Lab 2: One Bit Magnitude Comparator in Verilog

Logic Lab

By: Nathan Phipps, Erich Wanzek

Abstract

In this text, we analyze our verilog code for a Two bit magnitude comparator, which compares bits, A[0], A[1] to B[0], and B[1]. The comparator determines whether the output function results in values that are "less than," "greater than," or "equal to" each other from comparing the A and B bits. This text will be outlined as follows:

1.) Analysis of our Verilog codes and trials (as listed in the lab rubric), including a brief discussion on our
   a.) "One_bit_comparator" level,
   b.) "Two_bit comparator" level,
   c.) "Top_level" level,
2.) A brief discussion of each group members contributions,
3.) Our codes for each of the levels (listed above),
4.) Pictures of the wave diagram from our lab,
5.) Pictures from the designated trials (these trials are listed in the lab guidelines).

Analysis

In this lab we utilized verilog code to create a two bit magnitude comparator which consisted of three levels of verilog modules. These three modules were the one bit comparator, the two bit comparator, and the top level module. The one bit comparator module compared one bit magnitude binary numbers. Next the two bit comparator module instantiated two of the one bit comparator modules, in order to compare a pair of two bit binary numbers. The top level entity module instantiates the two bit comparator module and also defines the hardware inputs and outputs which consist of seven LEDRs and four switches. The objective of our three modules was to compare values within two bits for inputs A, as well as B (i.e. A[0], A[1], B[0], B[1]), allowing us to determine whether A was “greater than,” “less than,” or “equal to,” B and vice versa. The timing pulse inputs (i.e. A[0], A[1], B[0], B[1]) followed a binary pattern in nanoseconds, which represented all the possible combinations of two bit binary numbers to be compared.

The output waveforms correspond to the inputs, and resulted from the logical expressions executed within the modules (i.e. the expressions for the functions “less than,” “greater than,” and “equal to,” found within our verilog code). From our lab rubric we performed a series of trials to test our two bit comparator modules. After uploading our top level module, which encompassed all other module levels/code, our board performed as desired, successfully comparing, as well as displaying the correct values among our inputs and outputs.

Contributions

Throughout the entirety of this lab, both group members, Erich Wanzek, and Nathan Phipps, performed equal duties, in the development of the Verilog modules, testing and performance of trials, and working on the analytical report.

Code

Code for Top Level

```
module top_level (SW, LEDR); //define module name and inputs and outputs

        input wire [3:0] SW; //Assigns the three input switches
        output wire [6:0] LEDR; //Assigns the six output LEDs

        assign LEDR[3:0] = SW[3:0]; // assigns 4 LEDS to 4 switches

        two_bit_comparator inst0 (SW[3:2], SW[1:0], LEDR[6], LEDR[5], LEDR[4]); // calls the module two_bit_comparator

endmodule //module now ended
```

Code for Two Bit Comparator

```
module two_bit_comparator (A, B, LT, GT, EQ); // define module name and inputs and outputs

        input [1:0] A,B;     // assign inputs
        output GT, LT, EQ;  // assign outputs
        wire EQ_1, EQ_0 ,LT_1 ,LT_0, GT_1, GT_0; //assign wires

        one_bit_comparator inst0 (A[1], B[1], GT_1, LT_1, EQ_1); //call module one__bit_comparator to compare most significant
        bit

        one_bit_comparator inst1 (A[0], B[0], GT_0, LT_0, EQ_0); //call module one__bit_comparator to compare least significant
        bit

        //compare LSB bit and MSB bits

        assign GT = GT_1 + (EQ_1 & GT_0); //assigns Greater than value "GT" to the function GT_1 + (EQ_1 & GT_0)
        assign EQ = (EQ_1 & EQ_0); //assigns Equal to value "EQ to the function (EQ_1 & EQ_0)
        assign LT = LT_1 + (EQ_1 & LT_0);//assigns Less than value "LT" to the function LT_1 + (EQ_1 & LT_0)

endmodule //module now ended
```

Code for One Bit Comparator

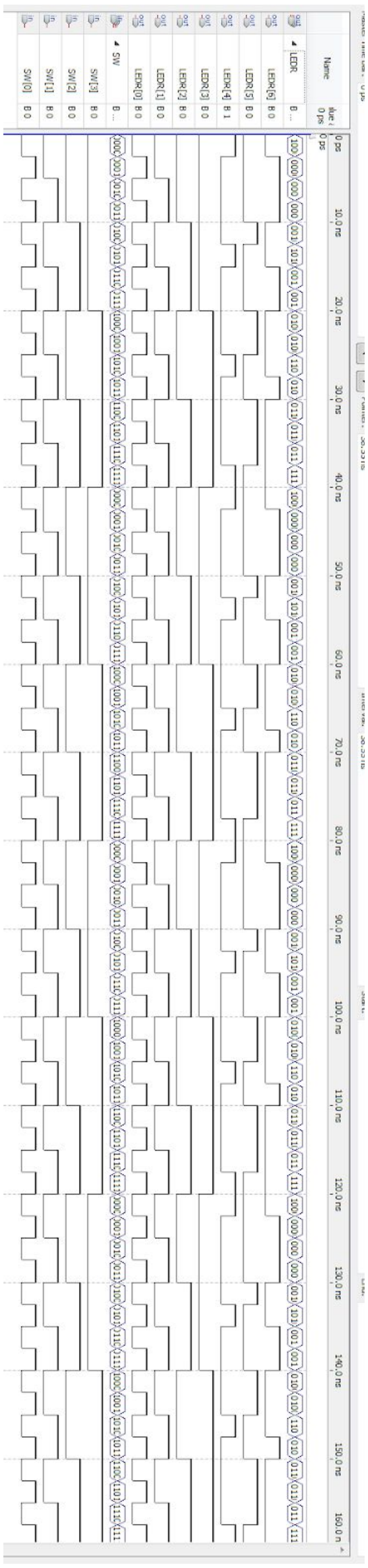
```
module one_bit_comparator (A, B, GT, LT, EQ); //define module name, and inputs and outputs

        input A,B; //assigns input A, B
        output GT, LT, EQ; //assigns outputs GT, LT, EQ


        assign GT = A & (~B); //assigns Greater than value "GT" to the function A & (~B)
        assign EQ = (~A&~B)+(A&B); //assigns Equal to value "EQ to the function (~A&~B)+(A&B)
        assign LT = ~A&B; //assigns Less than value "LT" to the function ~A&B

endmodule //module now ended
```

Timing Diagram

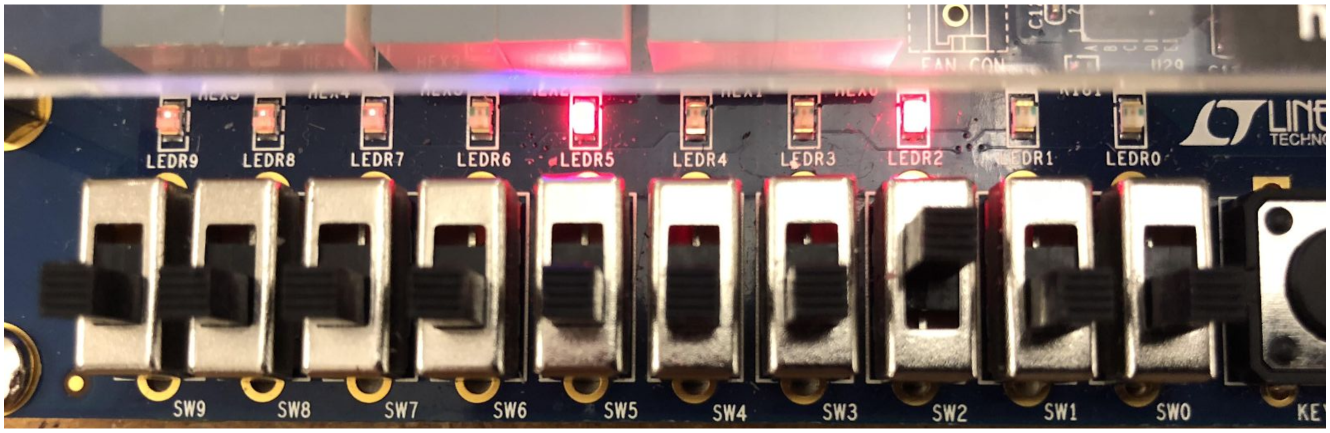


Trials

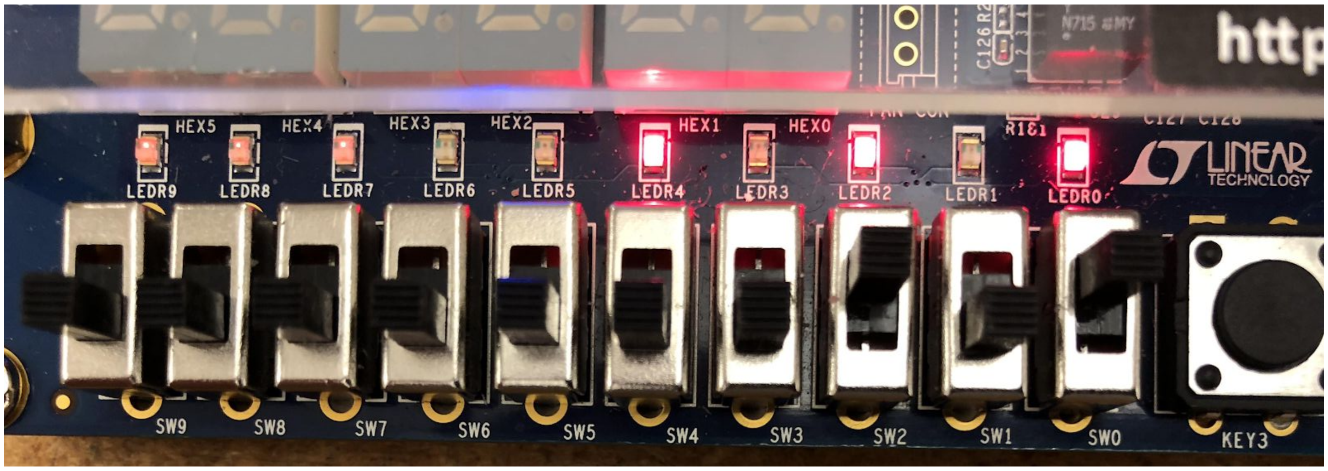
Input Binary Values				Output		
A1	A0	B1	B0	GT	EQ	LT
0	1	0	0	1	0	0
0	1	0	1	0	1	0
1	0	1	1	0	0	1

LEDR[6] = LT   LEDR[5] = GT   LEDR[4] = EQ

Trial 1: A[1] = 0, A[0] = 1, B[1] = 0, B[0] = 0, GT = 1, EQ = 0, LT = 0



Trial 2: A[1] = 0, A[0] = 1, B[1] = 0, B[0] = 1, GT = 0, EQ = 1, LT = 0



Trial 3: A[1] = 1, A[0] = 0, B[1] = 1, B[0] = 1, GT = 0, EQ = 0, LT = 1