Optimize RISC-V Processors with iEDA in "One Student One Chip" Initiative

Zihao Yu(余子濠)

2025.06

中国科学院大学
University of Chinese Academy of Sciences

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

OSOC Initiative

Based on open-sourced, practice-oriented, open learning

Everyone is welcomed. No limitation on

- university
- major
- grade
- basis

SW & HW co-design

Logical design & Physical design co-ordination

Application
Runtime
(Simple) OS
ISA(RISC-V)
Micro-architecture
Circuit
Synthesis
Physical design
Physical verification
GDSII

CS
EE

GCC U-boot OpenSBI
UEFI **Software** LLVM
QEMU Linux

XiangShan Coherency
OoO
IP
Chip Prefetch
Cache Branch
Extension Prediction

Technology mapping
Place Standard cell
EDA Floorplan
Route
Timing analysis
Clock tree Equivalence

Education equality

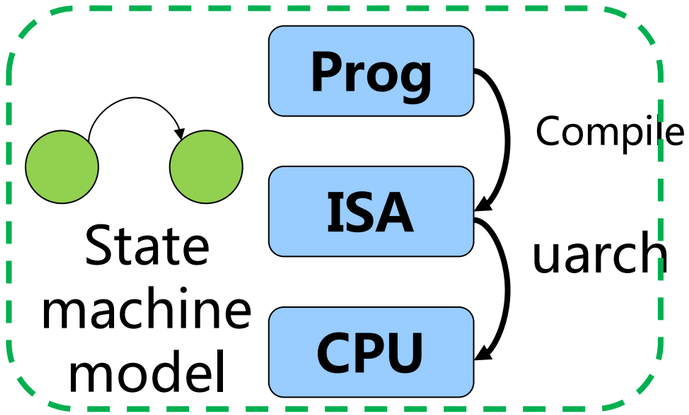
Full-stack training

Enter community or company

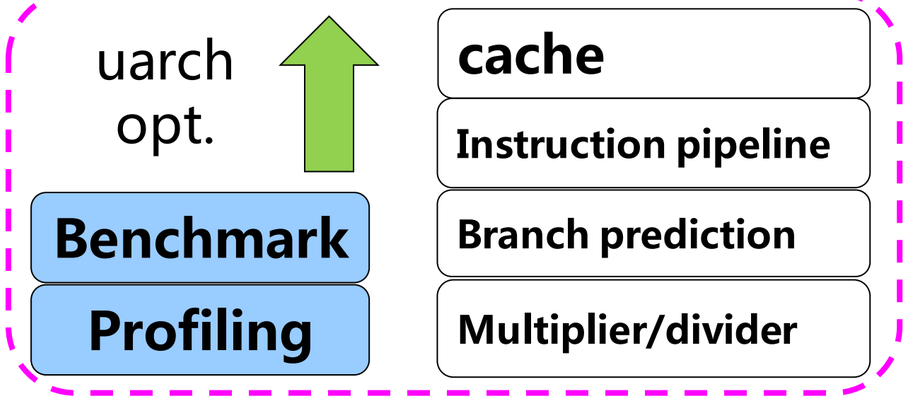
Knowledge Diagram

App	From C to binary; ELF and Linking	Game	OS app.
Runtime	Abstract Machine – Bare-metal runtime	I/O library	C Library
OS			Self-built OS
ISA	RISC-V; machine-level representation of program	I/O Inst.	Trap Inst.
uarch	ISA Simulation; Single-cycle CPU	Bus; Device	Exception
Circuit	SoC: UART, SPI controller, Flash chip……		
Synthesis	Open -EDA		
Physical design			
Physical verification			
GDSII			

CS EE



Build first, then opt.



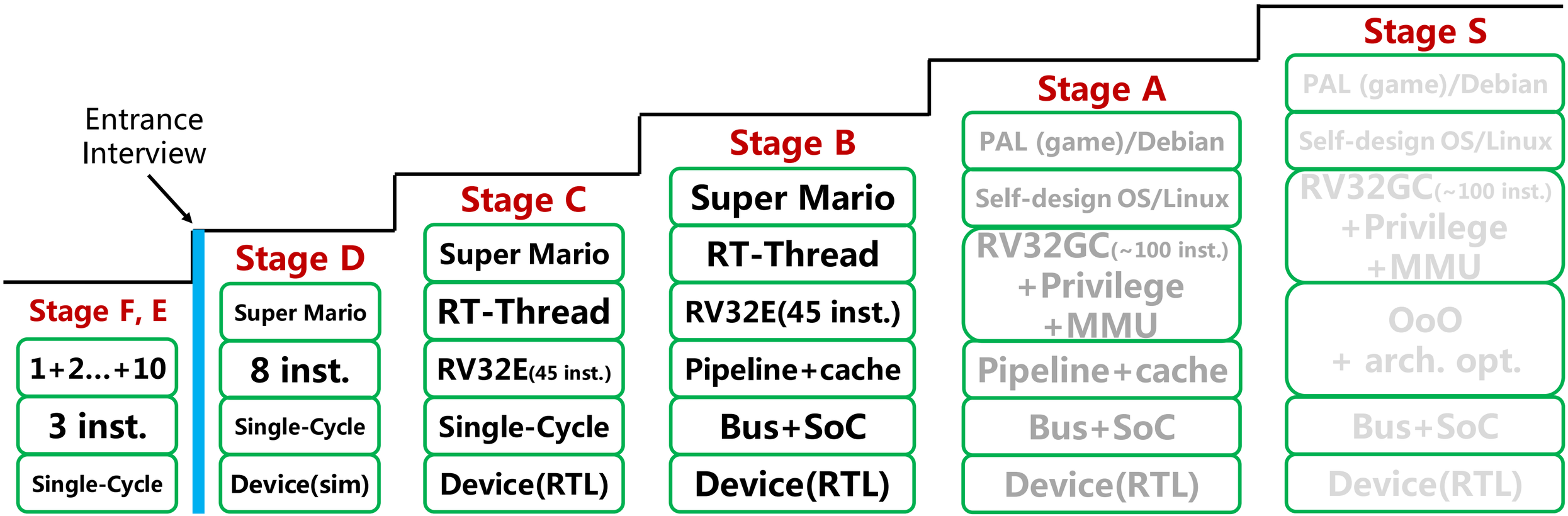
How to solve problem by yourself

- Try and analyze before asking
- RTFM, RTFSC, STFW
- Use/improve/create tools

Infrastructure : linter, sanitizer, printf, trace, gdb, waveform, profiler……
Good programming style, testing, assertion

Learning Stage Division

- Small, simple system -> Large, complex system
- Small program -> Real program



Building a Full System (Middle of Stage B)

- Understand the entire system stack
  - Programs, operating systems, ISA, processors, circuits
- Understand how the system works
  - How do games run on the system?

Super Mario

RTOS(RT-Thread)

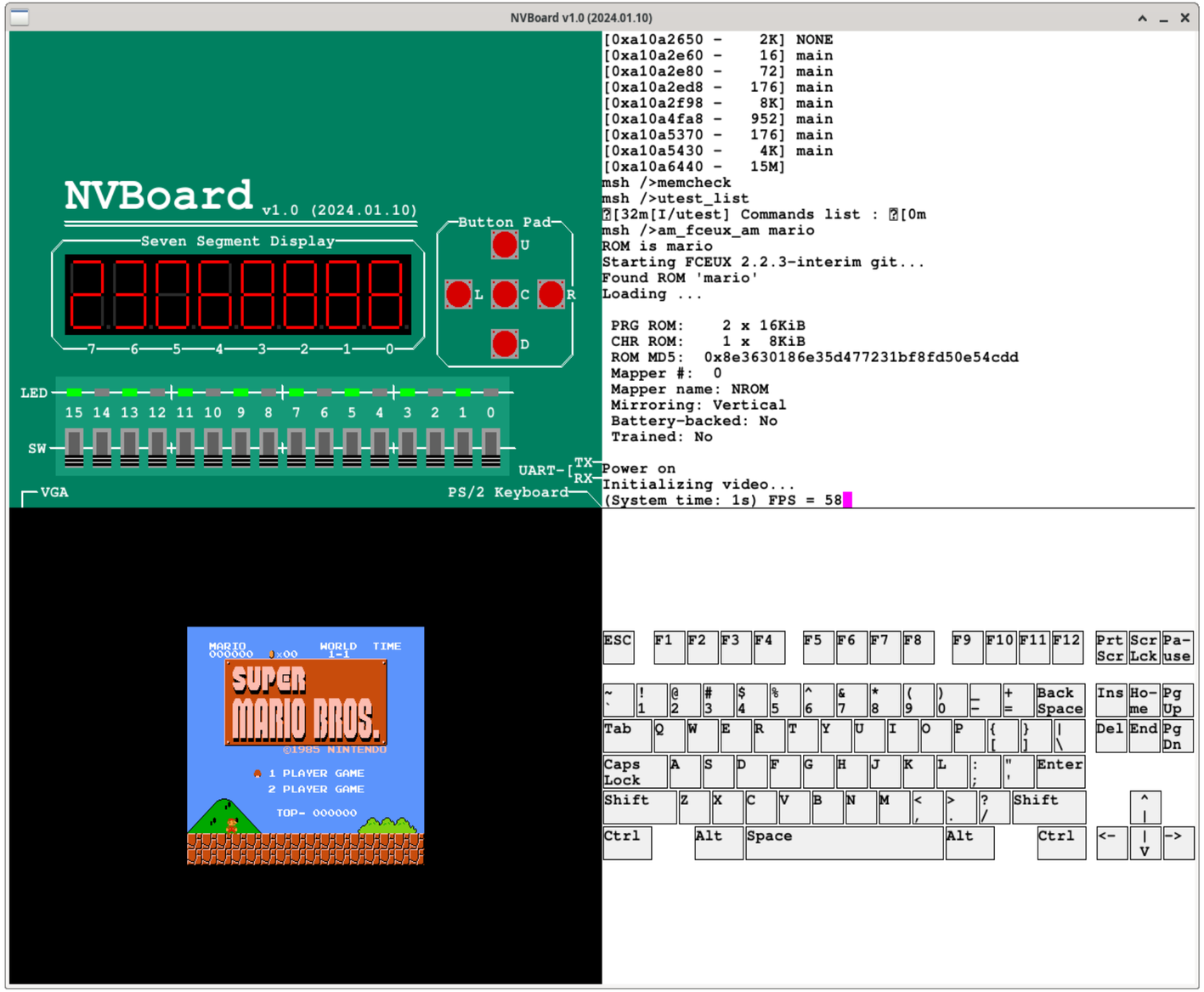
AM Runtime

RISC-V ISA

NPC Processor

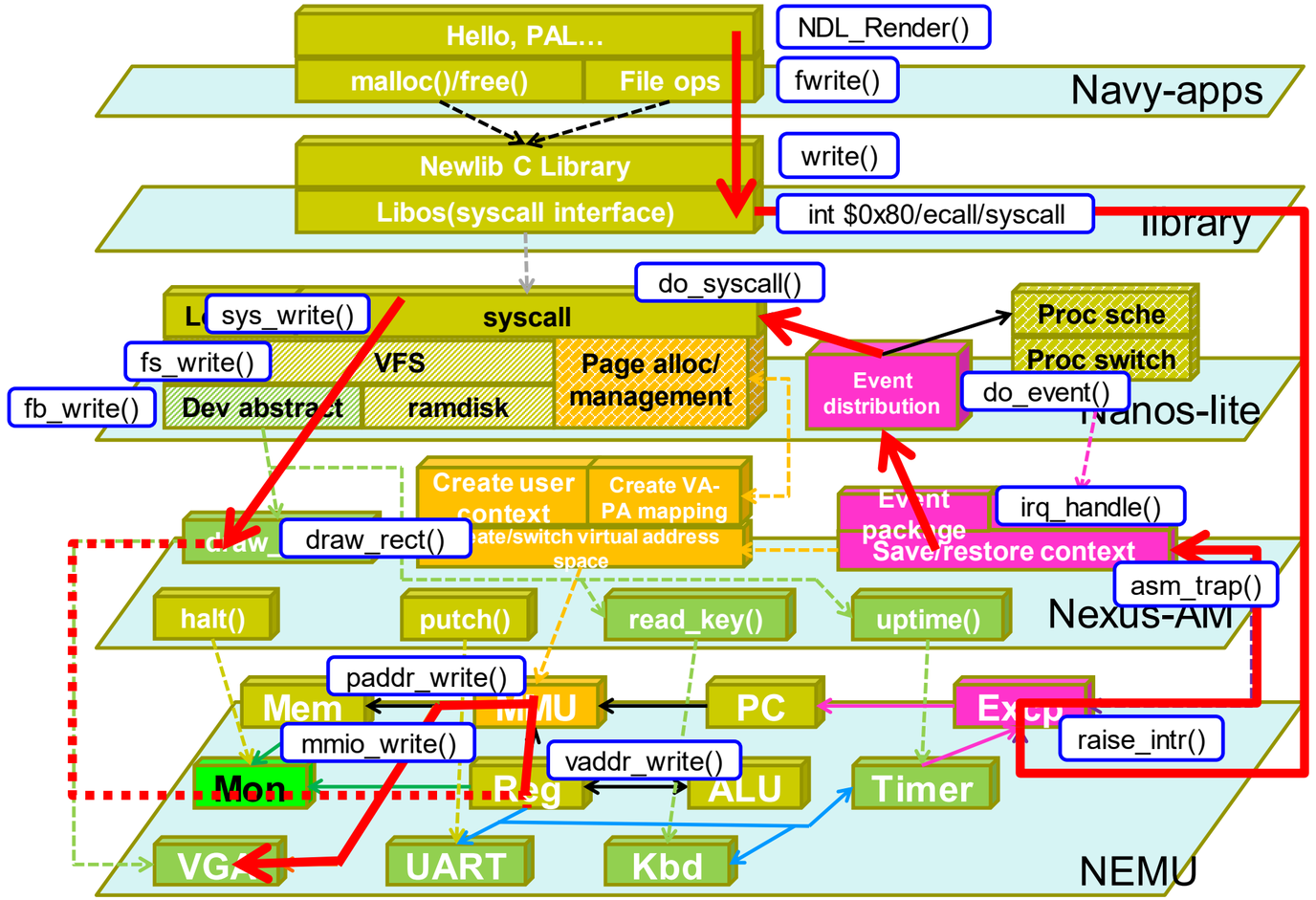
oSoC

NVBoard



Case – How games update screen

- Let students build a RISC-V computer system from scratch
  - Understand how program runs on their own processor

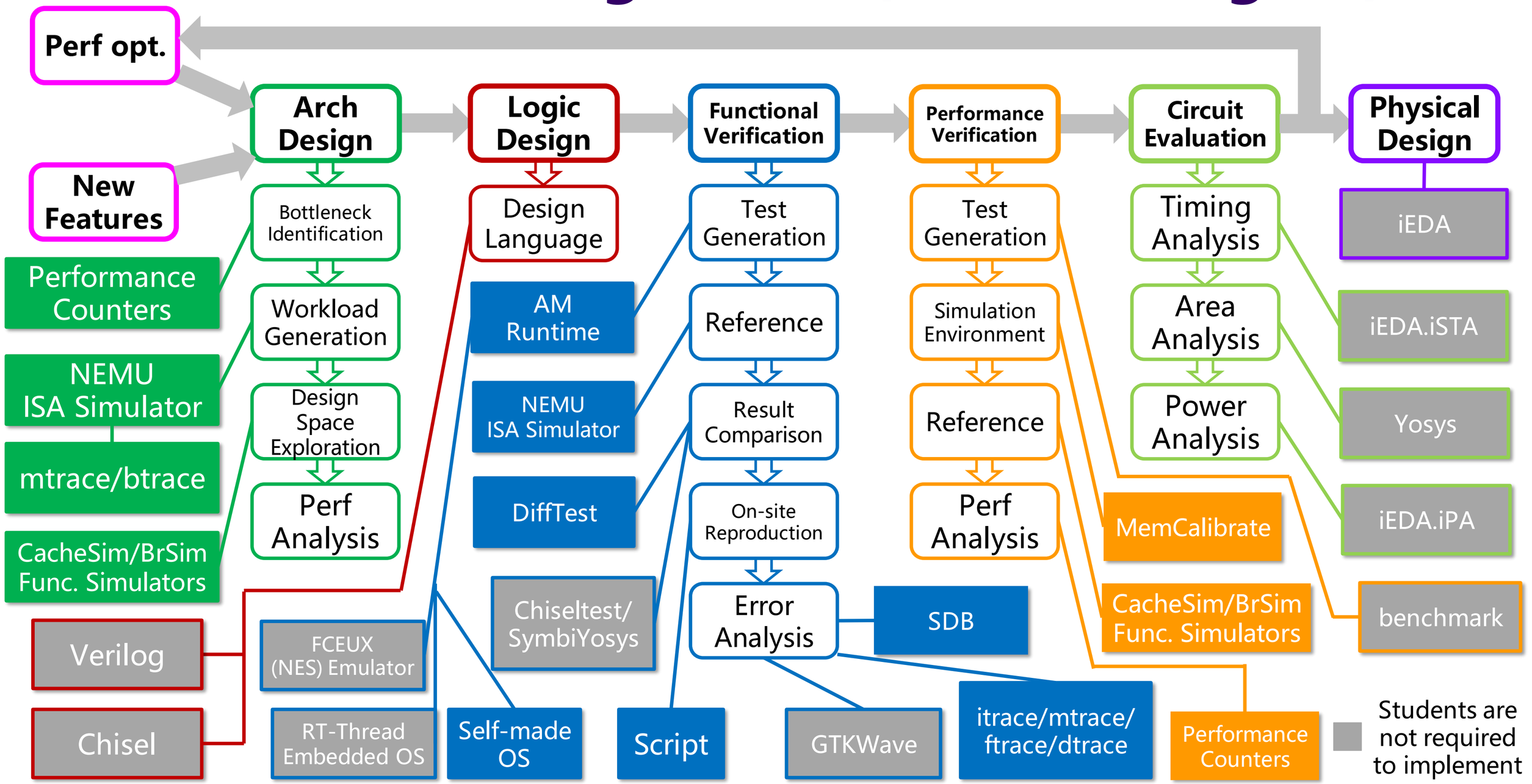


virtualization
concurrency
persistence

TRM
IOE
CTE
VME

Common data flow
Debug information
Memory mapped I/O
Port I/O
Interrupt

Processor Design Flow (Late of Stage B)



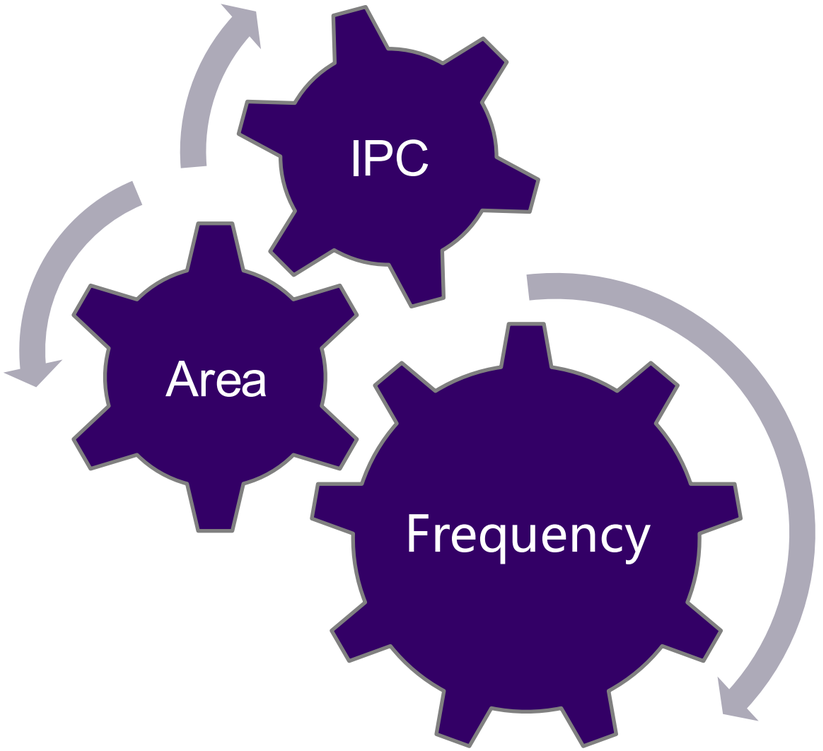
Learn More Beyond RTL

- **Advanced Verification Methods**
  - DiffTest – check instruction behavior on line
  - Formal verification – prove correctness or generate counterexamples
    - Memory w/ cache (DUT) v.s. memory w/o cache (REF)
    - Pipeline (DUT) v.s. single-cycle (REF)
- **Performance Evaluation and Optimization**
  - IPC - evaluate with simulators (CacheSim, BranchSim) + analyze with Performance Counter
  - Frequency - Yosys (Synthesis) + iEDA.iNO (Netlist Optimization) + iEDA.iSTA (Timing Analysis)
- **Power Evaluation and Optimization Methods**
  - iEDA.iPA(Power Analysis)
- **Full Physical Design Flow**
  - iEDA(Open-source EDA Tools)

Case – Trade-off in Optimization

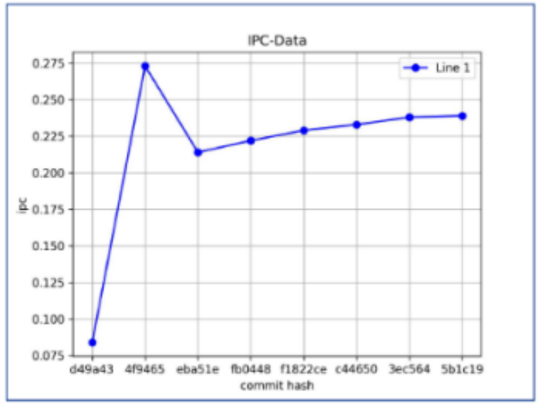
- Students learn to make trade-offs among frequency, area and IPC

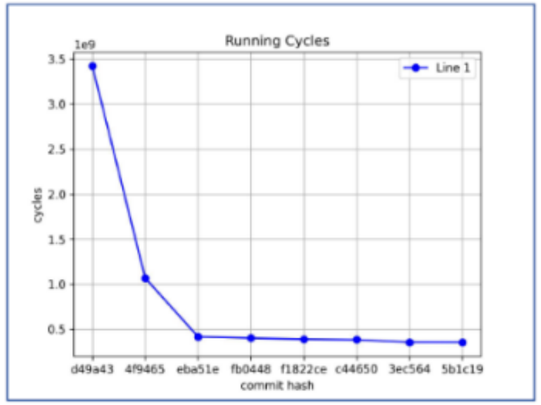
IPC	Frequency	Area



性能记录

	A	B	C	D	E	F	G	H	I
1	性能分析								
2	commit	bench	架构	说明	仿真周期数	仿真指令数	IPC	综合频率	综合面积
3	bf70feb4f77732ace4c720ee5a5f5a29fc8f5dae	dummy	Eryth-Multi		15316338	46509	0.00304	370.462MHz	26580.32
4	ad64212dbf61095221f92e2ad6500c518e70f57b		Eryth-Multi	STA时禁用commit模块				371.154MHz	26271.49
5	bb2c03f1d37014b86bb8b54b975686f9422d90db		Eryth-Qingyuan	单发射五级流水。静态预测。无CSR支持				399.381MHz	30805.19
6	489c97728a4526c24fb17fc78c8ab45239a0fbd0	dummy	Eryth-Qingyuan		18390887	39636	0.00216	397.640Mhz	30796.42
7	19184c7ad51acad412c30d04546f435a2a1deaf9	dummy	Eryth-Qingyuan	BPU动态预测	16396636	46266	0.00282	330MHz	37961.39
8	6edea36d4112e7bb12f3c3eec62ef2ac86db3dd6	dummy	Eryth-Qingyuan	使用AXI的SDRAM	16389426	141429	0.00863	330.234MHz	37878.13
9	f33d84ae7fecf49c3005cb48f841abbbae338c7d	dhrystone	Eryth-Qingyuan	循环次数：10000	141452948	10001872	0.07071	330.011MHz	39061.57
10	7a2afe007fa505b6f31f066cdd719a2bde4d0dca	dhrystone	Eryth-Qingyuan	循环次数：10000	141713484	10001906	0.07058	330.011MHz	39061.57





Case – Optimizing Timing

- Students can synthesize their own RISC-V processors with open-sourced PDK read the timing report
  - IHP PDK, nangate45…

Sg13g2_dfrbp_1

Point	Fanout	Capacitance	Resistance	Transition	Delta Delay	Incr	Path
clock (port)		5.357	0.000	0.000		0.000	0.000r
clock (clock net)	1851				NA		
_13173_:CLK (sg13g2_dfrbp_1)		0.003	0.000	0.000		0.000	0.000r
clock core_clock (rise edge)						0	0
clock network delay (ideal)						0.000	0.000
_13173_:CLK (sg13g2_dfrbp_1)		0.003	0.000	0.000		0.000	0.000r
_13173_:Q (sg13g2_dfrbp_1)		0.042	0.000	0.140		0.275	0.275f
npc_lsu_io_in_bits_rdecode_isMret (net)	5				0.000		
fanout_buf_1848:A (sg13g2_buf_8)		0.009	0.000	0.140		0.000	0.275f
fanout_buf_1848:X (sg13g2_buf_8)		0.024	0.000	0.029		0.122	0.397f
fanout_net_1848 (net)	5				0.000		
fanout_buf_1843:A (sg13g2_buf_8)		0.009	0.000	0.029		0.000	0.397f
fanout_buf_1843:X (sg13g2_buf_8)		0.023	0.000	0.024		0.076	0.472f
fanout_net_1843 (net)	5				0.000		
_07103_:C (sg13g2_nor4_1)		0.003	0.000	0.024		0.000	0.472f
_07103_:Y (sg13g2_nor4_1)		0.008	0.000	0.198		0.182	0.654r

Learning Materials are opened

The 6th "One Student One Chip" Program Home Page

- Time: Every Saturday 15:00~17:00 China Standard Time
  - Bilibil Live | recording

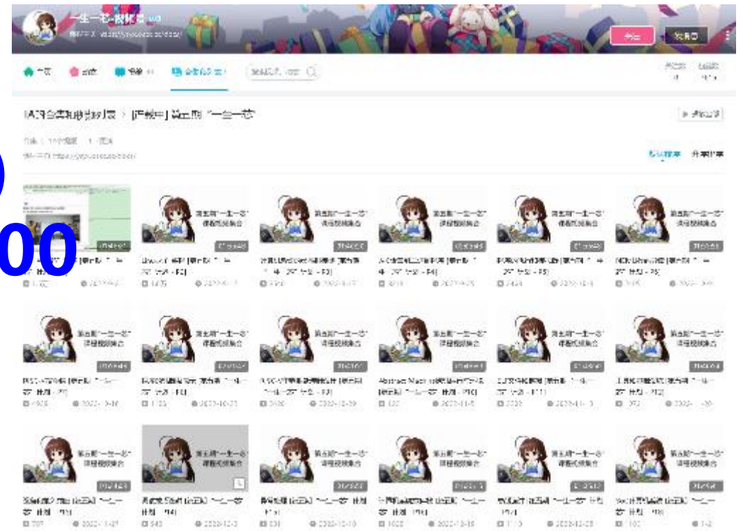
Learning Objectives

"One Student One Chip" will develop your general skills. At the end of the course, you will have a better understanding of the following questions:

1. how processors are designed?
2. how programs run on computers?
3. how to optimize the performance of a processor?
4. how to use/design the right tools for efficient debugging?
5. how to write your own test cases for unit testing?
6. how does an RTL design generate a flowable layout?

We will guide you to design a RISC-V pipeline processor. Run an operating system on your processor. Run a real game on the OS. The processor that achieves the target will be connected to the SoC and will be given the opportunity to tapeout.

- **Course Website**
- **Handouts(420,000 words)**
- **Slides(>1,000 pages, 85,000 words)**
- **Videos(> 50 hours)**



Account in Bilibili.com: 一生一芯-视频号

Linux system installation and basic usage

Install a Linux operating system

We're going to reuse the contents of the PA handout, and we're going to ask you to follow PA0 to install Linux OS.

Get "One Student One Chip" code framework

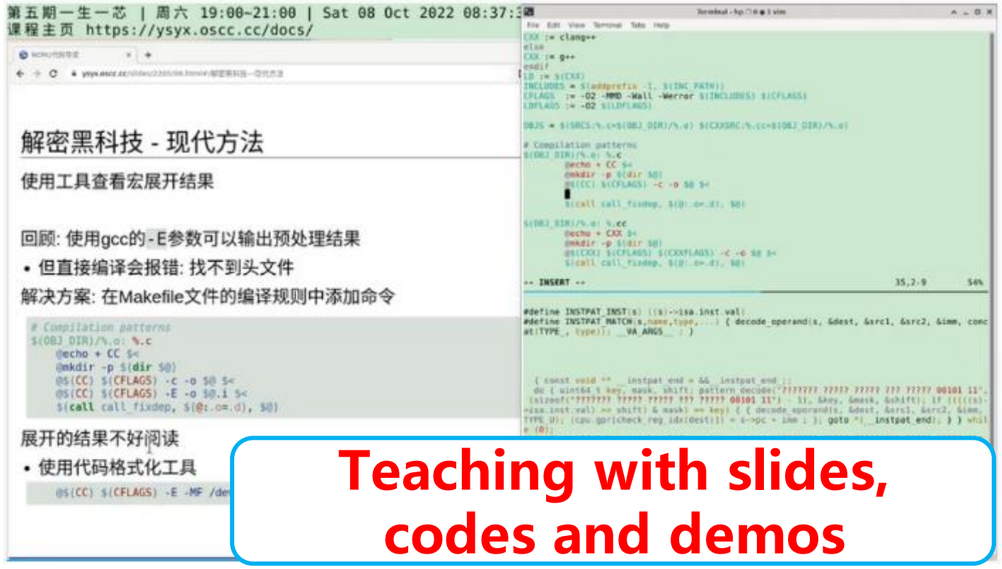
When you read the PA0 handout and proceed to the section on getting the PA framework code, you will be prompted with a box asking you to return to the content of the handout here.

First of all, please add a ssh key on github, please STFW on how to do that. Then get the framework code of "One Student One Chip" by the following command:

```
git clone -b master git@github.com:OSCPU/ysyx-workbench.git
```

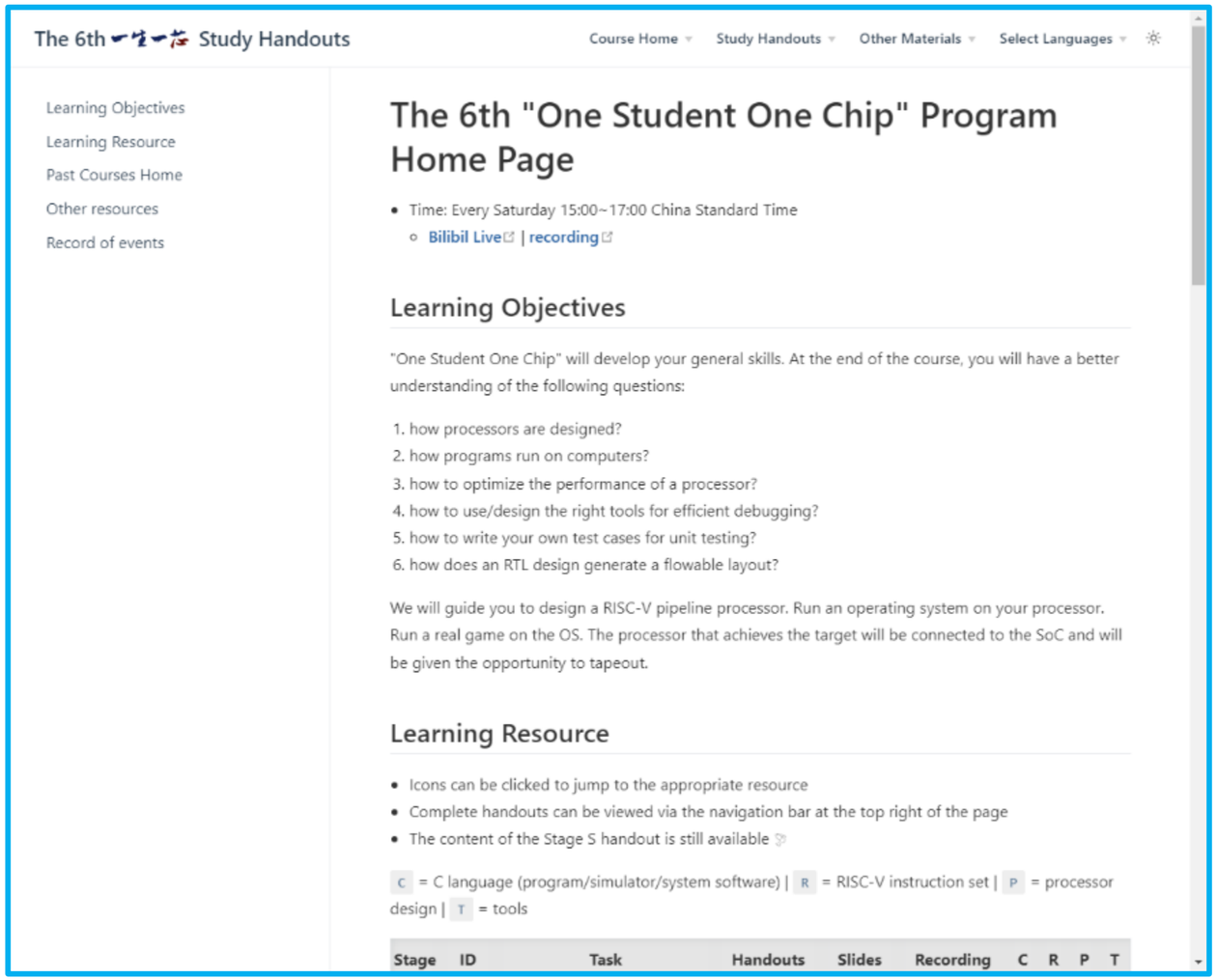
Once you have it, you can go back to the appropriate place in the PA handout and continue reading. But you should also note that:

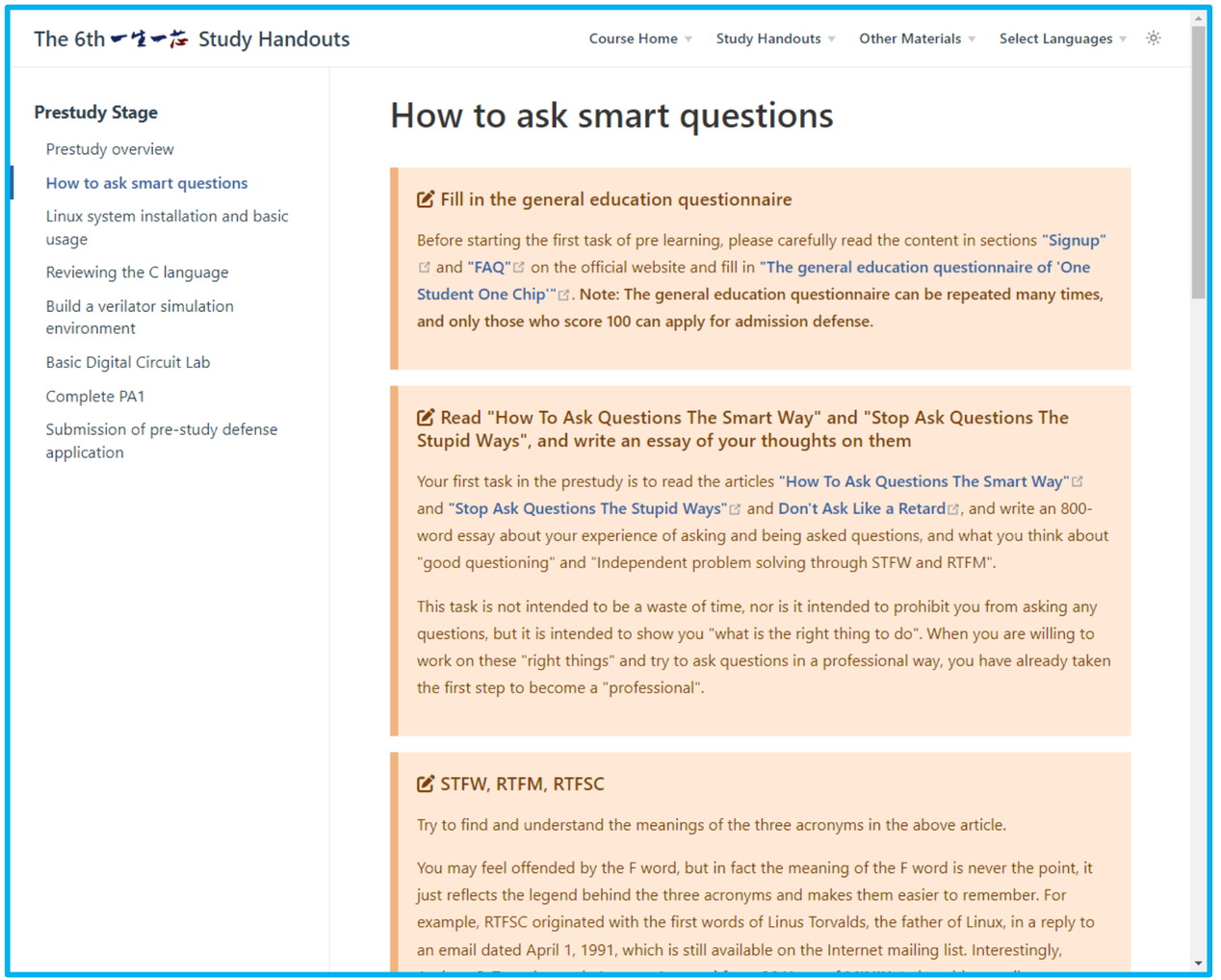
- Please use `ysyx-workbench` as the project directory in the PA handout, i.e. replace occurance of `ics2022` in the PA handout with `ysyx-workbench` .
- When change the student number and name in `ysyx-workbench/Makefile` , you can leave the student number unchanged until you have completed the preliminary.



Teaching with slides, codes and demos

English Version of Handouts





- **English version of slides and videos are WIP**
  Students can still learn without them

Vision: "Open-source" reshaping chip design

- Our ultimate vision is to innovate chip design methodology through the open-source concept, achieving the goal of "**designing open-source chips using open-source EDA tools and IPs**."

- **Step 1: Open-source SoC — In 3-5 years, provide the community with high-quality, tape-out verified RISC-V open-source cores and open-source SoC designs**
  - **including RISC-V processor core IP, peripheral IP, and more**
- **Step 2: Build Open-source SoC with Open-source Toolchain — Over the next 5-7 years, gradually establish an open-source SoC chip design process based on open-source EDA toolchains, open-source IP, and open-source process libraries**
  - **Commercial tools and IP will be gradually replaced with open-source versions**
  - **Undergraduate students will use open-source tools to develop open-source chips and graduate with their own chips**
- **Step 3: Automate Open-source Hardware Construction with Open-source Toolchain — Over the next 10-15 years, develop smarter and more automated open-source tools to improve design verification efficiency**
  - **Form an open-source chip design ecosystem and lower the barriers to chip development.**

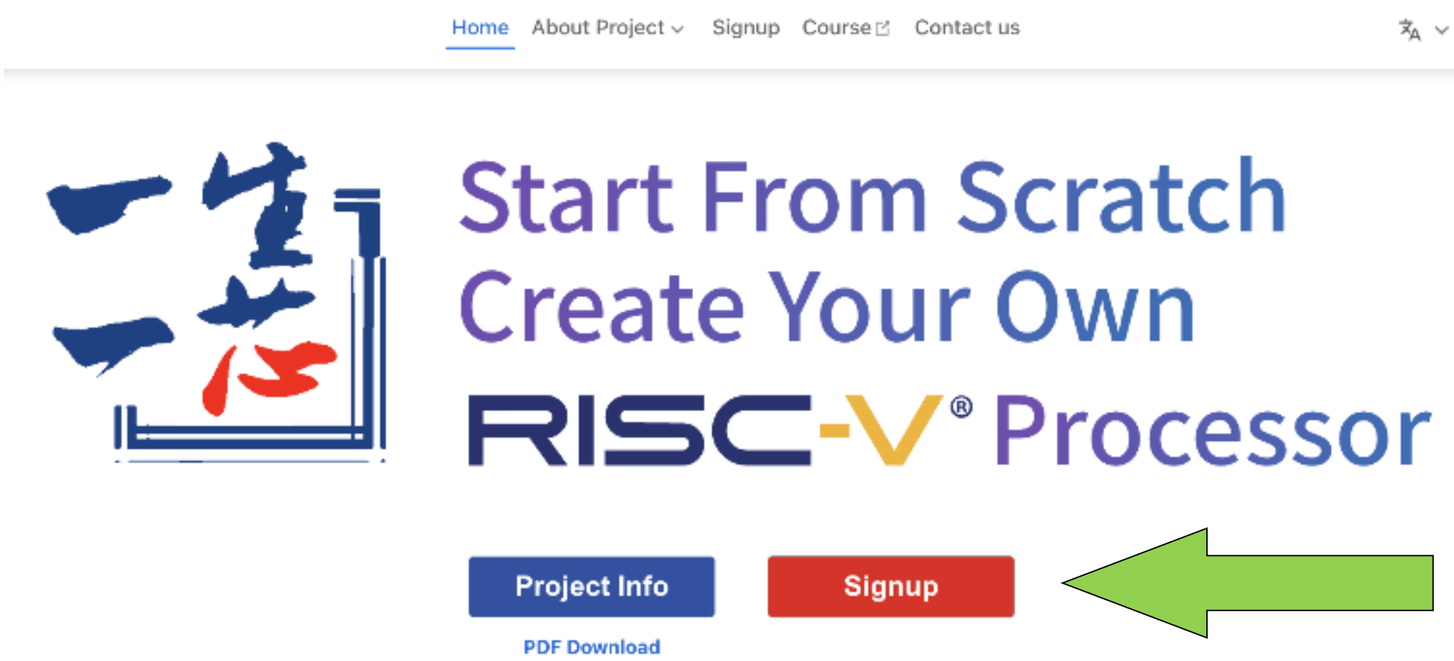
1st step, 2025:
a handheld game console

2nd step, 2026~2028:
a tablet that supports android and online games/videos (with Xiangshan and embedded GPUs)

3rd step, 2028~2030:
a laptop that can run ubuntu-desktop and office software smoothly (with Xiangshan)

Design chips and perform tape-out verification using open-source EDA, open-source IP, and open-source system software, to build functional prototype systems.

Thank you!



Website
ysyx.org/en/

Enroll Anytime, Open Year-Round
Visit ysyx.org/en/ and click "Signup"