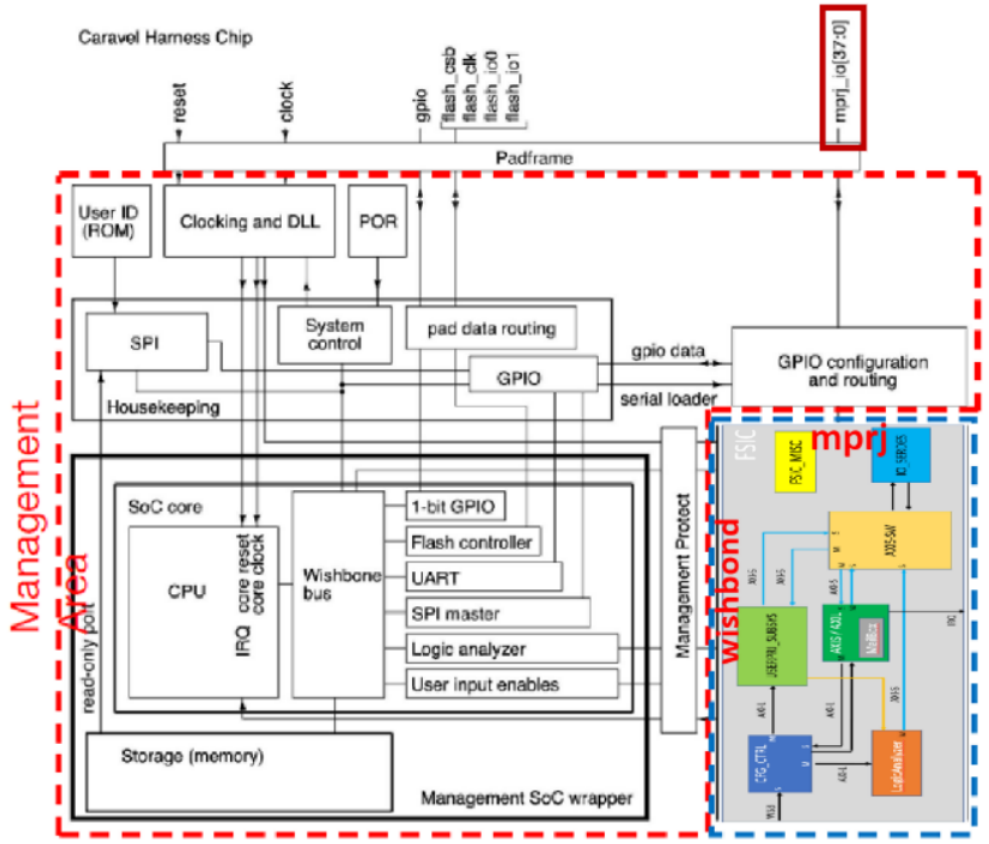
Advance SOC
Lab 1 – FSIC-sim

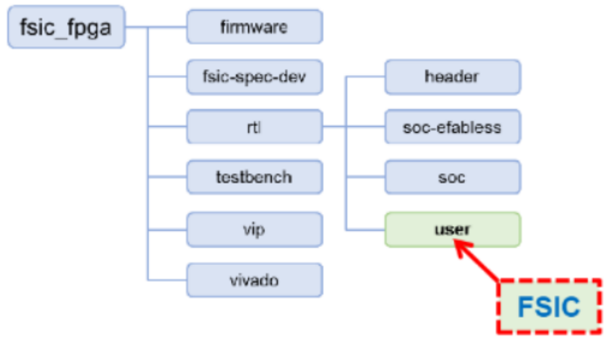
110590022 陳冠晰

GitHub Link: https://github.com/vic9112/Advance_SOC

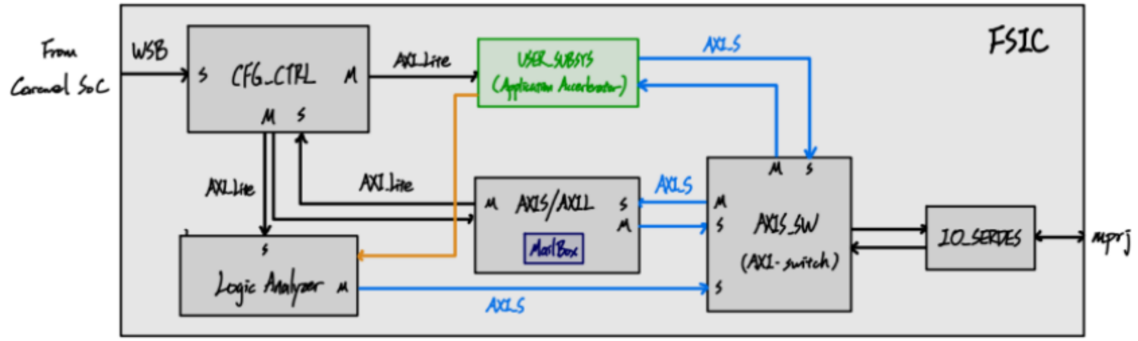
Brief Introduction About FSIC

- FSIC is an architecture to implement an IC validation system based on Caravel SOC





◆ Put **FSIC** into Caravel SoC **user project area**



https://hackmd.io/@vic9112/HykJ2Lpt6

Lab Content

Implementation:

1. Setup FSIC simulation environment
2. Integrate FIR into FSIC
3. Testbench (modify tb_fsic.v)
    1. Test#1
        - FIR Initialization from SOC side
        - Use Mailbox to notify FPGA side to start X, Y stream transfer
        - FIR data X, Y stream data from FPGA side
        - Check if output data Y are correct
    2. Test#2
        - FIR initialization from FPGA side
        - FIR data X, Y stream data from FPGA side
        - Check if output data Y are correct

We will focus on the implementation about testbench in the presentation

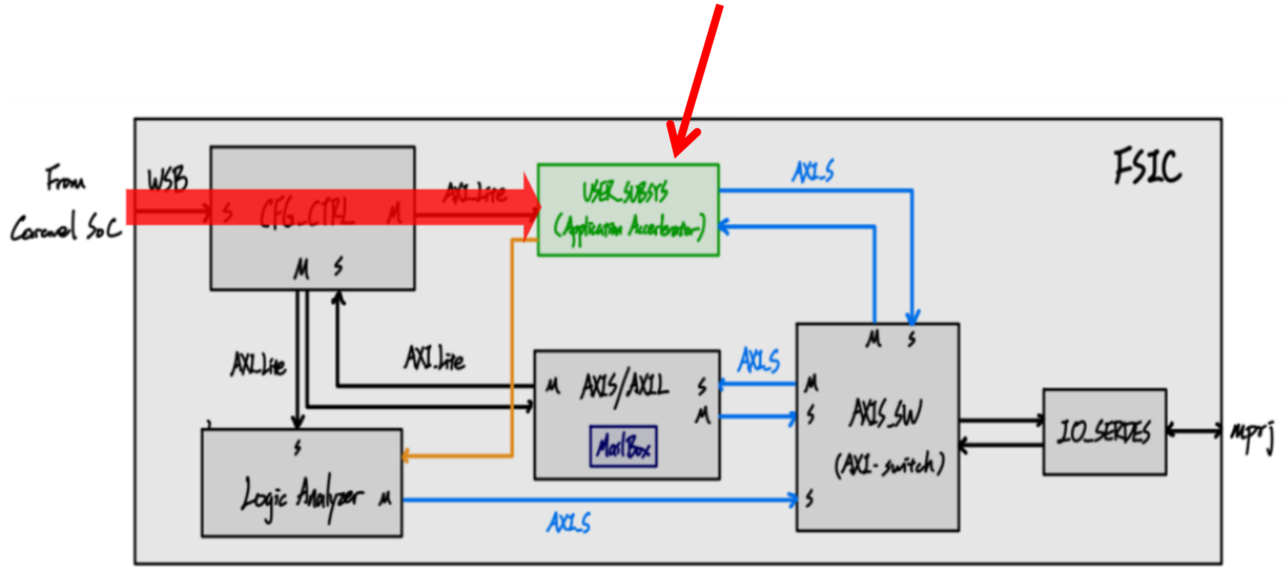
Test#1

Test#1 – FIR initialization from SOC side

- SOC configuration write (WB transaction)
  - Data length, coefficients, ap_start

```
task soc_cfg_write;
    //input  [3:0] target; // 4-bit for AA, IS, CC, regi
    input [31:0] adr;  // 4K range
    input  [3:0] sel;  // byte enable;
    input [31:0] data;
    begin
        @(posedge soc_coreclk);
        // Wishbone Cycle
        wbs_adr   <= adr;  // write address
        wbs_wdata <= data; // write data
        wbs_sel   <= sel;
        wbs_cyc   <= 1'b1;
        wbs_stb   <= 1'b1;
        wbs_we    <= 1'b1;
        // Stall if haven't receive ACK
        @(posedge soc_coreclk);
        while (wbs_ack == 0) @(posedge soc_coreclk);
```

Where we integrate our FIR



https://github.com/vic9112/Advance_SOC/blob/main/lab01%20-%20fsic-sim/fsic_fpga/rtl/user/testbench/tb_fsic.v

Test#1 – FIR initialization from SOC side

- Enable User Project 0

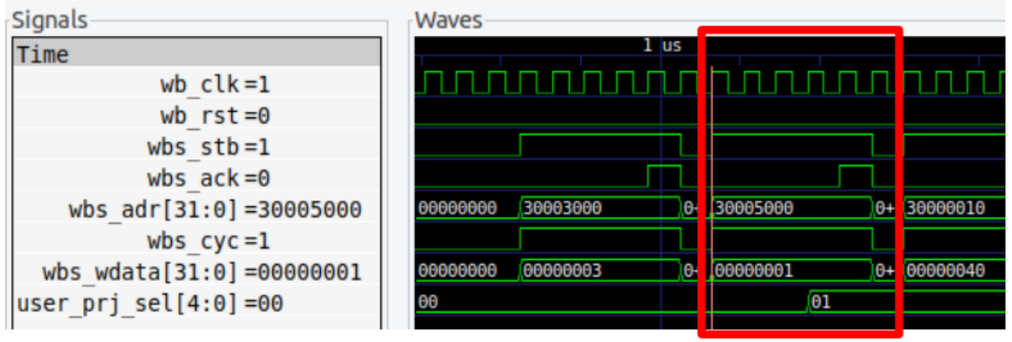
```
soc_cfg_write(32'h3000_5000, 4'b0001, 1);
```

  - Which program the module CC

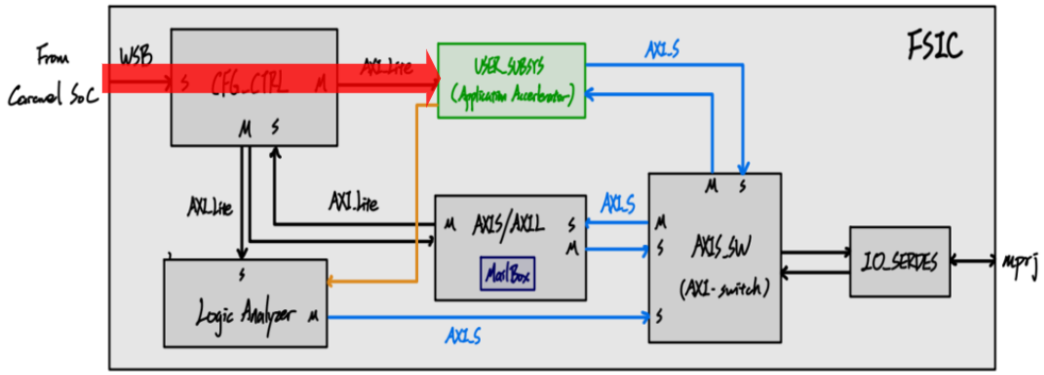
Why programming configuration address 32'h3000_5000, uspj select change accordingly?

- Write data length, coefficients, ap_start

```
soc_cfg_write(32'h3000_0010, 4'b0001, 64);
// Coefficients
for (i = 0; i < 11; i = i + 1) begin
    soc_cfg_write(32'h3000_0020 + 4 * i, 4'b0001, coef[i]);
end
// Start the FIR (ap_start)
soc_cfg_write(32'h3000_0000, 4'b0001, 1);
```

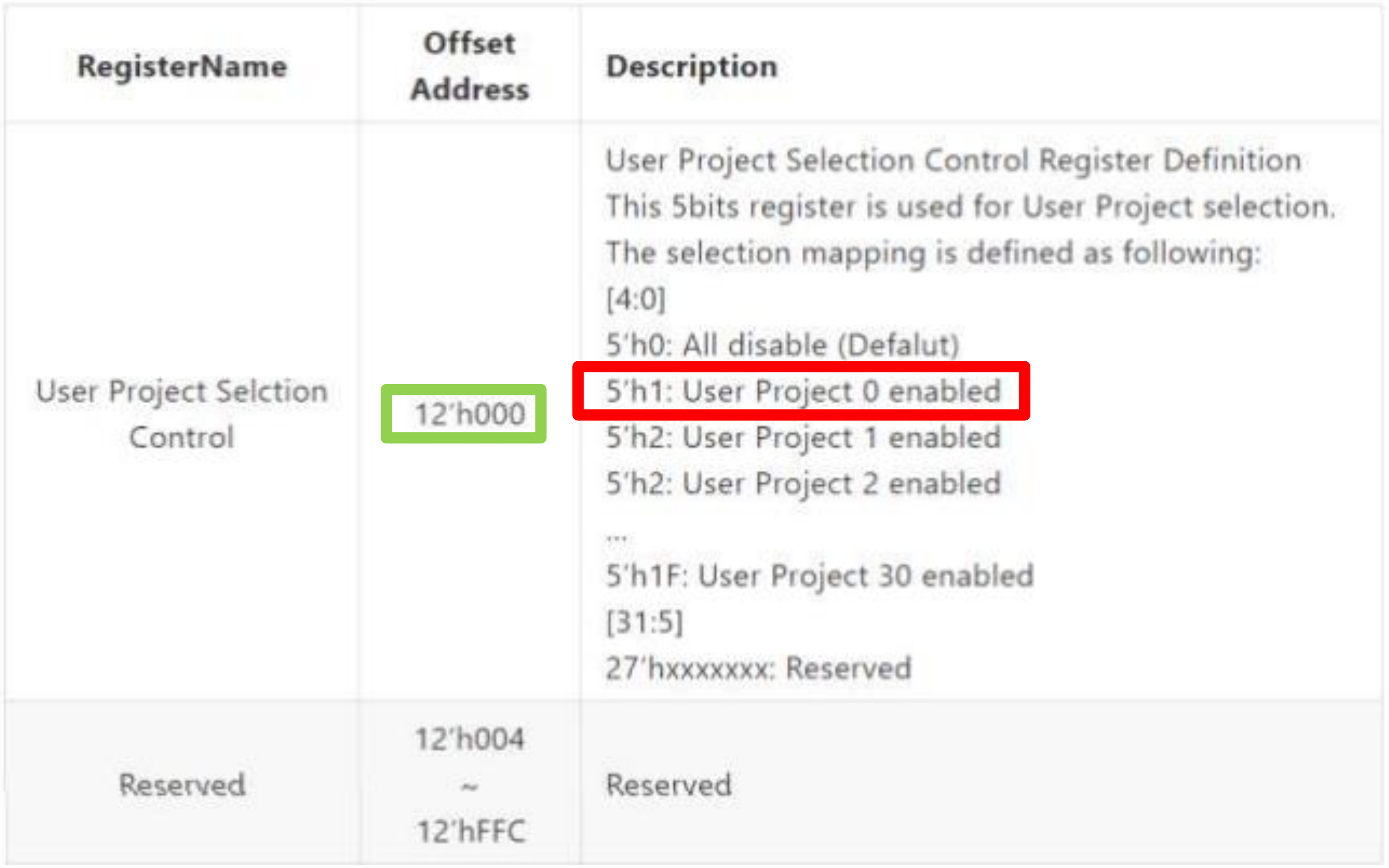


```
Signals                          Waves
Time                                                 1 us
wb_clk=1
wb_rst=0
wbs_stb=1
wbs_ack=0
wbs_adr[31:0]=30005000           00000000  30003000  0-  30005000  0+  30000010
wbs_cyc=1
wbs_wdata[31:0]=00000001         00000000  00000003  0-  00000001  0+  00000040
user_prj_sel[4:0]=00             00                        01
```



Test#1 – FIR initialization from SOC side

- Configuration control group (for 32'h3000_5000, 4'b0001)



RegisterName	Offset Address	Description
User Project Selction Control	12'h000	User Project Selection Control Register Definition This 5bits register is used for User Project selection. The selection mapping is defined as following: [4:0] 5'h0: All disable (Defalut) 5'h1: User Project 0 enabled 5'h2: User Project 1 enabled 5'h2: User Project 2 enabled ... 5'h1F: User Project 30 enabled [31:5] 27'hxxxxxxx: Reserved
Reserved	12'h004 ~ 12'hFFC	Reserved

https://hackmd.io/@vic9112/HykJ2Lpt6

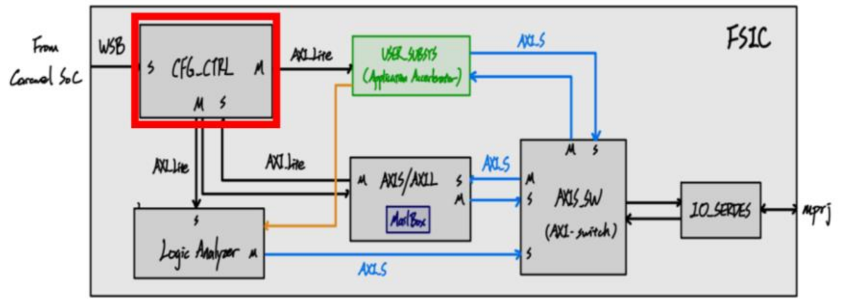
Test#1 – FIR initialization from SOC side

- In `config_ctrl.v` (line 273)

```
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;
```

- Addr[31:12] == 20’h3000_5 will trigger “cc_enable”, further generate an AXI transaction.

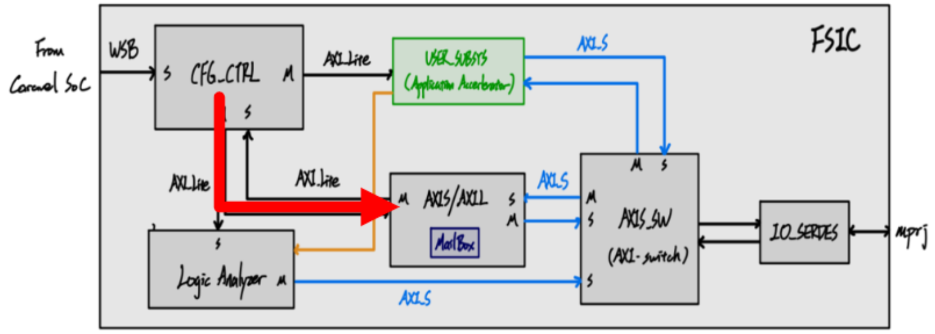
```
assign cc_axi_awvalid = axi_awvalid && cc_enable;
assign cc_axi_wvalid = axi_wvalid && cc_enable;
```



Test#1 – Use Mailbox to notify FPGA side

- Should program AA module
  - Configurate address: 32’h3000_2000 (in caravel.h)

```
#define reg_fsic_aa_mb0 (*(volatile uint32_t*)0x30002000)
```

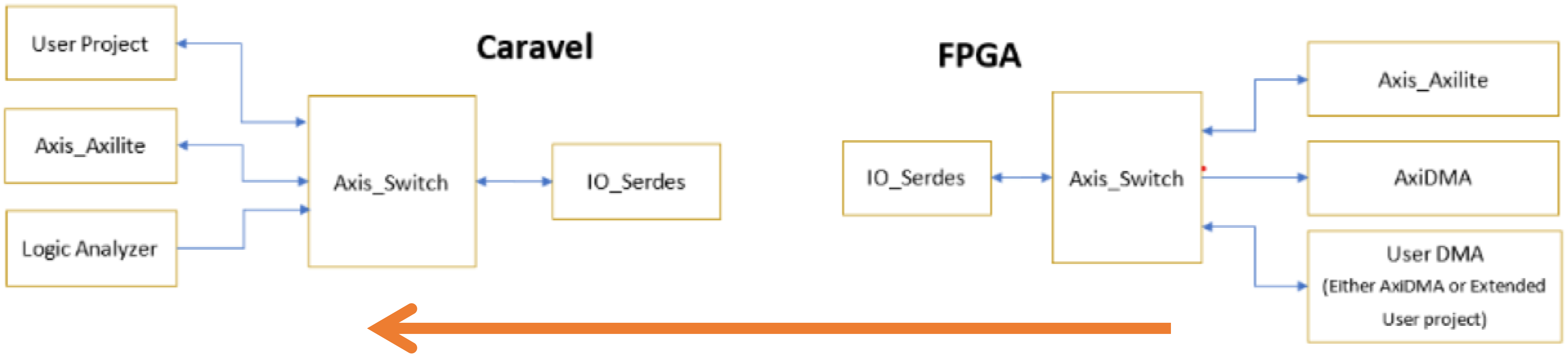


```
// Mailbox BaseAddr
soc_to_fpga_mailbox_write_addr_expect_value = SOC_to_FPGA_MailBox_Base;
// Byte Enable
soc_to_fpga_mailbox_write_addr_BE_expect_value = 4'b1111;
// Data Check
soc_to_fpga_mailbox_write_data_expect_value = 32'hA5A5_A5A5;
// Configuration
soc_cfg_write(32'h3000_2000, soc_to_fpga_mailbox_write_addr_BE_expect_value, soc_to_fpga_mailbox_write_data_expect_value);
// Wait untill FPGA finish the write from SOC
@ (soc_to_fpga_mailbox_write_event)
$display($time, "=> Receive the 'soc_to_fpga_mailbox_write_event'");
// Check the expect value in Mailbox
if (soc_to_fpga_mailbox_write_data_expect_value != soc_to_fpga_mailbox_write_data_captured) begin
```

https://github.com/vic9112/Advance_SOC/blob/main/lab01%20-%20fsic-sim/fsic_fpga/rtl/user/testbench/tb_fsic.v

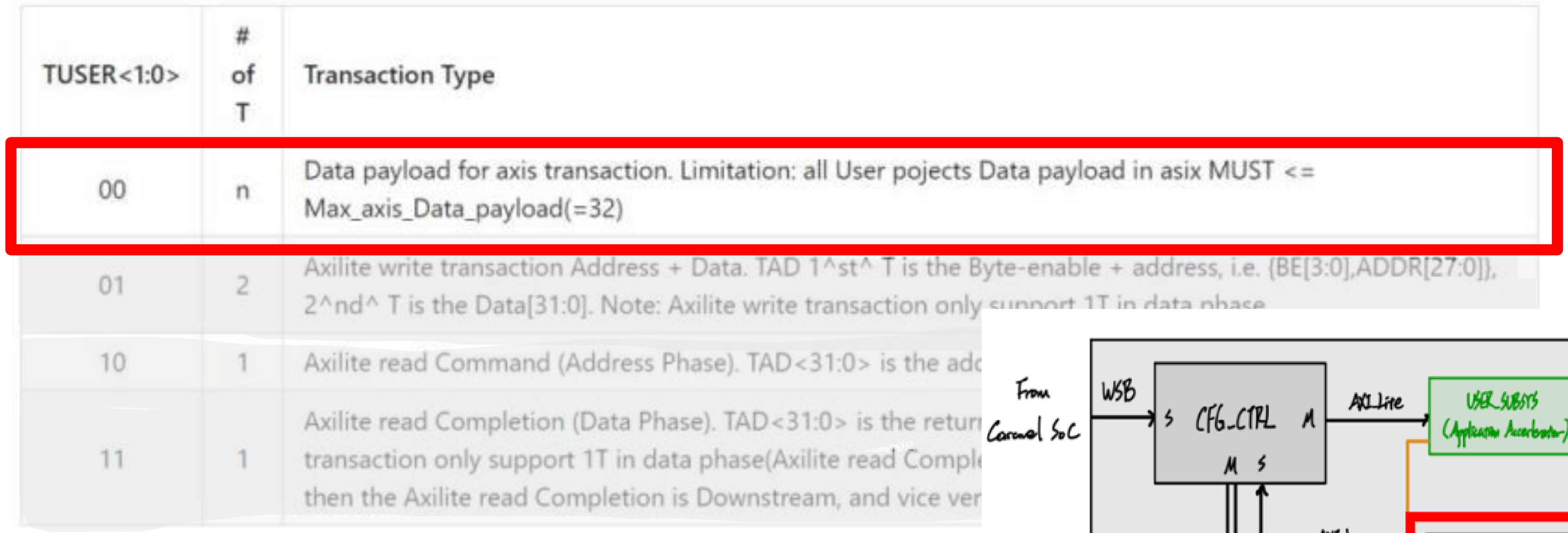
Test#1 – FIR stream data X, Y from FPGA side

- Feed data from FPGA side (downstream)
  - Program AA for destination, AS for transaction type

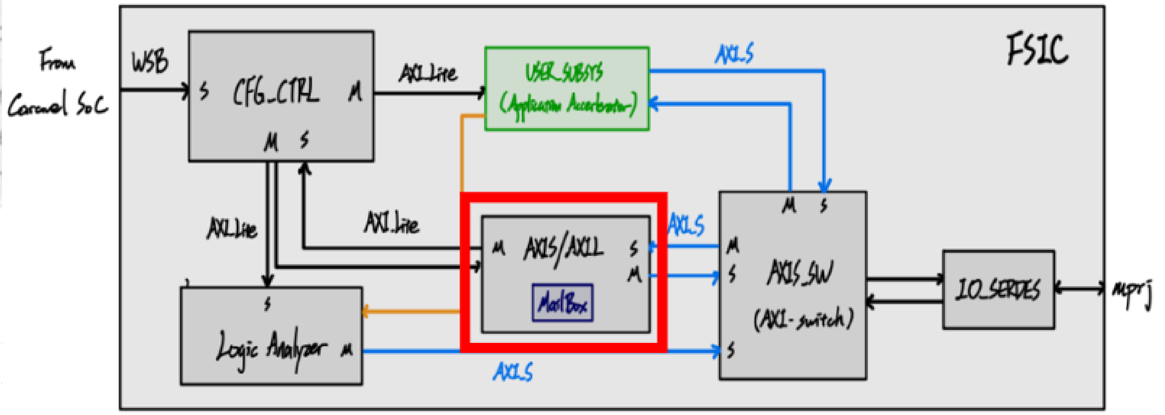


Test#1 – FIR stream data X, Y from FPGA side

- TUSER table defined under AA module (AXILite-AXIS)
  - We should choose 2'b00 to generate the AXI-Stream transaction



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

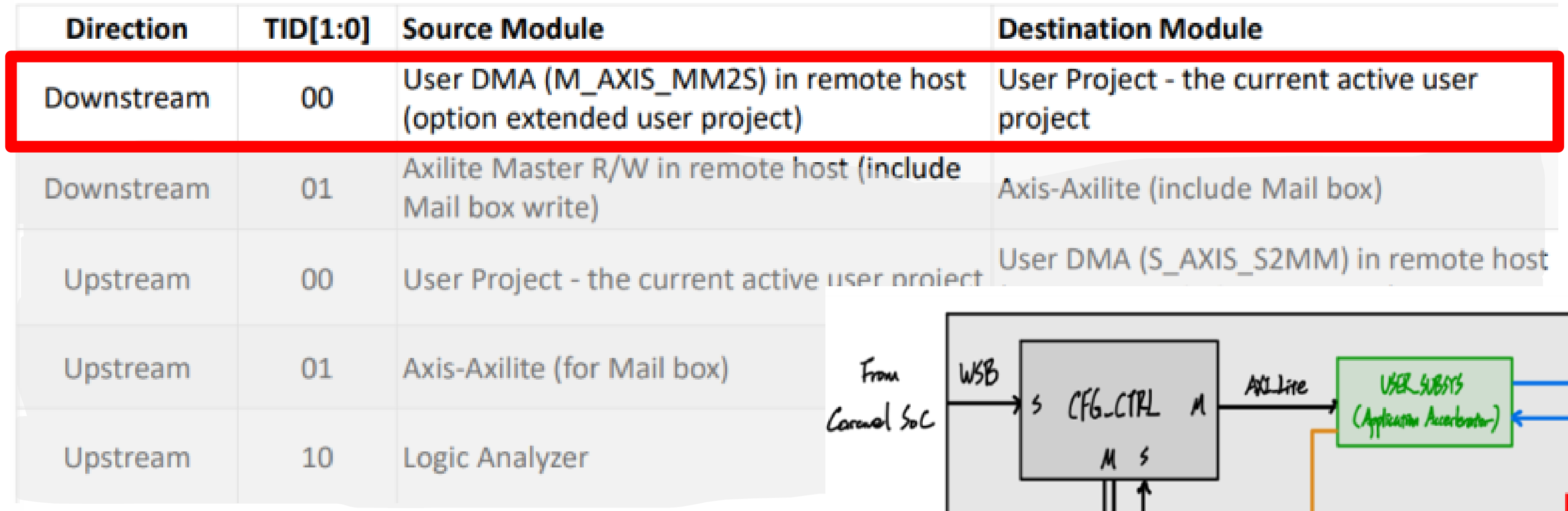


Notice that we have AA module at both side (FPGA/SOC

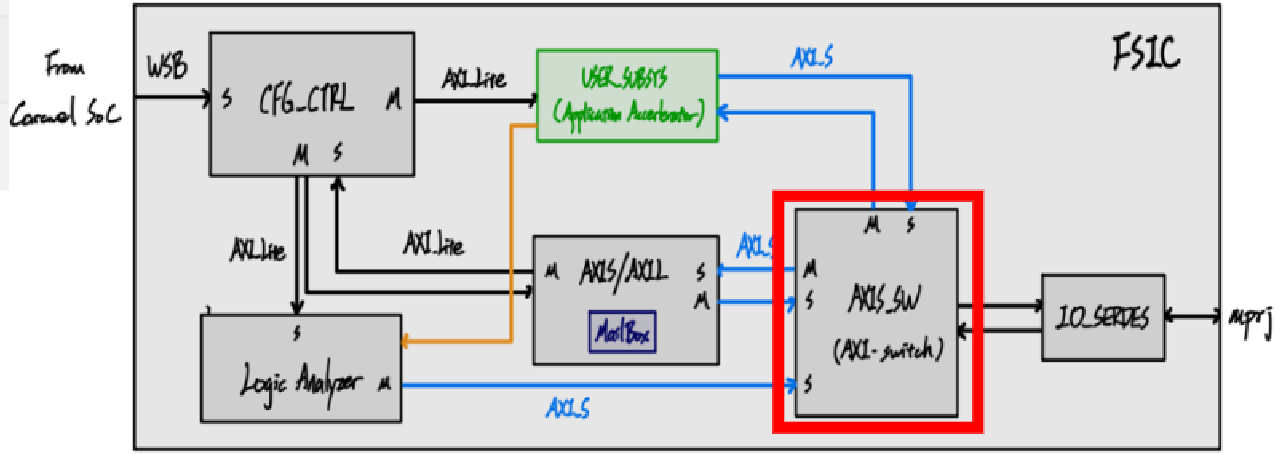
https://hackmd.io/@vic9112/HykJ2Lpt6

Test#1 – FIR stream data X, Y from FPGA side

- TID table defined under AS module (AXI-Switch)
  - We should choose 2’b00 to program user project (downstream)

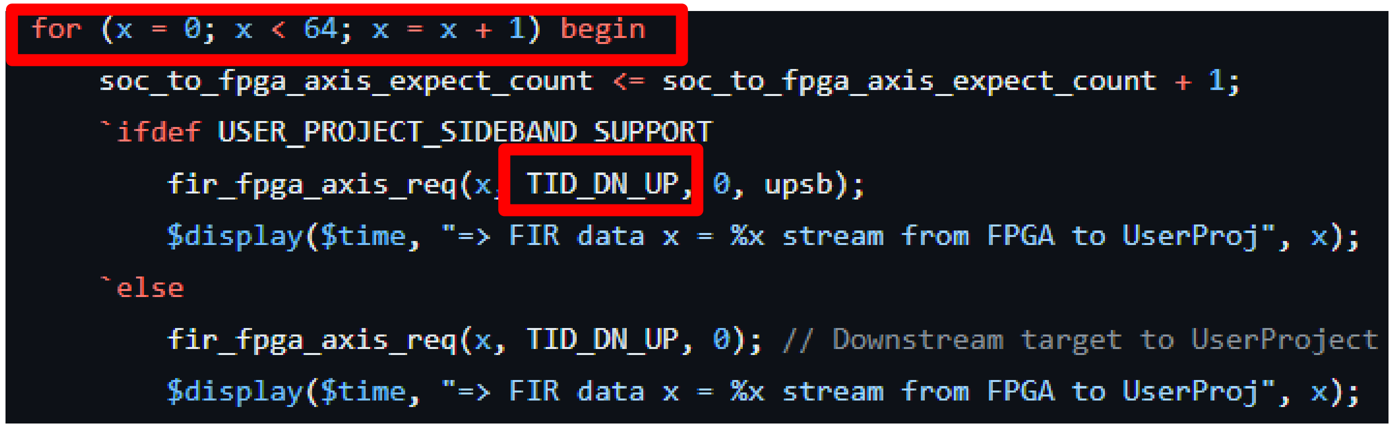


Direction	TID[1:0]	Source Module	Destination Module
Downstream	00	User DMA (M_AXIS_MM2S) in remote host (option extended user project)	User Project - the current active user project
Downstream	01	Axilite Master R/W in remote host (include Mail box write)	Axis-Axilite (include Mail box)
Upstream	00	User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

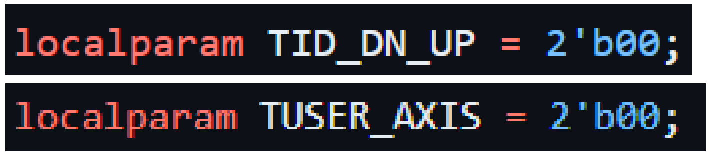


Test#1 – FIR stream data X, Y from FPGA side

- Looping 64 times for X_in (task: fpga_stream_x_in):

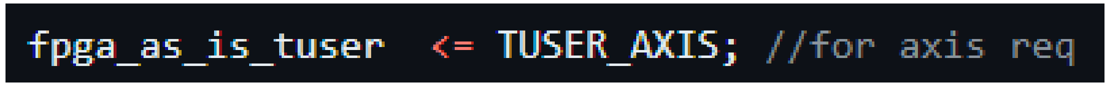


```
for (x = 0; x < 64; x = x + 1) begin
    soc_to_fpga_axis_expect_count <= soc_to_fpga_axis_expect_count + 1;
    `ifdef USER_PROJECT_SIDEBAND_SUPPORT
        fir_fpga_axis_req(x, TID_DN_UP, 0, upsb);
        $display($time, "=> FIR data x = %x stream from FPGA to UserProj", x);
    `else
        fir_fpga_axis_req(x, TID_DN_UP, 0); // Downstream target to UserProject
        $display($time, "=> FIR data x = %x stream from FPGA to UserProj", x);
```



```
localparam TID_DN_UP = 2'b00;
localparam TUSER_AXIS = 2'b00;
```

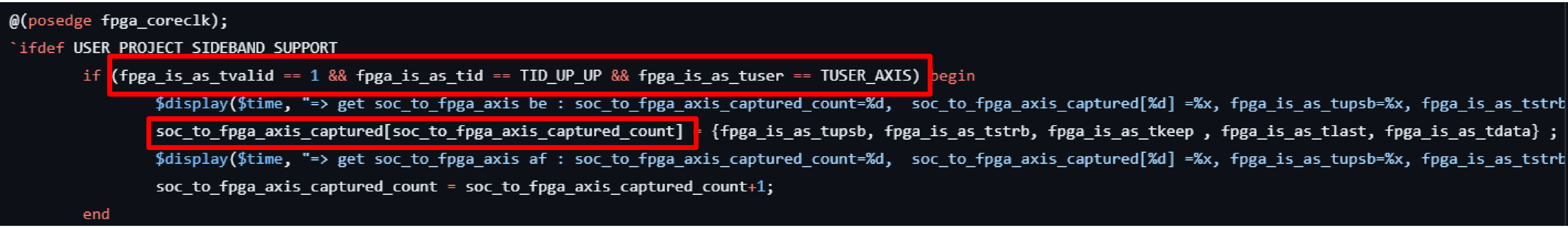
- In task fpga_stream_x_in:
  - Program TID_DN_UP (downstream, user project)
- In task fir_fpga_axis_req (refer to fpga_axis_req):
  - Program TUSER_AXIS(AXI-Stream transaction)



```
fpga_as_is_tuser  <= TUSER_AXIS; //for axis req
```

Test#1 – Check the Output Y

- Register soc_to_fpga_axis_captured&count
  - Which captured&count the upstream data from IO serdes to AS module
- fpga_is_as_tvalid, fpga_is_as_tid, fpga_is_as_tuser
  - tvalid for data handshake, tid for destination, tuser for transaction type



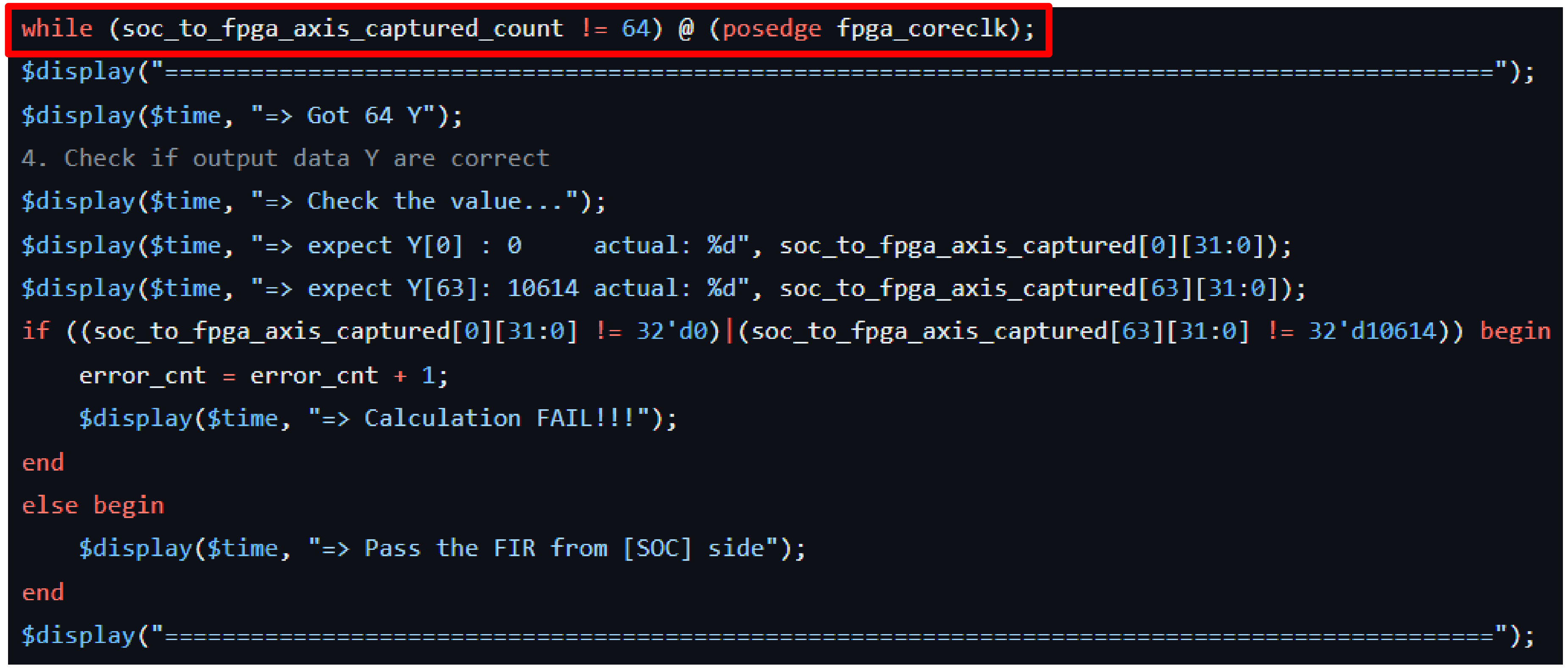
```
@(posedge fpga_coreclk);
`ifdef USER_PROJECT_SIDEBAND_SUPPORT
        if (fpga_is_as_tvalid == 1 && fpga_is_as_tid == TID_UP_UP && fpga_is_as_tuser == TUSER_AXIS) begin
                $display($time, "=> get soc_to_fpga_axis be : soc_to_fpga_axis_captured_count=%d,  soc_to_fpga_axis_captured[%d] =%x, fpga_is_as_tupsb=%x, fpga_is_as_tstr
                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} ;
                $display($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_tstr
                soc_to_fpga_axis_captured_count = soc_to_fpga_axis_captured_count+1;
        end
```

Above capture the upstream AXIS transaction data which destination is user project

https://github.com/vic9112/Advance_SOC/blob/main/lab01%20-%20fsic-sim/fsic_fpga/rtl/user/testbench/tb_fsic.v

Test#1 – Check the Output

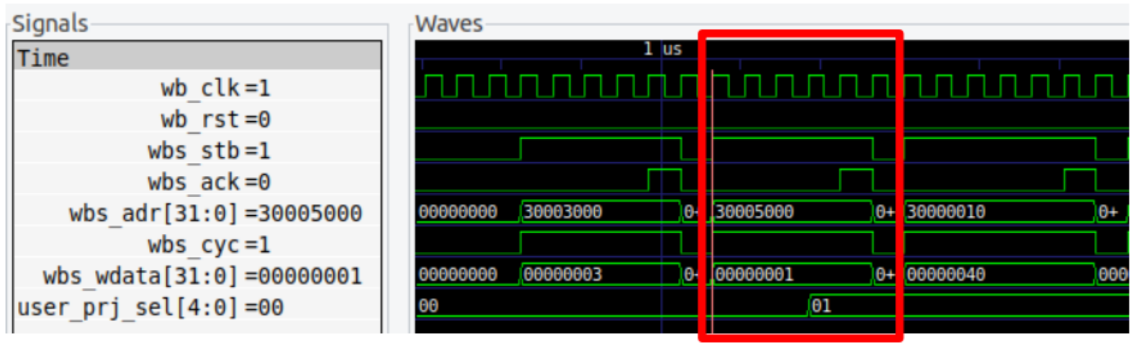
- By those characteristics:
  - Detect whether we've got 64(data length) Y and check the correctness



```
while (soc_to_fpga_axis_captured_count != 64) @ (posedge fpga_coreclk);
$display("============================================================================");
$display($time, "=> Got 64 Y");
4. Check if output data Y are correct
$display($time, "=> Check the value...");
$display($time, "=> expect Y[0] : 0     actual: %d", soc_to_fpga_axis_captured[0][31:0]);
$display($time, "=> expect Y[63]: 10614 actual: %d", soc_to_fpga_axis_captured[63][31:0]);
if ((soc_to_fpga_axis_captured[0][31:0] != 32'd0)|(soc_to_fpga_axis_captured[63][31:0] != 32'd10614)) begin
    error_cnt = error_cnt + 1;
    $display($time, "=> Calculation FAIL!!!");
end
else begin
    $display($time, "=> Pass the FIR from [SOC] side");
end
$display("============================================================================");
```

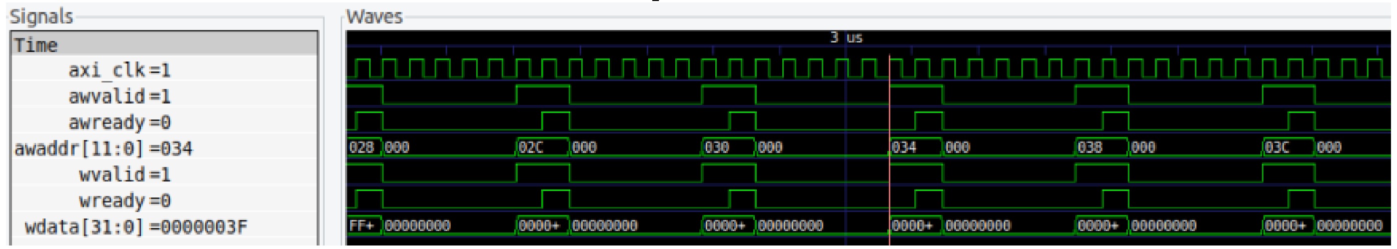
Test#1 – Waveform

- Configuration cycle (program 32'h3000_5000 to change user_prj_sel)

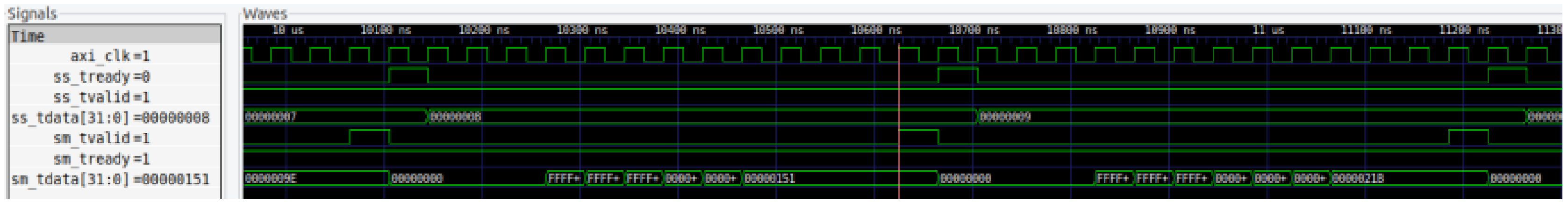


Takes 3T to change the user_prj_sel[4:0] and 1T ACK => latency: 5T

- AXI-Lite transaction cycles



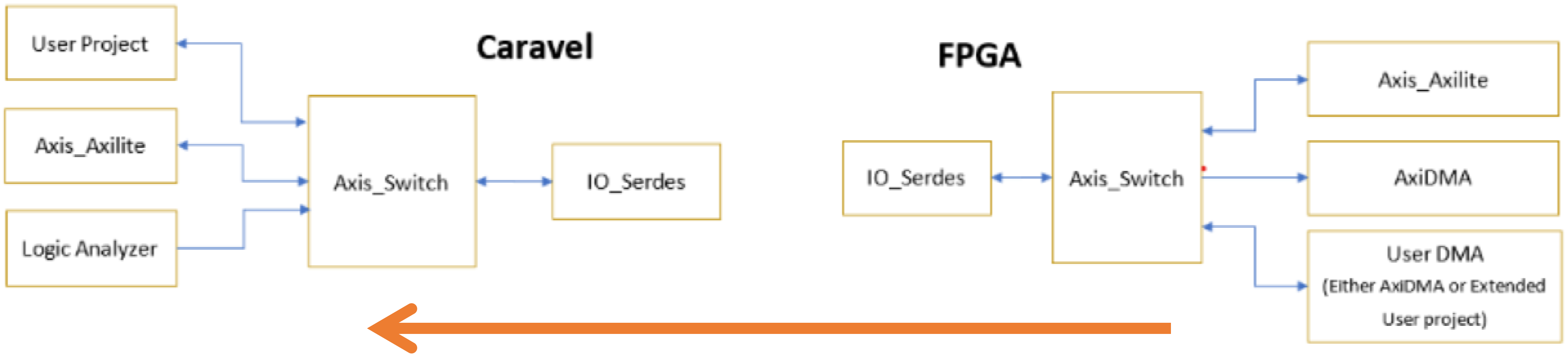
- Stream-in, stream-out



Test#2

Test#2 – FIR initialization from FPGA side

- FIR initialization from FPGA side (downstream)
  - Transaction type: AXI-Lite



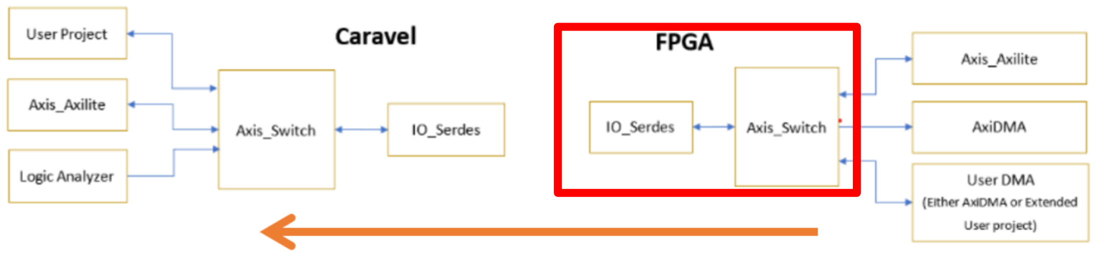
Test#2 – FIR initialization from FPGA side

- Purpose
  - AS module – destinate to AA s.t. AA can do the work
  - AA module generate the AXI-Lite write transaction

```
localparam TID_DN_AA = 2'b01;
localparam TUSER_AXILITE_WRITE = 2'b01;
```

- In task fpga_axilite_write_req:

```
fpga_as_is_tid <=  TID_DN_AA;         /
fpga_as_is_tuser <=  TUSER_AXILITE_WRITE;
```



- TID

Downstream	01	Axilite Master R/W in remote host (include Mail box write)	Axis-Axilite (include Mail box)

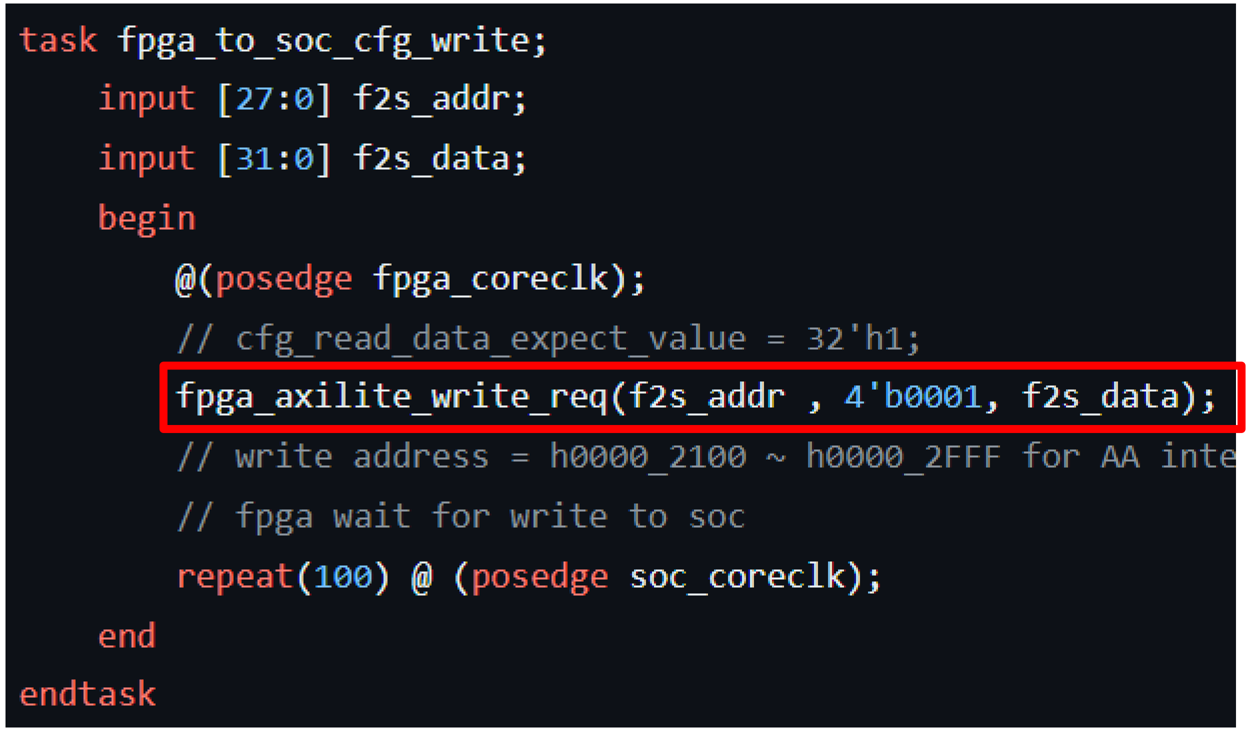
- TUSER

01	2	Axilite write transaction Address + Data. TAD 1^st^ T is the Byte-enable + address, i.e. {BE[3:0],ADDR[27:0]}, 2^nd^ T is the Data[31:0]. Note: Axilite write transaction only support 1T in data phase.

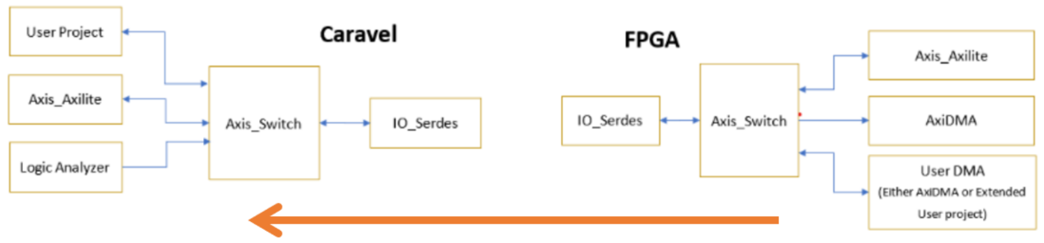
https://github.com/vic9112/Advance_SOC/blob/main/lab01%20-%20fsic-sim/fsic_fpga/rtl/user/testbench/tb_fsic.v

Test#2 – FIR initialization from FPGA side

- Task fpga_to_soc_cfg_write (refer to task6_fpga_to_soc_cfg_write)

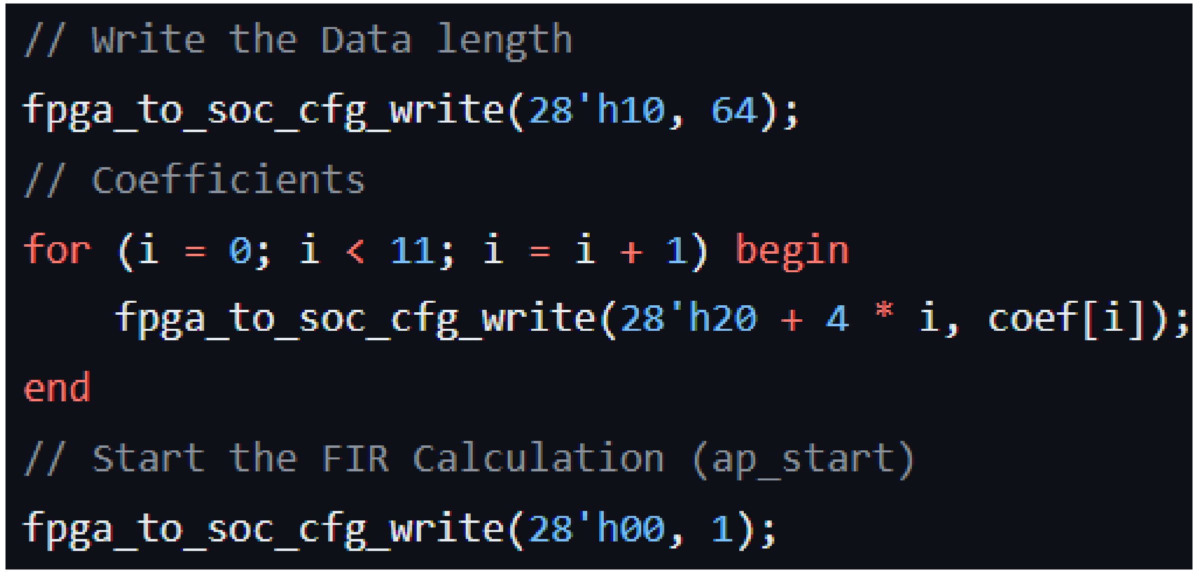


```
task fpga_to_soc_cfg_write;
    input [27:0] f2s_addr;
    input [31:0] f2s_data;
    begin
        @(posedge fpga_coreclk);
        // cfg_read_data_expect_value = 32'h1;
        fpga_axilite_write_req(f2s_addr , 4'b0001, f2s_data);
        // write address = h0000_2100 ~ h0000_2FFF for AA inte
        // fpga wait for write to soc
        repeat(100) @ (posedge soc_coreclk);
    end
endtask
```

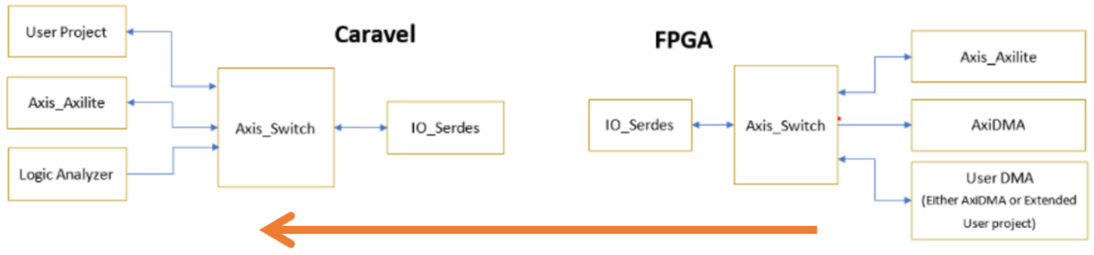


Test#2 – FIR initialization from FPGA side

- Initialization (data length, coefficient, ap_start)



```
// Write the Data length
fpga_to_soc_cfg_write(28'h10, 64);
// Coefficients
for (i = 0; i < 11; i = i + 1) begin
    fpga_to_soc_cfg_write(28'h20 + 4 * i, coef[i]);
end
// Start the FIR Calculation (ap_start)
fpga_to_soc_cfg_write(28'h00, 1);
```

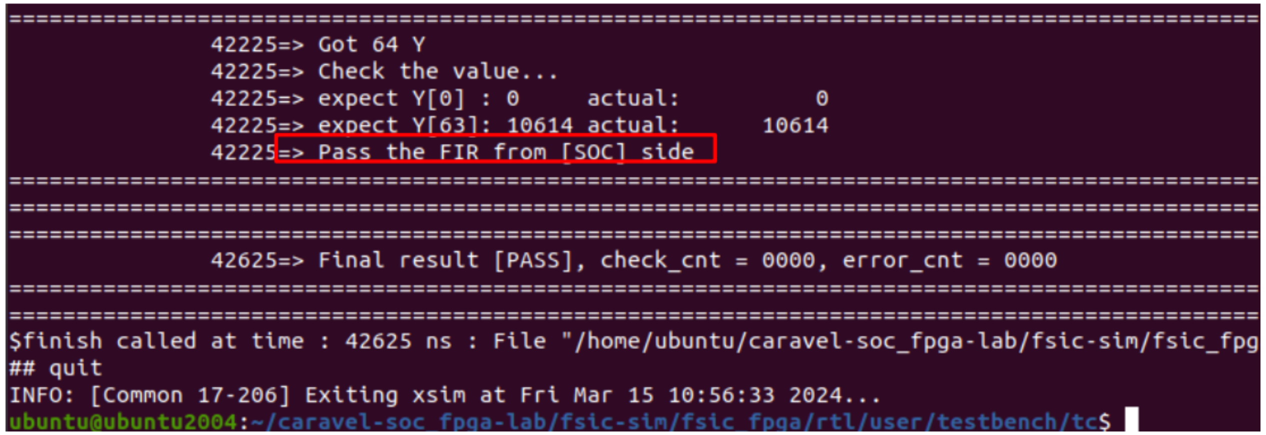


Test#2 – Stream X, Check Y

- Same as task#1

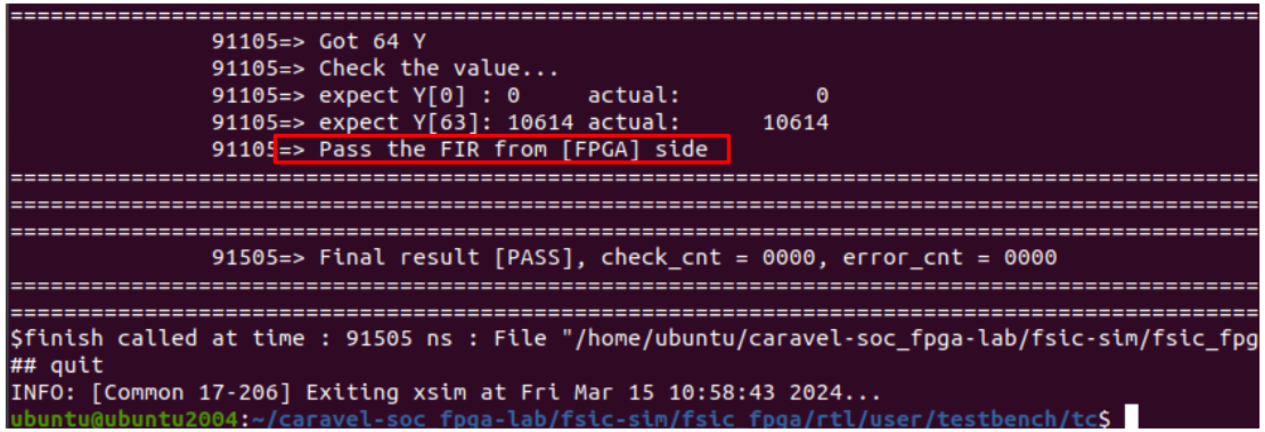
Result:

- Task#1



```
==========================================================================================
               42225=> Got 64 Y
               42225=> Check the value...
               42225=> expect Y[0] : 0      actual:          0
               42225=> expect Y[63]: 10614 actual:      10614
               42225=> Pass the FIR from [SOC] side
==========================================================================================
==========================================================================================
==========================================================================================
               42625=> Final result [PASS], check_cnt = 0000, error_cnt = 0000
==========================================================================================
==========================================================================================
$finish called at time : 42625 ns : File "/home/ubuntu/caravel-soc_fpga-lab/fsic-sim/fsic_fpg
## quit
INFO: [Common 17-206] Exiting xsim at Fri Mar 15 10:56:33 2024...
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab/fsic-sim/fsic_fpga/rtl/user/testbench/tc$
```

- Task#2



```
==========================================================================================
               91105=> Got 64 Y
               91105=> Check the value...
               91105=> expect Y[0] : 0      actual:          0
               91105=> expect Y[63]: 10614 actual:      10614
               91105=> Pass the FIR from [FPGA] side
==========================================================================================
==========================================================================================
==========================================================================================
               91505=> Final result [PASS], check_cnt = 0000, error_cnt = 0000
==========================================================================================
==========================================================================================
$finish called at time : 91505 ns : File "/home/ubuntu/caravel-soc_fpga-lab/fsic-sim/fsic_fpg
## quit
INFO: [Common 17-206] Exiting xsim at Fri Mar 15 10:58:43 2024...
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab/fsic-sim/fsic_fpga/rtl/user/testbench/tc$
```

Reference

- Original tb_fsic.v - (1)
- Self-Learning: FSIC Architecture – (2)

(1) https://github.com/bol-edu/caravel-soc_fpga-lab/blob/main/fsic-sim/fsic_fpga/rtl/user/testbench/tb_fsic.v
(2) https://hackmd.io/@vic9112/HykJ2Lpt6