ASOC Lab1 FSIC-SIM

111061631 張煒侖

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

參考原先提供的其他 task 的 cfg_write 寫法，改成 CC_Base 的 Address Range，再將 data 改為 32’h01。

```
        task soc_cc_cfg_write;
                input [11:0] offset;            //4K range
                input [3:0] sel;
                input [31:0] data;

                begin
                        @ (posedge soc_coreclk);
                        wbs_adr <= CC_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_cc_cfg_write : wbs_adr=%x, wbs_sel=%b, wbs_wdata=%x", wbs_adr,
wbs_sel, wbs_wdata);
                end
        endtask
```

```
        task test1;
                begin
                        fsic_system_initial();

                        soc_cc_cfg_write(12'h000 ,4'b1111 ,32'h01); //select user_prj1
                        soc_up_cfg_write(12'h010 ,4'b1111 ,32'd64);
                        for(k=0; k< 11; k=k+1) begin
                                soc_up_cfg_write(12'h20+4*k, 4'b1111, coef[k]);
                        end

                        @ (posedge fpga_coreclk) soc_up_cfg_write(12'h000 ,4'b1111 ,32'h01); //ap_start=1
```

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

cc_xx_enable 會根據不同的 module 依據他們對應到的 address range 來拉訊號

```
                begin
                        cc_aa_enable_o <= ( m_axi_request_add[31:12] == 20'h30002 )? 1'b1 : 1'b0;
                        cc_as_enable_o <= ( m_axi_request_add[31:12] == 20'h30004 )? 1'b1 : 1'b0;
                        cc_is_enable_o <= ( m_axi_request_add[31:12] == 20'h30003 )? 1'b1 : 1'b0;
                        cc_la_enable_o <= ( m_axi_request_add[31:12] == 20'h30001 )? 1'b1 : 1'b0;
                        cc_up_enable_o <= ( m_axi_request_add[31:12] == 20'h30000 )? 1'b1 : 1'b0;
                        cc_enable <= ( m_axi_request_add[31:12] == 20'h30005 )? 1'b1 : 1'b0;
                        cc_sub_enable <= ( (m_axi_request_add[31:12] >= 20'h30006) && (m_axi_request_add[31:12] <=
20'h3FFFF ) )? 1'b1 : 1'b0;
                end
```

cc_enable 拉起來的時候，write valid 也會被拉起來

```
assign cc_axi_awvalid = axi_awvalid && cc_enable;
assign cc_axi_wvalid = axi_wvalid && cc_enable;
```

write valid 被拉起來後，user_prj_sel 就開始選擇對應的 user_prj，所以在本 lab 中，將 32’h3000_5000 寫入 1，則 user_prj_sel 也會選擇 user_prj1

```
//////////////////////////////////////////
// Always for AXI-Lite CC Slave response //
//////////////////////////////////////////
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] == 1) ) begin //offset 0
                                user_prj_sel_o <= axi_wdata[4:0];
                        end
                        else begin
                                user_prj_sel_o <= user_prj_sel_o;
                        end
                end
        end
end
```

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

一開始根據講義所提供的 task 範例來進行 initial，然後自定義了一個新的 task(soc_cc_cfe_write)，是基於 config_control 的 address 為 base，將 1 寫入 32’h3000_5000，由此來 select user_prj1(fir)，然後將 data 數(64)寫入 32’h3000_0010，再將 fir 的係數寫進 32’h3000_0020~48，最後將 1 寫入 32’h0000_0000，啟動 ap_start。soc_mailbox_notify 是用於通知 FPGA 開始啟動 FIR，read_golden 是讀取測資的正確答案，最後透過 check_xy_value 檢查 FIR 運算結果是否於答案一致。

```
task test1;
        begin
                fsic_system_initial();

                soc_cc_cfg_write(12'h000 ,4'b1111 ,32'h01); //select user_prj1
                soc_up_cfg_write(12'h010 ,4'b1111 ,32'd64);
                for(k=0; k< 11; k=k+1) begin
                        soc_up_cfg_write(12'h20+4*k, 4'b1111, coef[k]);
                end

                @ (posedge fpga_coreclk) soc_up_cfg_write(12'h000 ,4'b1111 ,32'h01); //ap_start=1
                soc_mailbox_notify();
                read_golden();
                check_xy_value();
        end
endtask
```

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

與 lab1 所處理的方式大致相同，不同的地方是從 FPGA 寫入資料，並且要先將 lab1 的 counter 數歸 0。從 FPGA 寫入資料是使用範例樣本的 task(fpga_axilite_write)，fpga_axilite_write 會先寫入到 AA 中，再由 AA 根據目的地轉傳到 UP，因此，不需要再另外透過 mailbox 來通知。

```
task test2;
        begin
                fsic_system_initial();
                soc_to_fpga_axis_captured_count = 0;
                error_cnt = 0;
                check_cnt = 0;

                soc_cc_cfg_write(12'h000 ,4'b1111 ,32'h01); //select user_prj1
                fpga_axilite_write(FPGA_to_SOC_UP_BASE+12'h010, 4'b1111, 32'd64);
                for(i=0; i<11; i=i+1) begin
                        fpga_axilite_write(FPGA_to_SOC_UP_BASE+12'h020+4*i, 4'b1111, coef[i]);
                end
                @ (posedge fpga_coreclk) fpga_axilite_write(FPGA_to_SOC_UP_BASE, 4'b1111, 32'd1);

                test2_fpga_axis_req();
                read_golden();
                check_xy_value();

        end
endtask
```

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

使用的 task 是參考範例的 fpga_axis_req 並稍微修改，由於不會用到原先的 mode1(隨機測資)，直接使用 mode0 的部分。並將設定好的 64 筆資料經過 FPGA side 透過 AXIS 傳資料到 user_prj。tid、TUSER_AXIS 代表資料會透過 AXIS 傳送到 user_prj。

```
        task fpga_axis_req_modified;
                input [31:0] data;
                input [1:0] tid;
                reg [31:0] tdata;
                `ifdef USER_PROJECT_SIDEBAND_SUPPORT
                        reg [pUSER_PROJECT_SIDEBAND_WIDTH-1:0]tupsb;
                `endif
                reg [3:0] tstrb;
                reg [3:0] tkeep;
                reg tlast;
                begin
                        tdata = data;
                        `ifdef USER_PROJECT_SIDEBAND_SUPPORT
                                tupsb = tdata [4:0];
                        `endif
                        tstrb = 4'b0000;
                        tkeep = 4'b0000;
                        if(data==63) tlast = 1'b1;
                        else tlast = 1'b0;
                        `ifdef USER_PROJECT_SIDEBAND_SUPPORT
                                fpga_as_is_tupsb <= tupsb;
                        `endif
                        fpga_as_is_tstrb <=  tstrb;
                        fpga_as_is_tkeep <=  tkeep;
                        fpga_as_is_tlast <=  tlast;
                        fpga_as_is_tdata <= tdata;      //for axis write data
                        `ifdef USER_PROJECT_SIDEBAND_SUPPORT
                                $strobe($time, "=> fpga_axis_req send data, fpga_as_is_tupsb = %b, fpga_as_is_tstrb
= %b, fpga_as_is_tkeep = %b, fpga_as_is_tlast = %b, fpga_as_is_tdata = %x", fpga_as_is_tupsb, fpga_as_is_tstrb,
fpga_as_is_tkeep, fpga_as_is_tlast, fpga_as_is_tdata);
                        `else
                                $strobe($time, "=> fpga_axis_req send data, fpga_as_is_tstrb = %b, fpga_as_is_tkeep
= %b, fpga_as_is_tlast = %b, fpga_as_is_tdata = %x", fpga_as_is_tstrb, fpga_as_is_tkeep, fpga_as_is_tlast,
fpga_as_is_tdata);
                        `endif

                        fpga_as_is_tid <=  tid;         //set target
                        fpga_as_is_tuser <=  TUSER_AXIS;                //for axis req
                        fpga_as_is_tvalid <= 1;
                        soc_to_fpga_axis_expect_count <= soc_to_fpga_axis_expect_count+1;

                        @ (posedge fpga_coreclk);
                        while (fpga_is_as_tready == 0) begin            // wait util fpga_is_as_tready == 1 then
change data
                                @ (posedge fpga_coreclk);
                        end
                        fpga_as_is_tvalid <= 0;
                end
        endtask
```

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

由下圖可見，我所寫的這個 task 會讀取 golden data 的值，總共有 64 筆，並且會一筆一筆與 fir 運算的輸出 y 去進行比對，若有任何一筆的測資與正確答案不相符，error counter 都會+1，若最後比對 error counter 的數目為 0，才算是測試通過。

```
        task check_xy_value;
                begin
                        @ (posedge fpga_coreclk);
                        soc_to_fpga_axis_expect_count = 0;
                        for(idx3=0; idx3<64; idx3=idx3+1)begin          //
                                if(idx3 != 63)begin
                                        soc_to_fpga_axis_expect_value[soc_to_fpga_axis_expect_count] = {4'b0000,
4'b0000, 1'b0, golden_list[idx3]};
                                        soc_to_fpga_axis_expect_count = soc_to_fpga_axis_expect_count+1;
                                end
                                else
                                begin
                                        soc_to_fpga_axis_expect_value[soc_to_fpga_axis_expect_count] = {4'b0000,
4'b0000, 1'b1, golden_list[63]};
                                        soc_to_fpga_axis_expect_count = soc_to_fpga_axis_expect_count+1;
                                end
                        end


                        for(idx3=0; idx3<64; idx3=idx3+1)begin
                                        check_cnt = check_cnt + 1;
                    if (soc_to_fpga_axis_expect_value[idx3] != soc_to_fpga_axis_captured[idx3] ) begin
                        $display($time, "=> [ERROR] number=%d: output y[%d] = %x, golden_data[%d]  = %x", idx3,
idx3, soc_to_fpga_axis_expect_value[idx3], idx3, soc_to_fpga_axis_captured[idx3]);
                                                error_cnt = error_cnt + 1;
                                        end
                                        else
                        $display($time, "=> [PASS] number=%d: output y[%d] = %x, goldgen_data[%d]  = %x", idx3,
idx3, soc_to_fpga_axis_expect_value[idx3], idx3, soc_to_fpga_axis_captured[idx3]);
            end

                end
        endtask
```

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

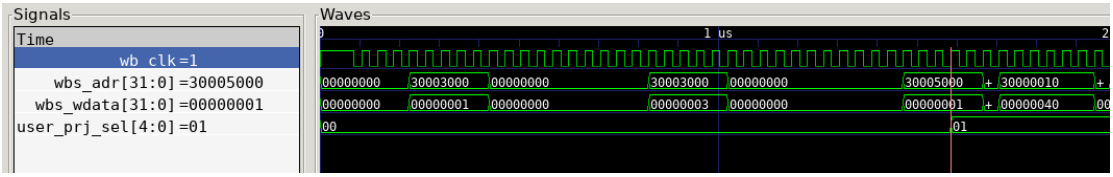
test1 與 test2 都有順利通過 64 筆的測資，並未有 error

```
            41945⇒ [PASS] number=       38: output y[       38] = 000000001797, goldgen_data[       38]  = xxxX00001797
            41945⇒ [PASS] number=       39: output y[       39] = 00000000184e, goldgen_data[       39]  = xxxX0000184e
            41945⇒ [PASS] number=       40: output y[       40] = 000000001905, goldgen_data[       40]  = xxxX00001905
            41945⇒ [PASS] number=       41: output y[       41] = 0000000019bc, goldgen_data[       41]  = xxxX000019bc
            41945⇒ [PASS] number=       42: output y[       42] = 000000001a73, goldgen_data[       42]  = xxxX00001a73
            41945⇒ [PASS] number=       43: output y[       43] = 000000001b2a, goldgen_data[       43]  = xxxX00001b2a
            41945⇒ [PASS] number=       44: output y[       44] = 000000001be1, goldgen_data[       44]  = xxxX00001be1
            41945⇒ [PASS] number=       45: output y[       45] = 000000001c98, goldgen_data[       45]  = xxxX00001c98
            41945⇒ [PASS] number=       46: output y[       46] = 000000001d4f, goldgen_data[       46]  = xxxX00001d4f
            41945⇒ [PASS] number=       47: output y[       47] = 000000001e06, goldgen_data[       47]  = xxxX00001e06
            41945⇒ [PASS] number=       48: output y[       48] = 000000001ebd, goldgen_data[       48]  = xxxX00001ebd
            41945⇒ [PASS] number=       49: output y[       49] = 000000001f74, goldgen_data[       49]  = xxxX00001f74
            41945⇒ [PASS] number=       50: output y[       50] = 00000000202b, goldgen_data[       50]  = xxxX0000202b
            41945⇒ [PASS] number=       51: output y[       51] = 0000000020e2, goldgen_data[       51]  = xxxX000020e2
            41945⇒ [PASS] number=       52: output y[       52] = 000000002199, goldgen_data[       52]  = xxxX00002199
            41945⇒ [PASS] number=       53: output y[       53] = 000000002250, goldgen_data[       53]  = xxxX00002250
            41945⇒ [PASS] number=       54: output y[       54] = 000000002307, goldgen_data[       54]  = xxxX00002307
            41945⇒ [PASS] number=       55: output y[       55] = 0000000023be, goldgen_data[       55]  = xxxX000023be
            41945⇒ [PASS] number=       56: output y[       56] = 000000002475, goldgen_data[       56]  = xxxX00002475
            41945⇒ [PASS] number=       57: output y[       57] = 00000000252c, goldgen_data[       57]  = xxxX0000252c
            41945⇒ [PASS] number=       58: output y[       58] = 0000000025e3, goldgen_data[       58]  = xxxX000025e3
            41945⇒ [PASS] number=       59: output y[       59] = 00000000269a, goldgen_data[       59]  = xxxX0000269a
            41945⇒ [PASS] number=       60: output y[       60] = 000000002751, goldgen_data[       60]  = xxxX00002751
            41945⇒ [PASS] number=       61: output y[       61] = 000000002808, goldgen_data[       61]  = xxxX00002808
            41945⇒ [PASS] number=       62: output y[       62] = 0000000028bf, goldgen_data[       62]  = xxxX000028bf
            41945⇒ [PASS] number=       63: output y[       63] = 000100002976, goldgen_data[       63]  = xxxX00002976
================================================================================
================================================================================
            42345⇒Test1 [PASS], error_cnt = 0000
================================================================================
================================================================================
```

```
            82405⇒ [PASS] number=       38: output y[       38] = 000000001797, goldgen_data[       38]  = xxxX00001797
            82405⇒ [PASS] number=       39: output y[       39] = 00000000184e, goldgen_data[       39]  = xxxX0000184e
            82405⇒ [PASS] number=       40: output y[       40] = 000000001905, goldgen_data[       40]  = xxxX00001905
            82405⇒ [PASS] number=       41: output y[       41] = 0000000019bc, goldgen_data[       41]  = xxxX000019bc
            82405⇒ [PASS] number=       42: output y[       42] = 000000001a73, goldgen_data[       42]  = xxxX00001a73
            82405⇒ [PASS] number=       43: output y[       43] = 000000001b2a, goldgen_data[       43]  = xxxX00001b2a
            82405⇒ [PASS] number=       44: output y[       44] = 000000001be1, goldgen_data[       44]  = xxxX00001be1
            82405⇒ [PASS] number=       45: output y[       45] = 000000001c98, goldgen_data[       45]  = xxxX00001c98
            82405⇒ [PASS] number=       46: output y[       46] = 000000001d4f, goldgen_data[       46]  = xxxX00001d4f
            82405⇒ [PASS] number=       47: output y[       47] = 000000001e06, goldgen_data[       47]  = xxxX00001e06
            82405⇒ [PASS] number=       48: output y[       48] = 000000001ebd, goldgen_data[       48]  = xxxX00001ebd
            82405⇒ [PASS] number=       49: output y[       49] = 000000001f74, goldgen_data[       49]  = xxxX00001f74
            82405⇒ [PASS] number=       50: output y[       50] = 00000000202b, goldgen_data[       50]  = xxxX0000202b
            82405⇒ [PASS] number=       51: output y[       51] = 0000000020e2, goldgen_data[       51]  = xxxX000020e2
            82405⇒ [PASS] number=       52: output y[       52] = 000000002199, goldgen_data[       52]  = xxxX00002199
            82405⇒ [PASS] number=       53: output y[       53] = 000000002250, goldgen_data[       53]  = xxxX00002250
            82405⇒ [PASS] number=       54: output y[       54] = 000000002307, goldgen_data[       54]  = xxxX00002307
            82405⇒ [PASS] number=       55: output y[       55] = 0000000023be, goldgen_data[       55]  = xxxX000023be
            82405⇒ [PASS] number=       56: output y[       56] = 000000002475, goldgen_data[       56]  = xxxX00002475
            82405⇒ [PASS] number=       57: output y[       57] = 00000000252c, goldgen_data[       57]  = xxxX0000252c
            82405⇒ [PASS] number=       58: output y[       58] = 0000000025e3, goldgen_data[       58]  = xxxX000025e3
            82405⇒ [PASS] number=       59: output y[       59] = 00000000269a, goldgen_data[       59]  = xxxX0000269a
            82405⇒ [PASS] number=       60: output y[       60] = 000000002751, goldgen_data[       60]  = xxxX00002751
            82405⇒ [PASS] number=       61: output y[       61] = 000000002808, goldgen_data[       61]  = xxxX00002808
            82405⇒ [PASS] number=       62: output y[       62] = 0000000028bf, goldgen_data[       62]  = xxxX000028bf
            82405⇒ [PASS] number=       63: output y[       63] = 000100002976, goldgen_data[       63]  = xxxX00002976
================================================================================
================================================================================
            82805⇒Test2 [PASS], error_cnt = 0000
================================================================================
```

8. Screenshot simulation waveform:

program [‘h3000_5000] = 32’h01, signal user_prj_sel = 1



data_length = 64, tap parameters = 0,-10,-9,23,56,63,56,23,-9,-10,0, ap_start = 1

X 跟 Y 的值皆正確