ASoc Lab1-1

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

```
668                 task user_prj1_cfg; //
669                 /////input [11:0] offset;            //4K range
670                 /////input [3:0] sel;
671                 input [31:0] data;
672
673                         begin
674                                 @ (posedge soc_coreclk);
675                                 wbs_adr <= 32'h3000_5000;
676                                 //wbs_adr[11:2] <= offset[11:2];      //only provide DW address
677
678                                 wbs_wdata <= data;
679                                 //wbs_sel <= sel;
680                                 wbs_sel <= 4'b1111;
681                                 wbs_cyc <= 1'b1;
682                                 wbs_stb <= 1'b1;
683                                 wbs_we <= 1'b1;
684
685                                 @(posedge soc_coreclk);
686                                 while(wbs_ack==0) begin
687                                         @(posedge soc_coreclk);
688                                 end
689
690                                 $display($time, "=> user_prj1_cfg : wbs_adr=%x, wbs_sel=%b, wbs_wdata=%x", wbs_adr, wbs_sel, wbs_wdata);
691                         end
692                 endtask
```

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

```
668                 task user_prj1_cfg;
669                 //input [11:0] offset;        //4K range
670                 //input [3:0] sel;
671                 input [31:0] data;
672
673                         begin
674                                 @ (posedge soc_coreclk);
675                                 wbs_adr <= 32'h3000_5000;
676                                 //wbs_adr[11:2] <= offset[11:2];      //only provide DW address
677
678                                 wbs_wdata <= data;
679                                 //wbs_sel <= sel;
680                                 wbs_sel <= 4'b1111;
681                                 wbs_cyc <= 1'b1;
682                                 wbs_stb <= 1'b1;
683                                 wbs_we <= 1'b1;
684
685                                 @(posedge soc_coreclk);
686                                 while(wbs_ack==0) begin
687                                         @(posedge soc_coreclk);
688                                 end
689
690                                 $display($time, "=> user_prj1_cfg : wbs_adr=%x, wbs_sel=%b, wbs_wdata=%x", wbs_adr, wbs_sel, wbs_wdata);
691                         end
692                 endtask
```

```
1665=> user_prj1_cfg : wbs_adr=30005000, wbs_sel=1111, wbs_wdata=00000001
```

User_prj_sel 預設為 0,透過 program 32’d1(wbs_wdata=00000001)會將 user_prj_sel 變為 user_prj1.

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

```
512
513                         //4.Program data length
514                         soc_up_cfg_write(12'h10, 4'b1111, DATA_LENGTH);
515
516                         //5. Program tap parameters
517                         soc_up_cfg_write(12'h20, 4'b1111, 32'd0);
518                         soc_up_cfg_write(12'h24, 4'b1111, -32'd10);
519                         soc_up_cfg_write(12'h28, 4'b1111, -32'd9);
520                         soc_up_cfg_write(12'h2C, 4'b1111, 32'd23);
521                         soc_up_cfg_write(12'h30, 4'b1111, 32'd56);
522                         soc_up_cfg_write(12'h34, 4'b1111, 32'd63);
523                         soc_up_cfg_write(12'h38, 4'b1111, 32'd56);
524                         soc_up_cfg_write(12'h3C, 4'b1111, 32'd23);
525                         soc_up_cfg_write(12'h40, 4'b1111, -32'd9);
526                         soc_up_cfg_write(12'h44, 4'b1111, -32'd10);
527                         soc_up_cfg_write(12'h48, 4'b1111, 32'd0);
528
529                         //6. Read back data length & check
530                         soc_UP_cfg_read_and_check(12'h10, 4'b1111, DATA_LENGTH);
531
532                         //7. Read back tap parameters & check
533                         soc_UP_cfg_read_and_check(12'h20, 4'b1111, 32'd0);
534                         soc_UP_cfg_read_and_check(12'h24, 4'b1111, -32'd10);
535                         soc_UP_cfg_read_and_check(12'h28, 4'b1111, -32'd9);
536                         soc_UP_cfg_read_and_check(12'h2C, 4'b1111, 32'd23);
537                         soc_UP_cfg_read_and_check(12'h30, 4'b1111, 32'd56);
538                         soc_UP_cfg_read_and_check(12'h34, 4'b1111, 32'd63);
539                         soc_UP_cfg_read_and_check(12'h38, 4'b1111, 32'd56);
540                         soc_UP_cfg_read_and_check(12'h3C, 4'b1111, 32'd23);
541                         soc_UP_cfg_read_and_check(12'h40, 4'b1111, -32'd9);
542                         soc_UP_cfg_read_and_check(12'h44, 4'b1111, -32'd10);
543                         soc_UP_cfg_read_and_check(12'h48, 4'b1111, 32'd0);
544
task soc_up_cfg_write;
        input [11:0] offset;          //4K range
        input [3:0] sel;
        input [31:0] data;

        begin
                @ (posedge soc_coreclk);
                wbs_adr <= UP_BASE;
                wbs_adr[11:2] <= offset[11:2];        //only provide DW address

                wbs_wdata <= data;
                wbs_sel <= sel;
                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, "=> soc_up_cfg_write : wbs_adr=%x, wbs_sel=%b, wbs_wdata=%x", wbs_adr, wbs_sel, wbs_wdata);
        end
endtask
```

利用 soc_up_cfg_write 這個 function 從 SoC side 將 tap parameter,length program 進去.

下圖為 soc_up_cfg_write 的 function.

會分別寫入 **tap parameter 和 data_length，先寫入 4. data_length 12'h10 後再寫入 tap parameter，這部分是 SoC side.**

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

```
//4.Program data length
FPGA_cfg_write(28'h10, DATA_LENGTH);

//5. Program tap parameters
FPGA_cfg_write(28'h20, 32'd0);
FPGA_cfg_write(28'h24, -32'd10);
FPGA_cfg_write(28'h28, -32'd9);
FPGA_cfg_write(28'h2C, 32'd23);
FPGA_cfg_write(28'h30, 32'd56);
FPGA_cfg_write(28'h34, 32'd63);
FPGA_cfg_write(28'h38, 32'd56);
FPGA_cfg_write(28'h3C, 32'd23);
FPGA_cfg_write(28'h40, -32'd9);
FPGA_cfg_write(28'h44, -32'd10);
FPGA_cfg_write(28'h48, 32'd0);

//6. Read back data length & check
FPGA_cfg_read_and_check(32'h10, DATA_LENGTH);

//7. Read back tap parameters & check
FPGA_cfg_read_and_check(32'h20, 32'd0);
FPGA_cfg_read_and_check(32'h24, -32'd10);
FPGA_cfg_read_and_check(32'h28, -32'd9);
FPGA_cfg_read_and_check(32'h2C, 32'd23);
FPGA_cfg_read_and_check(32'h30, 32'd56);
FPGA_cfg_read_and_check(32'h34, 32'd63);
FPGA_cfg_read_and_check(32'h38, 32'd56);
FPGA_cfg_read_and_check(32'h3C, 32'd23);
FPGA_cfg_read_and_check(32'h40, -32'd9);
FPGA_cfg_read_and_check(32'h44, -32'd10);
FPGA_cfg_read_and_check(32'h48, 32'd0);
```

```
//FPGA_cfg_write;
task FPGA_cfg_write;
        input [27:0] offset;
        input [31:0] data;

        begin
                @ (posedge fpga_coreclk);
                fpga_axilite_write_req(FPGA_to_SOC_UP_BASE + offset , 4'b0001, data);
                repeat(100) @ (posedge soc_coreclk);
        end
endtask
```

大致同 3 ,利用 FPGA_cfg_write 從 FPGA side 將 tap parameter,length program 進去.

下圖為 FPGA_cfg_wrrite 的 function.

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

```
764                     // FIR data X, Y stream data from FPGA side
765                     task Start_FIR_data_from_FPGA;
766                             soc_to_fpga_axis_expect_count = 0;
767                             GoldenZAxis();          //target to Axis Switch
768                             $display($time, "=> wait for soc_to_fpga_axis_event");
769             @(soc_to_fpga_axis_event);
770             $display($time, "=> soc_to_fpga_axis_expect_count = %d", soc_to_fpga_axis_expect_count);
771             $display($time, "=> soc_to_fpga_axis_captured_count = %d", soc_to_fpga_axis_captured_count);
772
773                             check_cnt = check_cnt + 1;
774                             if ( soc_to_fpga_axis_expect_count != DATA_LENGTH) begin
775                 $display($time, "=> [ERROR] soc_to_fpga_axis_expect_count = %d, soc_to_fpga_axis_captured_count = %d", soc_to_fpga_axis_expect_count, soc_to_fpga_axis_captured_count);
776                                     error_cnt = error_cnt + 1;
777                             end
778                             else
779                 $display($time, "=> [PASS] soc_to_fpga_axis_expect_count = %d, soc_to_fpga_axis_captured_count = %d", soc_to_fpga_axis_expect_count, soc_to_fpga_axis_captured_count);
780
781
782             for(j=0; j<DATA_LENGTH; j=j+1)begin
783                                 check_cnt = check_cnt + 1;
784                 if (soc_to_fpga_axis_expect_value[j] != soc_to_fpga_axis_captured[j] ) begin
785                     $display($time, "=> [ERROR] index(j)=%d, soc_to_fpga_axis_expect_value[%d] = %x, soc_to_fpga_axis_captured[%d]  = %x", j, j, soc_to_fpga_axis_expect_value[j], j,
    soc_to_fpga_axis_captured[j]);
786                                     error_cnt = error_cnt + 1;
787                             end
788                             else
789                      $display($time, "=> [PASS] index(j)=%d, soc_to_fpga_axis_expect_value[%d] = %x, soc_to_fpga_axis_captured[%d]  = %x", j, j, soc_to_fpga_axis_expect_value[j], j,
    soc_to_fpga_axis_captured[j]);
790
791                     end
792
793             //soc_to_fpga_axis_captured_count = 0;              //reset soc_to_fpga_axis_captured_count
794                             #200;
795                             $display("------Finish the data input(AXI-Stream ss) & data output(AXI-Stream sm) with checking------");
796                     endtask
```

這個 task 會開始 FIR 的 stream 以開始 data 的傳輸，將 X 從 0-63 經由 AXI-Stream 傳入.

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

```
                    /////// FIR data X, Y stream data from FPGA side ///////
                    task Start_FIR_data_from_FPGA_test2;
                        reset_capture_expect();
                                soc_to_fpga_axis_expect_count = 0;
                                Golden2Axis();          //target to Axis Switch
                                $display($time, "=> wait for soc_to_fpga_axis_event");
                    @(soc_to_fpga_axis_event);
                    $display($time, "=> soc_to_fpga_axis_expect_count = %d", soc_to_fpga_axis_expect_count);
                    $display($time, "=> soc_to_fpga_axis_captured_count = %d", soc_to_fpga_axis_captured_count);

                                check_cnt = check_cnt + 1;
                                if ( soc_to_fpga_axis_expect_count != DATA_LENGTH) begin
                        $display($time, "=> [ERROR] soc_to_fpga_axis_expect_count = %d, soc_to_fpga_axis_captured_count = %d", soc_to_fpga_axis_expect_count, soc_to_fpga_axis_captured_count);
                                    error_cnt = error_cnt + 1;
                                end
                                else
                        $display($time, "=> [PASS] soc_to_fpga_axis_expect_count = %d, soc_to_fpga_axis_captured_count = %d", soc_to_fpga_axis_expect_count, soc_to_fpga_axis_captured_count);


                    for(j=0; j<DATA_LENGTH; j=j+1)begin
                                    check_cnt = check_cnt + 1;
                        if (soc_to_fpga_axis_expect_value[j] != soc_to_fpga_axis_captured[j] ) begin
                            $display($time, "=> [ERROR] index(j)=%d, soc_to_fpga_axis_expect_value[%d] = %x, soc_to_fpga_axis_captured[%d]  = %x", j, j, soc_to_fpga_axis_expect_value[j], j,
    soc_to_fpga_axis_captured[j]);
                                    error_cnt = error_cnt + 1;
                                end
                                else
                            $display($time, "=> [PASS] index(j)=%d, soc_to_fpga_axis_expect_value[%d] = %x, soc_to_fpga_axis_captured[%d]  = %x", j, j, soc_to_fpga_axis_expect_value[j], j,
    soc_to_fpga_axis_captured[j]);

                    end
                //soc_to_fpga_axis_captured_count = 0;              //reset soc_to_fpga_axis_captured_count
                                #300;
                                $display("------Finish the data input(AXI-Stream ss) & data output(AXI-Stream sm) with checking------");
                    endtask
```

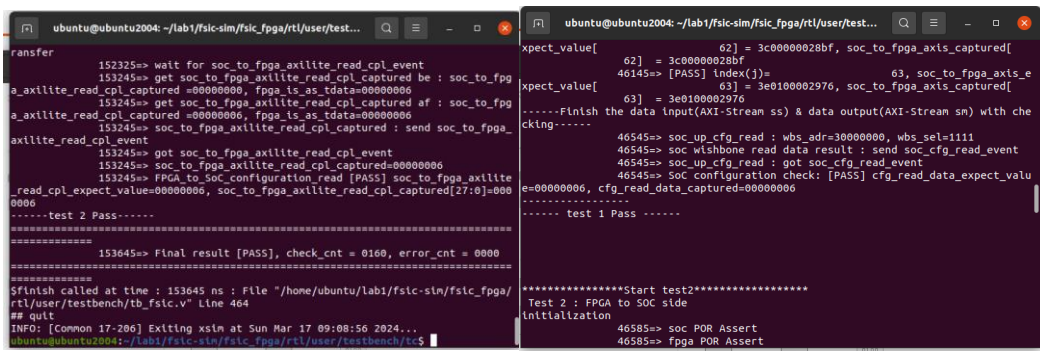
Y[n]的傳輸也和 5.大致相同由 AXI-Stream 傳輸,只是會多了 reset 的步驟讓 data 在 0-63 而不是從 64 開始.

```
            task Golden2Axis;

            reg [31:0]golden_y;

            begin
                    $display("Start Golden Y to AXIS");
                    @ (posedge fpga_coreclk);
                    fpga_as_is_tready <= 1;

                    for(j=0; j<DATA_LENGTH; j=j+1)begin
                            golden_y = 0*j+(-10)*(((j-1)<0)? 0:(j-1))+(-9)*(((j-2)<0)? 0:(j-2))+23*(((j-3)<0)? 0:(j-3))+50*(((j-4)<0)? 0:(j-4))+63*(((j-5)<0)? 0:(j-5))+50*(((j-6)<0)? 0:(j-6))-
    +23*(((j-7)<0)? 0:(j-7))+(-9)*(((j-8)<0)? 0:(j-8))+(-10)*(((j-9)<0)? 0:(j-9))+0*(((j-10)<0)? 0:(j-10));
                            `ifdef USER_PROJECT_SIDEBAND_SUPPORT
                                    if (j==DATA_LENGTH-1) begin
                                            soc_to_fpga_axis_expect_value[soc_to_fpga_axis_expect_count] <= {j[4:0], 4'b0000, 4'b0000, 1'b1, golden_y};
                                    end
                                    else begin
                                            soc_to_fpga_axis_expect_value[soc_to_fpga_axis_expect_count] <= {j[4:0], 4'b0000, 4'b0000, 1'b0, golden_y};
                                    end
                            `else
                                    if (j==DATA_LENGTH-1) begin
                                            soc_to_fpga_axis_expect_value[soc_to_fpga_axis_expect_count] <= {4'b0000, 4'b0000, 1'b1, golden_y};
                                    end
                                    else begin
                                            soc_to_fpga_axis_expect_value[soc_to_fpga_axis_expect_count] <= {4'b0000, 4'b0000, 1'b0, golden_y};
                                    end
                            `endif
                            soc_to_fpga_axis_expect_count <= soc_to_fpga_axis_expect_count+1;
                            fpga_axis_req_v2(j, TID_DN_UP, 0);
                    end

                    $display($time, "=> Golden Y to AXIS done !!");
                end
        endtask
```

```
                    for(j=0; j<DATA_LENGTH; j=j+1)begin
                            golden_y = 0*j+(-10)*(((j-1)<0)? 0:(j-1))+(-9)*(((j-2)<0)? 0:(j-2))+23*(((j-3)<0)? 0:(j-3))+50*(((j-4)<0)? 0:(j-4))+63*(((j-5)<0)? 0:(j-5))+50*(((j-6)<0)? 0:(j-6))
    +23*(((j-7)<0)? 0:(j-7))+(-9)*(((j-8)<0)? 0:(j-8))+(-10)*(((j-9)<0)? 0:(j-9))+0*(((j-10)<0)? 0:(j-10));
```

利用 golden2axis 這個 task 去比對,主要是用 for loop 裡面的判別式的 golden_y 來判斷跑出來的答案有沒有和 golden value 相同.

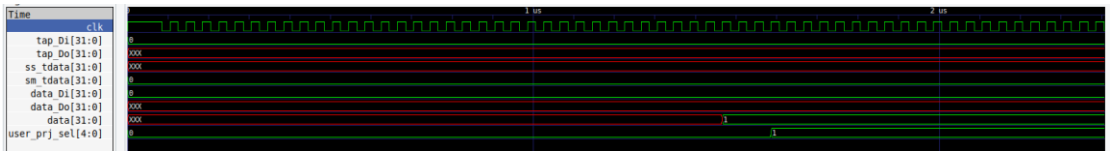
7. Screenshot simulation results printed on screen, to show that your Test#1 & Test#2 complete successfully



8. Screenshot simulation waveform:

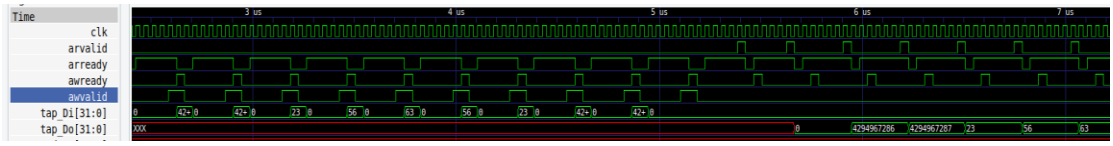
• Configuration cycle (when we program [‘h3000_5000] = 32’h01, signal user_prj_sel changes accordingly) • AXI-Lite transaction cycles (feed in tap parameters, data_length) • Stream-in, Stream-out

[‘h3000_5000] = 32’h01, signal user_prj_sel changes accordingly



圖中最下面的 data 為 testbench 中控制 enable user_prj1 的信號,輸出 32’h01 後 user_prj_sel 會變為 1.

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

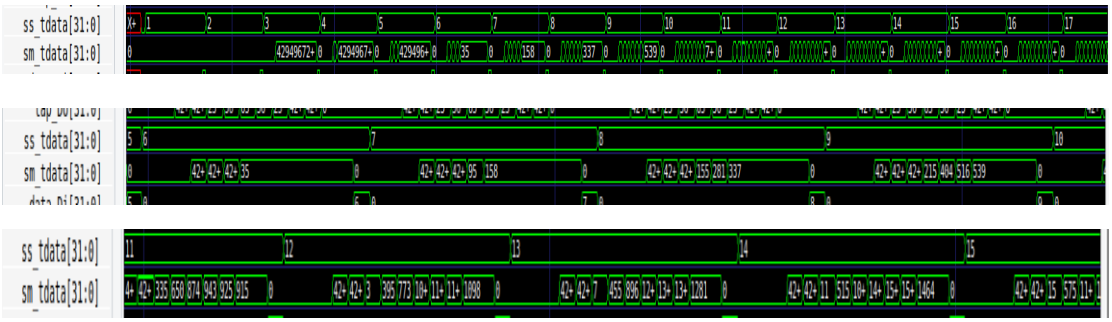


可由 axilite 的 R/W 來看到 tap_Di 和 tap_Do 的數值變化.

具體來說是 awvalid+awready→可以 feed data 給 tap

Aready 拉為 1,當 aready+arvalid→tap_Do,且下個 clk aready 拉為 0 後,過個 clk 再拉回 1.

Stream-in, Stream-out



```
46145=> [PASS] index(j)=                0, soc_to_fpga_axis_expect_value[                0] = 000000000000, soc_to_fpga_axis_captured[                0] = 000000000000
46145=> [PASS] index(j)=                1, soc_to_fpga_axis_expect_value[                1] = 020000000000, soc_to_fpga_axis_captured[                1] = 020000000000
46145=> [PASS] index(j)=                2, soc_to_fpga_axis_expect_value[                2] = 0400fffffff6, soc_to_fpga_axis_captured[                2] = 0400fffffff6
46145=> [PASS] index(j)=                3, soc_to_fpga_axis_expect_value[                3] = 0600ffffffe3, soc_to_fpga_axis_captured[                3] = 0600ffffffe3
46145=> [PASS] index(j)=                4, soc_to_fpga_axis_expect_value[                4] = 0800ffffffe7, soc_to_fpga_axis_captured[                4] = 0800ffffffe7
46145=> [PASS] index(j)=                5, soc_to_fpga_axis_expect_value[                5] = 0a0000000023, soc_to_fpga_axis_captured[                5] = 0a0000000023
46145=> [PASS] index(j)=                6, soc_to_fpga_axis_expect_value[                6] = 0c000000009e, soc_to_fpga_axis_captured[                6] = 0c000000009e
46145=> [PASS] index(j)=                7, soc_to_fpga_axis_expect_value[                7] = 0e0000000151, soc_to_fpga_axis_captured[                7] = 0e0000000151
46145=> [PASS] index(j)=                8, soc_to_fpga_axis_expect_value[                8] = 10000000021b, soc_to_fpga_axis_captured[                8] = 10000000021b
46145=> [PASS] index(j)=                9, soc_to_fpga_axis_expect_value[                9] = 1200000002dc, soc_to_fpga_axis_captured[                9] = 1200000002dc
46145=> [PASS] index(j)=               10, soc_to_fpga_axis_expect_value[               10] = 140000000393, soc_to_fpga_axis_captured[               10] = 140000000393
46145=> [PASS] index(j)=               11, soc_to_fpga_axis_expect_value[               11] = 16000000044a, soc_to_fpga_axis_captured[               11] = 16000000044a
```

ss_tdata 為 stream-in(X[n])信號, sm_tdata 為 stream-out(Y[n]).

下圖為在 terminal 運行的結果,都有正確接收到 value.

9.Debug experience (bug found, and how to fix it)

這個 lab 有遇到的問題是一開始 SoC side 寫過去的值為 0-63,但從 FPGA side 寫回來並未做 reset,因此 test2 會從 64 開始這會造成錯誤,因此需要對 data 的 reg 進行 reset.

```
865                 task reset_capture_expect;
866                 begin
867                         soc_to_fpga_axis_captured_count=0;
868                         for(j=0; j<DATA_LENGTH; j=j+1)begin
869                                 soc_to_fpga_axis_expect_value[j] = 0;
870                                 soc_to_fpga_axis_captured[j] = 0;
871                 end
872
873                         $display("Reset capture and expect done, start transfer X & Y data !!");
874                 end
875                 endtask
876
```