Lab1 Report

110061217 王彥智

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

```
soc_proj_cfg_write(4'b0001, 32'h01);
```

```
task soc_proj_cfg_write;
        input [3:0] sel;
        input [31:0] data;

        begin
                @ (posedge soc_coreclk);
                wbs_adr <= 32'h3000_5000;
                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_proj_cfg_write : wbs_adr=%x, wbs_sel=%b, wbs_wdata=%x", wbs_adr, wbs_sel, wbs_wdata);
        end
endtask
```

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

```
begin
        case (wb_fsm_reg)
                wb_fsm_idle:
                begin
                        wbs_ack_o <= 1'b0;
                        wbs_rdata_o <= 32'h0;
                        if ( !wbs_ack_o ) begin
                                if ( wbs_cyc && wbs_stb ) begin
                                        wb_axi_request <= 1'b1;
                                        wb_axi_request_rw <= wbs_we;
                                        wb_axi_request_add <= wbs_adr;          //Latch wbs_adr
                                        if ( wbs_we ) begin
                                                wb_axi_wdata <= wbs_wdata;      //Latch wbs_wdata;
                                                wb_axi_wstrb <= wbs_sel;
                                        end
                                        wb_fsm_reg <= wb_fsm_inprogress;
                                end
                        end
                end
```

Figure 1

```
reg axi_grant_o_reg = 1'b0;

assign m_axi_request_add = axi_grant_o_reg ? f_axi_request_add : wb_axi_request_add;
```

Figure 2

```
////////////////////////////////
// Always for Target Selection //
////////////////////////////////
always @ ( posedge axi_clk or negedge axi_reset_n)
begin
        if ( !axi_reset_n )
        begin
                cc_aa_enable_o <= 1'b0;
                cc_as_enable_o <= 1'b0;
                cc_is_enable_o <= 1'b0;
                cc_la_enable_o <= 1'b0;
                cc_up_enable_o <= 1'b0;
                cc_enable <= 1'b0;
                cc_sub_enable <= 1'b0;
        end else
        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
end
```

Figure 3

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

Figure 4

```
////////////////////////////////////////////
// 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

endmodule
```

Figure 5

```
assign user_prj_sel = user_prj_sel_o;
```

Figure 6

In figure 1, we can see when wbs_fsm_reg is wbs_fsm_idle (wbs_fsm_reg is reset to wbs_fsm_idle), wb_axi_request_add is assigned to wbs_addr. In figure 2, since axi_grant_o_reg is 0, m_axi_request_add is assigned to wb_axi_request_add. In figure 3, when m_axi_request_add[31:12] is 30005, cc_anable is assigned to 1. In figure4, when cc_enable is 1, cc_axi_awvalid and cc_axi_wvalid are assigned to 1 (since axi_awvalid and axi_wvalid are 1). In figure 5, when cc_axi_awvalid and cc_axi_wvalid are 1 and the axi_waddr offset is 0, user_prj_sel_o is assigned to axi_wdata. In figure 6, we can see that user_prj_sel is assigned to user_prj_sel_o. Therefore, we know that when wbs_addr is 3000_5000, user_prj_sel value is decided

by axi_wdata, so by programming configuration address [‘h3000_5000], signal user_prj_sel[4:0] will change accordingly.

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

```
$display("FIR Test1: FIR initialization from SOC side");

// write data len
soc_up_cfg_write(12'h10, 4'b0001, 32'd64);

// write coef.
for (i = 0; i < 11; i = i + 1) begin
        soc_up_cfg_write(12'h20+4*i, 4'b0001, coef[i]);
end

// write ap_start to start fir
soc_up_cfg_write(12'h00, 4'b0001, 1);
```

Figure 7

```
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;
                $display("wait ack");
                @(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
```

Figure 8

Write data length and coefficient to correspond address. In reference FIR, ap_control signal is written at 32’h3000_0000, data_length is written at 32’h3000_0010 and coefficients are written from 32’h3000_020.

Target Module	Address range	Enable signal
User Projects	32’h3000_0xxx	cc_up_enable
Logic Analyzer	32’h3000_1xxx	cc_la_enable
Axis_Axilite	32’h3000_2xxx	cc_aa_enable
IO_Serdes	32’h3000_3xxx	cc_is_enable
Axis_Switch	32’h3000_4xxx	cc_as_enable
Config_Control	32’h3000_5xxx	

Table 1. Address mapping in Lab1

```
if(awaddr==12'h00) begin
    next_state=AXI_Lite_WAIT;
    tap_EN_reg = 0;
    tap_WE_reg = 4'd0;
    tap_Di_reg = 0;
    tap_A_reg = 0;

    next_ap_idle_done_start=wdata[3:0];
    next_data_length=data_length;
end
else if(awaddr==12'h10) begin
    next_state=AXI_Lite_WAIT;
    tap_EN_reg = 0;
    tap_WE_reg = 4'd0;
    tap_Di_reg = 0;
    tap_A_reg = 0;

    next_ap_idle_done_start=ap_idle_done_start;
    next_data_length=wdata;
end
else begin
    next_state=AXI_Lite_WRITE;
    tap_EN_reg = 0;
    tap_WE_reg = 4'b1111;
    tap_Di_reg = wdata;
    tap_A_reg = awaddr-12'h20;

    next_ap_idle_done_start=ap_idle_done_start;
    next_data_length=data_length;
end
```

Figure 9. Address mapping in reference FIR

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

```
$display("FIR Test2: FIR initialization from FPGA side");

// write data len
fpga_to_soc_cfg_write(28'h10, 32'd64);

// write coef
for(i = 0; i < 11; i = i + 1) begin
        fpga_to_soc_cfg_write(28'h20+4*i, coef[i]);
end

// write ap_start to start fir
fpga_to_soc_cfg_write(28'h00, 1);
```

Figure 10

```
task fpga_to_soc_cfg_write;
   input [27:0]addr;
   input [31:0]data;
   begin
   @ (posedge fpga_coreclk);
               fpga_axilite_write_req(addr, 4'b0001, data);
               repeat(100) @ (posedge soc_coreclk);    //TODO fpga wait for write to soc
   end
endtask
```

```
      task fpga_axilite_write_req;
             input [27:0] address;
             input [3:0] BE;
             input [31:0] data;

             begin
                    fpga_as_is_tdata[27:0] <= address;     //for axilite write address phase
                    fpga_as_is_tdata[31:28] <= BE;
                    $strobe($time, "=> fpga_axilite_write_req in address phase = %x - tvalid",
fpga_as_is_tdata);
                    `ifdef USER_PROJECT_SIDEBAND_SUPPORT
                           fpga_as_is_tupsb <=  5'b00000;
                    `endif
                    fpga_as_is_tstrb <=  4'b0000;
                    fpga_as_is_tkeep <=  4'b0000;
                    fpga_as_is_tid <=  TID_DN_AA;          //target to Axis-Axilite
                    fpga_as_is_tuser <=  TUSER_AXILITE_WRITE;             //for axilite write req
                    fpga_as_is_tlast <=  1'b0;
                    fpga_as_is_tvalid <= 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
                    $display($time, "=> fpga_axilite_write_req in address phase = %x - transfer",
fpga_as_is_tdata);

                    fpga_as_is_tdata <= data;      //for axilite write data phase
                    $strobe($time, "=> fpga_axilite_write_req in data phase = %x - tvalid",
fpga_as_is_tdata);
                    `ifdef USER_PROJECT_SIDEBAND_SUPPORT
                           fpga_as_is_tupsb <=  5'b00000;
                    `endif
                    fpga_as_is_tstrb <=  4'b0000;
                    fpga_as_is_tkeep <=  4'b0000;
                    fpga_as_is_tid <=  TID_DN_AA;          //target to Axis-Axilite
                    fpga_as_is_tuser <=  TUSER_AXILITE_WRITE;             //for axilite write req
                    fpga_as_is_tlast <=  1'b1;             //tlast = 1
                    fpga_as_is_tvalid <= 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
                    $display($time, "=> fpga_axilite_write_req in data phase = %x - transfer",
fpga_as_is_tdata);


                    fpga_as_is_tvalid <= 0;

             end
      endtask
```

Figure 11. fpga_to_soc_cfg_write

```
localparam TUSER_AXIS = 2'b00;
localparam TUSER_AXILITE_WRITE = 2'b01;
localparam TUSER_AXILITE_READ_REQ = 2'b10;
localparam TUSER_AXILITE_READ_CPL = 2'b11;

localparam TID_DN_UP = 2'b00;
localparam TID_DN_AA = 2'b01;
localparam TID_UP_UP = 2'b00;
localparam TID_UP_AA = 2'b01;
localparam TID_UP_LA = 2'b10;
```

Figure 12. local parameters

Since there is only axi_stream in FPGA side, we need to use AXIS/AXIL module in FSIC to transform axi_stram protocol to axi_lite protocol. First we set fpga_as_is_tid to TID_DN_AA, so we can target to axis-axilite module and then we set fpga_as_is_tuser to TUSER_AXILITE_WRITE, so we can write address and data to axis-axilite module. Besides, TAD needs two T. 1st T is the Byte_enable and address and 2nd T is the Data.

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

```
task fpga_x_stream_in;
        `ifdef USER_PROJECT_SIDEBAND_SUPPORT
        reg [pUSER_PROJECT_SIDEBAND_WIDTH-1:0]upsb;
        `endif
        begin
               soc_to_fpga_axis_expect_count=0;
                @ (posedge fpga_coreclk);
                fpga_as_is_tready <= 1;
                for (j = 0; j < 64; j++)begin
                soc_to_fpga_axis_expect_count<=soc_to_fpga_axis_expect_count+1;
                   `ifdef USER_PROJECT_SIDEBAND_SUPPORT
                    upsb = 0;
                     fpga_axis_req_FIR(j, TID_DN_UP, 0, upsb);        //target to User Project
                   `else
                     fpga_axis_req_FIR(j, TID_DN_UP, 0);              //target to User Project
                   `endif
                end
                $display($time, "=> FIR x done");
        end
```

Figure 13. fpga_x_stream_in

```
        task fpga_axis_req_FIR;
                input [31:0] data;
                input [1:0] tid;
                input mode;     //o ffor noram, 1 for random data
                `ifdef USER_PROJECT_SIDEBAND_SUPPORT
                input [pUSER_PROJECT_SIDEBAND_WIDTH-1:0] upsb;
                `endif
                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;

                 reg [31:0] exp_data;

                begin
                        if (mode) begin                 //for random data
                                tdata = $random;
                                `ifdef USER_PROJECT_SIDEBAND_SUPPORT
                                        tupsb = $random;
                                `endif
                                tstrb = $random;
                                tkeep = $random;
                                tlast = $random;
                                 exp_data = tdata;
                        end
                        else begin
                                tdata = data;
                                `ifdef USER_PROJECT_SIDEBAND_SUPPORT
                                        //tupsb = 5'b00000;
                                        //tupsb = tdata[4:0];
                                          tupsb = upsb;
                                `endif
                                tstrb = 4'b0000;
                                tkeep = 4'b0000;
                                //tstrb = 4'b1111;
                                //tkeep = 4'b1111;
                                tlast = (data==63)?1'b1:1'b0;   //set tlast = eol
                                 //exp_data = {tst_img_out_buf[idx3+3], tst_img_out_buf[idx3+2],
tst_img_out_buf[idx3+1], tst_img_out_buf[idx3+0]};
                        end
                        `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,data = %x", fpga_as_is_tdata);
                        `else
                                $strobe($time, "=> fpga_axis_req send data,data = %x", 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
```

Figure 13. fpga_axis_req_FIR

Same as doing FIR initialization from FPGA side. First we set fpga_as_is_tid to TID_DN_UP(2'b00), so we can target to the current active project module (FIR) and

then we set fpga_as_is_tuser to TUSER_AXIS(2'b00), so we can do data payload for axis transection.

Routing: TID<1:0> Definition (used by Axis-switch AS)

Direction	TID[1:0]	Source Module	Destination Module
Downstream	00	User DMA (M_AXIS_MM2S) in remote host (option extended user project)	User Project - the current active user project
Downstream	01	Axilite Master R/W in remote host (include Mail box write)	Axis-Axilite (include Mail box)
Upstream	00	User Project - the current active user project	User DMA (S_AXIS_S2MM) in remote host (option extended user project)
Upstream	01	Axis-Axilite (for Mail box)	Axilite slave in remote host (for mail box write)
Upstream	10	Logic Analyzer	Logic Analyzer data receiver - DMA (S_AXIS_S2MM) in remote host

Figure 15. Routing table

Transaction Table - TUSER<1:0> Definition

TUSER<1:0>	# of T	Transaction Type
00	n	Data payload for axis transaction. Limitation: all User pojects Data payload in axis MUST <= Max_axis_Data_payload(=32)
01	2	Axilite write transaction Address + Data. (TAD) 1st T is the Byte-enable + address, i.e. {BE[3:0],ADDR[27:0]}, 2nd T is the Data[31:0]. Note: Axilite write transaction only support 1T in data phase.
10	1	Axilite read Command (Address Phase). TAD<31:0> is the address ADDR[31:0]
11	1	Axilite read Completion (Data Phase). TAD<31:0> is the return data DATA[31:0]. 1. Axilite read transaction only support 1T in data phase (Axilite read Completion). 2. If Axilite read Command is Upstream then the Axilite read Completion is Downstream, and vice versa.

Figure 14. transaction table

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

```
// check FPGA side AXI_stream data out
while (soc_to_fpga_axis_captured_count != 64) begin
        @ (posedge fpga_coreclk);
end
if ((soc_to_fpga_axis_captured[0][31:0] == 0) && (soc_to_fpga_axis_captured[63][31:0] == 10614)) begin
        $display($time, "=> Test1 PASS");
end
else begin
        $display($time, "=> Test1 FAIL");
        error_cnt = error_cnt + 1;
end
$display("--------------------");
```

Figure 15

```
reg soc_to_fpga_axis_event_triggered;

    initial begin         //get upstream soc_to_fpga_axis - for loop back test
    soc_to_fpga_axis_captured_count = 0;
    soc_to_fpga_axis_event_triggered = 0;
            while (1) begin
                @(posedge fpga_coreclk);
                `ifdef USER_PROJECT_SIDEBAND_SUPPORT
                    if (fpga_is_as_tvalid == 1 && fpga_is_as_tid == TID_UP_UP && fpga_is_as_tuser == TUSER_AXIS) begin
                        $display($time, "=> get soc_to_fpga_axis be : soc_to_fpga_axis_captured_count=%d,  soc_to_fpga_axis_captured[%d] =%x, fpga_is_as_tupsb=%x, fpga_is_as_tstrb=%x,
fpga_is_as_tkeep=%x , fpga_is_as_tlast=%x, fpga_is_as_tdata=%x", soc_to_fpga_axis_captured_count, soc_to_fpga_axis_captured_count, soc_to_fpga_axis_captured[soc_to_fpga_axis_captured_count],
fpga_is_as_tupsb, fpga_is_as_tstrb, fpga_is_as_tkeep , fpga_is_as_tlast, fpga_is_as_tdata);
                        soc_to_fpga_axis_captured[soc_to_fpga_axis_captured_count] = {fpga_is_as_tupsb, fpga_is_as_tstrb, fpga_is_as_tkeep , fpga_is_as_tlast,
fpga_is_as_tdata} ;            //use block assignment
                        $display($time, "=> get soc_to_fpga_axis af : soc_to_fpga_axis_captured_count=%d,  soc_to_fpga_axis_captured[%d] =%x, fpga_is_as_tupsb=%x, fpga_is_as_tstrb=%x,
fpga_is_as_tkeep=%x , fpga_is_as_tlast=%x, fpga_is_as_tdata=%x", soc_to_fpga_axis_captured_count, soc_to_fpga_axis_captured_count, soc_to_fpga_axis_captured[soc_to_fpga_axis_captured_count],
fpga_is_as_tupsb, fpga_is_as_tstrb, fpga_is_as_tkeep , fpga_is_as_tlast, fpga_is_as_tdata);
                        soc_to_fpga_axis_captured_count = soc_to_fpga_axis_captured_count+1;
                    end
                    if ( (soc_to_fpga_axis_captured_count == fpga_axis_test_length) && !soc_to_fpga_axis_event_triggered) begin
                        $display($time, "=> soc_to_fpga_axis_captured : send soc_to_fpga_axiis_event");
                        #0 -> soc_to_fpga_axis_event;
                        soc_to_fpga_axis_event_triggered = 1;
                    end
                `else
                    if (fpga_is_as_tvalid == 1 && fpga_is_as_tid == TID_UP_UP && fpga_is_as_tuser == TUSER_AXIS) begin
                        $display($time, "=> get soc_to_fpga_axis be : soc_to_fpga_axis_captured_count=%d,  soc_to_fpga_axis_captured[%d] =%x, fpga_is_as_tstrb=%x,
fpga_is_as_tkeep=%x , fpga_is_as_tlast=%x, fpga_is_as_tdata=%x", soc_to_fpga_axis_captured_count, soc_to_fpga_axis_captured_count, soc_to_fpga_axis_captured[soc_to_fpga_axis_captured_count],
fpga_is_as_tstrb, fpga_is_as_tkeep , fpga_is_as_tlast, fpga_is_as_tdata);
                        soc_to_fpga_axis_captured[soc_to_fpga_axis_captured_count] = {fpga_is_as_tstrb, fpga_is_as_tkeep , fpga_is_as_tlast, fpga_is_as_tdata} ;          //use block
assignment
                        $display($time, "=> get soc_to_fpga_axis af : soc_to_fpga_axis_captured_count=%d,  soc_to_fpga_axis_captured[%d] =%x, fpga_is_as_tstrb=%x,
fpga_is_as_tkeep=%x , fpga_is_as_tlast=%x, fpga_is_as_tdata=%x", soc_to_fpga_axis_captured_count, soc_to_fpga_axis_captured_count, soc_to_fpga_axis_captured[soc_to_fpga_axis_captured_count],
fpga_is_as_tstrb, fpga_is_as_tkeep , fpga_is_as_tlast, fpga_is_as_tdata);
                        soc_to_fpga_axis_captured_count = soc_to_fpga_axis_captured_count+1;
                    end
                    if ( (soc_to_fpga_axis_captured_count == fpga_axis_test_length) && !soc_to_fpga_axis_event_triggered) begin
                        $display($time, "=> soc_to_fpga_axis_captured : send soc_to_fpga_axiis_event");
                        #0 -> soc_to_fpga_axis_event;
                        soc_to_fpga_axis_event_triggered = 1;
                    end
                `endif
```

Figure 16

```
reg [(4+4+1+32-1):0] soc_to_fpga_axis_captured[127:0];

soc_to_fpga_axis_captured[soc_to_fpga_axis_captured_count] =
{fpga_is_as_tupsb, fpga_is_as_tstrb, fpga_is_as_tkeep , fpga_is_as_tlast, fpga_is_as_tdata} ;
```

Figure 17

In figure 16, we can see that it use a infinite while loop to check if there is stream out or not(fpga_is_as_valid == 1 && fpga_is_as_tid == TID_UP_UP && fpga_is_as_tuser == TUSER_AXIS) and then save the captured data in [40]soc_to_fpga_axis_captured[127:0] and since fpga_is_as_tdata is saved at soc_to_fpga_captured[31:0] so we check soc_tofpga_axis_captured[31:0] to get output y. In figure 15, I only check the first data and final data (63^{th}). If they are correct, then the tests pass.

7. Screenshot simulation results printed on screen, to show that your Test1 and Test2 complete successfully.

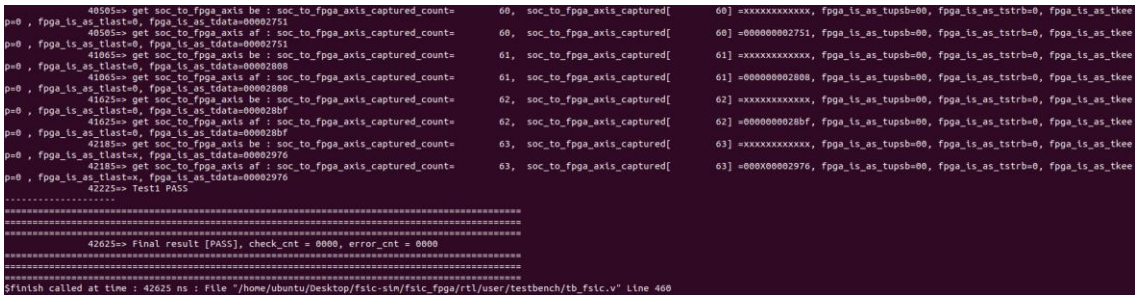

Figure 18. Test1 pass

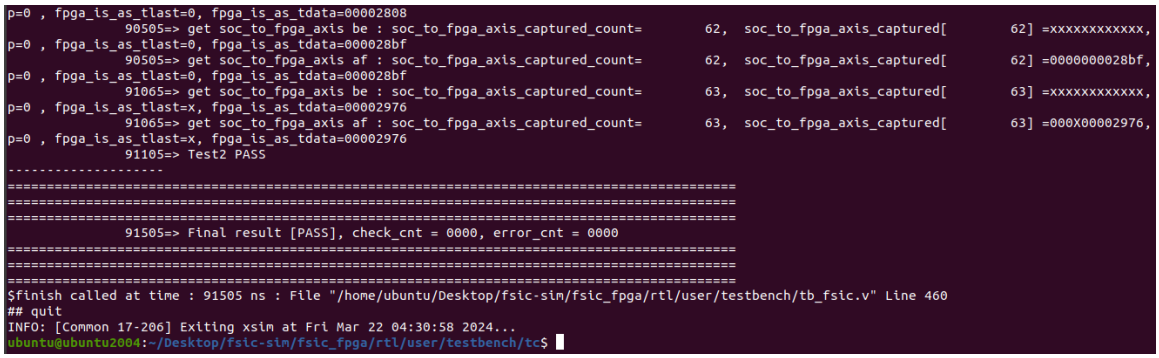

Figure 19. Test2 pass

8. Screenshot simulation waveform:
    a. Configuration cycle.

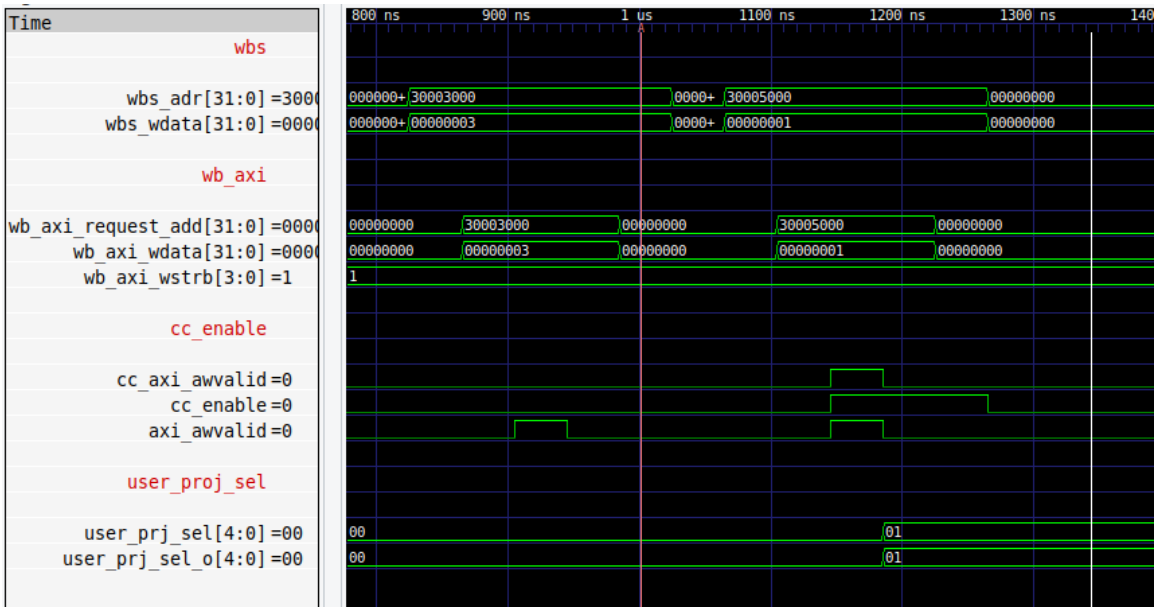

Figure 20

    b. AXI_Lite transaction cycles.

Figure 21

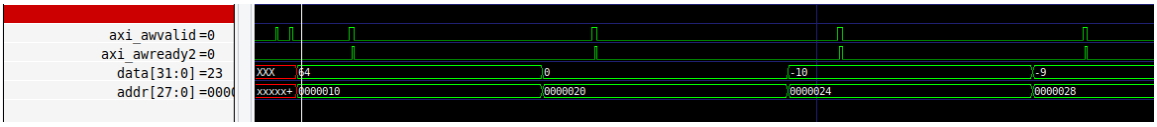

Figure 22

c. Stream-in, stream-out.

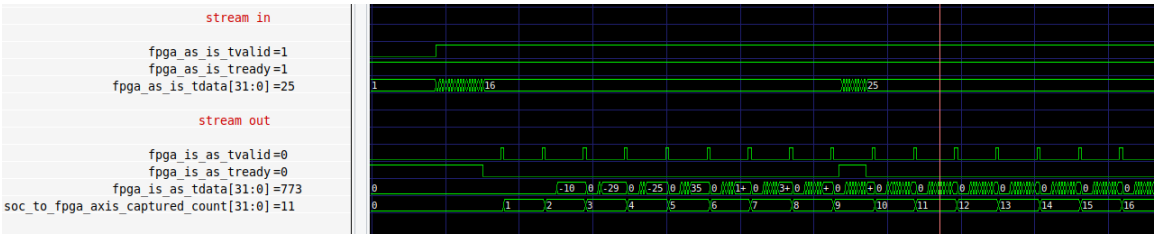

Figure 23

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

In test1, I also check Mailbox protocol by modifying task005.

```
                // mialbox (modify task005)
                soc_to_fpga_mailbox_write_addr_expect_value =  SOC_to_FPGA_MailBox_Base;
                soc_to_fpga_mailbox_write_addr_BE_expect_value = 4'b1111;
                soc_to_fpga_mailbox_write_data_expect_value = 32'ha5a5_a5a5;
                soc_aa_cfg_write(AA_MailBox_Reg_Offset, soc_to_fpga_mailbox_write_addr_BE_expect_value, soc_to_fpga_mailbox_write_data_expect_value);
                @ (soc_to_fpga_mailbox_write_event);        //wait for fpga get the mail box write from soc.
                $display($time, "=> Test1: got soc_to_fpga_mailbox_write_event");
                if ( soc_to_fpga_mailbox_write_addr_expect_value !== soc_to_fpga_mailbox_write_addr_captured[27:0]) begin
                        $display($time, "=> Test1 mailbox [ERROR] soc_to_fpga_mailbox_write_addr_expect_value=%x, soc_to_fpga_mailbox_write_addr_captured[27:0]=%x",
soc_to_fpga_mailbox_write_addr_expect_value, soc_to_fpga_mailbox_write_addr_captured[27:0]);
                        error_cnt = error_cnt + 1;
                end
                else
                        $display($time, "=> Test1 mailbox [PASS] soc_to_fpga_mailbox_write_addr_expect_value=%x, soc_to_fpga_mailbox_write_addr_captured[27:0]=%x",
soc_to_fpga_mailbox_write_addr_expect_value, soc_to_fpga_mailbox_write_addr_captured[27:0]);
```

Figure 24. mailbox check

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

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

Figure 25. soc_aa_cfg_write

```
initial begin          //when soc cfg write to AA, then AA in soc generate soc_to_fpga_mailbox_write,
   stream_data_addr_or_data = 0;
        while (1) begin
                @(posedge fpga_coreclk);
                //New AA version, all stream data with last = 1.
                if (fpga_is_as_tvalid == 1 && fpga_is_as_tid == TID_UP_AA && fpga_is_as_tuser == TUSER_AXILITE_WRITE && fpga_is_as_tlast == 1) begin

            if(stream_data_addr_or_data == 1'b0) begin
                //Address
                $display($time, "=> get soc_to_fpga_mailbox_write_addr_captured be : soc_to_fpga_mailbox_write_addr_captured =%x, fpga_is_as_tdata=%x", soc_to_fpga_mailbox_write_addr_captured,
fpga_is_as_tdata);
                soc_to_fpga_mailbox_write_addr_captured = fpga_is_as_tdata ;            //use block assignment
                $display($time, "=> get soc_to_fpga_mailbox_write_addr_captured af : soc_to_fpga_mailbox_write_addr_captured =%x, fpga_is_as_tdata=%x", soc_to_fpga_mailbox_write_addr_captured,
fpga_is_as_tdata);
                //Next should be data
                stream_data_addr_or_data = 1;
            end else begin
                //Data
                $display($time, "=> get soc_to_fpga_mailbox_write_data_captured be : soc_to_fpga_mailbox_write_data_captured =%x, fpga_is_as_tdata=%x", soc_to_fpga_mailbox_write_data_captured,
fpga_is_as_tdata);
                soc_to_fpga_mailbox_write_data_captured = fpga_is_as_tdata ;            //use block assignment
                $display($time, "=> get soc_to_fpga_mailbox_write_data_captured af : soc_to_fpga_mailbox_write_data_captured =%x, fpga_is_as_tdata=%x", soc_to_fpga_mailbox_write_data_captured,
fpga_is_as_tdata);
                #0 -> soc_to_fpga_mailbox_write_event;
                $display($time, "=> soc_to_fpga_mailbox_write_data_captured : send soc_to_fpga_mailbox_write_event");
                //Next should be address
                stream_data_addr_or_data = 0;
            end
                end
```

Figure 26. mailbox captured

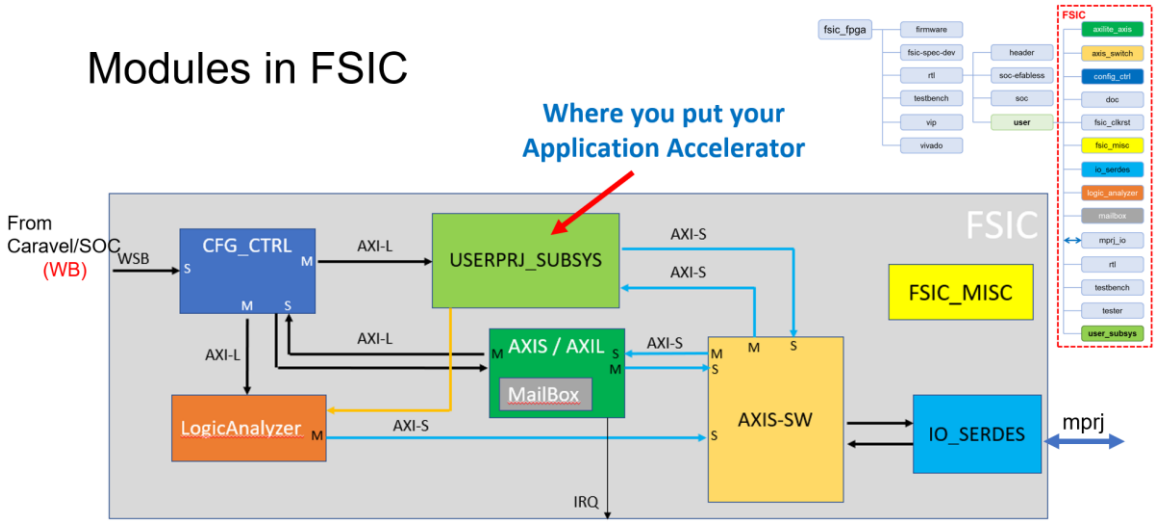

Figure 27. FSIC modules

I write 32'ha5a5_a5a5 to cfg_ctrl module (from soc side) and cfg_ctrl module send data to mailbox (axis_axil mdule) by axi_l. In table 1, we know that axis_axilite module is mapped at 32'h3000_2xxx, so I use soc_aa_cfj_write(figure 25) to write the data. After sending the data, same as stream, I use an infinite loop to capture the data(figured 26). If the except value and captured value are different, then report an error.

10. GitHub: https://github.com/kenny0915/Advance_System_on_Chip_Lab