   input clk,
   input reset,
   input increment,
   input count_direction,
   input set_value,
   input [7:0] value,
   output reg [15:0] count = 16'b0000000000000000
   );

                always@(posedge clk or posedge reset or posedge set_value)begin
                if(reset == 1) begin
                        count <= 16'b0000000000000000;
                end else begin
                        if(set_value == 1)begin
                                count <= value;
                        end else begin
                                if(increment == 1)begin
                                        if(count_direction == 1)begin
                                                count <= count + 16'b0000000000000001;
                                        end else begin
                                                count <= count - 16'b0000000000000001;
                                        end
                                end
                        end
                end
        end //always
endmodule
```

Counter16 Testbench:

```
module counter16_TB;
	// Inputs
	reg clk;
	reg reset;
	reg increment;
	reg count_direction;
	reg set_value;
	reg [7:0] value;

	// Outputs
	wire [15:0] count;
	// Instantiate the Unit Under Test (UUT)
	counter16 uut (
		.clk(clk),
		.reset(reset),
		.increment(increment),
		.count_direction(count_direction),
		.set_value(set_value),
		.value(value),
		.count(count)
	);

	initial begin
		// Initialize Inputs
		clk = 0;
		reset = 0;
		increment = 0;
		count_direction = 1;
		set_value = 0;
		value = 0;
		// Wait 100 ns for global reset to finish
		#100 increment = 1;
		#100 count_direction = 0;
		#200 value = 8'b10101010;
		#10 set_value = 1;
		#2 set_value = 0;
		#100 reset = 1;
		#1 reset = 0;
	end
	always begin
		#1 clk = ~clk;
	end
endmodule
```

Control_FSM Module:

```
module control_FSM(
   input direction_toggle_button,
   input set_value_button,
   input clk,
   input reset,
   output reg count_direction = 1,
   output reg set_value_counter = 0
   );

        always@(posedge clk)begin
                if(reset==1)begin
                        count_direction <= 1;
                        set_value_counter <= 0;
                end else begin
                        if(direction_toggle_button == 1)begin
                                count_direction <= ~count_direction;
                        end
                        set_value_counter <= set_value_button;
                end
        end
endmodule
```

Control_FSM Testbench:

```
module control_FSM_TB;
	// Inputs
	reg direction_toggle_button;
	reg set_value_button;
	reg clk;
	reg reset;
	// Outputs
	wire count_direction;
	wire set_value_counter;
	// Instantiate the Unit Under Test (UUT)
	control_FSM uut (
		.direction_toggle_button(direction_toggle_button),
		.set_value_button(set_value_button),
		.clk(clk),
		.reset(reset),
		.count_direction(count_direction),
		.set_value_counter(set_value_counter)
	);
	initial begin
		// Initialize Inputs
		direction_toggle_button = 0;
		set_value_button = 0;
		clk = 0;
		reset = 0;
		// Wait 100 ns for global reset to finish
		#100;
		#100;
		direction_toggle_button = 1;
		#2;
		direction_toggle_button = 0;
		#100;
		set_value_button = 1;
		#2;
		set_value_button = 0;
		#100;
		reset = 1;
		#2
		reset = 2;
	end
	always begin
		#1 clk = ~clk;
	end
endmodule
```

Alexander Salmi
U04201274

UCF File:

```
NET "clk" LOC = v10;
NET "reset" LOC = B8;
NET "direction_toggle_button" LOC = D9;
NET "set_value_button" LOC = C4;
NET "value[0]" LOC = T10;
NET "value[1]" LOC = T9;
NET "value[2]" LOC = V9;
NET "value[3]" LOC = M8;
NET "value[4]" LOC = N8;
NET "value[5]" LOC = U8;
NET "value[6]" LOC = V8;
NET "value[7]" LOC = T5;
NET "digit_select[0]" LOC = N16;
NET "digit_select[1]" LOC = N15;
NET "digit_select[2]" LOC = P18;
NET "digit_select[3]" LOC = P17;
NET "seven[6]" LOC = T17;
NET "seven[5]" LOC = T18;
NET "seven[4]" LOC = U17;
NET "seven[3]" LOC = U18;
NET "seven[2]" LOC = M14;
NET "seven[1]" LOC = N14;
NET "seven[0]" LOC = L14;
```

Lab5 Module

```
module Lab5(
   input clk,
   input reset,
   input set_value_button,
   input [7:0] value,
   input direction_toggle_button,
   output [3:0] digit_select,
   output [6:0] seven
   );
        wire clk_1kHz, clk_1Hz, set_val_button_debounce, direction_toggle_button_debounce,
direction, set_val;
        wire [3:0] seven_seg_in;
        wire [15:0] counter16_out;

        clock_divider_1kHz g1(.clk(clk), .clk_1kHz(clk_1kHz));
        clock_divider_1Hz g2(.clk(clk), .clk_1Hz(clk_1Hz));
        debouncer g3(.clk(clk), .reset(reset), .button_in(direction_toggle_button),
.button_out(direction_toggle_button_debounce));
        debouncer g4(.clk(clk), .reset(reset), .button_in(set_value_button),
.button_out(set_val_button_debounce));
        control_FSM g5(.direction_toggle_button(direction_toggle_button_debounce),
.set_value_button(set_val_button_debounce), .clk(clk), .reset(reset), .count_direction(direction),
.set_value_counter(set_val));
        counter16 g6(.clk(clk_1Hz), .reset(reset), .increment(1), .count_direction(direction),
.set_value(set_val), .value(value), .count(counter16_out));
        display_control g7(.clk(clk_1kHz), .num(counter16_out), .digit_select(digit_select),
.seven(seven_seg_in));
        seven_segment_decoder g8(.in(seven_seg_in), .out(seven));
endmodule
```