Nicholas Scott AKA “White Cheddar”

A01255181 | Jan 14th, 2024

ELEX 7660 Lab 1

7-Segment led display decoder

Contents

[Part 1 – Coding and Simulation 2](#_Toc156465354)

[decode7 2](#_Toc156465355)

[decode2 4](#_Toc156465356)

[bcitid 5](#_Toc156465357)

[Part 2 – Synthesis and Implementation 6](#_Toc156465358)

**Table of Figures**

[Figure 1 - 'decode7' Simulation Waveforms 3](#_Toc156465939)

[Figure 2 - 'decode2' Simulation Waveforms 4](#_Toc156465940)

[Figure 3 - 'bcitid' Simulation Waveforms 5](#_Toc156465941)

[Figure 4 - My First Compilation Report 6](#_Toc156465942)

[Figure 5 - RTL Netlist 0](#_Toc156465943)

# Part 1 – Coding and Simulation

In this section, I will present the code I wrote, and the simulation results they yielded.

## decode7

I started with the most difficult module. I’m a wild man.

// ELEX 7660 Lab 1

// Nicholas Scott AKA "White Cheddar"

// A01255181

// Instructor: Sweet Bobby T

module decode7 (input logic [3:0] num,

                output logic [7:0] leds) ;

    // assign leds to the necessary bit configuration to display num on 7-segment (in hexidecimal)

    always\_comb

        case(num)

            0 : leds = 'b00111111;

            1 : leds = 'b00000110;

            2 : leds = 'b01011011;

            3 : leds = 'b01001111;

            4 : leds = 'b01100111;

            5 : leds = 'b01101101;

            6 : leds = 'b01111101;

            7 : leds = 'b00000111;

            8 : leds = 'b01111111;

            9 : leds = 'b01100111;

            10 : leds = 'b01110111;     // displays hex character 'A'

            11 : leds = 'b01111100;     // displays hex character 'b'

            12 : leds = 'b00111001;     // displays hex character 'C'

            13 : leds = 'b01011110;     // displays hex character 'd'

            14 : leds = 'b01111001;     // displays hex character 'E'

            15 : leds = 'b01110001;     // displays hex character 'F'

        endcase

endmodule

This code simply implements the truth table from the Lab Prep. Craig always taught us that if you write good, legible code, you won’t need many comments. This code is very simple, and I only felt the need to comment the lines that implemented the hex digits A-F, as it is may not be obvious what is happening there.

Simulation of this code yielded the following waveforms:

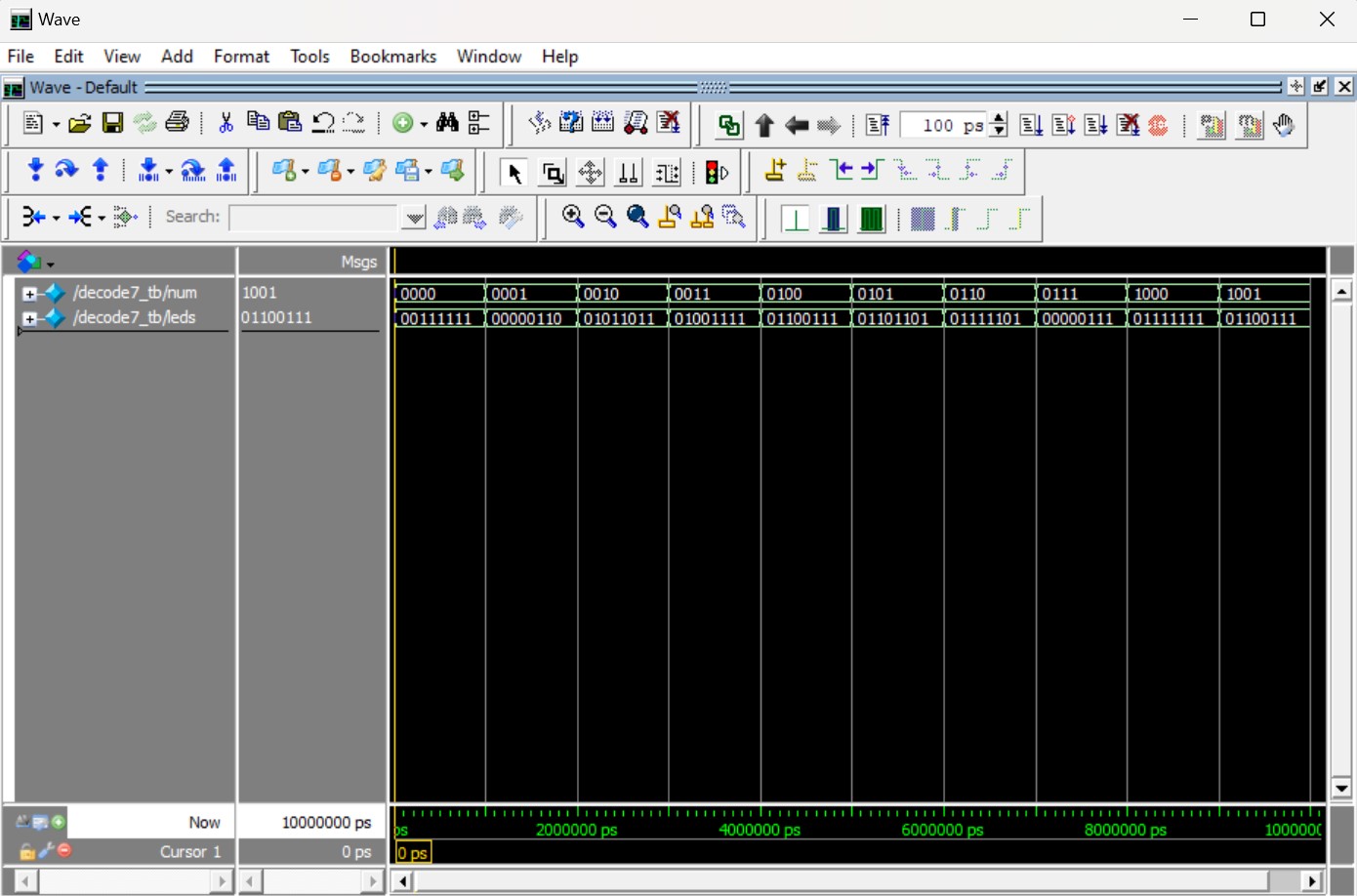


Figure - 'decode7' Simulation Waveforms

As shown in *Figure 1*, the code implements the truth table as expected. It is important to note that the test bench only tested digits 0-9, which will do for this lab, but for future labs it will be important to remember that hex digits A-F have not been tested yet.

## decode2

Since I started with **decode7**, the rest of the coding was simple in comparison. The only thinking involved in the remaining code is ensuring the proper sequence dictated by the *digit* input. The lab document claims that when *digit* is 0, the rightmost digit of the display is active, and when *digit* is 3, the leftmost digit is active.

// ELEX 7660 Lab 1

// Nicholas Scott AKA "White Cheddar"

// A01255181

// Instructor: Sweet Bobby T

module decode2 (input logic [1:0] digit,

                output logic [3:0] ct);

    // 'ct' enables one digit of 7-segment display based on 'digit' input

    always\_comb

        case(digit)

            0 : ct = 'b1110;    // rightmost digit enabled

            1 : ct = 'b1101;

            2 : ct = 'b1011;

            3 : ct = 'b0111;    // leftmost digit enabled

        endcase

endmodule

This code yielded the following simulation result:

A screenshot of a computer

Description automatically generated

Figure - 'decode2' Simulation Waveforms

Again, the simulation proved that the code effectively implements the truth table.

## bcitid

The code for this module is very similar to **decode2**. The same logic applies to the sequencing dictated by the two-bit *digit* input.

// ELEX 7660 Lab 1

// Nicholas Scott AKA "White Cheddar"

// A01255181

// Instructor: Sweet Bobby T

module bcitid (input logic [1:0] digit,

                output logic [3:0] idnum);

    // assign '5181' to digits 3 to 0, respectively

    always\_comb

        case(digit)

            0 : idnum = 1;  // rightmost digit

            1 : idnum = 8;

            2 : idnum = 1;

            3 : idnum = 5;  // leftmost digit

        endcase

endmodule

This code yielded the following simulation result:

A screenshot of a computer

Description automatically generated

Figure - 'bcitid' Simulation Waveforms

Yet again, the code implements the truth table perfectly. Almost too perfectly if you ask me. Something is bound to go wrong eventually.

# Part 2 – Synthesis and Implementation

I imported the files and made some modifications to change the name of the project to *x7660Lab1*. I then was able to compile the code and received the following report:

A screenshot of a computer program

Description automatically generated

Figure - My First Compilation Report

I’m going to have it framed. I then generated the RTL Netlist:

A computer screen shot of a diagram

Description automatically generated

Figure - RTL Netlist

I then uploaded the code and completed the demonstration.