**Lab Assignment 3**

Due: 09 Oct 2020

By: Alex Oswald

EECE 2323

**Introduction**

During this lab experiment an arithmetic logic unit (ALU) was designed and tested. It contained three inputs: a a number, b a number, and sel a code to select what operation is performed. It contained three outputs: f an 8-bit function output, ovf an overflow output, and take\_branch a 1-bit comparative output. The following are descriptions of the code for the ALU as well as the testbench that supported it.

**Verilog Code for Complete ALU:**

1. `timescale 1ns / 1ps
3. module eightbit\_alu(
4. input [7:0] a,
5. input [7:0] b,
6. input [2:0] sel,
7. output reg [7:0] f,
8. output reg ovf,
9. output reg take\_branch);
11. always @(a or b or sel)
12. **case**(sel)
13. 3'b000: begin
14. f = a + b;
15. ovf = (a[7] & b[7] & ~f[7]) | (~a[7] & ~b[7] & f[7]);
16. take\_branch = 0;
17. end
18. 3'b001: begin
19. f = ~b;
20. ovf = 0;
21. take\_branch = 0;
22. end
23. 3'b010: begin
24. f = a & b;
25. ovf = 0;
26. take\_branch = 0;
27. end
28. 3'b011: begin
29. f = a | b;
30. ovf = 0;
31. take\_branch = 0;
32. end
33. 3'b100: begin
34. f = a >>> 1;
35. ovf = 0;
36. take\_branch = 0;
37. end
38. 3'b101: begin
39. f = a << 1;
40. ovf = 0;
41. take\_branch = 0;
42. end
43. 3'b110:begin
44. f = 0;
45. ovf = 0;
46. take\_branch = (a == b);
47. end
48. 3'b111: begin
49. f = 0;
50. ovf = 0;
51. take\_branch = (a != b);
52. end
53. **default**: begin
54. f=0;
55. ovf = 0;
56. take\_branch = 0;
57. end
58. endcase
59. endmodule

**Verilog Code for Testbench:**

1. `timescale 1ns / 10ps
3. module alu\_unit\_tb ();
4. // Inputs
5. reg **signed** [7:0] a;
6. reg **signed** [7:0] b;
7. reg [2:0] sel;
8. // Outputs
9. wire **signed** [7:0] f;
10. wire ovf;
11. wire take\_branch;
13. // Initiate the Unit Under Test (UUT)
14. eightbit\_alu uut (
15. .a(a),
16. .b(b),
17. .sel(sel),
18. .f(f),
19. .ovf(ovf),
20. .take\_branch(take\_branch));
22. // Initialize Inputs (stimulus)
23. initial
24. begin
25. #0
26. sel = 0;
27. repeat (8)
28. begin
29. a  = -8'd1; b = -8'd1;      #10;
30. a = 8'd0;   b = 8'd0;       #10;
31. a = -8'd1;  b = 8'd0;       #10;
32. a = 8'd100; b = 8'd100;     #10;
33. a = -8'd69; b = -8'd115;    #10;
34. sel = sel + 1;
35. end
36. end
38. endmodule

**Waveform Simulation:**

Graphical user interface

Description automatically generated

The above figure demonstrates the testbench simulation data from the ALU. As can be seen, there is only overflow present in the first test selection which was an addition function as was programmed. Additionally, only the final two sel operation sectors contained take\_branch data which are inverse reflections of one another which is congruent with their functions.

**Conclusion**

During this lab assignment an arithmetic logic unit was constructed using Verilog and implemented on an FPGA board. This assignment demonstrated some of the functions of one of the central parts of a MIPS 8-bit processor. It succeeded in designing this operator through a case statement structure which was then tested successfully with a repetition-based testbench that moved through each select operator. To extend academic inquiry, one might examine another component of the MIPS 8-bit processor.

**Video evidence of this program implemented on hardware is attached with the submission.**

**\*\*UPDATE: I successfully configured the vio\_0 module. However, during bitstream generation there was an error that failed to set the proper number of bits for *probe\_in0[0:0]*, and as a result I couldn’t receive any virtual outputs. I tried to correct it for like 5 hours but unfortunately was not able to resolve the error despite my best efforts (which can be noted by the fact that I’m on my fifth iteration of recreating and attempting the project file in lab3proj5.xpr). Screenshots of what I did accomplish are in the screenshots below.**

Graphical user interface, text, application

Description automatically generated  
Figure 1: Filing inputs to add , and a demonstration of the improper bits listed for any of the vio/probe\_inX signals.

Graphical user interface, text, application

Description automatically generated  
Figure 2: Refreshing the previous inputs to the system, and the resulting broken output as seen in *vio/probe\_inX[0:0]* all listing [B] 1 as their resultant.

Graphical user interface, text, application, email

Description automatically generated  
Figure 3: Diagnosing the error, synthesis being unable to connect pin *probe\_in0 1* and *2*.

Graphical user interface, text, application, email

Description automatically generated  
Figure 4: Proof of the proper vio\_0 module definition.