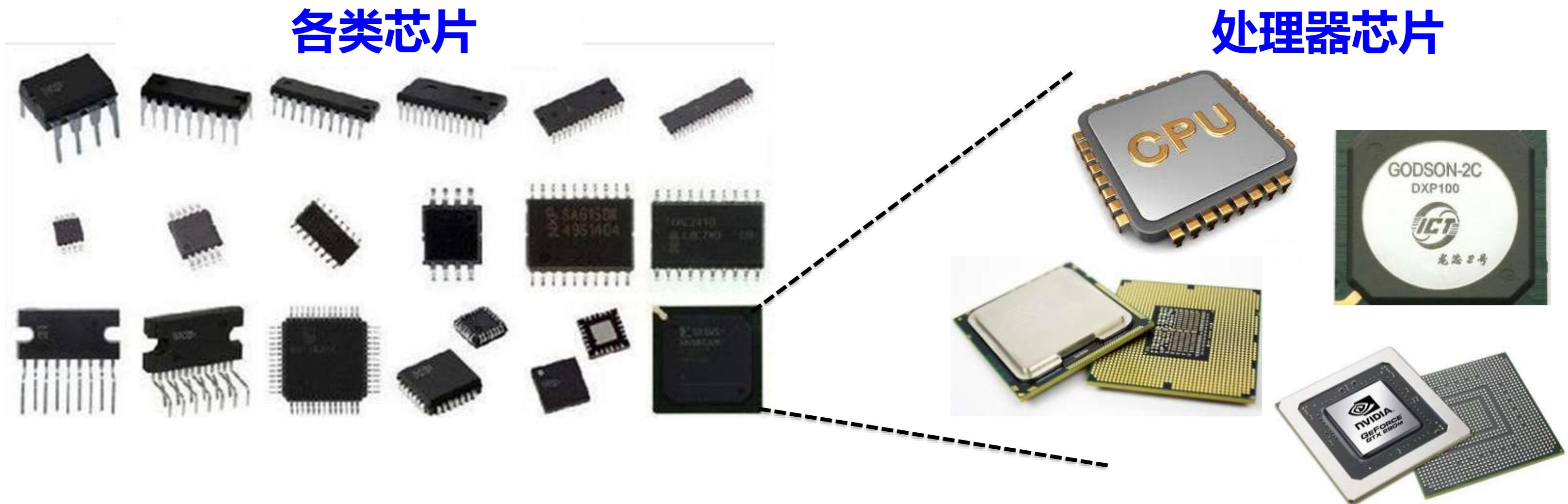




# 一、“一生一芯”计划初衷

# 处理器芯片是最为复杂的一类芯片

- 芯片是指内含**集成电路**的硅片，是**半导体产业**的核心
- 芯片有**几十种**大门类，**上千种**小门类
- **处理器芯片**被公认为**芯片产业皇冠上的明珠**，设计复杂度高、难度大



# 集成电路是中国第一大进口产品（2021年）

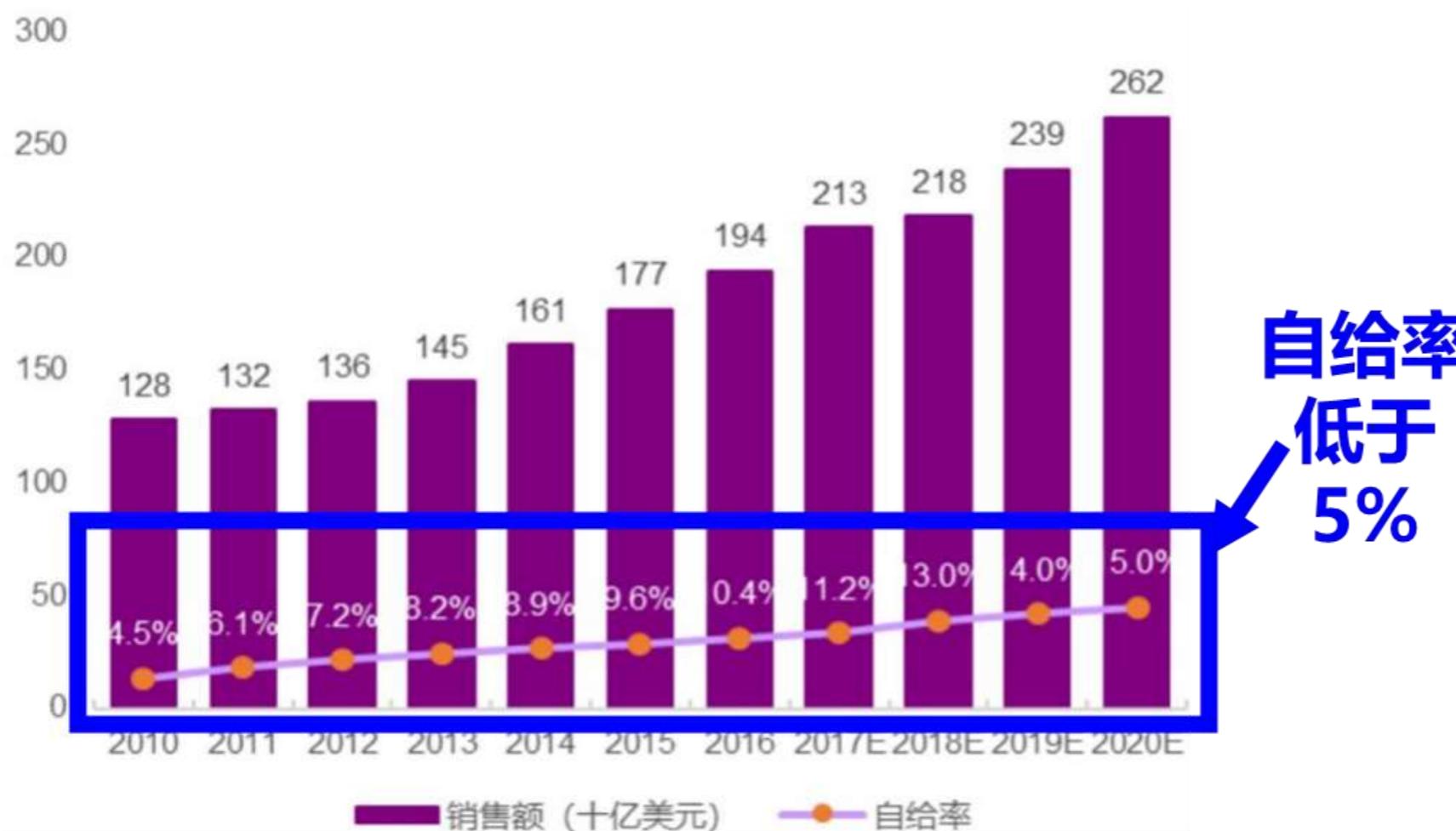
- 2021年进口**6355亿**个集成电路，进口额高达**4326亿美元**，远超过石油（**2573亿美元**）、铁矿（**1847亿美元**）
- 处理器与控制器芯片进口额超**2034亿美元**，占比高达**47%**

种类	进口金额 (亿美元)	占比
处理器及控制器	2034	47.0%
存储器	1220	28.2%
放大器	156	3.6%
其他	916	21.2%
<b>总数</b>	<b>4326</b>	<b>100%</b>

数据来源：中国海关

# 国产芯片任重道远

- 在很多领域，国产芯片市场占有率仍不足5%

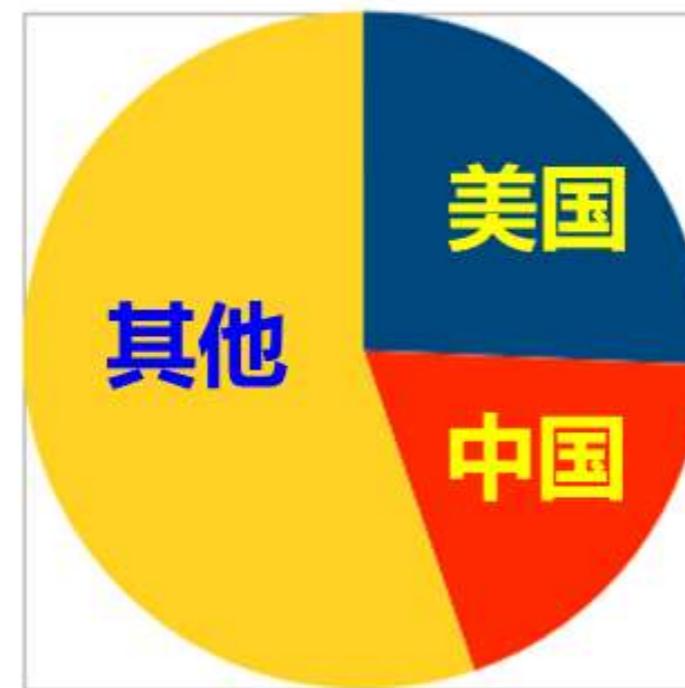


系统	设备	核心集成电路	国产芯片占有率
计算机系统	服务器	MPU	0%
	个人电脑	MPU	0%
	工业应用	MCU	2%
通用电子系统	可编辑逻辑设备	FPGA/EPLD	0%
	数字信号处理设备	DSP	0%
通信装备	移动通信终端	Application Processor	18%
		Communication Processor	22%
		Embedded MPU	0%
	核心网络设备	Embedded DSP	0%
存储设备	半导体存储器	NPU	15%
		DRAM	0%
		NAND Flash	0%
显示及视频系统	高清电视/智能电视	NOR Flash	5%
		Image Processor	5%
	Display Driver	Display Driver	0%

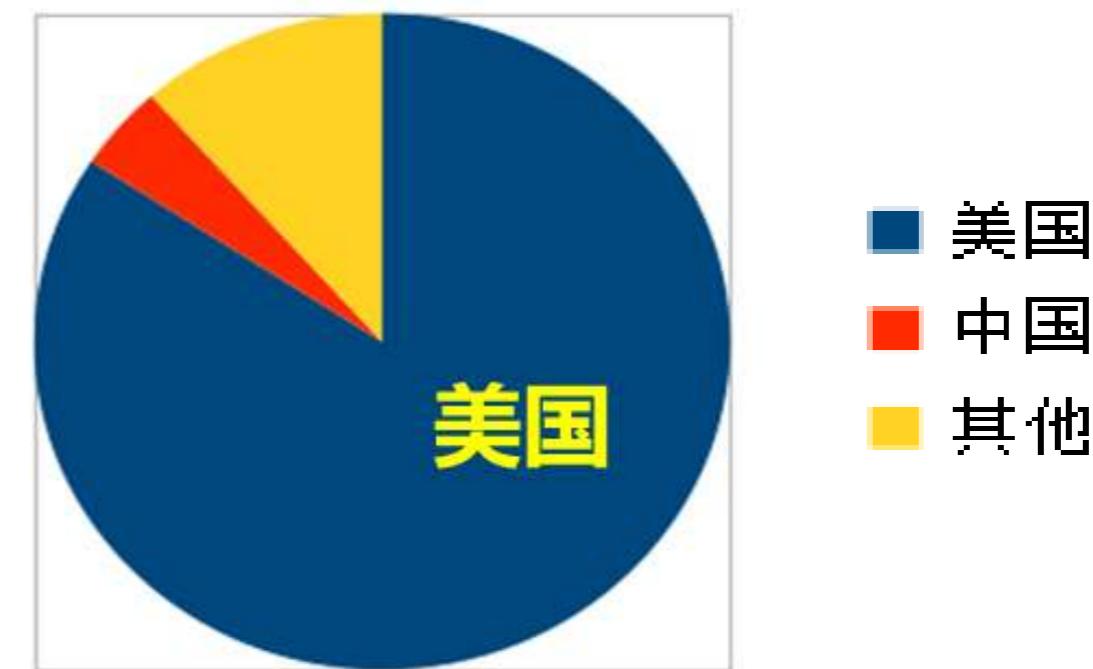
# 我国优秀处理器芯片人才储备严重不足

- **85% v.s. 4%**: 2008-2017十年体系结构国际顶级会议ISCA论文的第一作者**85%在美国**, 仅有**4%在中国**, 不足美国的**1/20**, 差距巨大
- 中国**加快处理器芯片人才培养规模与速度**, 迫在眉睫

一作国籍



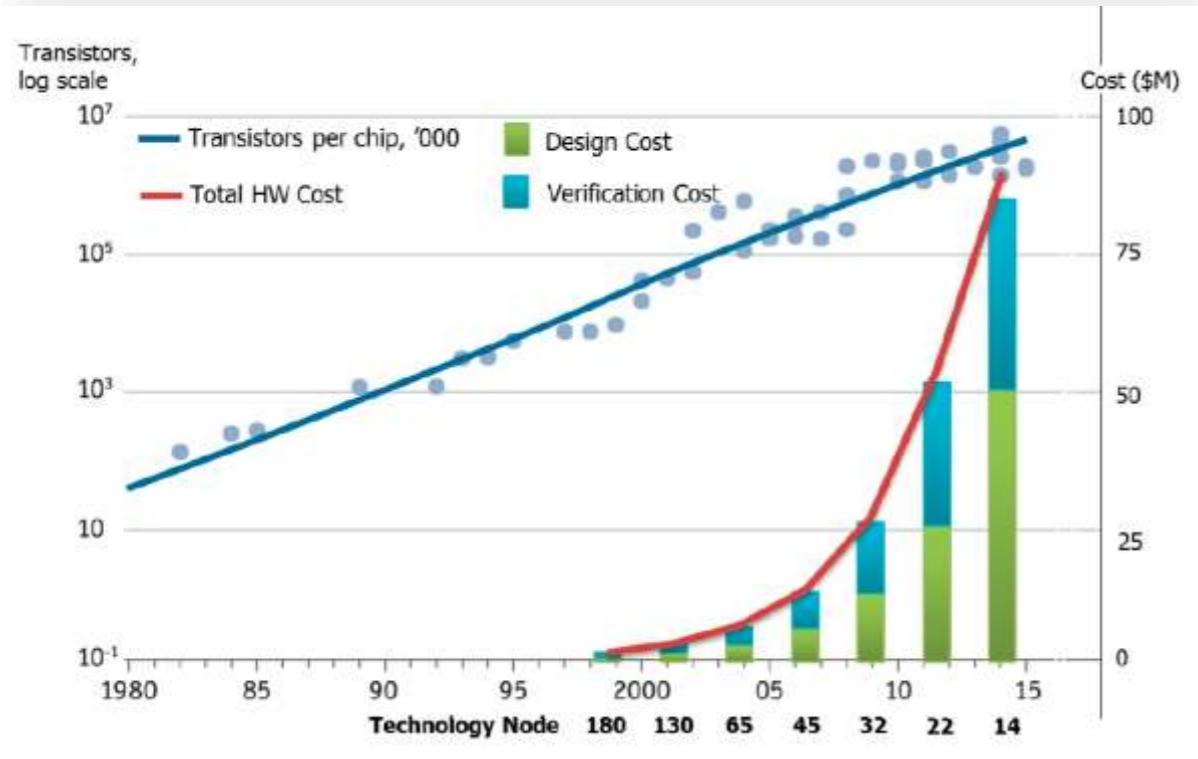
毕业去向



ISCA十年论文第一作者统计情况(2008-2017)

# 处理器芯片设计人才培养面临的挑战

## 芯片设计门槛高



Source: Andreas Olofsson, Intelligent Design of Electronic Assets (IDEA), 2017

## 传统专业/课程间关联弱



Image Source: <https://dzone.com/articles/how-does-the-asic-design-flow-cycle-works>

## 尚缺少类似临床实习环节



写RTL

学生  
预期

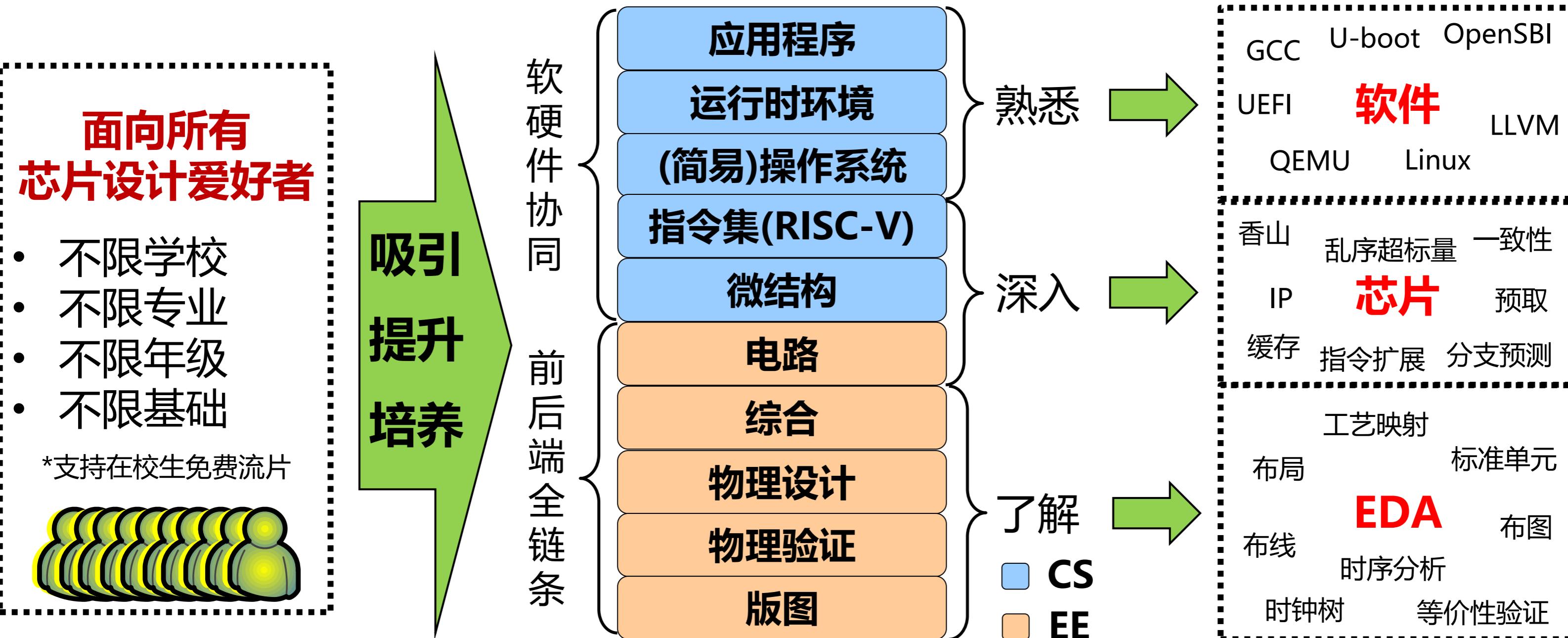
- 应用程序
- 运行时环境
- 操作系统
- 指令集
- 微结构
- 电路
- 综合
- 物理设计
- 物理验证
- 版图

实际是个  
系统工程

## 学习坡度高

# “一生一芯”计划总体介绍

基于开源新赛道的一种贯通课程的实践型开放式大规模人才培养计划



打破教育资源  
不平衡的壁垒

突破传统课程的边界，  
融合EE和CS的全栈人才培养

培养后进入开源社区/  
企业，攻关卡脖子领域

# “一生一芯”计划执行情况

- 自2019年起已历六期，**循序渐进，逐步放大规模，保持教学质量**
- 数据统计时间: 2023年8月19日

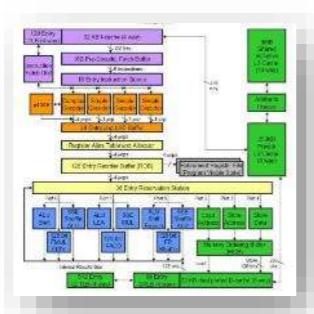
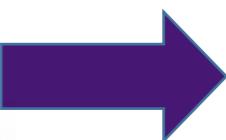
感兴趣      投入学习

期数	启动时间	报名结束时间	报名人数	覆盖高校	持续学习人数	达成流片指标人数
一	2019年8月	-	5	1	5	5
二	2020年8月	-	11	5	11	11
三	2021年7月	2021年9月	760	168	215	51
四	2022年2月	2022年8月	1753	328	215	18(学习指标提升)
五	2022年8月	2023年7月	1881	379	155	10(持续增加中)
六	2023年7月	未结束	821	190	37	未进行至流片阶段
七	预计2024年1月启动					

## 二、处理器芯片设计流程

# 处理器芯片设计流程

## ● 从指令集到流片版图

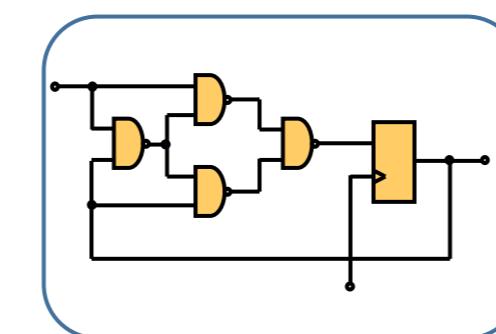


体系结构设计

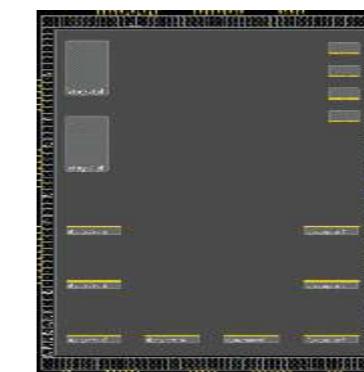
```
module conv;
reg [31:0] m[0:8192];
reg [12:0] pc;
reg [31:0] acc;
reg[15:0] ir;

always
begin
    ir = m[pc];
    if(ir[15:13] == 3b'000)
        pc = m[ir[12:0]];
    else if (ir[15:13] ==
3'b010)
        acc = -m[ir[12:0]];
    ...
end
endmodule
```

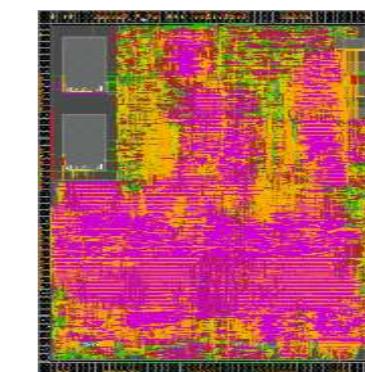
RTL代码



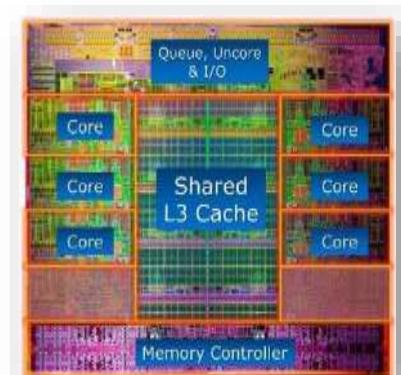
网表



布图规划



布局布线



版图

体系结构  
模拟器

体系结构探索

RTL功  
能仿真  
器

仿真验证

逻辑优  
化

工艺映  
射

形式化验  
证

可测性设计

测试生成

故障诊断

布图规  
划

全局布  
局

合法化

详细布  
局

时钟树综  
合

全局布  
线

详细布  
线

时序分  
析

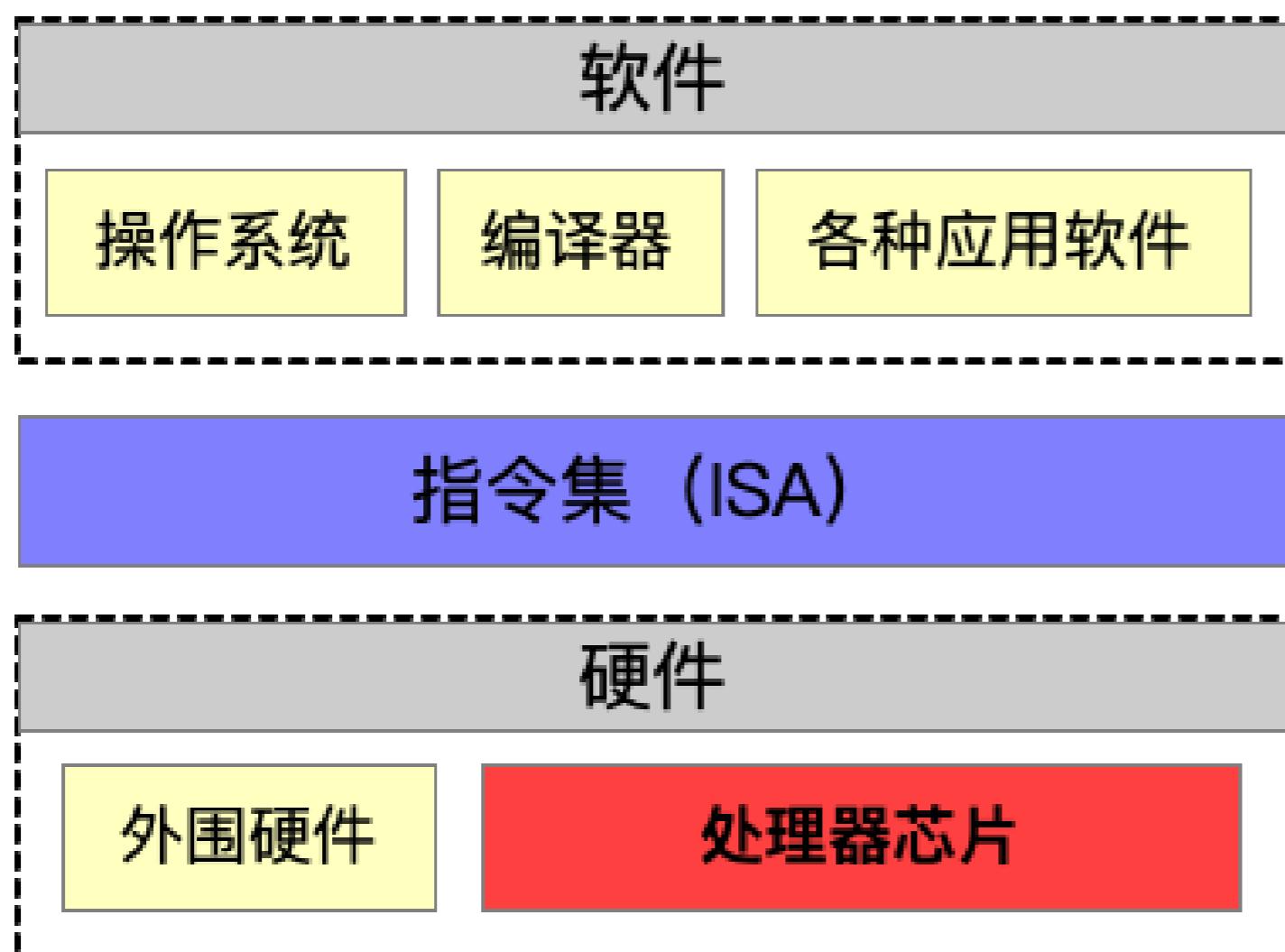
物理验  
证

功耗分  
析

签核分析

# 指令集 (ISA) 是一种规范标准

- 指令集架构 (Instruction Set Architecture)，简称指令集
- 计算机系统中硬件与软件之间交互的规范标准
- RISC-V是一种指令集



指令集可类比于螺母和  
螺钉的尺寸规范



# 指令集 → 微架构 → 版图 → 芯片产品

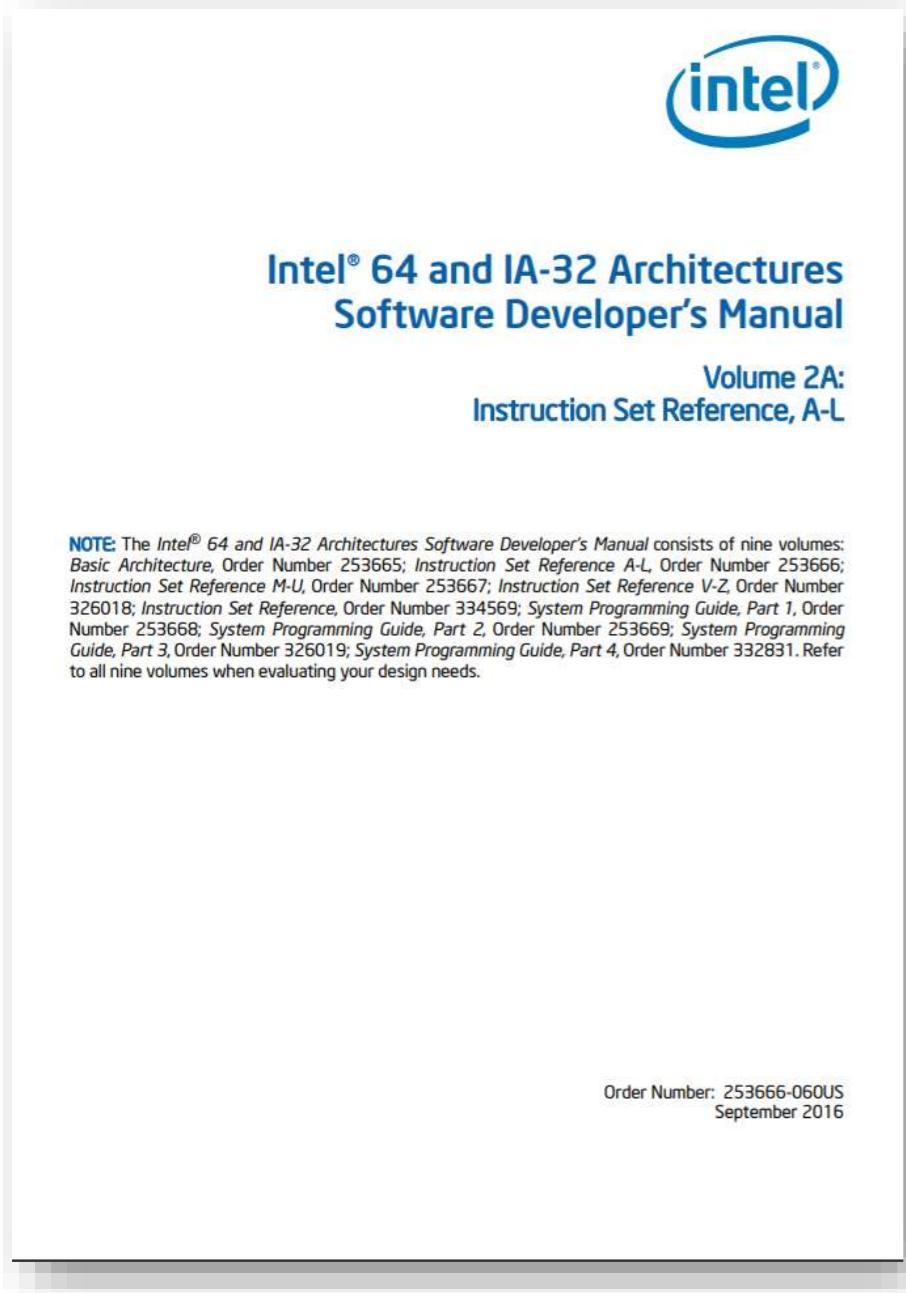
- 指令集是规范，定义指令的格式、功能

# x86 Opcode Structure and Instruction Overview

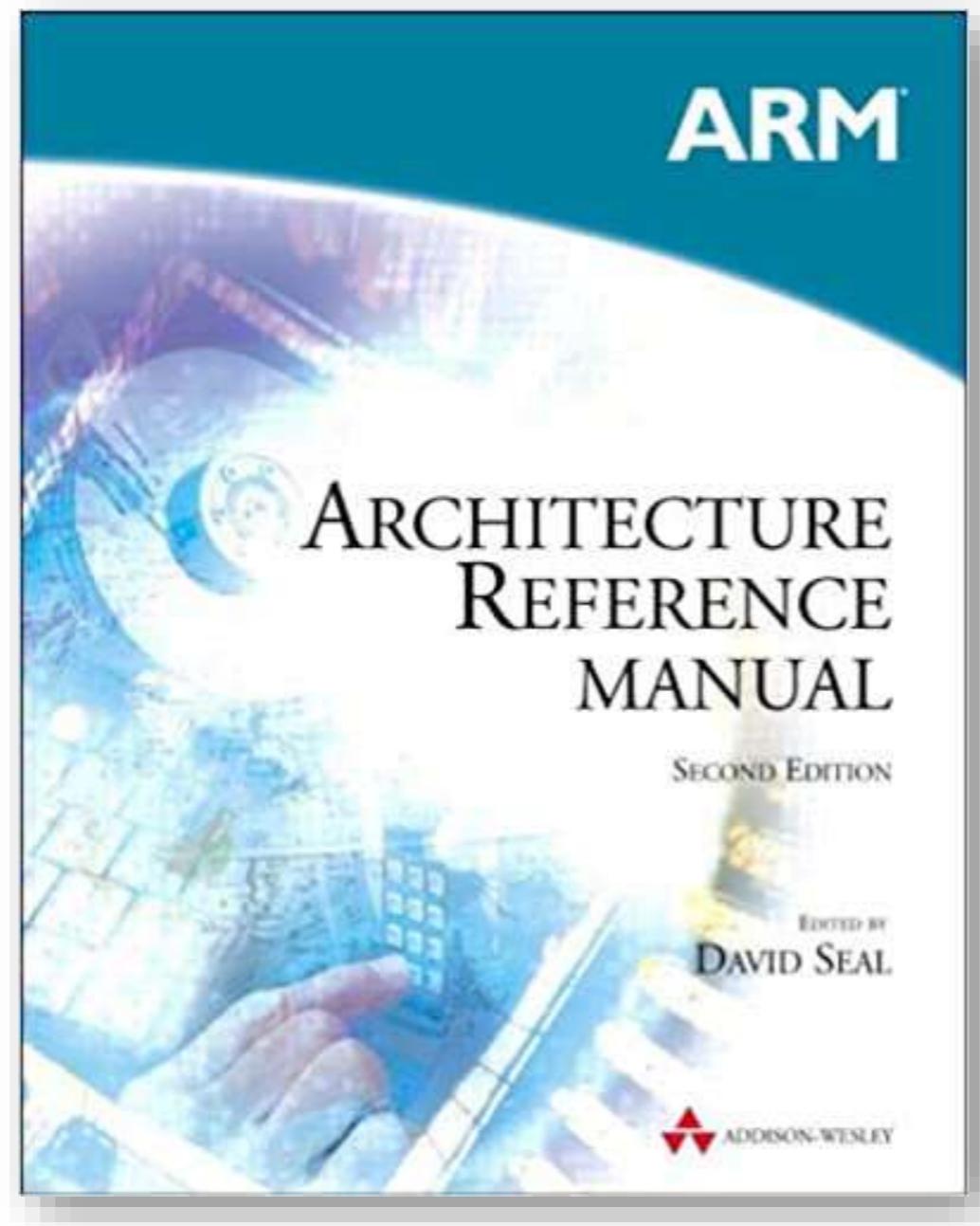
# 指令集 → 微架构 → 版图 → 芯片产品

- 指令集规范记录于**手册(Manual)**中

X86手册：约2200页



ARM手册：约2700页

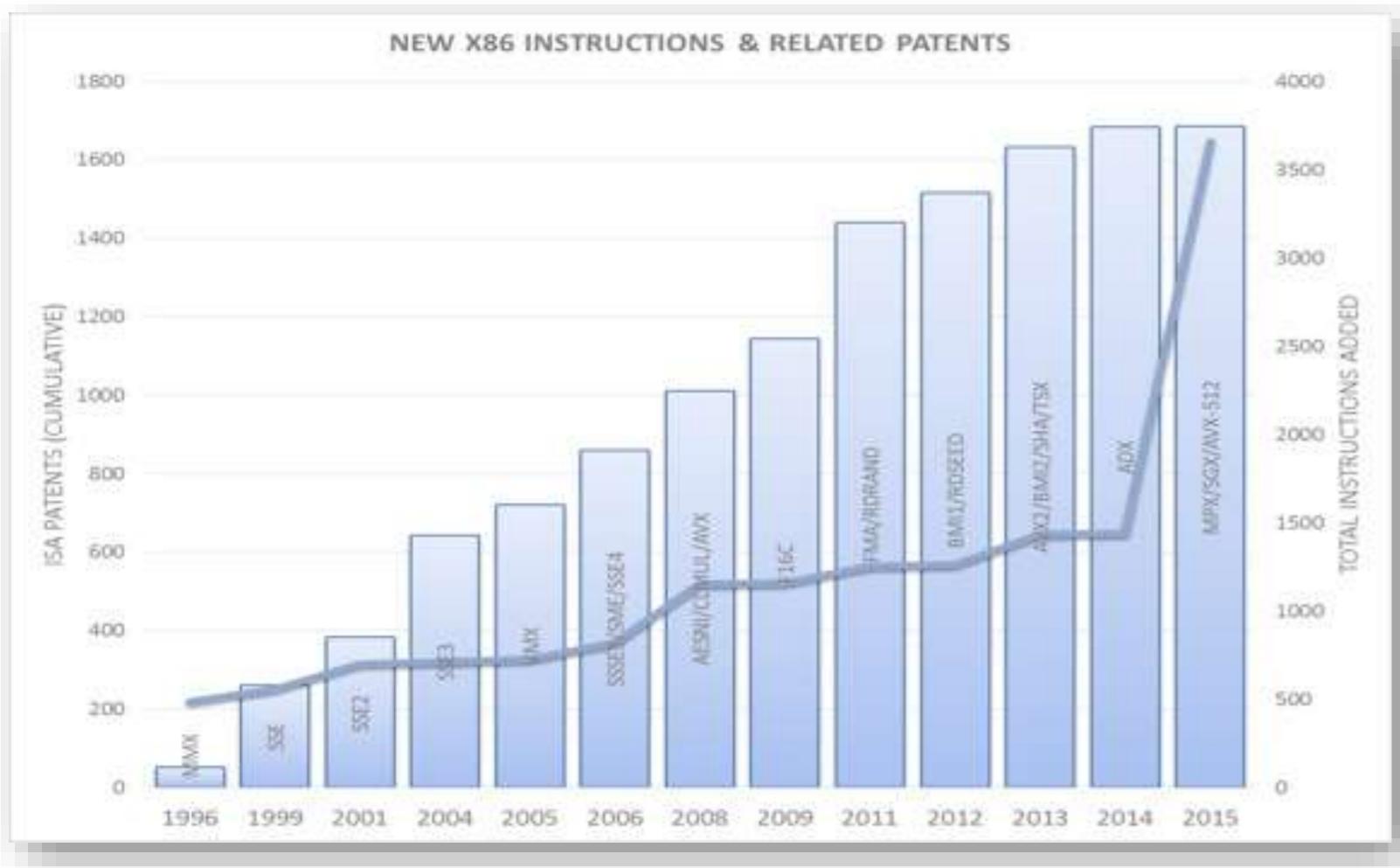


RISC-V手册：约200页



# RISC-V v.s. X86/ARM：指令更精巧

- X86从1978年的80条指令，增长到2015年的**3600条**
- RISC-V基础部分（RV32I）只有**47条**指令，并已冻结



# RISC-V v.s. X86/ARM：模块化

- 实现一个X86/ARM处理器，需实现所有上千条指令，复杂度极高
- RISC-V指令集采用模块化设计
  - 必要的RV32I只有47条指令
  - 其余指令可选扩展
- 可根据需求自由组合，灵活适配
  - 嵌入式(关注处理器成本) – RV32I
  - 嵌入式(关注存储器成本) – RV32IC
  - 教学 – RV32IMA
  - 桌面 – RV64GC
  - 高性能 – RV64GCV
  - 支持自定义指令

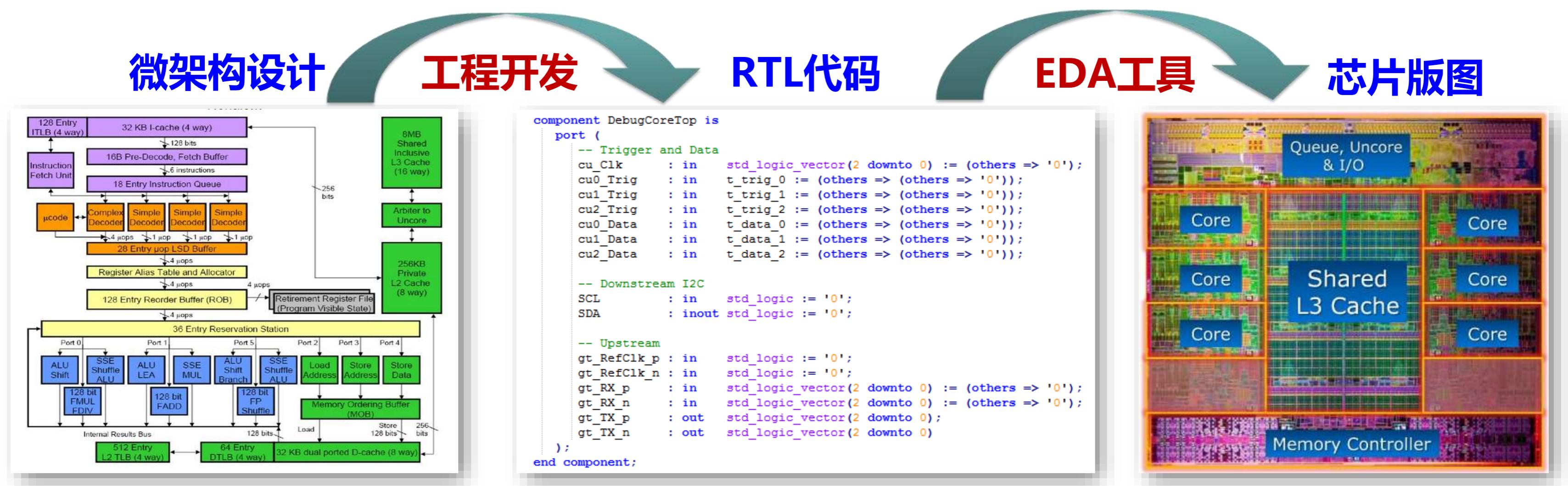
The table displays the RISC-V instruction set architecture, organized into several extension categories:

- Base Integer Instructions: RV32I and RV64I**: RV32I Base + RV64I. This section includes Shift Left Logical, Shift Right Logical, Arithmetic (ADD, ADDI, SUB, SUBI), Logical (XOR, OR, AND, ANDI), Compare (Set <, Set < Immediate, Set < Imm), Branches (JAL, JALR, BEQZ, BGEU), Stores (SW, SWP, FSWP, FSWSWP, FSD, FSDP), Loads (LW, LWSWP, FLW, FLWSWP, FLD, FLDSP), and Arithmetic/Logical instructions.
- Optional Compressed (16-bit) Instruction Extension: RV32C**: RV32C (Atomic). This section includes Load (Load Word, Load Word SP, Load Word SP), Store (Store Conditional, Store Word, Store Word SP, Store Double SP), Add (ADD, ADDI, ADD SP Imm \* 16, ADD SP Imm \* 4), Logical (AND, ANDI, XOR, OR, ORI), Min/Max (MINIMUM, MAXIMUM, MINIMUM Unsigned, MAXIMUM Unsigned), and Comparison (Set <, Set < Immediate, Set < Imm).
- Optional Multiply-Divide Instruction Extension: RVM**: RV32M (Multiply-Divide). This section includes Multiply (MULT, MULT High, MULT High Uns), Divide (DIV, DIV Unsigned), Remainder (REM, REM Unsigned), and Load/Store (VLD, VLDP, VLD Strided, VLDP Strided, VLD Indirect, VLDP Indirect).
- Optional Atomic Instruction Extension: RVA**: RV32A (Atomic). This section includes Load (Load Reserved, Load Unreserved), Store (Store Conditional, Store Unconditional), Swap (SWAP, SWAPD), Add (ADD, ADDD), Logical (XOR, AND, ANDD, OR, ORD), Min/Max (AMINUM, AMAX, AMINUMD, AMAXD), and Comparison (Set <, Set < Immediate, Set < Imm).
- Two Optional Floating Point Instruction Extensions: RVF/D**: RVF/D (SP, DP, Fl., Pt.). This section includes Move (FMV.W, FMV.X, FMV.D, FMV.E, FMV.W, FMV.X, FMV.D, FMV.E), Convert (FCVT.S->D, FCVT.D->S, FCVT.S->D, FCVT.D->S, FCVT.L->D, FCVT.D->L, FCVT.LU->D, FCVT.DU->L), and Comparison (FCMP.S, FCMP.D, FCMP.L, FCMP.DU).
- Calling Convention**: ABI Name Saver. This section defines register conventions for various calling contexts.

RISC-V常用的可选扩展，可适配从嵌入式到高性能的各种场景

# 指令集 → 微架构 → 版图 → 芯片产品

- 微架构 (microarchitecture)设计，就是将**指令集手册**定义的功能**实例化**，然后通过工程开发，变成**源代码**



# RISC-V v.s. X86/ARM: 开放免费

指令集与微架构均有  
三种知识产权模式

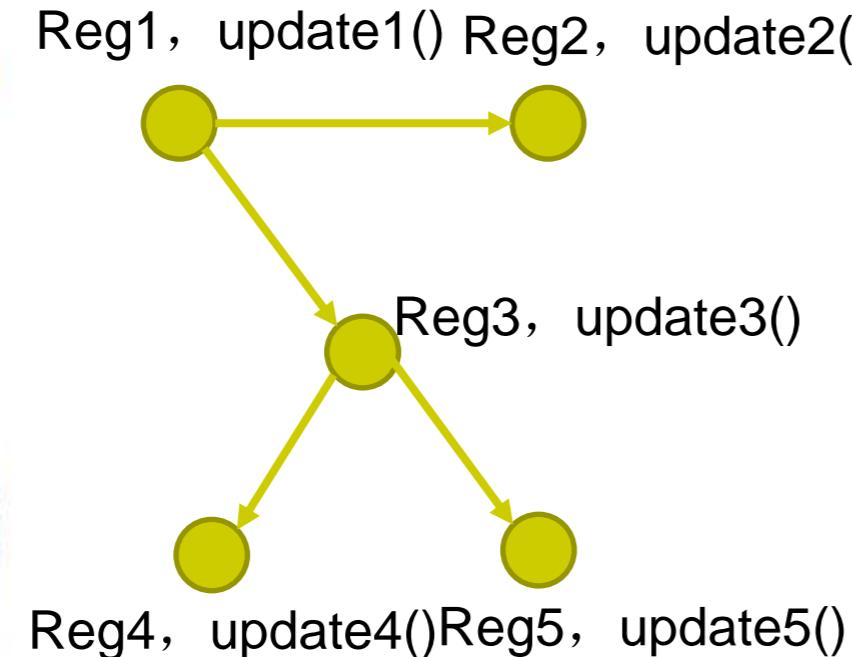
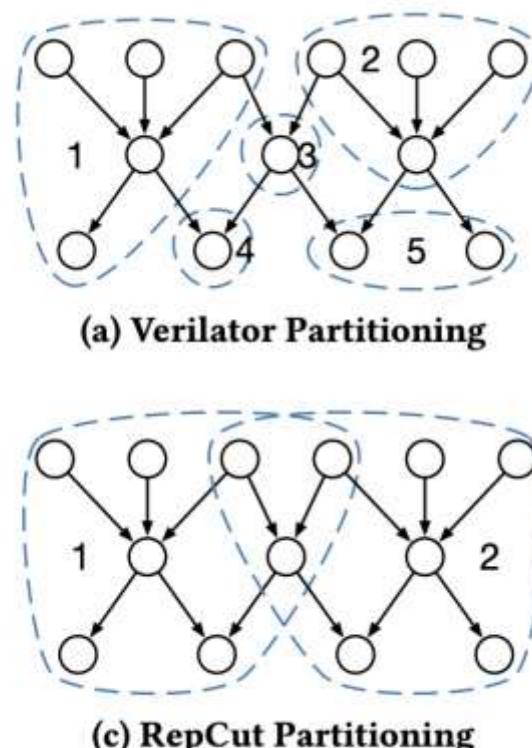
- 开放免费 (Open & Free)
- 可授权 (Licensable)
- 封闭 (Closed)

微架构设计 指令集	1 开放免费的设计	2 需授权的设计	3 封闭的设计	产品可选的设计 (对应各指令集)
开放免费的指令集 (RISC-V)	Berkeley的Rocket Chip/剑桥lowRISC/芯来科技蜂鸟E203	平头哥/SiFive/晶心科技Andes的RISC-V处理器核	Google和NVIDIA的自研RISC-V处理器	1 2 3
需授权的指令集 (ARM)		ARM的处理器设计,如Cortex-A76等	基于ARM架构的Apple处理器	2 3
封闭的指令集 (x86)			Intel和AMD的处理器	3

# 指令集 → 微架构 → 版图 → 芯片产品

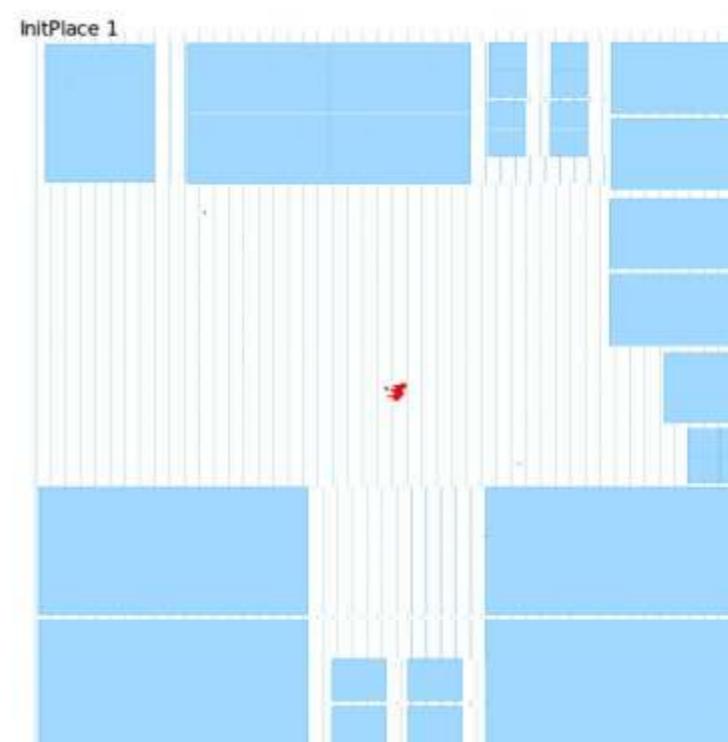
- 通过EDA工具将源代码生成版图

- 仿真验证：在一个超大的电路图（Graph）中，进行值传播，每个cycle 更新一次Reg值
- 综合：把一个亿门级电路，**转化为与非门等布尔逻辑表达**，实现**门数最少或面积时序最优**等目标
- 布局：在**不到1cm<sup>2</sup>的平面空间内**，摆放**数亿个长方形单元**（需同时考虑其拓扑连接关系等）
- 时钟树综合：构建一颗时钟树，从单一时钟源出发，使其**到达数亿单元的时延等长**
- 布线：在多层绕线空间内，解决**数亿单元的拓扑连线**问题，并实现**线长最短**
- 时序分析：把数亿条连线关系，**建图并找出时延最长的路径**

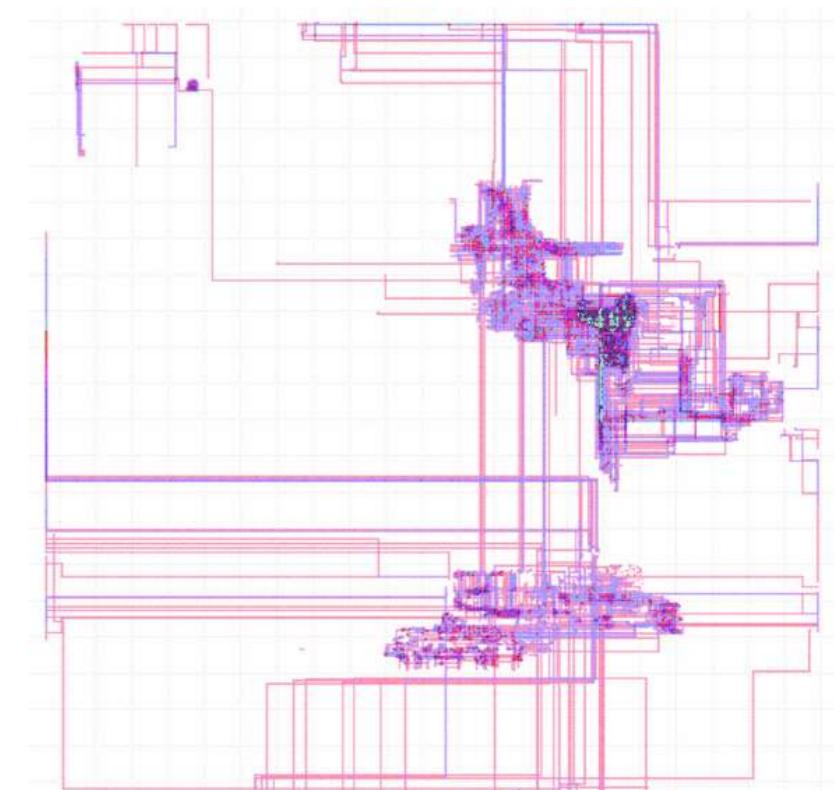


仿真验证中的图计算（同时压迫 icache和dcache, stall很多）

RepCut: Superlinear Parallel RTL Simulation with Replication-Aided Partitioning (ASPLOS 2023)



“果壳-1”芯片的Replace布局  
By 陈仕健

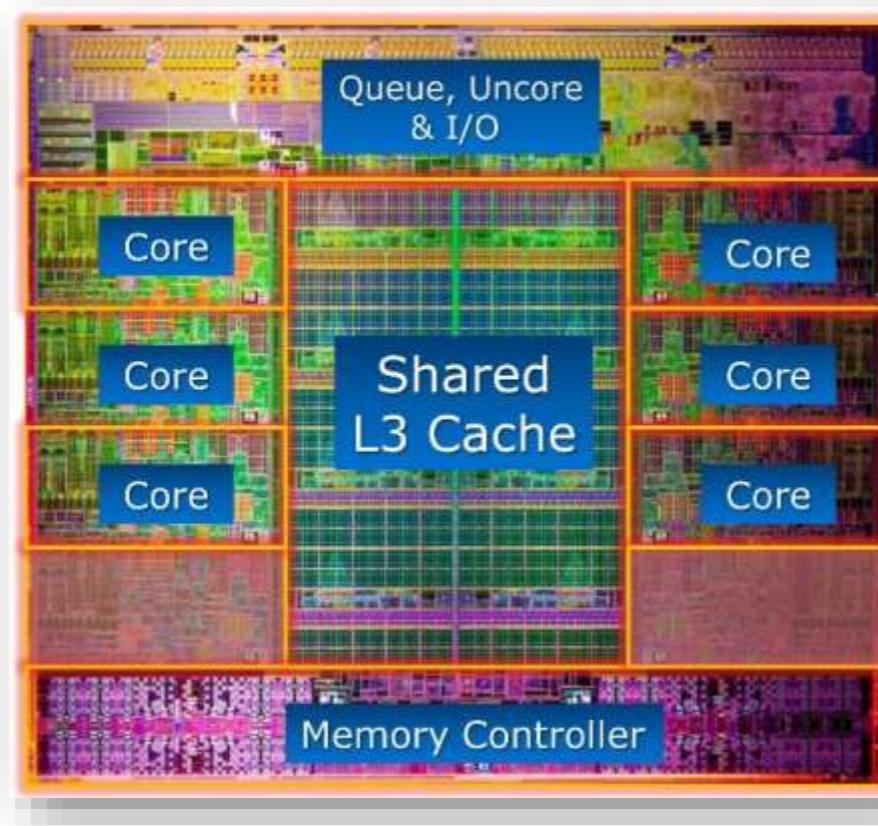


“果壳-1”芯片的iGR布线  
By 曾智圣

# 指令集 → 微架构 → 版图 → 芯片产品

- 将版图提交给台积电/中芯国际等企业**流片**，然后由长电等企业完成**封装**，获得芯片

芯片版图



流片封装

芯片产品



### **三、本科生带着自己的芯片毕业**

**首期“一生一芯”计划**  
**(2019年8月~12月)**

# 首期“一生一芯”计划

## ——让学生带着自己设计的处理器芯片毕业

- 2019年8月底，在国内**首次以流片为目标**，由**5位**2016级计算机学院本科生主导完成一款64位**RISC-V处理器SoC芯片**设计，于12月19日**流片**
- 芯片成功**运行Linux操作系统与国科大教学操作系统UCAS-Core**



金越



王华强



王凯帆

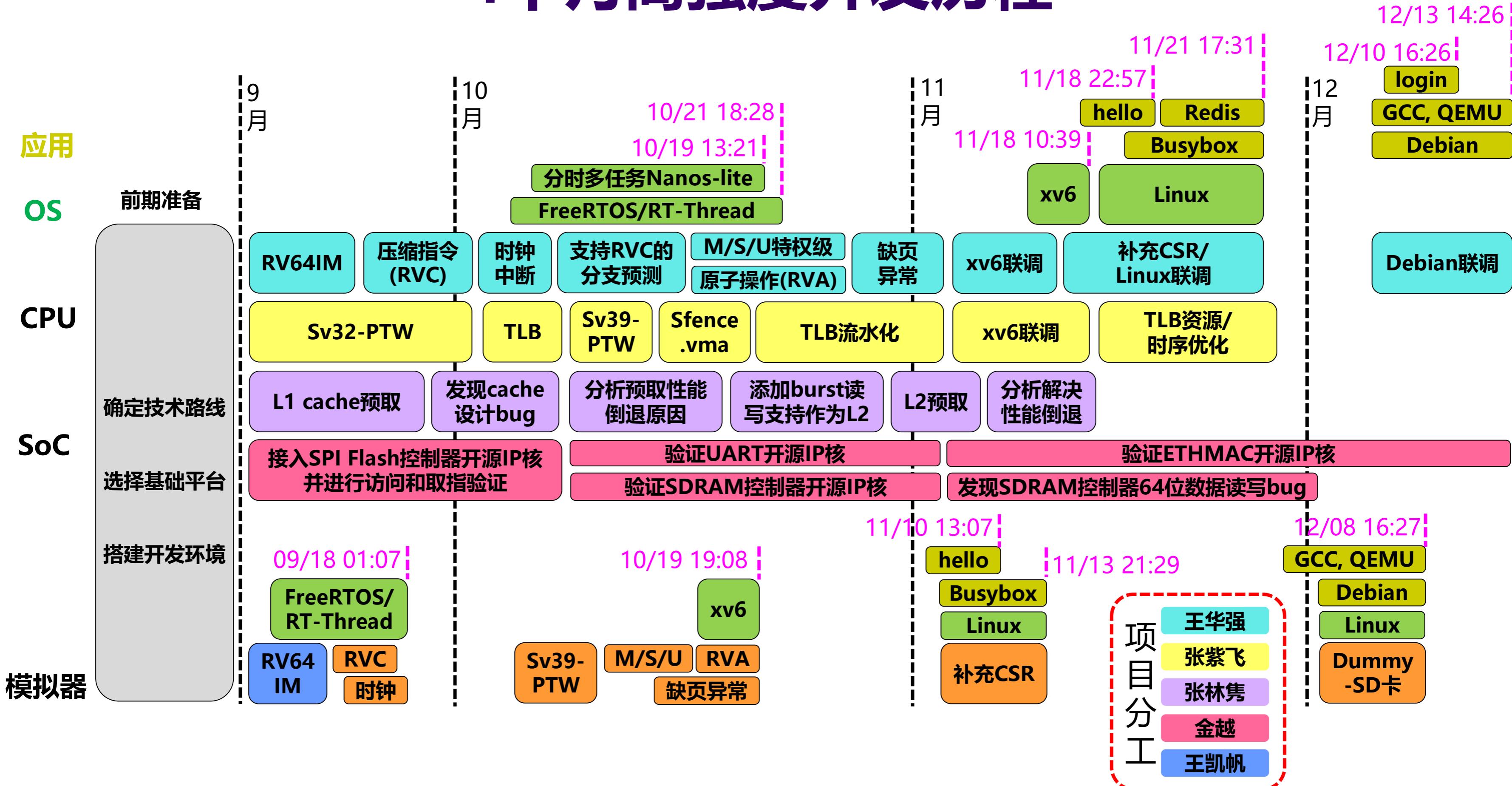


张林隽



张紫飞

# 4个月高强度发展历程

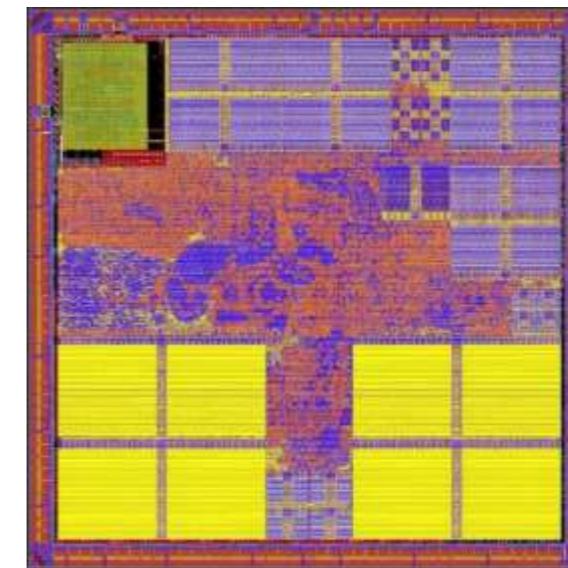


# 成果：果壳 (NutShell)

一款功能完整的RISC-V处理器

- 单发射9级流水顺序核
- RV64IMAC指令集，支持M/S/U特权级
- 两位饱和计数器分支预测，512项BTB，16项RAS
- 支持Sv39分页机制，支持硬件TLB填充
- 32K指令/数据L1 cache
- 支持L1 指令/数据cache读一致性
- 128K L2 cache，支持next line预取
- 使用Chisel语言开发
- 支持SDRAM、SPI flash、UART等外设
- 支持启动Linux 4.18.0内核
- 支持运行Busybox套件
- 在仿真/FPGA环境下支持启动Debian 11

基于国产110nm工艺完成流片



- 110nm工艺
- 10mm<sup>2</sup>
- 200mw@350MHz Typical
- TQFP100封装

# 学生感悟

## 依赖指导 => 主动探索

与之前实验最大的不同.....就是**没有先行者一步一步的详细指导**，而是要**自己寻找方法，独立实现**，然后进行验证甚至推倒重来。

## 使用者 => 创造者

胡伟武老师曾经说过，我们计算机系的同学应该学会怎么造计算机而不是怎么用计算机。我以前对这句话并不太有感触，相反曾经质疑国科大计算机系的课程设置这么多硬件的内容是否合理。但**真正参与到项目中才发现在大学里所学的知识和技能是真的有用**。

大部分知识在体系结构课程中...**工作原理也很简单**，只有短短的几行，但是**真正在代码中实现却比自己所想象的要困难得多**。

## 更自信、更有耐心

和4个月之前的自己相比.....**最重要的就是这种观念上的转变**。遇到bug不再在一个地方上死磕，而是从心理上告诉自己bug都是人写出来的，**只要有耐心，只要挖得足够深就一定能找到问题所在**。

## 成就感

真正参与到项目中才知道课程作业就像直接给人采摘的果园一样，但项目却是**给一片荒地和几颗果树苗，从开垦种植和施肥都要自己动手**，并且还不知道这样能不能结出果实。不知为何，总觉得**从0开始种出的果实要更甜一些**。

**提升专业知识，锤炼心理素质**

# 首期“一生一芯”计划取得成功

- 五位同学实现“**带着自己设计的处理器芯片毕业**”这一目标



# “果壳”设计被RISC-V Global Forum接收

## *NutShell: A Linux-Compatible RISC-V Processor Designed by Undergraduates*



# 向国际社区开源，已有近百位使用者

- 果壳已在GitHub开源
  - 超过200个Fork
- 典型案例：中科院软件所已将华为OpenEuler操作系统移植到果壳上，并建立果壳分支
  - 工业级操作系统成功应用在教学芯片上

Fork 204 Star 1.2k

wakafat Merge pull request #31 from jwjtjwj/wj/w/improve-settings-print 4bbff6f1 17 hours ago 1,293 commits

debug doc: revise documents for nutshell last month

fpga,fpga;doc: refine doc about standalone(pynq) 9 days ago

project update sbt version 2 years ago

ready-to-run ready-to-run: update nemu-so with latest version 11 days ago

scripts cache: use SyncReadMem 12 months ago

src TopMain.scala: print 'long int' in hex 17 hours ago

tools/readmemh tools: add generator to generate readmemh files for loadMemoryFromFile 2 years ago

.gitignore Merge branch 'master' into pynq-standalone-argo last month

LICENSE license: add copyright owner UCAS 24 days ago

Makefile makefile: set inorder-sim as default config 10 days ago

README.md readme: set pynq as default board in readme 11 days ago

build.sbt

build.sc

debian\_on\_fpga.gif

devlog.md

scalastyle-config.xml

scalastyle-test-config.xml

Open Source Chip Project by University (OSCPU)  
Let students design their own chips!

<https://github.com/OSCPU/NutShell>

```
2.680000] clk: Not disabling unused clocks
2.690000] Waiting for root device /dev/mmcblk0p2...
2.760000] mmc0: new high speed SDHC card at address aaaa
2.810000] mmcblk0: mmc0:aaaa SS08G 7.40 GiB
2.840000] mmcblk0: p1 p2
2.980000] EXT4-fs (mmcblk0p2): mounted filesystem with ordered data n
2.990000] VFS: Mounted root (ext4 filesystem) on device 179:2.
3.010000] devtmpfs: mounted
3.020000] Freeing unused kernel memory: 176K
3.030000] This architecture does not have kernel memory protection.
4.300000] random: fast init done
6.550000] systemd[1]: System time before build time, advancing clock.
6.630000] systemd[1]: Failed to lookup module alias 'autofs4': Function not found
6.910000] systemd[1]: systemd v243-18 running in system mode. (+PAM +
+IDN2 -IDN +PCRE2 default-hierarchy=legacy)
6.940000] systemd[1]: Detected architecture riscv64.
```

Welcome to openEuler 20.03 (LTS)!

```
7.040000] systemd[1]: Set hostname to <openEuler-RISCV-rare>.
```

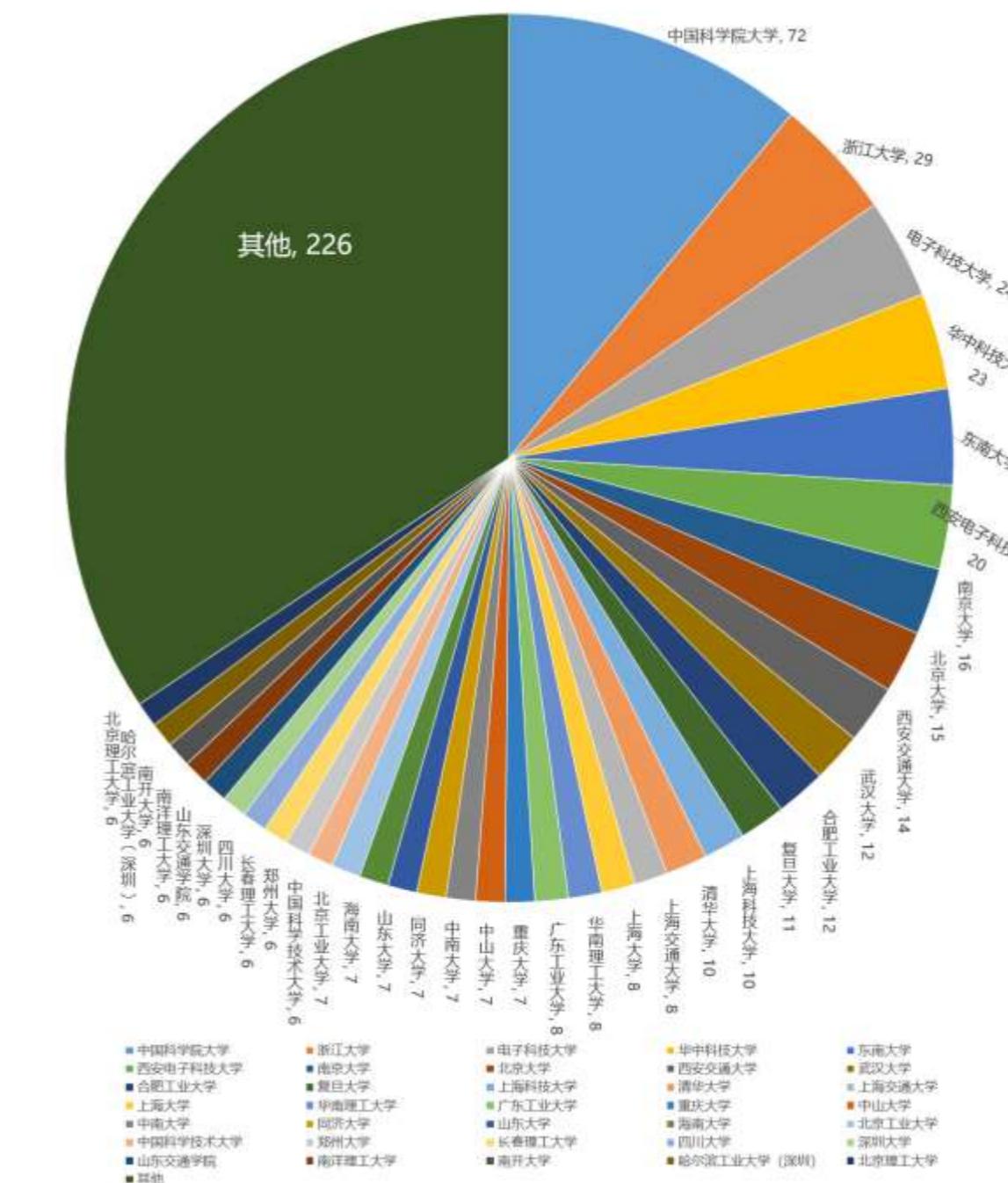
## 四、建设大规模人才培养流程

**第三期“一生一芯”计划**  
**(2021年7月~2021年12月)**

# 第三期报名情况简析

- 报名总数：760人

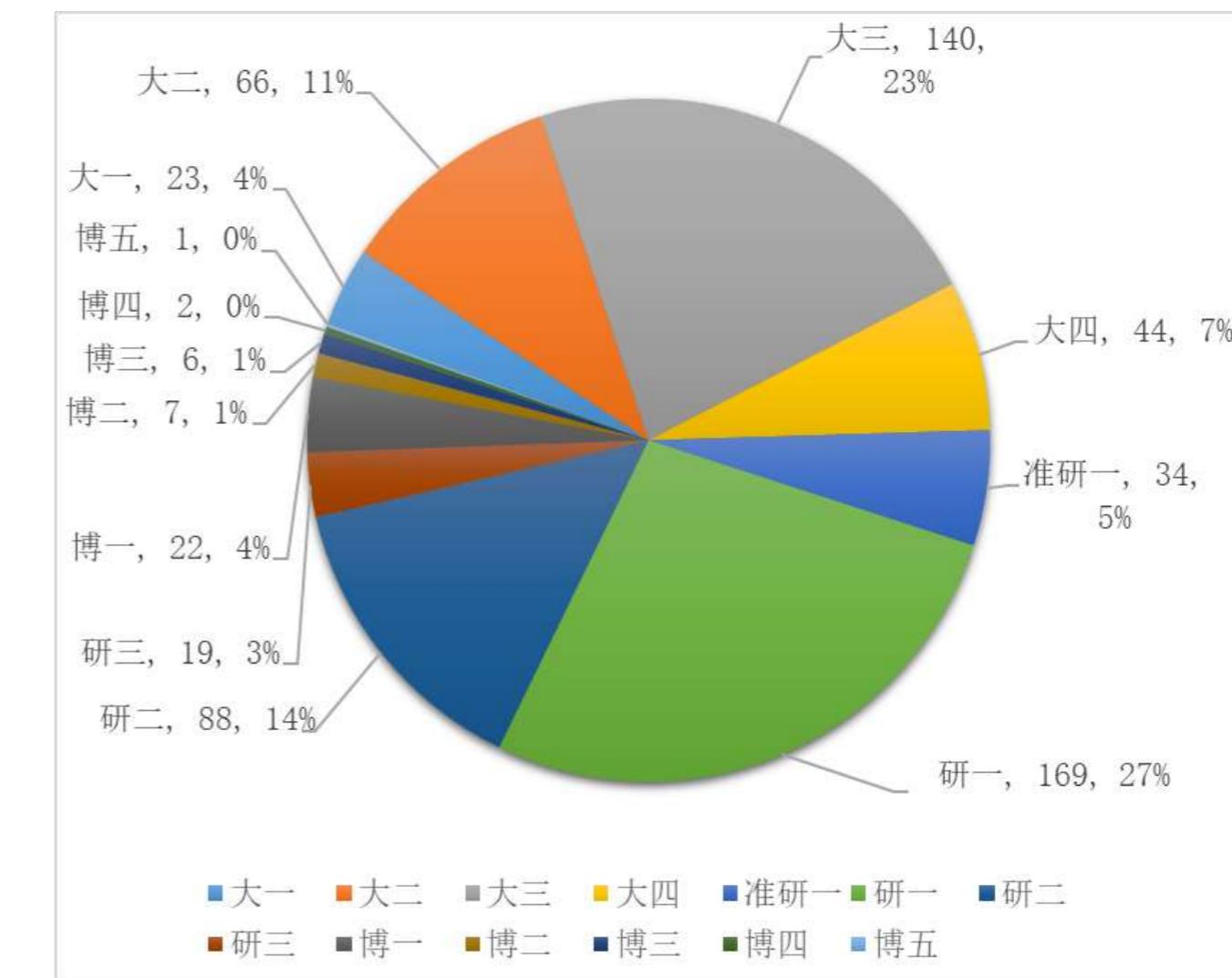
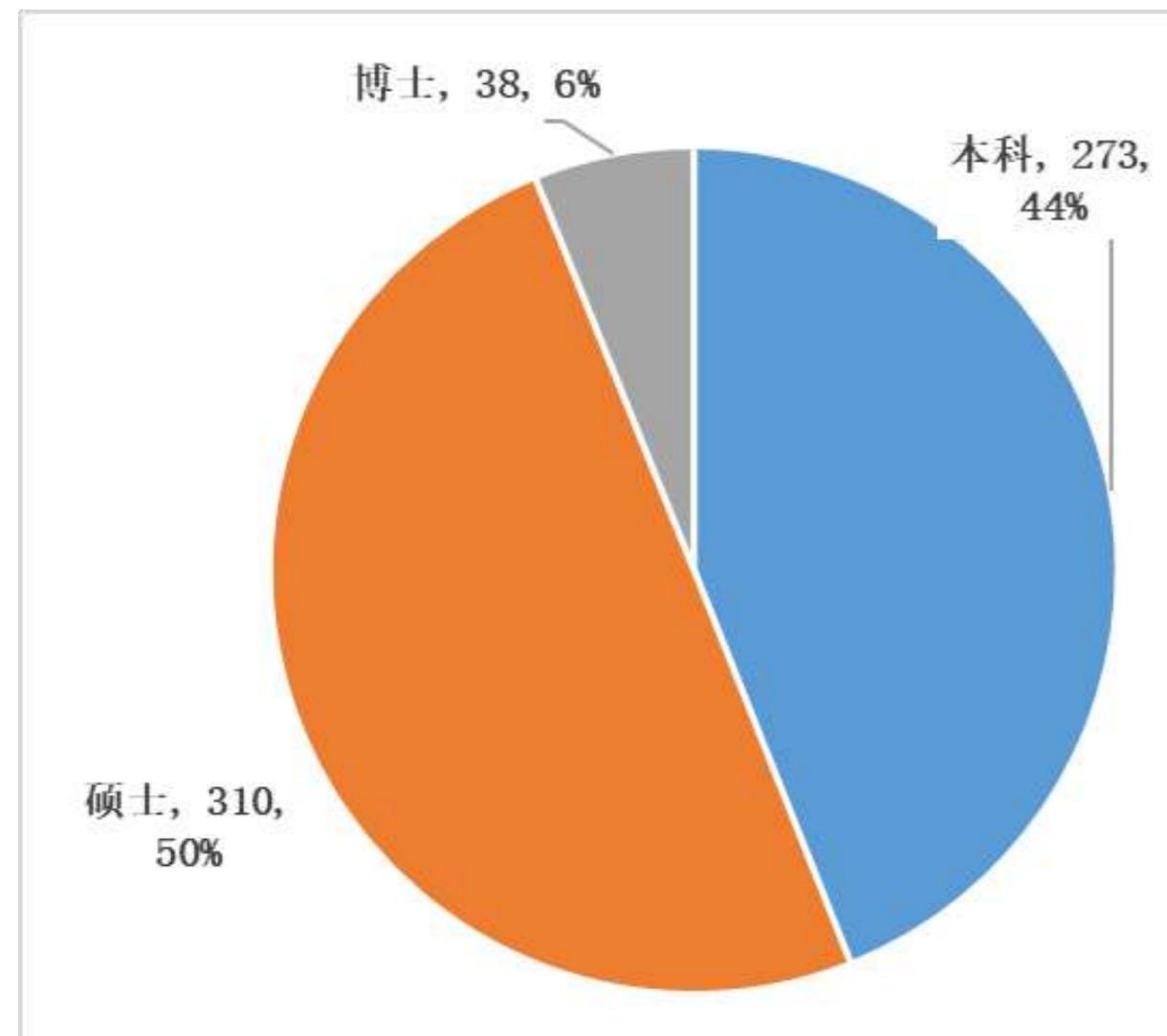
- 在校生是“一生一芯”的主力，共625人，占比82%；较多已毕业学生报名
- 覆盖168所高校（含国外30所）
- 国科大（72人）
- 浙江大学（29人）
- 电子科技大学（24人）
- 华中科技大学（23人）
- 东南大学（23人）
- 西安电子科技大学（20人）
- 南京大学（16人）
- 北京大学（15人）
- .....



# 报名情况简析

## ● 年级分布：

- 本硕博各年级均有分布
- 本科生占比 44%，硕士 50%，博士 6%
- 按细分来看，研一（27%），大三（23%），研二（14%），大二（11%）



# 教学团队

## ● 工作内容

- 准备讲义和其他辅助材料；组织报告和答疑

## ● 组织团队

- 技术指导：余子濠

- 技术负责：洪志博（深圳大学）

## ● 助教团队

- 线下助教（13人）：杨烨（深大）；薛臻（鹏城实验室）；夏斌（上海科大）；王唯一（浙大）；吴莹（浙大）；杨浩泽（东南大学）；郭明鑫（中科院自动化所）；祁民浩（深大）；段震伟（中科大）；梅晓龙（海南大学）；邓海文（海南大学）；史历（上海交大）；张宇轩（西安交大）；

- 线上助教（14人）：王嵩岳（国科大）；叶从容（集美大学）；桑乾龙（武大）；潘星雨（重庆邮电）；胡轩（中科院计算所）；刘实（浙大）；苑子琦（浙大）；刘定邦（哈工大（深圳））；周聪（天大）；肖天海（港科大）；尹承彬（上海交大）；陈春韵（南洋理工）；张行健（浙大）；唐伟康（涂鸦科技）

月	耗时	任务	运行的新程序
1	1周	完成PA2第一阶段(实现几条指令)	
	1周	单周期addi	addi指令
	1周	完成PA2第二阶段(实现完整rv32i/rv64i)	
	1周	实现更多指令，通过同步总线访存	cpu-test, riscv-tests
	1周	完成PA2第三阶段(输入输出)	
2	0.5周	加一条自定义指令，仿真时通过display输出字符	hello, coremark, dhystone, microbench(时钟返回0)
	0.5周	加mcycle寄存器	timer-test, 字符版超级玛丽
	1周	完成PA3第一阶段	
	1周	加CSR, ecall	yield-test, Nanos-lite, 仙剑
3	1周	加axi总线和模拟串口，去掉自定义指令 (需要握手，类似多周期处理器)	
	1周	完善CSR	RT-Thread
	1周	用axi访存(取指也需要握手)	
	1周	接入SoC, 从flash启动	

# 支撑团队

- **SoC团队 (4)**

- 工作内容: SoC集成和验证, 前后仿, 综合和时序约束等
- 预计周期: 10月7日到11月7日
- 技术指导: 刘彤
- 技术负责: 张文迪 (海南大学)
- 技术团队: 谢王照琪 (澳门科技), 吴泽辉 (华南理工), 龙康杰 (上海科大)

- **IC后端团队 (5)**

- 工作内容: 实施数理设计, 并生成可流片的 GDSII 版图
- 预计周期: 11月7日到11月30日
- 技术指导: 何伟及其团队
- 技术负责: 庄楚楠 (广东工业大学-新入职鹏城实验室)
- 技术团队: 张书涵 (海南大学), 方闻绩 (南航), 夏丽平 (华南师范), 熊启 (北大深)

# 维护学生的学习记录和组会情况

# 一位学生在“一生一芯”学习过程的成长记录

日期	计划任务	总时长	任务完成情况			卡了一段时间的bug		
2月25日	安装Verilator		nays库里现成的。尝试在cygwin环境中从源代码编译一份，编译出来的二进制					
2月26日	调试		3月28日 告诉完成MIPS的NEU	5h	基础部分MIPS和RISC-V差距不大。实现起来的思路也大同小异	貌似MIPS的ISA没有更新，直接make会报错。查看了报错部分的代码估计是前几年遗留还没改		
2月27日	完成		3月29日 完成MIPS的浮点计算	5h	完成浮点寄存器，可以将32位和64位浮点数放到寄存器中	浮点寄存器是64位的，而普通寄存器是32位的。一开始实现的时候没注意，导致在转换浮点数时，两个32位的浮点数相加会溢出		
3月1日	完成实验报告		3月30日 完成MIPS的浮点	5h	修SpinalHDL	SpinalHDL在Win11最新版上无法仿真，花了点时间定位了一下问题，提交到官	重装了nays包。升级了依赖包还是报错。问题点卡在native里面。靠反射方	
3月3日	完成		3月31日 完成MIPS的浮点	5h	修SpinalHDL	学习第三章，学了一下TLB实现的方法和Cache的实现方法。正好目前在实现		
3月6日	准备FPGA		4月1日 完成CSR指令	5h	学习《超标量》	Cache，就借鉴了里面的方法		
3月7日	完成FPGA		4月2日 完成CSR指令	5h	6月6日 学习《超标量》	跟着TLB设计写了一下TLB，感觉有点复杂。参考了一下开源项目，算是写出了	Cache实现多种多样，三级缓存和组相联实现起来很容易出错。靠着cadence手算发现了挺多细节上的问题	
3月8日	修正代码		4月3日 调试	5h	6月7日 学习《超标量》			
3月9日	完成		4月4日 调试	5h	6月8日 学习《超标量》			
3月10日	引入FPGA		4月5日 调试	5h	6月9日 调试AXI总线	7月18日 跑通diffest	之前的设计不太适合diffest仿真。diffest通过自定义总统进行数据传	
3月11日	完成		4月6日 FPG造型+起始	5h	6月10日 调试AXI总线	线特性引出了所需总统	人和传出，横跨了Fetch和PC部分	
3月12日	进一步		4月7日 完成DDR3内存	5h	6月11日 调试AXI总线	7月19日 跑通diffest	SpinalHDL和diffest结合起来就很麻烦，需要先导出才可以，不可以直接调用内部verilator	
3月14日	完成		4月8日 完成Flash绘图	5h	6月12日 调试AXI总线	7月20日 测试样例	每一次都要重新生成trace文件，导致重新跑一次需要很长时间。调了一	
3月15日	完成		4月9日 使用SFF和PMC作为外设	5h	6月13日 学习《超标量》	7月21日 测试样例	下makefile可以缓存一下	
3月16日	完成		4月10日 修正上电时序	5h	6月14日 学习《超标量》	7月22日 测试样例	官方的样例太大了，就算是缓存了跑一次也需要五分钟以上。PA中的有些样例	
3月17日	完成		4月11日 修正上电时序	5h	6月15日 切换到Lattice	7月23日 测试样例	会小很多。直接移植过来	
3月18日	完成		4月12日 布线	5h	6月16日 切换到Lattice芯片	7月24日 修复乘法器	由于只要测试指令是否实现正确，指令完整测试可以推到联调的时候。龙芯提供的gcc还挺快，Ubuntu不行，在CentOS下可以	
3月19日	完成		4月13日 布线	5h	6月17日 学习yosys	7月25日 使用IP核模拟	测试跑起来就比手写快多了，但是看着波形很容易卡，基本只能看上一小部分	
3月20日	完成		4月14日 打板	5h	6月18日 学习Reindl	7月26日 提升CPU频率	貌似官方的程序会在这里编译不过，有个类型写法有问题，会引起vivado报错	
3月21日	完成		4月15日 配置Verilator环境 & 学习	5h	6月19日 学习Reindl	7月27日		
3月22日	完成		4月16日 配置Virtuoso环境 & 开发	5h	6月20日 学习Reindl	7月28日		
3月23日	调试中断		4月17日 画CPU模块框图	5h	6月21日 学习Reindl	7月29日		
3月24日	完成		4月18日 修正模块架构	5h	6月22日 学习《超标量》	7月30日		
3月25日	完成		4月19日 完成指针和RV64I	5h	6月23日 学习《超标量》	7月31日		
3月26日	尝试引入		4月20日 完成RV64I	5h	6月24日 学习《超标量》			
3月27日	完成最基		4月21日 修改模块架构	5h	6月25日 学习《超标量》			
4月1日	五一放假		4月22日 编写Testbench	5h	6月26日 学习《超标量》			
4月2日	五一放假		4月23日 编写Testbench	5h	6月27日 写DCache	8月1日 分离APB总统时钟	一开始以为UART串口是内部分频至115200，后来翻来覆去看到分频器，而且选出的时钟都是一样的。最后才想到估计是pmco和linux软件定义了串口的波特率	
4月3日	五一放假		4月24日 编写Testbench	5h	6月28日 写DCache	8月2日 编译修改pmco	貌似pmco的APB总线频率定义有问题，改了后没改善。最后直接把波特率除以二编译	
4月4日	五一放假		4月25日 编写Testbench	5h	6月29日 写DCache	8月3日 修正initrd		
4月5日	五一放假		4月26日 取指部分重写	5h	6月30日 补JAL指	8月4日 编译上板测试coremark	pmco是能正常用，但是一运行linux波特率又回去了。但是这个波特率好像是闪存里定义的	
4月6日	五一放假		4月27日 编写计分板	5h	7月1日 补DIV指	8月5日 减少扇出数量	龙芯的coremark是直接烧录flash的模式，并不是在linux中跑coremark程序的模式。但是给的配置文件都是硬编码，只能从官方源拉一份。然后交叉编译上去跑	
4月7日	五一放假		4月28日 编写计分板	5h	7月2日 测试JAL和DIV	8月6日 忽略指定path	频率升高后时钟约束就跑不过。看了下net延迟比logic延迟大太多了，而且layout图上很明显跨区连线。估计是扇出大，限制扇出数量后报错数量减少了	
4月8日	五一放假		4月29日 学习香山代码	5h	7月3日 编译LA3	8月7日 修正no drive on错误	分析了一下剩下的粗指path，大部分都是经过mul。看路径推断是由mul引起PC改变，进而刷新流水线。因为日常根本不会有这种情况，直接忽略。然后就没有时钟报错了	
4月9日	五一放假		4月30日 学习香山代码	5h	7月4日 调试LA顶层	8月8日 修正no drive on错误	在SpinalHDL写CSR的时候，只要操作CSR寄存器就会编译报错。但是相同的逻辑在Chisel中是正常的。而且生成的Verilog代码也没有问题	
4月10日	五一放假		5月1日 学习香山代码	5h	7月5日 调试LA的AXI	8月9日 修正no drive on错误	貌似只要操作Bundle中的任何一个信号，都会导致Bundle里面所有的信号都报错	
4月11日	五一放假		5月2日 学习香山代码	5h	7月6日 调试LA的AXI	8月10日 修正diffest接口信号名	调试了半天才发现如果io的信号没有外部连接就会报错。只有把全部信号全连上才行	
4月12日	五一放假		5月3日 学习香山代码	5h	7月7日 调试LA的AXI	8月11日 新增中断信号	diffest运行起来报错，检查了很久才发现是接口处的信号名全部加上了io的前缀，即使是blackbox引入的也会自动加上。在源代码里一通翻，最后通过反射重命名信号名才可以	
4月13日	五一放假		5月4日 学习香山代码	5h	7月8日 调试LA的AXI	8月12日 新增中断信号	直接把外部信号引入CSR模块。一旦插入中断，就直接设置寄存器	
4月14日	五一放假		5月5日 学习香山代码	5h	7月9日 调试LA的AXI	8月13日 新增中断信号	不能直接设置寄存器，还必须洗刷流水线。这就导致了中断还和其他模块连在一起，只能全部加入了中断和异常信号量	
4月15日	五一放假		5月6日 学习香山代码	5h	7月10日 调试LA的pmco	8月14日	测试时候中断会引发当前寄存器内容丢失。在中断中出不来	
4月16日	五一放假							
4月17日	五一放假							
4月18日	五一放假							
4月19日	五一放假							
4月20日	五一放假							
4月21日	五一放假							
4月22日	五一放假							
4月23日	五一放假							
4月24日	五一放假							
4月25日	五一放假							
4月26日	五一放假							
4月27日	五一放假							
4月28日	五一放假							
4月29日	五一放假							
4月30日	五一放假							
5月1日	五一放假							
5月2日	五一放假							
5月3日	五一放假							
5月4日	五一放假							

# 第三期“一生一芯”成果

## ● 流片情况：

- 2021年12月底班车：**39个处理器核**（包括助教5个、测试核2个）
- 2022年2月底班车：**9个处理器核**

## ● 答辩情况

- **C9 (10), 985 (12), 211 (9), 普通高校 (11)**
- 电子/微电子/集成电路：**25人**；计软：**13人**；其他：**4人**
- **大一 (2), 大二 (3), 大三 (11), 大四 (3)**
- **研一 (8), 研二 (11), 研三 (1), 博一/二/三 (4)**

## ● 指标情况

- 分支预测：12个；
- 乱序执行：3个；
- Chisel:16个；Verilog/sv: 31个
- Cache: 12个

学号	学校	年级	
ysyx_210596	中山大学	研二	微电子
ysyx_210760	扬州大学	大三	计算机科学与技术
ysyx_210407	长春理工大学	大四	电子科学与技术
ysyx_210366	广东工业大学	研二	控制科学与工程
ysyx_210703	上海大学	大三	计算机科学与技术
ysyx_210324	西北工业大学	研三	控制工程
ysyx_210747	北京航空航天大学	研二	电子信息工程
ysyx_210746	电子科技大学	研一	电子科学与技术
ysyx_210448	山东交通学院	大二	物联网工程

ysyx_210340	中国科学院大学	直博二年级	微电子学与固体电子学
ysyx_210092	西安电子科技大学	大三（即将升）	计算机科学与技术专业嵌入
ysyx_210456	电子科技大学	研一	电子信息科学与技术
ysyx_210247	南京理工大学	研一	电子信息
ysyx_210243	华中科技大学	大三	电子信息与通信工程
ysyx_210544	南京航空航天大学	博士一年级	软件工程
ysyx_210232	青岛科技大学	大三	集成电路设计与集成系统
ysyx_210295	华东师范大学	研一	集成电路设计与集成系统
ysyx_210457	山东交通学院	大一	电子信息工程
ysyx_210458	太原理工大学	大二	水利
ysyx_210611	南京大学	大一	计算机科学与技术
ysyx_210285	南京大学	大二（准大三）	计算机科学与技术
ysyx_210128	上海交通大学	大四	电子与计算机工程
ysyx_210727	华中科技大学	研二	计算机科学与技术
ysyx_210718	深圳大学	研二	电子信息
ysyx_210133	电子科技大学	研二	电子科学与工程学院

ysyx_210292	集美大学	大四	计算机科学与技术
ysyx_210191	南京理工大学	大三	计算机科学与技术
ysyx_210195	西安电子科技大学	硕士二年级	电子科学与技术
ysyx_210413	大连理工大学	研一	软件工程
ysyx_210428	沈阳工业大学	研二	电子科学与技术
ysyx_210313	电子科技大学	大三	微电子
ysyx_210302	复旦大学	研一	微电子学与固体电子学
ysyx_210184	清华大学	研二	集成电路工程
ysyx_210479	太原理工大学	大三	计算机科学与技术
ysyx_210013	西安交通大学	研二	微电子学与固体电子学
ysyx_210438	南京大学	直博二年级	电子信息技术
ysyx_210555	南京大学	研一	集成电路工程
ysyx_210528	中国农业大学	大三	电子信息工程
ysyx_210669	北京工业大学	研二	计算机技术
ysyx_210417	中国科学技术大学	研一	集成电路工程
ysyx_210134	浙江大学	大三	计算机科学与技术
ysyx_210152	重庆邮电大学	大三	电子信息工程

# 第三期 “一生一芯”的成果



# 第三期“一生一芯”学生学习心得分享

- 项目组安排了8位同学的分享报告
  - **不同点：来自不同学校、不同年级、不同专业**
  - **相同点：之前均未设计过处理器**
    - 高泽宇，中国科学院大学，报名时大三，电子信息工程
    - 徐鑫，山东交通学院，报名时大一，电子信息工程
    - 孙际儒，南京大学，报名时大一，计算机科学与技术
    - 粟金伦，太原理工大学，报名时大二，水利
    - 于皓哲，沈阳工业大学，报名时研一，电子科学与技术
    - 许立达，中科院微电子所，报名时博一，微电子
    - 张文迪，海南大学，报名时研二，电子与通信工程，负责SoC集成和验证
    - 庄楚楠，广东工业大学，应届毕业生，微电子，负责后端物理设计

### 心得3 – 工程经验的积累

- 经验一：“先完成，后完美”原则指导任务规划  
- 在没有讲义的情况下，自己分解任务、规划步骤，并评估自己的规划



- **初版规划**: 没有验证新功能正确性, 急于加入流水线→bug层出不穷, 难以排查
- **最终规划**: 运行新测试保证功能正确性, 再提升性能→进度进展顺利



# 高泽宇@中国科学院大学 电子信息工程, 报名时大三

### 三、“一生一芯”带来的能力提升

## 能够独立探索

- 1天上手Verilog
  - 独立弄懂AXI
  - 能够灵活运用Verilator
    - 修正计组实验的测试框架
    - 链接自己的项目

知识不是老师教会的  
而是自己学会的



孙际儒@南京大学  
计算机科学与技术, 报名时大一

03 总结回顾

两种能力的提升

主动学习能力

- 硬件语言verilog的学习
  - 计算机基础的学习
  - 开发环境的学习
  - 指令集的学习

**用科学的方法快速解决问题的意识和能力**

周 期	7.15	2h	安装verilog, oscillator+gitware	2h	2h
	7.16	3h	学习verilog语法	3h	3h
	7.17	5h	我看verilog, 单周期处理器进阶, 测试cpu文件	2h+2h+1h	5h
	7.19	5h	学习计算机组成原理	34h	5h
	7.20	6h	计算机组成原理, 数字电路学习	6h+6h	6h
	7.21	6h	继续学verilog, 提炼原理	6h+6h	6h
	7.22	7h	学习状态机的结构和设计方法, 尝试上手设计	7h+7h	7h
	7.23	5h	学习μc的使用, 看讲座回顾	5h	5h
	7.24	6h	学习了IP核的数据通路结构	6h	6h
	7.26	必	安装ivadk并学习了alu, pc等基础结构	必	必
	7.27	5h	学习计算机组成原理体系结构(4-7章)	5h	5h
	7.28	必	学习了周期, 多周期设计进阶	必	必
	7.29	6h	看了addi的访存, 写回阶段	6h	6h



徐鑫@山东交通学院  
电子信息工程，报名时大一

### 一、工欲善其事，必先利其器

<sup>[1]</sup> 香山开发过程中使用大量的高效开发的工具，包括但不限于 Verilator（高速仿真器）、NEMU(指令集模拟器)、Difftest(差分验证框架) ...

**Verilator仿真coremark的速度比Vivado快约300倍**



用了“一生一芯”建议的开发工具，我认识到原来优秀的工具可以对开发起到如此大的提升。现在我在开发的时候首先会想到完善基础设施。

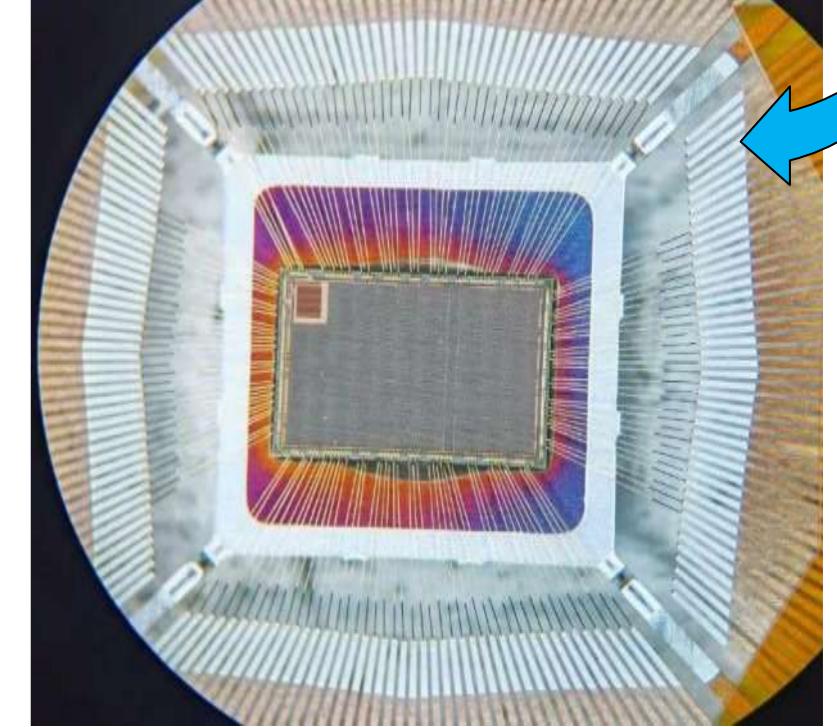
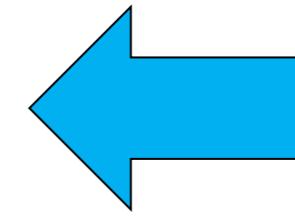
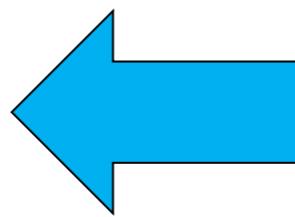
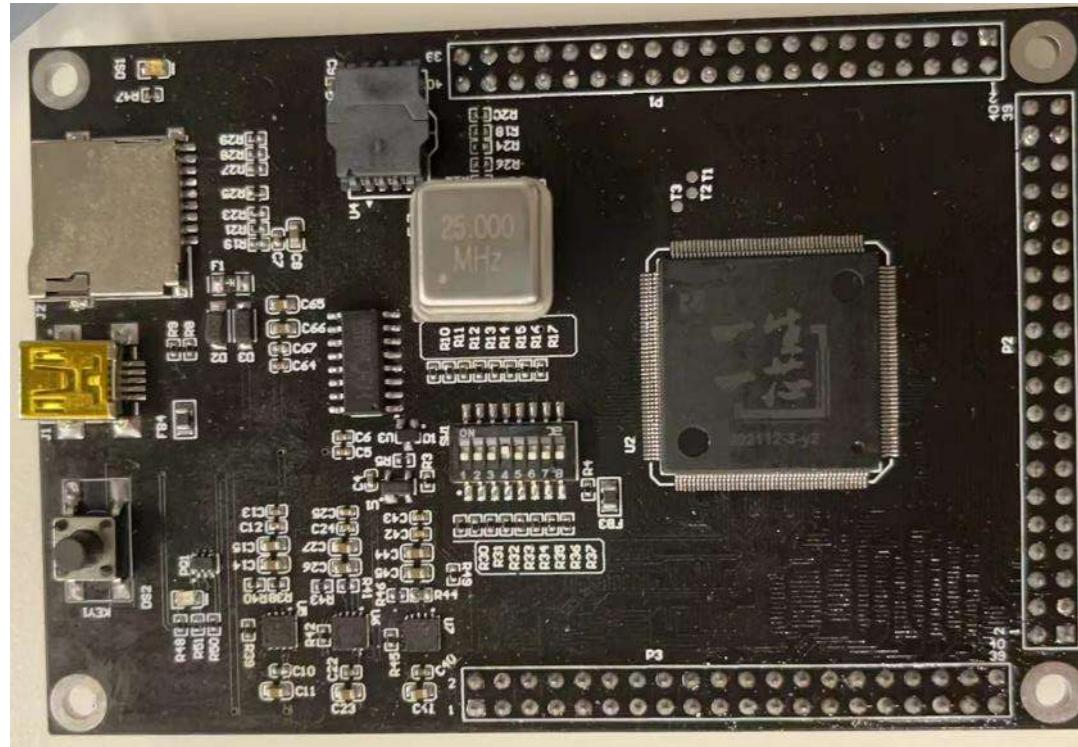
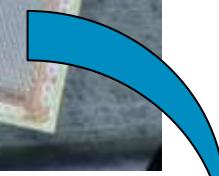
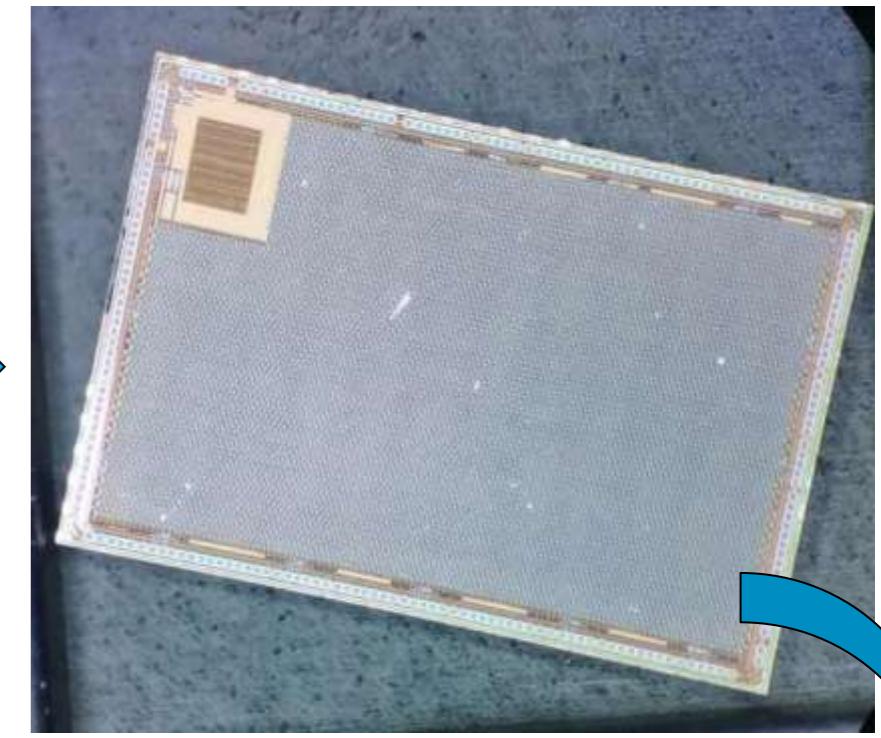
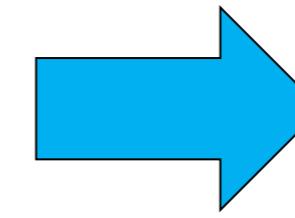
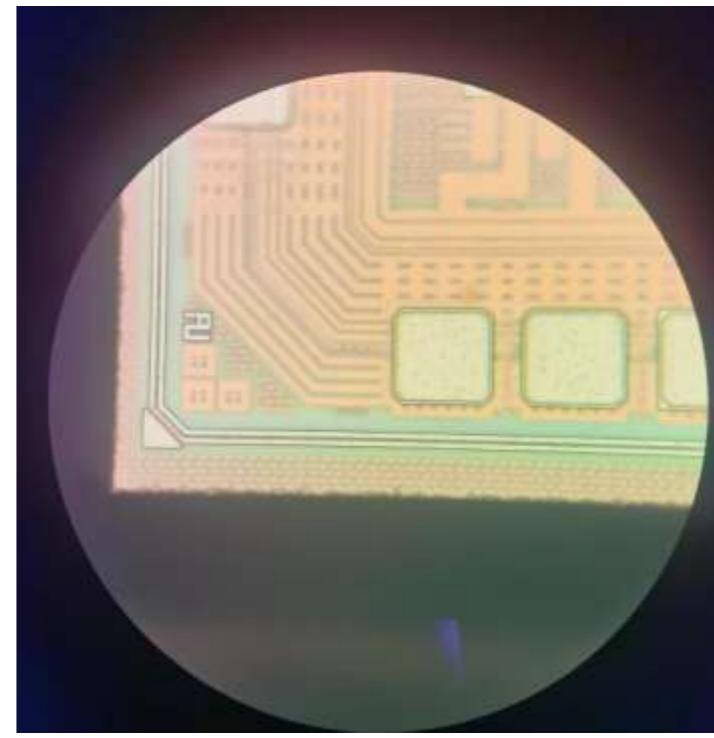
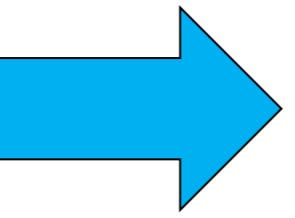
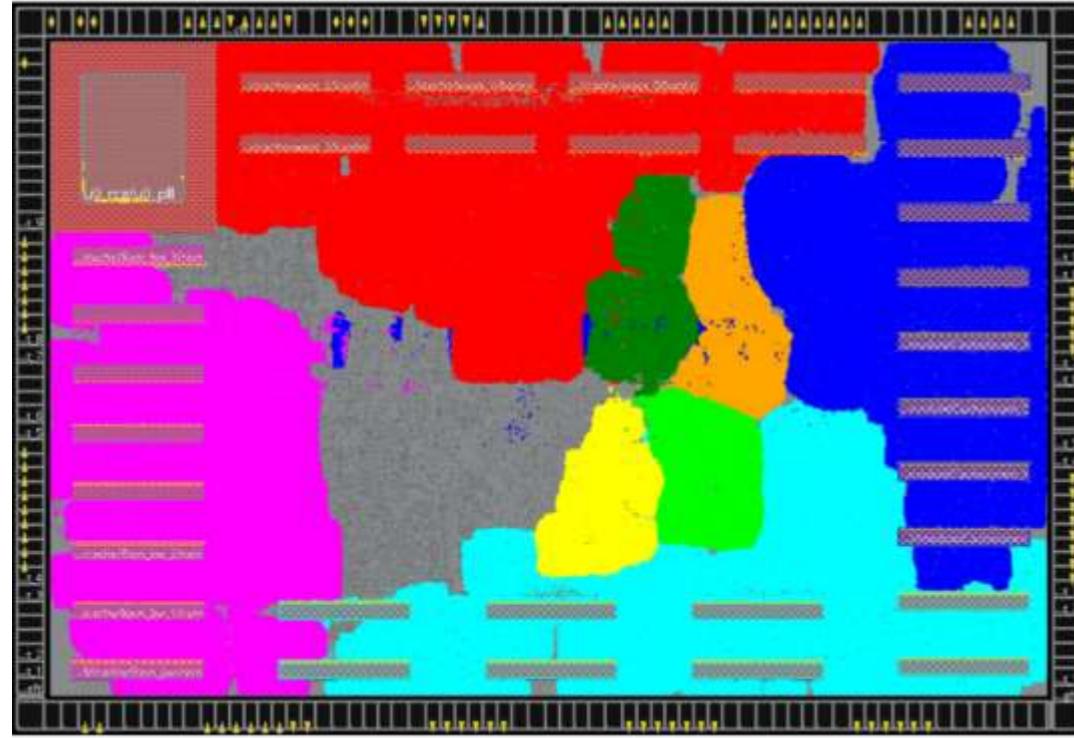


[1] <https://github.com/OpenXiangShan>

# 粟金伦@太原理工大学 水利，报名时大二



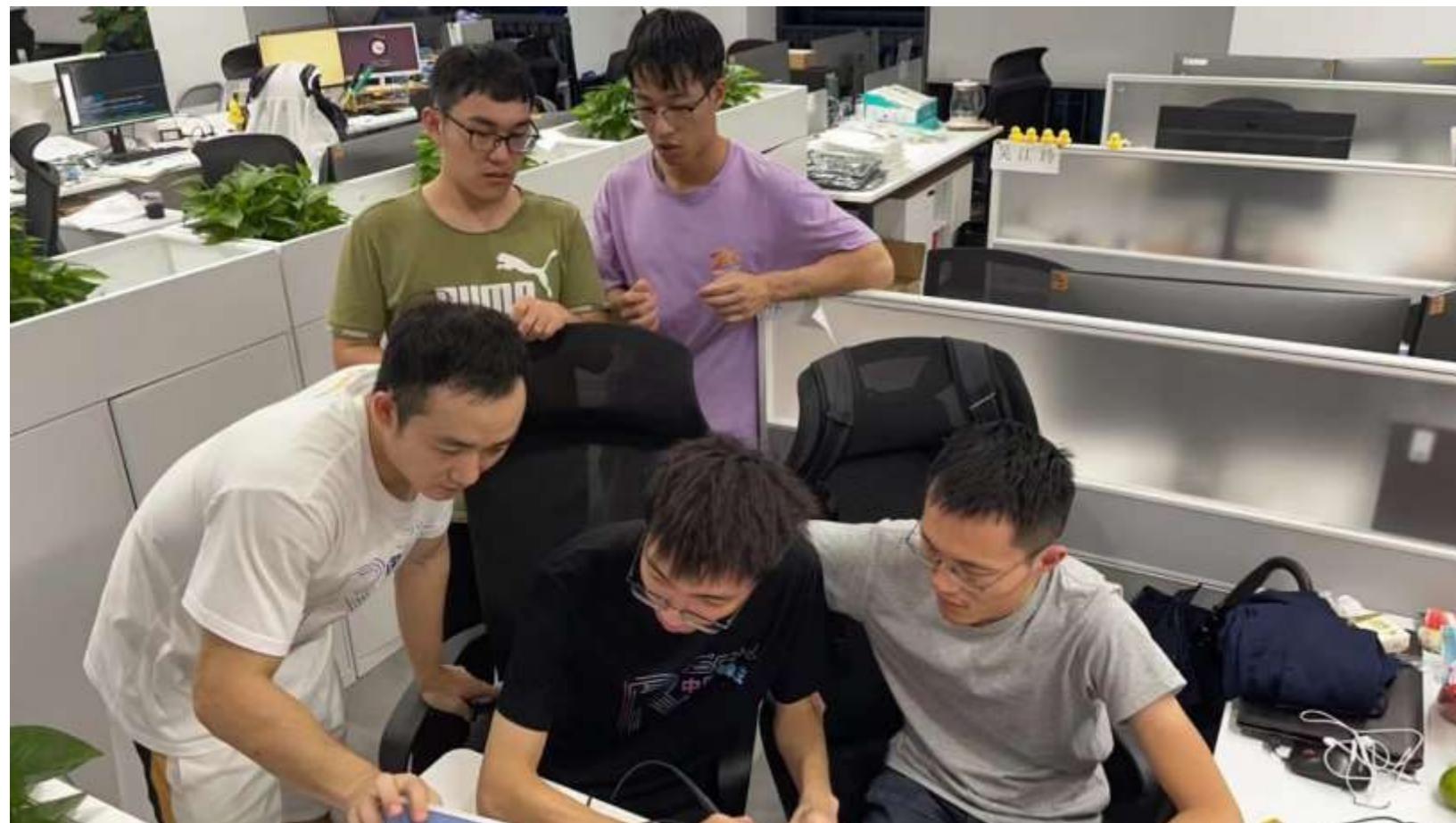
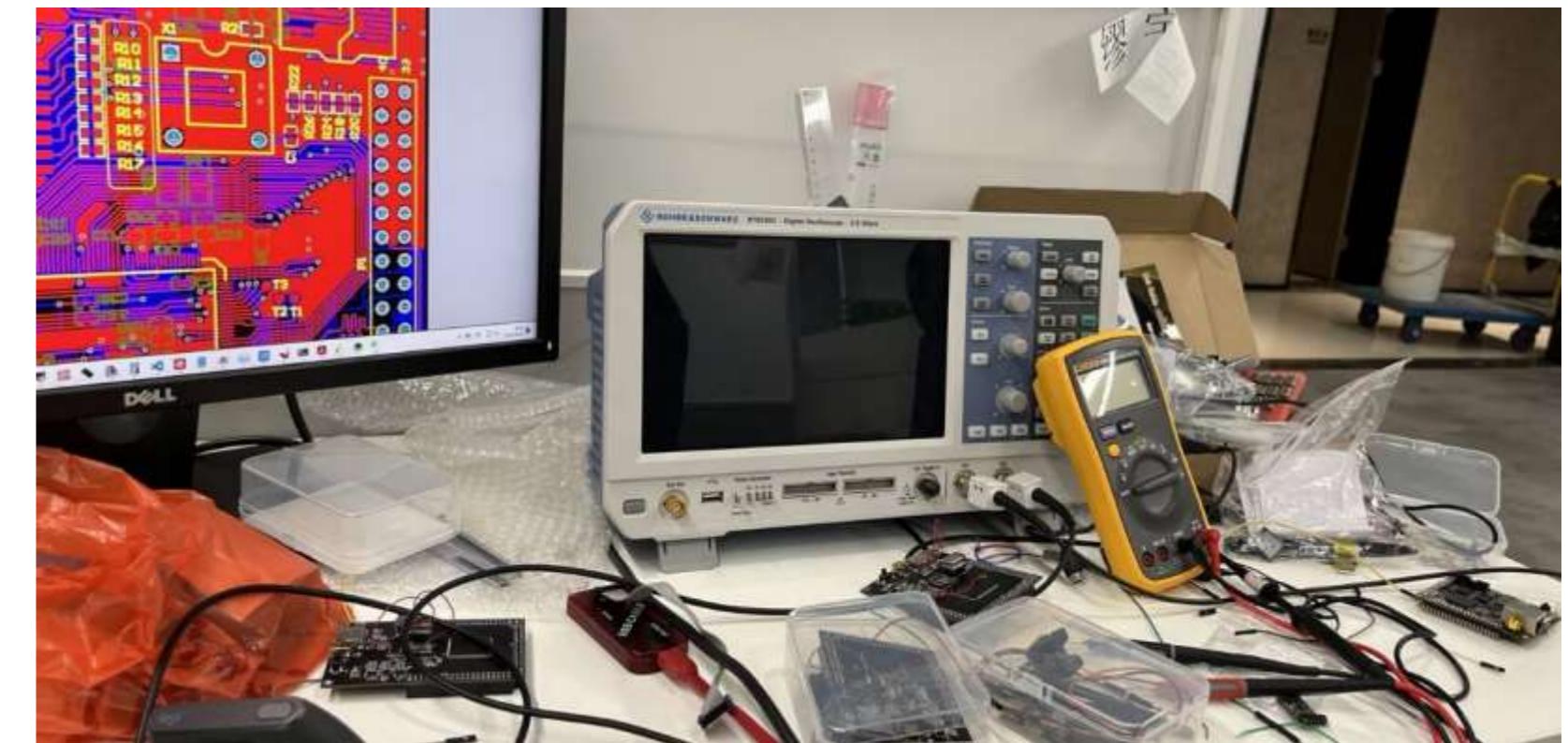
# 第三期处理器芯片和板卡展示



# 第三期板卡元件测试

## 测试团队

- 黄健明(海南大学)
- 卢非凡(西安财经大学)
- 马壮(中国科学技术大学)
- 缪宇驰(鹏城实验室)
- 许立达(中科院微电子所)



# 第三期板卡软件测试

	hello world	memtest
flash	[10:59:26.279]收←◆Hello World! [11:04:04.430]收←◆Hello World! [11:04:06.275]收←◆Hello World!	[14:51:55.890]收←◆start test... mem tests prepared mem tests passed!!
mem	[14:19:33.762]收←◆Loading program of size: 208 bytes, expect 128 '#' Loading.... ##### Load finished Exec app... Hello World!	[15:06:04.482]收←◆Loading program of size: 3840 bytes, expect 128 '#' Loading.... ##### [15:06:04.745]收←◆##### Load finished Exec app... start test... mem tests prepared mem tests passed!!

内存  
测试

[10:31:49.358]收←◆[mem data] cnt: 65929216(3ee0000), addr: 0x9f900000  
[10:31:49.416]收←◆[mem data] cnt: 65994752(3ef0000), addr: 0x9f980000  
[10:31:49.476]收←◆[mem data] cnt: 66060288(3f00000), addr: 0x9fa00000  
[10:31:49.535]收←◆[mem data] cnt: 66125824(3f10000), addr: 0x9fa80000  
[10:31:49.594]收←◆[mem data] cnt: 66191360(3f20000), addr: 0x9fb00000  
[10:31:49.653]收←◆[mem data] cnt: 66256896(3f30000), addr: 0x9fb80000  
[10:31:49.712]收←◆[mem data] cnt: 66322432(3f40000), addr: 0x9fc00000  
[10:31:49.772]收←◆[mem data] cnt: 66387968(3f50000), addr: 0x9fc80000  
[10:31:49.830]收←◆[mem data] cnt: 66453504(3f60000), addr: 0x9fd00000  
[10:31:49.889]收←◆[mem data] cnt: 66519040(3f70000), addr: 0x9fd80000  
[10:31:49.949]收←◆[mem data] cnt: 66584576(3f80000), addr: 0x9fe00000  
[10:31:50.007]收←◆[mem data] cnt: 66650112(3f90000), addr: 0x9fe80000  
[10:31:50.067]收←◆[mem data] cnt: 66715648(3fa0000), addr: 0x9ff00000  
[10:31:50.126]收←◆[mem data] cnt: 66781184(3fb0000), addr: 0x9ff80000  
[10:31:50.185]收←◆mem tests passed!!  
[10:32:25.838]收←◆\0\0

RT-Thread  
测试

Load finished  
Exec app...  
heap: [0x80022590 - 0x86422590]  
\\ /  
- RT - Thread Operating System  
/ | \ 4.0.4 build Nov 29 2022  
2006 - 2021 Copyright by rt-thread team  
Hello RISC-V!  
thread1 count: 0  
thread2 count: 0  
msh />  
[10:41:52.743]收←◆thread1 count: 1  
thread2 count: 1  
[10:41:53.385]收←◆thread1 count: 2  
thread2 count: 2

# 第三期学生进行beta测试



中区(B) 编程操作(O) 自动选项(A) 硬件(H) 语言(I) 关于(I)

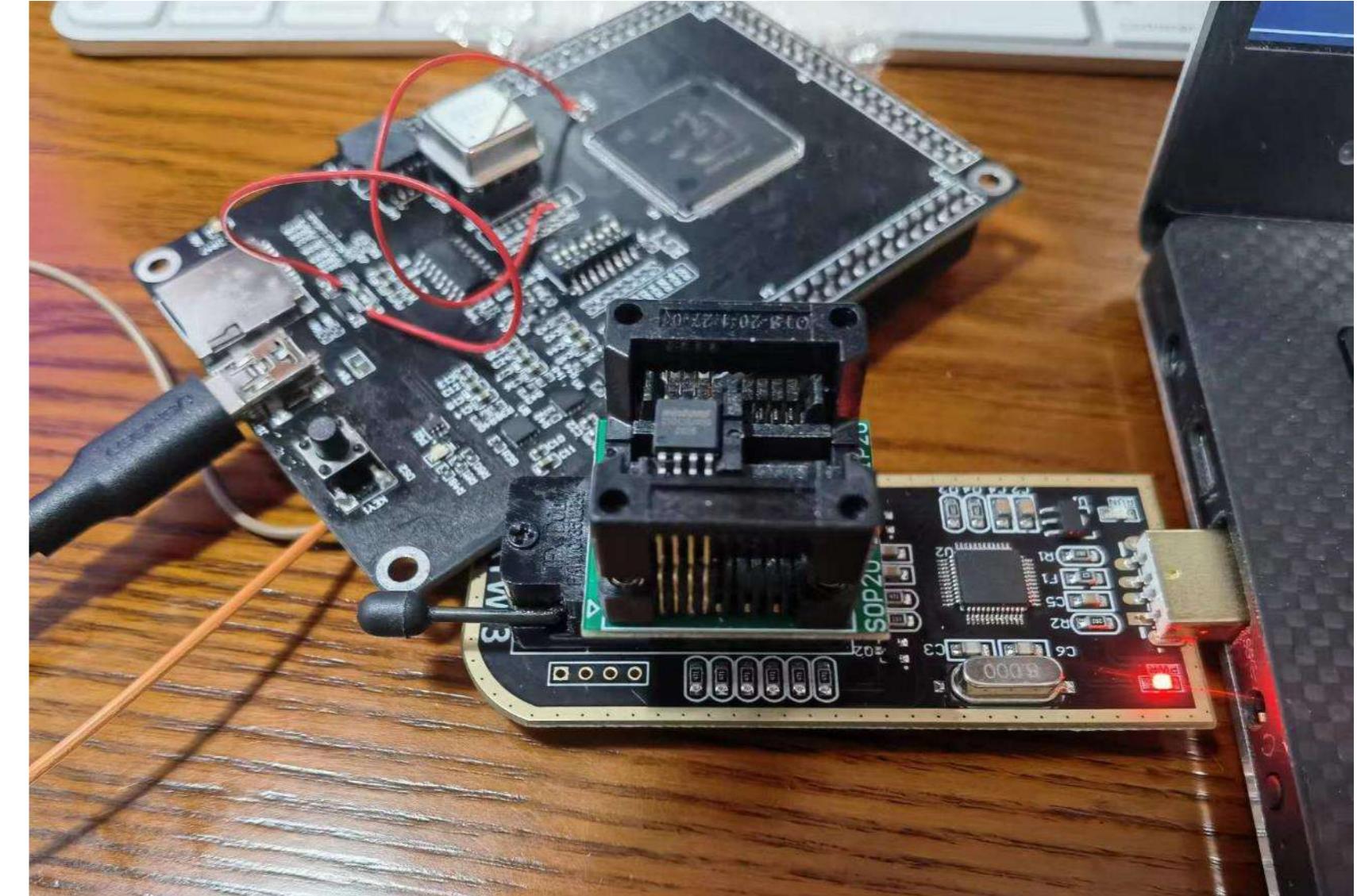
保存 填充 自动 检查 清除 写入 读出 校验 批量 停止 设备 关于

SH  
ND  
尔:  
28JV

容量: 16MB  
页大小: 256B  
芯片电压: 3.3V

查找

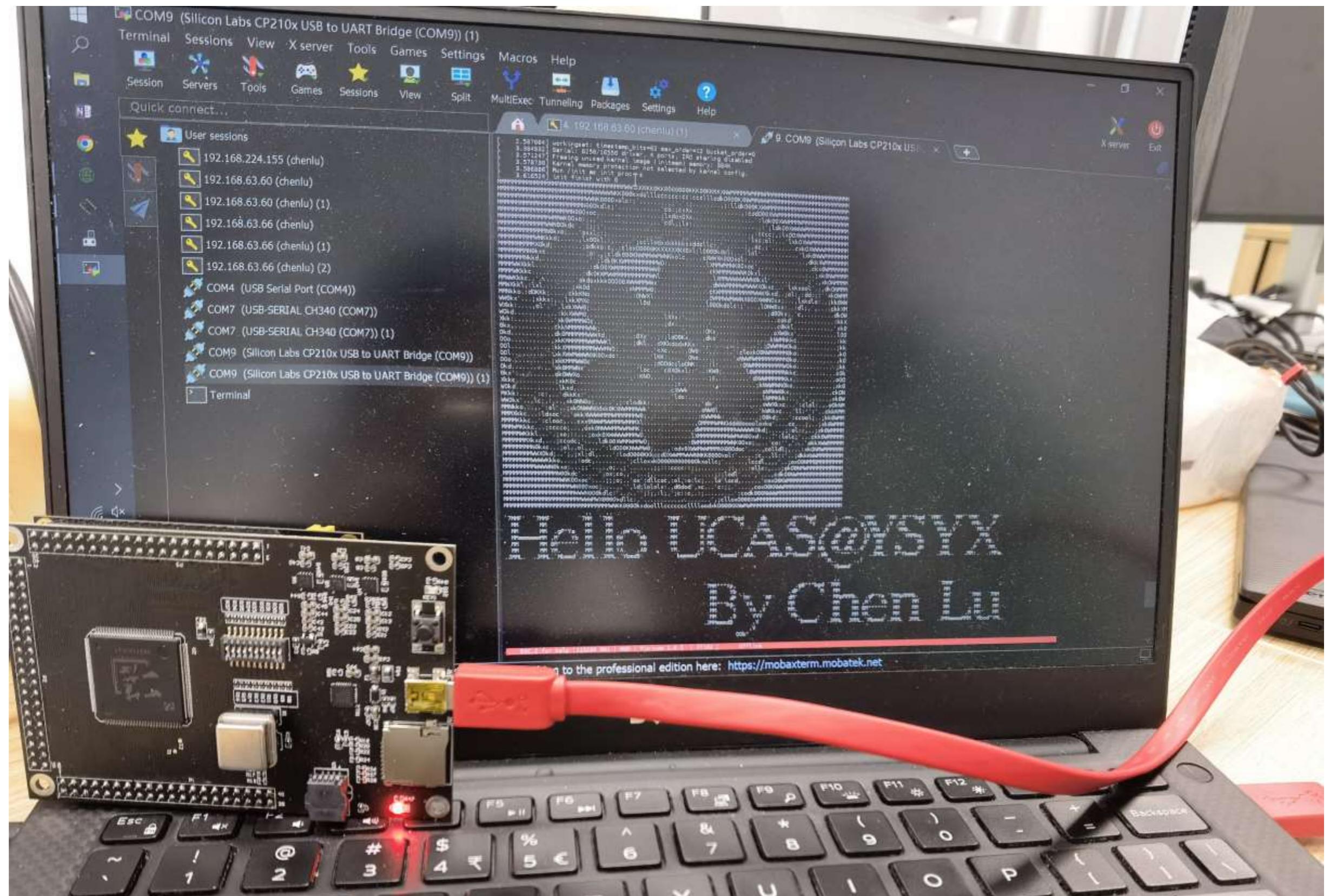
ADDR	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ASCII
00000000H	37	07	00	10	83	47	37	00	93	E7	07	F8	93	F7	F7	0F	7...G7...
00000010H	A3	01	F7	00	93	07	D0	00	23	00	F7	00	93	07	30	00	#...0...
00000020H	A3	01	F7	00	17	04	00	00	13	04	44	06	13	05	80	04	D...
00000030H	13	04	14	00	EF	00	C0	03	03	45	04	00	E3	1A	05	FE	E...
00000040H	13	03	10	00	93	03	60	00	17	04	00	00	13	04	E4	04	...
00000050H	13	05	80	04	13	04	14	00	EF	00	80	01	03	45	04	00	...
00000060H	E3	1A	05	FE	13	03	13	00	E3	00	63	FE	6F	00	00	00	...
00000070H	37	07	00	10	83	47	57	00	93	F7	07	02	E3	8C	07	FE	7...GW...
00000080H	23	00	A7	00	67	80	00	00	48	65	6C	6C	6F	2C	20	59	...g...Hello, Y...
00000090H	53	59	58	21	0A	00	48	61	76	65	20	61	20	67	6F	6F	SYX! Have a goo...
000000A0H	64	20	6C	75	63	6B	21	0A	00	FF	...						
000000B0H	FF	...															
000000C0H	FF	...															
000000D0H	FF	...															
000000E0H	FF	...															
000000F0H	FF	...															
00000100H	FF	...															
00000110H	FF	...															
00000120H	FF	...															
00000130H	FF	...															
00000140H	FF	...															
00000150H	FF	...															
00000160H	FF	...															



Hello, YSYX!  
Have a good luck!  
Have a good luck!  
Have a good luck!  
Have a good luck!

# 第三期学生单人作品展示

- 陈璐@南京大学, 报名时大三, 计算机科学与技术
  - 从Flash中加载Linux并启动, 展示中科院logo



# 第三期学生单人作品展示(2)

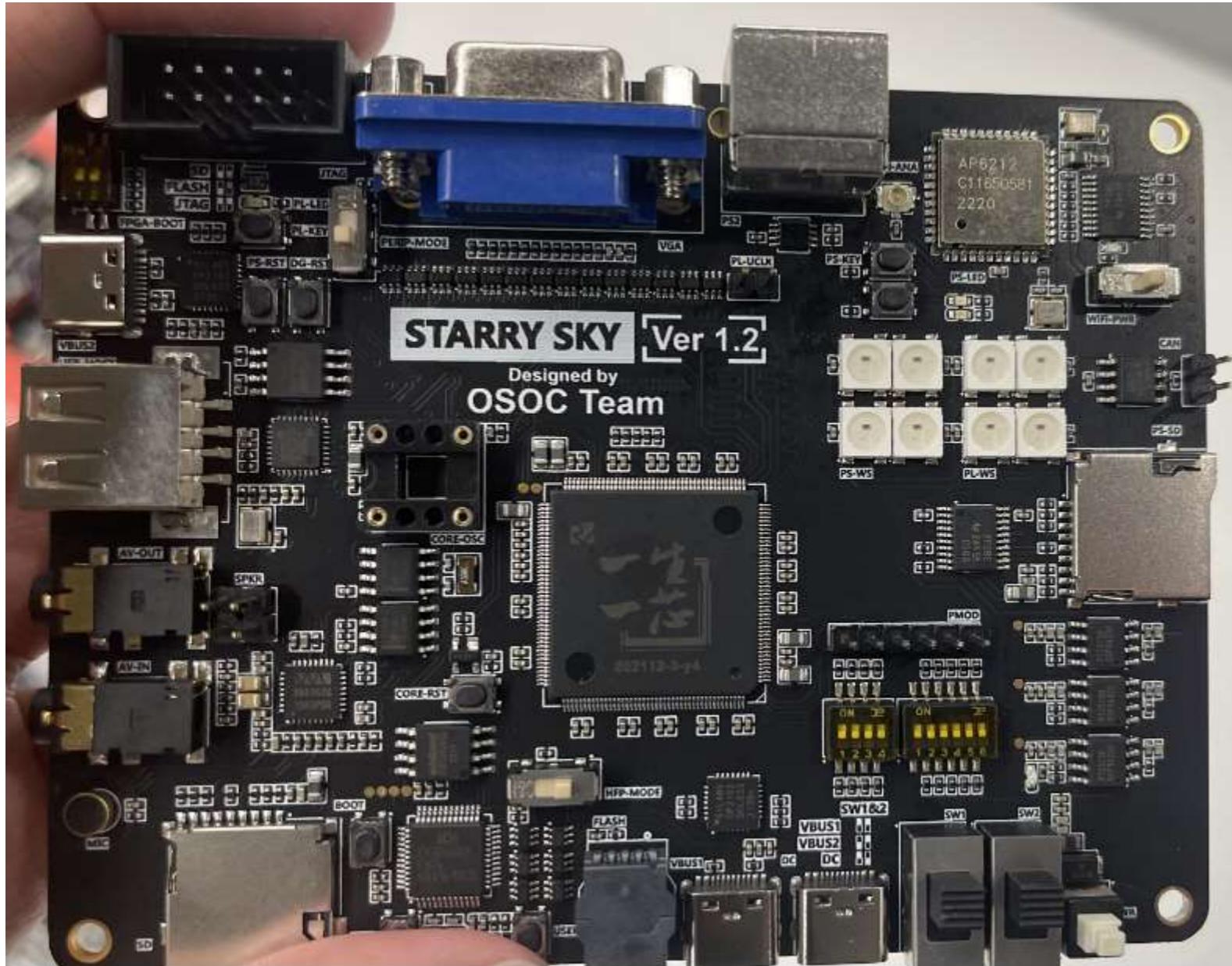
- 唐浩晋@中国科学院大学, 报名时大三, 电子信息工程
  - 在Linux上运行字符版2048游戏



如果不小心输成了sl

<https://www.bilibili.com/video/BV1CL411X7wV>

# 板卡设计和发放

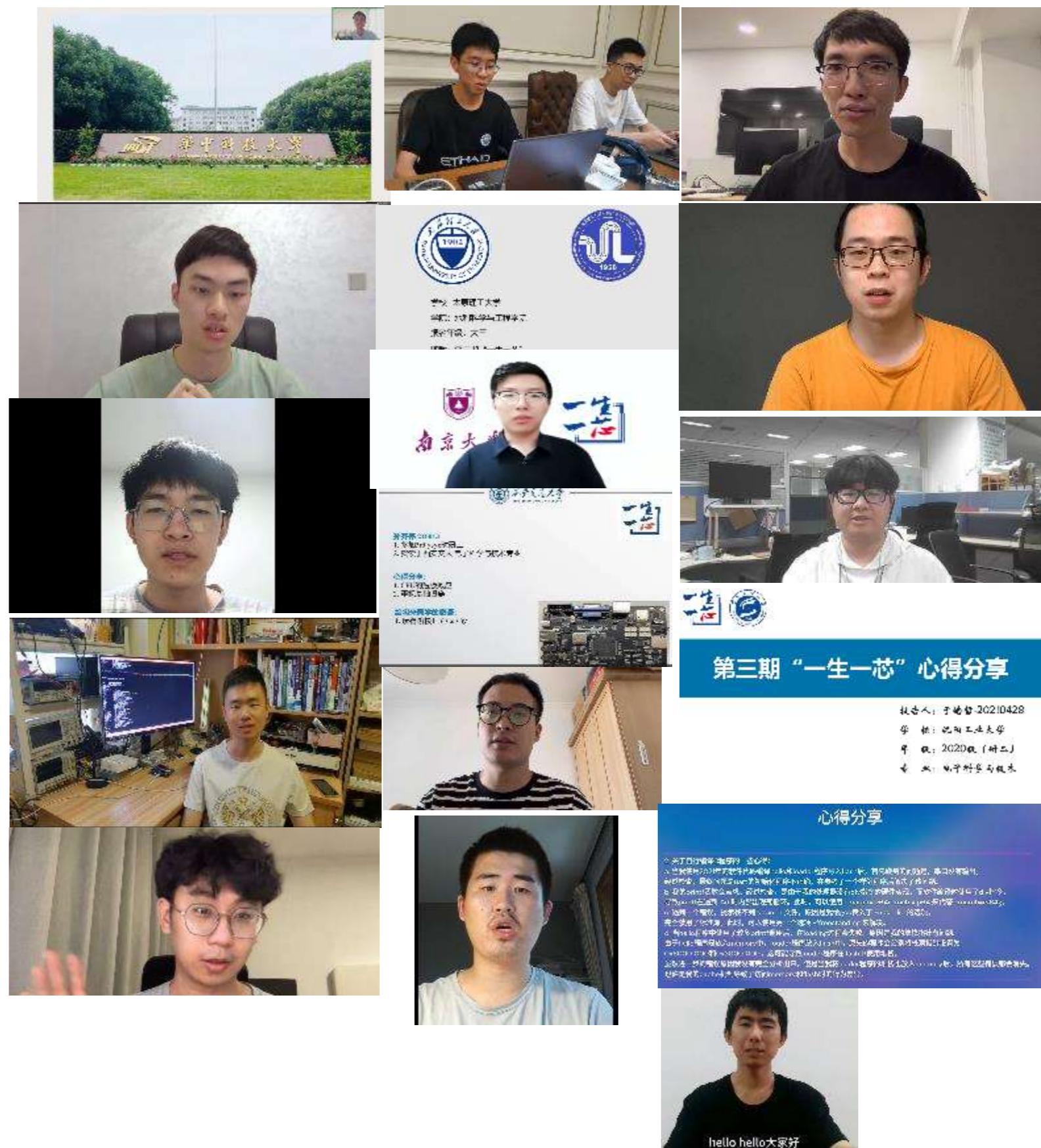
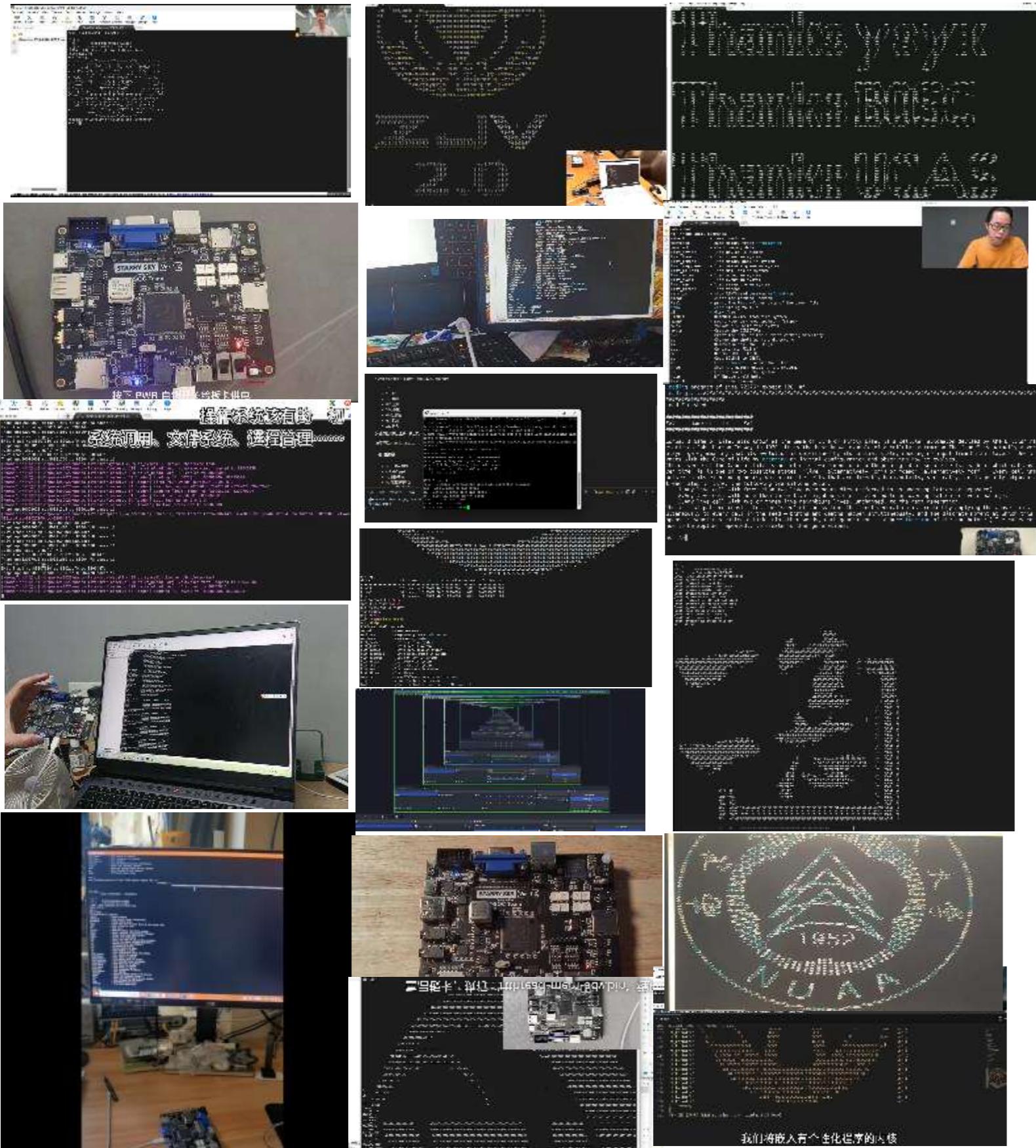


搭载“一生一芯”芯片的“星空”板卡



准备寄出的物料和板卡

# 学生录制的板卡点亮视频和心得分享视频



# **五、建设高标准、低坡度的 学习流程**

## **第四~六期**

### **(2022年2月~今)**

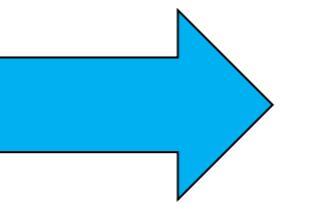
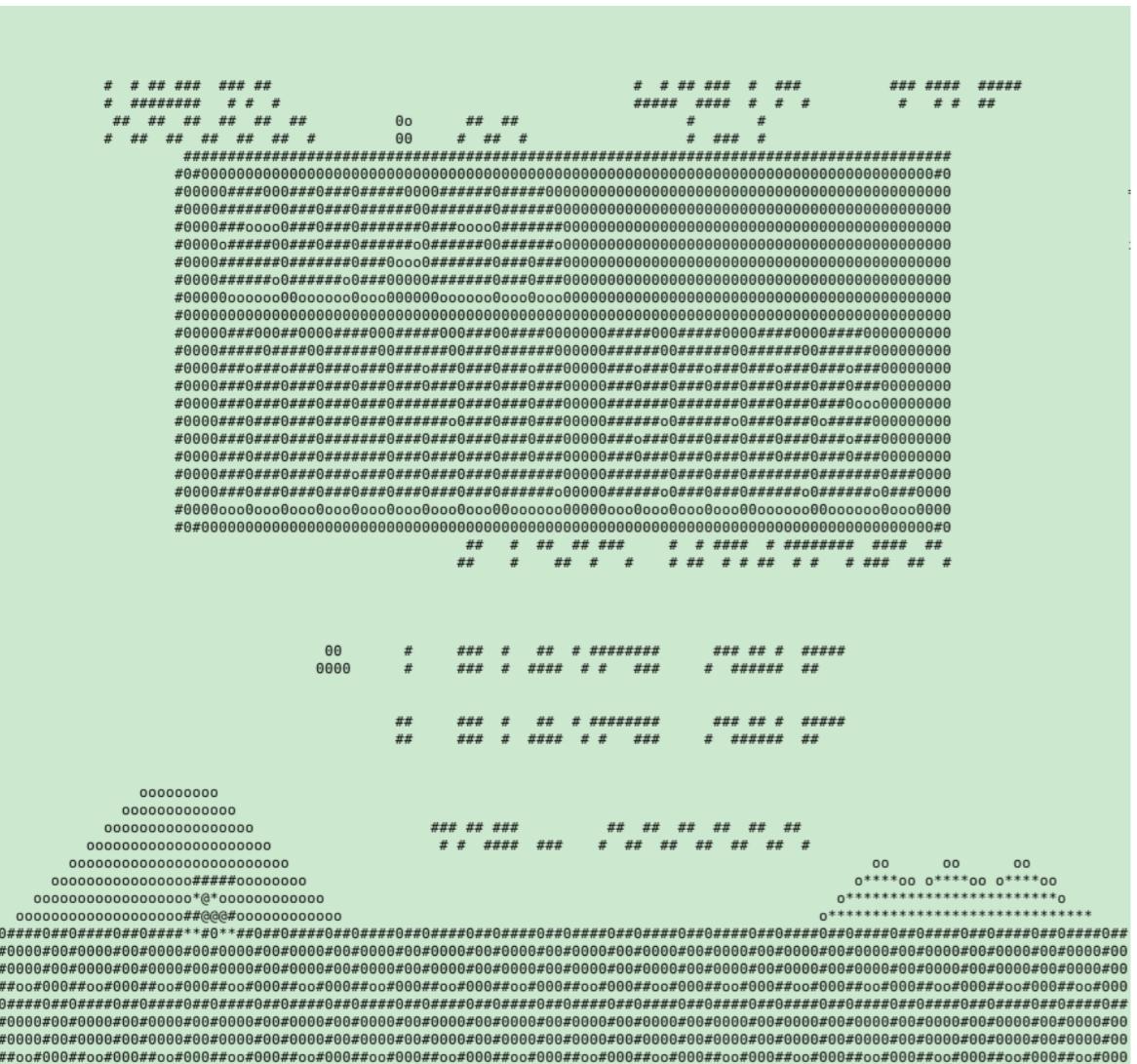
## 第四/五期

- 提高学习指标: RV64IM启动自制OS软件栈, 运行游戏 “仙剑奇侠传”
- 加入南京大学的PA实验, **强化软硬件协同的训练**
  - 自行实现模拟器/编译程序/编写自制OS和运行时环境/搭建仿真和测试环境...
- **完善讲义, 共26万字**(含PA部分)
  - 预学习/B(aseline)阶段/A(dvanced)阶段, 指导学生学习
  - 螺旋上升的融合式计算机系统设计
- **培养和流片分离**, 学习不设截止日期
- 取消组队, 只允许单人参加
- 第五期**新增教学视频和课件**

# 第四期学习成果

- 新增VGA, 提升展示效果

第三期的  
字符版本



B阶段学习成果: 运行80年代游戏《超级玛丽》

# 第四期学习成果

- 新增VGA, 提升展示效果



A阶段学习成果：  
运行90年代游戏《仙剑奇侠传》



S阶段学习成果（部分）：  
运行00年代游戏《CLANNAD》

# 报名无需费用，学习资源均开放

## 第五期“一生一芯”课程主页

- 课时: 每周六19:00~21:00
  - B站直播 | 录播链接
- 答疑: 每周日19:00~20:00 (通过预学习答辩后由助教通知)
- 报名流程 | 报名常见问题

### 课件和讲义

0. C = C语言(程序/模拟器/系统软件) | R = RISC-V指令集 | P = 处理器设计 | T = 工具

Pre-study Phase:

1. “一生一芯”概述 | 如何科学地提问
2. 工具是第一生产力——Linux入门教程 | Linux系统安装和基本使用
3. 计算机系统的状态机模型 | C | R | P | 复习C语言
4. 从C语言到二进制程序 | C | T
5. 程序的执行和模拟器 | C | P | 搭建verilator仿真环境 | 数字电路基础实验
6. NEMU代码导读 | T | 完成PA1

PA Phase:

7. RISC-V指令集 | C | R | P | 支持RV64IM的NEMU
8. 程序的机器级表示 | C | R | P
9. RISC-V单周期处理器设计 | P | 用RTL实现最简单的处理器
10. Abstract Machine裸机运行时环境 | C | 运行时环境和基础设施
11. ELF文件和链接 | C | R | P
12. 工具和基础设施 | T | 支持RV64IM的单周期NPC
13. 设备和输入输出 | C | R | P | 设备和输入输出
14. 调试技巧进阶 | T | P

AP Phase:

15. 异常处理 | C | R | P | 简单的异常处理机制
16. 计算机系统软件 | C | 用户程序与系统调用 | 精彩纷呈的用户程序

### 计算机系统的状态机模型

余子濠



中国科学院  
计算技术研究所

计算机系统与处理器  
芯片课程虚拟教研室

- **课程主页**
- **讲义26万字**
- **课件800+页, 8.5万字**
- **教学视频时长40+小时**

### Linux系统安装和基本使用

安装一个Linux操作系统

我们复用PA讲义的内容,请大家根据PA安装Linux操作系统。

获取“一生一芯”框架代码

当你阅读PA讲义,并进行到获取PA框架代码的部分,将会有提示框请你返回到此处的讲义内容。

首先请你在github上添加一个ssh key,具体操作请STFW,然后通过以下命令获取“一生一芯”的框架代码:

```
git clone -b ysyx2284 git@github.com:OSCPU/ysyx-workbench.git
```

获取后,你就可以回到PA讲义的相应位置,继续阅读了。不过你还需要注意:

- 请把 `ysyx-workbench` 作为PA讲义中的项目目录,即PA讲义中的 `11.2022` 目录是 `ysyx-workbench`
- 修改 `ysyx-workbench/Makefile` 中的学号和姓名时,请使用“一生一芯”的学号和真实姓名

这种来回跳转的做法可能会给你带来一些麻烦,但我们之所以这样做,是希望把文档看作代码来管理。我们希望做到类似“一生一芯”讲义模块化讲义的效果,因此我们在PA讲义中尽可能少地提到“一生一芯”,而把“一生一芯”的相关内容都放到“一生一芯”本身的讲义中,如果不得不提这些麻烦,不仅会使我们维护讲义时感到困难,而且大家阅读讲义时也不知道应该到哪里寻找相关的内容。

安装系统是独立解决问题的最简单的训练

如果你是第一次安装并使用Linux,你可能会遇到非常多的问题,不用担心,因为全世界都在使用Linux,因此你遇到的问题,很大概率别人也遇到过,在互联网上搜索关键字,很大概率就能找到解决方案。

树立正确的价值观,接受最大程度的训练

B站账号: 一生一芯-视频号

第五期一生一芯 | 周六 19:00-21:00 | Sat 08 Oct 2022 08:37:33

课程主页 <https://ysyx.oscc.cc/docs/>

NEMU代码导读

echo > C https://ysyx.oscc.cc/slides/2205/06.html#NEMU代码-现代方法

CC := clang++

LD := \$(CC)

INCLUDES := \$`INC_PATH`

CFLAGS := -O2 -WMD -Wall -Werror \$(INCLUDES) \$(CFLAGS)

LDFLAGS := -O2 \$(LDFLAGS)

OBJS := \$(OBJS)%:c=\$(OBJ\_DIR)/%.o \$(CXXSRC)%.cc=\$(OBJ\_DIR)/%.o

# Compilation patterns

\$(OBJ\_DIR)/%.o: %.c

@echo + CC \$<

@mkdir -p \$(dir \$@)

@\$(CC) \$(CFLAGS) -c -o \$@ \$<

\$call call\_fixdep, \$(@:o=d), \$@

-- INSERT --

ifeq \$(INSTPAT\_INST(s)) ((s)->isa.inst\_val)

#define INSTPAT\_MATCH(s, name, type...) { decode\_operands, &dest, &src1, &src2, &mimm, conc

at(TYPE\_, type); \_VA\_ARGS\_ ; }

{ const void \*\* \_\_instpat\_end = &s\_\_instpat\_end;

do { uint64\_t key, mask, shift; pattern Decode("?????? ???? ??0101 11",

sizeoff"?????? ???? ??0101 11", &key, &mask, &shift); if (((((s)->isa.inst\_val >> shift) & mask) == key) && decode\_operands, &dest, &src1, &src2, &mimm,

TYPE\_U); (cpu.gpr[check\_reg\_idx(dest)]) = s->pc + lmm; } goto \_\_instpat\_end; } while

! 0;

do { uint64\_t key, mask, shift; pattern Decode("?????? ???? 00000 11",

sizeoff"?????? ???? 00000 11", &key, &mask, &shift); if (((((s)->

解密黑科技 - 现代方法

使用工具查看宏展开结果

回顾: 使用gcc的-E参数可以输出预处理结果

• 但直接编译会报错: 找不到头文件

解决方案: 在Makefile文件的编译规则中添加命令

# Compilation patterns

\$(OBJ\_DIR)/%.o: %.c

@echo + CC \$<

@mkdir -p \$(dir \$@)

@\$(CC) \$(CFLAGS) -c -o \$@ \$<

\$call call\_fixdep, \$(@:o=d), \$@

展开的结果不好阅读

• 使用代码格式化工具

\$(CC) \$(CFLAGS) -E -MF /dev/null

课件讲解+代码讲解+代码演示

# “一生一芯” 资源

- “一生一芯” 主页 - [ysyx.org](http://ysyx.org)
- 课程主页 - [ysyx.org/docs/](http://ysyx.org/docs/)
  - 包含报名链接, 常见问题, 课件, 讲义
  - 以及直播链接, 录播链接等



- B站账号 - 一生一芯-视频号
  - B站主页
    - <https://space.bilibili.com/2107852263>
  - 第五期录播集合
    - <https://space.bilibili.com/2107852263/channel/collectiondetail?sid=690279>
  - 第四期学生心得分享集合
    - <https://space.bilibili.com/2107852263/channel/collectiondetail?sid=1173655>

# 助教团队(部分)

- 选拔学生担任助教，支撑超过百名同学的学习
- 引导学生思考，而不是直提供解决方案
- 给学生最大的成长机会

部分助教名单(2023年6月时)

序号	姓名	学校	年级
1	苗金标	中国科学技术大学	研二
2	段震伟	中国科学技术大学	研三
3	刘汉章	太原理工大学	大三
4	曹勋	中国科学技术大学	研二
5	杨海帆	浙江工商大学	大四
6	曹世洋	中国科学技术大学	研二
7	倪仁涛	东北大学	研一
8	魏人	兰州大学	大四
9	吴佳宾	青岛大学	研一
10	陈璐	中国科学院大学	博一
11	粟金伦	太原理工大学	大四



苗金标



段震伟



刘汉章



曹勋



杨海帆



曹世洋



倪仁涛



魏人



陈璐



粟金伦



吴佳宾

# 助教团队(第四/第五期)

- 助教工作: 技术答疑, 组织组会和考核, 了解学生的学习进展
- 第四期助教
  - 线下助教: 邓海文(海南大学), 段震伟(中国科学技术大学), 刘一鸣(华中科技大学), 梅晓龙(海南大学)
  - 线上助教: 陈泱宇(重庆大学), 冯浩原(中国科学院大学), 李志锐(厦门大学), 刘知杭(四川建筑职业技术学院), 罗昊洋(北京大学), 倪仁涛(东北大学), 粟金伦(太原理工大学), 曾广森(华南理工大学)
- 第五期助教
  - 线下助教: 曹勋(中国科学技术大学), 曹世洋(中国科学技术大学), 邓海文(海南大学), 段震伟(中国科学技术大学), 刘汉章(太原理工大学), 卢非凡(西安财经大学), 马建露(北京大学), 梅晓龙(海南大学), 苗金标(中国科学技术大学)
  - 线上助教: 陈泱宇(重庆大学), 李志锐(厦门大学), 罗昊洋(北京大学), 倪仁涛(东北大学), 粟金伦(太原理工大学), 孙际儒(南京大学), 魏人(兰州大学), 吴佳宾(青岛大学), 徐鑫(山东交通学院), 曾广森(华南理工大学), 赵博涵(杭州电子科技大学)

# 设置代码考核环节

- 考核软硬件系统观和工具使用等技能
- 考核独立解决问题的能力

所以

我们来玩真的: 在线调试考核, 通过才能获得流片机会

- 助教在你提交的项目中随机注入3个bug 😈
  - 涵盖硬件, 软件和环境
- 你需要在30分钟内排除bug 😈
  - 强迫大家理所有细节, 吸收科学的调试方法
    - 参考代码, 抱大腿, “看波形就够” 都没戏 😈
  - 抓紧每一次调试锻炼的机会
    - 突击是没用的, 考的就是真本事 😈
    - 抱一次大腿 = 少一次训练

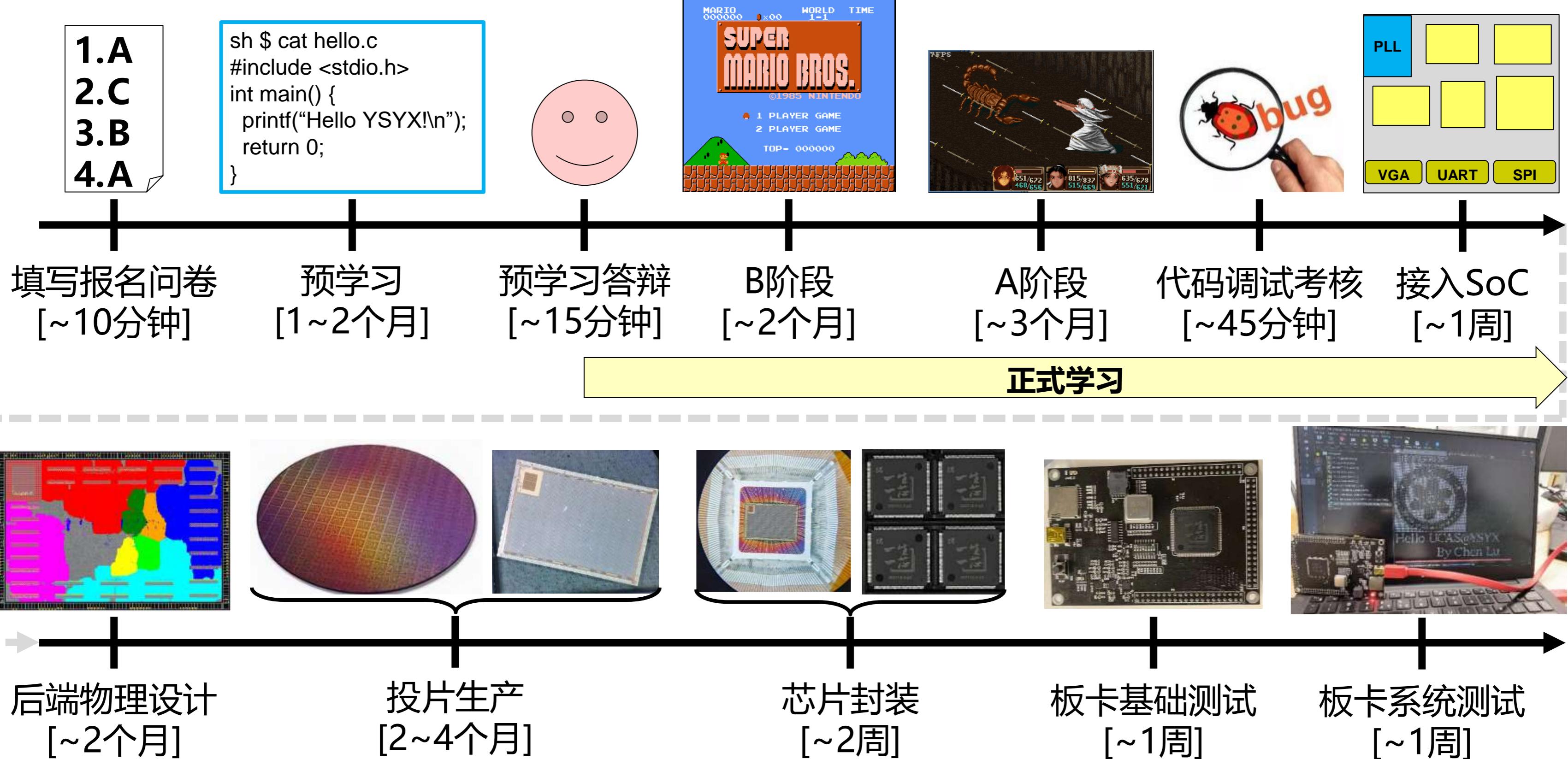


一下子Accepted不了吧

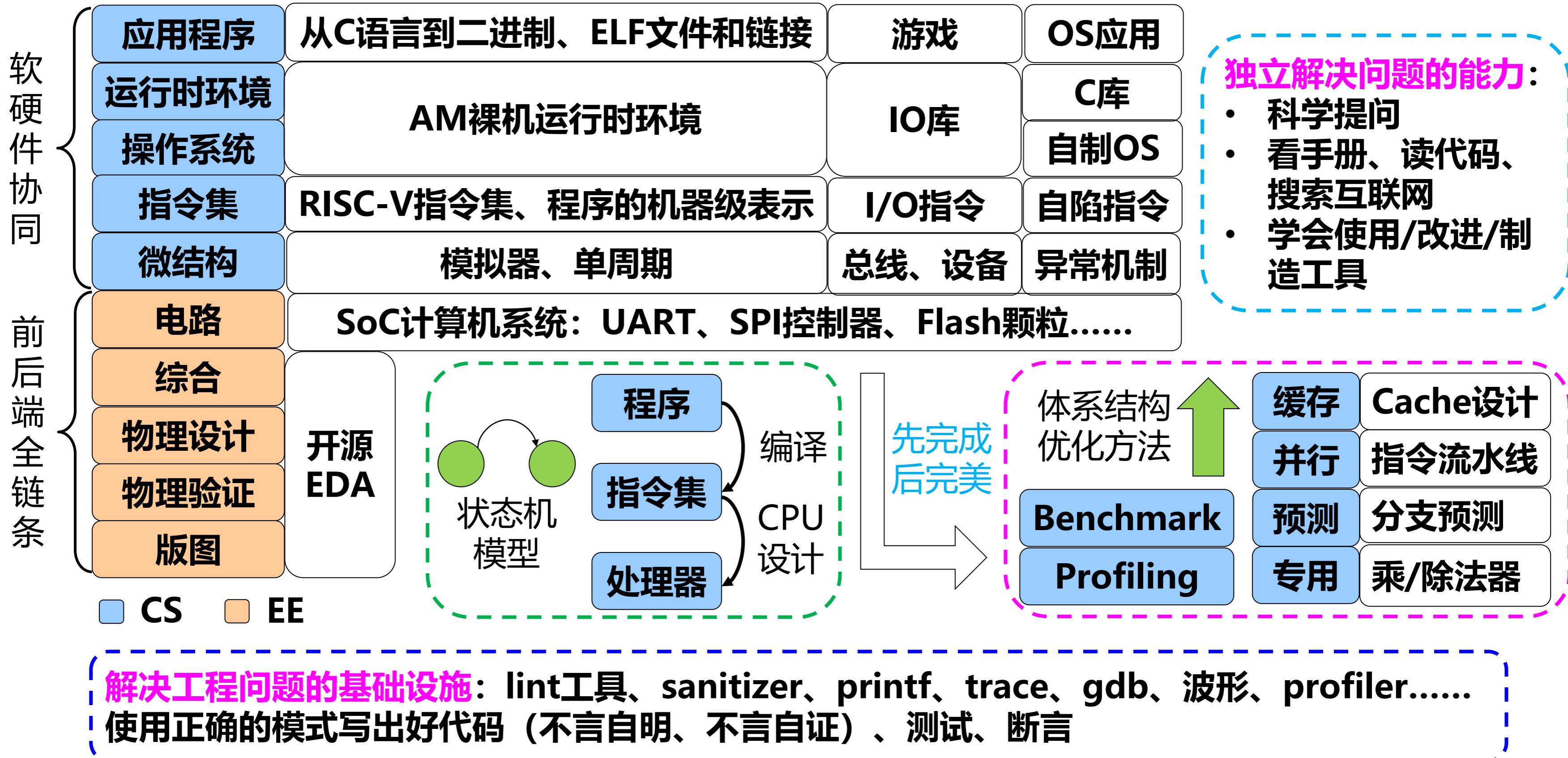
这和大家将来遇到的实际问题非常接近

- 如果你能解决实际问题, 你就是专家

# 第四/五期学习路线图

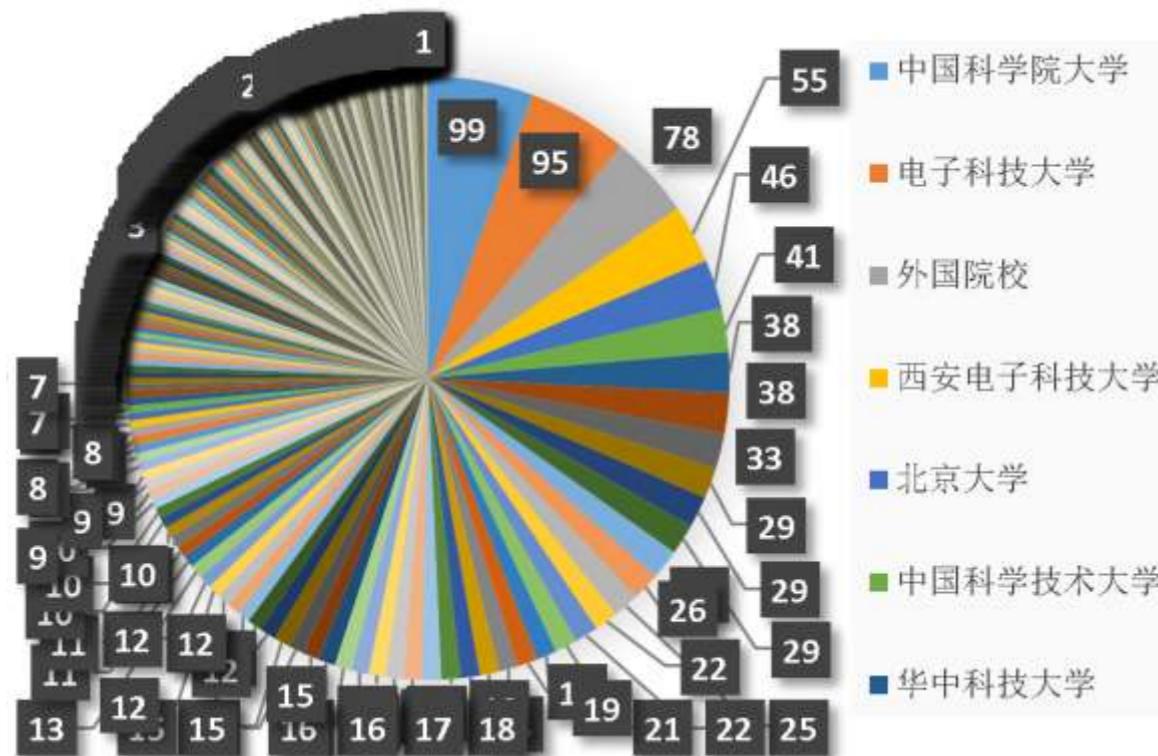


# 知识图谱

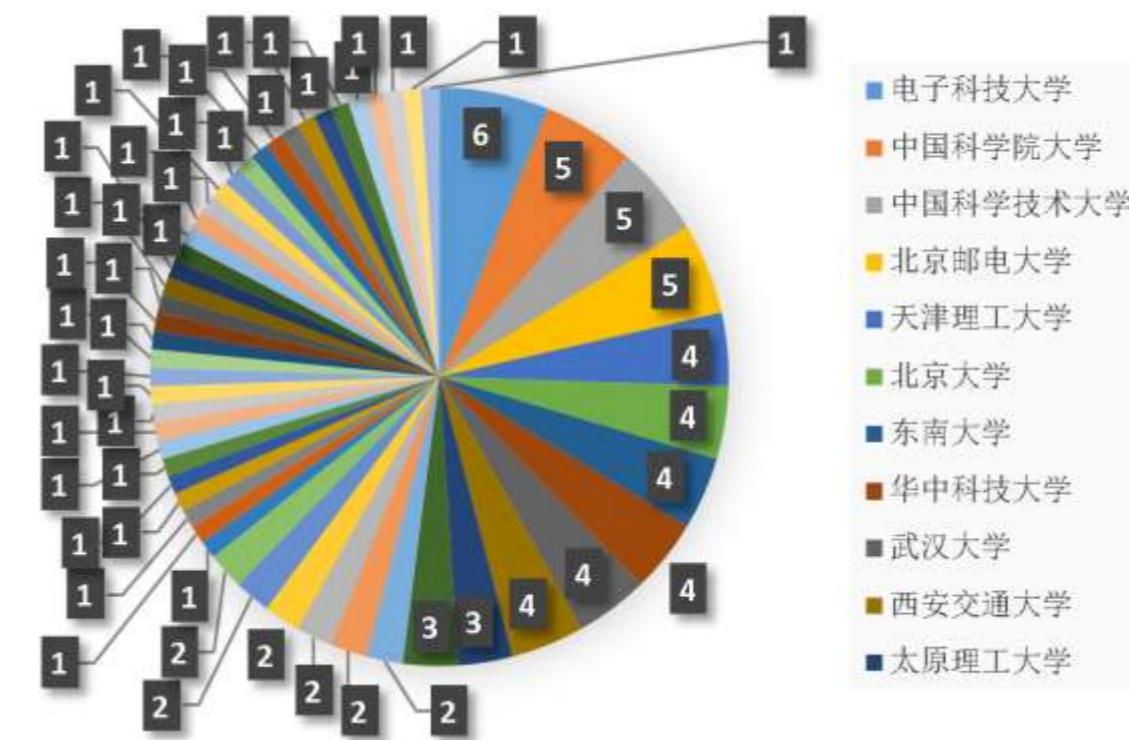


# 第四期“一生一芯”学习情况统计(2023年2月)

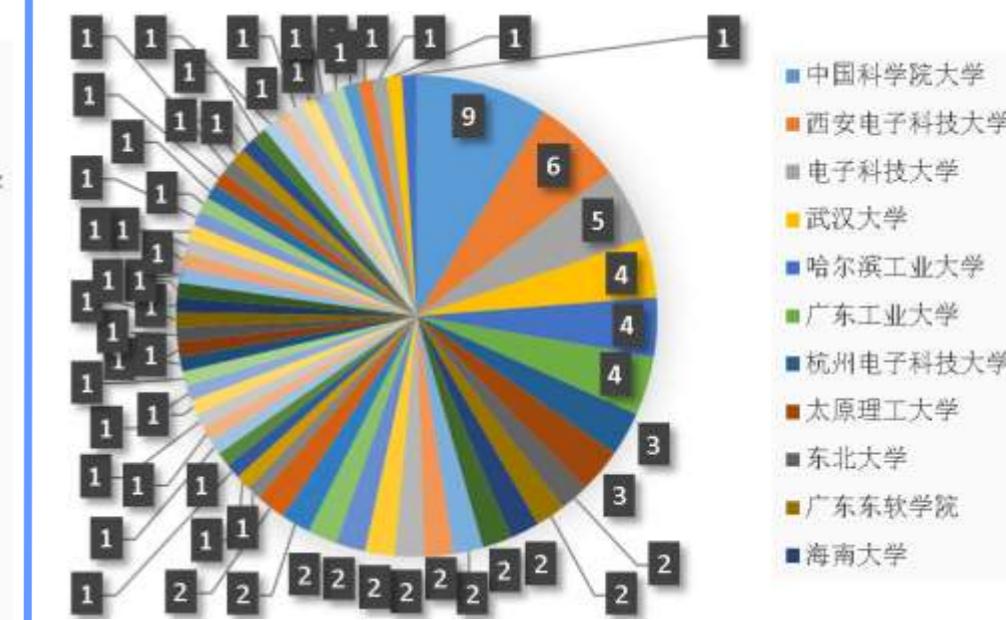
报名(1753人)



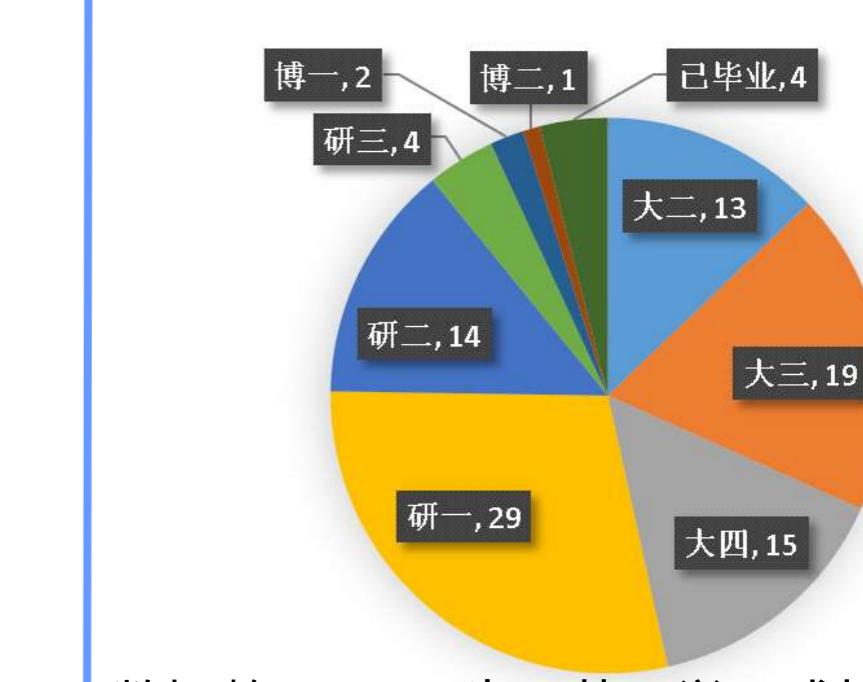
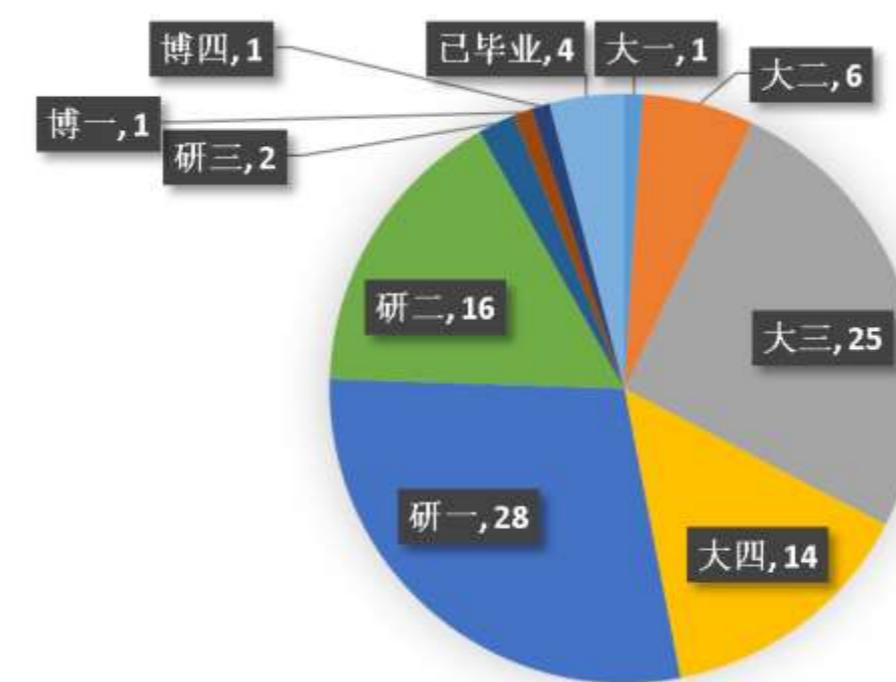
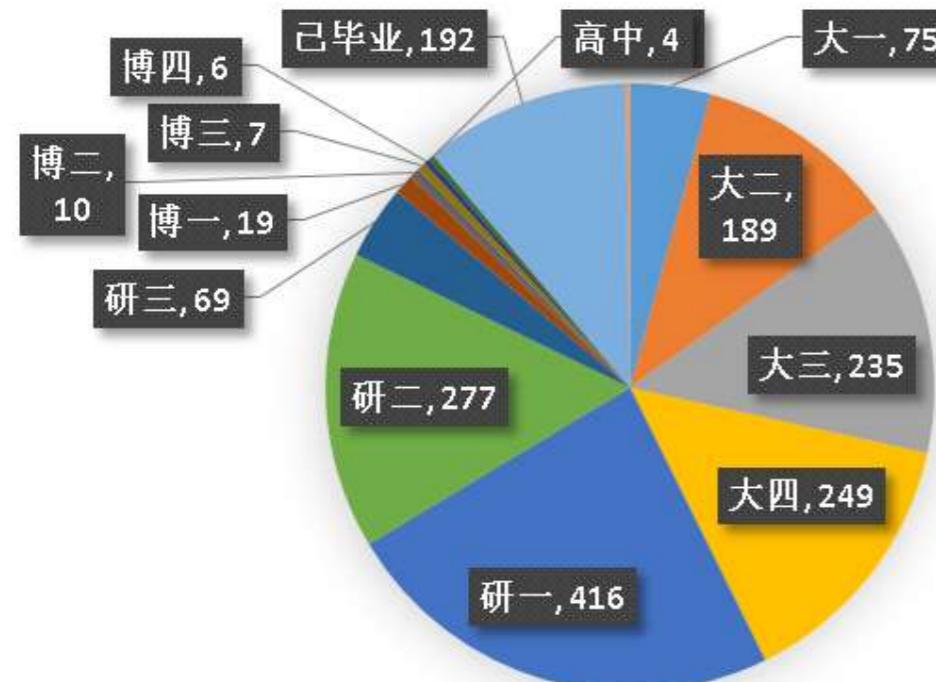
学习到B阶段(98人)



学习到A阶段(101人)



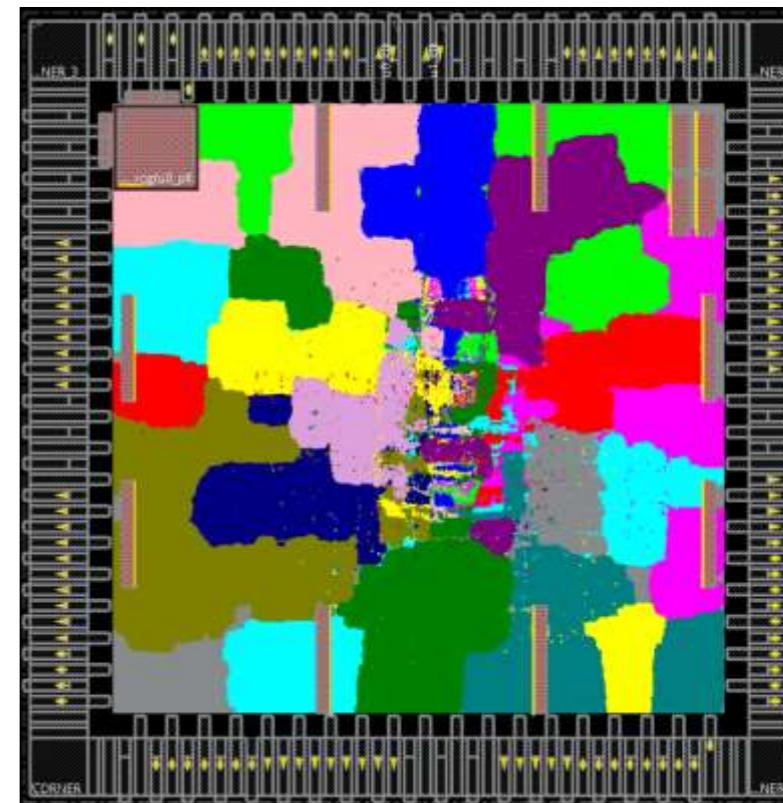
高校分布



数据整理：“一生一芯”学习成长追踪小组

# 第四期“一生一芯”流片名单

序号	学号	学校	专业	年级 (报名时)
1	ysyx_22040053	南京大学	计算机科学与技术	大二
2	ysyx_22040066	南京大学	计算机科学与技术	大二
3	ysyx_22040656	中国计量大学	计算机科学与技术	大二
4	ysyx_22040163	南通大学	计算机科学与技术	大二
5	ysyx_22040091	中国科学院大学	计算机科学与技术	大三
6	ysyx_22040596	华南理工大学	电子科学与技术	大四
7	ysyx_22041812	南方科技大学	电子科学与技术	研零6月
8	ysyx_22040127	东南大学	人工智能	研一
9	ysyx_22040654	福州大学	集成电路工程	研一
10	ysyx_22040978	中国科学院大学	电子信息	研一
11	ysyx_22041514	杭州电子科技大学	计算机技术	研一
12	ysyx_22040213	中国科学院大学	微电子学与固体电子学	研二
13	ysyx_22040561	北京大学	集成电路工程	研二
14	ysyx_22041461	四川大学	微电子科学与工程	大二
15	ysyx_22040886	北京理工大学	电子信息	大三
16	ysyx_22050228	东北大学	电子科学与技术	大三
17	ysyx_22050920	杭州电子科技大学	电子科学与技术	大四
18	ysyx_22040501	哈尔滨工业大学	信息与通信工程	研一
19	ysyx_22050133	北京大学	机械	研二



第一批流片SoC

- ysyx\_040053
- ysyx\_040066
- ysyx\_040091
- ysyx\_040127
- ysyx\_040163
- ysyx\_040213
- ysyx\_040561
- ysyx\_040596
- ysyx\_040654
- ysyx\_040656
- ysyx\_040978
- ysyx\_041514
- ysyx\_041812

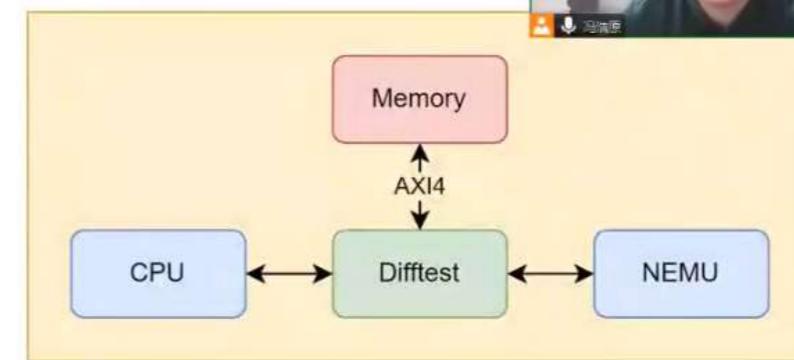
# 第四期“一生一芯”学生学习心得分享

- 冯浩原@中国科学院大学, 计算机科学与技术, 报名时大三
  - 大四时在北京开源芯片研究院实习, 成为“香山”团队的技术骨干
  - <https://www.bilibili.com/video/BV1C54y1T7hw>



## 心得分享①：理解系统

- “一生一芯”仿真系统
  - 逐步建立了对系统的认识, 增加调试的信心与效率
  - 仔细阅读源码和手册说明
    - 通过阅读源码, 了解 NEMU 和 Spike 的对比机制
    - 接入 YsyxSoC 后, 为什么无法从 SRAM 中读出数据
- “香山”团队中的学习、工作
  - 处理器架构和验证框架复杂很多
  - 主要负责 MMU 模块的开发
    - 涉及到和前端取指、后端访存、内存访问的交互
    - 包含 L1 TLB、L2 TLB、PMP、PMA 等子模块
  - 实践一生一芯的系统观念
    - 从整体入手, 关注前端取指和后端访存对 MMU 的查询行为
    - 关注细节, 理清 MMU 对于页表查询请求的处理过程
    - 仔细阅读源码, 不放过每一处内容, 对模块充分掌握



“一生一芯”仿真系统示意图

端口	说明
output [127:0] Q	读数据
input CLK	时钟
input CEN	使能信号, 低电平有效
input WEN	写使能信号, 低电平有效
input [5:0] A	读写地址
input [127:0] D	写数据

YsyxSoC 中对 SRAM 行为的描述



# 第四期 “一生一芯” 学生学习心得分享

## 成长2: Debug能力的提升——difftest的应用



- 将difftest的思想应用到其他科研项目中
- 帮助我快速验证算法的正确性，节省大量时间

nemu中的difftest

```
Welcome to Riscv64-NEMU!
For help, type "help"
gpr[ 8] different after executing inst at pc = 80000000 , right = 0 wrong = 1
pc 0x0000000000000004
ra 0x0000000000000000 sp 0x0000000000000000 gp 0x0000000000000000
t0 0x0000000000000000 t1 0x0000000000000000 t2 0x0000000000000000
s0 0x0000000000000000 s1 0x0000000000000000 a0 0x0000000000000000 a1 0x0000000000000000
s0 0x0000000000000001 a2 0x0000000000000000 a3 0x0000000000000000 a4 0x0000000000000000 a5 0x0000000000000000
a6 0x0000000000000000 a7 0x0000000000000000 s2 0x0000000000000000 s3 0x0000000000000000
s4 0x0000000000000000 s5 0x0000000000000000 s6 0x0000000000000000 s7 0x0000000000000000
s8 0x0000000000000000 s9 0x0000000000000000 s10 0x0000000000000000 s11 0x0000000000000000
t2 0x0000000000000000 t4 0x0000000000000000 t5 0x0000000000000000 t6 0x0000000000000000
utvec :0000000000000000
mepc :0000000000000000
mstatus:000000000001800
mcause :0000000000000000
[src/cbu/cpu-exec.c:242 cp0_exec] need: ABORT at pc = 0x0000000000000000
```

2023/2/3

博学 慎思 明辨 笃行

SpMV算法科研项目中的difftest

```
check row 0, ans = 17.000000, res = 17.000000, pass
check row 1, ans = 35.000000, res = 35.000000, pass
check row 2, ans = 21.000000, res = 10.000000, wrong
check row 3, ans = 16.000000, res = 16.000000, pass
check row 4, ans = 56.000000, res = 56.000000, pass
check row 5, ans = 33.000000, res = 33.000000, pass
check row 6, ans = 18.000000, res = 18.000000, pass
```



## 体会2: KISS

- Keep It Simple, Stupid 从易到难、逐步推进
- 将复杂的目标进行分解，为自己做好阶段性的规划
- 勤写代码勤踩坑



我的任务分解方案



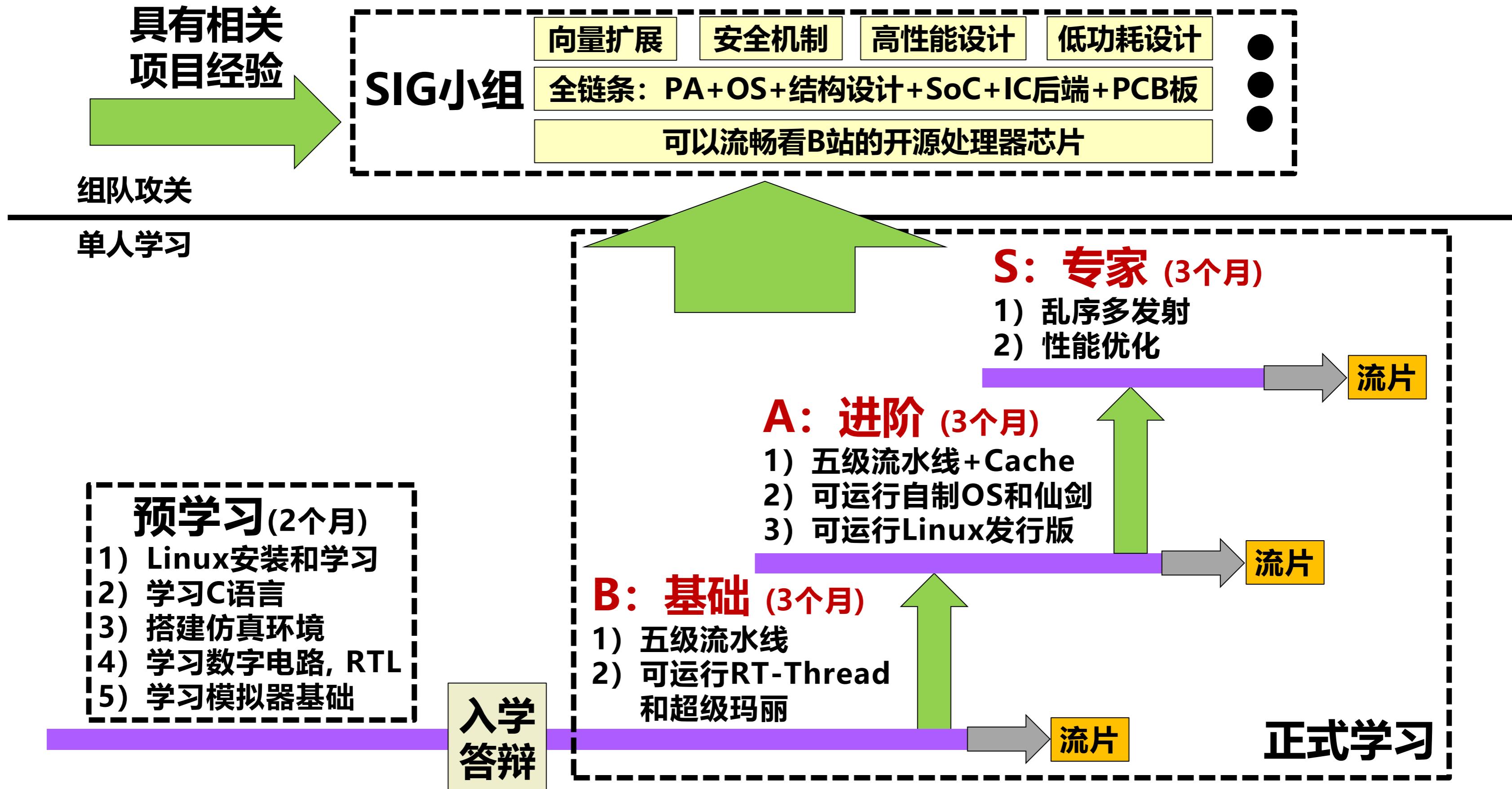
曾广森@华南理工大学  
电子科学与技术, 报名时大四  
**收获: 将学到的方法和思想应用到科研项目**  
<https://www.bilibili.com/video/BV1sY41167kE/>

王晨宇@南通大学  
计算机科学与技术, 报名时大二  
**收获: 学会分解复杂目标, 逐步解决新问题**  
<https://www.bilibili.com/video/BV1g24y1G7Hw/>

# 第六期新增B阶段流片，提升A阶段指标

	第三期	第四/五期	第六期(B阶段)	第六期(A阶段)
ISA	RV64I	RV64IM	RV32E	RV64IMAC
处理器	流水线	流水线	流水线	流水线
乘除法器	-	有	-	有
异常	M模式ecall	M模式ecall	M模式ecall	M/S/U所有异常
分页	-	-	-	有
中断	时钟	时钟	-	时钟, 软件, 外部
总线	AXI4	AXI4	AXI4	AXI4
Cache	-	4KB指令+4KB数据	-	16KB指令+16KB数据
IPC	-	-	-	高于X
主频	-	-	-	高于Y
目标程序	RT-Thread	自制OS+仙剑	RT-Thread	Linux内核+发行版
系统软件	-	模拟器, AM, 自制OS	模拟器, AM	模拟器, AM, 自制OS
环境和工具	-	DiffTest, trace, 仿真环境.....	DiffTest, trace, 仿真环境.....	DiffTest, trace, 仿真环境.....

# 学习阶段划分



# 第六期新增开源EDA

- 开源EDA团队设计了多个点工具(黄色部分)[蓝色部分是将来工作]

- 大家是开源EDA工具的首批用户

- 7月：综合(yosys)+时序分析工具

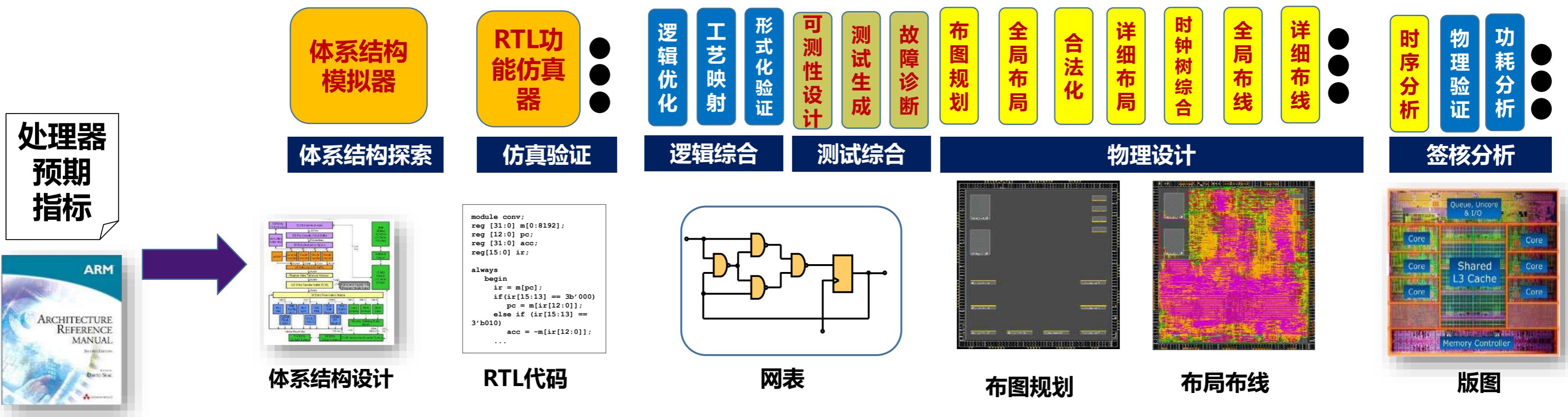
- 10月：发布其他点工具，可线下学习后端流程

- 12月：芯片设计云平台上线，基于开源EDA工具进行可流片设计

开源芯片技术生态论坛 (原“一生一芯”技术论坛)

Session2: 开源EDA

8月25日14:00-15:15, 北京市香格里拉饭店三层翡翠厅



# 第六期“一生一芯”火热报名中

扫码或登录 [ysyx.org](http://ysyx.org) 点击“立即参与”



The screenshot shows the homepage of the [ysyx.org](http://ysyx.org) website. At the top, there is a navigation bar with links for '主页' (Home), '公告' (Announcements), '文档' (Documents), '讨论区' (Forum), '学习记录' (Learning Record), '注册' (Register), and '登录' (Login). The main content area features a large graphic with the text '一生一芯' (One Life, One Chip) in blue and red, followed by the text '从零开始创造属于你的 RISC-V® 处理器' (Start from scratch to create your own RISC-V® processor). Below this is a large green arrow pointing right, followed by a blue button labeled '立即参与' (Participate Now). There are also buttons for '学习记录' (Learning Record) and '赞助' (Sponsorship). At the bottom of the page, there is a section labeled '合作/赞助机构' (Cooperation/Sponsorship Institutions).

官方网站 [ysyx.org](http://ysyx.org)

## 六、“一生一芯”社区活动

# 第六期新增入学大礼包

- 通过入学答辩，进入正式学习，获得入学大礼包一份！
- 内容丰富，专属于“一生一芯”正式学员

入学大礼包



定制文化衫一件



“一生一芯”徽章

“一生一芯”  
正式学员学号



定制双肩包一个

# 毕业成果

- 完成“一生一芯”学习，可获得芯片、板卡以及证书
- 随附毕业大礼包（具体内容待定）

## 毕业收获



芯片



板卡



认证证书

# 积分兑换周边

- 参加志愿活动，积累积分，可兑换周边
- 大家可以**发起提案**，增加新的文化周边
- 提案被采纳后，**有额外奖励**

## 积分任务（部分）

### 提议新的文化周边

参与“一生一芯”调研问卷

“一生一芯”官方录用文章

“一生一芯”官方录用分享视频

“一生一芯”官方录用报告视频

“一生一芯”分享报告

组织“一生一芯”巡回演讲

完成B/A阶段学习的认证

任务内容持续更新中



(效果图)

## 可兑换周边（部分）

“一生一芯”文化衫

RISC-V文化衫

“一生一芯”帆布包

“一生一芯”双肩包



更多周边陆续上线，尽请期待

# SIG小组 – 组队攻坚感兴趣的项目

- 已成立的SIG小组

开源芯片技术生态论坛（原“一生一芯”技术论坛），Session3：“一生一芯” SIG小组  
8月25日15:15-15:40，北京市香格里拉饭店三层翡翠厅

- 高性能体系结构模拟器——设计用于体系结构探索的模拟器
- 高性能RTL仿真器——目标性能优于Verilator
- 开源处理器核与外围IP——为“一生一芯” SoC提供丰富功能
- 开源芯片数据集——收集并运行现有的开源芯片项目，为开源EDA提供数据集
- 国际交流与翻译——传播开源芯片国际最新动态

- 有想法但还没开设的SIG小组(需要有组长)

- 看B站——设计一颗可流畅看B站的开源处理器芯片
- 芯片上天——与航天院所合作，学习设计高可靠的处理器芯片，所设计芯片产品有机会在轨验证！
- 二进制翻译——在RISC-V处理器上玩windows游戏
- ..... 大家如果有新想法，可联系我们进行立项讨论，通过后可提供经费支持！

# 开源社区贡献 – RVFA认证考试中文翻译



RISC-V翻译工作组 (18)

刘汉章  
这种语序的调整是可以的吧，但是忽略了它原本的一些连词 (thus) 但是更符合中国人的语言好像

刘汉章  
因此，一个 I

刘汉章  
针对正确地

刘汉章  
我觉得没问题，在不丢失原本语义的情况下，可以按照中文习惯进行适当地改造。

## 翻译小组斟酌用词

文对照表已经整理好了 sheet/DRVFPRES2，但是由于缺乏具体的语境，有一些单词没办法识别是动词还是名词，所以中文只能作为参考，要具体问题具体分析（翻译有问题可直接在线修改）。

2、昨天提到的每篇文档中都有的内容或者跟格式强相关得标准我统一放到这个文档里面 (<https://docs.google.com/document/d/1TgFLATI4hb3sC37cvUWBeoSu75iQL9P9q73WcqWtdZ8/edit#heading=h.320f465ob4d3>)，如果大家没有异议的话，可以直接复制到自己的文档中（翻译有问题可直接在线修改）。

3、像rs1、jar、opcode、qemu这种专有名词我觉得保持原样就好，像ISA、IPC这种由几个单词首字母拼接在一起的专有名词，可以像昨天讨论的那样翻译时保持原样。关于是否在第一次出现的地方添加括号解释，我觉得可以最后等到Review时由一个人统一添加一下会比较好。

### Chapter 1: RISC-V Overview

Conventions used in this file:

- Heading 1 is used to mark the beginning of a course page.
- Heading 2 is used for subtitles within a page.
- Bold for references to buttons or menu options, and first sentences in bullet points.
- Bold Italics for introducing new terms.

#### According to LF Author's Guide:

- Bold Consolas Font** (black) for source code.
- Bold Consolas Font** (brown) for file and folder names.
- Bold Consolas Font (Dark Blue)**: Text typed at the command line.
- Bold Consolas Font (Green)**: Output.
- Hyperlinks** are left in GoogleDocs format.

### Learning Objectives

By the end of this chapter, you should be able to:

- Understand what RISC-V is and what it is not.
- Identify the characteristics of an ISA to decide whether it is a CISC or a RISC type ISA.
- Describe the history of RISC-V.
- Identify the most notable differences between RISC-V and the leading commercial ISAs, like x86 and ARM.
- Understand the structure and operation of RISC-V International.
- Analyze the documentation of RISC-V specifications.
- Explore ways to contribute to the RISC-V effort.

### Chapter Introduction

In this chapter, you will get acquainted with RISC-V. Please note that you are expected to already be familiar with computer architecture and to have some exposure to some specific ISA. We will provide some refreshers on basic terms, but this is certainly not an introduction to computer architecture.

Among the topics we will cover, you will learn about the history of RISC-V and how it differs from today's dominating ISAs. You will also get the chance to browse through the vast documentation of RISC-V, and you will learn how the documents are organized.

You will be introduced to RISC-V International and the RISC-V ecosystem, and you will learn about quite a few different ways you may contribute to the RISC-V community.

Let us get started!

### History of RISC-V: The Free and Open ISA

RISC-V (pronounced "risk-free") is an open standard instruction set architecture (ISA) based on principles, enabling a new era of processor innovation through open standard collaboration.

RISC stands for Reduced Instruction Set Computer, a computer architecture proposed in 1980 based on simplicity, as opposed to current microprocessors at the time, dubbed Complex Computers, or CISC. The RISC architecture was born in an academic environment, so the design was motivated by commercial interests at the time. RISC is the opposite of CISC in many ways.

原文和译文

### 第1章：RISC-V概述

#### 本文中使用的约定文件：

- 1级标题用于标记课程页面的开始。
- 2级标题用于页面内的子标题。
- 粗体 用于按钮或菜单选项的引用，以及要点中的第一句话。
- 粗体斜体 用于引入新术语。

#### 根据 LF 作者指南：

- Bold Consolas 字体（黑色）** 用于源代码。
- Bold Consolas 字体（棕色）** 用于文件和文件夹名称。
- Bold Consolas 字体（深蓝色）**：命令行文本。
- Bold Consolas 字体（绿色）**：输出。
- Hyperlinks** 保留为 GoogleDocs 的格式。

#### 学习目标

到本章结束时，您应该能够：

- 了解什么是 RISC-V，什么不是。
- 识别 ISA (instruction set architecture，指令集体系结构) 的特征以决定它是 CISC 还是 RISC 类型的 ISA。
- 描述 RISC-V 的历史。
- 确定 RISC-V 与领先的商业 ISA (如 x86 和 ARM) 之间最显著的区别。
- 了解 RISC-V 国际基金会的组织结构和运作模式。
- 分析 RISC-V 规范。
- 探索为 RISC-V 工作做出贡献的方法。

#### 章节介绍

在此章中，您将开始了解 RISC-V。请注意，在阅读前你应该已经熟悉计算机体系结构并接触过某些特定的 ISA。我们将提供一些基本术语的复习，但很显然这不是对计算机体系结构的介绍。

在我们将涵盖的主题中，您将了解 RISC-V 的历史以及它与当今占主导地位的 ISA 有何不同。您还将有机会浏览 RISC-V 文档，并了解这些文档的组织方式。

您将了解 RISC-V 国际基金会和 RISC-V 生态系统，并且您将了解可以为 RISC-V 社区做出贡献的多种不同方式。

让我们开始吧！

#### RISC-V的历史：自由开放的ISA

RISC-V (pronounced "risk-free") 是一种基于 RISC 原则的开放标准指令集架构 (ISA)，通过开放标准协作促进了创新。

指令集计算机，这是一种 1980 年代初期提出的基于简单性的计算机体系结构，与当时被称为 CISC (Complex Instruction Set Computer) 的现代处理器相对。RISC 架构诞生于学术环境。因此在设计上力求简洁和高效，同时接受商业利益驱动的 CISC 架构并不相同的特性。RISC 在许多方面与 CISC 颠然不同。CPU 有几个寄存器和很多指令，其中大部分指令可以访问内存，而 RISC CPU 有很多不同的指令的组合。内存访问仅限于一些加载和存储指令。

姓名	学校/单位	年级
缪宇飚	中科院计算所	- (组长)
苗金标	中国科学技术大学	研二
段震伟	中国科学技术大学	研三
刘汉章	太原理工大学	大三
曹勋	中国科学技术大学	研二
杨海帆	浙江工商大学	大四
曹世洋	中国科学技术大学	研二
倪仁涛	东北大学	研一
魏人	兰州大学	大四
吴佳宾	青岛大学	研一
陈璐	中国科学院大学	博一
栗金伦	太原理工大学	大四



缪宇飚



苗金标



段震伟



刘汉章



曹勋



杨海帆



曹世洋



倪仁涛



魏人



陈璐



栗金伦



吴佳宾

# 月度报告和技术论坛

- **月度报告**: 7月开始，每月第一个周末，汇报项目进展，举行入学仪式等
- **技术论坛**: 每年组织 1~2次技术论坛（独立或联合其他主题）
- **活动预告**: 8月25日，开源芯片技术生态论坛（原“一生一芯”技术论坛）
  - 地点: 北京市香格里拉饭店三层翡翠厅
  - 包含专家邀请报告，“一生一芯”技术分享等

祝贺北京一零一中学的烟雨松  
同学于2023年2月通过预学习答辩，  
成为“一生一芯”计划  
**第一位**进入正式学习的**高中生**！



包老师报告



第一届论坛大合影



第一位高中生正式学员

# 暑期夏令营活动及学员去向



2022年6月-8月，“一生一芯”暑期夏令营  
地点：北京开源芯片研究院

学习深造	就业
卡耐基梅隆大学	北京开源芯片研究院
佛罗里达大学	华为海思
香港科技大学	比特大陆
中科院计算所	北京嘉楠捷思
中科院微电子所	北京微核芯
清华大学	进迭时空
复旦大学	中科院计算所
南京大学	鹏城国家实验室
浙江大学	.....

# 2023年度暑期夏令营

- 时间：7月~8月，上海处理器技术创新中心
- 近距离认识芯片，高密度学习芯片

姓名	学校	专业	年级
烟雨松	北京一零一中	-	高三
曾宇航	东华理工大学	计算机科学与技术	大一
张宇驰	太原理工大学	电气工程及其自动化	大一
袁永强	西安邮电大学	电子信息工程	大二
潘岩	哈尔滨工业大学	计算机科学与技术	大二
孟沛	电子科技大学	软件工程	大二
王翩	上海科技大学	计算机科学与技术	研零
唐德宇	中国科学技术大学	软件工程	研一
丁旻昊	中南大学	电子科学与技术	研一
赵树钰	中国科学技术大学	软件工程	研一



# “一生一芯”官网及公众号

The screenshot shows the official website for the '一生一芯' project. At the top left is the logo '一生一芯'. The top navigation bar includes links for '主页' (Home), '公告' (Announcements), '文档' (Documents), '讨论区' (Forum), '学习记录' (Learning Record), '注册' (Register), and '登录' (Login). The main content area features a large graphic with the text '从零开始创造属于你的 RISC-V® 处理器' (Start from scratch to create your own RISC-V® processor) and the '一生一芯' logo. Below this are buttons for '立即参与' (Join Now), '学习记录' (Learning Record), and '赞助' (Sponsor). A QR code is displayed next to the text '官方网站 [ysyx.org](http://ysyx.org)'.

对写文章、视频剪辑、新媒体运营感兴趣的同学  
欢迎加入宣传SIG小组  
发送简历：[ysyx@bosc.ac.cn](mailto:ysyx@bosc.ac.cn)

The screenshot shows the WeChat public account profile for '一生一芯'. The profile picture is the '一生一芯' logo. The account name is '一生一芯'. Below it, a brief introduction states: '一生一芯是2019年由中国科学院大学发起的公益性人才培养项目，理念就是让一个学生可以带着自己...' (YiSheng-YiXin is a public welfare talent cultivation project initiated by University of Chinese Academy of Sciences in 2019, with the concept being that a student can take themselves...). It also indicates '4篇原创内容' (4 original articles) and '35个朋友关注' (35 friends followed). Below the profile is a QR code with the '一生一芯' logo in the center.

“一生一芯”公众号

# “一生一芯”校园行巡回演讲活动

- 2023年3月至今，“一生一芯”深入校园，开展多场巡讲活动
- 秋季学期，我们的行程还会继续，尽情期待！
  - 如果想邀请“一生一芯”去你的学校，欢迎同学们随时联系项目组

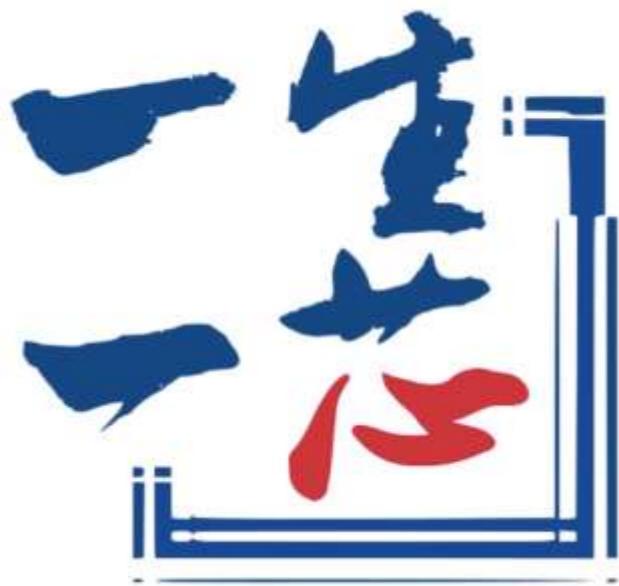
时间	巡讲学校	巡讲内容
3月19日	北京工业大学	宣讲
4月13日	北京科技大学	宣讲+教学交流
5月17日	东北大学（秦皇岛分校）	宣讲
5月25日	青岛大学	宣讲+教学交流
6月10日	天津工业大学	宣讲
6月14日	天津理工大学	宣讲+教学交流
6月16日	太原理工大学	宣讲

更多活动正在排期中，欢迎同学们邀请



# “一生一芯”计划

- 硅上教学：融合本科阶段EE和CS专业知识点的实践项目
- 战略意义：为攻关卡脖子领域培养人才，输送到企业和开源社区
- 公益培养：学习资源全公开，免费学习；在校生免费流片



设计人生中的  
第一颗处理器芯片



官方网站  
[ysyx.org](http://ysyx.org)



“一生一芯”公众号



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



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



计算机系统与处理器  
芯片课程虚拟教研室



北京开源芯片研究院  
BEIJING INSTITUTE OF OPEN SOURCE CHIP



鹏城实验室  
PENGCHENG LABORATORY



中国开放指令生态 (RISC-V) 联盟  
China RISC-V Alliance



ByteDance  
字节跳动



youdao  
有道



上海处理器技术创新中心  
SHANGHAI INNOVATION CENTER FOR COMPUTING TECHNOLOGY

# 七、常见问题

# 大家会问到的问题 - 1

## ■ 加入“一生一芯”需要具备哪些条件？

- 零基础即可加入，不论年级、专业和学校；在校生和已毕业都可以报名学习
- 如果具备计算机或电路方面的基础知识，那当然更好，但并不强制，因为“一生一芯”的预学习和B阶段也会安排相关的学习内容。
- 在关键节点上指引大家，但并不会手把手教学。比如，讲义告诉大家掌握 verilog，会推荐教程，但是并不会安排详细的课堂教学，因为这方面教材和在线课程已经很多

## ■ 流片条件？

- 达到指标，通过答辩考核，即可进入流片，会尽量安排最近一批次班车（涉及拼片）
- 原创性和提升性，代码是自己写的，且参与“一生一芯”有较大提升（或社区服务）
- 流片只支持在校生（应届毕业8.31前提交）；已毕业可提交费用流片（视乎面积）
- 回馈社区：获得流片后，建议参加社区服务（助教和一生一芯的开源项目、科普等）

# 大家会问到的问题 - 2

## ■ 参与方式?

- 以远程参加为主，部分同学会被邀请到现场来做助教（可报名）
- 每周一次会议，查看进展并答疑
- 同学们需要每 1~3 天记录一次学习过程，便于我们指导和跟踪大家工作过程
- 大家需要保持 Git commit, Git log 也是我们跟踪大家学习过程的重要依据

## ■ 研究生是否可以参加? 研究生是在校生，若是国内高校，可以参与学习和流片

## ■ 非在校生是否可以参加?

- 可以参加学习流程，但无法免费流片（现有经费仅可用于支持国内外在校生流片）
- 刚毕业或已毕业但即将入学的同学，视为在校生（比如9月开学，或毕业的8.31前）

## ■ 参加“一生一芯”，是否还可以参加其他比赛?

- “一生一芯”注重的是人才培养，不与任何比赛冲突，大家参与“一生一芯”写的代码，可以拿去参加比赛。如果获奖，非常希望大家能在致谢中，cite一下“一生一芯”

# 大家会问到的问题 - 3

## ■ 是否可以基于现有开源芯片项目修改后提交？

- 不可以，“一生一芯”是一个训练项目，每一行代码必须都是自己写的才行

## ■ 是否可以提高指标？

- 可以，鼓励同学们冲刺更高指标

## ■ 是否可以降低指标？

- 流片指标不会降低，没有达成流片指标者不能进入流片

## ■ 是否可以组队参加？

- 不可以，“一生一芯”是培养性项目，我们希望大家都能获得最大程度的训练
- 组队参加，仅在必要情况下，面向基础较好的同学，针对一部分特定主题和目标的开源项目开放，例如组队做一个以太网IP，此时数字和模拟都需要不同的队员加入
- 大家完成培养，进入社区，或者Sig小组以及开源IP项目，自然就可以跟其他同学组队了

# 大家会问到的问题 - 4

## ■ 参加“一生一芯”的名额，以及流片的名额

- 参加“一生一芯”的名额，目前没有限制，且随到随学。极端情况下，受限于支撑团队的规模，我们会结合报名人数和助教团队的状况，综合考虑后确定。

## ■ 是否达到流片标准就一定可以流片？

- 不一定。达到流片标准是最基础要求，还需评审通过后方可流片
- 极端情况下，如果达到流片标准的数量大幅高于预期，拟评审后择优流片
- **关于流片名额和班车等，项目组承诺Best effort，但并不做确定性保证（目前看来问题不大）**

## ■ 如果被误判作弊怎么办？

- 判断是否作弊的依据是大家日常提交的学习记录，以及Git log
- 如果误判，可向项目组提起申诉，我们会组织二轮专家评审来确定

## ■ 参与“一生一芯”期间，是否可以中间暂停一段时间后再继续？

- 可以，大家可以根据自己的知识基础和时间，适当调整计划；但需要事先声明和请假

# 大家会问到的问题 - 5

## ■ 是否一定要每周汇报，并按每 1~3 天更新一次学习记录？

- 对于能力较好，可冲刺更高指标的同学，可以自己安排
- 对于能力较弱，需要提升的同学，还是请按照如上节奏推进
- 此外，Git log，无论对于哪些同学，都是需要的

## ■ 跟不上队怎么办？

- 一生一芯提供的Schedule，只是一个参考的时间区段。大家可以根据自己的基础能力和可分配的时间，逐步推进就好。我们依旧会安排答疑

## ■ 答疑可以指导哪些问题？

- 会指明方向，告诉大家思路。比如告诉大家需要用Git，但是不会教Git
- 希望大家先主动探索，不到万不得已情况（Bug卡了一礼拜没进展），不会指导太细
- 很多材料课本上和网上都有，大家只要花一些时间去搜索和实践，是可以自己解决的
- **主动探索和动手实践的能力，才是整个“一生一芯”计划培养的核心**

# 大家会问到的问题 - 6

## ■ “一生一芯”一定要使用 Chisel 吗？

- 不要求一定使用 Chisel，大家可以自己选定语言，项目组都会给大家答疑
- VHDL 暂时还不支持，多语言的话，SoC集成的时候也会复杂一些

## ■ “一生一芯”一定要基于 RISC-V？

- 是的，RV64

## ■ 是否可以报名参加助教、SoC 和 IC 后端的工作

- 可以。SoC 和 IC 后端很重要，需要训练一段时间，才能进入项目

## ■ Cache的规格有多大？

- 接入SoC时会跟大家讲

## ■ 报名了第4期，是否还需要报名第5期？

- 不需要，可以按照第4期的讲义继续学习
- 将来A阶段讲义会调整指导方式，但具体的学习任务不会大幅调整