EECE 2323 Digital Logic Design Lab Report

Lab 1 8-Bit Adder

Student Name:Ousmane Toure

Section #: 1

Instructor Name: Dr. Aboelela

Date: 07/09/22

Lab TA Name: Keshav Bharadwaj Vaidyanathan, Srinidi Somasundaram Subbulakshmi, Yuchao Su

1. Background & Purpose:

In this experiment, we designed and implemented an 8-bit adder which takes two 8-bit numbers and outputs the sum of both numbers and an overflow flag. 8-bit Adders are important because they are used for basic arithmetic operations. This is important because when creating computational circuits, you might run into a case where numbers needed to be added together, and if a computer can't even add two numbers it most likely wouldn't be able to do complex calculations. The goal was to utilize verilog code in vivado to represent 8-bit adders, building confidence and proficiency in the matter.

2. Pre-Lab Response:

Table 1: 2's Complement Truth Table

a	b	f	ovf
8'd0 00000000	8'd0 00000000	8'd0 0000000	0
8'd12 00001100	8'd34 00100010	8'd46 00101110	0
-8'd12 11110100	-8'd34 11011110	-8'd46 11010010	0
8'd100 01100100	-8'd50 11001110	8'd50 00110010	0
-8'd100 10011100	8'd50 00110010	-8'd50 11001110	0
8'd100 01100100	8'd100 01100100	-8'd56 11001000	1
-8'd100 10011100	-8'd100 10011100	8'd56 00111000	1
-8'd13 11110011	8'd12 00001100	-8'd1 11111111	0

-8d13 11110011	-8d13 11110011	-8'd26 11100110	0

2. Write a Boolean or verilog equation for the overflow output ovf based on input values a and

a[7] b[7] f[7]

A'B'F+ABF'

((A[7] & B[7]) &!F[7]) | ((!A[7] & !B[7])&F[7])

3. A good simulation testbench tests that every input and output bit can take the values zero and one. For the values in question 1 answer the following questions:

(a) The adder has 16 bits of input and 9 bits of output. Do the values in question 1 taken together set every input bit to both one and zero and every output bit to both one and zero? Explain your answer. Identify which bits are not tested in this manner.

No, because the [0] bit is not tested as there is no negative numbers. this means that they are always 0 and never checked for 1

(b) Add additional input (a and b) values that test every bit in both the inputs and outputs so that every bit takes either a zero or a one value in your test cases.

highlighted in Table 1, a negative value plus a positive and two negative values which had a 1 in the LSB were used to ensure all bits were accounted for.

3. Summary of Design Implementation

3.1.Results and Analysis:

When conducting this experiment, the 8 bit adder took two 8-bit data inputs and output one 8-bit data output and one overflow flag. After creating our code represented in Appendix A for our Test Bench, we plugged in values and ran a simulation to verify that an overflow would in fact occur when two negative numbers or two positive numbers were added together, but the sum was the opposite sign. Our results, as stated in Appendix B Figure 1. The results of the test bench verified that our code in fact does work and gives us the green light to program straight into the PYNQ-Z2 board. After programming our board and connecting the add on board, we displayed -100 + -100 resulting in a value of 56. This can be seen as correct as the overflow bit, which is LD3, is highlighted, indicating an overflow. Trends were seesaw hen adding other two negative numbers and positive numbers such as 100+100. Errors could occur if you used wrong bitwise vs logical operators when writing your verilog code. For example, using && and not & would produce an error from vivado telling you your formatting is incorrect. Furthermore, incorrectly setting up your test bench would produce inaccurate results.

3.2.Conclusion & Recommendations:

Based on our results, we can conclude that
Lab 1 consisted of creating an 8-bit adder in Vivado utilizing verilog code, testing its implementation with a test bench and then programming it to our FPGA board. The lab resulted in successfully being able to add two 8 bit numbers while taking into account any overflow that could occur. Figure 2 in Appendix B, represents the successful demonstration of this. Recommendations

going forward would be to test more than 4 test values. This ensures that you can test a negative and a negative, positive and a negative, positive and a positive, and then a positive and a negative. Ensuring you don't overlook any combinations. Overall this lab reinforced the fundamentals of digital circuit design using verilog code.

Appendix A: Design Program Files (Verilog modules, testbenches, etc)

```
`timescale 1ns / 1ps
//Ousmane Toure
module xorgate_tb();
    //inputs
    reg [7:0] A;
    reg [7:0] B;
    wire [7:0] F;
    wire OVF;
    //wrapper
    eightbit_adder UUT
        (.A(A),
         .B(B),
         .F(F),
         .OVF(OVF)
         );

    initial
        begin
            #100 //delay then start 100 ms
             A = 8'd0;
             B = 8'd0;
             //0+0
            #100 //200 ms
            A = 8'd12;
            B=  8'd34;
            #100 //300 ms
            A = -8'd12;
            B=  -8'd34;
            #100 // 400ms
            A = 8'd100;
            B=  -8'd50;
            #100 //500 ms
            A = -8'd100;
            B=  8'd50;

            #100 //600ms
            A = 8'd100;
            B=  8'd100;
            #100 //700ms
            A = -8'd100;
            B=  -8'd100;
            #100 //800ms
            A = -8'd13;
            B=  8'd12;
            #100 //900ms
            A = -8'd13;
```

```
        B=  -8'd13;
    end
endmodule
```

Appendix B: Captures of the output screens and simulation waveforms.

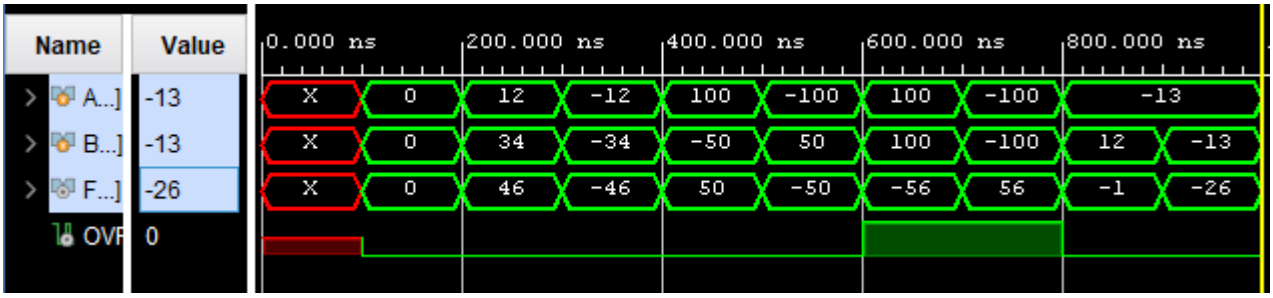

Figure 1: Test Bench WaveForm Simulation

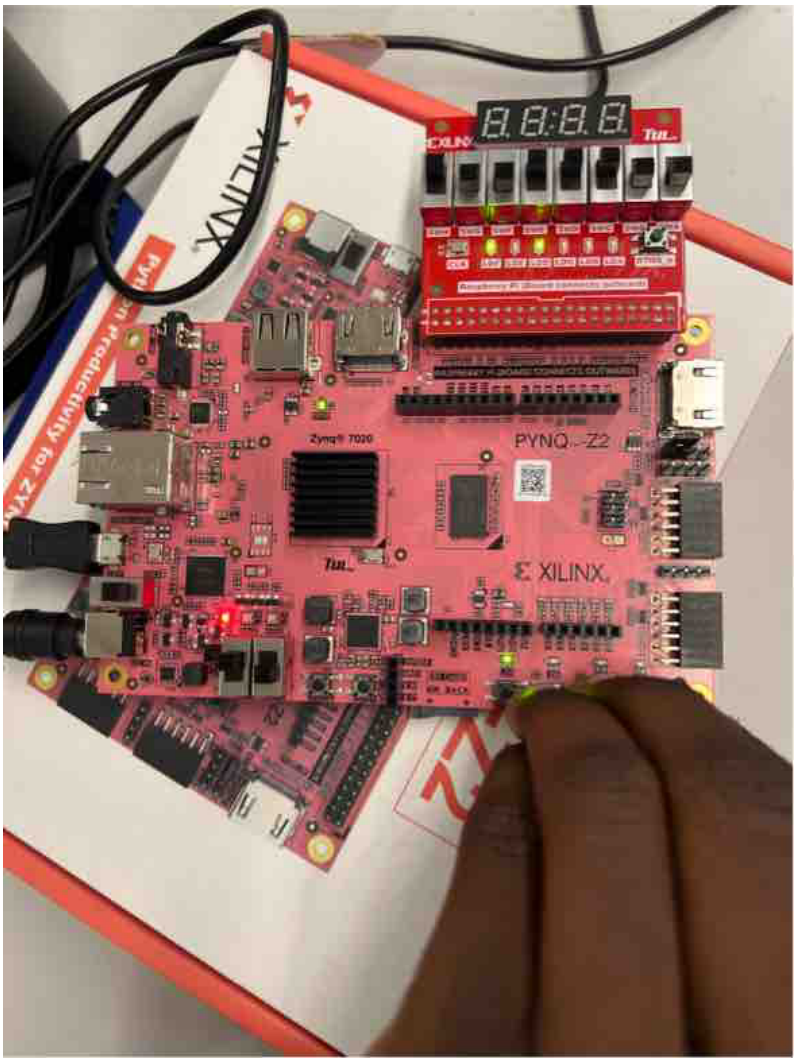

Figure 2: LED Results