SOC Lab Summer Study Plan

6/20/2023

Jiin Lai

Target

- Lab-Driven Study on the following topics
  - Verilog
  - Logic Design
  - VLSI Electrical Property
  - RISCV
  - Computer Architecture
  - Script language

- Get ready for graduate level course in architecture/digital system

Reference Material

- Berkeley: Introduction to Digital Design and Integrated Circuits
  - Berkeley EECS 151 https://inst.eecs.berkeley.edu/~eecs151/fa22/
    - Lecture note: https://drive.google.com/drive/folders/1MW8sFs1gKWi86wfhuXOqe0HVXAEsVGQy?usp=drive_link
- Verilog
  - NTU CSVD Verilog handout
    - https://drive.google.com/drive/folders/1npOiZi_TIMm7E-ThwZnlZrl_Y-QCg6Zw?usp=sharing
  - SpyGlass Lint Rules Reference Guide
    - https://drive.google.com/drive/folders/1ebHXZCYNUpUiZOyEs7zu19O32wpxHWWF?usp=sharing
  - Stuart : Verilog and System Verilog Gotchas
    - https://drive.google.com/file/d/1m0l-4LMGMWgvn2sWUKu4mK9qgacssoyj/view?usp=sharing
- Logic Design
  - Udemy High-Level-Synthesis for FPGA, Part I, Part II
  - Vaibbhav Taraate: "Digital logic design using Verilog: coding and RTL synthesis
    - https://drive.google.com/file/d/1tzX4DvgWTeP8t1nU_UK19rkGtLfwmemC/view?usp=drive_link
- HLS RISCV
  - https://github.com/bol-edu/course-lab_riscv
- Computer System
  - Sarah/David Harris: "Digital Design and Computer Architecture"
    - https://drive.google.com/file/d/1E3NZBR22JBVPx6255S4KGBn1unRQRdrg/view?usp=drive_link

eBOOK Download: https://libgen.is/

Study Plan

wk	Date	Meeting (Q&A + Presentation)	Area	Reading Assignment	Lab
-1	6/20	Summer program introduction	Verilog	L3 + NTU Verilog / Lint / Gotchas	Lab1: Tool installation: Vitis HLS/ Vivado Tool
0				L4 + NTU Verilog / Lint / Gotchas, PYNQ Video	Lab#2 - PYNQ Lab - GCD (xsim)
1	7/5	Verilog/Lint	Logic Design	L#5(CL I), L#6(CL II), L#21 (FF), Script, Udemy (Com)	Lab - combinational (1.5 w)
2	7/12	Verilog/Lint		L#18 (Adders), L#19(Multiplier), L#20 (Multiplier II) Udemy(Comb), Script: Perl	Lab - Combinational + Sequential (2.5 w)
3	7/19	Lab - combinational		L#22, L#23 SRAM, Udemy(Seq), Script: Perl	Lab - Sequential
4	7/26	Lab - combinational, Script	RISCV	L#7, L#8 - RISCV	Lab - Sequential
5	8/2	Lab - Sequential, Script		L#9, L#10 - RISCV , Script: Tcl	Lab - RISCV HLS/Verilog (1/5w)
6	8/9	Lab - Sequential + Lint + Script		L#10 - RISCV, Script: Tcl	Lab - RISCV (2/5w)
7	8/16	Logic Design/ Vaibbhav	VLSI Physics	L#13 - CMOS Transistor / Logic Gate L#14 - Inverter-Delay L#15 - Inverter Chain Delay	Lab - RISCV (3/5w)
8	8/23	Lab-RISCV		L#16 - Logical Effort L#17 - Wire Energy, Static / Dynamic Power	Lab - RISCV (4/5w)
9	8/30	Lab - RISCV	Computer System	Chap8 - Memory (Cache, Virtual Memory)	Lab - RISCV (5/5w)
10	9/6	Computer System		Chap 9: Embedded IO	

Content

- Off Class
  - Reading assignment
  - View Udemy - High-Level-Synthesis for FPGA, Part I (6 hr), Part II (9 hr)
    - 3 weeks (Monday – Friday – 1 hr/day, 2 sessions)
  - Labs
  - Expect workload 15-20 hour / week
- In Class
  - Key concept review (you will be asked in the review session)
  - Q & A (Submit your question in the following Google form)
    - https://docs.google.com/forms/d/e/1FAIpQLSeExQQQVT0Gy8FpZVe1YNlh3jxx8d-h0qYBCuI3PgeiB9gteQ/viewform?usp=sharing
  - Presentation & Code review (Lint, Lab, Script) – sign-up
    - Sign-up sheet: https://docs.google.com/spreadsheets/d/1Bmc2dfkX-SC3sEBGm3PmO4l_gr7F38NvWj0PE2xC48c/edit?usp=sharing

Course Infrastructure

- 2-3 people / Group - find your partners
- Google Meet - meeting notice, meeting recording
- Recording upload to youtube channel
- Access FPGA online
- Slack Channel (soclab) for general communication
  - You will receive invitation
- Github: https://github.com/bol-edu/soclab-nthusp23
- HackMD – everyone maintain a study_record.md
  - https://hackmd.io

Lab Logistic - Kevin

- 實名註冊 （ PYNQ-Z2, KV260)
  - Account: your email
  - Password: (You will receive the password, by email)
- Slack
- Github
- Xilinx Xsim/GTKWave

Github - soclab-nthusp23
https://github.com/bol-edu/soclab-nthusp23

- Discussion - Raise question here!
- Github structure

```
|- ppt                          # presentation ppt
|- students
|    |- name
|         |- labname
|              |- .v, _tb.v     # Verilog code & testbench,
|              | README.md      #  lab report
|              | Synthesis.log  #  Xilinx synthesis log
|         | study_record.md # Learning journal  -> link to your HackMD
```

Lab

Given HLS code, Implement with Verilog and Testbench

Compare timing/resource for HLS & Verilog implementation

Lab-Tool Installation - Week -1, 0

- Vitis/Vivado Tool installation
  - https://github.com/bol-edu/course-lab_1
  - Ref: HLS Tools Installation Guide 2022.1_Ubuntu.md
    - https://github.com/bol-edu/course-lab_1/blob/2022.1/HLS%20Tools%20Installation%20Guide%202022.1_ubuntu.md
  - 2022.1-Workbook-Lab1.pdf
- HLS Lab
  - https://github.com/bol-edu/course-lab_2
  - Using 2022.1-Workbook-Lab2-KV260.pdf   - We will use KV260
- Verilog Lab (Xilinx xsim) – GCD example
  - https://github.com/bol-edu/soclab-nthusp23/tree/main/lab/02.xsim-gcd

Lab – Combination (1.5 w)

- HLS Sources:
  - https://drive.google.com/drive/folders/18RtPmHE-3GeCPPSNA9iw87qRM-6Ya-O1?usp=sharing
- Labs
  - parity_generator
  - Integer_division_modulus
  - leading_one
  - binary2bcd_div
  - binary2bcd_double_dabble
- Given HLS code, Implement with Verilog and Testbench
- Compare timing/resource for HLS & Verilog implementation

Lab – Sequential (2.5 w)

- HLS Source:
  - https://drive.google.com/drive/folders/1zI78hQ-1eFtofW3YxeU8RGKxLD1inVyV?usp=sharing
- Labs
  - serial2parallel
  - parallel2seral
  - combination_lock
  - vending_machine
  - bcd_counter-multicycle
  - iir
  - uart_receiver
  - uart_transmitter

- Utilities
  - timer
  - debouncer
  - counter
  - clock_generator
  - pulse_generator
  - single_cycle_regular_pulse
  - edge_detector

- Given HLS code, Implement with Verilog and Testbench
- Compare timing/resource for HLS & Verilog implementation

Lab-RISCV (5 w)

- Berkeley EECS 151 https://inst.eecs.berkeley.edu/~eecs151/fa22/ (L7-L10)
- A HLS implementation for RISCV
  - Lecture Recording
    - https://www.youtube.com/watch?v=Zlgf7JqyOg4&list=PL5CoDA0gtOHVncdI7JZw9djVa-BS0WFz4&index=16&t=1193s
    - https://www.youtube.com/watch?v=RFKYyFZEDkA&list=PL5CoDA0gtOHVncdI7JZw9djVa-BS0WFz4&index=17
  - Github for RISCV HLS
    - https://github.com/bol-edu/course-lab_riscv
- Lab
  - Implement a 5-stage RISCV
  - Compare area/timing (Compare HLS & Verilog) – Timing analysis & Optimization
  - Enhance with branch prediction, data-forwarding, register renaming (optional), cache, add new instruction (multicycle)
  - Benchmark (matmul.c, qsort.c) & compare cycle count
  - Use the new RISCV in your future SOC design

Lint Rules

- Document Link
  - https://drive.google.com/drive/folders/1ebHXZCYNUpUiZOyEs7zu19O32wpxHWWF?usp=sharing
  - STARCRules
  - ClockResetRules
    - Clock Domain Crossing Synchronization
  - LatchRules
    - Rules in SpyGlass latch
  - LintRules
    - Rules in SpyGlass lint
  - SimulationRules

Basic Design Constrains

- Rules for Naming Conventions
- Rules for Synchronous Design
- Rules for Initial Reset
- Rules for Clocks
- Rules for Asynchronous Circuits
- Rules for Hierarchical Design

RTL Description Techniques

- Rules for Combinational Logic
- Rules for always Constructs of Combinational Logic
- Rules for Flip-Flop Inference
- Rules for Latch Description
- Rules for Tristate Buffer
- Rules for always/process construct that takes circuit structure into account
- Rules for if Statement Description
- Rules for case Statement Description
- Rules for Operator Description
- Rules for Finite-State Machine Description

Study Lint Rules

- Describe the rule
- Potential issues,
- How to Debug and Fix
- Example Code and/or Schematic

Rule:
States are not updated on the same clock phase

Potential Issues:
1. The synthesis tool can get confused about which edge to use for updating the register.
2. RTL and gate-level simulation results may not match

How to Debug and Fix:
Break the sequential logic into multiple sequential logic blocks, each of which can independently meet the requirement. If the register depends on both edges of the clock, describe the sequential nature separately and use the combinational logic to generate the final output.

Example Code:

```
module test(out1,out2);
  output out1,out2;
  reg out1,out2,a,c,clk;
  always begin
    @(posedge clk) out1 <= c;
    @(negedge clk) out2 <= a;
  end
endmodule
```

Verilog and SystemVerilog Gotchas - Stuart Sutherland

- Gotchas_101: Stuart
- Chap2: Declaration and Literal Number
- Chap3: RTL Modelting
- Chap4: Operator
- Chap5:: General Programming

Why Scripting ?

- Scripting languages are used to automate tasks that one would otherwise do manually. you spend more than 10 minutes doing something manually at least every two days, it's better to automate it.
- Why are Perl and TCL scripting languages used in the VLSI industry?
    - https://www.quora.com/Why-are-Perl-and-TCL-scripting-languages-used-in-the-VLSI-industry-Why-not-using-other-scripting-languages?encoded_access_token=1c5411c7f5eb4ff6b6ef686ce3656eaf&force_dialog=1&provider=google&success=True
- What scripting languages to learn

• Bash (awk/sed)	
• Makefile	- automate IC design flow
• Perl	- log file processing
• Tcl	- access EDA tool database

- Exercise: Convert Xilinx HLS-generated Verilog to be synthesizable by ASIC flow

Script Resource

- Perl & TCL lecture material
  - https://drive.google.com/drive/folders/11qFShiNu-3_HPkCnP6jNVZB6Dz1XBYDM?usp=sharing
- R.L. Schwartz, B.D. Foy, T. Phoenix. Learning Perl: Making Easy Things Easy and Hard Things Possible. O'Reilly Media;
- Xilinx Tcl
  - command reference guide https://docs.xilinx.com/v/u/2019.2-English/ug835-vivado-tcl-commands
  - tcl scripting : https://docs.xilinx.com/r/en-US/ug894-vivado-tcl-scripting/Tcl-Scripting-in-Vivado

Digital Logic Design - Vaibbhav

- Chap 15 – Non-synthesizable Verilog Construct
- Chap 22 – Mulitple Clock Domain
- Chap 23 – FIFO Design
- Chap 24 – Low Power Design

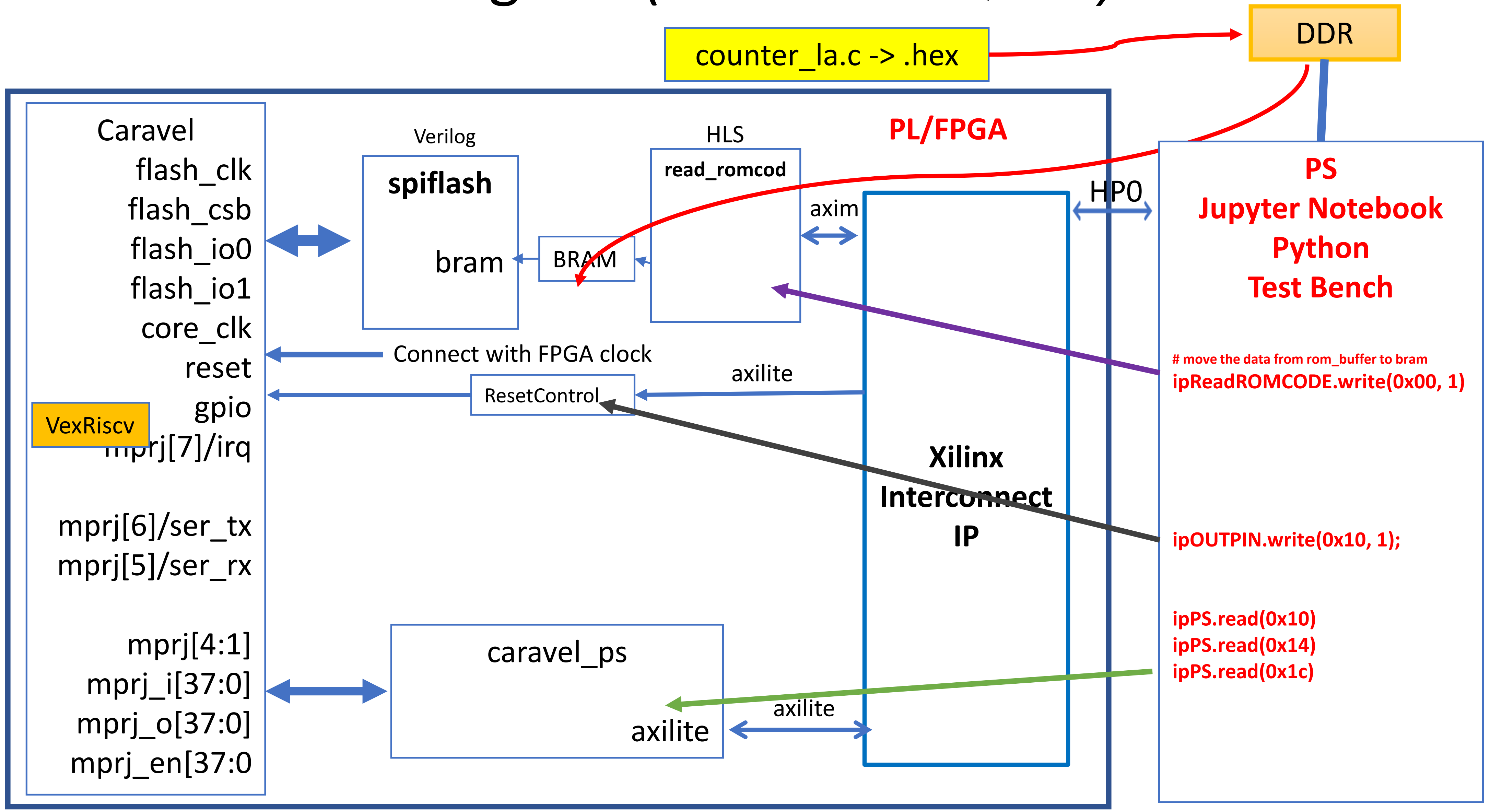
Computer System – Sara/David

- Chapter 8 – Memory System
  - Cache
  - Virtual Memory
- Chapter 9 – Embedded I/O System
  - Memory-mapped IO
  - GPIO
  - Serial I/O – SPI, UART

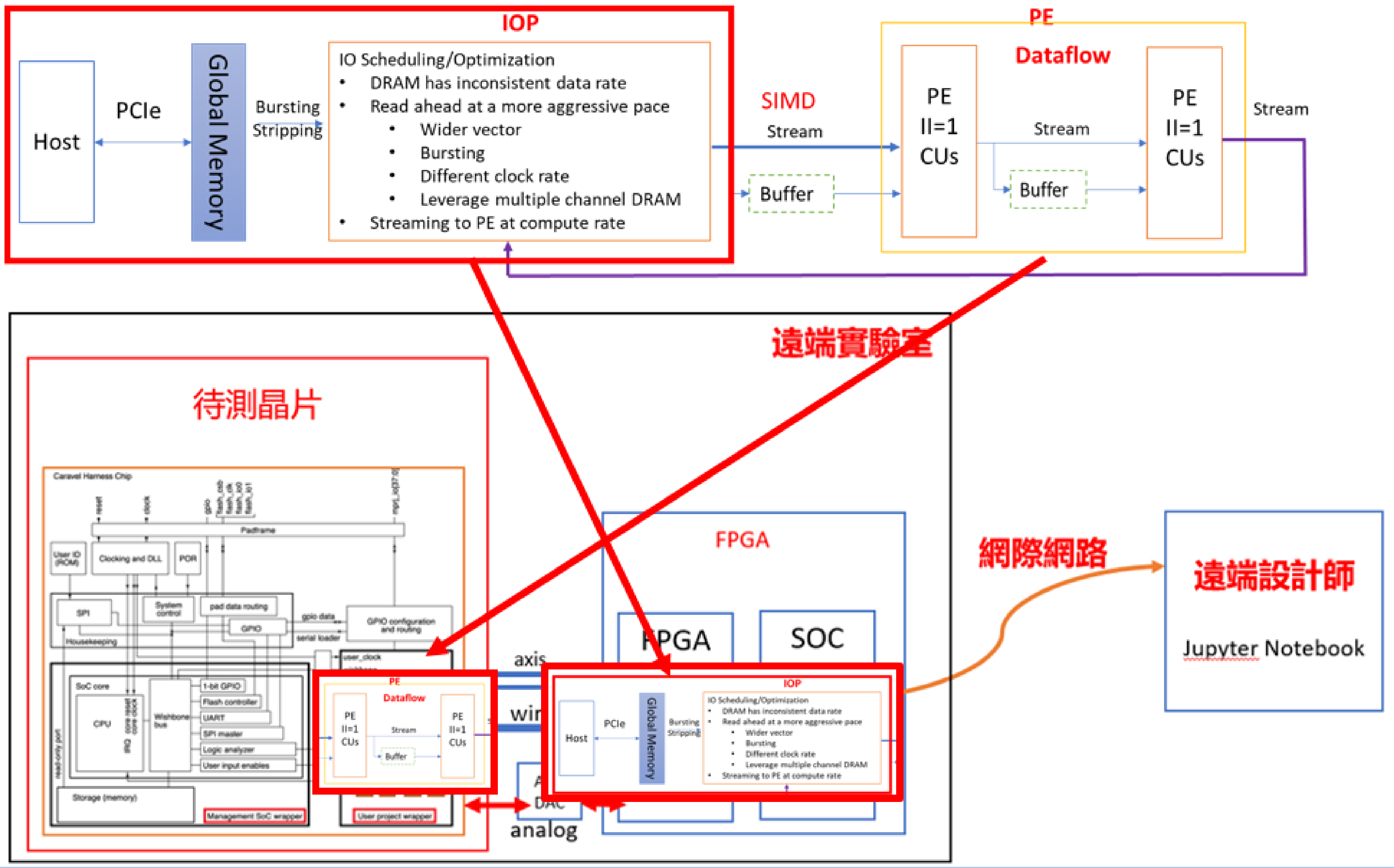
Final Project & Grade

- Summer : Design a high performance RISCV Core
- 1st Semester : Design an SOC - FPGA
  - RISCV + Peripherals + Memory + Bus Interconnect
  - Complete a task in less cycles
- 2nd Semester : Application Accelerator Integrated with SOC
  - SOC + FPGA
  - Tapeout Process
- Special Project Contest

CaravelFPGA Block Diagram (Based PYNQ-Z2)



Embed Application Accelerator in FSIC


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