清大實作專題
系統晶片實驗
SOC Lab

Jiin Lai

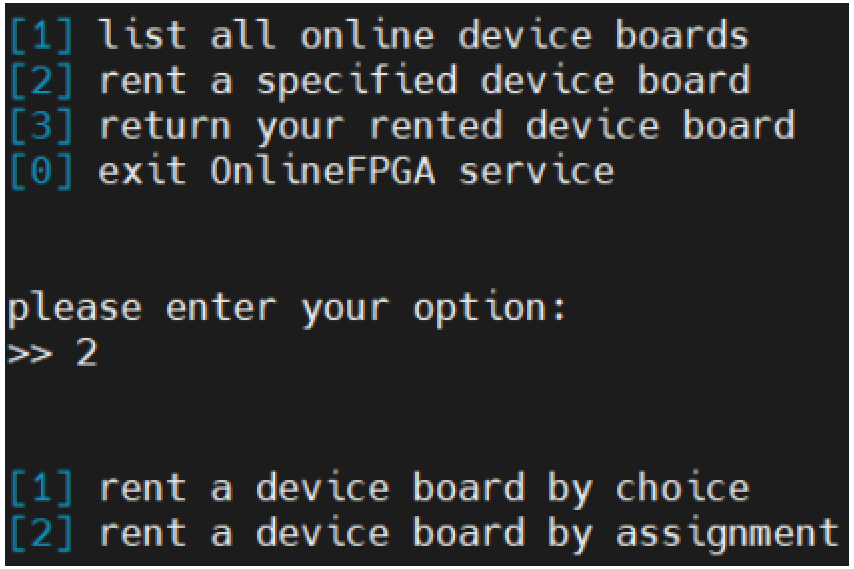
jiinlai@gmail.com

Agenda

- Sign-up and Self Introduction (1-2 minutes)
- Lab introduction
- Collaborative Learning Model
- Summer Study Program
- Elect 2 representatives – no TA, collaborative learning environment in introduce
- Q & A

Lab Facility

- 台大慶齡中心 320A 室
- 線上FPGA 管理系統實驗室，自動工作排程
- 實驗設配
  - 6 x Servers
  - 5 x U50
  - 20 x PYNQ-Z2
  - 10 x KV260 – 2X capacity of PYNQ-Z2
  - VCK 5000

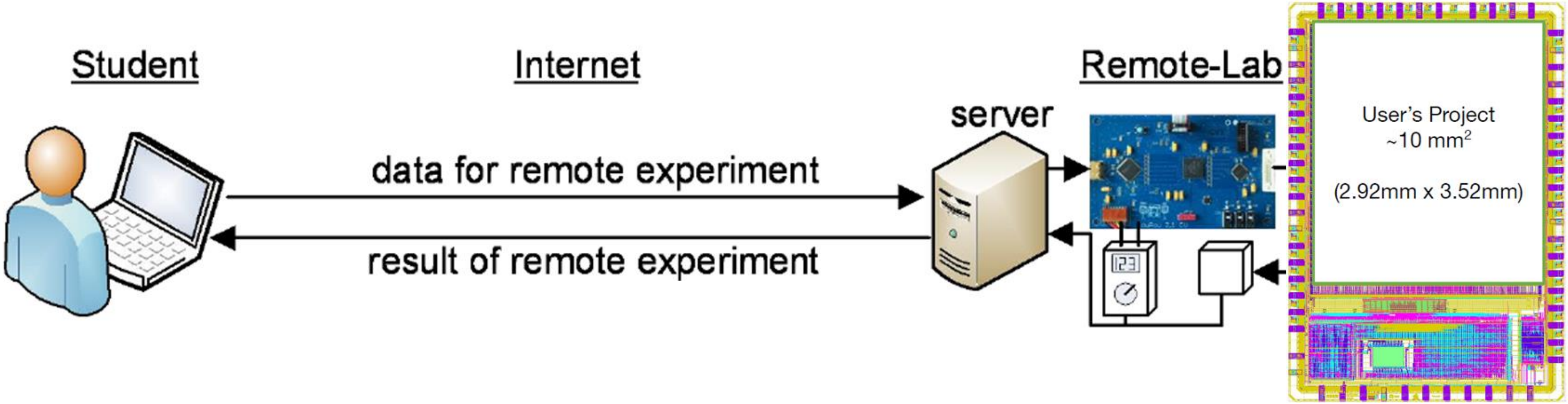


```
[1] list all online device boards
[2] rent a specified device board
[3] return your rented device board
[0] exit OnlineFPGA service


please enter your option:
>> 2


[1] rent a device board by choice
[2] rent a device board by assignment
```

Online IC Validation System



Course Syllabus

Course Objectives

Educate a SOC IC designer with the technical skills in

IC design

FPGA design

Embedded Programming

SoC Lab – Fall Semester

Objective:

- Learn Verilog and HLS Design Implementation on FPGA and ASIC
- Implement an IP and Integrate it into a SOC design
- Realize the SOC design and emulate it in FPGA

Design

1. Introduction to HLS and tools
2. VLSI Circuit Basics
3. Logic Design
4. Structure Design with HLS and Verilog
5. HLS pipeline, dataflow, datatype
6. Verification
7. Introduction to FPGA and its architecture
8. SOC module level specification & Design
9. Embedded Programming – 1st part

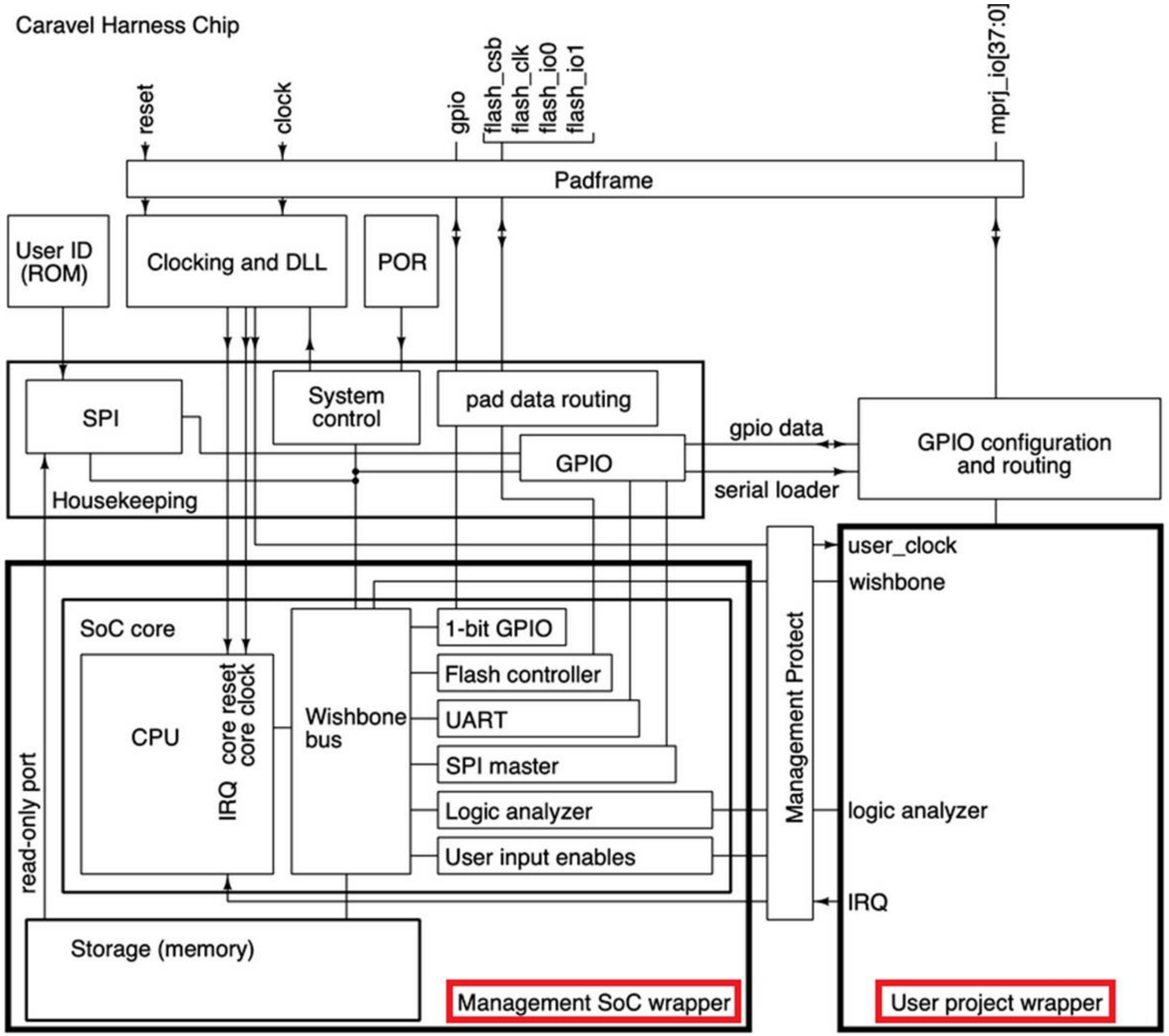
Design Flow

1.Tools – Tcl, Perl, Makefile
2.Simulator - Verilog, Coverage, assertion
3.Synthesis
4.Timing analysis
5.Verification Methodology

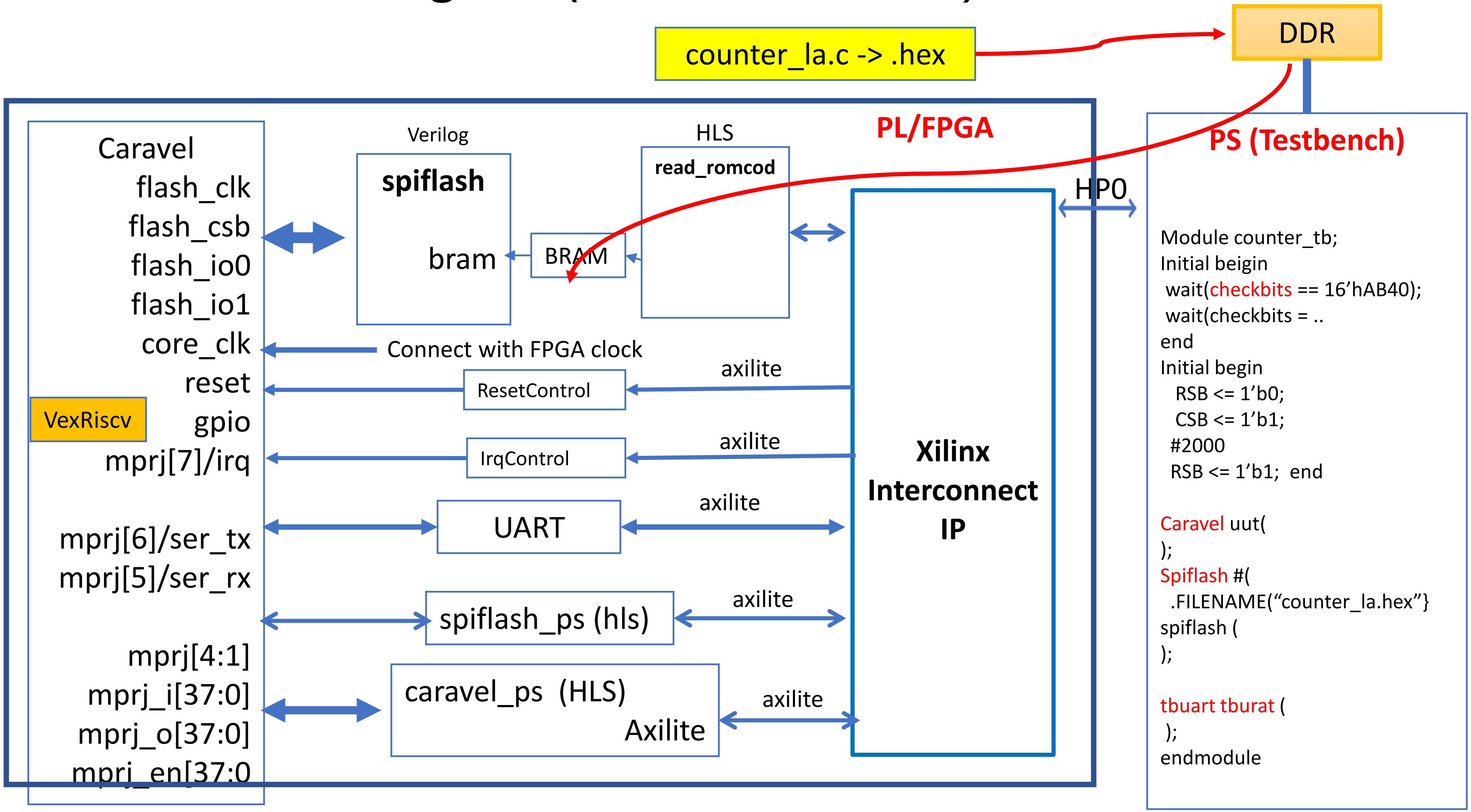
Laboratory

1.Tool Installation
2.Lab#1 - FIR Filter (AXI master, AXI Stream)
3.Caravel SOC Simulation Environment
4.HLS - Design Analysis & Optimization
5.HLS/Verilog Lab
6.Caravel SOC FPGA Implementation
7.User IP Integration with Caravel SOC
8.Final Project

Caravel SOC Block Diagram



Caravel FPGA Block Diagram (Based PYNQ-Z2)



```
Module counter_tb;
Initial beigin
 wait(checkbits == 16’hAB40);
 wait(checkbits = ..
end
Initial begin
   RSB <= 1’b0;
   CSB <= 1’b1;
  #2000
  RSB <= 1’b1;  end

Caravel uut(
);
Spiflash #(
 .FILENAME(“counter_la.hex”}
spiflash (
);

tbuart tburat (
 );
endmodule
```

Windbond spiflash model: https://www.winbond.com/hq/support/documentation/?__locale=en&line=/product/code-storage-flash-memory/index.html&family=/product/code-storage-flash-memory/serial-nor-flash/index.html&category=/.categories/resources/verilog-model/

Advanced SoC Lab - Spring Semester

Objective:

1.Learn Advanced topics in IC Design, SOC chip-level design

2.Develop an Application Accelerator

3.Complete IC design flow, and be ready for tape out. （June）

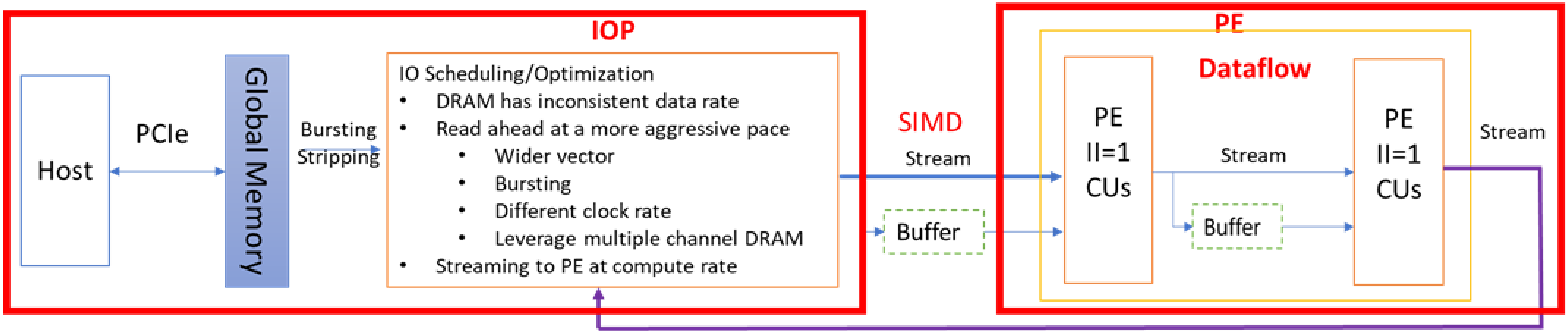
4.A contest to compete for tape-out opportunity

Design
1.Selected topics on high-performance Design
2.Advanced HLS Topics
3.Advanced Static Timing Analysis
4.SOC Chip level Design Issues
5.Advanced Embedding Programming
6.Advanced Design issues

Design Flow
1.Floorplan
2.Placement
3.Clock tree
4.Routing
5.DFT and Testing
6.LVD, DR ， ERC
7.Post-layout Timing Analysis

Laboratory
1.Algorithmic HLS Lab
2.FSIC Module Design & Verification
3.User Project Development
4.FISC Integration and Simulation
5.FSIC Physical Implementation and Tapeout (in June)

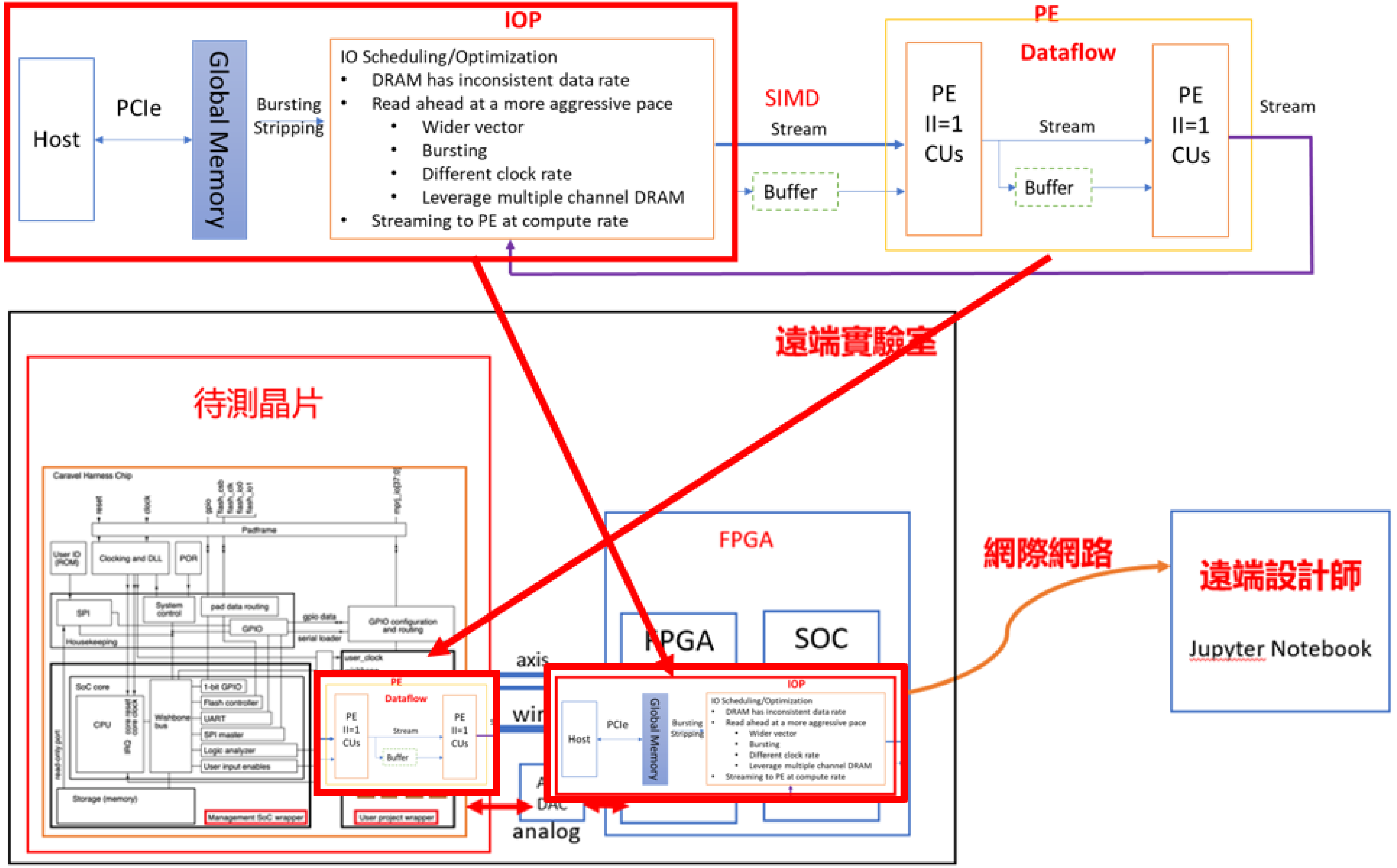
A Proposed Architecture for Accelerator Design



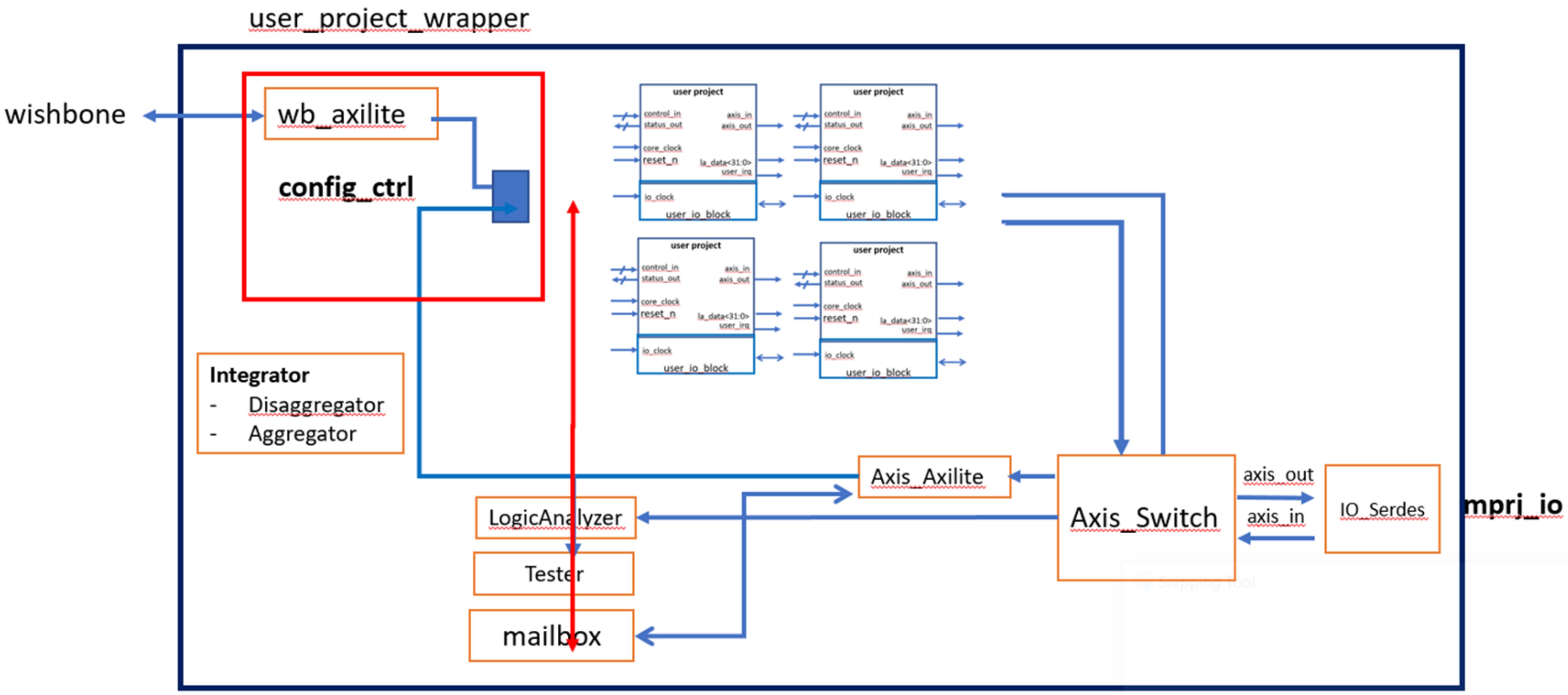
IO Scheduling/Optimization

- DRAM has inconsistent data rate
- Read ahead at a more aggressive pace
  - Wider vector
  - Bursting
  - Different clock rate
  - Leverage multiple channel DRAM
- Streaming to PE at compute rate

Spring Semester Lab Harness



IO Scheduling/Optimization
- DRAM has inconsistent data rate
- Read ahead at a more aggressive pace
  - Wider vector
  - Bursting
  - Different clock rate
  - Leverage multiple channel DRAM
- Streaming to PE at compute rate



user_project_wrapper

wishbone

wb_axilite

config_ctrl

Integrator
- Disaggregator
- Aggregator

LogicAnalyzer

Tester

mailbox

Axis_Axilite

Axis_Switch

axis_out

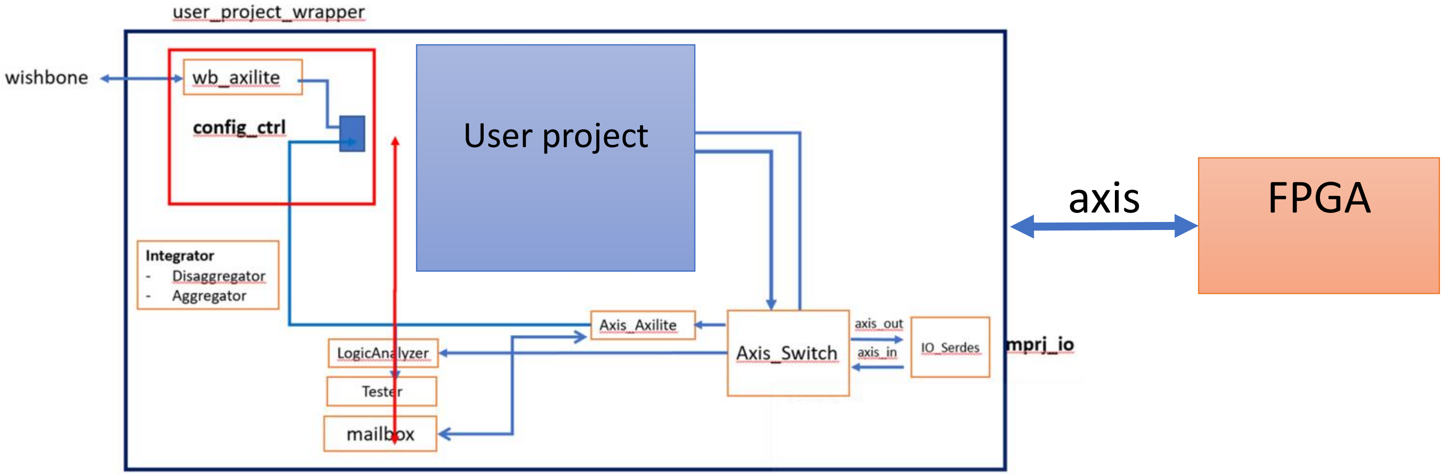
axis_in

IO_Serdes

mprj_io

TSRI 學生晶片 下線

- Die size: 2mmx2mm
- 40 pins
- Using FSIC Architecture



Special Project – Silicon Validation

Objective:

1. FPGA implementation for FSIC Validation System with user project integrated.
2. Test Program development, including RISC-V and Host program
3. Silicon Validation (in Oct)
4. Final Report

Collaborative Learning

- Work together in groups to solve problems, complete assignment, learn from each others
- Benefit: improve critical thinking skill, increased motivation, higher levels of engagement, develop social and communication skills
- Use technology tools – slack, discord, Github, youtube live, markdown
- Challenge
  - Actively engaged in the process
  - Setting goals and deadlines, assigning roles and responsibilities
  - Providing feedback and support
  - Increase peer evaluation

Collaborative Learning Workflow/Tools

- Cloud workplace tools - Google Form, spreadsheet
- Google Calendar
  - https://calendar.google.com/calendar/u/0/r/eventedit/MXZmZzBsanM1bnRyNjlldGE1cDB0c25qaWEgamlpbmxhaUBt
- Zoom/Google Meeting & Recording
  - https://us02web.zoom.us/j/89202408320#success
- Youtube live / Youtube channels
- Community for discussion
  - Slack https://boledu.slack.com/archives/C04SX3DLP5W
- Github
  - https://github.com/bol-edu/caravel-soc
  - https://github.com/bol-edu/caravel-soc_fpga
- HackMD
  - https://hackmd.io/?nav=overview
  - https://github.com/bol-edu/fsic-spec-dev

Summer Study Program

- Berkeley EECS151 - Introduction to Digital Design and IC
  - https://inst.eecs.berkeley.edu/~eecs151/fa22/
- Udemy Computer Architecture and Computer Organization Masterclass
  - https://www.udemy.com/course/computer-organization-and-architecture-course-masterclass/
- MIT Computer System Architecture (Advanced)
  - https://ocw.mit.edu/courses/6-823-computer-system-architecture-fall-2005/pages/lecture-notes/
- HLS/Verilog Tool installation, and basic labs
- Skills
  - Makefile, Scripting language (shell, awd/sed, perl, tcl)
  - Github
  - Markdown
  - How to raise questions in forum

Logistics

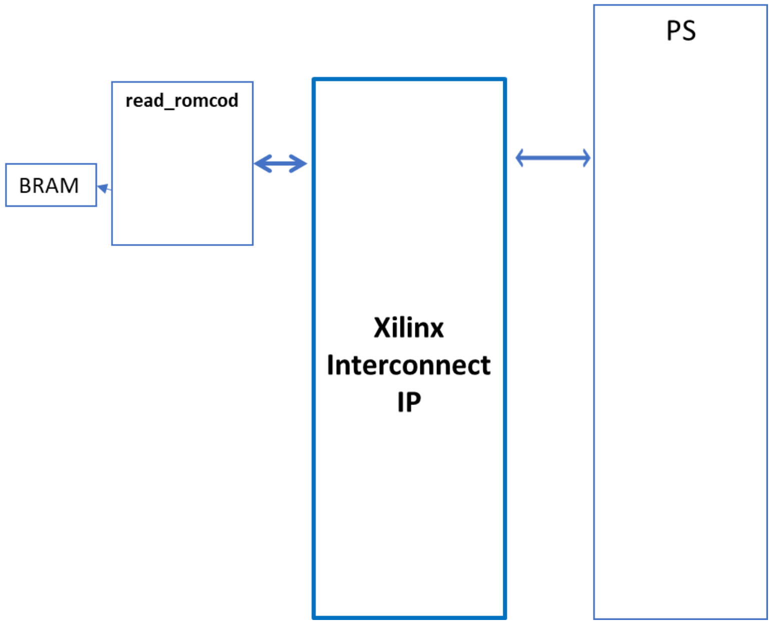
- 2-3 people in a team
- Autonomous course viewing / and group discussion
  - Audit SOC Design Course
- Weekly Presentation & Discussion Session
- Class representatives
  - Meeting logistics, meeting room booking
  - Taking meeting minutes
  - School communication

Q & A

Supplement

Lab#1 : AXI Master to read/write BRAM

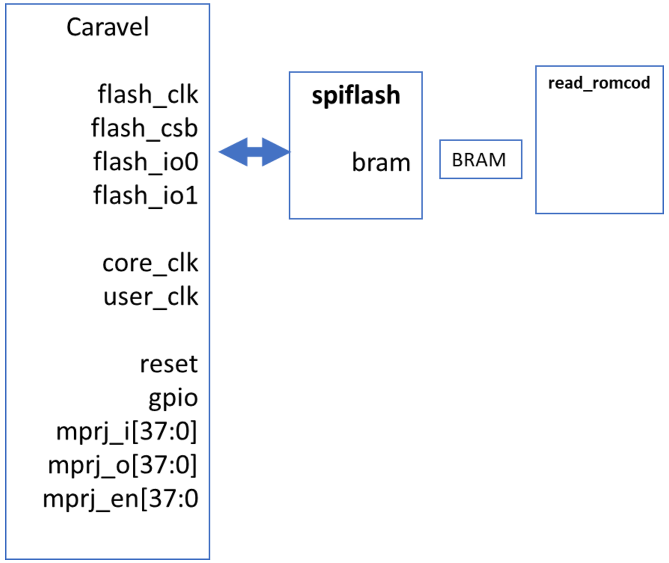
- Design source: read_romcode.cpp
- Lab Content
  1. Add another axi-master path to write to PS Memory
  2. Load program.hex (RISCV code from any of the Caravel testbench) to PS memory buffer
  3. Develop host code to load program.hex to BRAM, and read from BRAM.
  4. Compare the input and output buffer content is the same



Lab2 – spiflash protocol design and validation

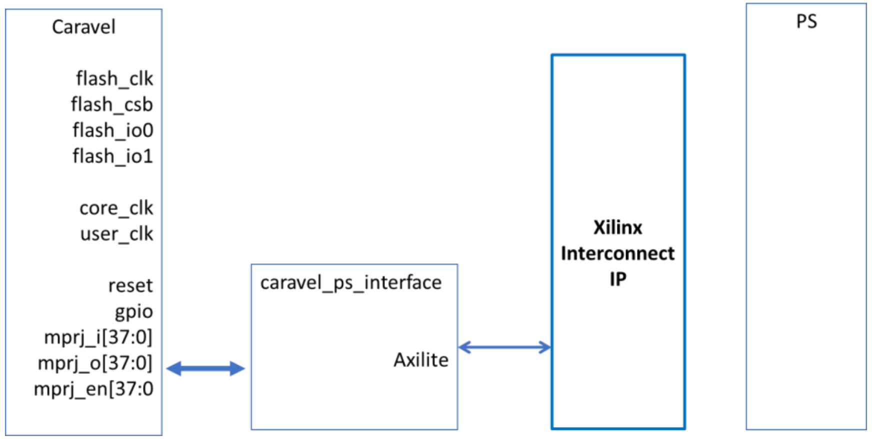
- Design source: spiflash.cpp
  - Note: this spiflash design only implements: single bit spi flash read.
- Lab Content:
  1. Develop flash controller spiflash_ctrl.v (you can find the design from Caravel) in Verilog
  2. Synthesize spiflash.cpp (this is flash memory device)
  3. Develop a simple BRAM behavior model, preload the content in testbench
  4. Integrate the spiflash_ctrl.v + spiflash.v and verify spiflash_ctrl.v can correctly read data from BRAM through spiflash.cpp

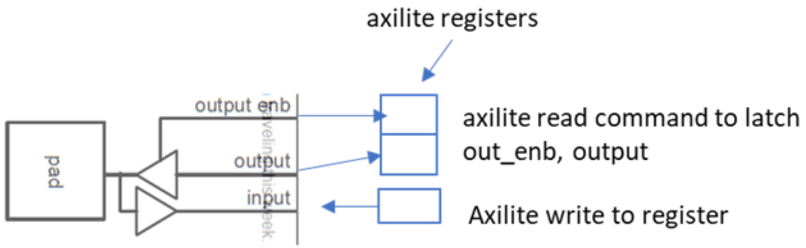
  Note: spiflash.cpp is not verified.



Lab3: Axilite access GPIO pins

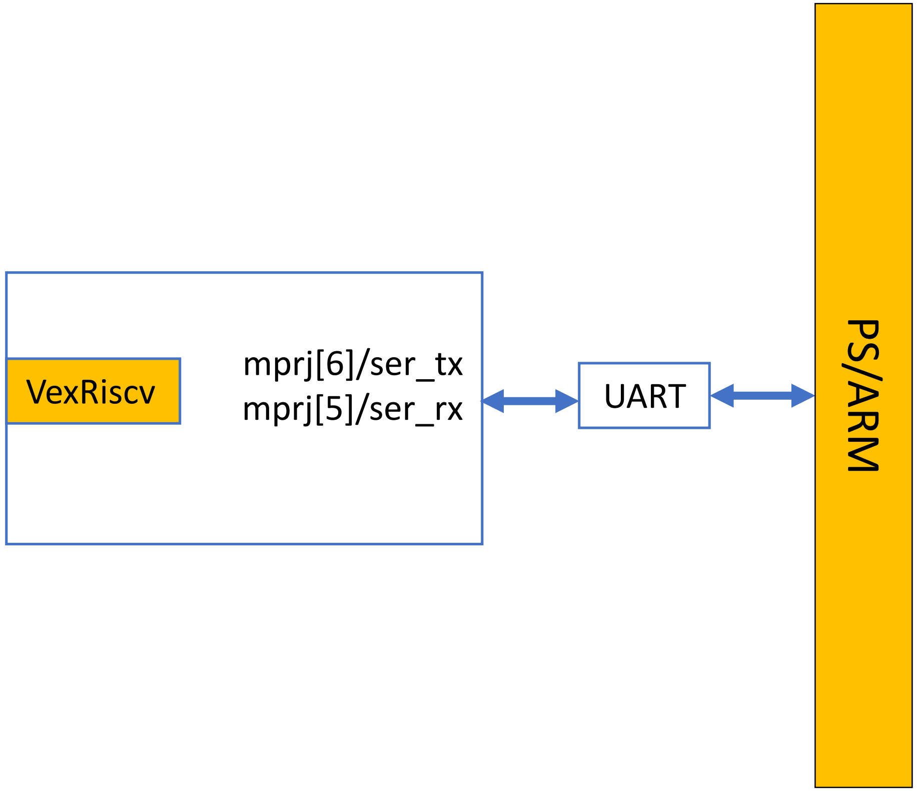
- Reference Design: caravel_ps.cpp
- Lab Content
  - Design a simple module mprj_control.v
    - Use one mprj_i pin (synchronize with host code) to stage through several steps, e.g.
      - Change mprj_o pins value
      - Some of mprj pins used for loop-back, e.g.
        - mprj_o[x] = mprj_i[n]
        - Control mprj_en accordingly
      - Host use axilite to read mprj_o, mprj_en values
  - Integrate mprj_control.v & caravel_ps.v in Block design – generate bitstream
  - Develop Python host code to verify its behavior




axilite registers
pad
output enb
output
input
axilite read command to latch out_enb, output
Axilite write to register

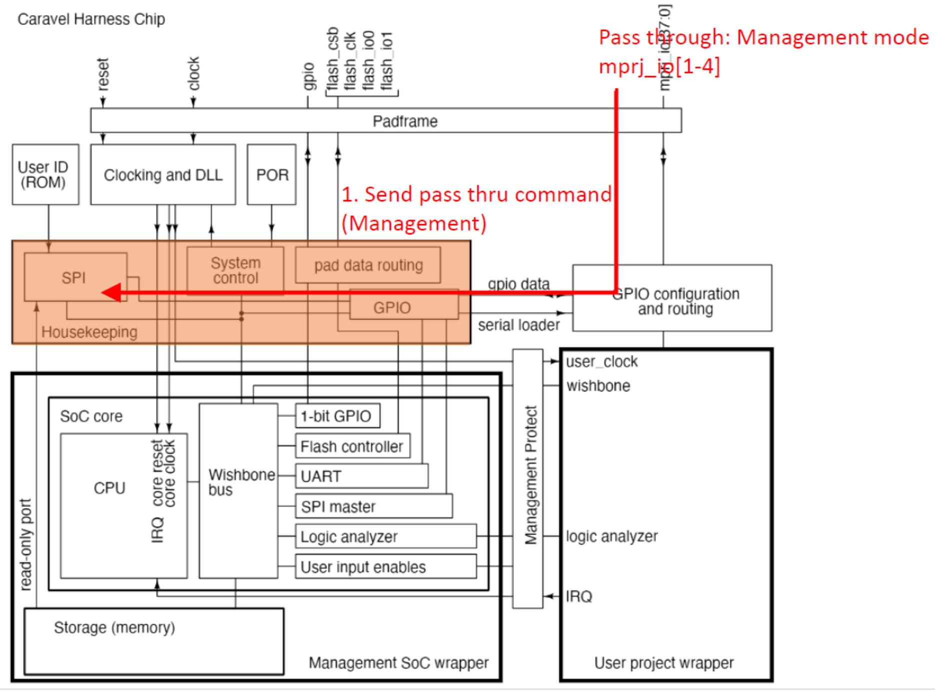
Lab4: UART RISCV-PS Communication

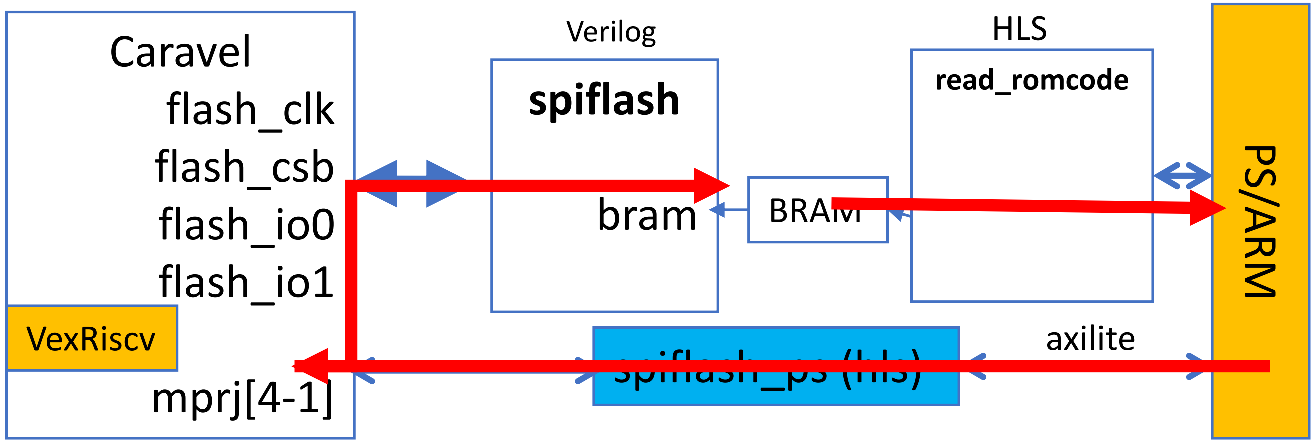
- Lab Content
  - Reference: tbuart.v - uart model (rx-only) in testbench
  - Design a module: uart.v
    - Implement uart tx/rx on Caravel side
    - Implement AxiLite interface on PS side
  - Integrate uart.v in Block design – generate bitstream
  - Develop Host Python code and RISCV Interrupt service routine to communicate
    - PS sends a character to RISCV by program UART axilite
    - RISCV interrupted to receive the character
    - RISCV send the character to PS by UART
    - RISCV sends the character to PS by mprj pins
    - PS read the same character either from UART registers, or mprj pins
- How to combine Debugplug ?



Lab5: Firmware Update

- Reference: Lab1, Lab2, spiflash.v
- Design a module: spiflash_ps.v
  - Add axilite interface on spiflash_ps.v
  - Add write command in spiflash_ps.v (controller)
  - Add spiflash write command
- Integrate spiflash.v, read_romcode, spiflash_ps in Block design
- Develop Host Python code and RISCV code
  - PS write .hex file to BRAM by spiflash_ps.v
  - Program read_romcode to move data rom BRAM to PS/Memory
  - Check data
- Implement firmware update procedure



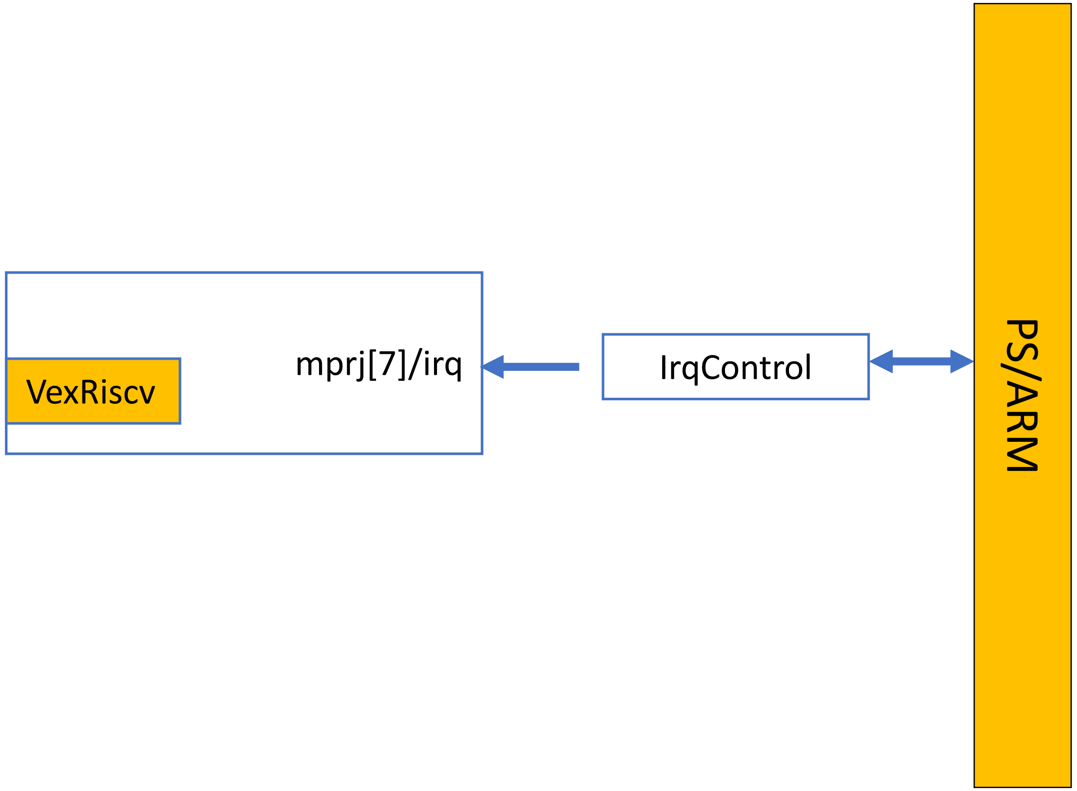


https://github.com/bol-edu/caravel-soc/files/11024963/caravel_SPI.pdf

https://github.com/bol-edu/caravel-soc/files/11024947/Housekeeping.function.by.Willy.pdf

Lab6: IRQ Handling

- Reference ResetControl
- Duplicate the IP ： IrqControl
  - Connect output pin to mprj[7]/irq
- Integrate IrqControl in Block design
- Develop RISCV irq handler
  - Link irq handler entry to interrupt vector
  - Enable irq
  - When receive the irq, send a code to mprj port
- Develop Host Python
  - Enable irq
  - Monitor mprj ports
  - Disable irq
- Replicate Matt Venn example below



https://github.com/mattvenn/caravel_user_project/tree/irq-demo
https://www.youtube.com/watch?app=desktop&v=pPgnVBguNW8