A blue square with white text

Description automatically generated

ELEX 7660: Digital System Design

Lab 3

|  |  |  |
| --- | --- | --- |
| Student Name: Taewoo Kim | Student Number: A01284763 | Set: T |

Table of Contents

[1 Screenshot of the simulations 3](#_Toc189948016)

[2 Source code of the module 3](#_Toc189948017)

[2.1 Lab3.sv code 3](#_Toc189948018)

[2.2 enc2freq.sv code 5](#_Toc189948019)

[2.3 enc2freq\_tb.sv 6](#_Toc189948020)

[2.4 tongen.sv 8](#_Toc189948021)

[3 Quartus compilation report 9](#_Toc189948022)

[4 RTL Netlist 9](#_Toc189948023)

[4.1 Overall view 9](#_Toc189948024)

[4.2 enc2freq.sv 10](#_Toc189948025)

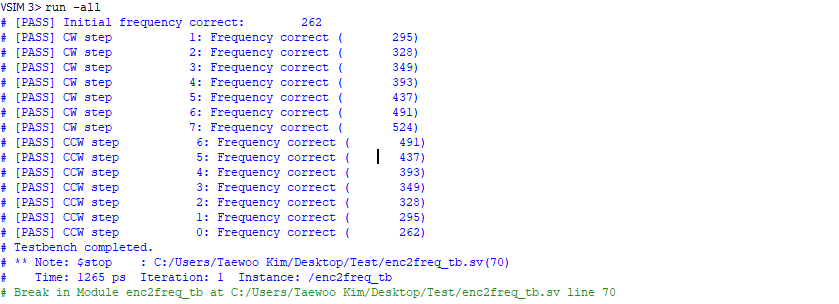
[4.3 tongen.sv 10](#_Toc189948026)

[4.4 decode2.sv 10](#_Toc189948027)

[4.5 decode7.sv 11](#_Toc189948028)

[4.6 encoder.sv 12](#_Toc189948029)

# Screenshot of the simulations

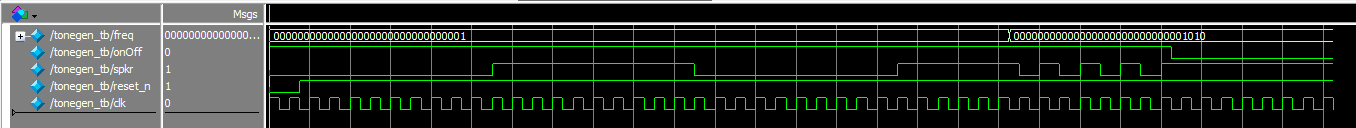


A screenshot of a computer

AI-generated content may be incorrect.

A computer code with text

AI-generated content may be incorrect.



# Source code of the module

## Lab3.sv code

|  |
| --- |
| // File: lab3.sv  // Description: ELEX 7660 lab3 top-level module.  //              We can scale the buzzer to make different frequency sounds  //           and 7-segment display the corresponding frequency.  //              display module.  // Author: Taewoo Kim  // Date: 2025-02-02  module lab3 ( input logic CLOCK\_50,       // 50 MHz clock                input logic s1,                input logic s2,                (\* altera\_attribute = "-name WEAK\_PULL\_UP\_RESISTOR ON" \*)                input logic enc1\_a, enc1\_b, //Encoder 1 pins                (\* altera\_attribute = "-name WEAK\_PULL\_UP\_RESISTOR ON" \*) input logic                enc2\_a, enc2\_b,             //Encoder 2 pins                output logic [7:0] leds,    // 7-seg LED enables                output logic [3:0] ct,      // digit cathodes                output logic spkr ) ;       // speaker     logic [1:0] digit;  // select digit to display     logic [3:0] disp\_digit;  // current digit of count to display     logic [15:0] clk\_div\_count; // count used to divide clock     logic [31:0] desired\_freq;     logic enc1\_cw, enc1\_ccw;     // instantiate modules to implement design     decode2 decode2\_0 (.digit,.ct) ;     decode7 decode7\_0 (.num(disp\_digit), .leds) ;    // instantiate encoders     encoder encoder\_1 (.clk(CLOCK\_50), .a(enc1\_a), .b(enc1\_b), .cw(enc1\_cw), .ccw(enc1\_ccw));    // instantiate encoder to bcd     enc2freq enc2freq\_1 (.cw(enc1\_cw), .ccw(enc1\_ccw), .freq(desired\_freq), .reset\_n(s1), .clk(CLOCK\_50));     tonegen tonegen\_1 ( .freq(desired\_freq), .onOff(s2), .spkr(spkr), .reset\_n(s1), .clk(CLOCK\_50));       // use count to divide clock and generate a 2 bit digit counter to determine which digit to display     always\_ff @(posedge CLOCK\_50)       clk\_div\_count <= clk\_div\_count + 1'b1 ;    // assign the top two bits of count to select digit to display    assign digit = clk\_div\_count[15:14];    // 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        // according to enc1\_counts or enc2\_counts value set disp\_digit accordingly to set the leds       case (digit)          2'b00: disp\_digit = desired\_freq[3:0]; // if digit is 0          2'b01: disp\_digit = desired\_freq[7:4]; // if digit is 1          2'b10: disp\_digit = desired\_freq[11:8]; // if digit is 2          2'b11: disp\_digit = desired\_freq[15:12]; // if digit is 3          default: disp\_digit = 4'b0000;       // Default or error value       endcase    end  endmodule |

## enc2freq.sv code

|  |
| --- |
| // File: enc2freq.sv  // Description: enc2freq module using cw or ccw generate the desired freq scale  // Author: Taewoo Kim  // Date: 2025-02-02  module enc2freq ( input logic cw, ccw,        // input cw and ccw                    output logic [31:0] freq,   // desired frequency                    input logic reset\_n, clk ); // reset and clock      // array of frequency scale      logic [31:0] freq\_array [7:0] = '{524, 491, 437, 393, 349, 328, 295, 262};      // Count until 3      localparam PULSE\_COUNTER = 2'b11;      // initalize the logic count (0~3 and wrap back)      logic [1:0] cw\_counter = 0, ccw\_counter = 0;      logic [2:0] counter = 0;      logic cw\_ready, ccw\_ready;      // using if statement to check if counter is ready and if it is send out the ccw/cw\_ready signal.      always\_ff @(posedge clk, negedge reset\_n) begin          if (~reset\_n) begin              counter <= 0;        end        // CW handling check if it's mutually exclusive        else if (cw && !ccw) begin          // once it reaches the third state (0~3) send cw\_ready and reset the counter           if (cw\_counter == PULSE\_COUNTER) begin              cw\_counter <= 0;              counter <= counter + 1;          // if it didn't reached, send 0 for cw\_ready and count up the counter           end else begin              cw\_counter <= cw\_counter + 1;           end        end          // CCW handling (similar) check if it's mutually exclusive        else if (ccw && !cw) begin          // once it reaches the third state (0~3) send ccw\_ready and reset the counter           if (ccw\_counter == PULSE\_COUNTER) begin              ccw\_counter <= 0;              counter <= counter - 1;          // if it didn't reached, send 0 for ccw\_ready and count up the counter           end else begin              ccw\_counter <= ccw\_counter + 1;           end        end      end      // assign the desired frequency      assign freq = freq\_array[counter];    endmodule |

## enc2freq\_tb.sv

|  |
| --- |
| // File: enc2freq\_tb.sv  // Description: This is simple testbench to test enc2freq module  // Author: Taewoo Kim, ChatGPT (got some help)  // Date: 2025-02-02  module enc2freq\_tb();      // Testbench signals      logic cw, ccw, reset\_n, clk; // Clockwise, counter-clockwise, reset, and clock signals      logic [31:0] freq; // Output frequency signal        // Instantiate the DUT (Device Under Test)      enc2freq dut ( .cw(cw), .ccw(ccw), .freq(freq), .reset\_n(reset\_n), .clk(clk) );        // Clock generation      always #5 clk = ~clk; // 10 ns period (100 MHz)        // Frequency array (expected values)      logic [31:0] expected\_freq [7:0] = '{524, 491, 437, 393, 349, 328, 295, 262};        // Task to generate pulses, chatGPT generated send\_pulses task.      task send\_pulses(input logic dir);          integer i;          for (i = 0; i < 4; i = i + 1) begin // Generate 4 pulses per call              @(posedge clk); // Wait for a clock edge              if (dir) cw = 1; else ccw = 1; // Set appropriate direction signal              @(posedge clk);              cw = 0; ccw = 0; // Reset signals          end      endtask        // Test sequence      initial begin          // Initialize signals          clk = 0;          reset\_n = 0;          cw = 0;          ccw = 0;          @(posedge clk);          reset\_n = 1; // Release reset after one clock cycle            // Verify initial frequency          if (freq !== expected\_freq[0])  // Check if initial frequency matches expected value              $display("[FAIL] Initial frequency incorrect: Expected %d, Got %d", expected\_freq[0], freq);          else              $display("[PASS] Initial frequency correct: %d", freq);            // Test CW direction          for (int i = 1; i < 8; i++) begin              send\_pulses(1); // Send pulses in CW direction              @(posedge clk); // Wait for one clock cycle to allow frequency update              if (freq !== expected\_freq[i])                  $display("[FAIL] CW step %d: Expected %d, Got %d", i, expected\_freq[i], freq);              else                  $display("[PASS] CW step %d: Frequency correct (%d)", i, freq);          end            // Test CCW direction          for (int i = 6; i >= 0; i--) begin              send\_pulses(0); // Send pulses in CCW direction              @(posedge clk); // Wait for one clock cycle to allow frequency update              if (freq !== expected\_freq[i])                  $display("[FAIL] CCW step %d: Expected %d, Got %d", i, expected\_freq[i], freq);              else                  $display("[PASS] CCW step %d: Frequency correct (%d)", i, freq);          end            // End simulation          $display("Testbench completed.");          $stop; // Halt simulation      end  endmodule |

## tongen.sv

|  |
| --- |
| // File: tongen.sv  // Description: simple tongen module to create the tone using  //              input as desired frequency.  // Author: Taewoo Kim  // Date: 2025-02-02  module tonegen #(parameter FCLK = 50000000)                // clock frequency, Hz          ( input logic [31:0] freq,      // frequency to output on speaker            input logic onOff,            // 1 -> generate output, 0-> no output            output logic spkr,            // speaker output            input logic reset\_n, clk);    // reset and clock    // initialize the counter and desired\_freq variable    logic [31:0] count;    logic [31:0] desired\_freq;    // shift by 1 to multiply 2 to freq without using multiplication    assign desired\_freq = freq << 1;    // using always\_ff to count to get desired cycles    always\_ff @( posedge clk, negedge reset\_n ) begin        // rest the spkr and counter      if(~reset\_n) begin        spkr <= 0;        count <= 0;      end      // if tonegen is on start count until desired cycle      else if (onOff) begin        // count the counter using desired freq        count <= count + desired\_freq;        // once it reached desired cycle toggle spkr to create tone        if(count >= (FCLK-desired\_freq)) begin          spkr <= ~spkr;          count <= 0;        end      end        end  endmodule |

# Quartus compilation report

A screenshot of a computer

AI-generated content may be incorrect.

# RTL Netlist

## Overall view

A diagram of a computer

AI-generated content may be incorrect.

## enc2freq.sv

A computer screen shot of a diagram

AI-generated content may be incorrect.

## tongen.sv

A diagram of a circuit

AI-generated content may be incorrect.

## decode2.sv

A diagram of a computer program

AI-generated content may be incorrect.

## decode7.sv

A screenshot of a computer program

AI-generated content may be incorrect.

## encoder.sv

A diagram of a computer program

AI-generated content may be incorrect.