Advanced SoC Design Report

Student ID: 311511022

Lab #1 FSIC-SIM

1. Show the code that you use to program configuration address [‘h3000_5000]

```
task user_project_select;
    input [4:0] up_index;
        begin
            @(posedge soc_coreclk);
            wbs_adr <= 32'h3000_5000;
            wbs_wdata <= {27'd0, up_index};
            wbs_sel <= 4'b1111;
            wbs_cyc <= 1'b1;
            wbs_stb <= 1'b1;
            wbs_we <= 1'b1;
            @(posedge soc_coreclk);
            while(wbs_ack==0) begin
                    @(posedge soc_coreclk);
            end
            $display($time, "=> user_project %d is enable", up_i
        end
endtask
```

2. Explain why “By programming configuration address [‘h3000_5000], signal user_prj_sel[4:0] will change accordingly”?

The modules `axil_slav` , `axil_mstr` , `axis_slav` , `iqr_mux` and `la_mux` mux the signals from `user_prj0` to `user_prj3` . The select signal is `user_prj_sel` which is connect to the memory map register of `config_ctrl` output, and the address is `0x3000_5000` . The follow is the corresponding code in `config_ctrl.v` .

```
always @ ( posedge axi_clk or negedge axi_reset_n ) begin
    if ( !axi_reset_n ) begin
        user_prj_sel_o <= 5'b0;
    end else begin
        if ( cc_axi_awvalid && cc_axi_wvalid ) begin
            if (axi_awaddr[11:0] == 12'h000 && (axi_wstrb[0] ==
                user_prj_sel_o <= axi_wdata[4:0];
            end else begin
                user_prj_sel_o <= user_prj_sel_o;
            end
        end
    end
end
```

The `cc_axi_awvalid` and `cc_axi_wvalid` is activated by `cc_enable <= ( m_axi_request_add[31:12] == 20'h30005 )? 1'b1 : 1'b0` .

3. Briefly describe how you do FIR initialization (tap parameter, length) from SOC side (Test#1).

```
task firTest_initFromSoc;
    begin
        $display("----Start the coefficient input(AXI-lite)----'
        soc_up_cfg_write(FIR_LEN_OFFSET, 4'b1111, data_length);
        for (i=0; i<Tape_Num; i=i+1) begin
            soc_up_cfg_write(FIR_TAP_OFFSET+4*i, 4'b1111, coef[
        end
        $display(" Tape programming done ...");
        $display(" Start FIR");
        @(posedge soc_coreclk) soc_up_cfg_write(FIR_CTRL_OFFSET,
        $display("----End the coefficient input(AXI-lite)----");
        soc_to_fpga_axis_captured_count = 0;
    end
endtask
```

Use the task `soc_up_cfg_write` to program the configuration in the user project from SoC side. And the address of configuration registers is defined with `FIR_LEN_OFFSET = 8'h10` and `parameter FIR_TAP_OFFSET = 8'h40` .

4. Briefly describe how you do FIR initialization (tap parameter, length) from FPGA side (Test#2).

```
task firTest_initFromFPGA;
    begin
        $display("----Start the coefficient input(AXI-lite)----"
        fpga_axilite_write(FPGA_to_SOC_UP_BASE+FIR_LEN_OFFSET,
        for (i=0; i<Tape_Num; i=i+1) begin
            fpga_axilite_write(FPGA_to_SOC_UP_BASE+FIR_TAP_OFFSE
        end
        $display(" Tape programming done ...");
        $display(" Start FIR");
        @(posedge soc_coreclk) soc_up_cfg_write(FIR_CTRL_OFFSET,
        $display("----End the coefficient input(AXI-lite)----");
        soc_to_fpga_axis_captured_count = 0;
    end
endtask
```

Use the task `fpga_axilite_write` to program the configuration in the user project from FPGA side. And the address of configuration registers is defined with `localparam FPGA_to_SOC_UP_BASE=28'h000_0000` , `FIR_LEN_OFFSET = 8'h10` and `parameter FIR_TAP_OFFSET = 8'h40` .

5. Briefly describe how you feed in X data from FPGA side.

```
// Read simulation data from file
initial begin
    data_length = 0;
    Din = $fopen("./pattern/samples_triangular_wave.dat","r");
    golden = $fopen("./pattern/out_gold.dat","r");
    for(m=0;m<Data_Num;m=m+1) begin
        input_data = $fscanf(Din,"%d", firDinList[m]);
        golden_data = $fscanf(golden,"%d", firGoldenList[m]);
        data_length = data_length + 1;
    end
end

task firTest_xStreamIn;
    $display("----Start FIR data input(AXI-Stream)----");
    for(i=0; i<DATA_LENGTH; i=i+1) begin
        fpga_axis_req(firDinList[i], TID_UP_UP, 0);
    end
    $display("------End FIR data input(AXI-Stream)------");
endtask
```

First, read the data from prepared file. Feed the data by task `fpga_axis_req` in a for loop.

6. Briefly describe how you get output Y data in testbench, and how to do comparison with golden values.

The AXI-stream signal from SoC to FPGA is auto captured by the followimg code:

```
initial begin          //get upstream soc_to_fpga_axis - for lo
    soc_to_fpga_axis_captured_count = 0;
    soc_to_fpga_axis_event_triggered = 0;
    while (1) begin
    @(posedge fpga_coreclk);
        if (fpga_is_as_tvalid == 1 && fpga_is_as_tid == TID_UP_
            $display($time, "=> get soc_to_fpga_axis be : soc_t
            soc_to_fpga_axis_captured[soc_to_fpga_axis_captured_
            $display($time, "=> get soc_to_fpga_axis af : soc_t
            soc_to_fpga_axis_captured_count = soc_to_fpga_axis_
        end
        if ( (soc_to_fpga_axis_captured_count == fpga_axis_test_
            $display($time, "=> soc_to_fpga_axis_captured : sen
            #0 -> soc_to_fpga_axis_event;
            soc_to_fpga_axis_event_triggered = 1;
        end
        if (soc_to_fpga_axis_captured_count != fpga_axis_test_le
            soc_to_fpga_axis_event_triggered = 0;
    end
end
```

The `ss_tdata` which is the axis data from FIR is stored in `soc_to_fpga_axis_captured[soc_to_fpga_axis_captured_count][31:0]` , and the following code compare the output with the golden answer.

```
task firTest_checkAnswer;
    reg signed [31:0] firDout;
    begin
        error = 0;
        for(k=0; k<data_length; k=k+1) begin
            firDout = soc_to_fpga_axis_captured[k][31:0];
            if (soc_to_fpga_axis_captured[k][31:0] != firGoldenI
                $display("[ERROR] [Pattern %d] Golden answer: %(
                error <= 1;
                end else begin
                    $display("[PASS] [Pattern %d] Golden answer:
            end
        end
        if (error == 0) begin
            $display("----------------------------------------
            $display("-----------Congratulations! Pass---------
        end else begin
            $display("--------Simulation Failed---------");
        end
    end
endtask
```

7. Screenshot simulation results printed on screen, to show that your Test#1 & Test#2 complete successfully

Test#1

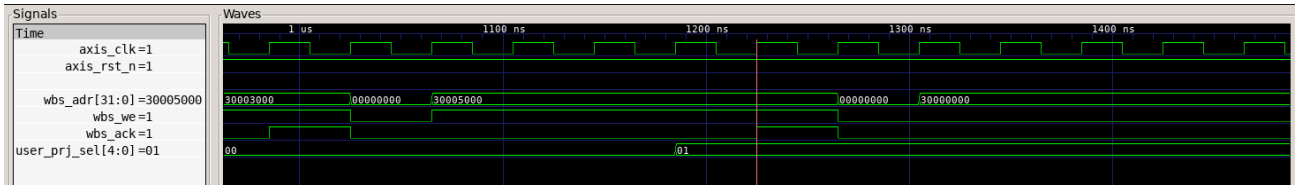
```
[PASS] [Pattern        590] Golden answer:      -2562, Your answer:      -2562
[PASS] [Pattern        591] Golden answer:      -2379, Your answer:      -2379
[PASS] [Pattern        592] Golden answer:      -2196, Your answer:      -2196
[PASS] [Pattern        593] Golden answer:      -2013, Your answer:      -2013
[PASS] [Pattern        594] Golden answer:      -1830, Your answer:      -1830
[PASS] [Pattern        595] Golden answer:      -1647, Your answer:      -1647
[PASS] [Pattern        596] Golden answer:      -1464, Your answer:      -1464
[PASS] [Pattern        597] Golden answer:      -1281, Your answer:      -1281
[PASS] [Pattern        598] Golden answer:      -1098, Your answer:      -1098
[PASS] [Pattern        599] Golden answer:       -915, Your answer:       -915
----------------------------------------
----------Congratulations! Pass----------
```

Test#2

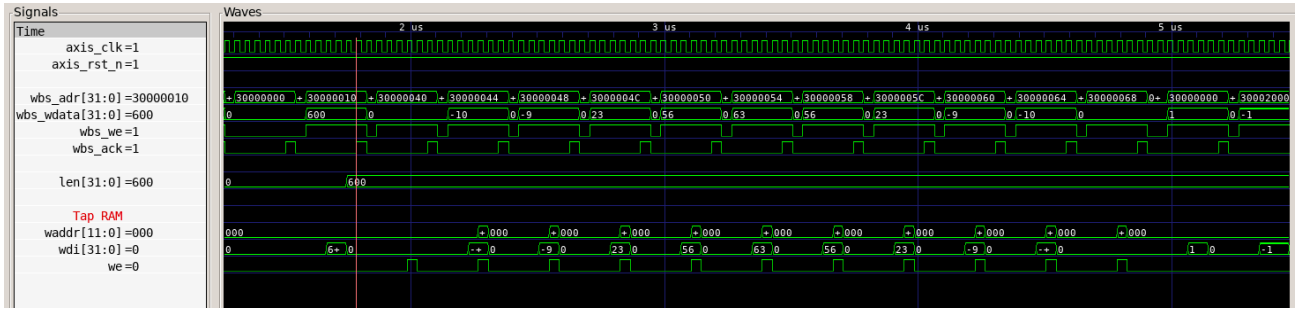
```
[PASS] [Pattern  590] Golden answer:  -2562, Your answer:  -2562
[PASS] [Pattern  591] Golden answer:  -2379, Your answer:  -2379
[PASS] [Pattern  592] Golden answer:  -2196, Your answer:  -2196
[PASS] [Pattern  593] Golden answer:  -2013, Your answer:  -2013
[PASS] [Pattern  594] Golden answer:  -1830, Your answer:  -1830
[PASS] [Pattern  595] Golden answer:  -1647, Your answer:  -1647
[PASS] [Pattern  596] Golden answer:  -1464, Your answer:  -1464
[PASS] [Pattern  597] Golden answer:  -1281, Your answer:  -1281
[PASS] [Pattern  598] Golden answer:  -1098, Your answer:  -1098
[PASS] [Pattern  599] Golden answer:   -915, Your answer:   -915
------------------------------------------------

----------Congratulations! Pass----------
```

8. Screenshot simulation waveform

8-1. Configuration cycle (when we program [‘h3000_5000] = 32’h01, signal user_prj_sel changes accordingly)



8-2. AXI-Lite transaction cycles (feed in tap parameters, data_length)



8-3. Stream-in, Stream-out