Name: Alexander Wise **Lab 1 – 16-Bit Adder**

Instructor Name: Professor Consi **January 23, 2024**

Background

System Verilog is a vital Hardware Description Language for modeling digital circuits through code and serves as the base of many embedded systems. Previously, we designed a full adder through digital circuits. The purpose of this lab was to expand on the previously designed full-adder to design a 16-bit adder. Similarly, this lab served as an introduction to designed circuits entirely in System Verilog which will be used extensively in later labs.

Pre-Lab

1.

a	b	f	ovf
16'h25	16'h45	16'h6A	0
16'hA415	16'hA555	16'h496A	1
16'hF115	16'hF215	16'hE32A	0
16'h9D00	16'h9E00	16'h3B00	1
16'hED00	16'hEF03	16'hDC03	0
16'h8A10	16'h7110	16'hFB20	0
16'h21AA	16'h21BB	16'h4365	0
16'h9A00	16'h9F4F	16'h394F	1

Table 1: Table of expected outputs for the 16-bit adder

2.

ovf = (a[15] & b[15] & !f[15]) | (!a[15] & !b[15] & f[15])

This works by checking the 15^{th} bit on each number. If the 15^{th} bit of a and b are both equal to 1 and the 15^{th} bit of f is equal to 0, then there is an overflow. Similarly, if a and b are both equal to 0 and f is equal to 1, then there is also an overflow.

3. No, the prelab test cases do not test every input and output bit. The cases for when a and b are both equal to -1, when both bit are 0, and when both bits are the maximum value represented with 16-bits. These cases are listed below:

a	b
16'hFFFF	16'hFFFF
16'h0000	16'h0000
16'h7FFF	16'h7FFF

Table 2: Table of test cases not considered in Table 1

Design Implementation

3.2 Entering Your Design:

First, System Verilog code was written for the 16 bit adder. The adder had two input ports: a, and b, and two output ports: f, and ovf (overflow). How the 16-bit adder works is by adding a and b together and the value of this operation is outputted through f. Using the formula for ovf found in the pre-lab, the code can calculate whether or not there is an overflow. The code for this can be seen in Figure 1 found in Appendix A.

3.3 Simulating Your Design

To simulate the System Verilog code developed in the previous part, a test bench module was created. This module was developed using the test cases from the pre-lab as well as the other test cases not considered in the initial table. The code for this can be seen in Figure 2 found in Appendix A.

After the test bench was run, a waveform was created showing the outputs for each test case. This waveform can be seen in Figure 3 in Appendix A

3.4 Testing in Hardware

Utilizing the PYNQ-Z2, the code was tested on hardware. The output values were displayed using the 7-segment display module that is attached to the board through pins. Examples of test cases run on the board can be found in Appendix B.

Conclusion

Across the lab, we learned how to apply System Verilog code to hardware, utilize test cases to find the correct outputs and code, and broaden our knowledge and understanding of the Vivado IDE. Based on the results of this procedure, a large number of test cases is generally a good idea as it allows the programmer to ensure the desired results and catch any possible errors in the code.

Appendix A: Xilinx Vivado Screen Captures

```
module sixteenbit_adder(
    input [15:0] a,
    input [15:0] b,
    output [15:0] f,
    output ovf
    );

    assign f = a + b;
    assign ovf = (a[15] & b[15] & !f[15]) | (!a[15] & !b[15] & f[15]);
endmodule
```

Figure 1: System Verilog code for the 16-bit adder

```
14      // Dependencies:
15      //
16      // Revision:
17      // Revision 0.01 - File Created
18      // Additional Comments:
19      //
20      //////////////////////////////////////////////////////////////////////////////////
21
22
23      module adder16_tb();
24      logic [15:0] a;
25      logic [15:0] b;
26      logic [15:0] f;
27      logic ovf;
28
29      sixteenbit_adder UUT(.a(a), .b(b), .f(f), .ovf(ovf));
30
31      initial begin
32          # 100 a = 16'h25; b = 16'h45;
33          # 100 a = 16'hA415; b = 16'hA555;
34          # 100 a = 16'hF115; b = 16'hF215;
35          # 100 a = 16'h9D00; b = 16'h9E00;
36          # 100 a = 16'hED00; b = 16'hEF03;
37          # 100 a = 16'h8A10; b = 16'h7110;
38          # 100 a = 16'h21AA; b = 16'h21BB;
39          # 100 a = 16'h9A00; b = 16'h9F4F;
40          # 100 a = 16'hFFFF; b = 16'hFFFF;
41          # 100 a = 16'h0000; b = 16'h0000;
42          # 100 a = 16'h7FFF; b = 16'h7FFF;
43      end
44
45      endmodule
46
```

Figure 2: Testbench module for running pre-lab testcases.

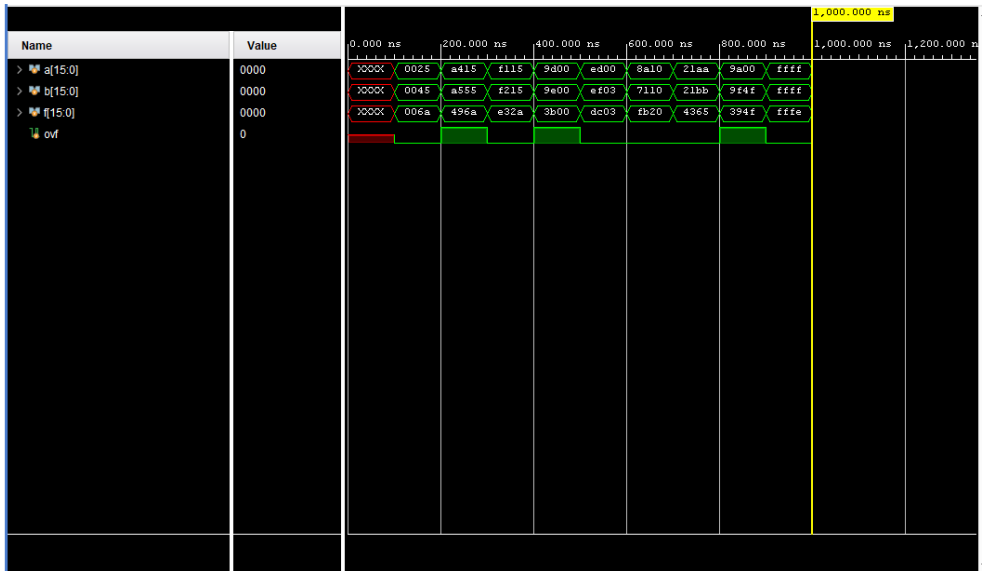

Figure 3: Waveform of testcases and overflow values

Appendix B: Hardware Images

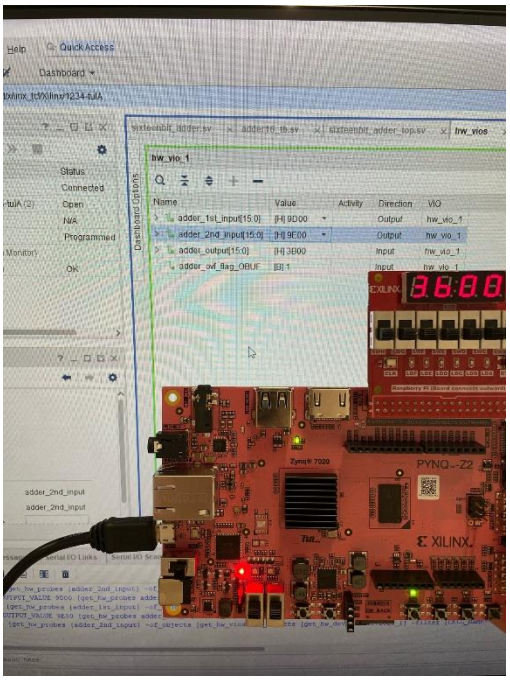

Figure 4: Hardware test showing a case with overload

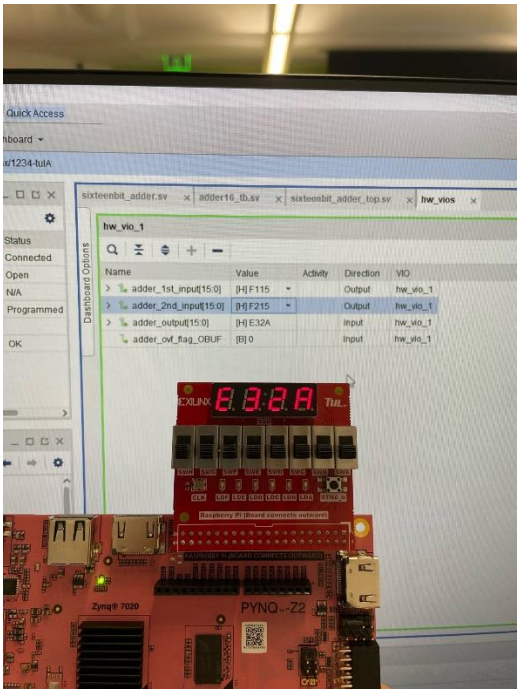

Figure 5: Hardware test showing a case without overload