ELEX 7660

LAB 2 – ROTARY ENCODER INPUT

Nicholas Scott AKA “White Cheddar”

A01255181 | JAN 22^{ND}, 2023

ELEX 7660 Lab 2 Nicholas Scott
A01255181

Contents

Table of Figures

Prelab – encoder.sv

I implemented the following code for the encoder.sv module:

```
// ELEX 7660 Lab 2
// Author: Nicholas Scott AKA "White Cheddar" - SN: A01255181
// Date: Jan 23rd, 2024
// Instructor: Sweet Bobby T

module encoder (input logic a, b, clk,
                output logic cw, ccw);

    logic [1:0] state;  // State of the encoder output
    logic [1:0] previous_state;

    always_ff @(posedge clk) begin

        state <= {a,b};
        previous_state <= state;

        if (((state == 'b10) && (previous_state == 'b00)) ||
            ((state == 'b11) && (previous_state == 'b10)) ||
            ((state == 'b01) && (previous_state == 'b11)) ||
            ((state == 'b00) && (previous_state == 'b01))) begin    // True for all cases of cw rotation
                cw <= 1;
                ccw <= 0;
            end
        else if (((state == 'b01) && (previous_state == 'b00)) ||
                ((state == 'b11) && (previous_state == 'b01)) ||
                ((state == 'b10) && (previous_state == 'b11)) ||
                ((state == 'b00) && (previous_state == 'b10))) begin // True for all cases of ccw rotation
                    cw <= 0;
                    ccw <= 1;
                end
        else begin  // Set both cw and ccw to 0 if no rotation
            cw <= 0;
            ccw <= 0;
        end
    end
endmodule
```

Not the most elegant code I've ever written, but I don't really give a darn. This code yielded the following simulation waveforms and transcript:

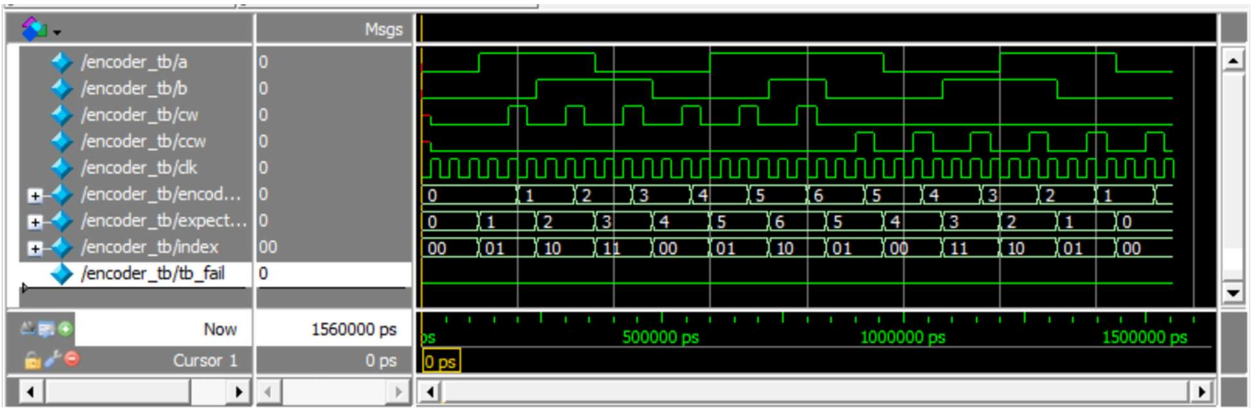
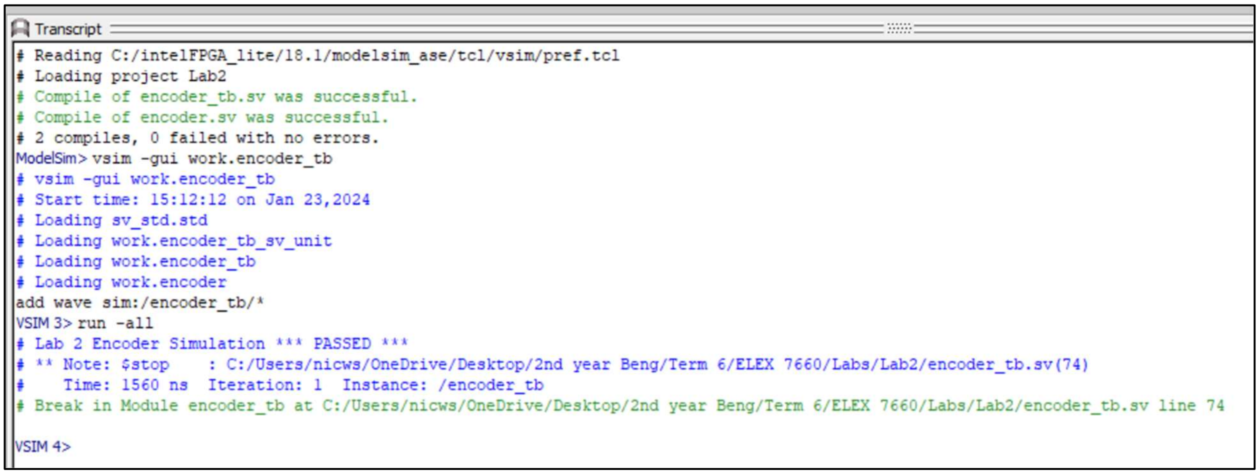


Figure 1 - 'encoder.sv' Simulation Waveforms



```
Transcript
# Reading C:/intelFPGA_lite/18.1/modelsim_ase/tcl/vsim/pref.tcl
# Loading project Lab2
# Compile of encoder_tb.sv was successful.
# Compile of encoder.sv was successful.
# 2 compiles, 0 failed with no errors.
ModelSim> vsim -gui work.encoder_tb
# vsim -gui work.encoder_tb
# Start time: 15:12:12 on Jan 23,2024
# Loading sv_std.std
# Loading work.encoder_tb_sv_unit
# Loading work.encoder_tb
# Loading work.encoder
add wave sim:/encoder_tb/*
VSIM 3> run -all
# Lab 2 Encoder Simulation *** PASSED ***
# ** Note: $stop    : C:/Users/nicws/OneDrive/Desktop/2nd year Beng/Term 6/ELEX 7660/Labs/Lab2/encoder_tb.sv(74)
#    Time: 1560 ns  Iteration: 1  Instance: /encoder_tb
# Break in Module encoder_tb at C:/Users/nicws/OneDrive/Desktop/2nd year Beng/Term 6/ELEX 7660/Labs/Lab2/encoder_tb.sv line 74

VSIM 4>
```

Figure 2 - 'encoder.sv' Simulation Transcript

Lab2.sv Modifications

I added the following code to lab2.sv to display the digits:

```
// Select digit to display (disp_digit)
// Left two digits (3,2) display encoder 1 hex count and right two digits (1,0)
// display encoder 2 hex count
  always_comb begin

    case (digit)
      0 : disp_digit = enc2_count[3:0]; // Set digit 0 to encoder 2 count LSN
      1 : disp_digit = enc2_count[7:4]; // Set digit 1 to encoder 2 count MSN
      2 : disp_digit = enc1_count[3:0]; // Set digit 2 to encoder 1 count LSN
      3 : disp_digit = enc1_count[7:4]; // Set digit 3 to encoder 1 count MSN
    endcase

  end
```

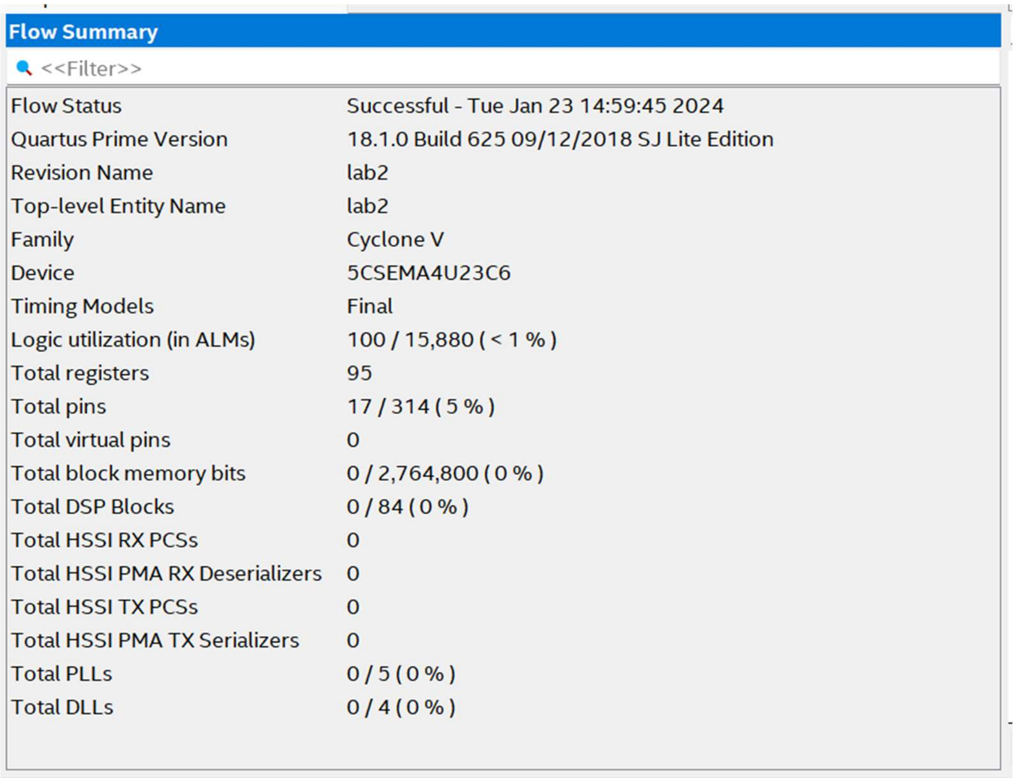
It's always been my life goal to coin a phrase. Maybe LSN (Least Significant Nibble) will catch on.

Improvements – enc2bcd.sv

The code I wrote to this point worked great. I then began working on the lab improvements. I wrote the following code for the enc2bcd.sv:

```
// ELEX 7660 Lab 2
// Author: Nicholas Scott AKA "White Cheddar" - SN: A01255181
// Date: Jan 23rd, 2024
// Instructor: Sweet Bobby T

module enc2bcd (input logic clk, cw, ccw,
                output logic [7:0] bcd_count);

    logic [7:0] countup;
    logic [7:0] countdown;

    always_ff @(posedge clk) begin

        if (cw)
            countup <= countup + 1; // Count up for cw movement
        else if (ccw)
            countdown <= countdown + 1; // Count "down" for ccw movement

        if (countup >= 4) begin
            bcd_count <= bcd_count + 1; // Increment bcd_count if countup reaches 4
            countup <= 0;
        end
        else if (countdown >= 4) begin
            bcd_count <= bcd_count - 1; // Decrement bcd_count if countdown reaches 4
            countdown <= 0;
        end

        if ((cw) &&
            ((bcd_count == 'h9) ||
            (bcd_count == 'h19) ||
            (bcd_count == 'h29) ||
            (bcd_count == 'h39) ||
            (bcd_count == 'h49) ||
            (bcd_count == 'h59) ||
            (bcd_count == 'h69) ||
            (bcd_count == 'h79) ||
            (bcd_count == 'h89)))   // If right digit is 9 and movement is cw

            bcd_count <= bcd_count + 7; // Increment count by 7 to skip hex A-F
        else if ((ccw) &&
                ((bcd_count == 'h10) ||
                (bcd_count == 'h20) ||
                (bcd_count == 'h30) ||
                (bcd_count == 'h40) ||
                (bcd_count == 'h50) ||
                (bcd_count == 'h60) ||
                (bcd_count == 'h70) ||
                (bcd_count == 'h80) ||
                (bcd_count == 'h90))) // If right digit is 0 and movement is ccw

            bcd_count <= bcd_count - 7; // Decrement count by 7 to skip hex F-A

        else if ((cw) && (bcd_count == 'h99))
                bcd_count <= 0;                 // cw rollover from 99->0

        else if ((ccw) && (bcd_count == 0))
                bcd_count <= 'h99;              // ccw rollover from 0->99
    end
endmodule
```

This code is a bit of a disasterpiece, but it worked! That's good enough for me. I instantiated two of these in lab2.sv and compiled. It yielded the following compilation report and RTL Netlists:



Flow Summary

<<Filter>>

Flow Status	Successful - Tue Jan 23 14:59:45 2024
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	lab2
Top-level Entity Name	lab2
Family	Cyclone V
Device	5CSEMA4U23C6
Timing Models	Final
Logic utilization (in ALMs)	100 / 15,880 (< 1 %)
Total registers	95
Total pins	17 / 314 (5 %)
Total virtual pins	0
Total block memory bits	0 / 2,764,800 (0 %)
Total DSP Blocks	0 / 84 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 5 (0 %)
Total DLLs	0 / 4 (0 %)

Figure 3 - Compilation Report

I can’t get rid of this page for some reason. Word does weird stuff when you make some of the document ‘landscape’ and some ‘portrait’. I guess I’ll use the space to show the code I added to lab2.sv to instantiate these modules:

```
  enc2bcd enc2bcd_1 (.clk(CLOCK_50), .cw(enc1_cw), .ccw(enc1_ccw), .bcd_count(enc1_count));
  enc2bcd enc2bcd_2 (.clk(CLOCK_50), .cw(enc2_cw), .ccw(enc2_ccw), .bcd_count(enc2_count));
```

I then commented out this part:

```
 // encoder counts: enc1_count & enc2_count (increment when cw=1, decrement when ccw=1)
 // always_ff @(posedge CLOCK_50)  begin

 //   // encoder 1 count
 //   if (enc1_cw) enc1_count <= enc1_count + 1'b1;
 //   else if (enc1_ccw) enc1_count <= enc1_count - 1'b1;

 //   //encoder 2 count
 //   if (enc2_cw) enc2_count <= enc2_count + 1'b1;
 //   else if (enc2_ccw) enc2_count <= enc2_count - 1'b1;

 // end
```

The rest of lab2.sv was left as-is, aside from what I added before. I wasn’t sure if you wanted to see that or not.

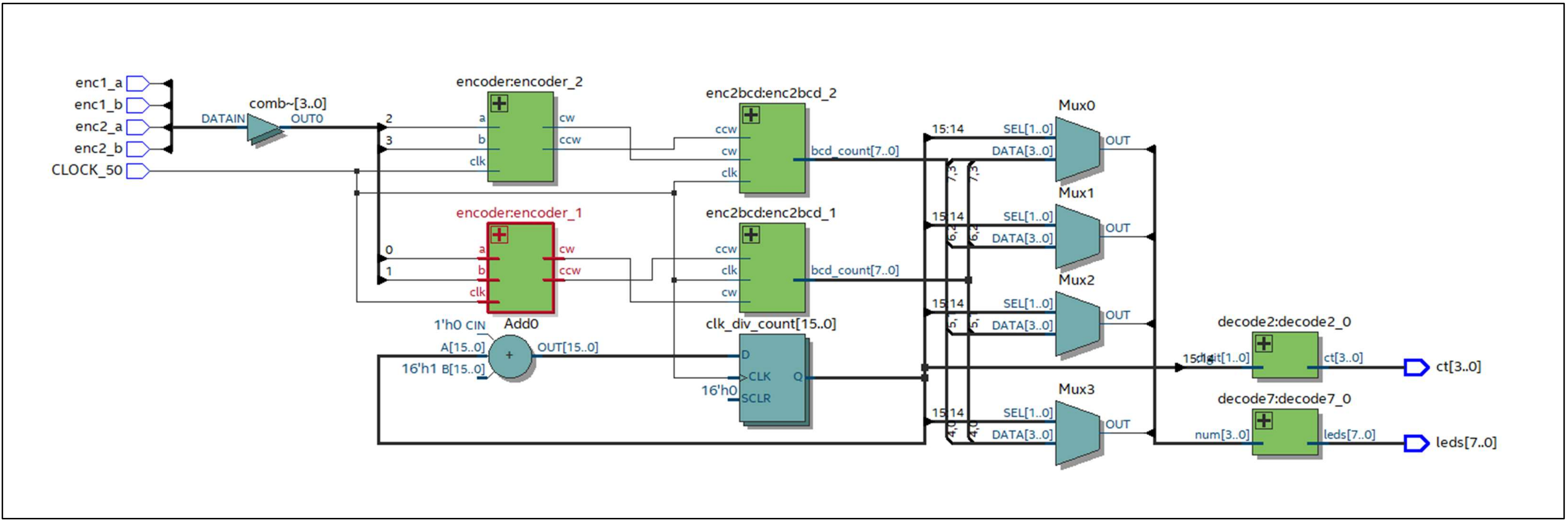


Figure 4 - Full RTL Netlist

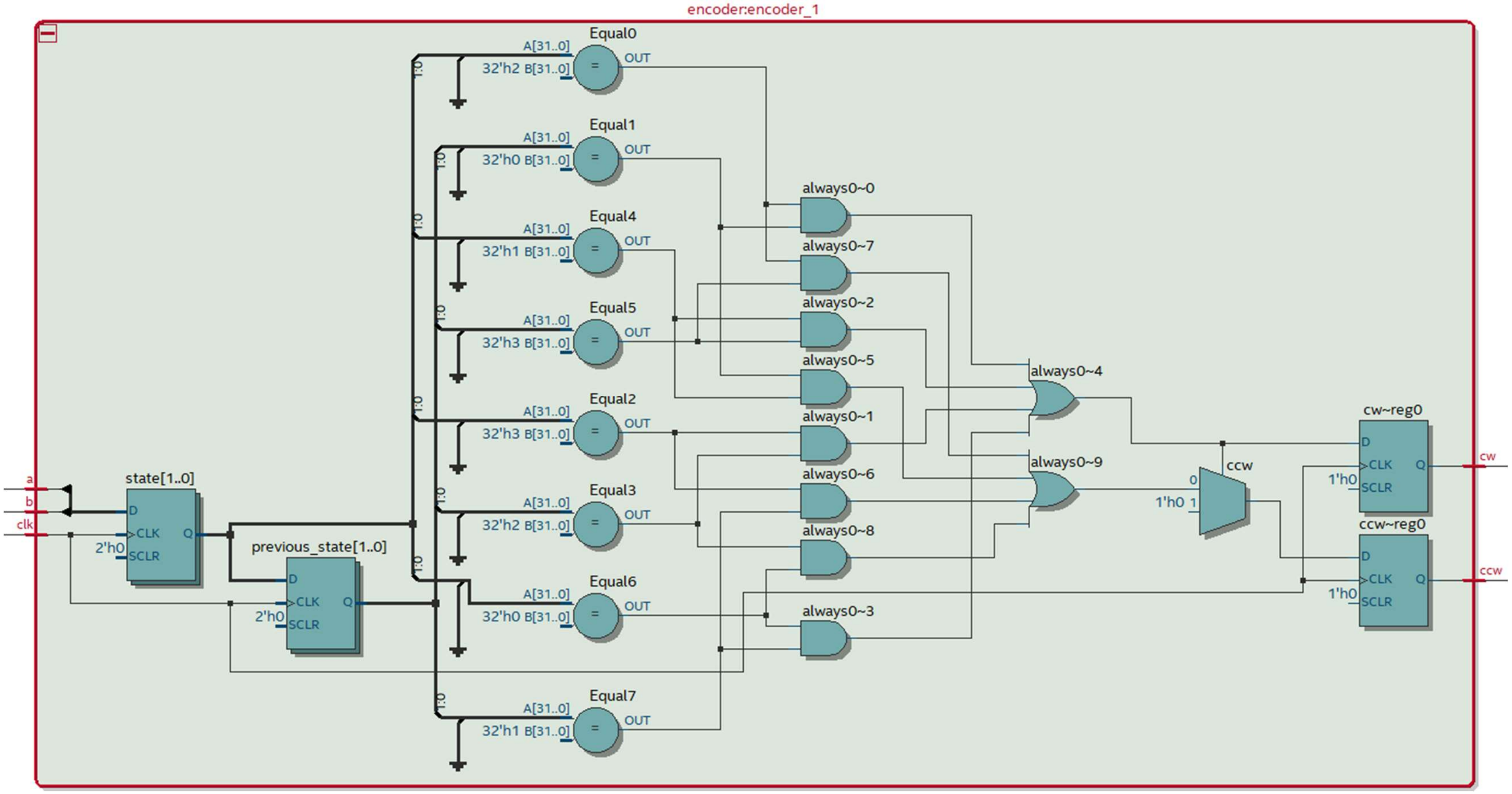


Figure 5 - 'encoder.sv' RTL Netlist

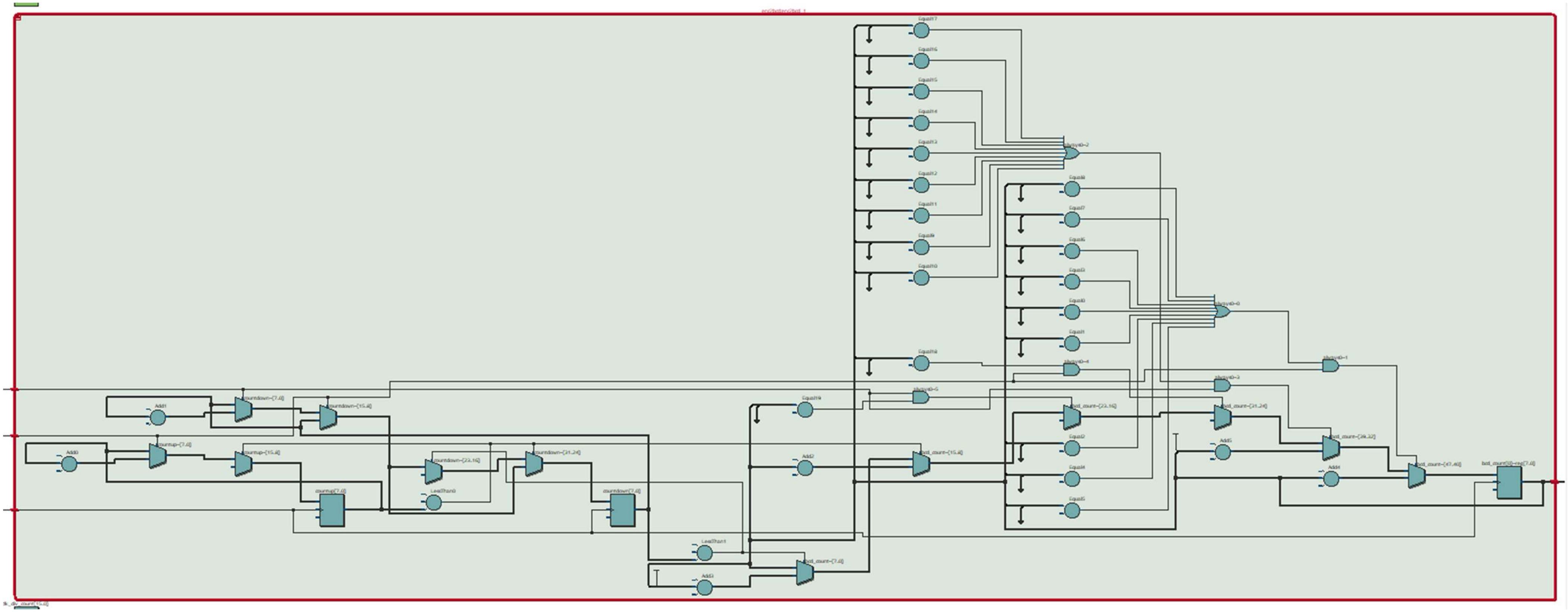


Figure 6 - 'enc2bcd.sv' RTL Netlist - "The Abomination"