Lab 10 – Pulse Distancing Part 2, Designing a Controller

Name: Von Kaukeano *Date*: 11/13/18

Course: Digital Circuit Design Lab – ECE 2613 *Section #*: 001

Summary/Abstract

Using the detailed specifications, we created a finite state machine which is a controller for the data path. There are four inputs, the clock, reset signal, start signal, and the four switches that provide the four- bits of data. The main input is a pulse start signal and a second called strt which controls the start of the finite state machine. The dbounce interval time is generated from a divide by 1,000,000 divider module from a previous lab which creates pulses separated by .01 seconds. There are three output signals, the start button pulse, the four-bit data value, and tc_modulator signal from the data path module. We used the finite state diagram to create the Verilog in order to succeed in this lab.

Introduction

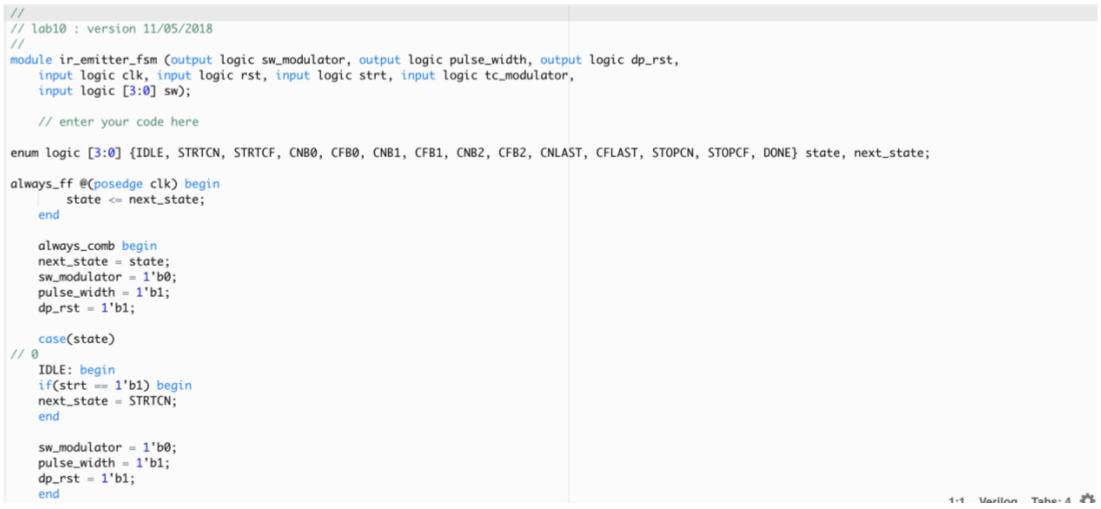
In this lab we designed a finite state machine which is the controller for the data path that we created the last lab. The controller will wait for a button push and then send the control signals that will transmit up to four bits of data to an IR emitter. The four-bit data pattern will be defined by four switches on our Basys 3 board. This lab uses a few modules created in the past labs in order to build our controller.

Procedure

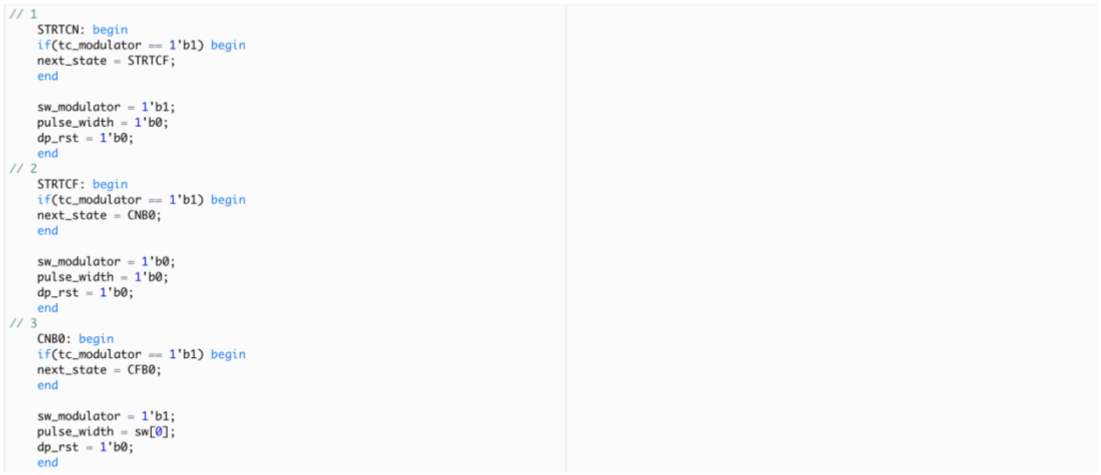
Using the specifications in the finite state machine diagram, we created a case statement in Verilog to step through each state necessary to build our controller. Starting with idle it activates the reset signal on the data path module and the pulse width to long. The next two states turn the carrier on for long period and off for short period. Next eight are carrier off and on for bit zero, one, two, three. Finally, back to carrier on then off and a done state which it stays in this state until reset.

Results

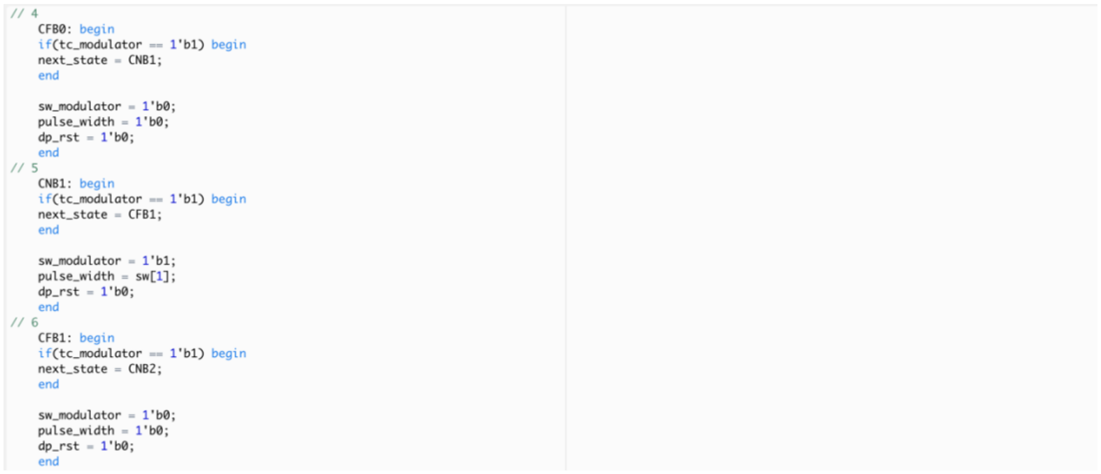
Design Code



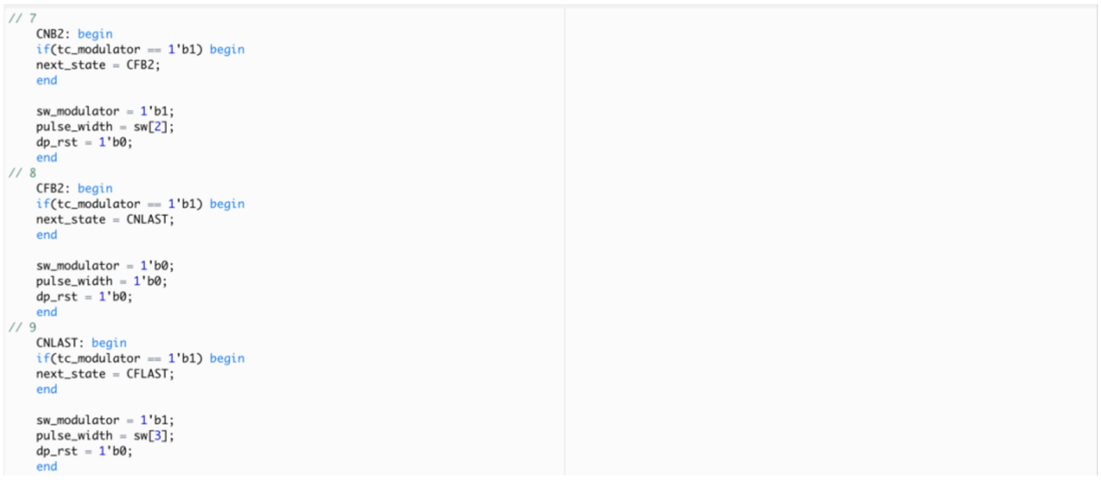
```
//
// lab10 : version 11/05/2018
//
module ir_emitter_fsm (output logic sw_modulator, output logic pulse_width, output logic dp_rst,
    input logic clk, input logic rst, input logic strt, input logic tc_modulator,
    input logic [3:0] sw);

    // enter your code here

enum logic [3:0] {IDLE, STRTCN, STRTCF, CNB0, CFB0, CNB1, CFB1, CNB2, CFB2, CNLAST, CFLAST, STOPCN, STOPCF, DONE} state, next_state;

always_ff @(posedge clk) begin
        state <= next_state;
    end

    always_comb begin
    next_state = state;
    sw_modulator = 1'b0;
    pulse_width = 1'b1;
    dp_rst = 1'b1;

    case(state)
// 0
    IDLE: begin
    if(strt == 1'b1) begin
    next_state = STRTCN;
    end

    sw_modulator = 1'b0;
    pulse_width = 1'b1;
    dp_rst = 1'b1;
    end
```



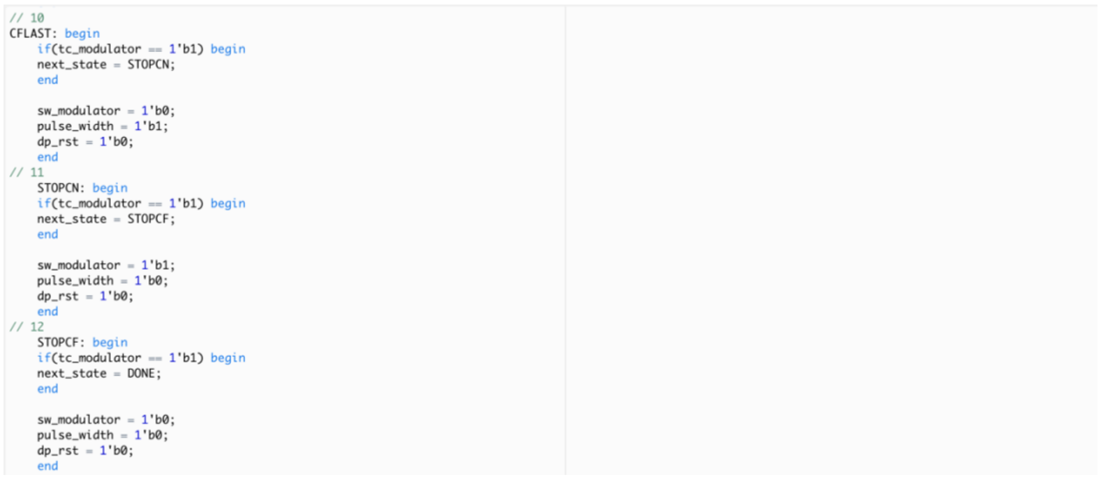
```
// 1
    STRTCN: begin
    if(tc_modulator == 1'b1) begin
    next_state = STRTCF;
    end

    sw_modulator = 1'b1;
    pulse_width = 1'b0;
    dp_rst = 1'b0;
    end
// 2
    STRTCF: begin
    if(tc_modulator == 1'b1) begin
    next_state = CNB0;
    end

    sw_modulator = 1'b0;
    pulse_width = 1'b0;
    dp_rst = 1'b0;
    end
// 3
    CNB0: begin
    if(tc_modulator == 1'b1) begin
    next_state = CFB0;
    end

    sw_modulator = 1'b1;
    pulse_width = sw[0];
    dp_rst = 1'b0;
    end
```



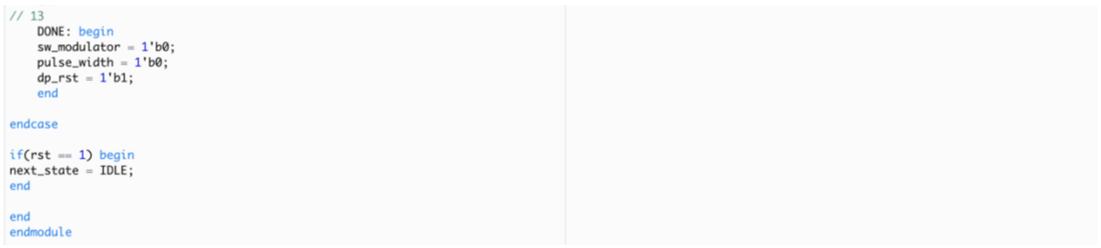
```
// 4
    CFB0: begin
    if(tc_modulator == 1'b1) begin
    next_state = CNB1;
    end

    sw_modulator = 1'b0;
    pulse_width = 1'b0;
    dp_rst = 1'b0;
    end
// 5
    CNB1: begin
    if(tc_modulator == 1'b1) begin
    next_state = CFB1;
    end

    sw_modulator = 1'b1;
    pulse_width = sw[1];
    dp_rst = 1'b0;
    end
// 6
    CFB1: begin
    if(tc_modulator == 1'b1) begin
    next_state = CNB2;
    end

    sw_modulator = 1'b0;
    pulse_width = 1'b0;
    dp_rst = 1'b0;
    end
```



```
// 7
    CNB2: begin
    if(tc_modulator == 1'b1) begin
    next_state = CFB2;
    end

    sw_modulator = 1'b1;
    pulse_width = sw[2];
    dp_rst = 1'b0;
    end
// 8
    CFB2: begin
    if(tc_modulator == 1'b1) begin
    next_state = CNLAST;
    end

    sw_modulator = 1'b0;
    pulse_width = 1'b0;
    dp_rst = 1'b0;
    end
// 9
    CNLAST: begin
    if(tc_modulator == 1'b1) begin
    next_state = CFLAST;
    end

    sw_modulator = 1'b1;
    pulse_width = sw[3];
    dp_rst = 1'b0;
    end
```



```
// 10
CFLAST: begin
    if(tc_modulator == 1'b1) begin
    next_state = STOPCN;
    end

    sw_modulator = 1'b0;
    pulse_width = 1'b1;
    dp_rst = 1'b0;
    end
// 11
    STOPCN: begin
    if(tc_modulator == 1'b1) begin
    next_state = STOPCF;
    end

    sw_modulator = 1'b1;
    pulse_width = 1'b0;
    dp_rst = 1'b0;
    end
// 12
    STOPCF: begin
    if(tc_modulator == 1'b1) begin
    next_state = DONE;
    end

    sw_modulator = 1'b0;
    pulse_width = 1'b0;
    dp_rst = 1'b0;
    end
```



```
// 13
    DONE: begin
    sw_modulator = 1'b0;
    pulse_width = 1'b0;
    dp_rst = 1'b1;
    end

endcase

if(rst == 1) begin
next_state = IDLE;
end

end
endmodule
```

Simulation Results

```
source xsim.dir/work.tb_ir_emitter_top/xsim_script.tcl
# xsim {work.tb_ir_emitter_top} -autoloadwcfg -tclbatch {tb_ir_emitter_top.tcl} -onerror quit
Vivado Simulator 2018.2
Time resolution is 1 ps
source tb_ir_emitter_top.tcl
## run 35000us
Modulation delta time: 20000000 ns
Modulation delta time: 605360 ns
Modulation delta time: 605360 ns
Modulation delta time: 605360 ns
Modulation delta time: 1710800 ns
Modulation delta time: 1710800 ns
Simulation complete!!!
$finish called at time : 33001120 ns : File "/home/tuh42003/2613_2018f/lab10/tb_ir_emitter_top.sv" Line 67
run: Time (s): cpu = 00:00:00.03 ; elapsed = 00:00:20 . Memory (MB): peak = 1359.285 ; gain = 0.000 ; free physical = 4317 ; free virtual = 13263
## set systemTime [clock seconds]
## puts "----Simulation date/time: [clock format $systemTime -format %D] [clock format $systemTime -format %H:%M:%S]"
----Simulation date/time: 11/06/2018 13:43:31
## exit
INFO: [Common 17-206] Exiting xsim at Tue Nov  6 13:43:35 2018...
Compressing vcd file to lxt2 file. ...
```

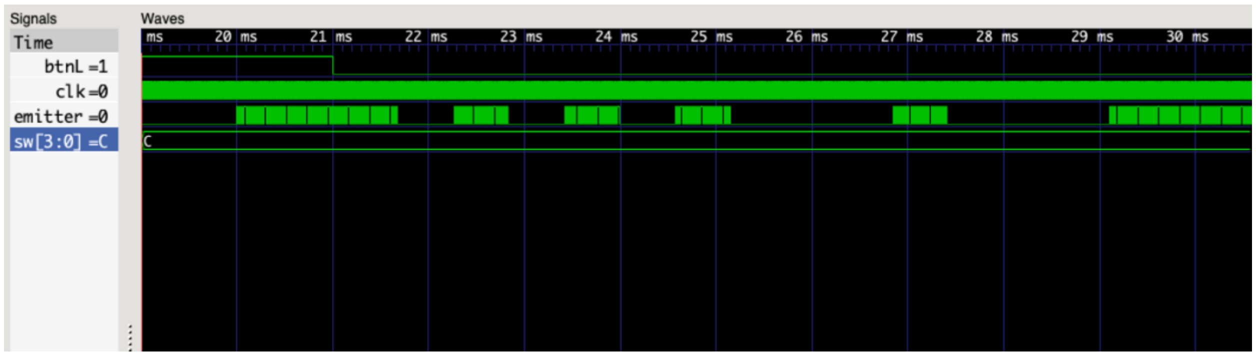
Hardware Implementation

My design was demonstrated to Catherine on the afternoon of 11/6 during the lab period.

Conclusion

This lab is a great representation of what digital circuit design can be used in real application. Using the prior labs is a great way to build what we learned and can still be used to create a finite state machine controller which are used in everyday life. This is interesting to see how complicated of a design can be created by using the knowledge of past labs in the later ones.

Appendix


Signals
Waves
Time
btnL =1
clk =0
emitter =0
sw[3:0] =C
ms
20 ms
21 ms
22 ms
23 ms
24 ms
25 ms
26 ms
27 ms
28 ms
29 ms
30 ms
C