Bridge of Life Education

Advanced SOC Design
Lab 4 – Caravel-FSIC FPGA

(teamwork)

Jiin Lai

Outline

- Introduction of the Lab
- Lab 4-1 Caravel-FSIC FPGA Simulation
- Lab 4-2 Caravel-FSIC FPGA Validation
- Lab 4-3 Requirement

Bridge of Life Education

Introduction of the Lab

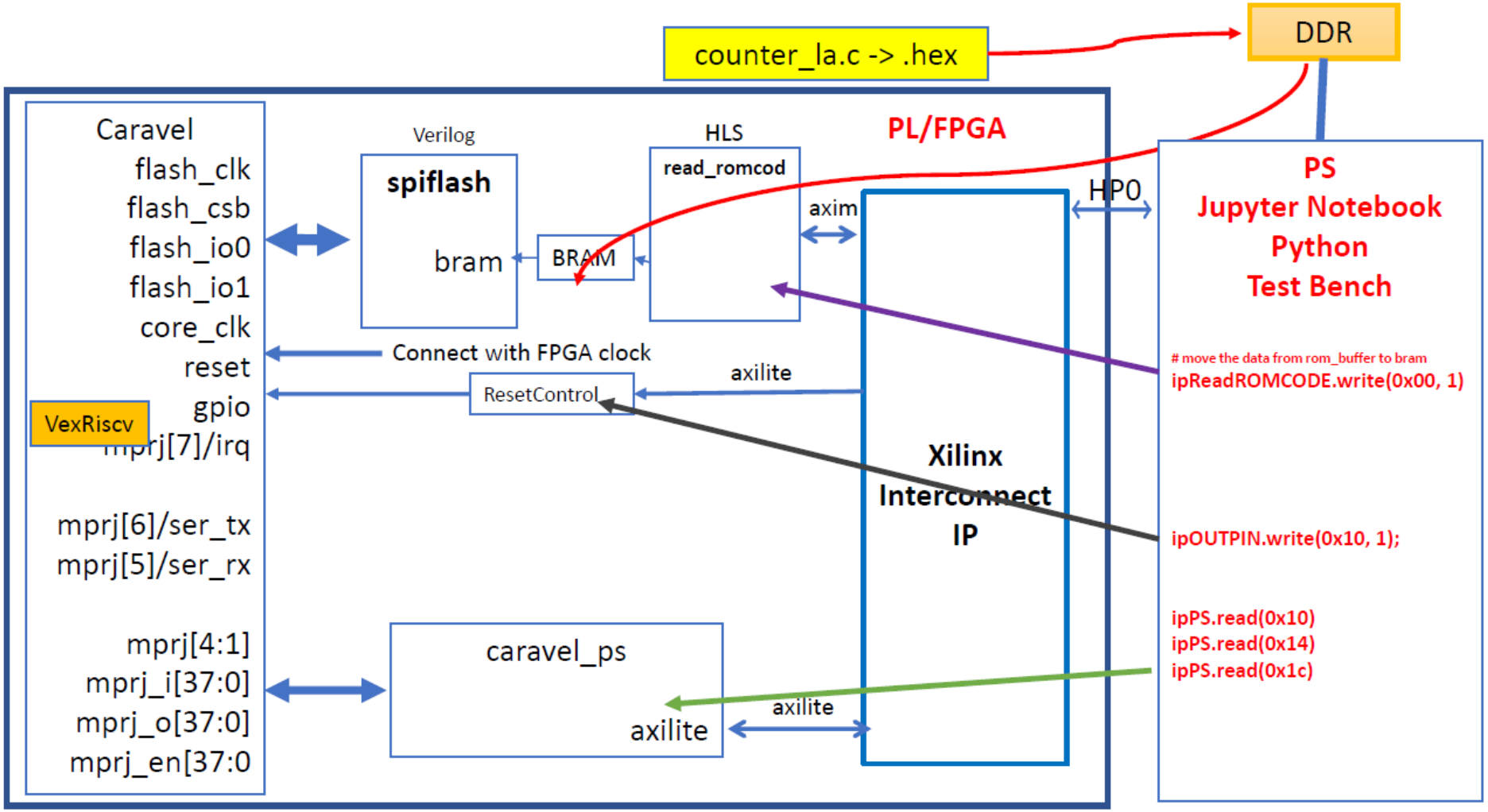
Objectives (Caravel+FSIC FPGA System)

- Target goal
  - Implement complete Caravel-FSIC, FPGA-FSIC and HLS-DMA
  - Use for Caravel Chip pre-tapeout validation
- Implementation
  - Replicate and study the Github implementation
    - Caravel-FSIC FPGA Simulation
    - Caravel-FSIC FPGA Validation
  - Integrate FIR design into user project
    - Add User-DMA for FIR in FPGA
    - System validation with Python

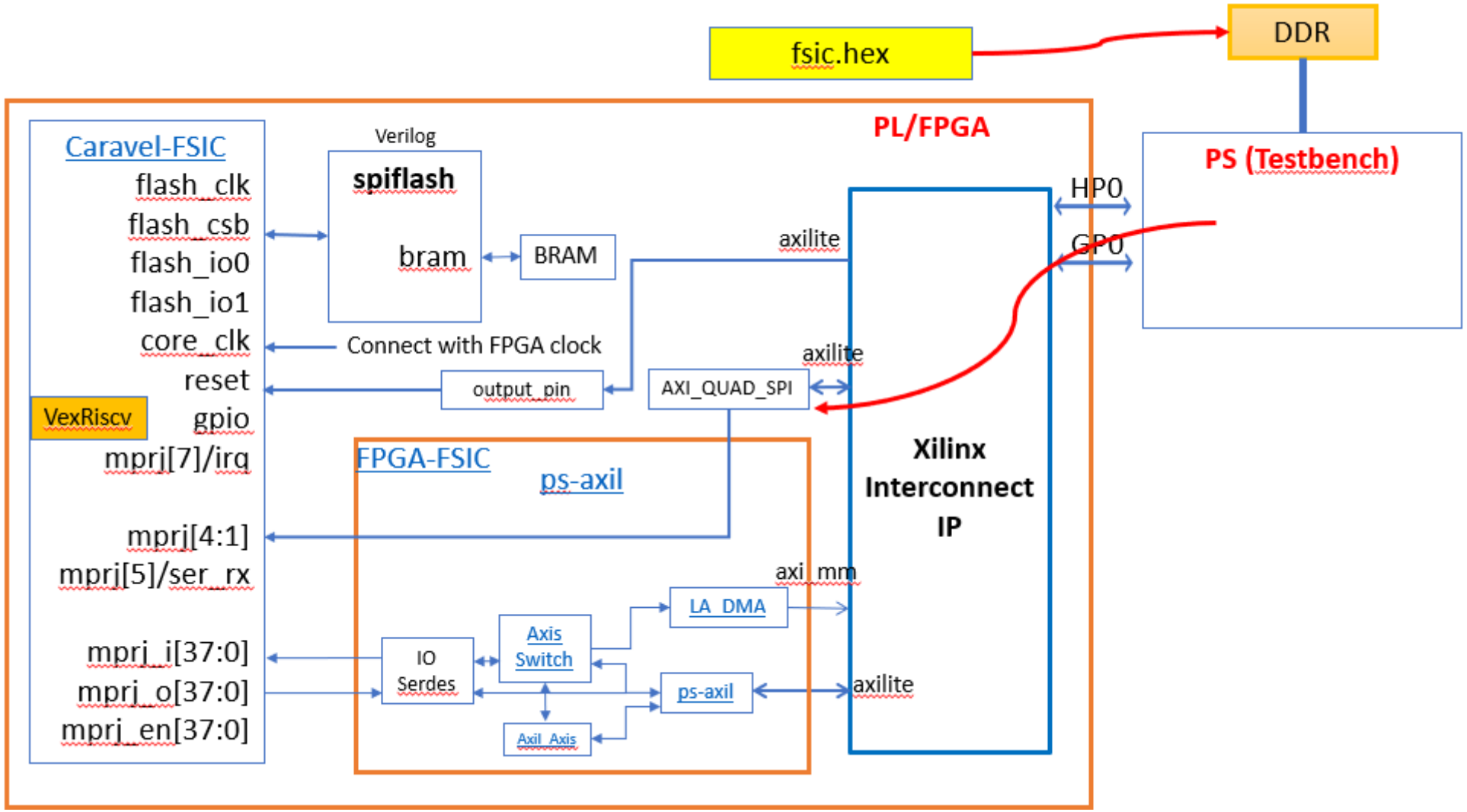
Lab Material

- PPT:
  - https://docs.google.com/presentation/d/1Fl_k2uk0jcEVM41cUaz2nSBOg_9lrjip/edit#slide=id.p1
- Video:
  - https://www.youtube.com/watch?v=xC7MUxudTwg
- Github:
  - https://github.com/bol-edu/fsic_fpga/tree/main
- UserDMA Reference:
  - https://github.com/JoshSu0/fsic_fpga/tree/fsic-231107/vivado/vitis_prj/hls_userdma

System Overview (Old system of lab5 in SoC Design)

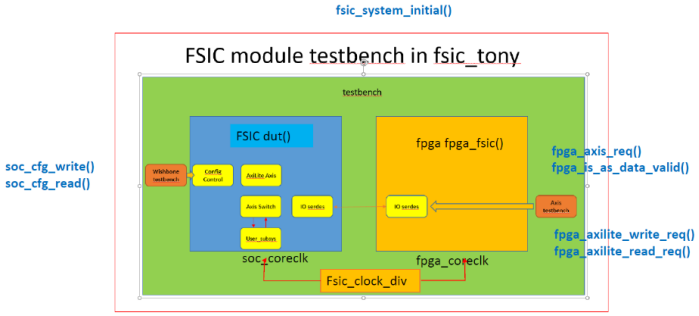


System Overview (The lab4 – fsic-fpga)

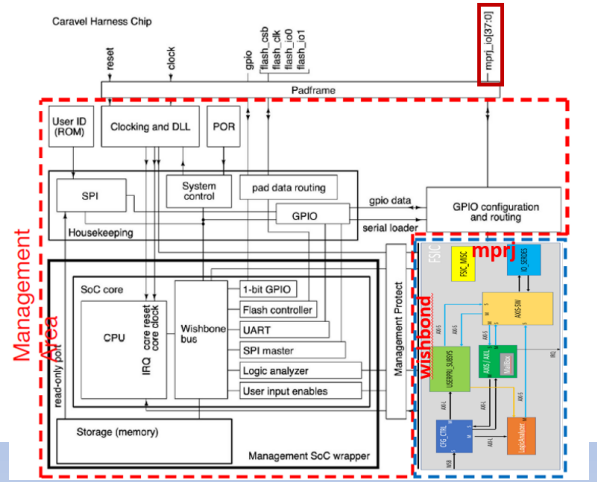


System Architecture Test Level

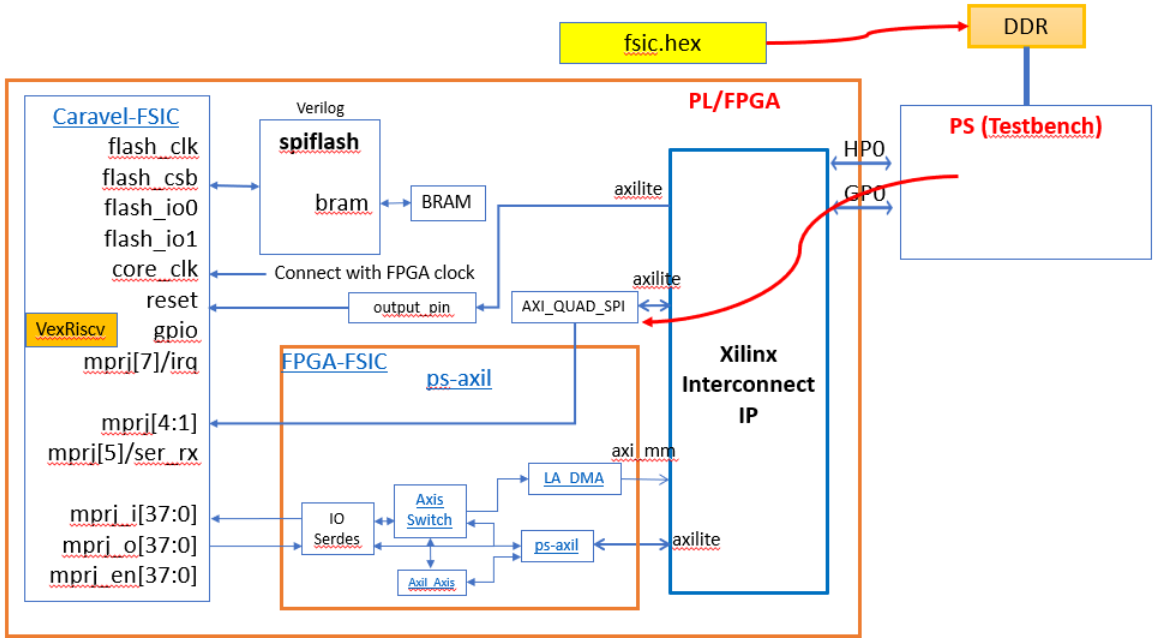
- IP design simulation
  - fsic_fpga/rtl/user/testbench/tb_fsic.v (lab1)



- Caravel soc simulation
  - ./testbench/fsic/fsic_tb.v



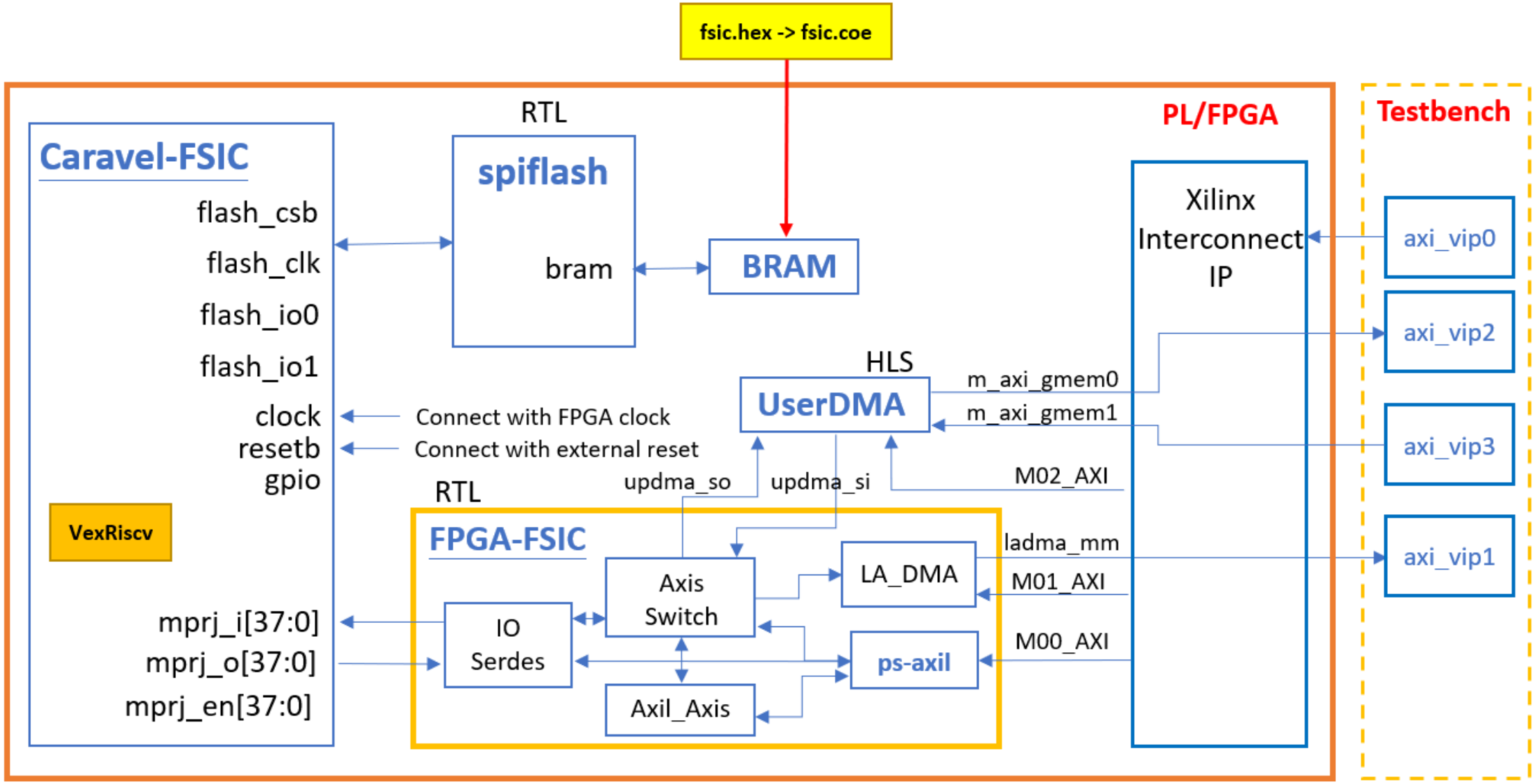
- System simulation by vivado
  - ./vivado/fsic_tb.sv



Bridge of Life Education

Lab 4-1 Caravel-FSIC FPGA Simulation

Lab4-1. Introduction



AXI Verification IP (VIP)

- AXI Verification IP (VIP) core is developed to support the simulation of customer designed AXI-based IP

- Reference: https://docs.amd.com/r/en-US/pg267-axi-vip/Introduction

Figure: AXI Master VIP

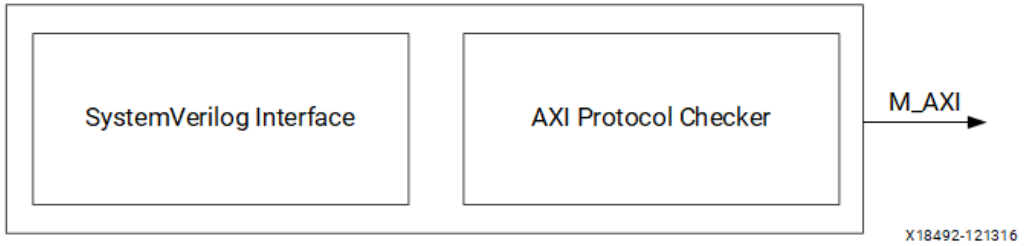
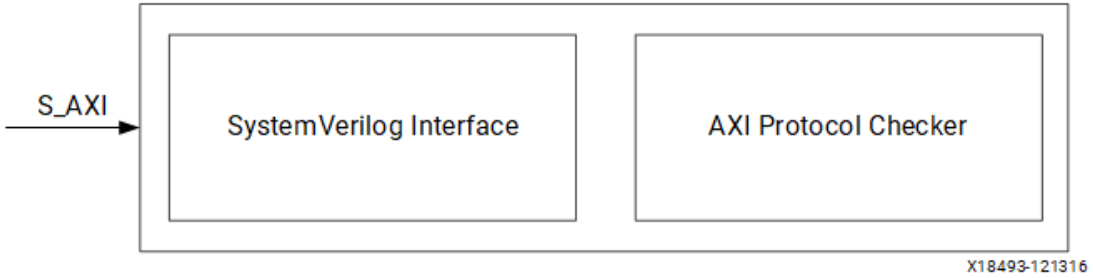
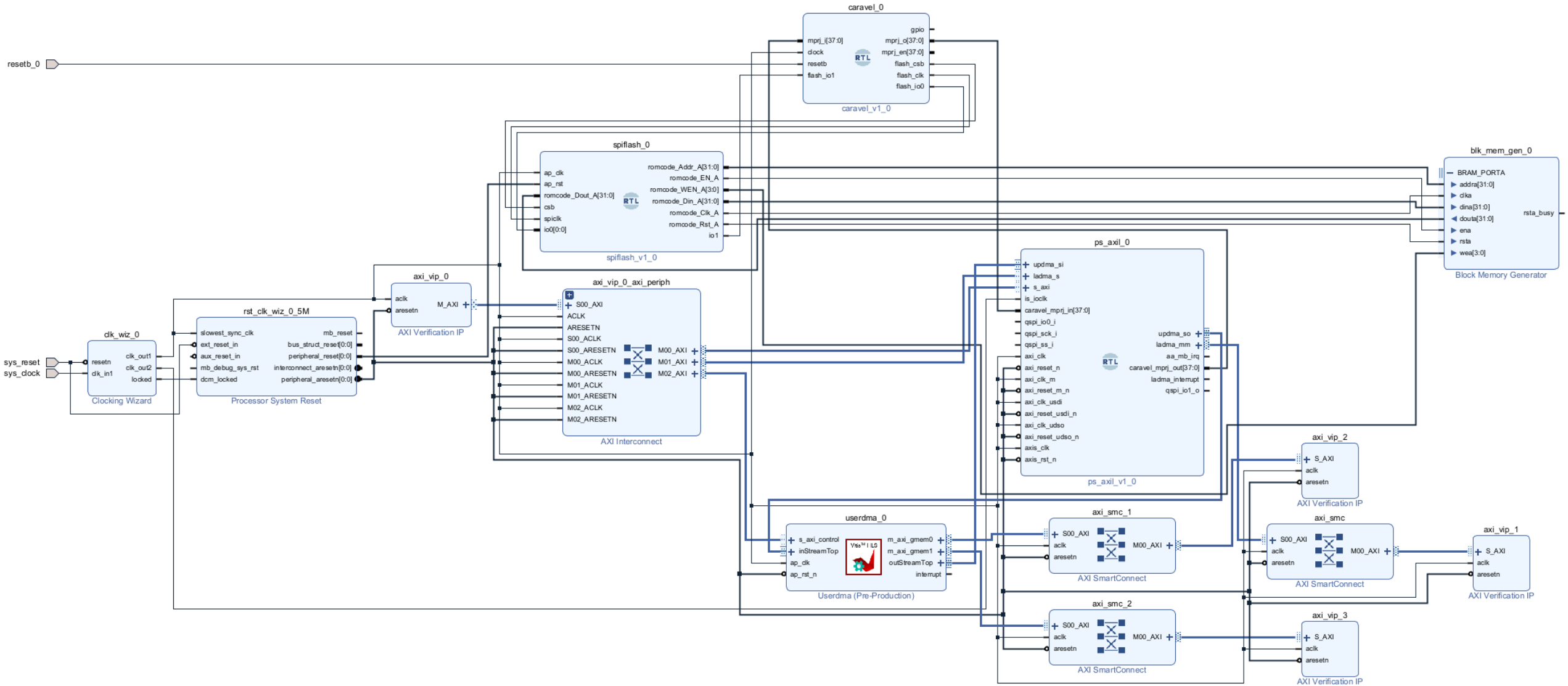


Figure: AXI Slave VIP



Caravel-FSIC FPGA Simulation Block Design



Block Design

- ps_axi
  - PS side request forwarding
  - DMA request forwarding
  - For simulation or FPGA
  - User dma stream in、out
    - If user design need transaction to user DMA
  - LA dma
    - Used one configuration cycle by PS side to config la DMA interface
  - s_axi(slave axi)
    - Target is remote side or internal AXIL_AXIS module
    - If we need to send cycle from PS side to caravel side, it will go trough AXIL_AXIS to AXIS_SW to IO_SERDES to mprj out that to pass cycle to the caravel side
  - MPRJ_IN/OUT
    - The interface for IO_SERDES connect to caravel
  - LADMA_MM
    - LA_DMA receive request from caravel side, it will flesh to ps side memory by LADMA_MM
  - PS_AXIL
    - spi interface (qspi_io0_i、qspi_sck_i、qspi_ss_i、qspi_io1_o)
    - Board level design for remote control spi rom on the daughter board
    - fw preload to spi rom
  - ladma_interrupt
    - Transfer finish will send interrupt
    - Not used, instead by polling mode
  - aa_mb_irq
    - The mail box of PS side receive remote side write update
    - Not used, instead by polling mode

Block Design

- axi_vip * 4
  - axi_vip_0
    - PS master cycle generation
    - Config module in the caravel and ps_axi
  - axi_vip_1
    - Memory slave module
    - Receive data from LA in caravel side
    - Flesh to the memory slave module from ps_axi
  - axi_vip_2、3
    - To be the user DMA target
    - One is read data pattern, the other one is to write data pattern
- There are two clock
  - proc_clock
  - io_clock
- Block memory
  - Load coe file for RISC-V

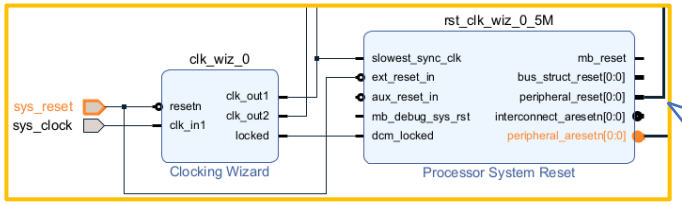
Build Flow

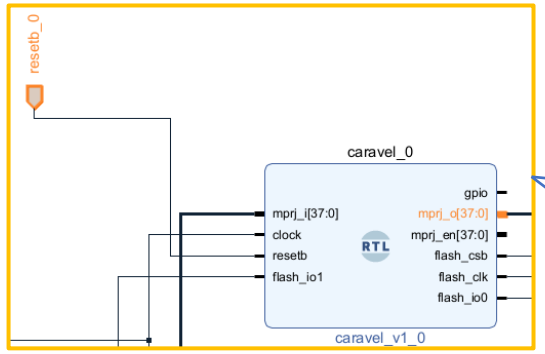
- Building Environment
  - Ubuntu 20.04 with Vivado 2022.1
- Simulation Building Steps
  - git clone https://github.com/bol-edu/fsic_fpga
  - cd fsic_fpga
  - cd vivado
  - ./run_vivado_fsic_sim
    - Tcl file: vvd_caravel_fpga_fsic_sim.tcl
    - Coe file: fsic.coe
- Waveform review
  - ./open_wave
- Open built Vivado Project by Vivado GUI
  - File->Open->Project
  - Target Project file: fsic_fpga/vivado/vvd_caravel_fpga_sim/vvd_caravel_fpga_sim.xpr

Firmware Code Explained

- main()@/fsic_fpga/testbench/fsic/fsic.c
- Major behaviors
  - mprj_io configuration
    - [37]: signal for talking between SoC and PS
    - [36]: io_seders IO_CLK
    - [35]: io_seders TXC
    - [34:22]: io_seders TXD
    - [21]: io_seders RXC
    - [20:8] io_seders RXD
  - Simple FSIC module read/write test
    - Indication to mprj[37]
  - Mailbox write request handling
    - Judgement by value by CC's register
      - 5: PS side request FW to do FSIC mailbox write
      - 6: PS side request FW to do FSIC mailbox write interrupt enable
      - 7: PS side request FW to do FSIC mailbox write value checking
      - 8: PS side request FW to do FSIC mailbox interrupt checking and clear

Testbench code flow -- Initial





RegisterName	Offset Address	Description
IO Serdes Control	'h0	Control Register block Definition bit 0: rxen_ctl FSIC FW set this bit to 1 when GPIO config setting is done for io serdes. set this bit within 100ms after reset de-assert bit 1: txen_ctl FSIC FW set this bit to 1 when remote side had enable rx. set this bit after 200ms after reset de-assert

system_reset_t

peripheral_reset_t

caravel_reset_t

fw_worked_t

is_txen_t

Fpga2Soc_CfgRead()

Fpga2Soc_CfgWrite()

FpgaLocal_CfgRead()

SocLocal_MbWrite()

FpgaLocal_MbWrite()

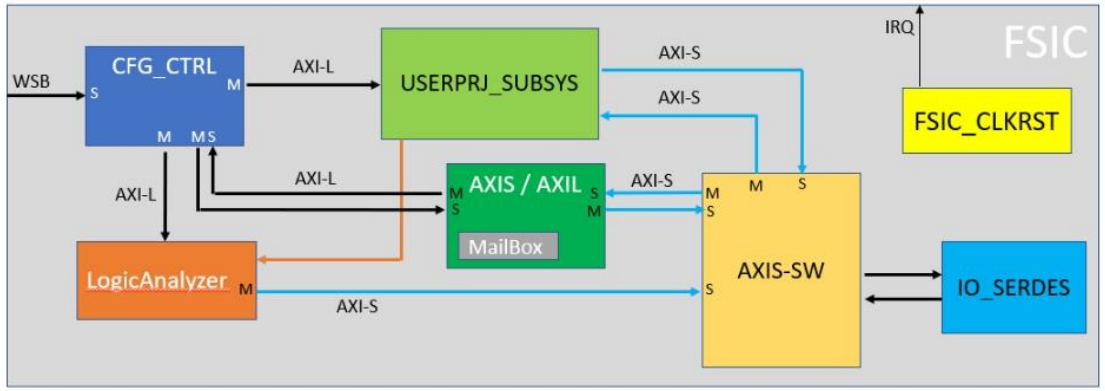
SocLa2DmaPath()

SocUp2DmaPath()

Fpga2Soc_CfgRead()

SoC side: Config_Ctrl Implementation Specification

- User Projects (32'h3000_**0**000~32'h3000_**0**xxx)
- Logic_Analyzer (32'h3000_**1**000~32'h3000_**1**xxx)
- Axis_Axilite (32'h3000_**2**000~32'h3000_**2**xxx)
  - Mailbox
- IO Serdes (32'h3000_**3**000~32'h3000_**3**xxx)
- Axis_Switch (32'h3000_**4**000~32'h3000_**4**xxx)
- Config Control (32'h3000_**5**000~32'h3000_**5**xxx)



FPGA side: Base address

Diagram | Address Editor | Address Map | ps_axil.v | config_ctrl.v | axil_axis.sv | fsic_mprj_io.v | fsic_clkrst.v | user_subsys.v

Assigned (6) Unassigned (0) Excluded (0) Hide All

Name	Interface	Slave Segment	Master Base Address	Range	Master High Address
Network 0					
/axi_vip_0					
/axi_vip_0/Master_AXI (32 address bits : 4G)					
/ps_axil_0/ladma_s	ladma_s	reg0	0x6000_8000	4K	0x6000_8FFF
/ps_axil_0/s_axi	s_axi	reg0	0x6000_0000	32K	0x6000_7FFF
/userdma_0/s_axi_control	s_axi_control	Reg	0x6000_9000	4K	0x6000_9FFF

```
8177458=> Starting Fpga2Soc_CfgRead() test...
8177458=> ============================================================
8177458=> Fpga2Soc_Read testing: SOC_CC
8188058=> AXI4LITE_READ_BURST 60005000, value: 001f, resp: 00
8198058=> Fpga2Soc_Read SOC_CC offset 000 = 0000001f, PASS
8198058=> Fpga2Soc_Read testing: SOC_AS
8208458=> AXI4LITE_READ_BURST 60004000, value: 0006, resp: 00
8218458=> Fpga2Soc_Read SOC_AS = 00000006, PASS
8218458=> Fpga2Soc_Read testing: SOC_IS
8228858=> AXI4LITE_READ_BURST 60003000, value: 0001, resp: 00
8238858=> Fpga2Soc_Read SOC_IS = 00000001, PASS
8238858=> Fpga2Soc_Read testing: SOC_LA
8249258=> AXI4LITE_READ_BURST 60001000, value: 0000, resp: 00
8259258=> Fpga2Soc_Read SOC_LA = 00000000, PASS
8259258=> End Fpga2Soc_CfgRead() test...
8259258=> ============================================================
```

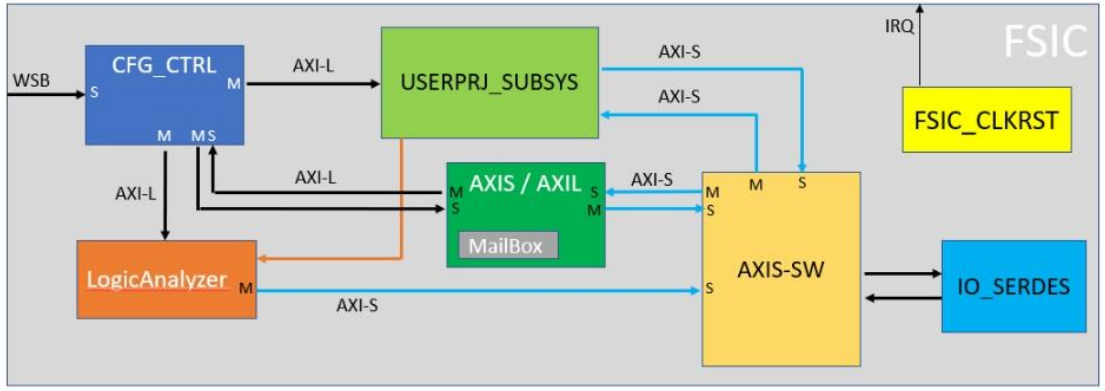
https://drive.google.com/file/d/1DA6b1rjjipHwwhJwoaZ6WLQZgJnhVeM7/view?usp=sharing

Fpga2Soc_CfgWrite()

Config_Ctrl Implementation Specification

Configuration Control Group: 32'h3000_5000~32'h30000_5FFF

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




```
8259258=> Starting Fpga2Soc_CfgWrite() test...
8259258=> ================================================================
8259258=> Fpga2Soc_Write testing: SOC_CC
8261658=> AXI4LITE_WRITE_BURST 60005000, value: 0000, resp: 00
8273858=> AXI4LITE_READ_BURST 60005000, value: 0000, resp: 00
8273858=> #00000000, Fpga2Soc_Write SOC_CC offset 000 = 00000000, PASS
8273858=> Fpga2Soc_Read testing: SOC_UP
8284458=> AXI4LITE_READ_BURST 60000000, value: 0000, resp: 00
8294458=> Fpga2Soc_Read SOC_UP offset 000 = 00000000, PASS
8294458=> Fpga2Soc_Read testing: SOC_UP
8304858=> AXI4LITE_READ_BURST 60000004, value: 0280, resp: 00
8314858=> Fpga2Soc_Read SOC_UP offset 004 = 00000280, PASS
8314858=> Fpga2Soc_Read testing: SOC_UP
8325258=> AXI4LITE_READ_BURST 60000008, value: 01e0, resp: 00
8335258=> Fpga2Soc_Read SOC_UP offset 008 = 000001e0, PASS
8335258=> Fpga2Soc_Read testing: SOC_UP
8345658=> AXI4LITE_READ_BURST 6000000c, value: 0001, resp: 00
8355658=> Fpga2Soc_Read SOC_UP offset 00c = 00000001, PASS
8358058=> AXI4LITE_WRITE_BURST 60005000, value: 0001, resp: 00
8370258=> AXI4LITE_READ_BURST 60005000, value: 0001, resp: 00
8370258=> #00000001, Fpga2Soc_Write SOC_CC offset 000 = 00000001, PASS
8372858=> AXI4LITE_WRITE_BURST 60005000, value: 0002, resp: 00
8385058=> AXI4LITE_READ_BURST 60005000, value: 0002, resp: 00
8385058=> #00000002, Fpga2Soc_Write SOC_CC offset 000 = 00000002, PASS
8387658=> AXI4LITE_WRITE_BURST 60005000, value: 0003, resp: 00
8399858=> AXI4LITE_READ_BURST 60005000, value: 0003, resp: 00
8399858=> #00000003, Fpga2Soc_Write SOC_CC offset 000 = 00000003, PASS
8402458=> AXI4LITE_WRITE_BURST 60005000, value: 0004, resp: 00
8414658=> AXI4LITE_READ_BURST 60005000, value: 0004, resp: 00
8414658=> #00000004, Fpga2Soc_Write SOC_CC offset 000 = 00000004, PASS
8414658=> End Fpga2Soc_CfgWrite() test...
8414658=> ================================================================
```

https://drive.google.com/file/d/1DA6b1rjjipHwwhJwoaZ6WLQZgJnhVeM7/view?usp=sharing

FpgaLocal_CfgRead()

Diagram | Address Editor | Address Map | ps_axil.v | config_ctrl.v | axil_axis.sv | fsic_mprj_io.v | fsic_clkrst.v | user_subsys.v

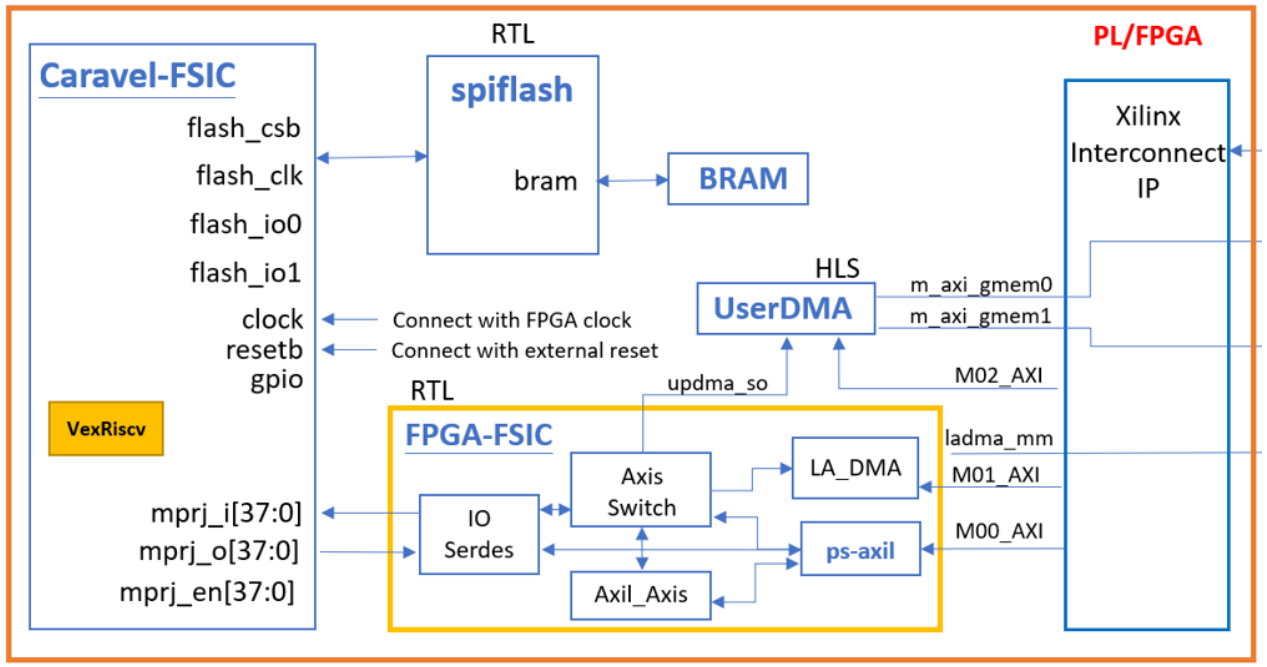
Assigned (6) Unassigned (0) Excluded (0) Hide All

Name	Interface	Slave Segment	Master Base Address	Range	Master High Address
Network 0					
/axi_vip_0					
/axi_vip_0/Master_AXI (32 address bits : 4G)					
/ps_axil_0/ladma_s	ladma_s	reg0	0x6000_8000	4K	0x6000_8FFF
/ps_axil_0/s_axi	s_axi	reg0	0x6000_0000	32K	0x6000_7FFF
/userdma_0/s_axi_control	s_axi_control	Reg	0x6000_9000	4K	0x6000_9FFF

```
//////////////////////
// Always for Target Selection //
//////////////////////
always @ ( posedge axi_clk or negedge axi_reset_n)
begin
    if ( !axi_reset_n )
    begin
        aa_enable_o <= 1'b0;
        as_enable_o <= 1'b0;
        is_enable_o <= 1'b0;
    end else
    begin
        aa_enable_o <= ( (ps_axi_request_add[31:12] >= 20'h60000) && (ps_axi_request_add[31:12] <= 20'h60005) )? 1'b1 : 1'b0;
        as_enable_o <= ( ps_axi_request_add[31:12] == 20'h60006 )? 1'b1 : 1'b0;
        is_enable_o <= ( ps_axi_request_add[31:12] == 20'h60007 )? 1'b1 : 1'b0;
    end
end
```

fsic_tb.v

```
localparam  SOC_UP = 16'h0000;
localparam  SOC_LA = 16'h1000;
localparam  PL_AA_MB = 16'h2000;
localparam  PL_AA = 16'h2100;
localparam  SOC_IS = 16'h3000;
localparam  SOC_AS = 16'h4000;
localparam  SOC_CC = 16'h5000;
localparam  PL_AS = 16'h6000;
localparam  PL_IS = 16'h7000;
localparam  PL_DMA = 16'h8000;
localparam  PL_UPDMA = 16'h9000;
```




```
8414658=> Starting FpgaLocal_CfgRead() test...
8414658=> ============================================================================
8414658=> FpgaLocal_CfgRead testing: PL_AS
8416458=> AXI4LITE_READ_BURST 60006000, value: 0006, resp: 00
8426458=> FpgaLocal_CfgRead PL_AS offset 000 = 00000006, PASS
8426458=> FpgaLocal_CfgRead testing: PL_IS
8428058=> AXI4LITE_READ_BURST 60007000, value: 0003, resp: 00
8438058=> FpgaLocal_CfgRead PL_IS = 00000003, PASS
8438058=> FpgaLocal_CfgRead testing: PL_DMA
8439058=> AXI4LITE_READ_BURST 60008000, value: 0004, resp: 00
8449058=> FpgaLocal_CfgRead PL_DMA = 00000004, PASS
8449058=> FpgaLocal_CfgRead testing: PL_AA
8452458=> AXI4LITE_READ_BURST 60002100, value: 0000, resp: 00
8462458=> FpgaLocal_CfgRead PL_AA = 00000000, PASS
8462458=> FpgaLocal_CfgRead testing: PL_UPDMA
8463458=> AXI4LITE_READ_BURST 60009000, value: 0004, resp: 00
8473458=> FpgaLocal_CfgRead PL_UPDMA = 00000004, PASS
8473458=> End FpgaLocal_CfgRead() test...
8473458=> ============================================================================
```

write address = h0000_2100 ~ h0000_2FFF for AA internal register

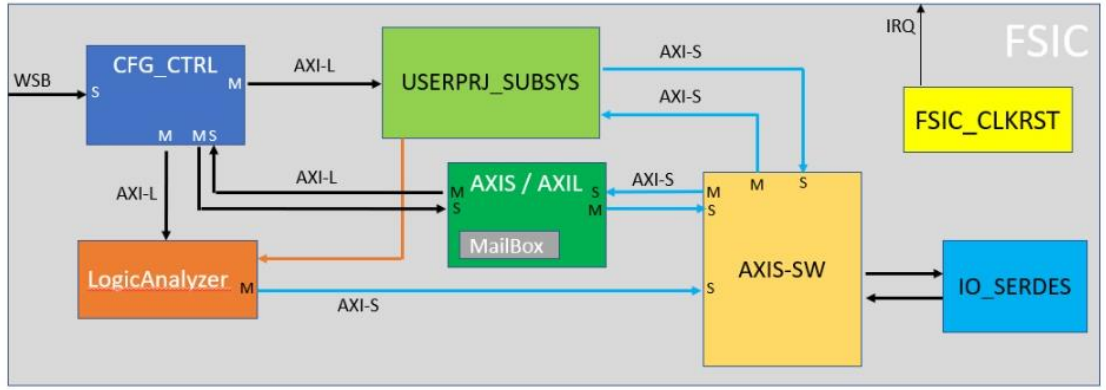
SocLocal_MbWrite()

fisc.c (firmware code)

```
while(1) {
    value = reg_fsic_cc;
    switch (value)
    {
    case 5:
        reg_mprj_datah = 0x00;  //set mprj_io[37] to 1'b0 to indicate FW going to waiting fpga MB test
        reg_fsic_aa_mb0 = 0x5a5a5a5a;
        reg_fsic_cc = 0x00000004;
        reg_mprj_datah = 0x20;  //set mprj_io[37] to 1'b0 to indicate FW going to waiting fpga MB test
        break;
```

Read interrupt status, aa_regs offset 4, bit 0 should be 0 by default

Clear interrupt status, write aa_regs offset 4, bit 0 = 1



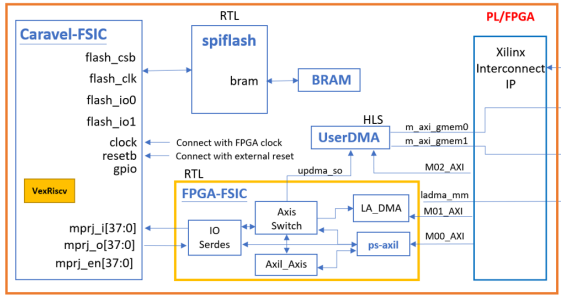

```
8473458=> Starting SocLocal_MbWrite() test...
8473458=> ================================================================================
8473458=> FpgaLocal_Write : PL_AA
8475858=> AXI4LITE_WRITE_BURST 60002100, value: 0001, resp: 00
8479458=> AXI4LITE_READ_BURST 60002100, value: 0001, resp: 00
8479458=> FpgaLocal_Write PL_AA offset 000 = 00000001, PASS
8479458=> Fpga2Soc_Write : SOC_CC
8482058=> AXI4LITE_WRITE_BURST 60005000, value: 0005, resp: 00
9937658=> FW starts MB writing, caravel_0_mprj_o[37] = 0
10155858=> FW finishs MB writing, caravel_0_mprj_o[37] = 1
10155858=> Fpga2Soc_Read: SOC_CC
10166458=> AXI4LITE_READ_BURST 60005000, value: 0004, resp: 00
10166458=> Fpga2Soc_Read SOC_CC = 00000004, PASS
10166458=> FpgaLocal_Read: PL_AA_MB
10170058=> AXI4LITE_READ_BURST 60002000, value: 5a5a5a5a, resp: 00
10170058=> FpgaLocal_Read PL_AA_MB = 5a5a5a5a, PASS
10170058=> aa_mb_irq status = 1
10170058=> FpgaLocal_Read: PL_AA
10173658=> AXI4LITE_READ_BURST 60002104, value: 0001, resp: 00
10173658=> FpgaLocal_Read PL_AA = 00000001, PASS
10173658=> FpgaLocal_Write : PL_AA
10176258=> AXI4LITE_WRITE_BURST 60002104, value: 0001, resp: 00
10179858=> AXI4LITE_READ_BURST 60002104, value: 0000, resp: 00
10179858=> FpgaLocal_Write PL_AA offset 004 = 00000000, PASS
10179858=> End SocLocal_MbWrite() test...
10179858=> ================================================================================
```

Reference code: axi_ctrl_logic.sv、tb_fsic.v

FpgaLocal_MbWrite()

fisc.c (firmware code)

```
case 6:
    reg_mprj_datah = 0x20;  //set mprj_io[37] to 1'b1 first
    reg_fsic_aa_irq_en = 1;
    value = reg_fsic_aa_irq_en;
    if (value==1) {
        reg_mprj_datah = 0x00;  //set mprj_io[37] to 1'b0 to indicate correct data
        reg_fsic_cc = 0x00000004;
    } else {
        reg_mprj_datah = 0x20;  //set mprj_io[37] to 1'b1 to indicate incorrect data
        reg_fsic_cc = 0x00000004;
    }
    break;
case 7:
    reg_mprj_datah = 0x20;  //set mprj_io[37] to 1'b1 first
    value = reg_fsic_aa_mb0;
    if (value==0x5555aaaa) {
        reg_mprj_datah = 0x00;  //set mprj_io[37] to 1'b0 to indicate correct data
        reg_fsic_cc = 0x00000004;
    } else {
        reg_mprj_datah = 0x20;  //set mprj_io[37] to 1'b1 to indicate incorrect data
        reg_fsic_cc = 0x00000004;
    }
    break;
case 8:
    reg_mprj_datah = 0x20;  //set mprj_io[37] to 1'b1 first
    value = reg_fsic_aa_irq_sts;
    if (value==1) {
        reg_fsic_aa_irq_sts = 1;
        value = reg_fsic_aa_irq_sts;
        if (value==0) {
            reg_mprj_datah = 0x00;  //set mprj_io[37] to 1'b0 to indicate correct data
            reg_fsic_cc = 0x00000004;
        } else {
            reg_mprj_datah = 0x20;  //set mprj_io[37] to 1'b1 to indicate incorrect data
            reg_fsic_cc = 0x00000004;
        }
    } else {
        reg_mprj_datah = 0x20;  //set mprj_io[37] to 1'b1 to indicate incorrect data
        reg_fsic_cc = 0x00000004;
    }
    break;
```




```
10179858=> Starting FpgaLocal_MbWrite() test...
10179858=> ================================================================
10179858=> Fpga2Soc_Write: SOC_CC
10182458=> AXI4LITE_WRITE_BURST 60005000, value: 0006, resp: 00
10182458=> Wating FW complete the request enabling aa irq...
11998058=> FW complete the request. offset 000 = 00000004, PASS
11998058=> caravel_0_mprj_o[37] = 0, PASS
11998058=> FpgaLocal_Write : PL_AA_MB
12000658=> AXI4LITE_WRITE_BURST 60002000, value: 5555aaaa, resp: 00
12006058=> AXI4LITE_READ_BURST 60002000, value: 5555aaaa, resp: 00
12006058=> FpgaLocal_Read PL_AA_MB = 5555aaaa, PASS
12006058=> Fpga2Soc_Write: SOC_CC
12008658=> AXI4LITE_WRITE_BURST 60005000, value: 0007, resp: 00
12008658=> Wating FW complete the request SocLocal MB data checking ...
13375458=> FW complete the request. offset 000 = 00000004, PASS
13375458=> caravel_0_mprj_o[37] = 0, PASS
13375458=> Fpga2Soc_Write: SOC_CC
13378058=> AXI4LITE_WRITE_BURST 60005000, value: 0008, resp: 00
13378058=> Wating FW complete the request clear SocLocal aa irq...
14969258=> FW complete the request. offset 000 = 00000004, PASS
14969258=> caravel_0_mprj_o[37] = 0, PASS
14969258=> End FpgaLocal_MbWrite() test...
14969258=> ================================================================
```

SocLa2DmaPath()

LADMA/FPGA (PL_DMA)

```
//------------------------Address Info-------------------
// 0x00 : Control signals
//        bit 0  - ap_start (Read/Write/COH)
//        bit 1  - ap_done (Read/COR)
//        bit 2  - ap_idle (Read)
//        bit 3  - ap_ready (Read/COR)
//        bit 7  - auto_restart (Read/Write)
//        bit 9  - interrupt (Read)
//        others - reserved
// 0x04 : Global Interrupt Enable Register
//        bit 0  - Global Interrupt Enable (Read/Write)
//        others - reserved
// 0x08 : IP Interrupt Enable Register (Read/Write)
//        bit 0 - enable ap_done interrupt (Read/Write)
//        bit 1 - enable ap_ready interrupt (Read/Write)
//        others - reserved
// 0x0c : IP Interrupt Status Register (Read/COR)
//        bit 0 - ap_done (Read/COR)
//        bit 1 - ap_ready (Read/COR)
//        others - reserved
// 0x10 : Data signal of out_buf_sts
//        bit 0  - out_buf_sts[0] (Read)
//        others - reserved
// 0x14 : Control signal of out_buf_sts
//        bit 0  - out_buf_sts_ap_vld (Read/COR)
//        others - reserved
// 0x20 : Data signal of sts_clear
//        bit 0  - sts_clear[0] (Read/Write)
//        others - reserved
// 0x24 : reserved
// 0x28 : Data signal of s2m_len
//        bit 31~0 - s2m_len[31:0] (Read/Write)
// 0x2c : reserved
// 0x30 : Data signal of pattern
//        bit 31~0 - pattern[31:0] (Read/Write)
// 0x34 : reserved
// 0x38 : Data signal of outbuf
//        bit 31~0 - outbuf[31:0] (Read/Write)
// 0x3c : Data signal of outbuf
//        bit 31~0 - outbuf[63:32] (Read/Write)
// 0x40 : reserved
// (SC = Self Clear, COR = Clear on Read, TOW = Toggle on Write, COH = Clear on Handshake)
```

LogicAnalyzer.v

```
case ( axi_addr_reg[11:2] )
    10'h000:
    begin
        la_enable <= axi_wdata_reg[23:0];
    end
    10'h001:
    begin
        h_thresh <= axi_wdata_reg[15:0];
    end
    10'h002:
    begin
        l_thresh <= axi_wdata_reg[15:0];
    end
    10'h003:
    begin
        pop_cond <= axi_wdata_reg[15:0];
    end
    default:
    begin
        axi_wdata_reg <= 32'b0;
    end
endcase
```

user_prj3.v

```
assign sm_tstrb      = 4'b0;
assign sm_tkeep      = 1'b0;
assign sm_tlast      = 1'b0;
assign low__pri_irq  = 1'b0;
assign High_pri_req  = 1'b0;
assign la_data_o     = 24'b0;


endmodule // USER_PRJ3
```

Algorithm[3]
24bits

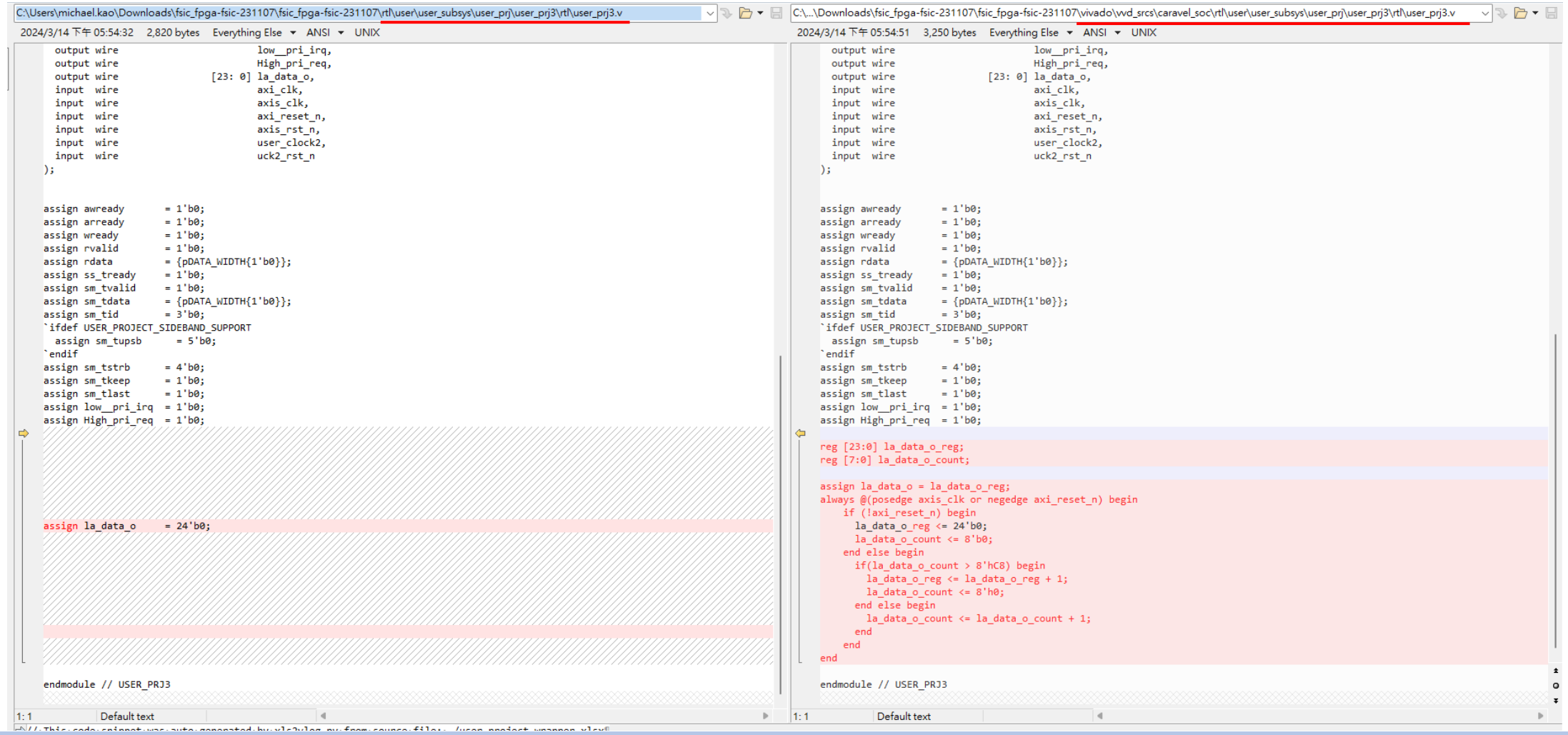
```
Packet format :
   |-----------------------------------|
   |RC[31:24]|        Data[23:0]       |
   |-----------------------------------|
   Packet size is 32bits = {RC[7:0],Data[23:0]}
   RC(Repeat count) : Start at 1 and ends at 255.
    **Note:**
       If RC exceed 255, the packet is released. A new count starts.
       Packet = 32'b0, indicate FIFO overflow because RC start at 1.
   Maximum transfer period
       8bit = 256bits count
       If clock is 100MHz
       10ns*256=2.5us/packet
```

ladma_captured.log

Compare la_data_output in the user_prj3

rtl\user\user_subsys\user_prj\user_prj3\rtl\user_prj3.v

vivado\vvd_srcs\caravel_soc\rtl\user\user_subsys\user_prj\user_prj3\rtl\user_prj3.v



C:\Users\michael.kao\Downloads\fsic_fpga-fsic-231107\fsic_fpga-fsic-231107\rtl\user\user_subsys\user_prj\user_prj3\rtl\user_prj3.v

2024/3/14 下午 05:54:32 2,820 bytes Everything Else ANSI UNIX

```
    output wire                     low__pri_irq,
    output wire                     High_pri_req,
    output wire               [23: 0] la_data_o,
    input  wire                     axi_clk,
    input  wire                     axis_clk,
    input  wire                     axi_reset_n,
    input  wire                     axis_rst_n,
    input  wire                     user_clock2,
    input  wire                     uck2_rst_n
);


assign awready       = 1'b0;
assign arready       = 1'b0;
assign wready        = 1'b0;
assign rvalid        = 1'b0;
assign rdata         = {pDATA_WIDTH{1'b0}};
assign ss_tready     = 1'b0;
assign sm_tvalid     = 1'b0;
assign sm_tdata      = {pDATA_WIDTH{1'b0}};
assign sm_tid        = 3'b0;
`ifdef USER_PROJECT_SIDEBAND_SUPPORT
  assign sm_tupsb      = 5'b0;
`endif
assign sm_tstrb      = 4'b0;
assign sm_tkeep      = 1'b0;
assign sm_tlast      = 1'b0;
assign low__pri_irq  = 1'b0;
assign High_pri_req  = 1'b0;

assign la_data_o     = 24'b0;

endmodule // USER_PRJ3
```

1: 1 Default text

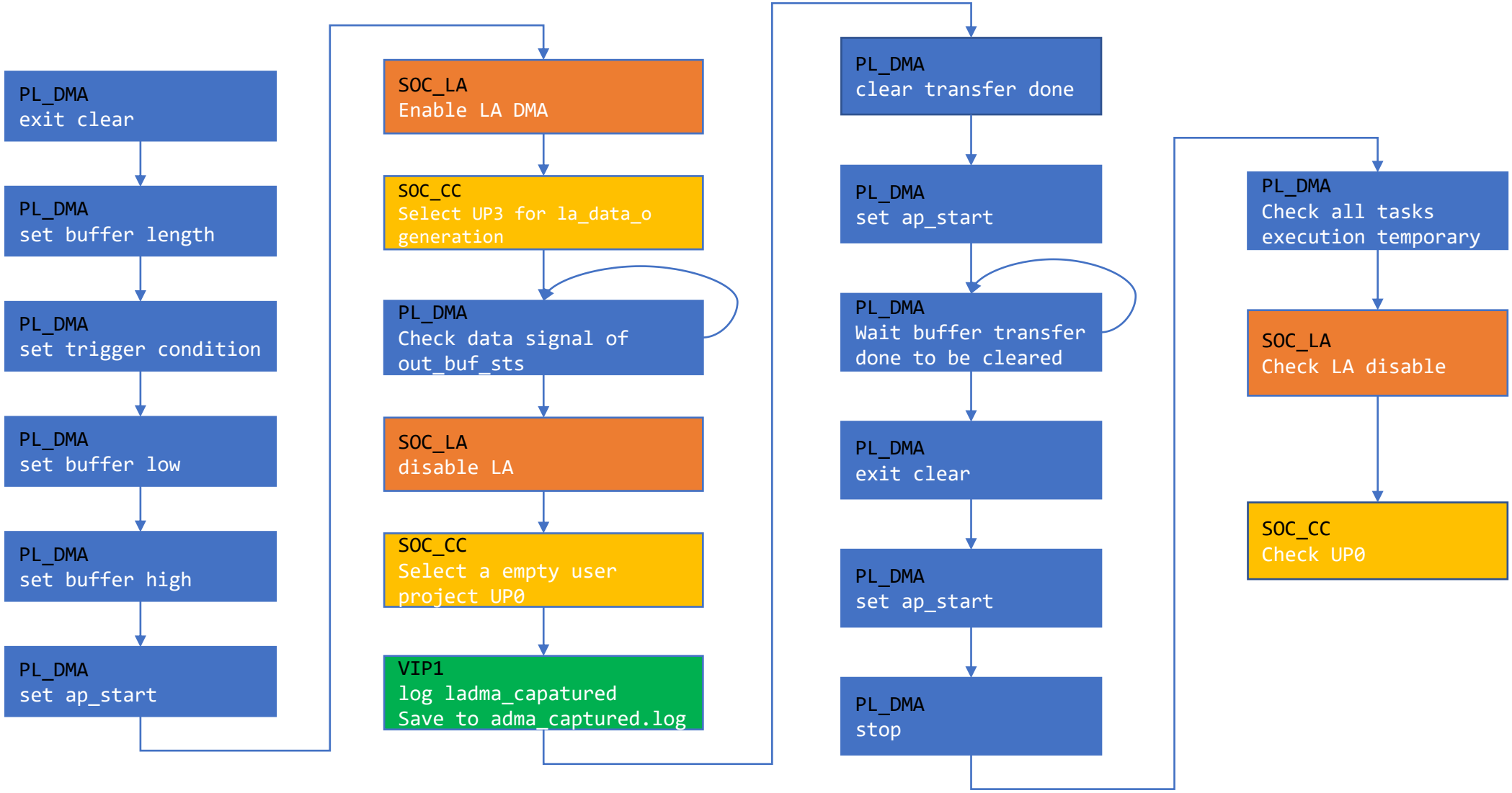
C:\...\Downloads\fsic_fpga-fsic-231107\fsic_fpga-fsic-231107\vivado\vvd_srcs\caravel_soc\rtl\user\user_subsys\user_prj\user_prj3\rtl\user_prj3.v

2024/3/14 下午 05:54:51 3,250 bytes Everything Else ANSI UNIX

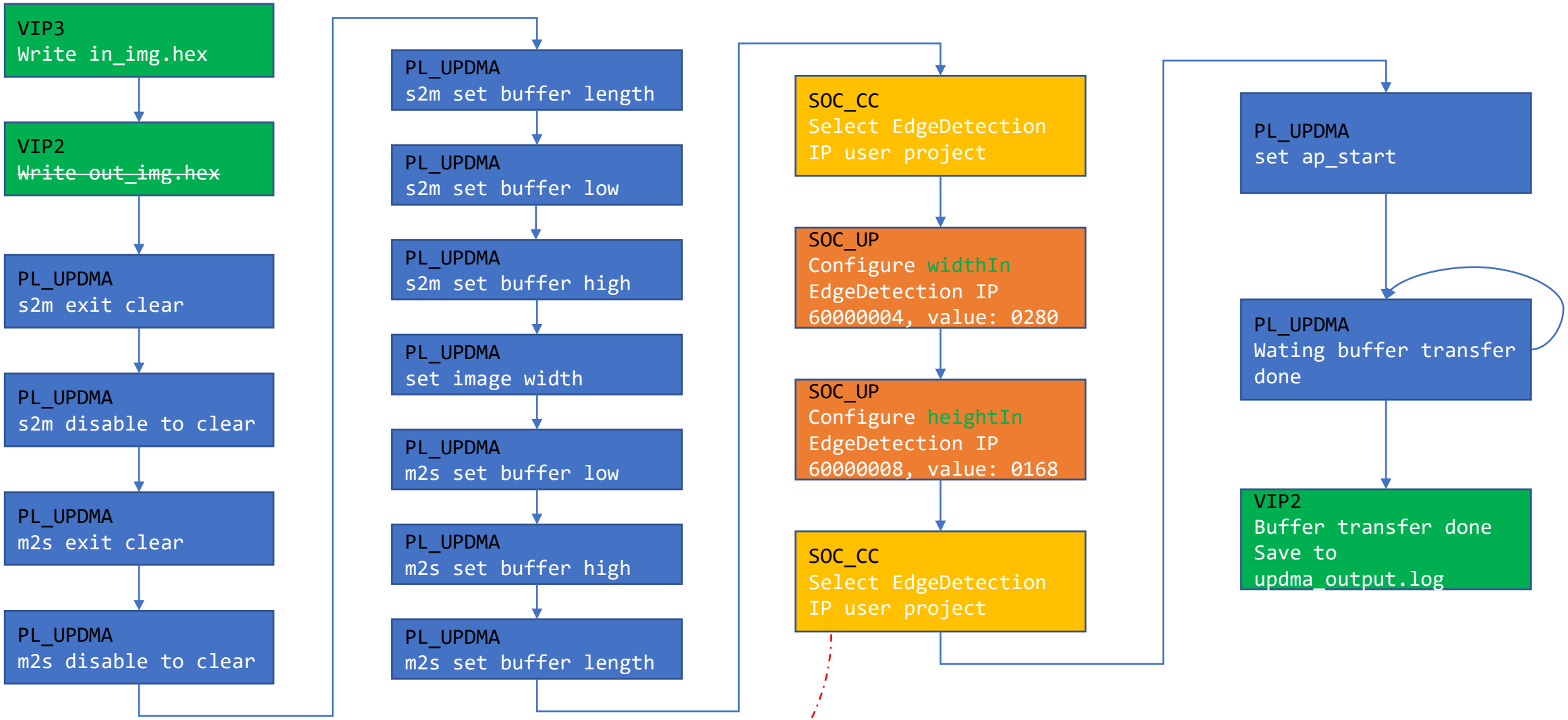
```
    output wire                     low__pri_irq,
    output wire                     High_pri_req,
    output wire               [23: 0] la_data_o,
    input  wire                     axi_clk,
    input  wire                     axis_clk,
    input  wire                     axi_reset_n,
    input  wire                     axis_rst_n,
    input  wire                     user_clock2,
    input  wire                     uck2_rst_n
);


assign awready       = 1'b0;
assign arready       = 1'b0;
assign wready        = 1'b0;
assign rvalid        = 1'b0;
assign rdata         = {pDATA_WIDTH{1'b0}};
assign ss_tready     = 1'b0;
assign sm_tvalid     = 1'b0;
assign sm_tdata      = {pDATA_WIDTH{1'b0}};
assign sm_tid        = 3'b0;
`ifdef USER_PROJECT_SIDEBAND_SUPPORT
  assign sm_tupsb      = 5'b0;
`endif
assign sm_tstrb      = 4'b0;
assign sm_tkeep      = 1'b0;
assign sm_tlast      = 1'b0;
assign low__pri_irq  = 1'b0;
assign High_pri_req  = 1'b0;

reg [23:0] la_data_o_reg;
reg [7:0] la_data_o_count;

assign la_data_o = la_data_o_reg;
always @(posedge axis_clk or negedge axi_reset_n) begin
    if (!axi_reset_n) begin
      la_data_o_reg <= 24'b0;
      la_data_o_count <= 8'b0;
    end else begin
      if(la_data_o_count > 8'hC8) begin
        la_data_o_reg <= la_data_o_reg + 1;
        la_data_o_count <= 8'h0;
      end else begin
        la_data_o_count <= la_data_o_count + 1;
      end
    end
end

endmodule // USER_PRJ3
```

1: 1 Default text

SocLa2DmaPath()



SocUp2DmaPath()

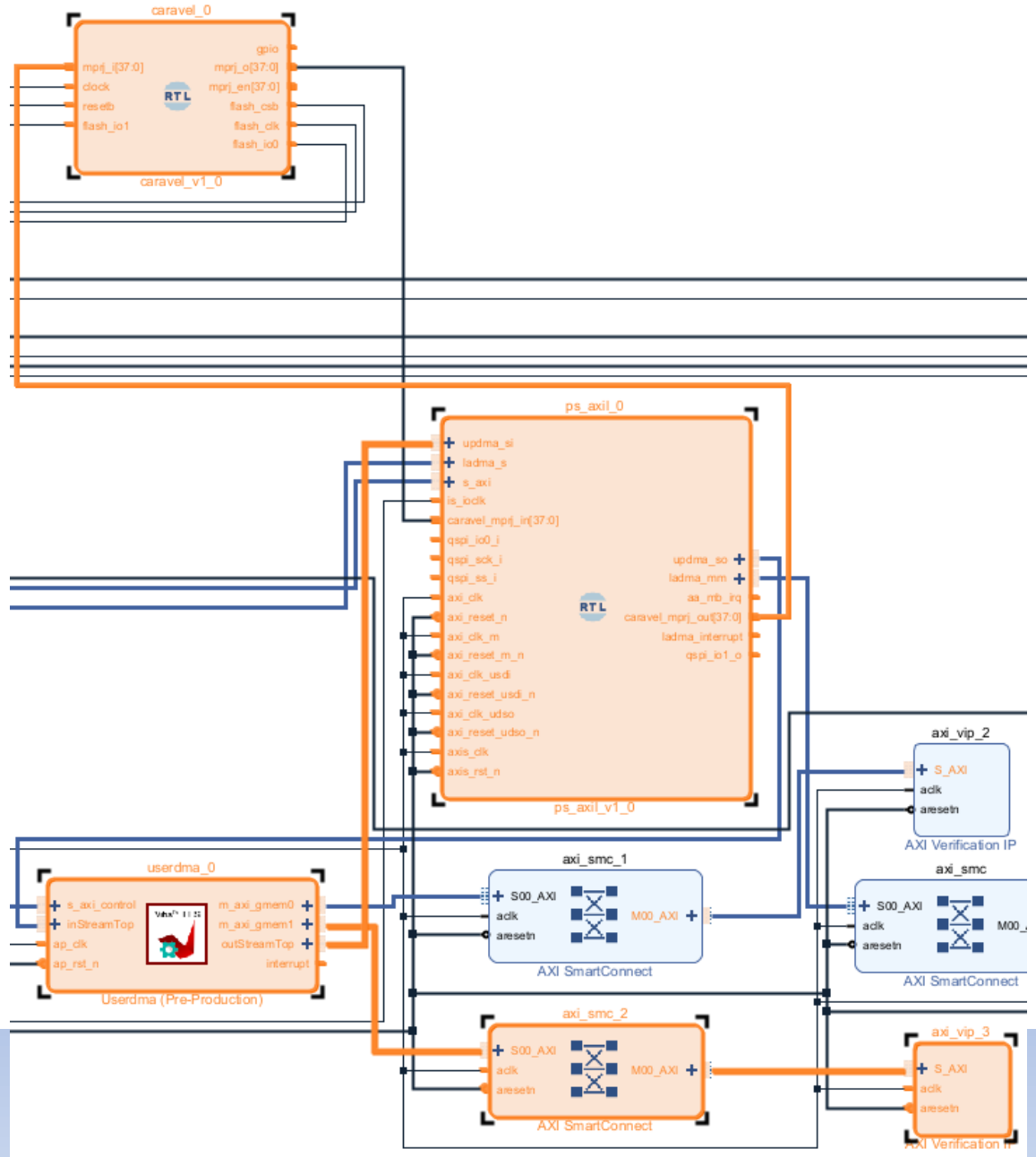


1.3.7.1.7 A userDMA data pattern to be sampled two times in SoC side AS->IS
- Found an issue to userDMA test results, a pattern seems to be sampled two times during AS->IS in case of IS deassert/assert its is_as_tready.

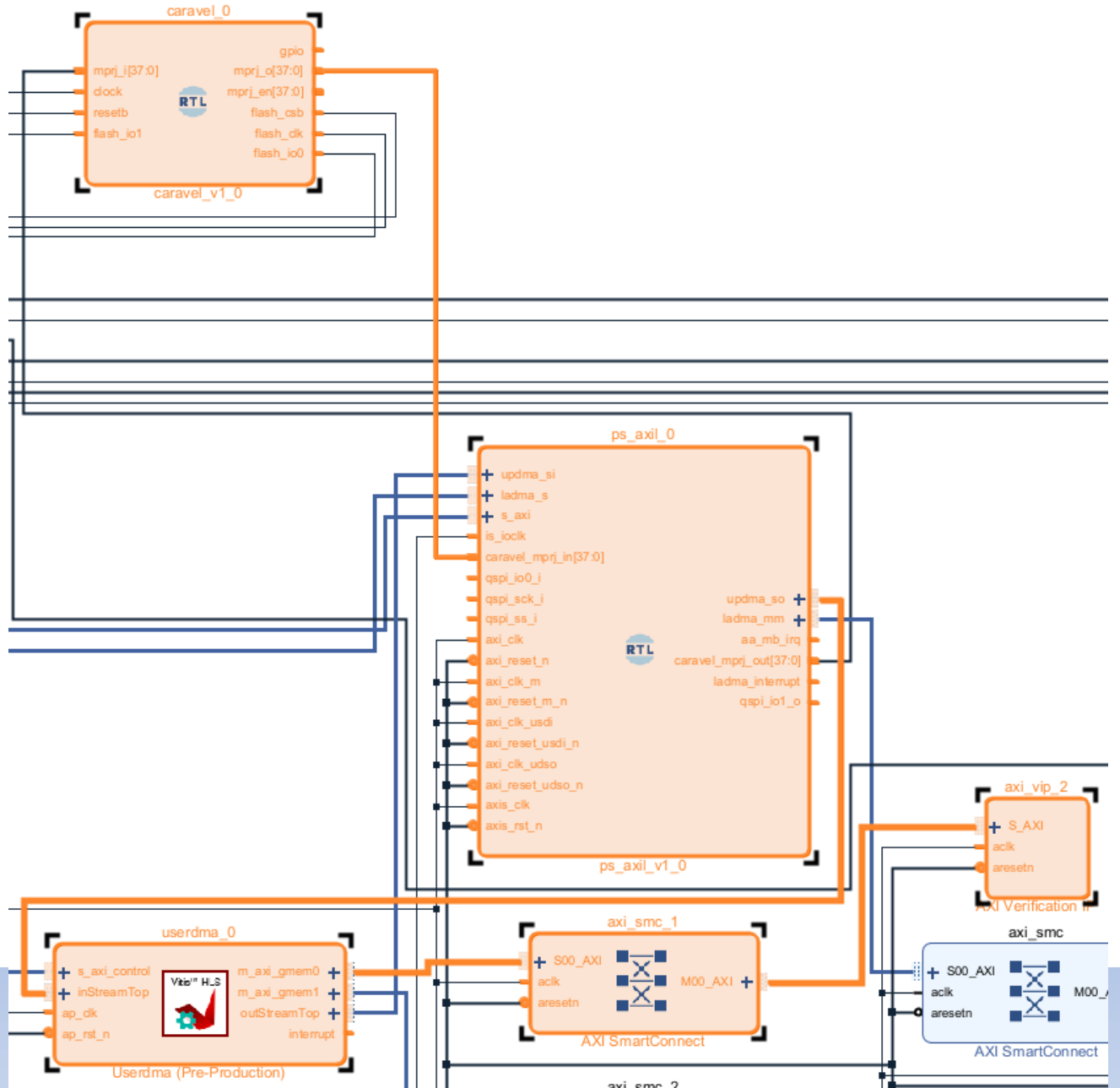
34 [01/10 Hurry] Closed. Will sort out the current issues and hand them over to Jiin

updma_input.log updma_output_gold.log updma_output.log

User DMA Traffic - Downstream



User DMA Traffic - Upstream



User DMA

```
void userdma(hls::stream<trans_pkt >& inStreamTop,
        bool *s2m_buf_sts,
        bool s2m_sts_clear,
        ap_uint<32> s2m_len,
        ap_uint<1> s2m_enb_clrsts,
        ap_uint<32> s2mbuf[BUF_LEN],
        bool *s2m_err,
        ap_uint<32> Img_width,
        ap_uint<32> m2sbuf[BUF_LEN],
        bool *m2s_buf_sts,
        bool m2s_sts_clear,
        int m2s_len,
        ap_uint<1> m2s_enb_clrsts,
        hls::stream<trans_pkt >& outStreamTop) {
#pragma HLS INTERFACE axis register_mode=both register port=inStreamTop
#pragma HLS INTERFACE m_axi max_write_burst_length=64 latency=10 depth=1024 bundle=gmem0 port=s2mbuf offset = slave
#pragma HLS INTERFACE s_axilite port = s2m_buf_sts bundle = control
#pragma HLS INTERFACE s_axilite port = s2m_sts_clear bundle = control
#pragma HLS INTERFACE s_axilite port = s2m_len bundle = control
#pragma HLS INTERFACE s_axilite port = s2m_enb_clrsts bundle = control
#pragma HLS INTERFACE s_axilite port = s2mbuf bundle = control
#pragma HLS INTERFACE s_axilite port = s2m_err bundle = control
#pragma HLS INTERFACE s_axilite port = Img_width bundle = control
#pragma HLS INTERFACE axis register_mode=both register port=outStreamTop
#pragma HLS INTERFACE m_axi max_write_burst_length=64 latency=10 depth=1024 bundle=gmem1 port=m2sbuf offset = slave
#pragma HLS INTERFACE s_axilite port = m2s_buf_sts bundle = control
#pragma HLS INTERFACE s_axilite port = m2s_sts_clear bundle = control
#pragma HLS INTERFACE s_axilite port = m2s_len bundle = control
#pragma HLS INTERFACE s_axilite port = m2s_enb_clrsts bundle = control
#pragma HLS INTERFACE s_axilite port = m2sbuf bundle = control
#pragma HLS INTERFACE s_axilite port = return bundle = control

#pragma HLS DATAFLOW

  hls::stream<data,DATA_DEPTH > inbuf;
  hls::stream<int,COUNT_DEPTH> incount;
  hls::stream<out_data,DATA_DEPTH > outbuf;
  hls::stream<int,COUNT_DEPTH> outcount;

  getinstream(inStreamTop, s2m_enb_clrsts, s2m_len, *s2m_err, Img_width, inbuf, incount);
  streamtoparallelwithburst(inbuf, incount, s2m_enb_clrsts, *s2m_buf_sts, s2m_sts_clear, s2m_len, s2mbuf);
  paralleltostreamwithburst(m2sbuf, m2s_enb_clrsts, Img_width, m2s_len, outbuf, outcount);
  sendoutstream(outbuf, outcount, m2s_enb_clrsts, *m2s_buf_sts, m2s_sts_clear, outStreamTop);

}
```

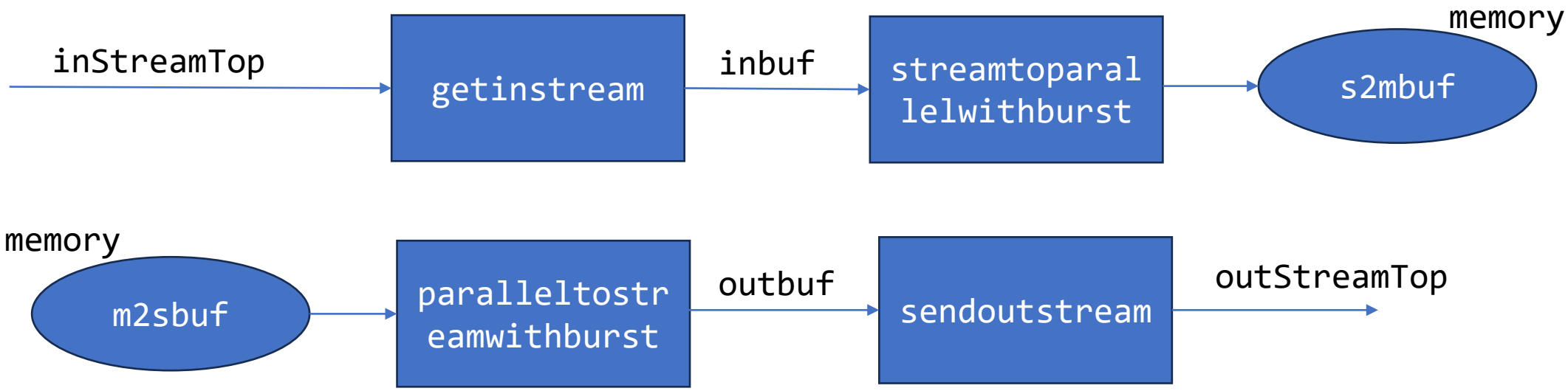
Register Name	Offset Address	Description
Control signals	0x00	Control signals Definition bit 0 - ap_start (Read/Write/COH) Set 1 to start Axis DMA function bit 1 - ap_done (Read/COR) bit 2 - ap_idle (Read) bit 3 - ap_ready (Read/COR)
s2m Buffer transfer done status register	0x10	bit 0 – s2m buffer transfer done status (Read) Set this register to 1 if stream data has written to memory and data length is equal to Buffer Length(640*480/4 DW)
s2m Buffer transfer done status clear register	0x20	bit 0 – clear s2m buffer transfer done status (Read/Write) Set 1 to clear s2m buffer done status/s2m error status and reset internal state, then set 0 if finish to clear buffer done status Note: Before set this register to 1 to clear s2m buffer transfer done status/s2m error status, Clear status control register must set to 1. After buffer transfer done status is clear, this register needs to be cleared for next operation.
s2m Buffer Length	0x28	Set s2m buffer length, must set to 640*480/4.
s2m Clear Status Control register	0x30	bit 0 –Enable to clear s2m buffer transfer done status (Read/Write)
s2m Buffer Lower base address register	0x38	bit 31~0 – The memory base address [31:0] of s2m buffer (Read/Write)
s2m Buffer Upper base address register	0x3C	bit 31~0 – The memory base address [63:32] of s2m buffer (Read/Write)
s2m err status register	0x44	bit 0 – s2m err status when sof(start of frame) signal or eol(end of line) signal of side band of user project is incorrect (Read)
Image width	0x54	bit 31~0 – Image_width[31:0] (Read/Write) Note: The value of this register is DW unit, so the value should be real width divided by 4. This value must be 160, due to Image is 640(width)*480(height)
m2s Buffer Lower base address register	0x5C	bit 31~0 – The memory base address [31:0] of m2s buffer (Read/Write)
m2s Buffer Upper base address register	0x60	bit 31~0 – The memory base address [63:32] of m2s buffer (Read/Write)
m2s Buffer transfer done status register	0x68	bit 0 – m2s buffer transfer done status (Read) Set this register to 1 if data has fetched from memory and data length is equal to Buffer Length(640*480/4 DW)
m2s Buffer transfer done status clear register	0x78	bit 0 – clear m2s buffer transfer done status (Read/Write) Set 1 to clear m2s buffer done status/s2m error status and reset internal state, then set 0 if finish to clear buffer done status Note: Before set this register to 1 to clear m2s buffer transfer done status/ m2s error status, Clear status control register must set to 1. After buffer transfer done status is clear, this register needs to be cleared for next operation.
m2s Buffer Length	0x80	Set m2s buffer length, must set to 640*480/4.
m2s Clear Status Control register	0x88	bit 0 –Enable to clear m2s buffer transfer done status (Read/Write)

Document and Source: https://github.com/bol-edu/fsic_fpga/tree/main/vivado/vitis_prj/hls_userdma

User DMA – Data flow

```
// Expects max bandwidth at 64 beats burst (for 64-bit data)
static constexpr int MAX_BURST_LENGTH = 16;
static constexpr int BUFFER_FACTOR = 64;

// Buffer sizes
static constexpr int DATA_DEPTH = MAX_BURST_LENGTH * BUFFER_FACTOR;
static constexpr int COUNT_DEPTH = BUFFER_FACTOR;
```

```
#pragma HLS DATAFLOW

  hls::stream<data,DATA_DEPTH > inbuf;
  hls::stream<int,COUNT_DEPTH> incount;
  hls::stream<out_data,DATA_DEPTH > outbuf;
  hls::stream<int,COUNT_DEPTH> outcount;

  getinstream(inStreamTop, s2m_enb_clrsts, s2m_len, *s2m_err, Img_width, inbuf, incount);
  streamtoparallelwithburst(inbuf, incount, s2m_enb_clrsts, *s2m_buf_sts, s2m_sts_clear, s2m_len, s2mbuf);
  parralleltostreamwithburst(m2sbuf, m2s_enb_clrsts, Img_width, m2s_len, outbuf, outcount);
  sendoutstream(outbuf, outcount, m2s_enb_clrsts, *m2s_buf_sts, m2s_sts_clear, outStreamTop);
```



User DMA - getinstream

```
void getinstream(hls::stream<trans_pkt >& in_stream, ap_uint<1> in_en_clrsts, ap_uint<32> in_s2m_len, bool &s2m_err, ap_uint<32> in_Img_width,
            hls::stream<data > &out_stream, hls::stream<int>& out_counts)
{
    int count = 0;
    static ap_uint<32> in_len = 0;
    trans_pkt in_val;
    static int width_count = 0;
    if(!in_en_clrsts){
        do {
          #pragma HLS PIPELINE
            in_val = in_stream.read();
            data out_val = {in_val.data, in_val.last};
            out_stream.write(out_val);

            s2m_err=0;

            if((in_len==0)&&(in_val.user(2,2)!=1))
              s2m_err=1;
            if((in_len!=0)&&(in_val.user(2,2)==1))
              s2m_err=1;

            if((width_count==in_Img_width-1)&&(in_val.user(3,3)!=1))
              s2m_err=1;

            if(width_count==in_Img_width-1)
              width_count = 0;
            else
              width_count++;

            count++;
            in_len++;
            if (count >= MAX_BURST_LENGTH) {
                out_counts.write(count);
                count = 0;
            }
        } while(in_len < in_s2m_len);
    } else {
        in_len=0;
        s2m_err=0;
    }
}
```

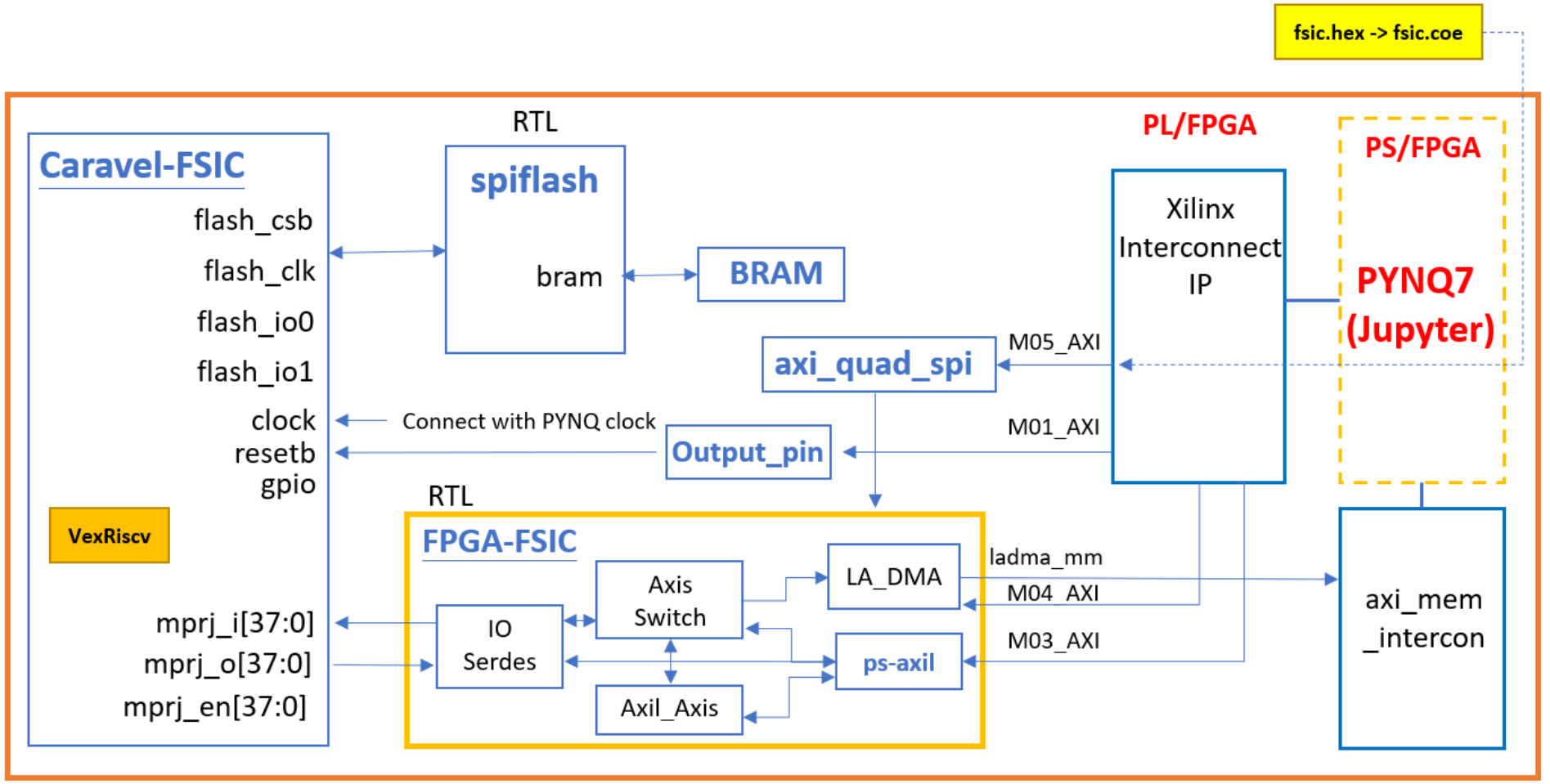
User DMA - streamtoparallelwithburst

```
void streamtoparallelwithburst(hls::stream<data> &in_stream, hls::stream<int> &in_counts, ap_uint<1> in_en_clrsts,
    bool &buf_sts, bool sts_clear, ap_uint<32> in_s2m_len, ap_uint<32> *out_memory) {
  data  in_val;

  int count;
  static bool out_sts=0;
  static ap_uint<32> final_s2m_len=0;

  if(in_en_clrsts) {
      if(sts_clear == 1) {
          out_sts = 0;
          out_memory -= final_s2m_len;
          final_s2m_len = 0;
      }
      buf_sts = out_sts;
  } else {
      do {
          count = in_counts.read();

          for (int i = 0; i < count; ++i) {
        #pragma HLS PIPELINE
              in_val = in_stream.read();
              out_memory[i] = in_val.data_filed;
          }
          out_memory += count;
          final_s2m_len += count;

          if(final_s2m_len == in_s2m_len){
              out_sts = 1;
          }

          buf_sts = out_sts;
      } while(final_s2m_len < BUF_LEN);
  }
}
```

Bridge of Life Education

Lab 4-2 Caravel-FSIC FPGA Validation

Lab4-2 Introduction



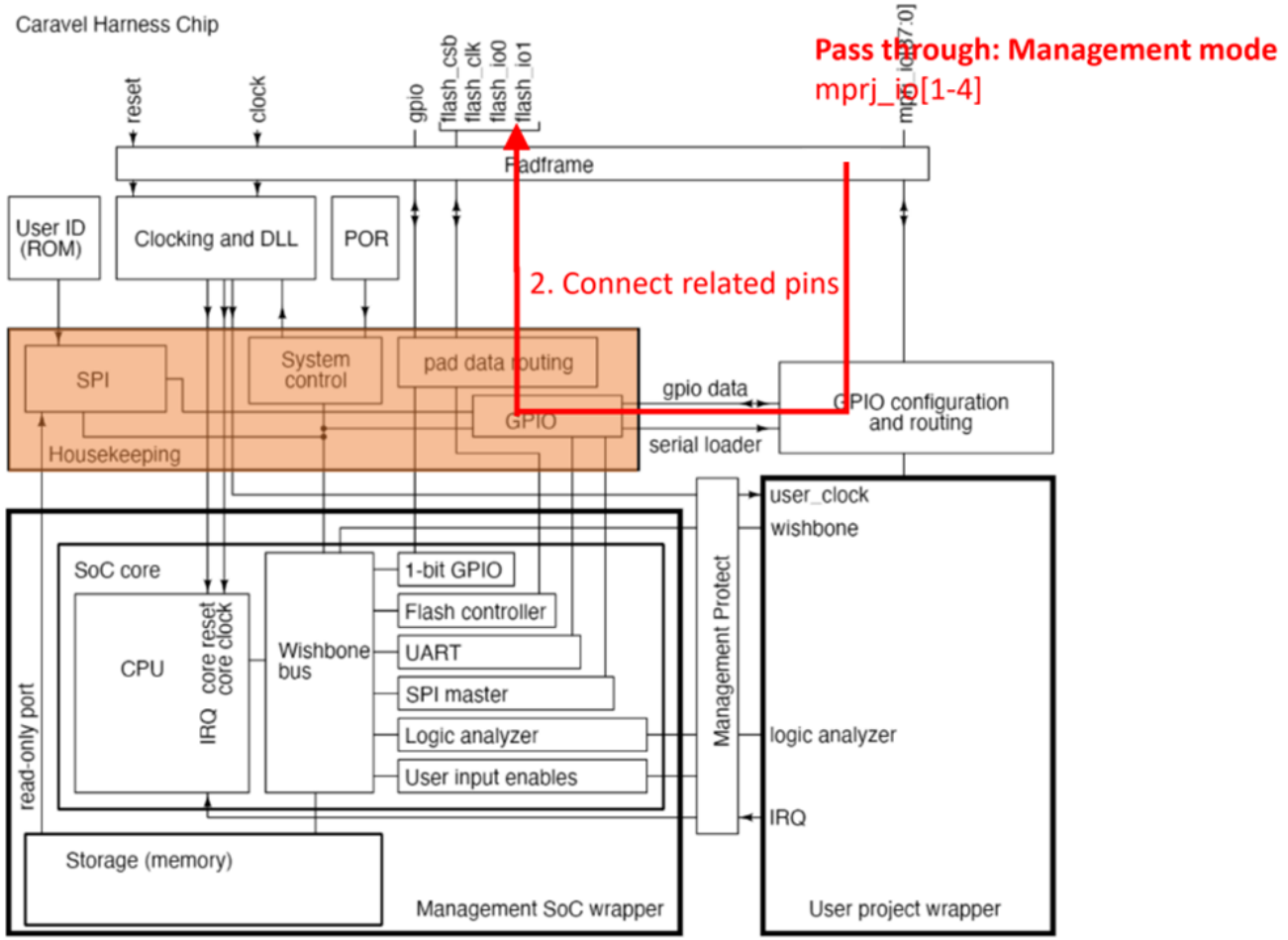
AXI Quad SPI

- Quad SPI is an enhancement to the standard SPI protocol and provides a simple method for data exchange between a master and a slave

- Reference: https://docs.amd.com/r/en-US/pg153-axi-quad-spi/AXI-Quad-SPI-v3.2-LogiCORE-IP-Product-Guide

Load Firmware Code

- Jupyter Notebook
  - PS side
  - Load fw code
  - Transfer for
- Caravel's pass-through mode
  - PL side
  - open the path of the external spi controller to the caravel soc spiflash interface by caravel's mprj interface.



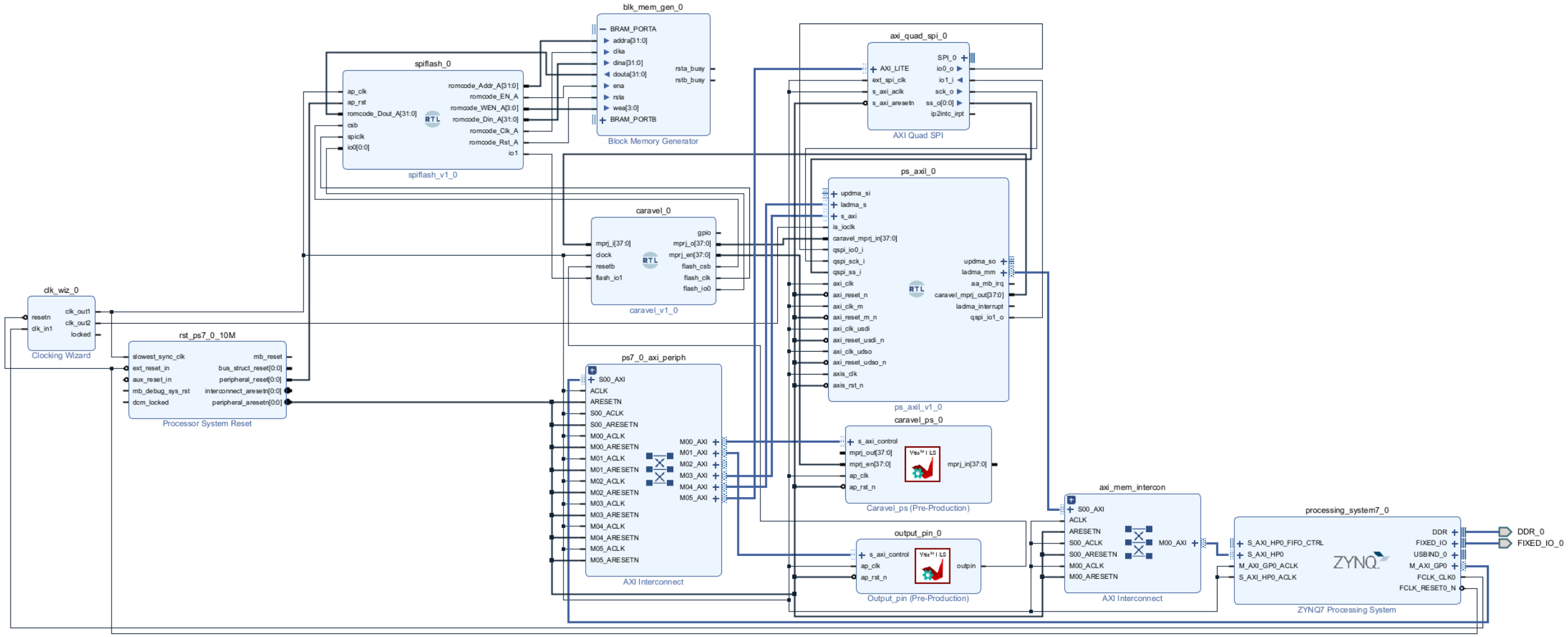
fsic_fpga\vivado\vvd_srcs\fpga\rtl\PL\ps_axil.v

```
325     assign caravel_mprj_out = {1'bz, is_ioclk, 1'bz, 13'bz, is_serial_tclk, is_serial_txd, 3'bz, qspi_sck_i, qspi_ss_i, qspi_io0_i, 2'bz};
```

Caravel-FSIC FPGA Validation

- Building Environment
  - Ubuntu 20.04 with risc-v cross compiler
- Bitfile building steps
  - git clone https://github.com/bol-edu/fsic_fpga
  - cd fsic_fpga
  - cd vivado
  - ./run_vivado_fsic
    - Bitfile: /jupyter_notebook/caravel_fpga.bit
    - Hwh: /jupyter_notebook/caravel_fpga.hwh
- JupyterNotebook Code
  - /jupyter_notebook/caravel_fpga_fsic.ipynb

Caravel-FSIC FPGA Validation Block Design



Block Design

- AXI_QUAD_SPI
  - It will enable pass through mode for PS side to write firmware code pass through FPGA-FSIC and caravel to the spiflash.

- Reset_Control(output_pin)
  - It release the reset signal and the caravel will read the preload firmware code from spiflash.

- Caravel_ps (not used)
  - Read the mprj_i、mprj_o、mprj_en
  - Can be remove

LA_DMA Config from PS Side

Caravel_fpga_fsic.ipynb

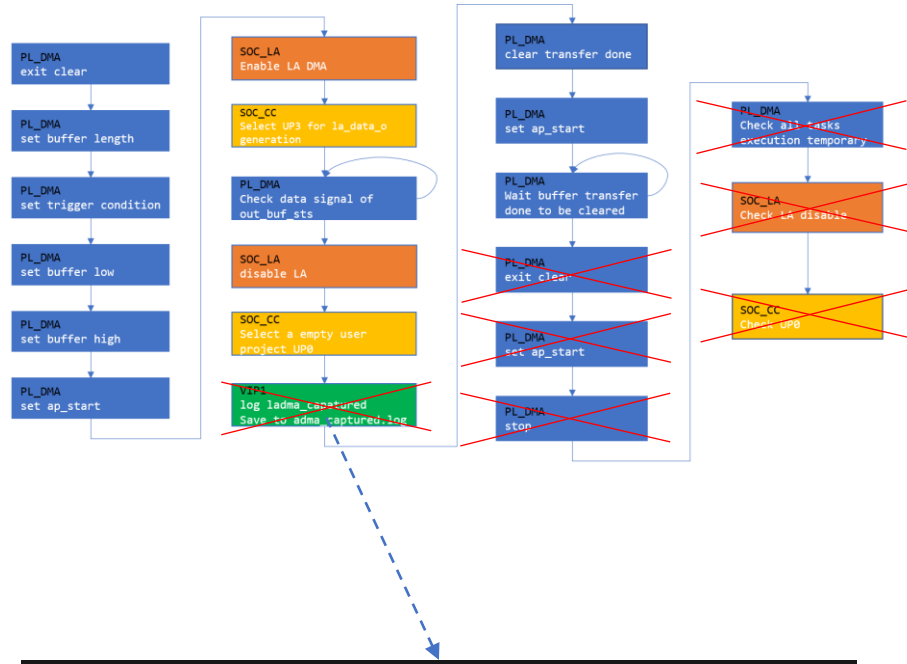
```
# Allocation memory
ladma_buf = allocate(shape=(1024,), dtype=np.uint32)
print("ladma_buf.device_address: ", hex(ladma_buf.device_address))

IP_BASE_ADDRESS = ladma_buf.device_address
ADDRESS_RANGE = 0x1000
buf_mmio = MMIO(IP_BASE_ADDRESS, ADDRESS_RANGE)
# 0x00 : Control signals
#        bit 0 - ap_start (R/W/COH)
#        bit 1 - ap_done (R/COR)
#        bit 2 - ap_idle (R)
#        bit 3 - ap_ready (R/COR)
# 0x10 : Buffer transfer done status register
#        bit 0 - buffer transfer done status (R)
# 0x20 : Buffer transfer done status clear register
#        bit 0 - clear buffer transfer done status (R/W)
# 0x28 : Buffer Length
#        bit 31~0 - set buffer length (must 1024)
# 0x30 : Triggered condition
#        bit 23~0 - set triggered condidtion (R/W)
#        others  - reserved
# 0x34 : Buffer Lower base address
#        bit 31~0 - (R/W)
# 0x38 : Buffer High base address
#        bit 31~0 - (R/W)
# ladma Configuration
ADDRESS_OFFSET = PL_DMA # 0x8000
# exit clear operation
mmio.write(ADDRESS_OFFSET + 0x20, 0x00000000)
# set buffer length
mmio.write(ADDRESS_OFFSET + 0x28, 0x00000400)
# set trigger condition
mmio.write(ADDRESS_OFFSET + 0x30, 0x00000000)
# set buffer low
mmio.write(ADDRESS_OFFSET + 0x38, ladma_buf.device_address)
# set buffer high
mmio.write(ADDRESS_OFFSET + 0x3C, 0x00000000)
```

```
# ladma Configuration
ADDRESS_OFFSET = PL_DMA # 0x8000
# set ap_start
mmio.write(ADDRESS_OFFSET + 0x00, 0x00000001)
```

```
# enable la 0xFFFFFF
ADDRESS_OFFSET = SOC_LA # 0x1000
mmio.write(ADDRESS_OFFSET, 0x00FFFFFF)
#print("mmio.read(ADDRESS_OFFSET): ", hex(mmio.read(ADDRESS_OFFSET)))
# select target UP
ADDRESS_OFFSET = SOC_CC # 0x5000
mmio.write(ADDRESS_OFFSET, 0x00000003)
#print("mmio.read(ADDRESS_OFFSET): ", hex(mmio.read(ADDRESS_OFFSET)))
```

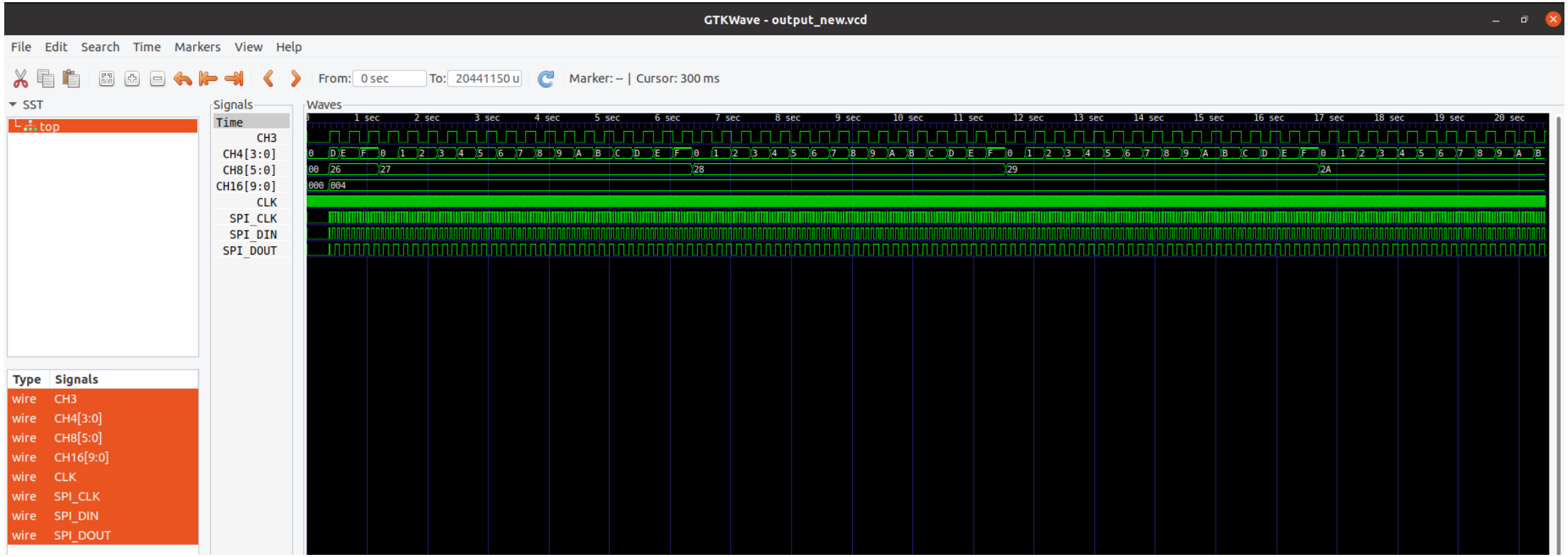
```
# ladma Configuration
ADDRESS_OFFSET = PL_DMA # 0x8000
while True:
    if mmio.read(ADDRESS_OFFSET+0x10) == 0x01:
        break
print("mmio.read(ADDRESS_OFFSET+0x10): ", hex(mmio.read(ADDRESS_OFFSET+0x10)))

# disable la 0x000000
ADDRESS_OFFSET = SOC_LA # 0x1000
mmio.write(ADDRESS_OFFSET, 0x00000000)
# select fake UP
ADDRESS_OFFSET = SOC_CC # 0x5000
mmio.write(ADDRESS_OFFSET, 0x00000000)

ADDRESS_OFFSET = PL_DMA # 0x8000
# clear buffer transfer done operation
mmio.write(ADDRESS_OFFSET + 0x20, 0x00000001)
# set ap_start
mmio.write(ADDRESS_OFFSET + 0x00, 0x00000001)
while True:
    if mmio.read(ADDRESS_OFFSET+0x10) != 0x01:
        break
print("mmio.read(ADDRESS_OFFSET+0x10): ", hex(mmio.read(ADDRESS_OFFSET+0x10)))
```

SocLa2DmaPath()




```
#dump la log to file
file = open("simulate.log", "w")
for i in range(0,0xFFF,4):
    file.write('{:08x}'.format(buf_mmio.read(i))+"\n")
file.close()
```

Gtkwave Dump Waveform of LA Log



Lab 4-3 Requirement

Lab (**fsic-fpga**) Lab Work

Implementation

1. Refer to lab-fsim-sim, Integrate FIR into PRJ1 (axilite, axi-stream in/out)
2. Refer to hls_userdma, design a dma for FIR
   - https://github.com/JoshSu0/fsic_fpga/tree/fsic-231107/vivado/vitis_prj/hls_userdma
3. Build FPGA
4. Firmware code
5. Jupyter-notebook Python code

Firmware Code Explained

- Building Environment
  - Ubuntu 20.04 with risc-v cross compiler
- Firmware Building Steps
  - git clone https://github.com/bol-edu/fsic_fpga
  - cd fsic_fpga
  - cd testbench/fsic
  - ./run_fw
    - fsic.hex
      - Firmware file used for JupyterNotebook firmware loading
    - fsic.coe
      - Firmware file used for simulation

Submission

- Please submit the following files to {NTHU eeclass / NTU COOL / NYCU E3} before 2024/4/25:
  - **1.report_StudentID.pdf**
  - **2.Github_link.txt**
- Please check the next pages for more detailed information.

Bridge of Life Education

Appendix

Project Reminding

- Simulation
  - Design
    - No SPI flash mechanism support
  - Issue
    - LADMA ap_start control register can't be set to 1'b0 after the enabled
    - LADMA buffer transfer done can't be cleared if executing the task SocLa2Dma() before task FpgaLocal_CfgRead()
    - Some userDMA data pattern to be sampled two times in SoC side AS->IS
- Board
  - Design
    - caravel_ps module removing
    - userDMA integration
      - EdgeDetection tupsb[4:0] is mapped to userDMA tuser[6:2]
  - Issue
    - Jyupter notebook will recive 0x00 or incorrect read data after jyputer notebook issues read cycle to the modules on caraval soc