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.

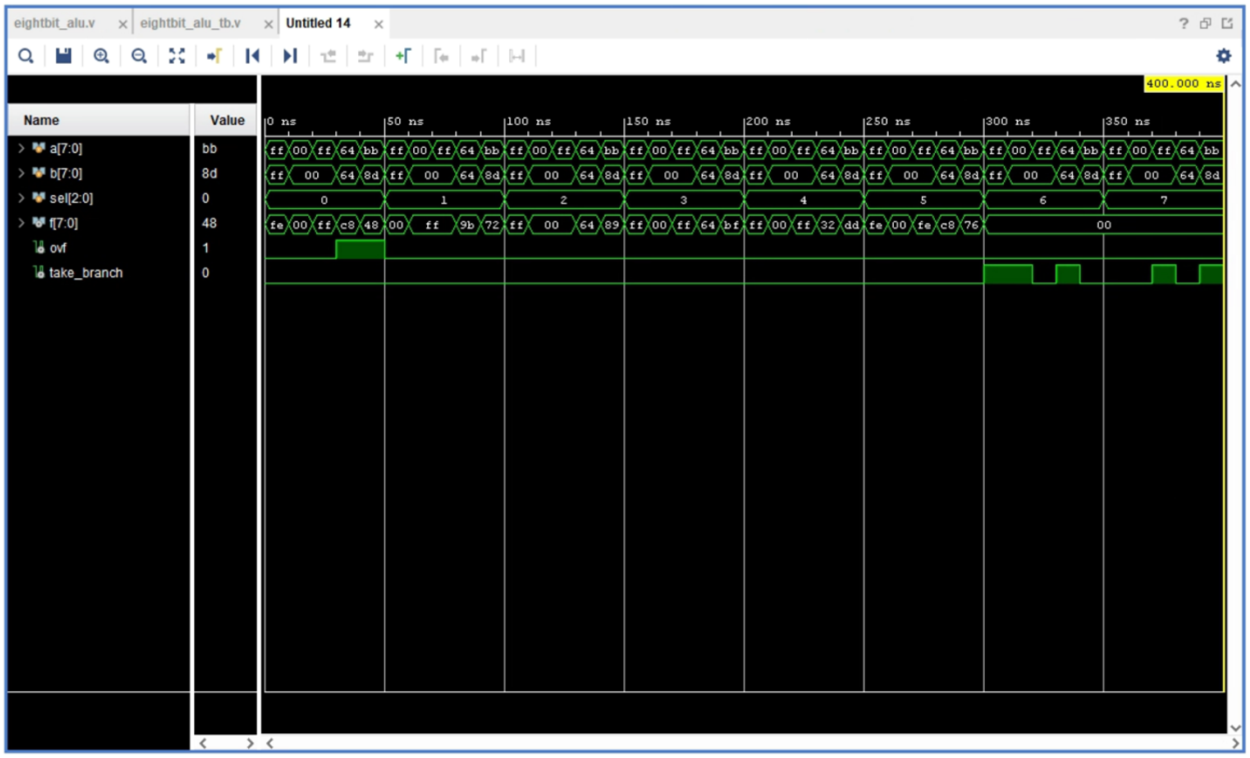
Verilog Code for Complete ALU:

```verilog
`timescale 1ns / 1ps

module eightbit_alu(
    input [7:0] a,
    input [7:0] b,
    input [2:0] sel,
    output reg [7:0] f,
    output reg ovf,
    output reg take_branch);

    always @(a or b or sel)
        case(sel)
            3'b000: begin
                f = a + b;
                ovf = (a[7] & b[7] & ~f[7]) | (~a[7] & ~b[7] & f[7]);
                take_branch = 0;
                end
            3'b001: begin
                f = ~b;
                ovf = 0;
                take_branch = 0;
                end
            3'b010: begin
                f = a & b;
                ovf = 0;
                take_branch = 0;
                end
            3'b011: begin
                f = a | b;
                ovf = 0;
                take_branch = 0;
                end
            3'b100: begin
                f = a >>> 1;
                ovf = 0;
                take_branch = 0;
                end
            3'b101: begin
                f = a << 1;
                ovf = 0;
                take_branch = 0;
                end
            3'b110:begin
                f = 0;
                ovf = 0;
                take_branch = (a == b);
                end
            3'b111: begin
                f = 0;
                ovf = 0;
                take_branch = (a != b);
                end
            default: begin
                f=0;
                ovf = 0;
                take_branch = 0;
                end
        endcase
endmodule
```

Verilog Code for Testbench:

```
`timescale 1ns / 10ps

module alu_unit_tb ();
    // Inputs
    reg signed [7:0] a;
    reg signed [7:0] b;
    reg [2:0] sel;
    // Outputs
    wire signed [7:0] f;
    wire ovf;
    wire take_branch;

    // Initiate the Unit Under Test (UUT)
    eightbit_alu uut (
        .a(a),
        .b(b),
        .sel(sel),
        .f(f),
        .ovf(ovf),
        .take_branch(take_branch));

    // Initialize Inputs (stimulus)
    initial
    begin
        #0
        sel = 0;
        repeat (8)
        begin
            a  = -8'd1; b = -8'd1;      #10;
            a = 8'd0;   b = 8'd0;       #10;
            a = -8'd1;  b = 8'd0;       #10;
            a = 8'd100; b = 8'd100;     #10;
            a = -8'd69; b = -8'd115;    #10;
            sel = sel + 1;
        end
    end

endmodule
```

Waveform Simulation:



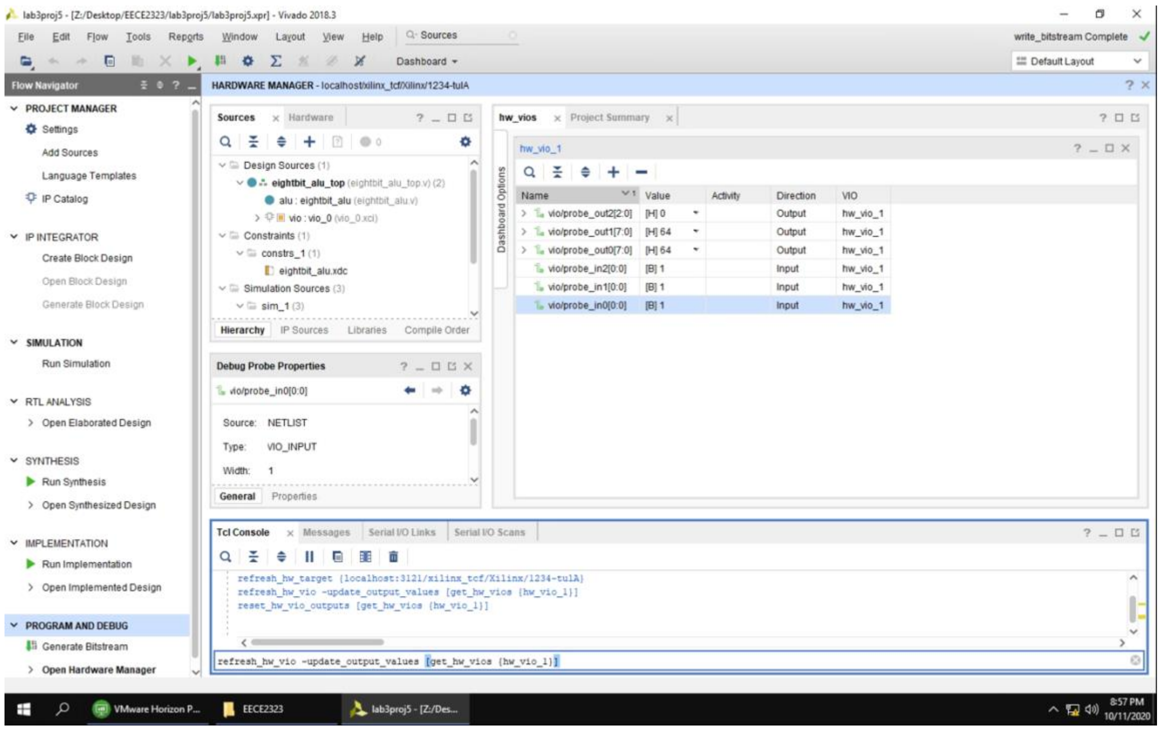
The above figure demonstrates the testbench simulation data from the ALU. As can be seen, there is only overflow present in the first test selection which was an addition function as was programmed. Additionally, only the final two sel operation sectors contained take_branch data which are inverse reflections 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.**



lab3proj5 - [Z:/Desktop/EECE2323/lab3proj5/lab3proj5.xpr] - Vivado 2018.3

HARDWARE MANAGER - localhost/xilinx_tcf/Xilinx/1234-tulA

hw_vio_1

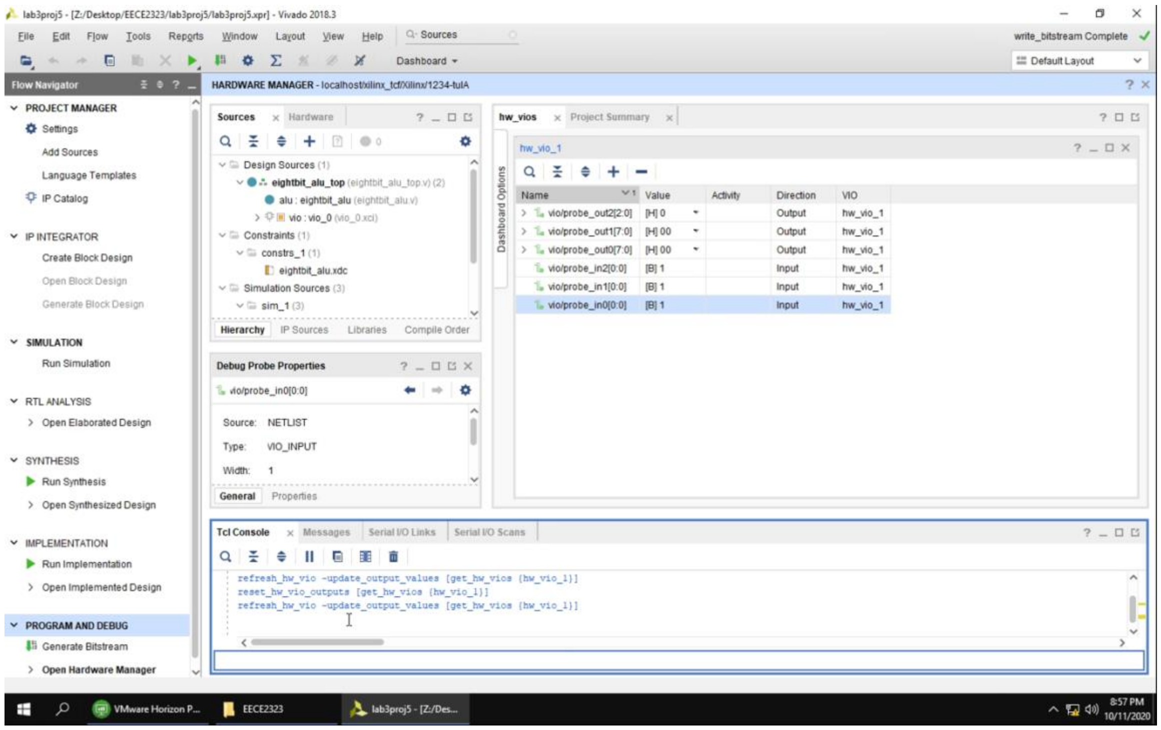
Name	Value	Activity	Direction	VIO
vio/probe_out2[2:0]	[H] 0		Output	hw_vio_1
vio/probe_out1[7:0]	[H] 64		Output	hw_vio_1
vio/probe_out0[7:0]	[H] 64		Output	hw_vio_1
vio/probe_in2[0:0]	[B] 1		Input	hw_vio_1
vio/probe_in1[0:0]	[B] 1		Input	hw_vio_1
vio/probe_in0[0:0]	[B] 1		Input	hw_vio_1

Debug Probe Properties: vio/probe_in0[0:0]
Source: NETLIST
Type: VIO_INPUT
Width: 1

```
refresh_hw_target {localhost:3121/xilinx_tcf/Xilinx/1234-tulA}
refresh_hw_vio -update_output_values [get_hw_vios {hw_vio_1}]
reset_hw_vio_outputs [get_hw_vios {hw_vio_1}]
```

```
refresh_hw_vio -update_output_values [get_hw_vios {hw_vio_1}]
```

Figure 1: Filing inputs to add $100_{10} + 100_{10}$, and a demonstration of the improper bits listed for any of the vio/probe_inX signals.



lab3proj5 - [Z:/Desktop/EECE2323/lab3proj5/lab3proj5.xpr] - Vivado 2018.3

HARDWARE MANAGER - localhost/xilinx_tcf/Xilinx/1234-tulA

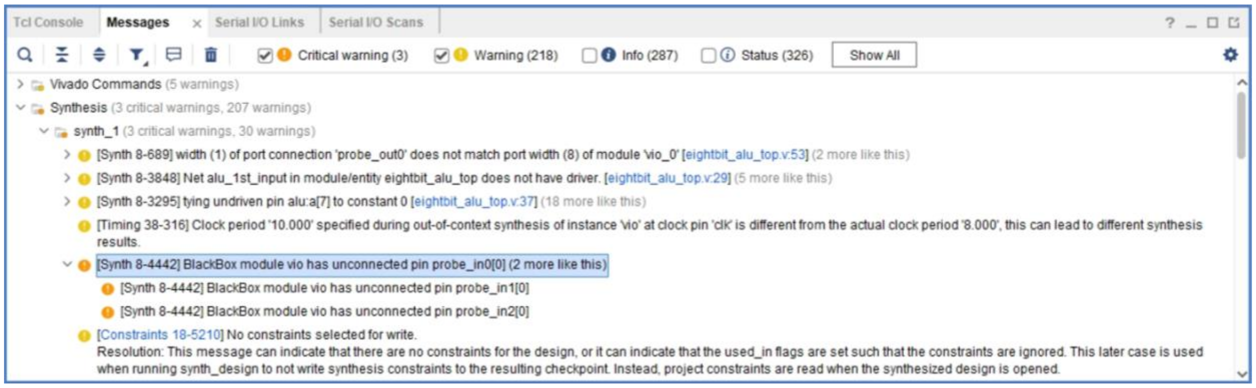
hw_vio_1

Name	Value	Activity	Direction	VIO
vio/probe_out2[2:0]	[H] 0		Output	hw_vio_1
vio/probe_out1[7:0]	[H] 00		Output	hw_vio_1
vio/probe_out0[7:0]	[H] 00		Output	hw_vio_1
vio/probe_in2[0:0]	[B] 1		Input	hw_vio_1
vio/probe_in1[0:0]	[B] 1		Input	hw_vio_1
vio/probe_in0[0:0]	[B] 1		Input	hw_vio_1

Debug Probe Properties: vio/probe_in0[0:0]
Source: NETLIST
Type: VIO_INPUT
Width: 1

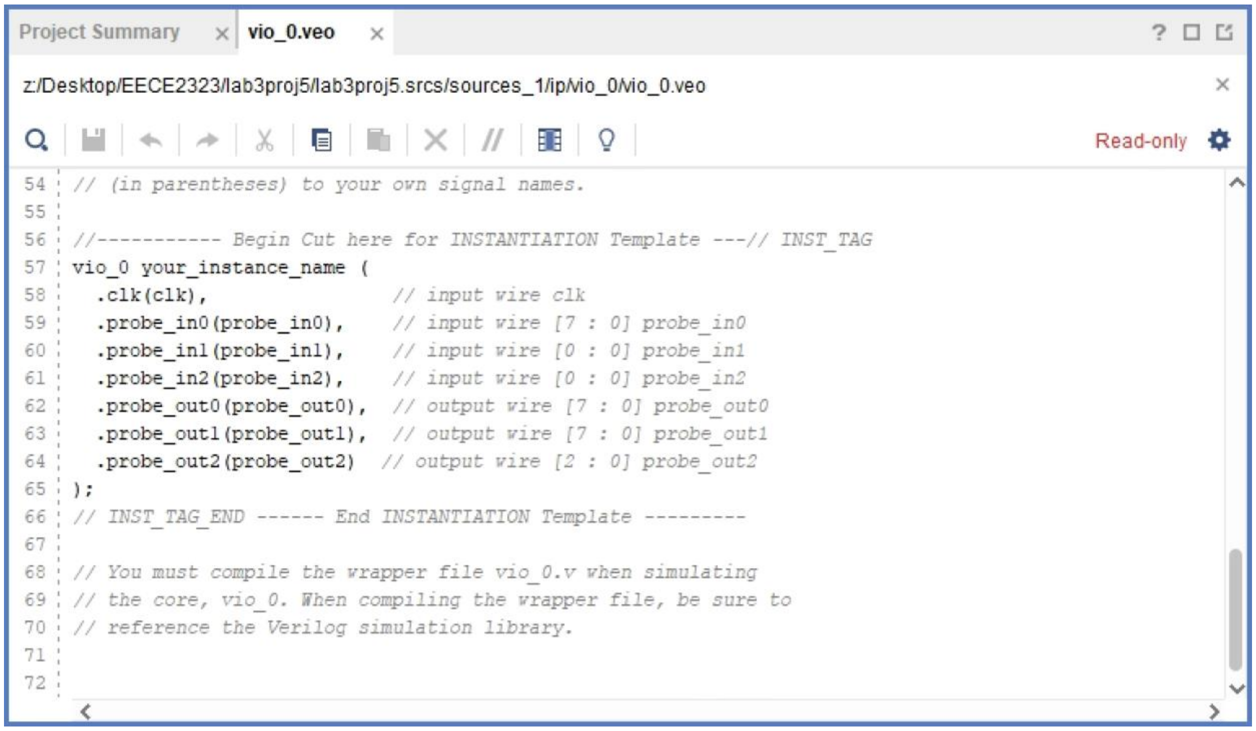
```
refresh_hw_vio -update_output_values [get_hw_vios {hw_vio_1}]
reset_hw_vio_outputs [get_hw_vios {hw_vio_1}]
refresh_hw_vio -update_output_values [get_hw_vios {hw_vio_1}]
```

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.



```
> Vivado Commands (5 warnings)
v Synthesis (3 critical warnings, 207 warnings)
  v synth_1 (3 critical warnings, 30 warnings)
    > [Synth 8-689] width (1) of port connection 'probe_out0' does not match port width (8) of module 'vio_0' [eightbit_alu_top.v:53] (2 more like this)
    > [Synth 8-3848] Net alu_1st_input in module/entity eightbit_alu_top does not have driver. [eightbit_alu_top.v:29] (5 more like this)
    > [Synth 8-3295] tying undriven pin alu:a[7] to constant 0 [eightbit_alu_top.v:37] (18 more like this)
      [Timing 38-316] Clock period '10.000' specified during out-of-context synthesis of instance 'vio' at clock pin 'clk' is different from the actual clock period '8.000', this can lead to different synthesis results.
    v [Synth 8-4442] BlackBox module vio has unconnected pin probe_in0[0] (2 more like this)
        [Synth 8-4442] BlackBox module vio has unconnected pin probe_in1[0]
        [Synth 8-4442] BlackBox module vio has unconnected pin probe_in2[0]
      [Constraints 18-5210] No constraints selected for write.
      Resolution: This message can indicate that there are no constraints for the design, or it can indicate that the used_in flags are set such that the constraints are ignored. This later case is used when running synth_design to not write synthesis constraints to the resulting checkpoint. Instead, project constraints are read when the synthesized design is opened.
```

Figure 3: Diagnosing the error, synthesis being unable to connect pin *probe_in0 1* and *2*.



z:/Desktop/EECE2323/lab3proj5/lab3proj5.srcs/sources_1/ip/vio_0/vio_0.veo

```verilog
// (in parentheses) to your own signal names.

//----------- Begin Cut here for INSTANTIATION Template ---// INST_TAG
vio_0 your_instance_name (
  .clk(clk),                // input wire clk
  .probe_in0(probe_in0),    // input wire [7 : 0] probe_in0
  .probe_in1(probe_in1),    // input wire [0 : 0] probe_in1
  .probe_in2(probe_in2),    // input wire [0 : 0] probe_in2
  .probe_out0(probe_out0),  // output wire [7 : 0] probe_out0
  .probe_out1(probe_out1),  // output wire [7 : 0] probe_out1
  .probe_out2(probe_out2)  // output wire [2 : 0] probe_out2
);
// INST_TAG_END ------ End INSTANTIATION Template ---------

// You must compile the wrapper file vio_0.v when simulating
// the core, vio_0. When compiling the wrapper file, be sure to
// reference the Verilog simulation library.
```

Figure 4: Proof of the proper vio_0 module definition.