Lab 7 – Counters/Dividers

Name: Von Kaukeano *Date*: 10/23/18

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

Summary/Abstract

The lab consisted of two different modules, a parameterized module which was the divider and the ir emitter module. The divider module takes an input and counts from n to 0 and will output an impulse at tc being 1 when count is 0, thus dividing the 100Mhz internal clock by n + 1. The second module is a series of instantiations of the divider module with four implementations using the same divider design, but different BIT_SIZE parameters to create the required 38 kHz signal. After instantiating them an AND gate combined the signals creating the output signal emitter out.

Introduction

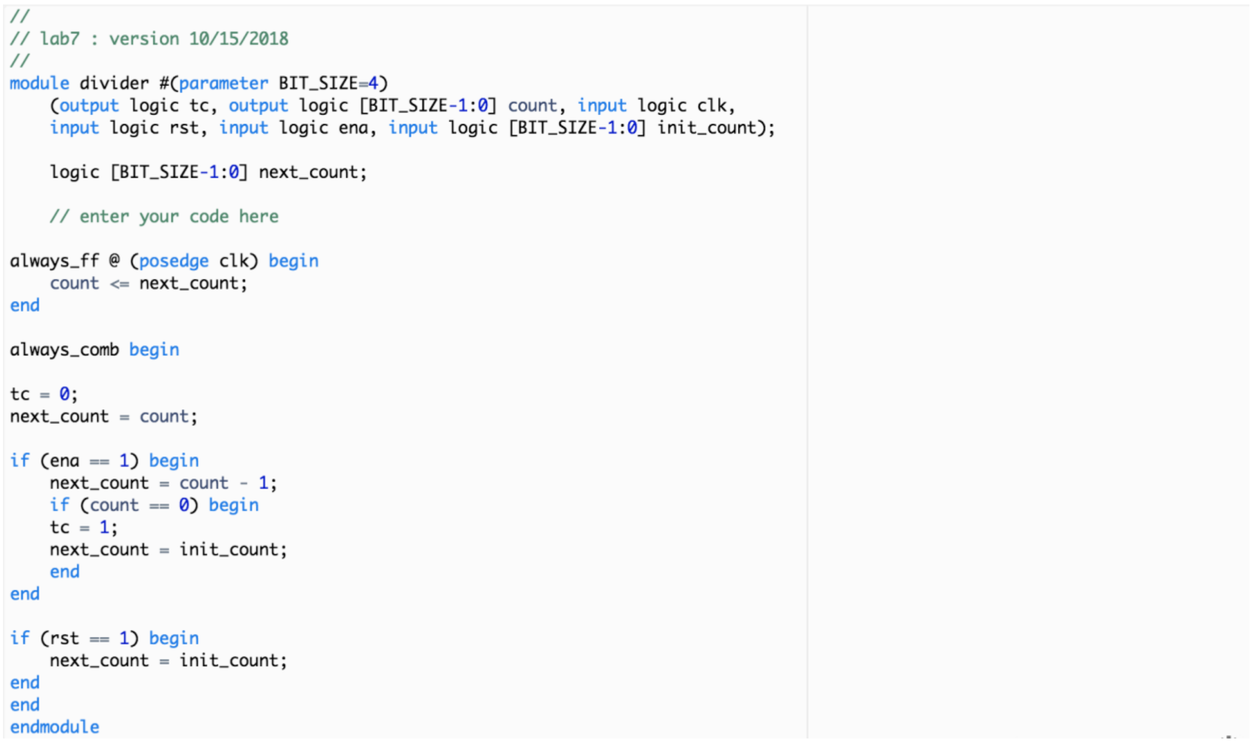
In this lab we designed circuit that will modulate a signal to drive an IR emitter with characteristics that match a particular IR receiver module. We created a design to create signals with two frequencies which are a carrier and a modulation frequency. The carrier frequency is the fundamental frequency and the modulation frequency defines how this carrier wave is turned on and off. In our design the signals with these two frequencies are logically AND'ed and resultant output appears to be bursts of carrier frequency signals with the burst frequency being the modulation frequency.

Procedure

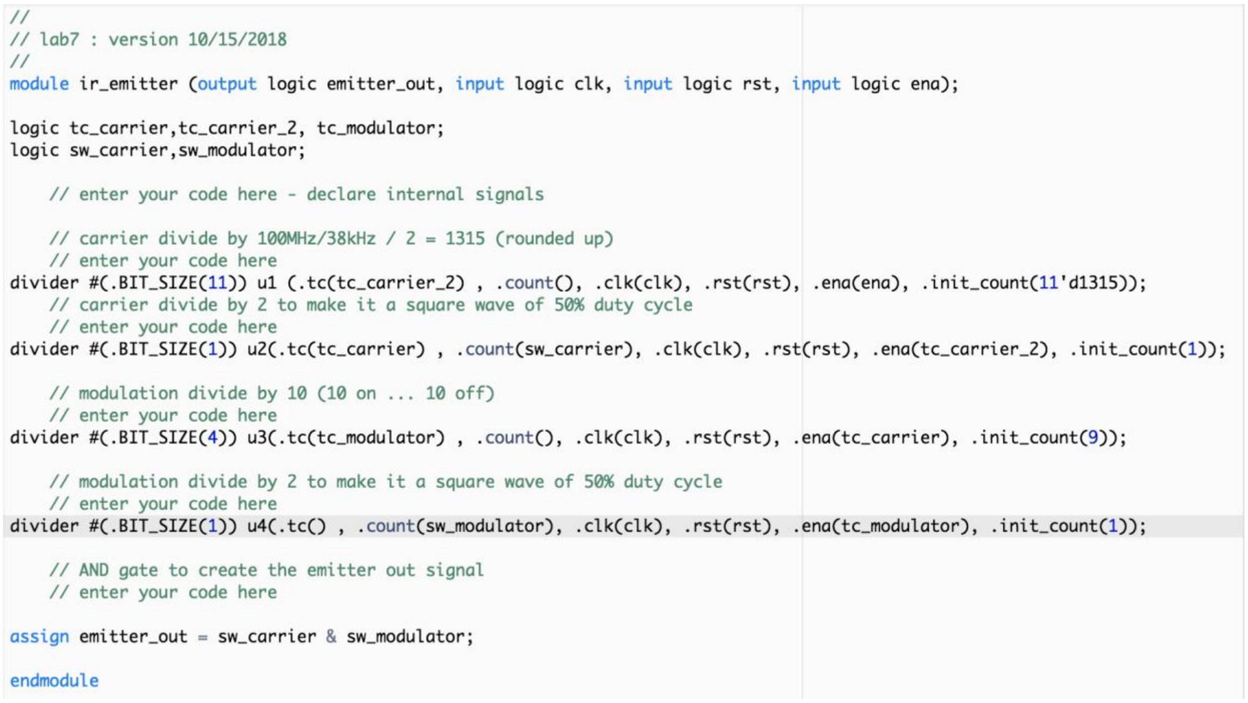
We designed two separate modules. The divider module which we learned how to design in class. For the second module, the first instance was designed to generate a 76 kHz pulse train. The signal from the first will connect to the enable input of another divider to create a square wave by dividing by 2. The third created the 10 cycles on and 10 cycles off modulation. The final divider generates the modulation square wave signal by toggling from high to low for groups of 10 cycles. After creating these modules, we AND gated the outputs to create a final output emitter out.

Results

Design Code

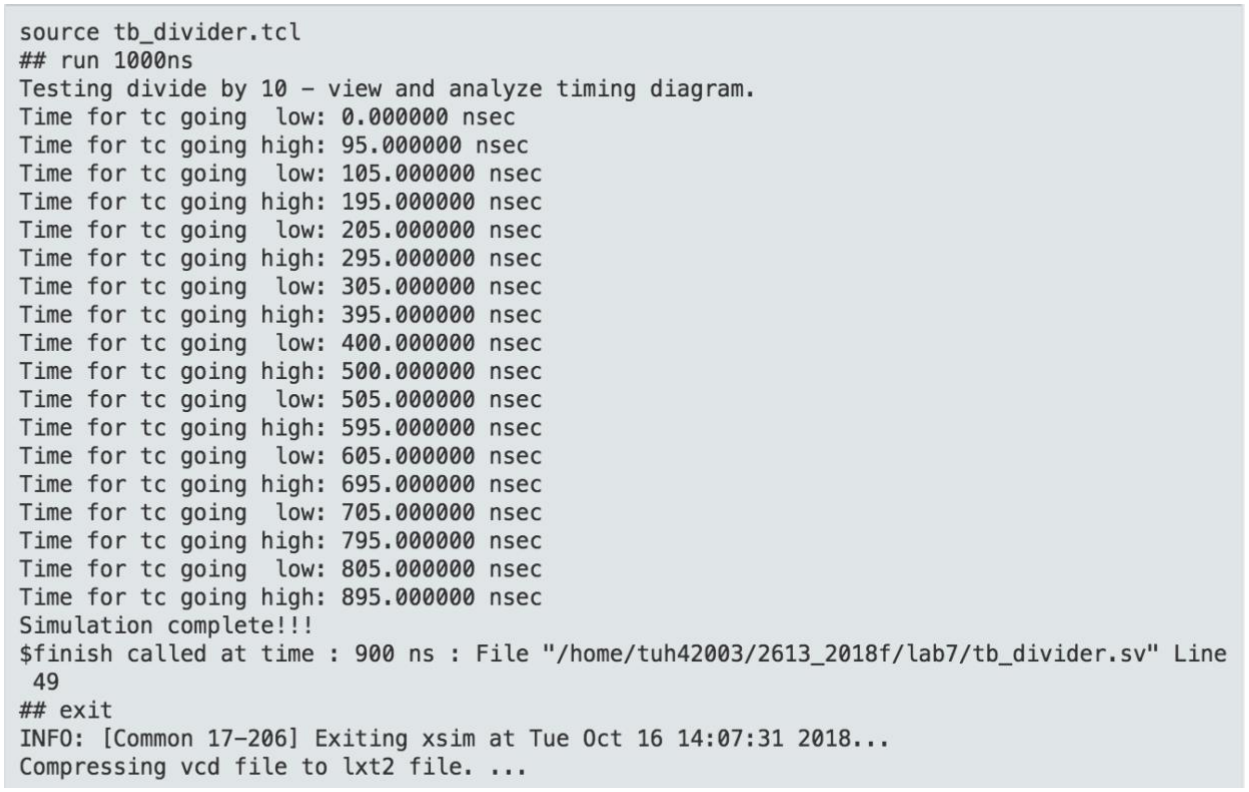


```
//
// lab7 : version 10/15/2018
//
module divider #(parameter BIT_SIZE=4)
    (output logic tc, output logic [BIT_SIZE-1:0] count, input logic clk,
    input logic rst, input logic ena, input logic [BIT_SIZE-1:0] init_count);

    logic [BIT_SIZE-1:0] next_count;

    // enter your code here

always_ff @ (posedge clk) begin
    count <= next_count;
end

always_comb begin

tc = 0;
next_count = count;

if (ena == 1) begin
    next_count = count - 1;
    if (count == 0) begin
    tc = 1;
    next_count = init_count;
    end
end

if (rst == 1) begin
    next_count = init_count;
end
end
endmodule
```

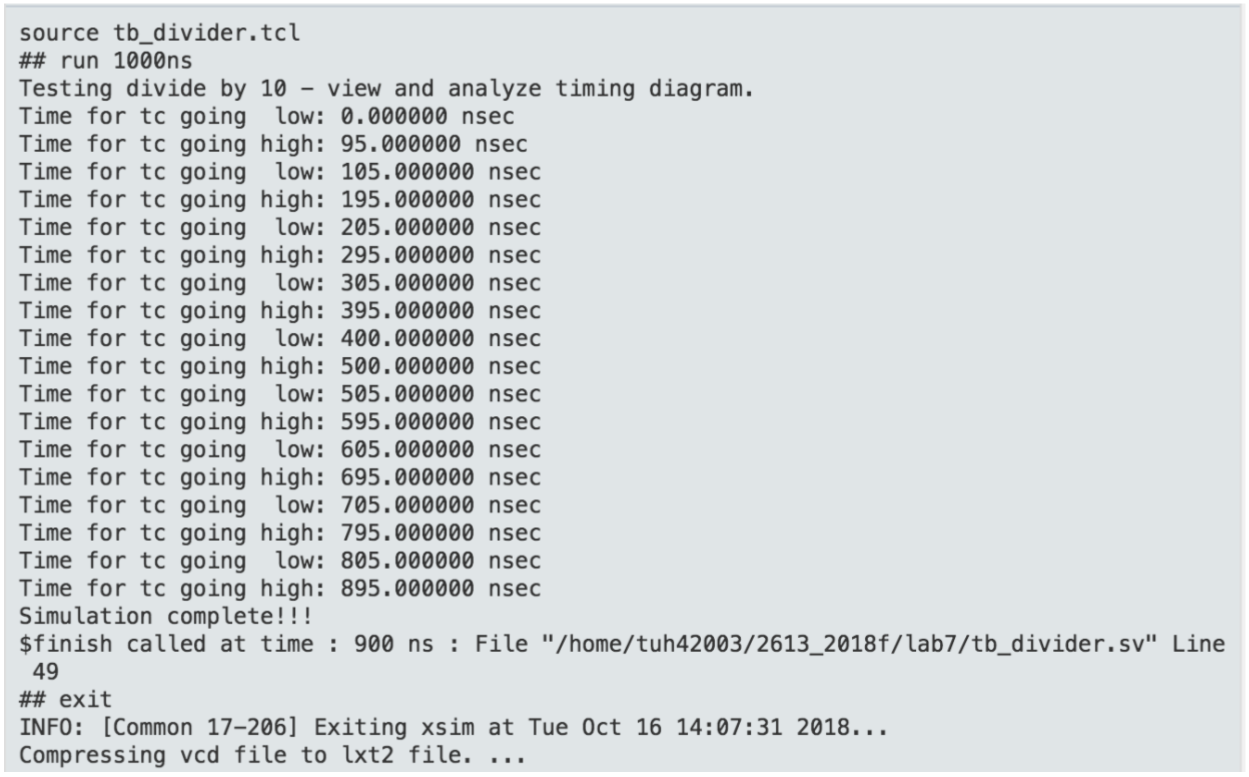


```
//
// lab7 : version 10/15/2018
//
module ir_emitter (output logic emitter_out, input logic clk, input logic rst, input logic ena);

logic tc_carrier,tc_carrier_2, tc_modulator;
logic sw_carrier,sw_modulator;

    // enter your code here - declare internal signals

    // carrier divide by 100MHz/38kHz / 2 = 1315 (rounded up)
    // enter your code here
divider #(.BIT_SIZE(11)) u1 (.tc(tc_carrier_2) , .count(), .clk(clk), .rst(rst), .ena(ena), .init_count(11'd1315));
    // carrier divide by 2 to make it a square wave of 50% duty cycle
    // enter your code here
divider #(.BIT_SIZE(1)) u2(.tc(tc_carrier) , .count(sw_carrier), .clk(clk), .rst(rst), .ena(tc_carrier_2), .init_count(1));

    // modulation divide by 10 (10 on ... 10 off)
    // enter your code here
divider #(.BIT_SIZE(4)) u3(.tc(tc_modulator) , .count(), .clk(clk), .rst(rst), .ena(tc_carrier), .init_count(9));

    // modulation divide by 2 to make it a square wave of 50% duty cycle
    // enter your code here
divider #(.BIT_SIZE(1)) u4(.tc() , .count(sw_modulator), .clk(clk), .rst(rst), .ena(tc_modulator), .init_count(1));

    // AND gate to create the emitter out signal
    // enter your code here

assign emitter_out = sw_carrier & sw_modulator;

endmodule
```

Simulation Results



```
source tb_divider.tcl
## run 1000ns
Testing divide by 10 - view and analyze timing diagram.
Time for tc going  low: 0.000000 nsec
Time for tc going high: 95.000000 nsec
Time for tc going  low: 105.000000 nsec
Time for tc going high: 195.000000 nsec
Time for tc going  low: 205.000000 nsec
Time for tc going high: 295.000000 nsec
Time for tc going  low: 305.000000 nsec
Time for tc going high: 395.000000 nsec
Time for tc going  low: 400.000000 nsec
Time for tc going high: 500.000000 nsec
Time for tc going  low: 505.000000 nsec
Time for tc going high: 595.000000 nsec
Time for tc going  low: 605.000000 nsec
Time for tc going high: 695.000000 nsec
Time for tc going  low: 705.000000 nsec
Time for tc going high: 795.000000 nsec
Time for tc going  low: 805.000000 nsec
Time for tc going high: 895.000000 nsec
Simulation complete!!!
$finish called at time : 900 ns : File "/home/tuh42003/2613_2018f/lab7/tb_divider.sv" Line
 49
## exit
INFO: [Common 17-206] Exiting xsim at Tue Oct 16 14:07:31 2018...
Compressing vcd file to lxt2 file. ...
```



```
source tb_divider.tcl
## run 1000ns
Testing divide by 10 - view and analyze timing diagram.
Time for tc going  low: 0.000000 nsec
Time for tc going high: 95.000000 nsec
Time for tc going  low: 105.000000 nsec
Time for tc going high: 195.000000 nsec
Time for tc going  low: 205.000000 nsec
Time for tc going high: 295.000000 nsec
Time for tc going  low: 305.000000 nsec
Time for tc going high: 395.000000 nsec
Time for tc going  low: 400.000000 nsec
Time for tc going high: 500.000000 nsec
Time for tc going  low: 505.000000 nsec
Time for tc going high: 595.000000 nsec
Time for tc going  low: 605.000000 nsec
Time for tc going high: 695.000000 nsec
Time for tc going  low: 705.000000 nsec
Time for tc going high: 795.000000 nsec
Time for tc going  low: 805.000000 nsec
Time for tc going high: 895.000000 nsec
Simulation complete!!!
$finish called at time : 900 ns : File "/home/tuh42003/2613_2018f/lab7/tb_divider.sv" Line
 49
## exit
INFO: [Common 17-206] Exiting xsim at Tue Oct 16 14:07:31 2018...
Compressing vcd file to lxt2 file. ...
```

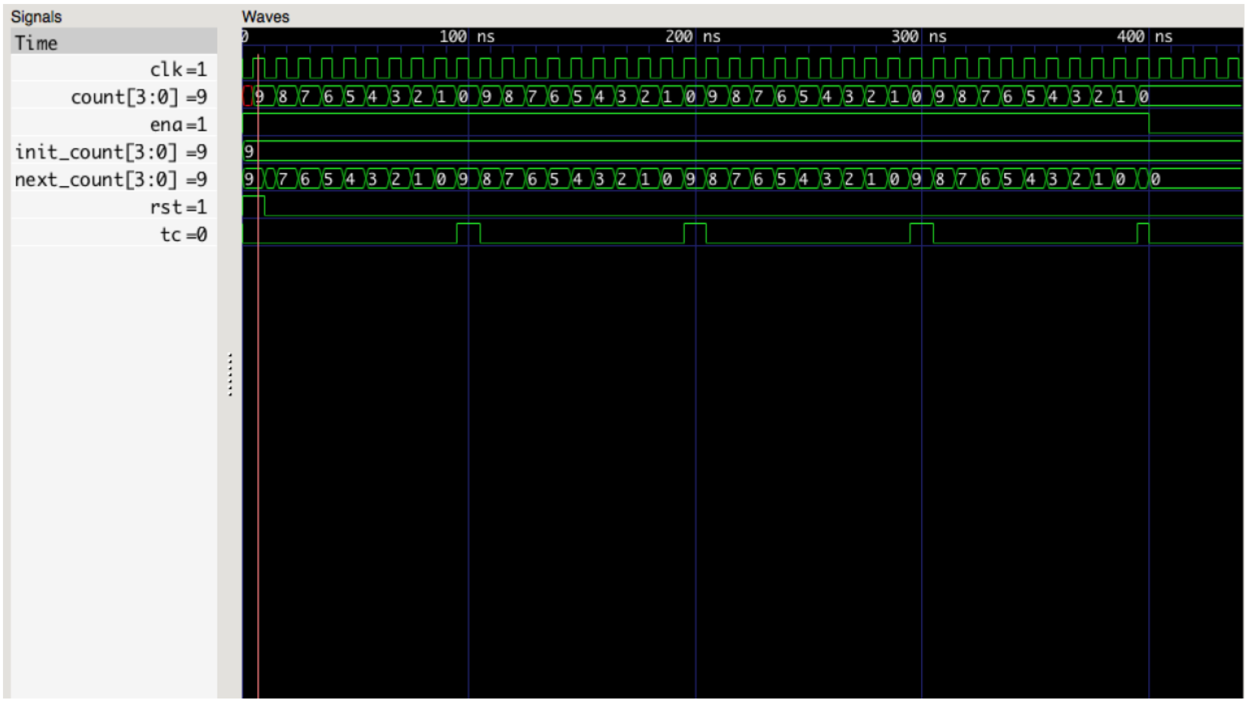
Hardware Implementation

My design was demonstrated to Catherine on the afternoon of 10/16 during the lab period.

Conclusion

After creating both the divider and then instantiating them we used the gtkwave application which was visually appealing for understand the clock cycles and what happens at ever cycle. This lab was the most interesting, but one of the hardest to understand. At the end the final signal was shown along with the two signal that we AND'ed.

Appendix


Signals
Waves
Time
0
100 ns
200 ns
300 ns
400 ns
clk=1
count[3:0] =9
ena=1
init_count[3:0] =9
next_count[3:0] =9
rst=1
tc=0