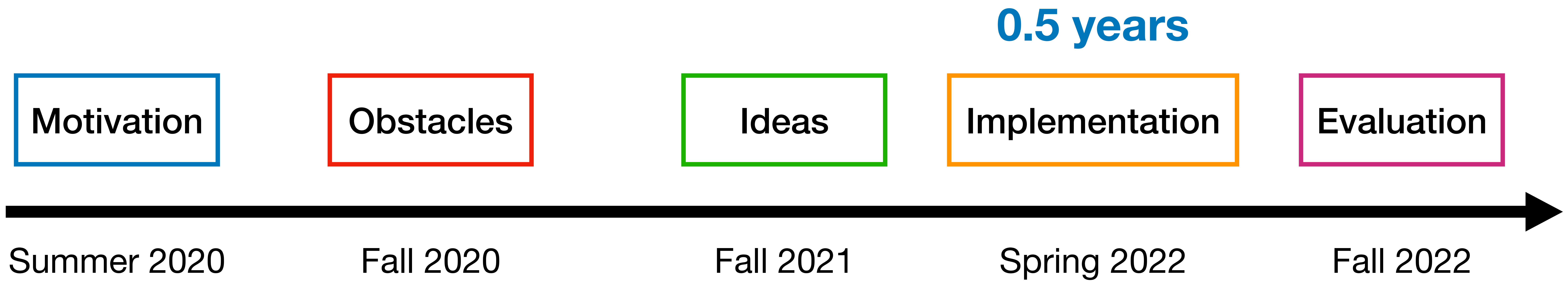
Writing an operating system in 2.5 years

Yunhao Zhang

But first, writing an OS in 0.5 years

- P0: understand computer architecture
- P1: understand context-switch and multi-threading
- P2: understand interrupt and exception
- P3: understand privilege levels and protection
- P4: understand bus and I/O devices
- P5: understand file systems

Why 2.5 years? An overview



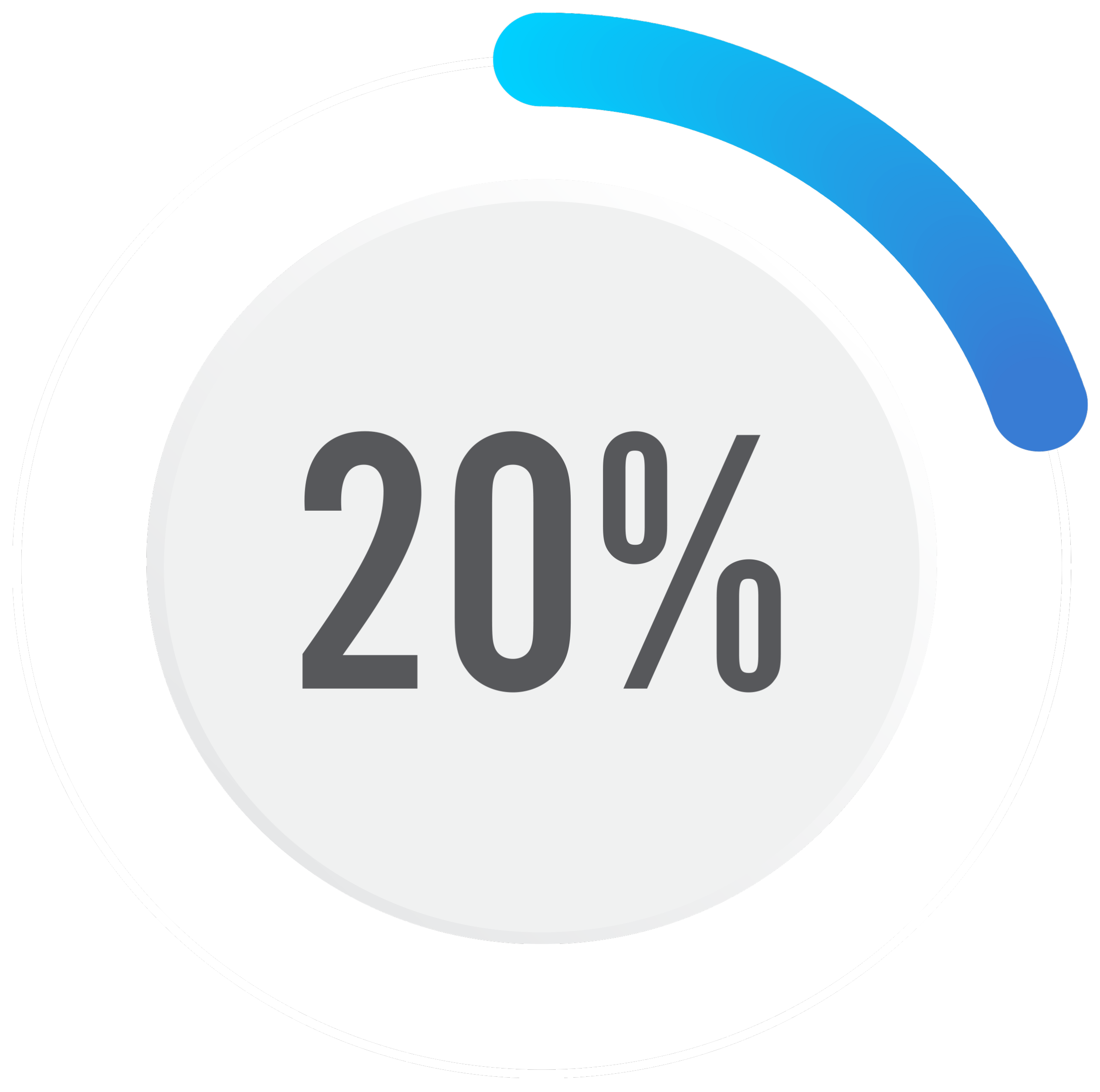
In June 2020, we have

EGOS

~20K lines of code

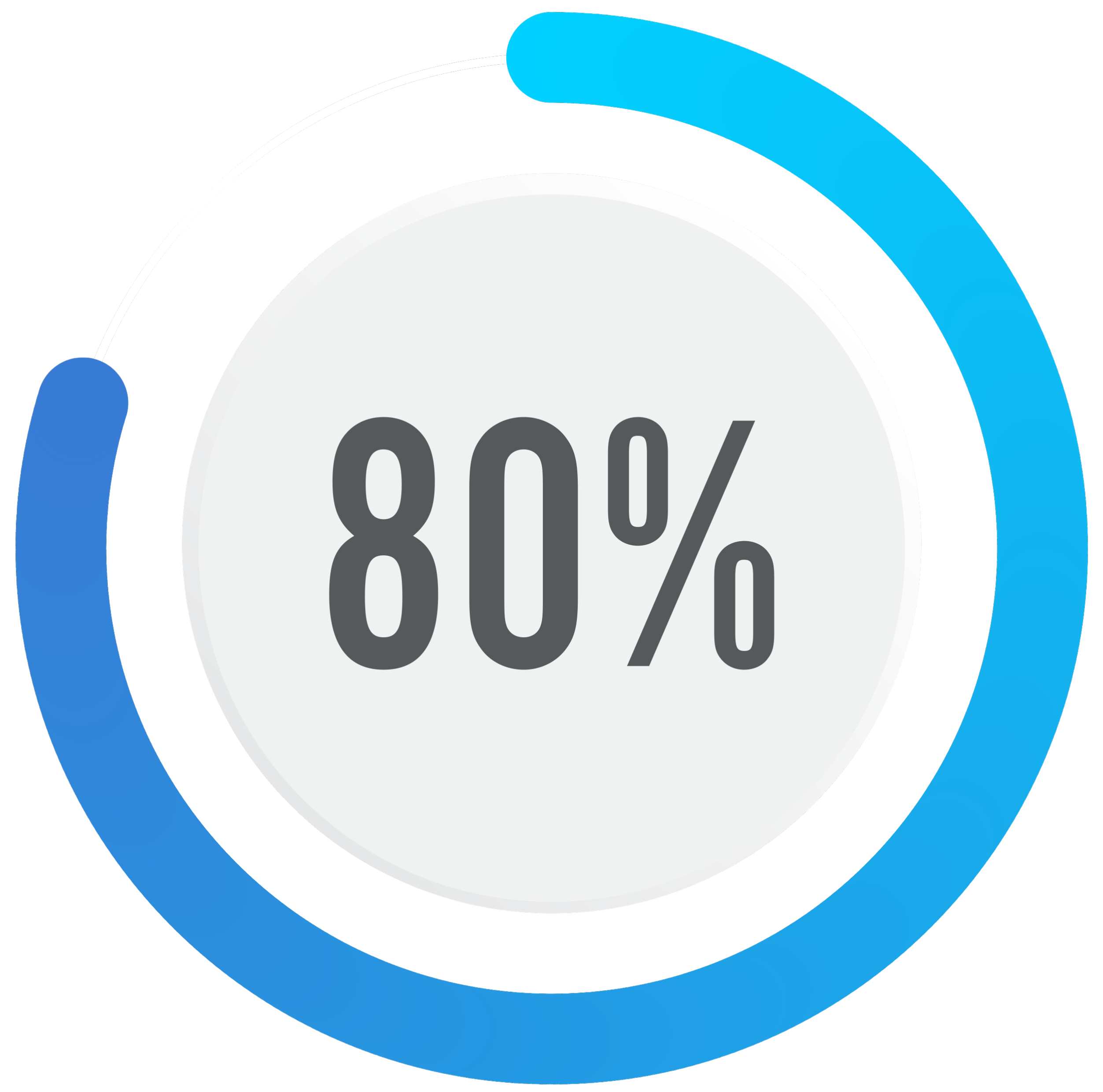
run on Intel CPU

run on Linux / MacOS



20K lines of code

Students read a **very small** portion



2K lines of code

Students read a **large** portion

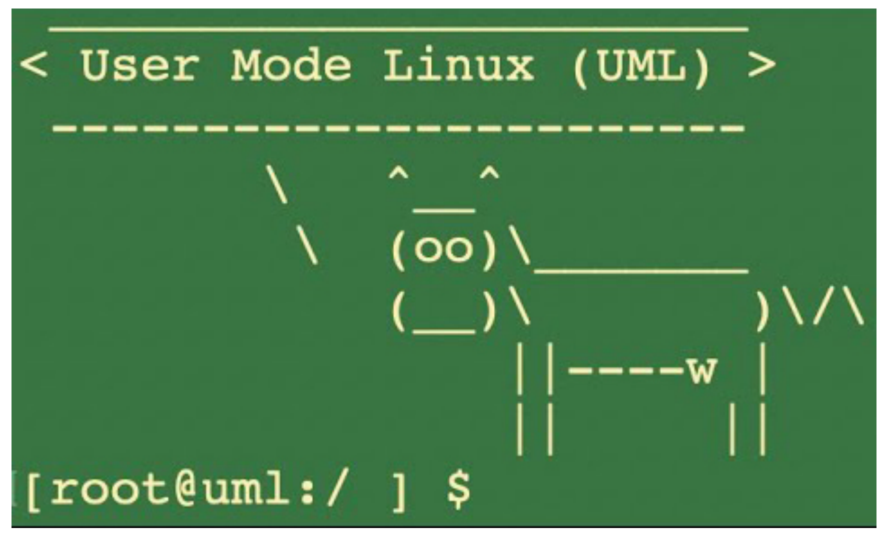
CS 3410
CS 4420

Intel x86 (1987)

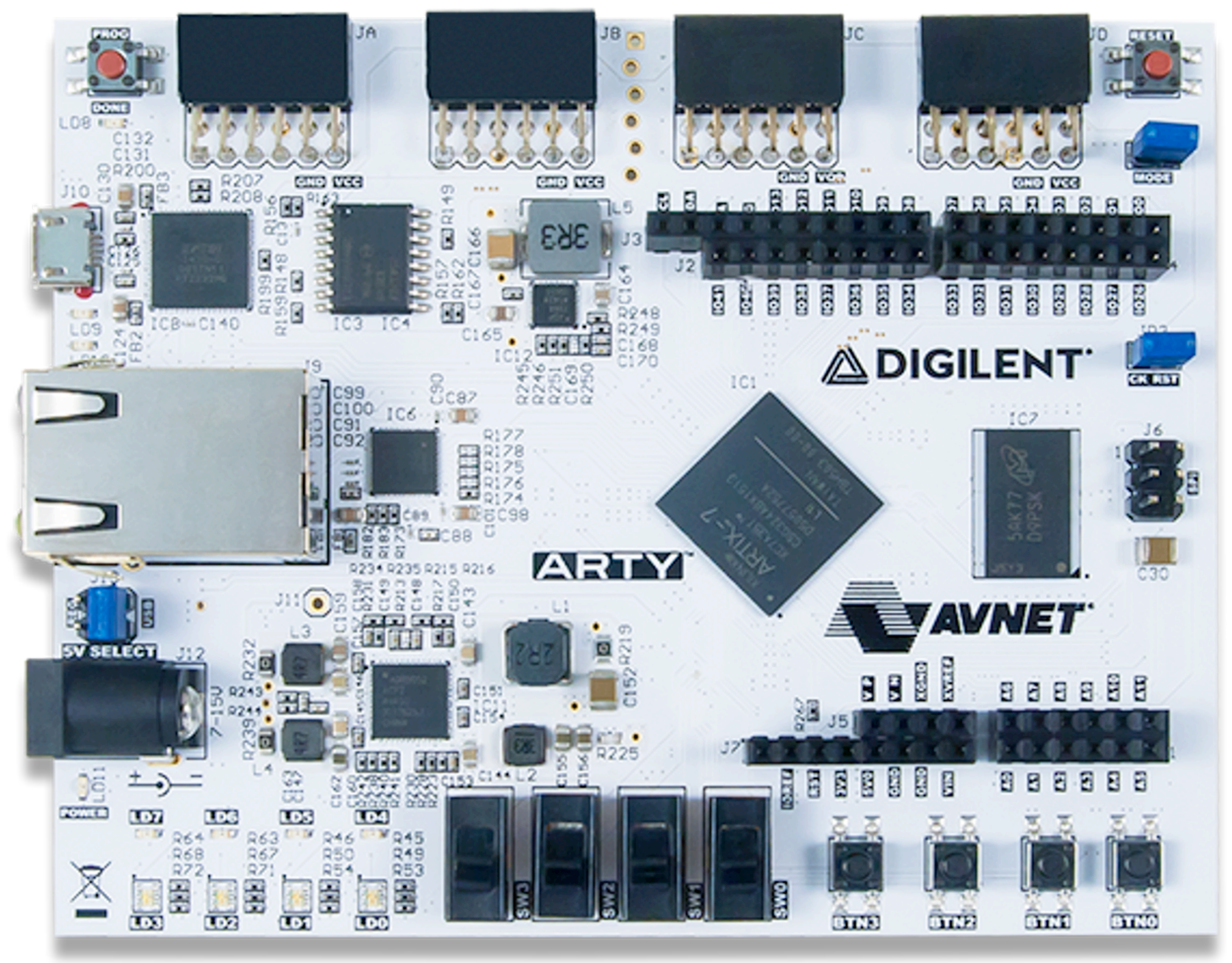
CPU documents have **several thousands** of pages

RISC-V (2010)

CPU documents have **<100** of pages



```
 ________________________
< User Mode Linux (UML) >
 ------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
[root@uml:/ ] $
```



User-mode OS

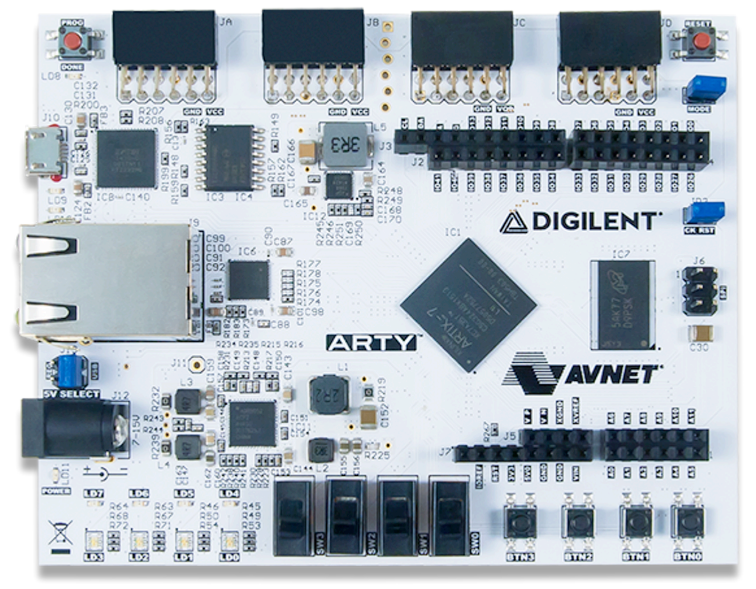
Easier to deploy and run

OS on hardware

More realistic and fun

3 motivations

CS 3410
CS 4420



~20K → ~2K

Intel → RISC-V

Linux / MacOS → real hardware

Lesson

Good motivations should convince non-experts why the work is important

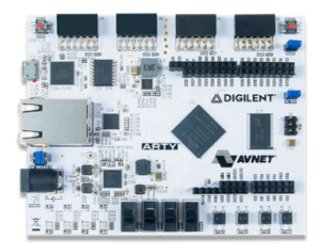
non-experts like students, my friend doing ML theory, etc.

Yet, ideal $\neq$ possible

80%

CS 3410
CS 4420

RISC-V



Obstacles

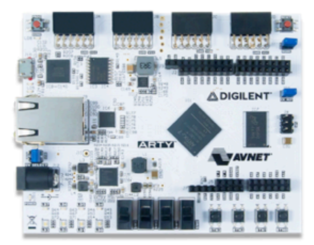
Summer 2020

Fall 2020

Hello World

80%

CS 3410
CS 4420

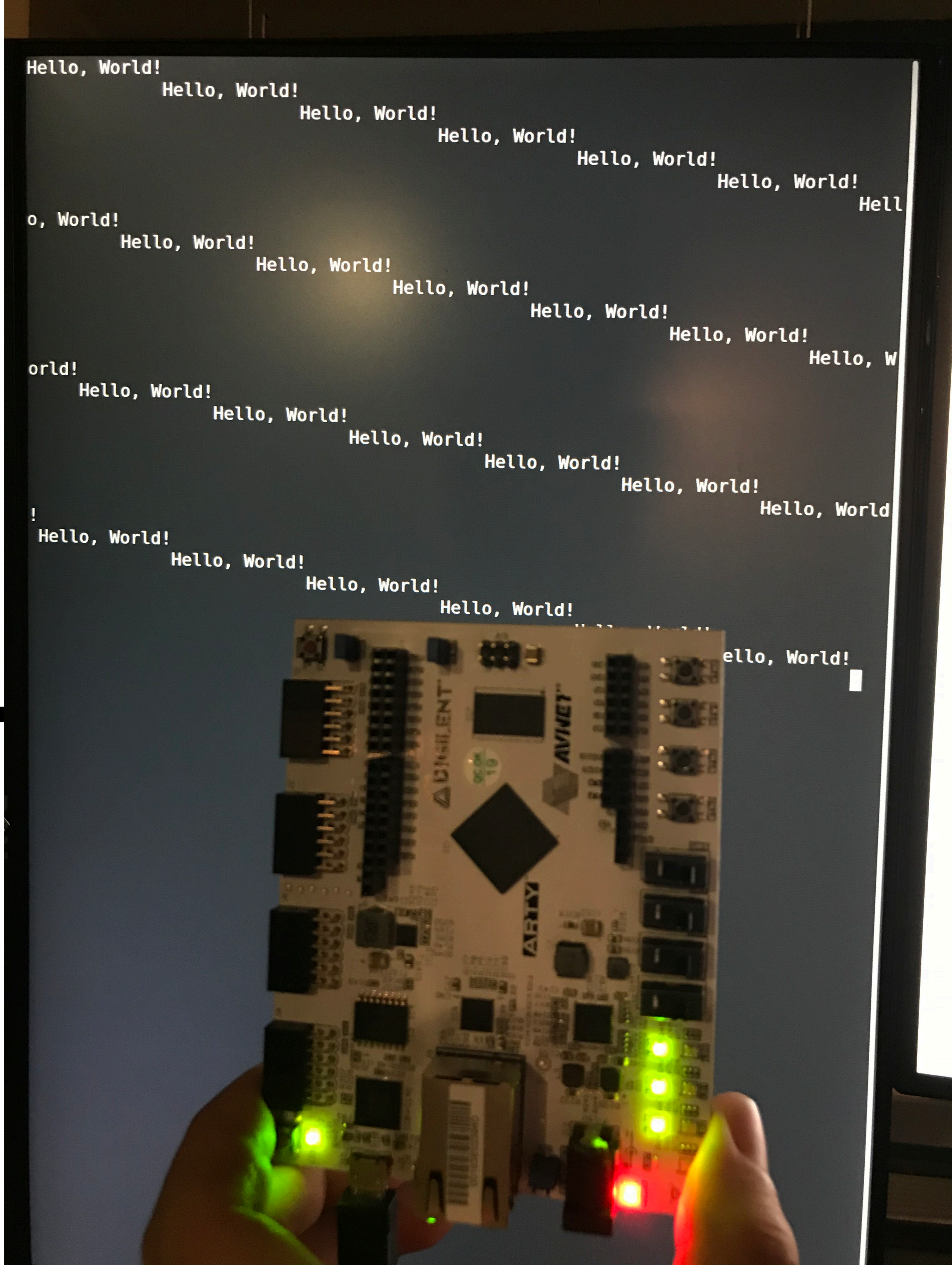


Obstacles

Summer 2020

Fall 2020

Sep 12, 2020

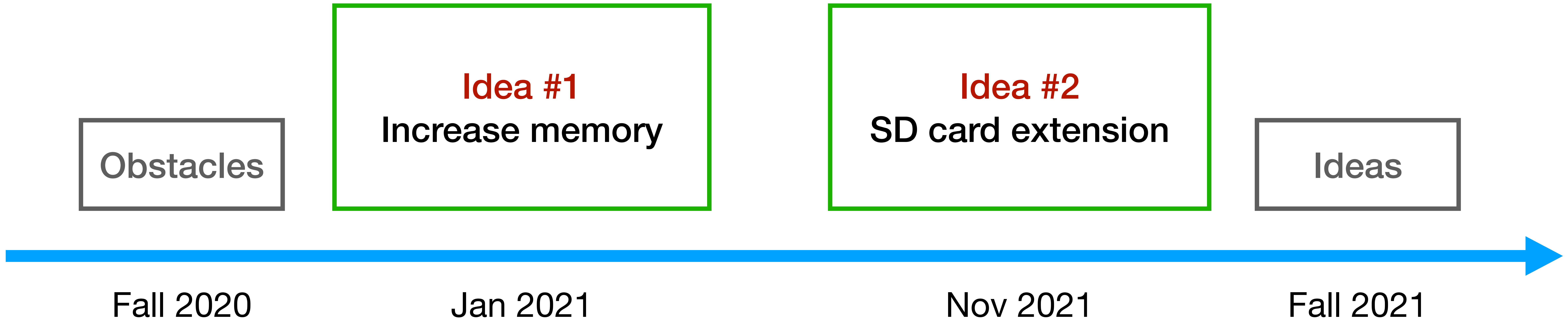


```
Hello, World!
           Hello, World!
                      Hello, World!
                                 Hello, World!
                                            Hello, World!
                                                       Hello, World!
                                                                  Hell
o, World!
        Hello, World!
                   Hello, World!
                              Hello, World!
                                         Hello, World!
                                                    Hello, World!
                                                               Hello, W
orld!
     Hello, World!
                Hello, World!
                           Hello, World!
                                      Hello, World!
                                                 Hello, World!
                                                            Hello, World
!
 Hello, World!
            Hello, World!
                       Hello, World!
                                  Hello, World!

                                                        ello, World!
```

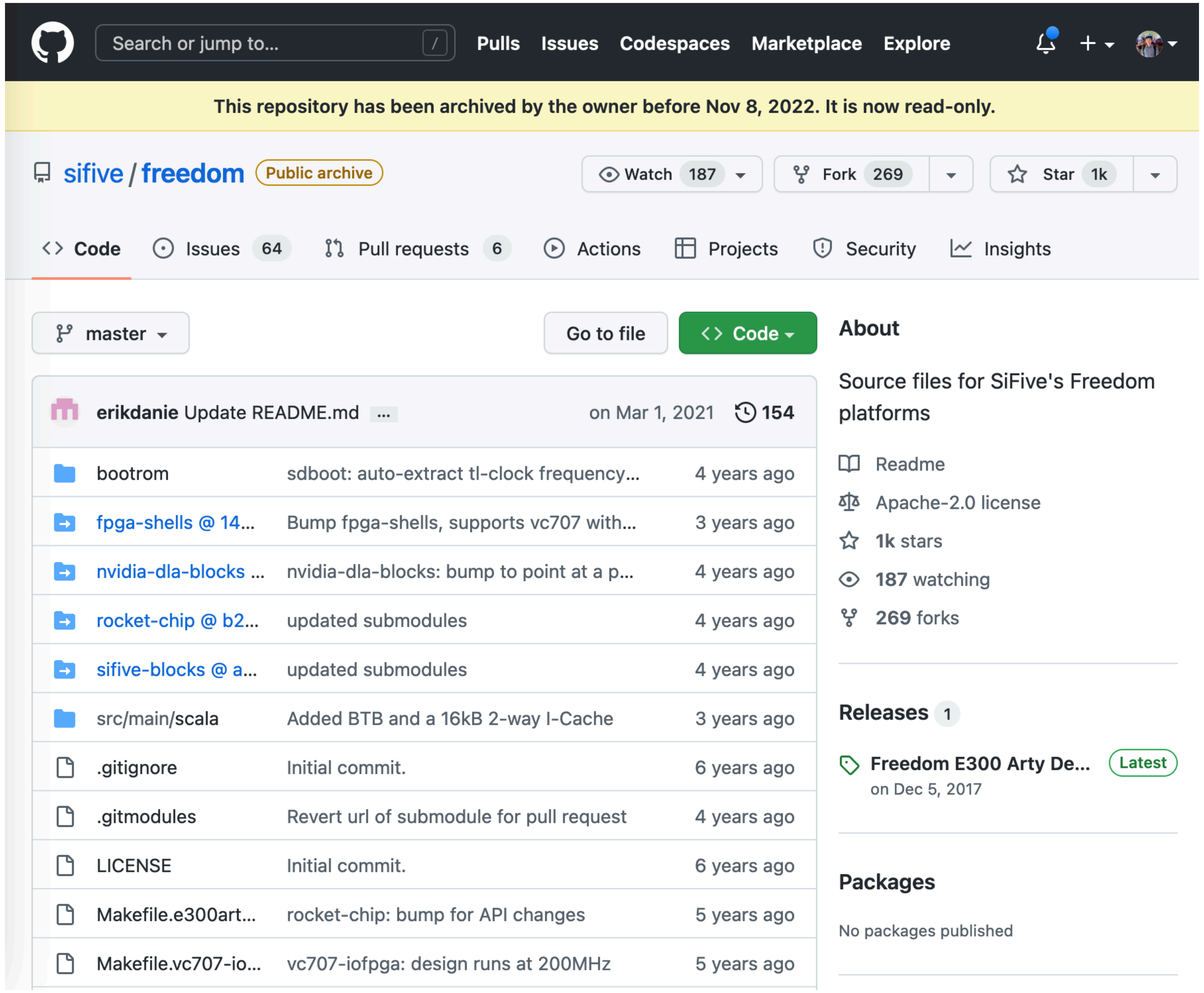
Obstacles & Hope

- ✗ The hardware had only 24KB memory
- ✗ There was no disk
- ✓ The hardware supports timer interrupt
- ✓ The hardware supports privilege levels

Overcome obstacles with ideas!



Open-source hardware



This repository has been archived by the owner before Nov 8, 2022. It is now read-only.

sifive / freedom Public archive

erikdanie Update README.md ... on Mar 1, 2021 154

bootrom	sdboot: auto-extract tl-clock frequency...	4 years ago
fpga-shells @ 14...	Bump fpga-shells, supports vc707 with...	3 years ago
nvidia-dla-blocks ...	nvidia-dla-blocks: bump to point at a p...	4 years ago
rocket-chip @ b2...	updated submodules	4 years ago
sifive-blocks @ a...	updated submodules	4 years ago
src/main/scala	Added BTB and a 16kB 2-way I-Cache	3 years ago
.gitignore	Initial commit.	6 years ago
.gitmodules	Revert url of submodule for pull request	4 years ago
LICENSE	Initial commit.	6 years ago
Makefile.e300art...	rocket-chip: bump for API changes	5 years ago
Makefile.vc707-io...	vc707-iofpga: design runs at 200MHz	5 years ago

About

Source files for SiFive's Freedom platforms

Readme
Apache-2.0 license
1k stars
187 watching
269 forks

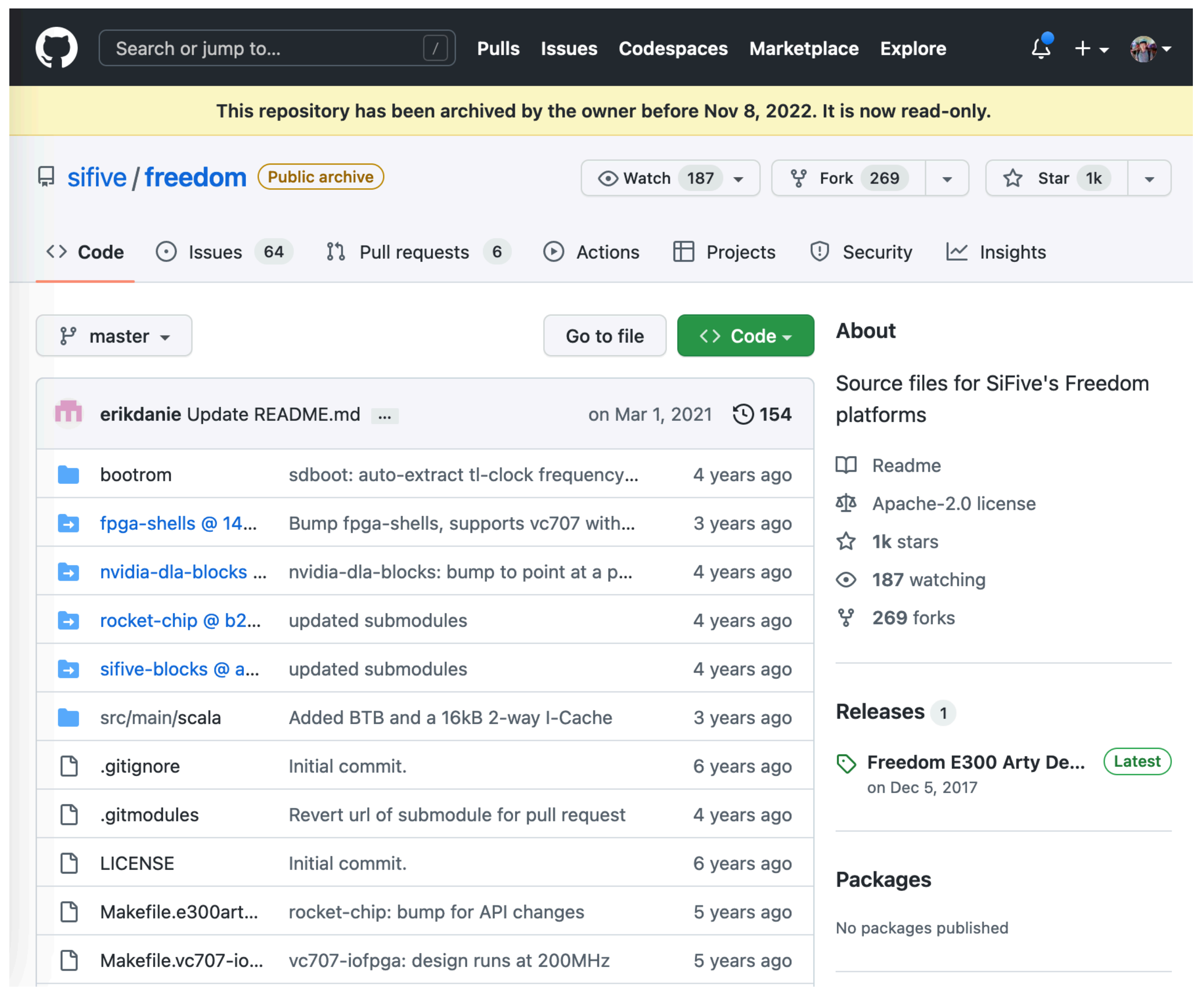
Releases 1

Freedom E300 Arty De... Latest
on Dec 5, 2017

Packages

No packages published

Running open-source hardware



This repository has been archived by the owner before Nov 8, 2022. It is now read-only.

sifive / freedom Public archive

Watch 187 · Fork 269 · Star 1k

Code · Issues 64 · Pull requests 6 · Actions · Projects · Security · Insights

master · Go to file · Code

erikdanie Update README.md ... on Mar 1, 2021 154

bootrom	sdboot: auto-extract tl-clock frequency...	4 years ago
fpga-shells @ 14...	Bump fpga-shells, supports vc707 with...	3 years ago
nvidia-dla-blocks ...	nvidia-dla-blocks: bump to point at a p...	4 years ago
rocket-chip @ b2...	updated submodules	4 years ago
sifive-blocks @ a...	updated submodules	4 years ago
src/main/scala	Added BTB and a 16kB 2-way I-Cache	3 years ago
.gitignore	Initial commit.	6 years ago
.gitmodules	Revert url of submodule for pull request	4 years ago
LICENSE	Initial commit.	6 years ago
Makefile.e300art...	rocket-chip: bump for API changes	5 years ago
Makefile.vc707-io...	vc707-iofpga: design runs at 200MHz	5 years ago

About

Source files for SiFive's Freedom platforms

Readme
Apache-2.0 license
1k stars
187 watching
269 forks

Releases 1

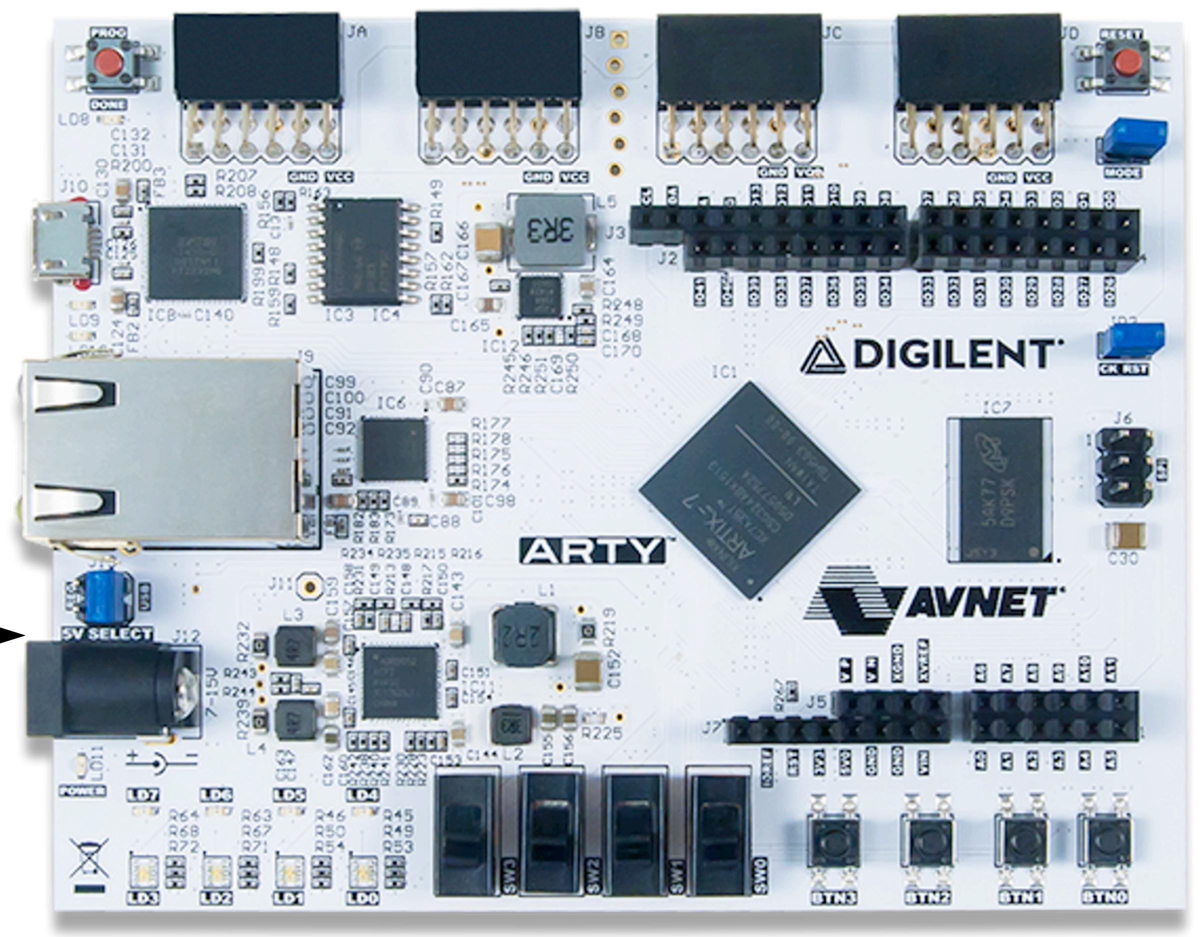
Freedom E300 Arty De... Latest
on Dec 5, 2017

Packages

No packages published

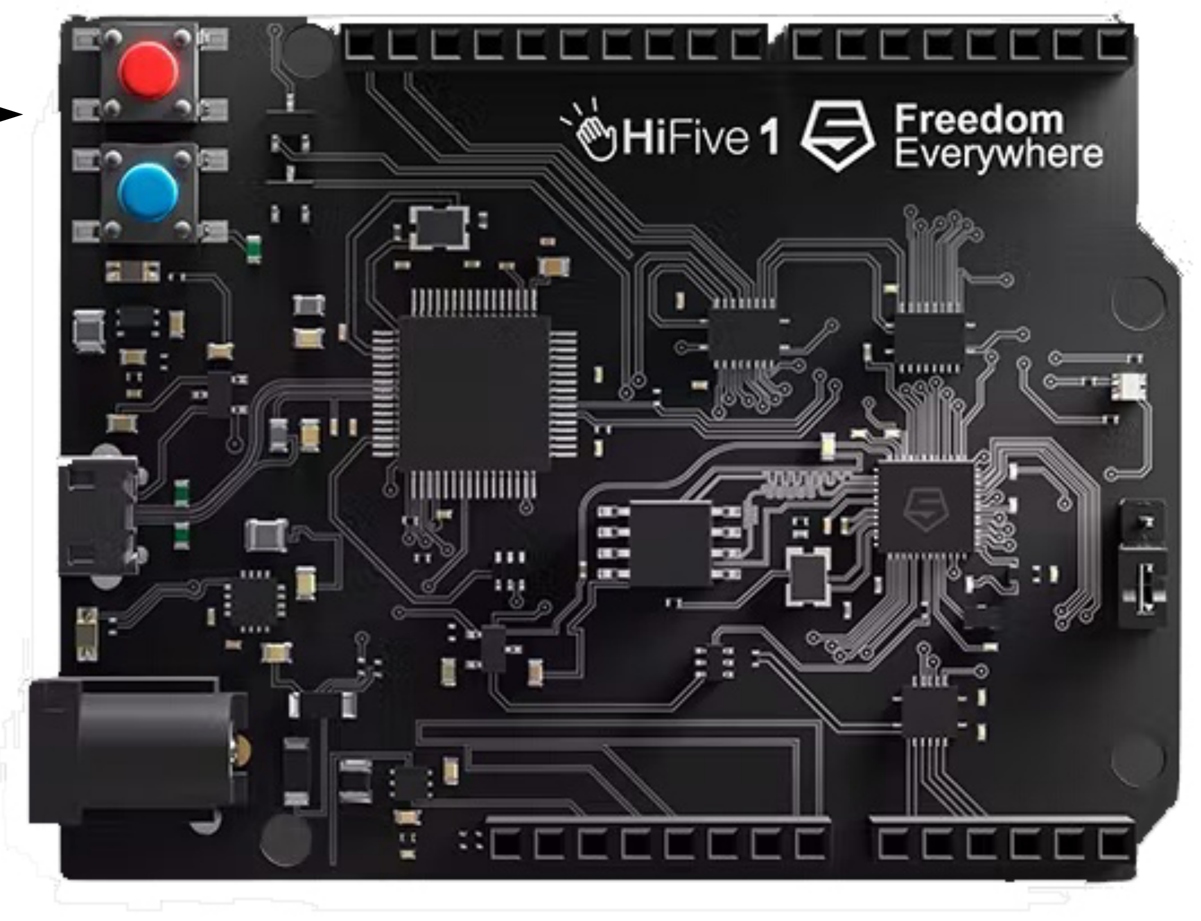
A **binary file**
encoding the hardware design

$\longrightarrow$

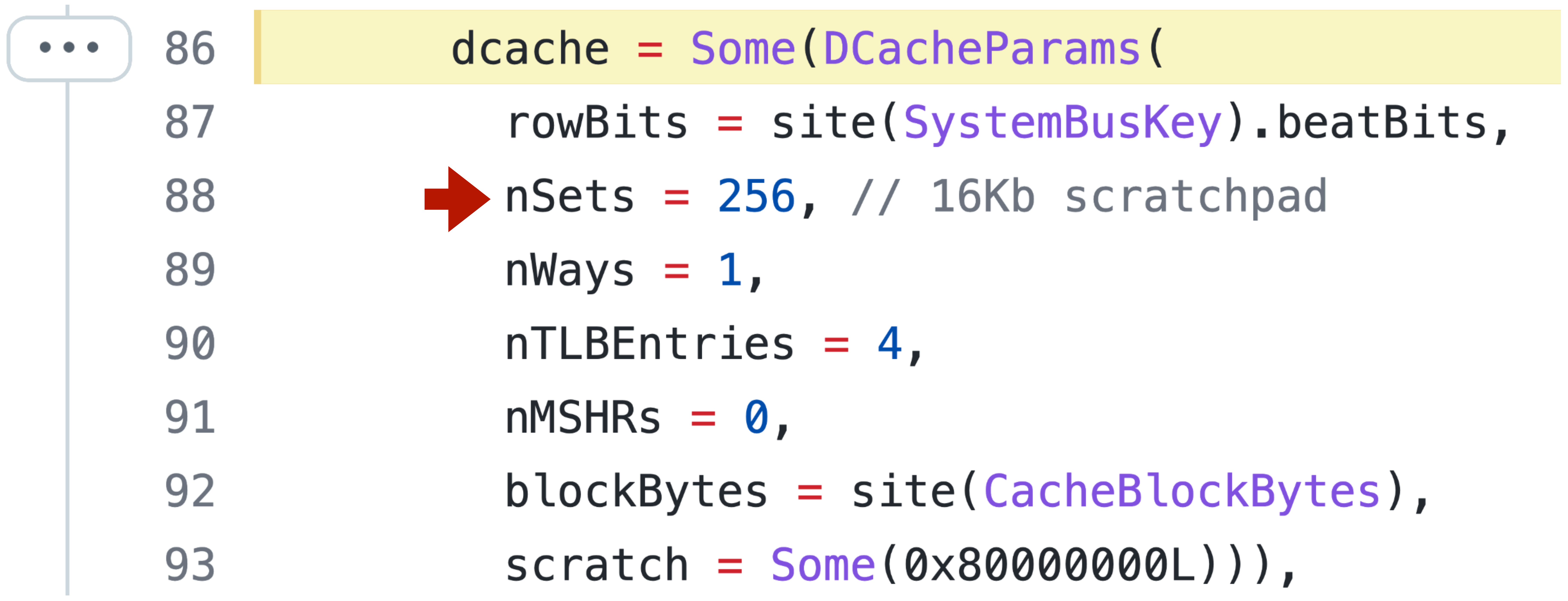


Lithography:
a physical / chemical process

$\longrightarrow$



Idea #1: Increase memory



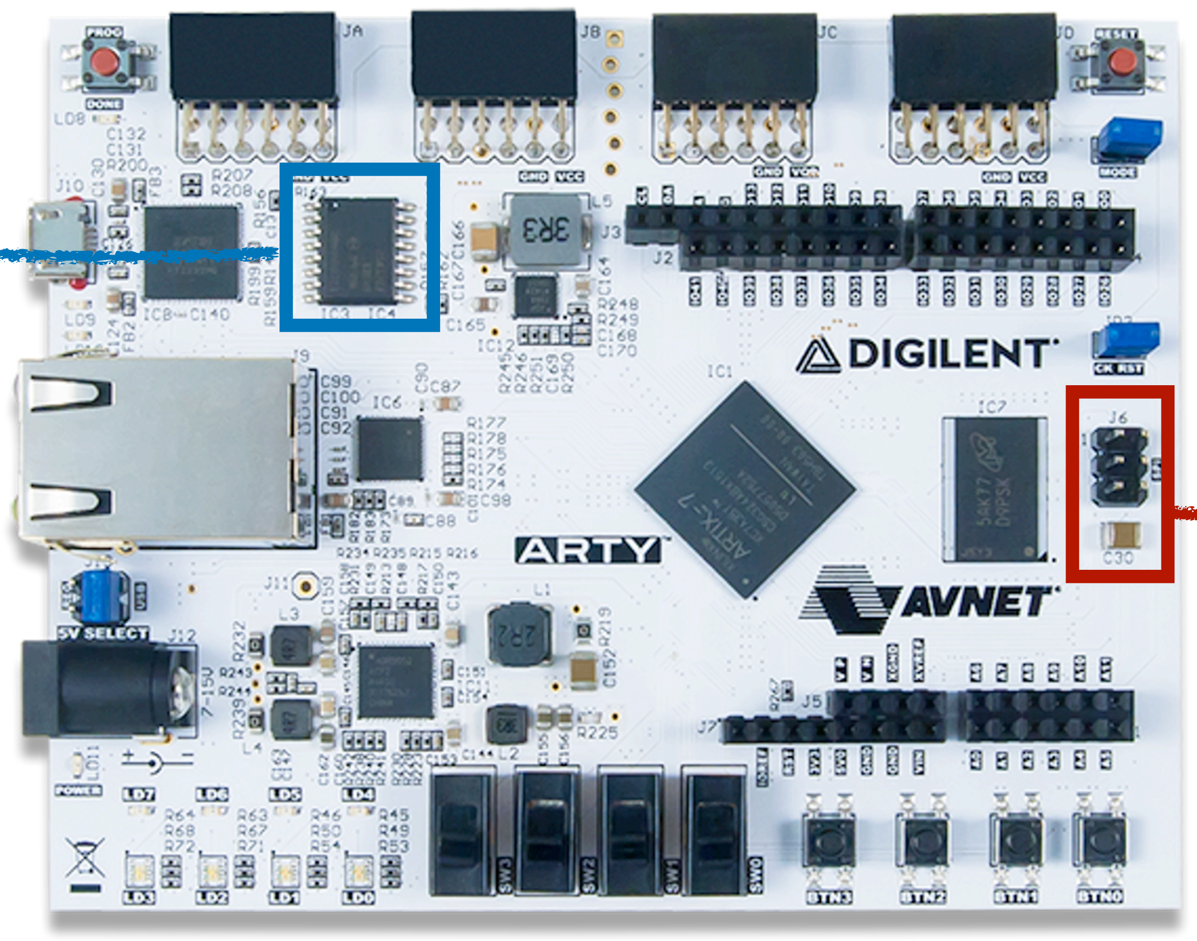
```
    dcache = Some(DCacheParams(
      rowBits = site(SystemBusKey).beatBits,
      nSets = 256, // 16Kb scratchpad
      nWays = 1,
      nTLBEntries = 4,
      nMSHRs = 0,
      blockBytes = site(CacheBlockBytes),
      scratch = Some(0x80000000L))),
```

https://github.com/chipsalliance/rocket-chip/blob/b21c7879b3ea22f69cb8457109561f37c225f8ea/src/main/scala/subsystem/Configs.scala#L78

Idea #2: Background

Instance	Flash Controller	Address	cs_width	div_width
QSPI 0	Y	0x10014000	1	12
SPI 1	N	0x10024000	4	12
SPI 2	N	0x10034000	1	12

Table 64: SPI Instances

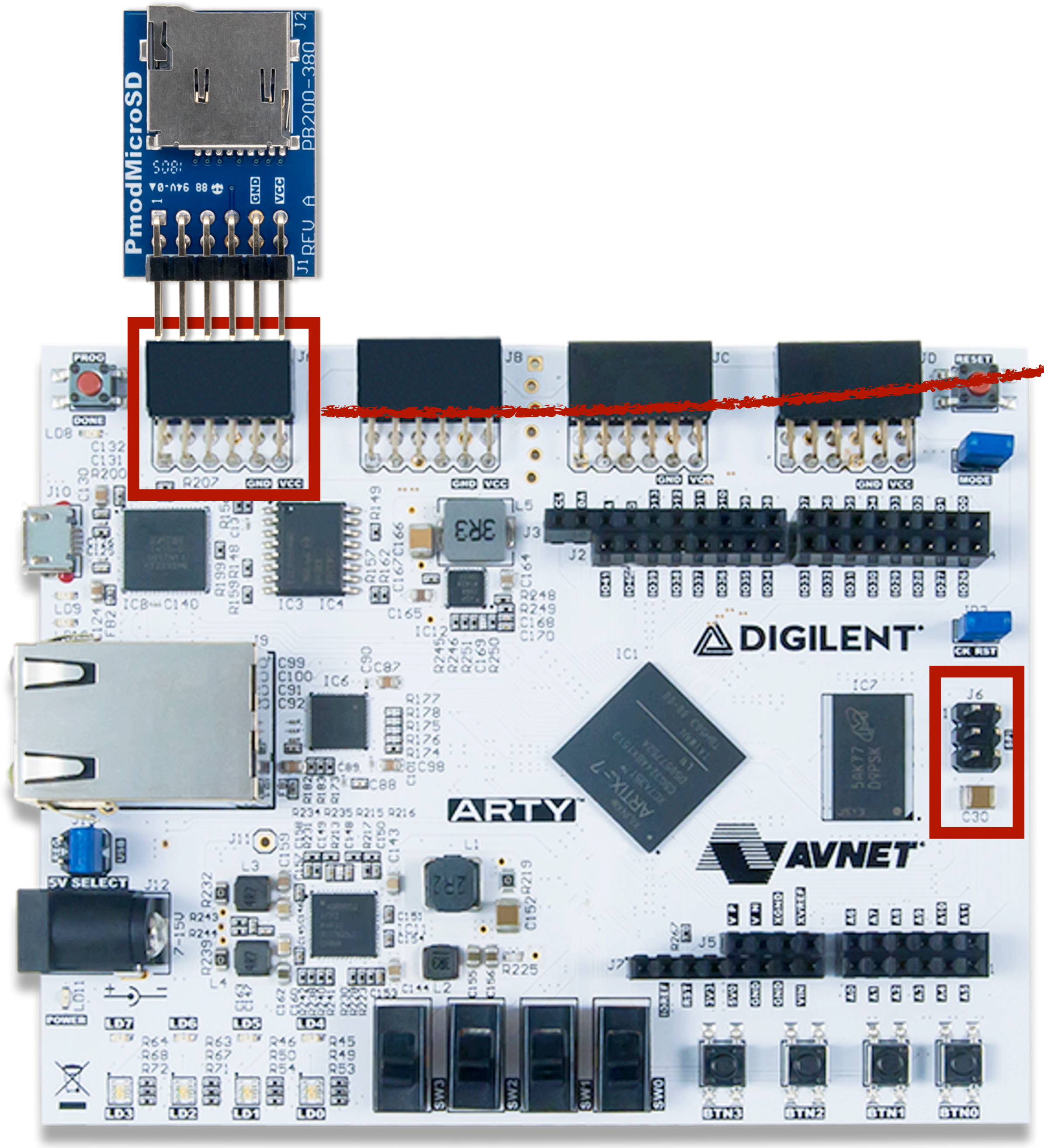


Chapter 19 of Sifive FE310 manual, v19p04
https://github.com/yhzhang0128/egos-2000/blob/main/references/sifive-fe310-v19p04.pdf

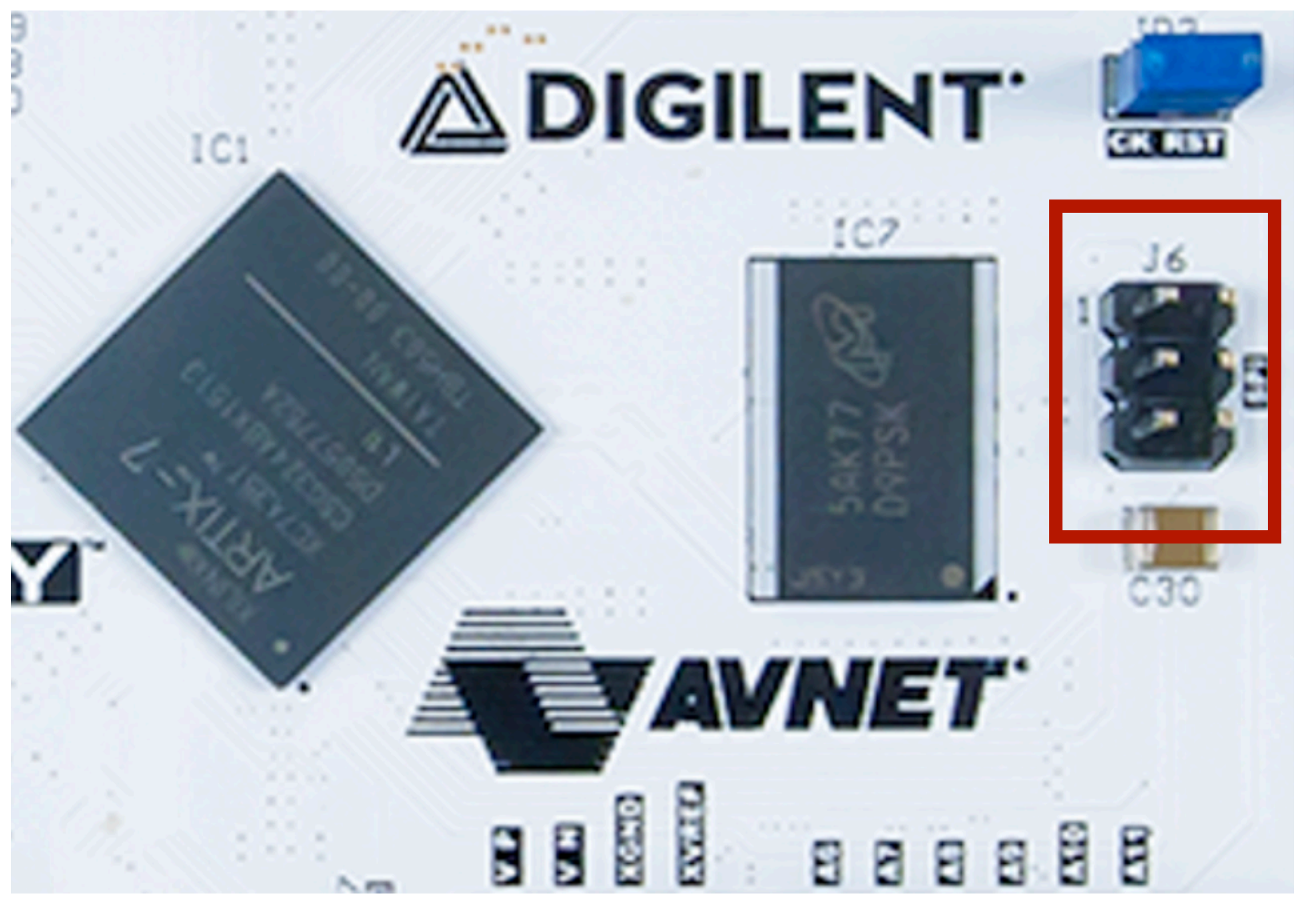
Remap SPI1 to Pmod1 for microSD card

Instance	Flash Controller	Address	cs_width	div_width
QSPI 0	Y	0x10014000	1	12
SPI 1	N	0x10024000	4	12
SPI 2	N	0x10034000	1	12

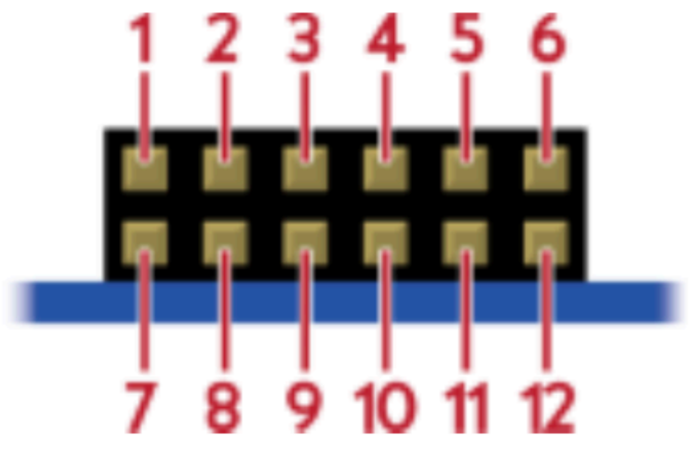
Table 64: SPI Instances



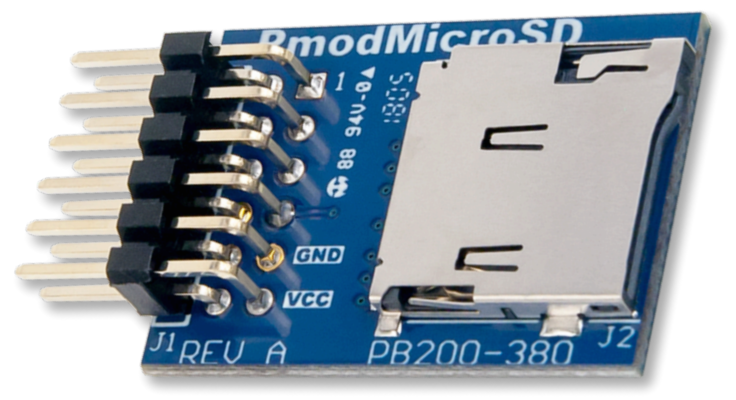
Remap SPI1 to Pmod1 for microSD card



6 pins
GND + VCC + SPI (4)



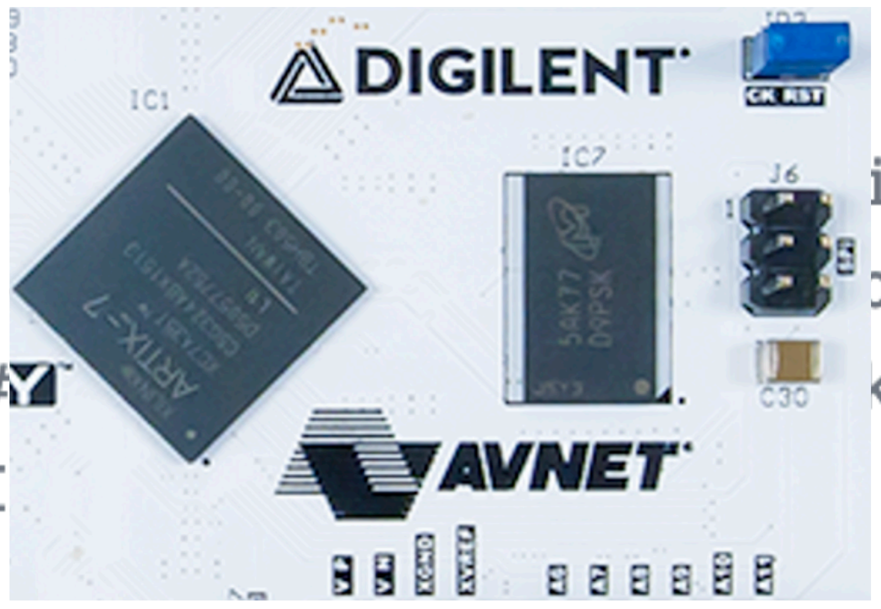
Pin 1	~CS
Pin 2	MOSI
Pin 3	MISO
Pin 4	SCK
Pin 5	GND
Pin 6	VCC



Old SPI1 mapping **New SPI1 mapping**

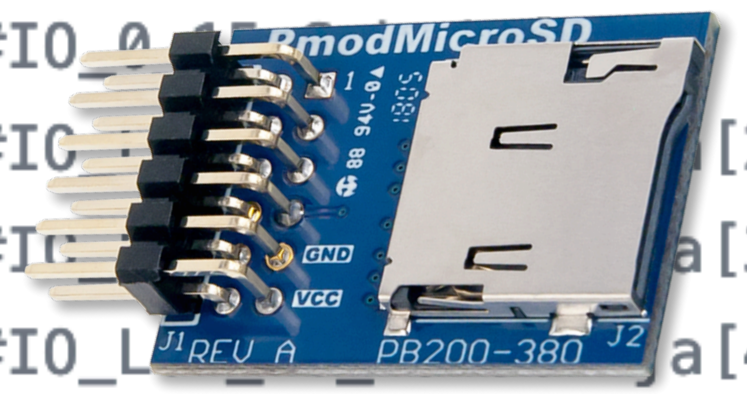
Modifying the hardware design

```
## ChipKit SPI

set_property -dict { PACKAGE_PIN G1    IOSTANDARD LVCMOS33 } [get_ports { ck_miso }];                iso
set_property -dict { PACKAGE_PIN H1    IOSTANDARD LVCMOS33 } [get_ports { ck_mosi }];                osi
set_property -dict { PACKAGE_PIN F1    IOSTANDARD LVCMOS33 } [get_ports { ck_sck }]; #               k
set_property -dict { PACKAGE_PIN C1    IOSTANDARD LVCMOS33 } [get_ports { ck_ss }]; #I
```



Find these 4 wires in the repo and **replace** them

```
##Pmod Header JA

set_property -dict { PACKAGE_PIN G13   IOSTANDARD LVCMOS33 } [get_ports { ja_0 }]; #IO_0
set_property -dict { PACKAGE_PIN B11   IOSTANDARD LVCMOS33 } [get_ports { ja_1 }]; #IO               [2]
set_property -dict { PACKAGE_PIN A11   IOSTANDARD LVCMOS33 } [get_ports { ja_2 }]; #I               a[3]
set_property -dict { PACKAGE_PIN D12   IOSTANDARD LVCMOS33 } [get_ports { ja_3 }]; #IO_L            a[4]
```



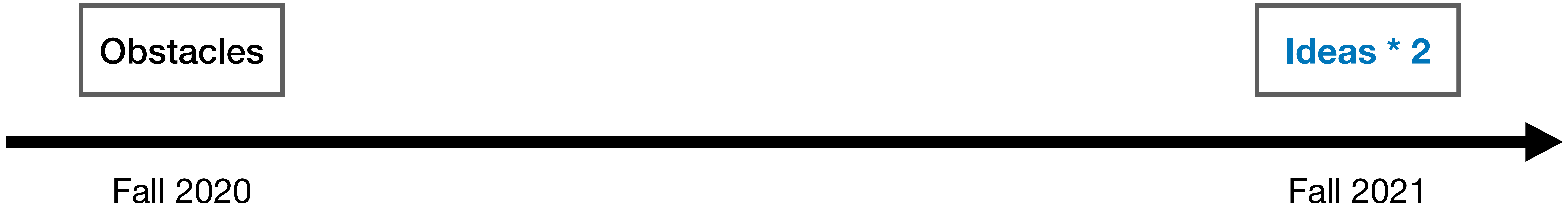
https://github.com/sifive/fpga-shells/blob/14297af2878dc648ffd5751010fa72094ff444b0/xilinx/arty/constraints/arty-master.xdc#L48

Road towards ideas is difficult

No concrete progress for >1 year

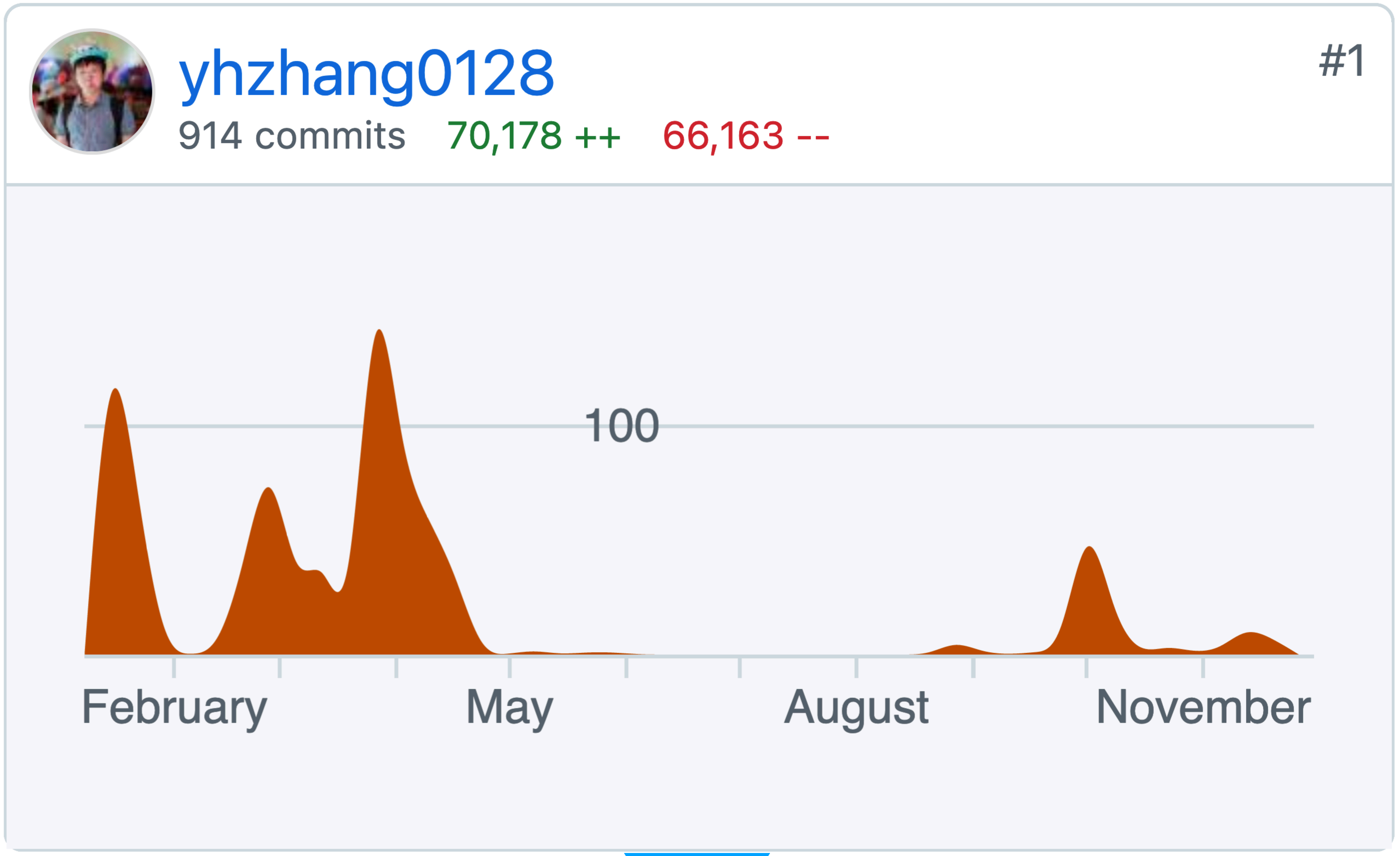
Not sure whether this can succeed at all

Only person working on this project



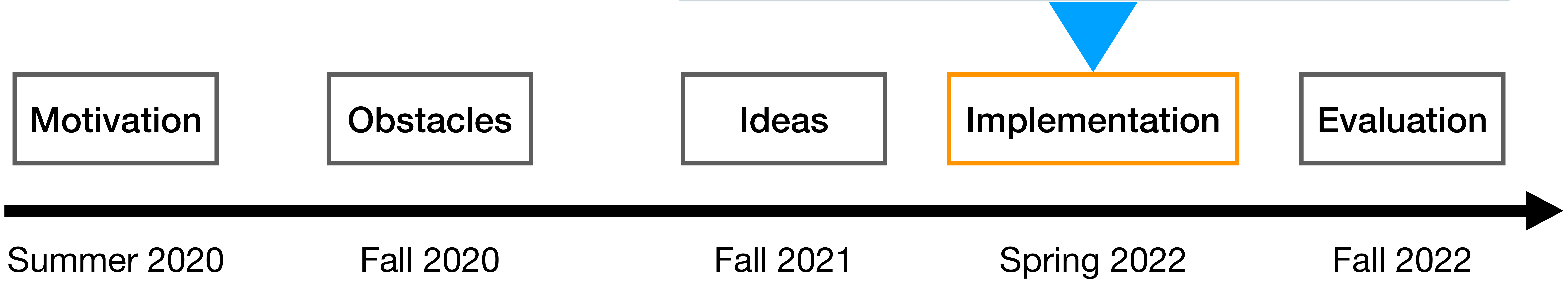
Take-away:

Ideas are difficult to come up with and there is no guarantee of success 🥲


yhzhang0128

914 commits 70,178 ++ 66,163 --

#1



https://github.com/yhzhang0128/egos-2000/blob/main/references/README.md#software-development-history

Iteration #0
* [2020.09] Setup the toolchain provided by SiFive; Compile and run Hello World on Arty
* [2020.09] Test the basic input and print functionalities using the SiFive Metal library

Iteration #1
* [2020.12] Increase the processor memory from 24KB to 160KB (128KB + 32KB)
* [2020.12] Confirm that the memory cannot be further increased due to the limitation of Artix-7 35T FPGA chip

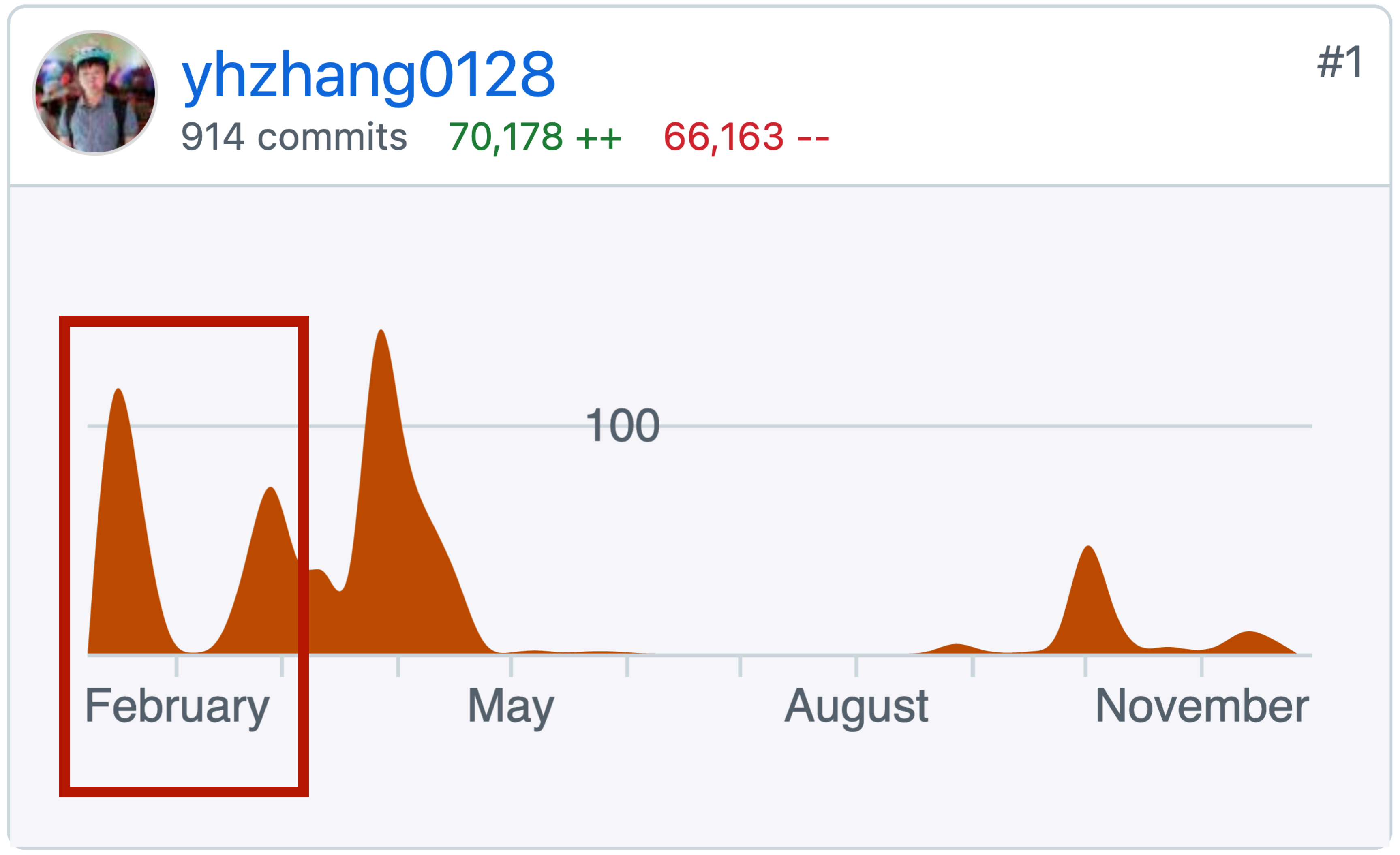
Iteration #2
* Yeah, I didn't work on this project in most of 2021...
* [2021.12] Create a docker image for portable toolchain setup: [Docker Hub repo](https://hub.docker.com/repository/docker/yhzhang0128/arty-toolchain)
* [2021.12] Reconnect the processor SPI bus controller to the Arty Pmod1 pins
* [2021.12] Implement the SD card initialize, read and write functions
* [2021.12] Increase the processor clock frequency from 32MHz to 65MHz so that read/write blocks become faster

Iteration #3

* [2022.01] Confrim that the processor clock frequency cannot be further increased
* [2022.01] Implement the `dev_tty`, `dev_disk`, `cpu_intr` and `cpu_mmu` interfaces in earth
* [2022.01] Load the ELF format grass kernel binary file from the SD card to memory
* [2022.01] Implement the control transfer from earth to grass kernel and then to a user application
* [2022.01] Implement `mkrom` so that creating the bootROM image no longer require Vivado or Docker

Iteration #4

* [2022.02] Read Chapter 1, 2 and 3 of [RISC-V manual](riscv-privileged-v1.10.pdf)
* [2022.02] Read Chapter 8, 9 and 10 of [FE310 manual](sifive-fe310-v19p04.pdf)
* [2022.02] Implement timer reset, preemptive scheduling and inter-process communication
* [2022.02] Implement the kernel processes: GPID_PROCESS, GPID_FILE, GPID_DIR, GPID_SHELL

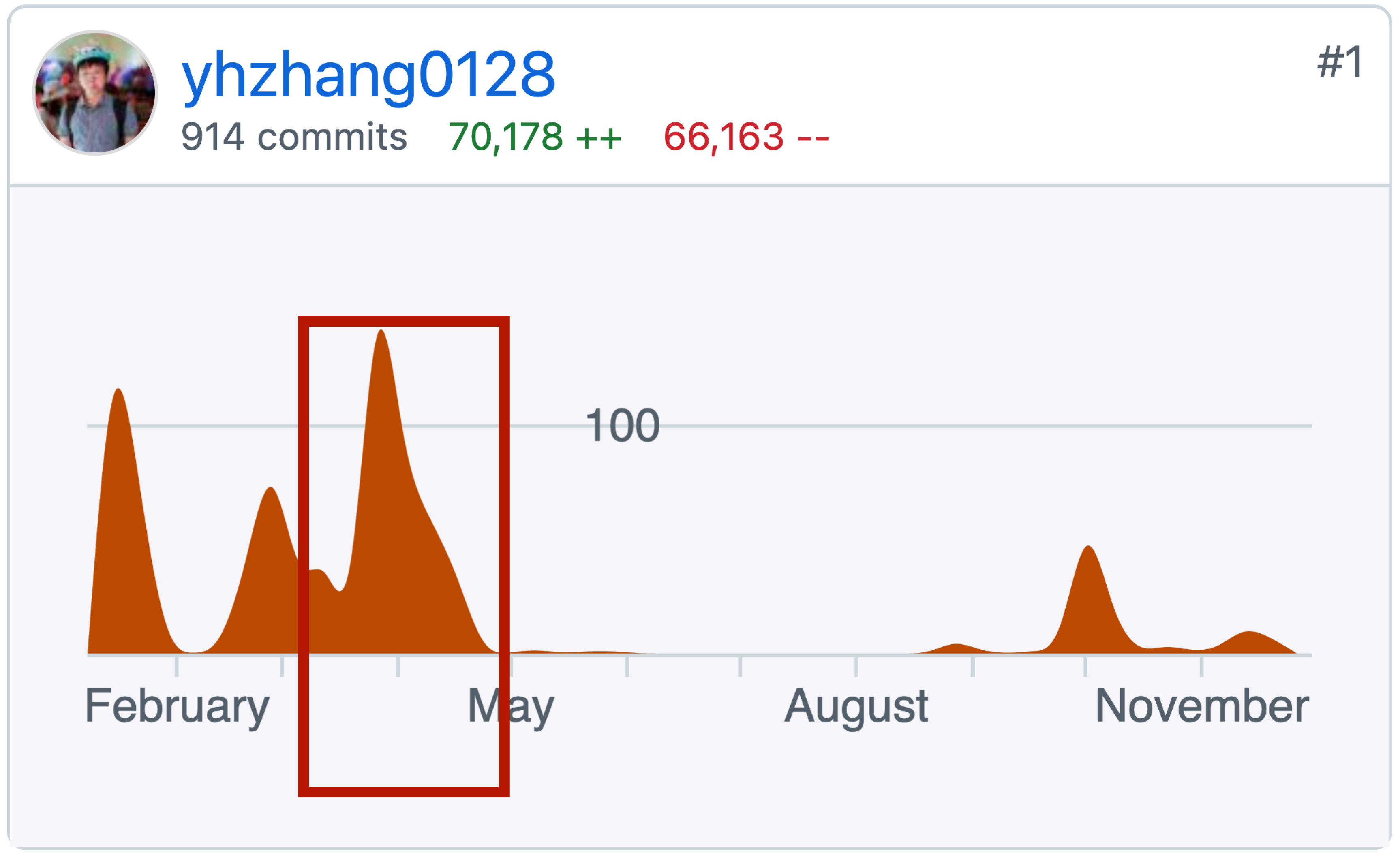


Iteration #5

* [2022.03] Implement system calls and 4 shell commands: `pwd`, `ls`, `cat` and `echo`
* [2022.03] Add support of background shell commands and `killall`
* [2022.03] Add servers in `library` and cleanup access to the file system
* [2022.03] Cleanup the code controlling UART and SPI; Remove dependency on the Metal library

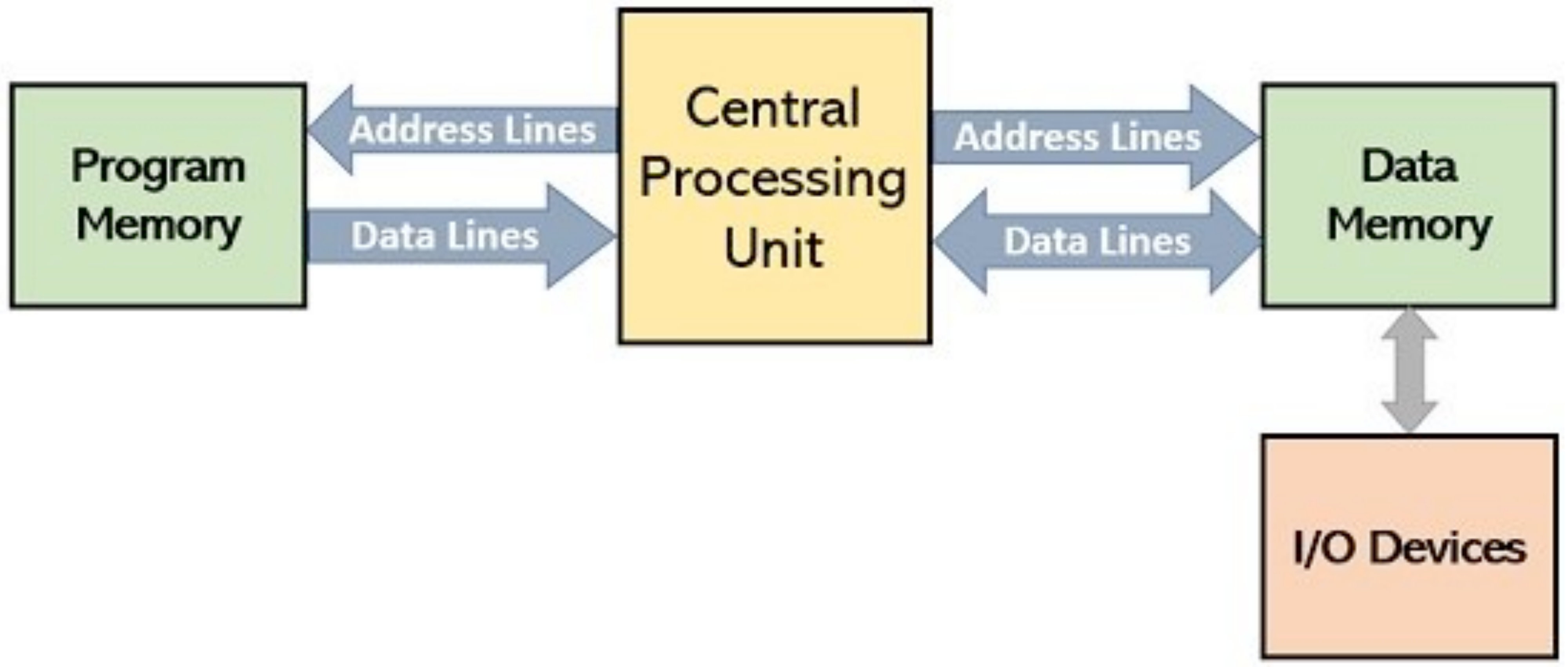
Iteration #6

* [2022.04] Add a simple boot loader `earth/earth.S`; Remove dependency on the Metal library
* [2022.04] Add `_write()` and `_sbrk()` in `library/libc`; Remove the Metal library entirely
* [2022.04] Enrich `struct grass` in order to improve clarity of the architecture
* [2022.04] Experiment with Physical Memory Protection (PMP) and switching privilege level (machine <-> user)



The bug taking me >1 day to fix

```
core = RocketCoreParams(
  useVM = false,
  fpu = None,
  mulDiv = Some(MulDivParams(mulUnroll = 8))),
btb = None,
dcache = Some(DCacheParams(
  rowBits = site(SystemBusKey).beatBits,
  nSets = 256, // 16Kb scratchpad
  nWays = 1,
  nTLBEntries = 4,
  nMSHRs = 0,
  blockBytes = site(CacheBlockBytes),
  scratch = Some(0x80000000L))),
icache = Some(ICacheParams(
  rowBits = site(SystemBusKey).beatBits,
  nSets = 64,
  nWays = 1,
  nTLBEntries = 4,
  blockBytes = site(CacheBlockBytes)))))
```



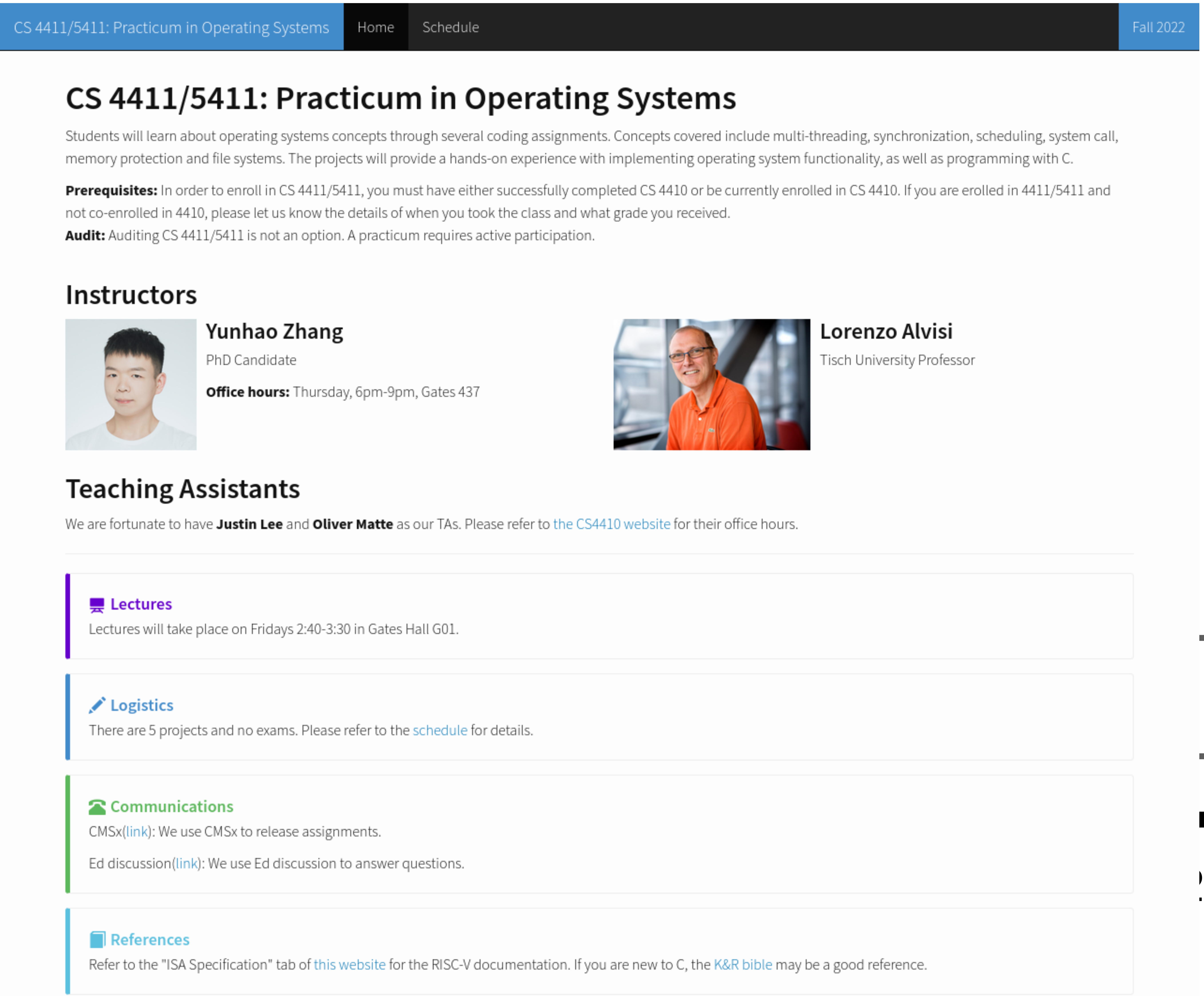
Harvard Architecture

https://github.com/chipsalliance/rocket-chip/blob/b21c7879b3ea22f69cb8457109561f37c225f8ea/src/main/scala/subsystem/Configs.scala#L78

Lesson

Implementing a system is non-trivial and requires determination and hard work

Evaluating the ideas: a class!



CS 4411/5411: Practicum in Operating Systems | Home | Schedule | Fall 2022

CS 4411/5411: Practicum in Operating Systems

Students will learn about operating systems concepts through several coding assignments. Concepts covered include multi-threading, synchronization, scheduling, system call, memory protection and file systems. The projects will provide a hands-on experience with implementing operating system functionality, as well as programming with C.

Prerequisites: In order to enroll in CS 4411/5411, you must have either successfully completed CS 4410 or be currently enrolled in CS 4410. If you are erolled in 4411/5411 and not co-enrolled in 4410, please let us know the details of when you took the class and what grade you received.
Audit: Auditing CS 4411/5411 is not an option. A practicum requires active participation.

Instructors

Yunhao Zhang
PhD Candidate
Office hours: Thursday, 6pm-9pm, Gates 437

Lorenzo Alvisi
Tisch University Professor

Teaching Assistants

We are fortunate to have **Justin Lee** and **Oliver Matte** as our TAs. Please refer to the CS4410 website for their office hours.

Lectures
Lectures will take place on Fridays 2:40-3:30 in Gates Hall G01.

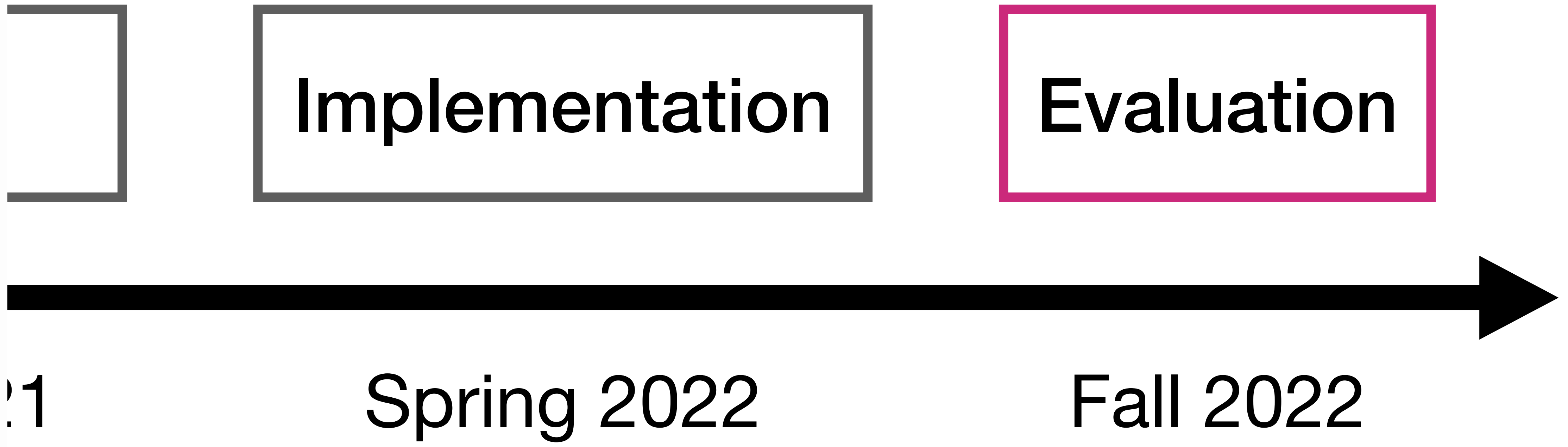
Logistics
There are 5 projects and no exams. Please refer to the schedule for details.

Communications
CMSx(link): We use CMSx to release assignments.
Ed discussion(link): We use Ed discussion to answer questions.

References
Refer to the "ISA Specification" tab of this website for the RISC-V documentation. If you are new to C, the K&R bible may be a good reference.

Remember to write the course evaluations!

What can be improved?



A 4.5-year research process

2 years: Becoming familiar with OS education

Summer 2018 Summer 2020

Then, challenge state-of-the-art

Motivation Obstacles Ideas Implementation Evaluation

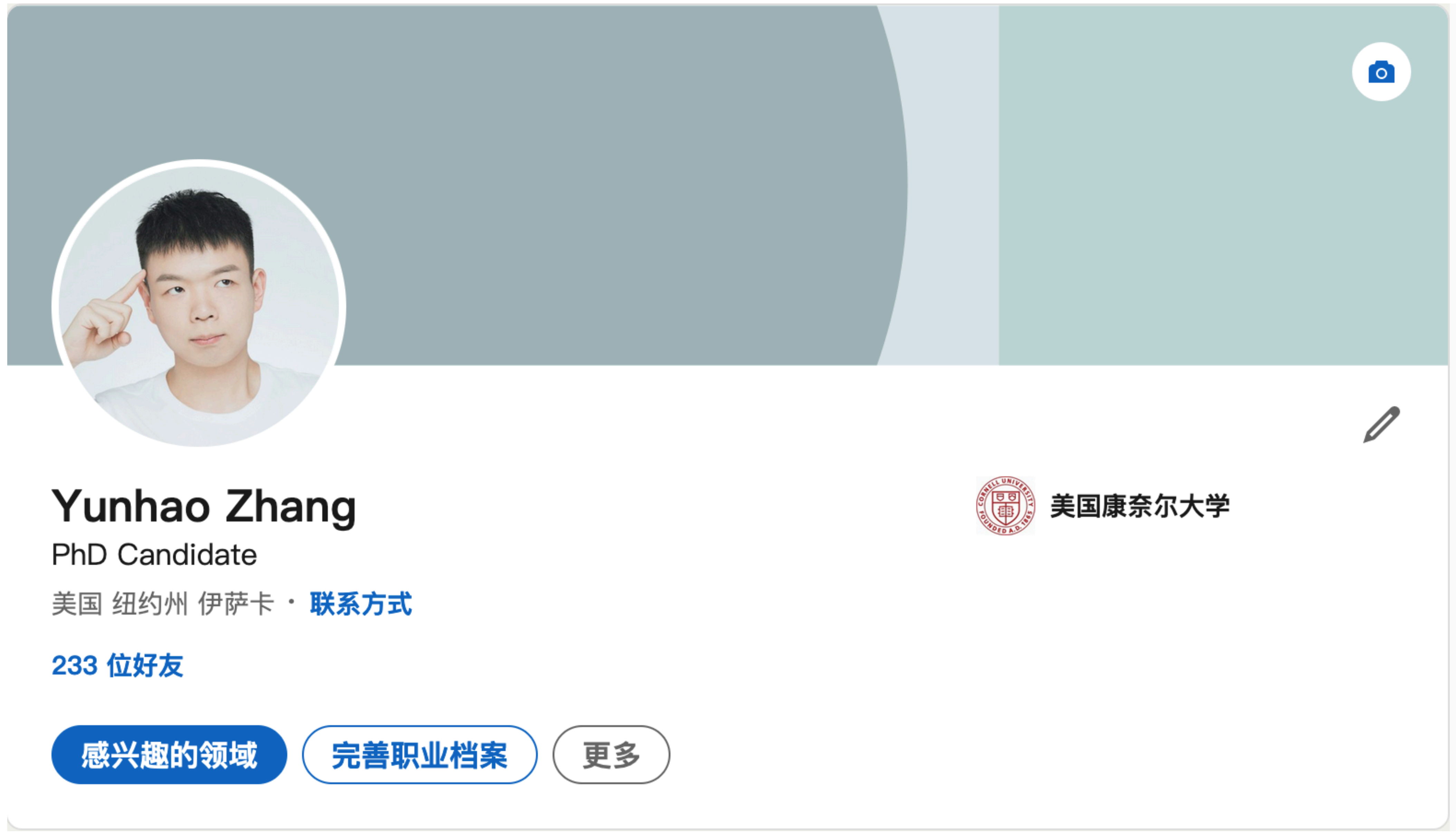
Summer 2020 Fall 2022

Publish the research

Fighting for a world where every college student can read all the code of an operating system

Lines of Code	What?	Lines of Code	What?
199	Boot Loader & TTY Driver	336	File System
182	SD Card Driver	264	Applications & Daemons
32	Interrupt & Exception Handling	269	Library & Networking (TBA)
137	Page Table & Software Translation	64	Makefile
345	Timer, Scheduler & System Call	172	RISC-V Emulator & Board Tools

Stay in touch: LinkedIn or Instagram



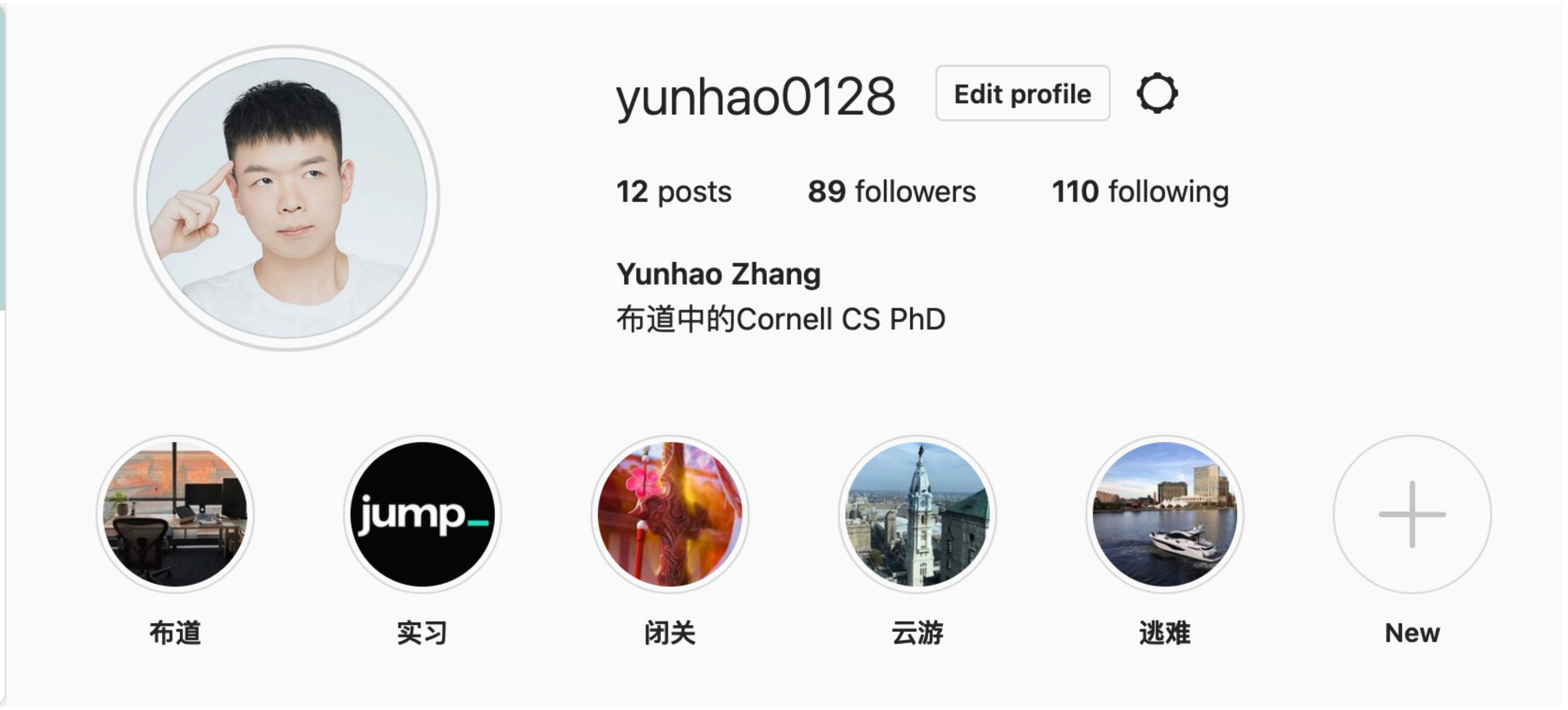
Yunhao Zhang

PhD Candidate

美国 纽约州 伊萨卡 · 联系方式

233 位好友

美国康奈尔大学



yunhao0128

12 posts 89 followers 110 following

Yunhao Zhang

布道中的Cornell CS PhD

Homework

- P4 is optional
- P5 is due on Dec. 7 (extended)
- Please help with course evaluations!
  - see the pinned post #276 on Ed discussion