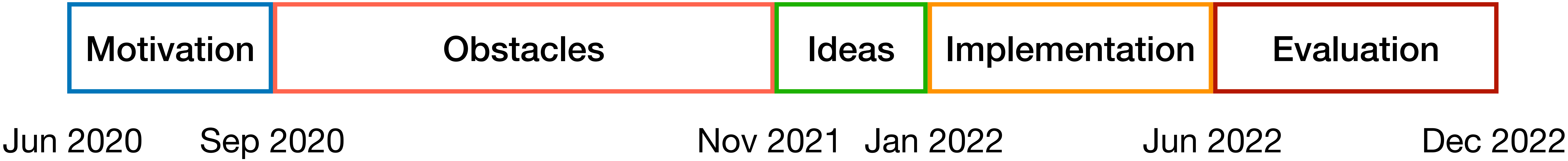
Writing an operating system in 2.5 years

Yunhao Zhang

But first, writing an OS in one semester

- P0: understand C and user-level instructions
- P1: understand context-switch and multi-threading
- P2: understand exception and privilege levels
- P3: understand the disk abstraction
- P4: understand the file abstraction
- P5 (optional): understand I/O bus and devices

Why 2.5 years? An overview



Motivation

Obstacles

Ideas

Implementation

Evaluation

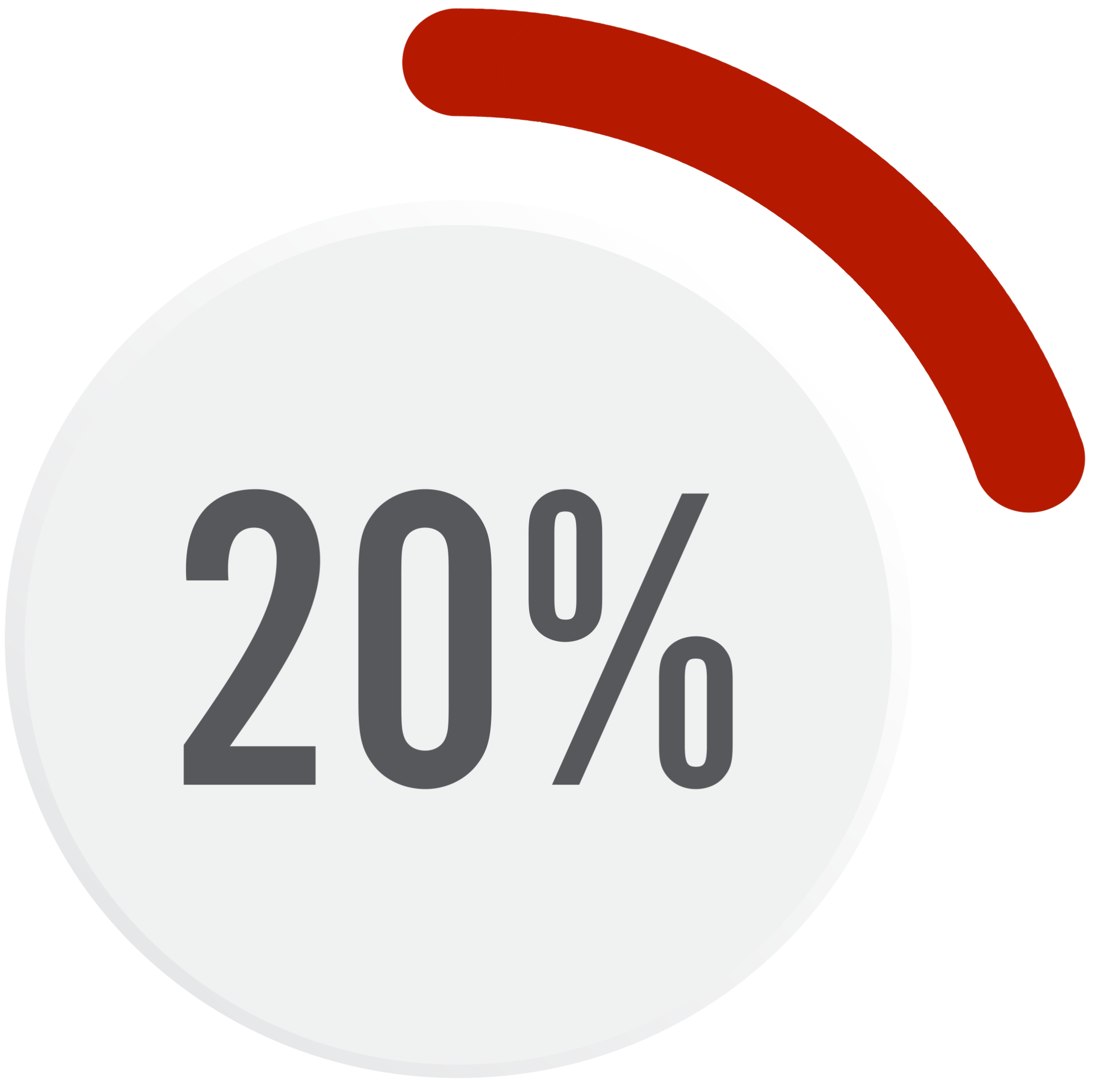
Jun 2020 Sep 2020 Nov 2021 Jan 2022 Jun 2022 Dec 2022

By June 2020, we only had egos-classic

~20K lines of code

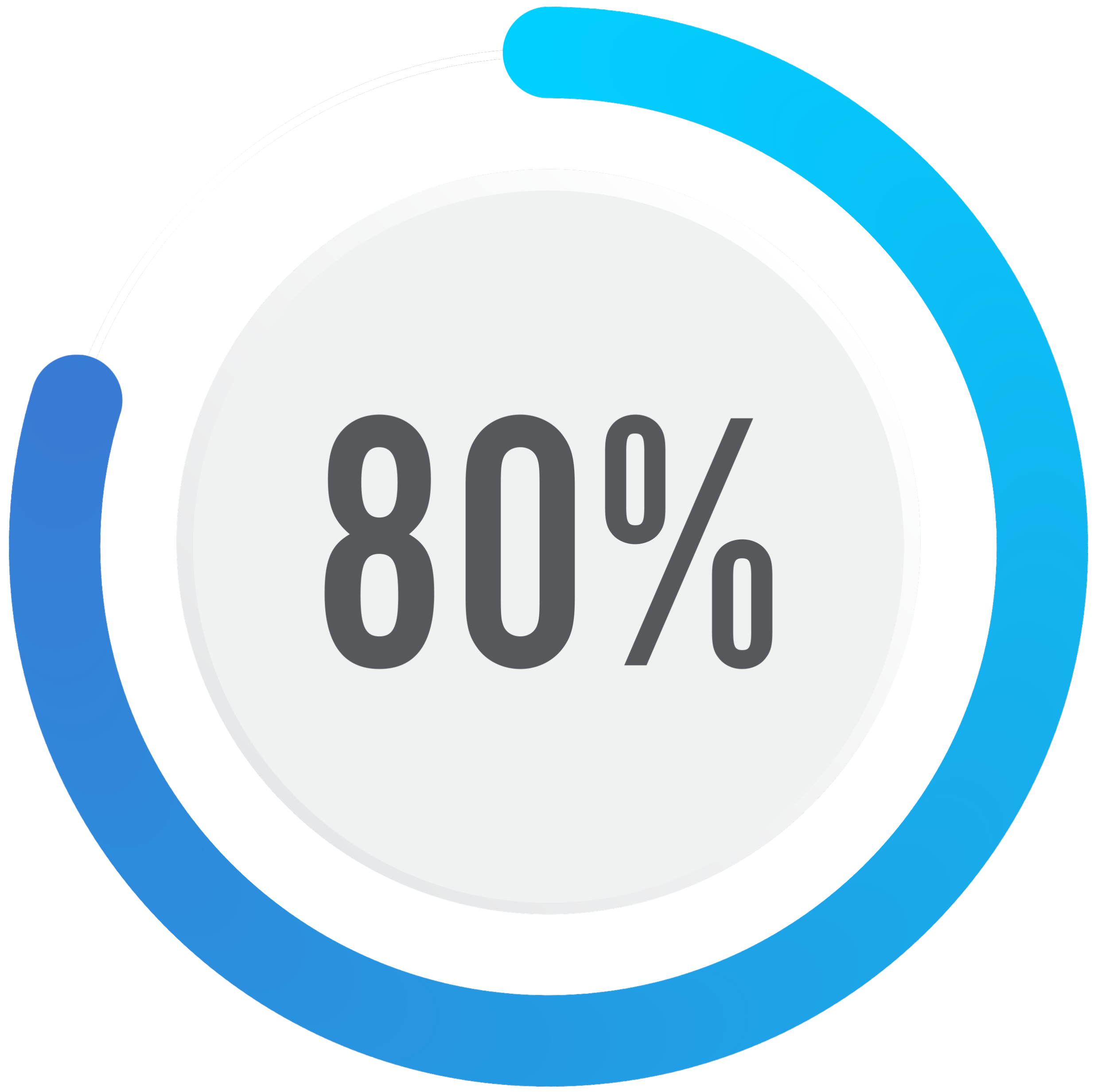
Intel / Arm CPU

Linux / MacOS user process



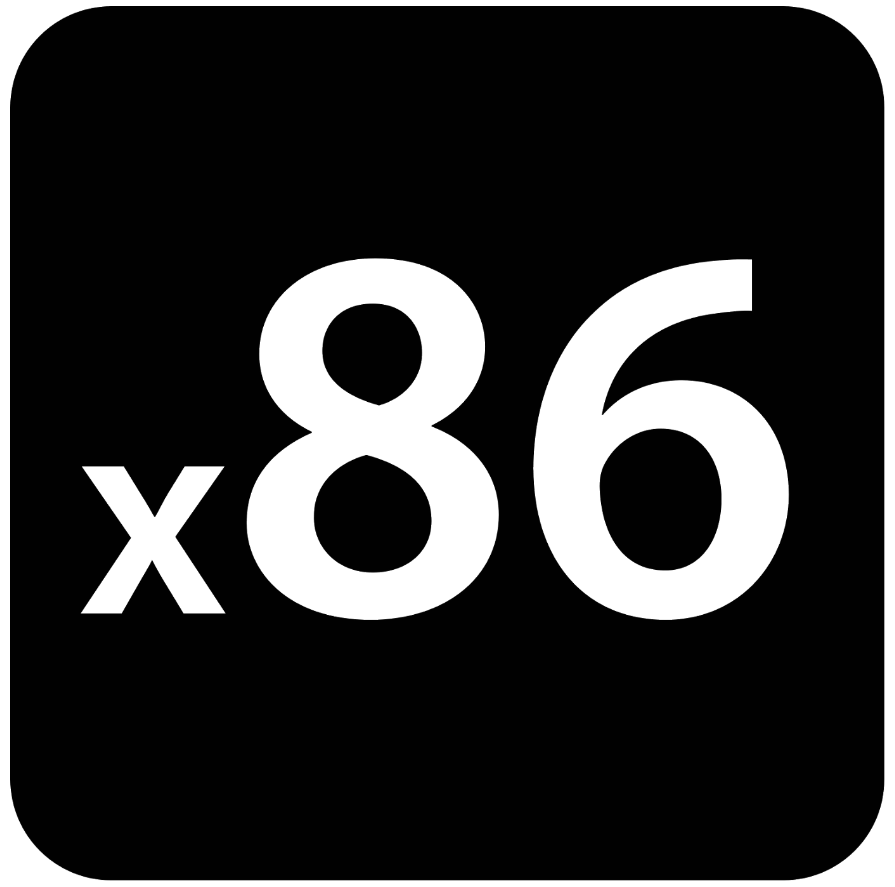
20K lines of code

Students read a **very small** portion



2K lines of code

Students read a **large** portion



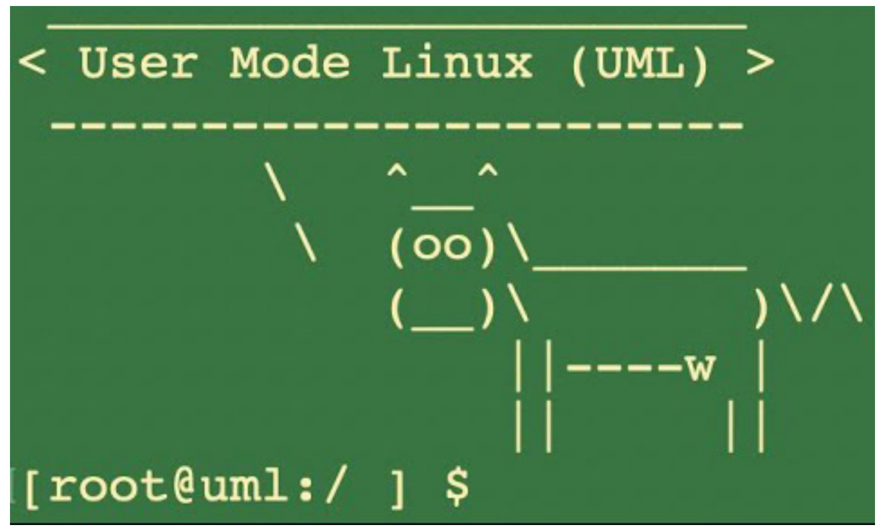
Intel x86 (1987)

CPU document has
several thousands of pages

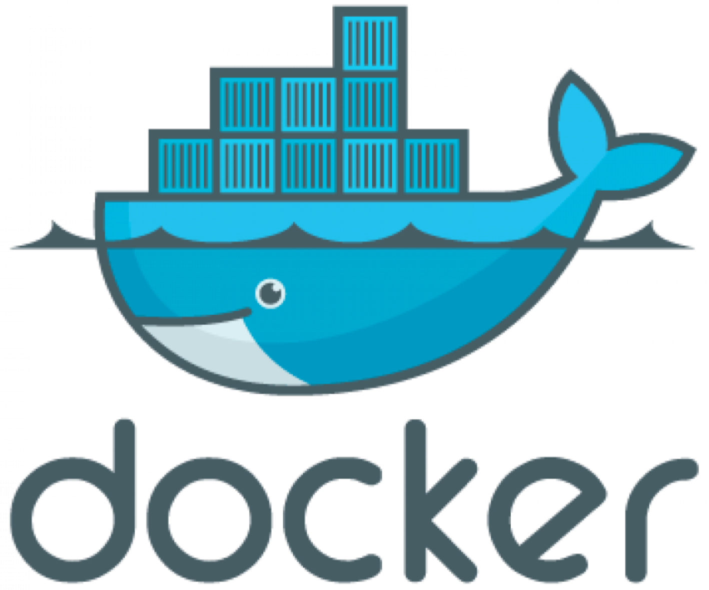
CS 3410
ECE 4750

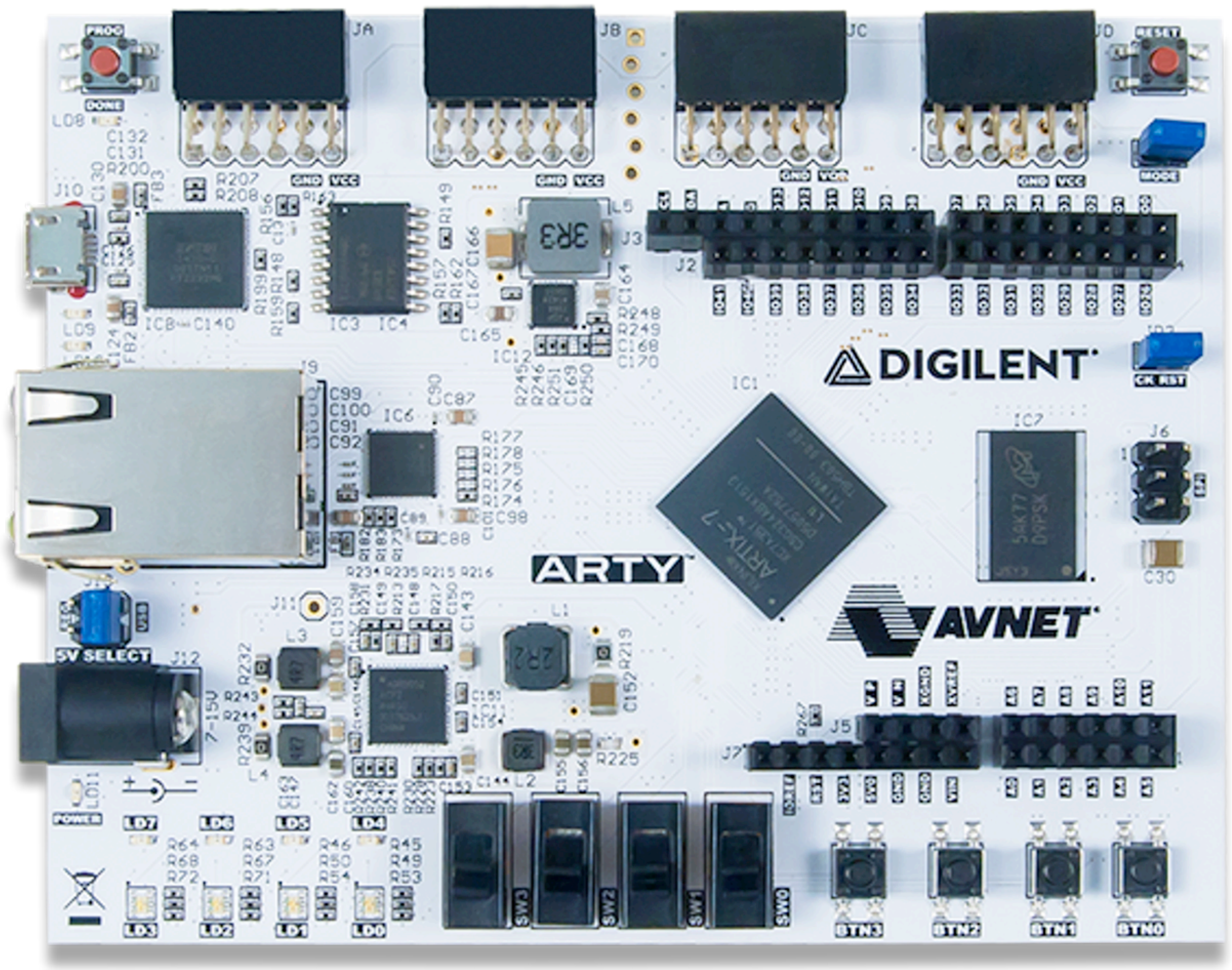
RISC-V (2010)

CPU document
has **<100** of pages



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





User-mode OS

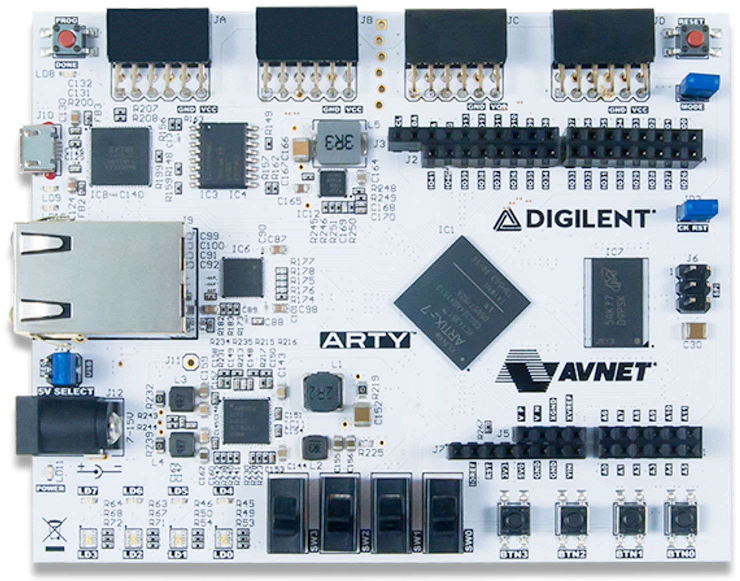
Easier to compile and run

OS on real hardware

More realistic to play with

Motivations

CS 3410
ECE 4750



~20K → **2K**

x86 / ARM → **RISC-V**

Linux / MacOS → **QEMU / board**

Lesson

Good motivations should convince non-experts why the work is valuable.

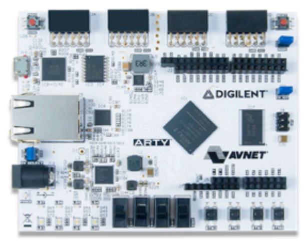
Motivation	Obstacles	Ideas	Implementation	Evaluation

Hello World

80%

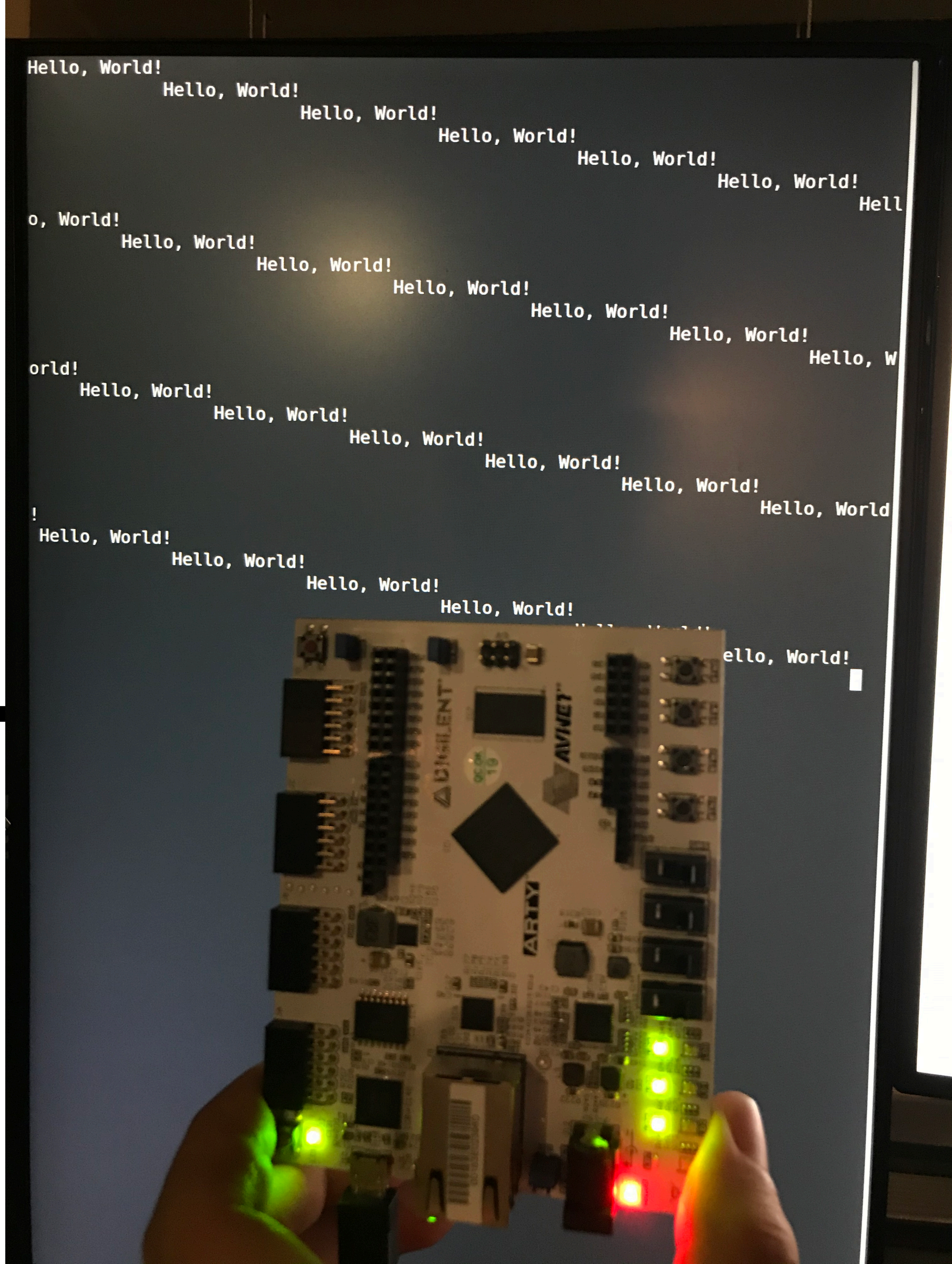
CS 3410
CS 4420

RISC-V



Sep 12, 2020

Summer 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!
                                                     Hello, World!
                                                                   ello, World!
```

ideal ≠ possible; OS ≠ hello-world

80%
CS 3410
CS 4420
RISC-V

Obstacles

Summer 2020

Fall 2020

Obstacles & Hope

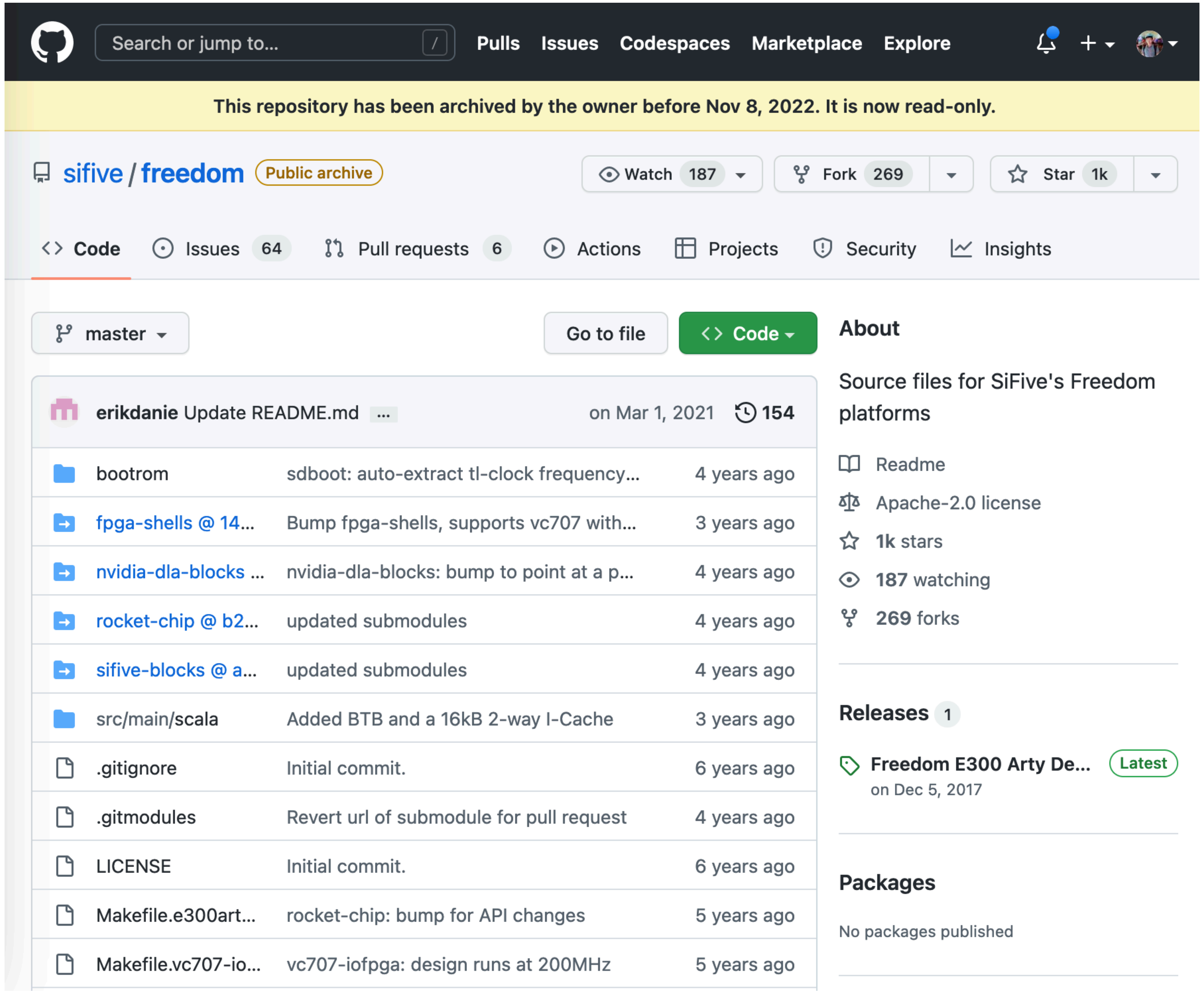
- ✗ Only 24KB memory
- ✗ No disk device
- ✓ Timer interrupt is supported
- ✓ Privilege levels and exceptions are supported
- ✓ CPU is well-documented and board is not too expensive

Obstacles & Hope: What to do?

✗ Need to modify the hardware design

✓ Need to write a kernel with the CPU support and documents

Background: 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

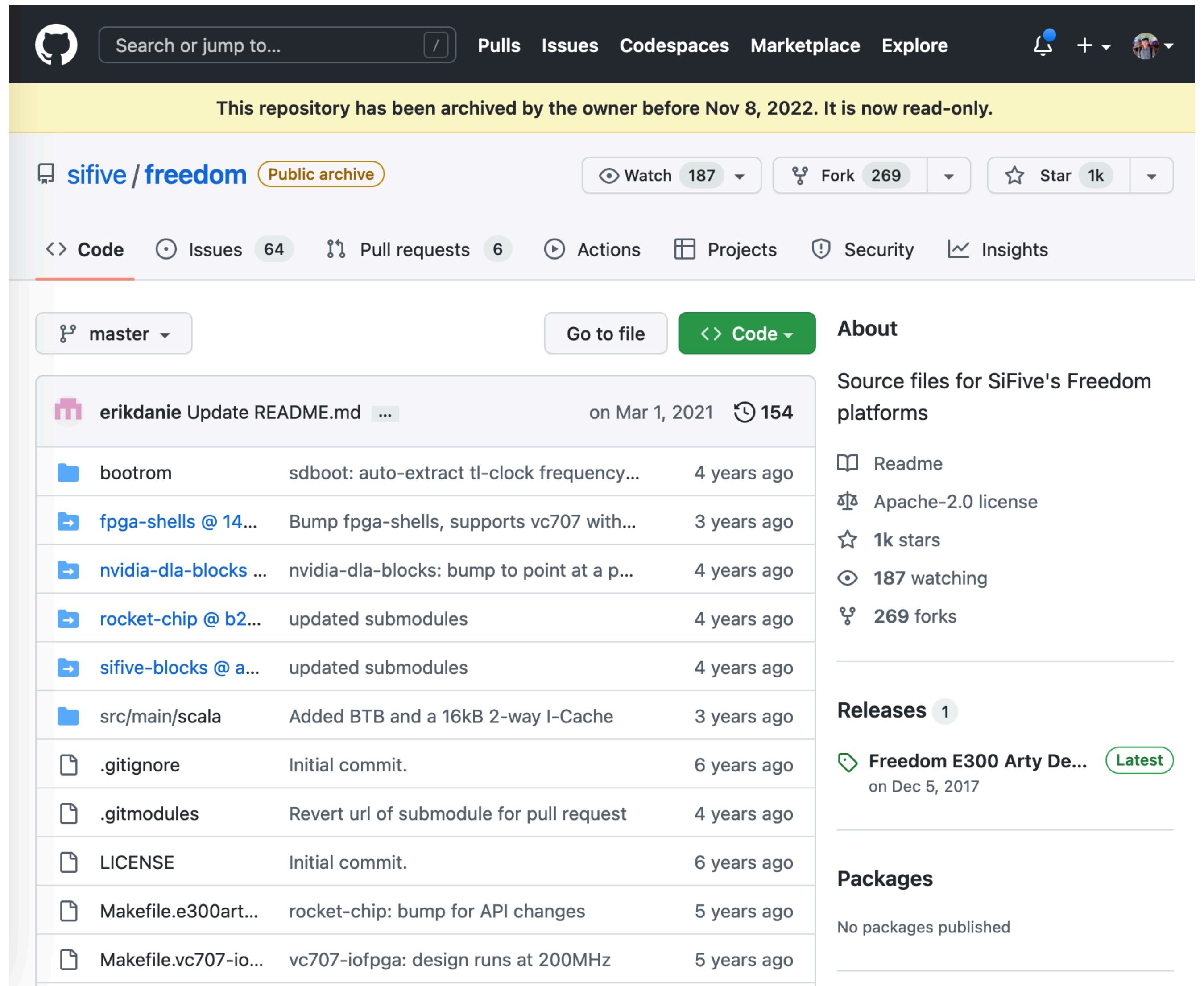
Releases 1

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

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

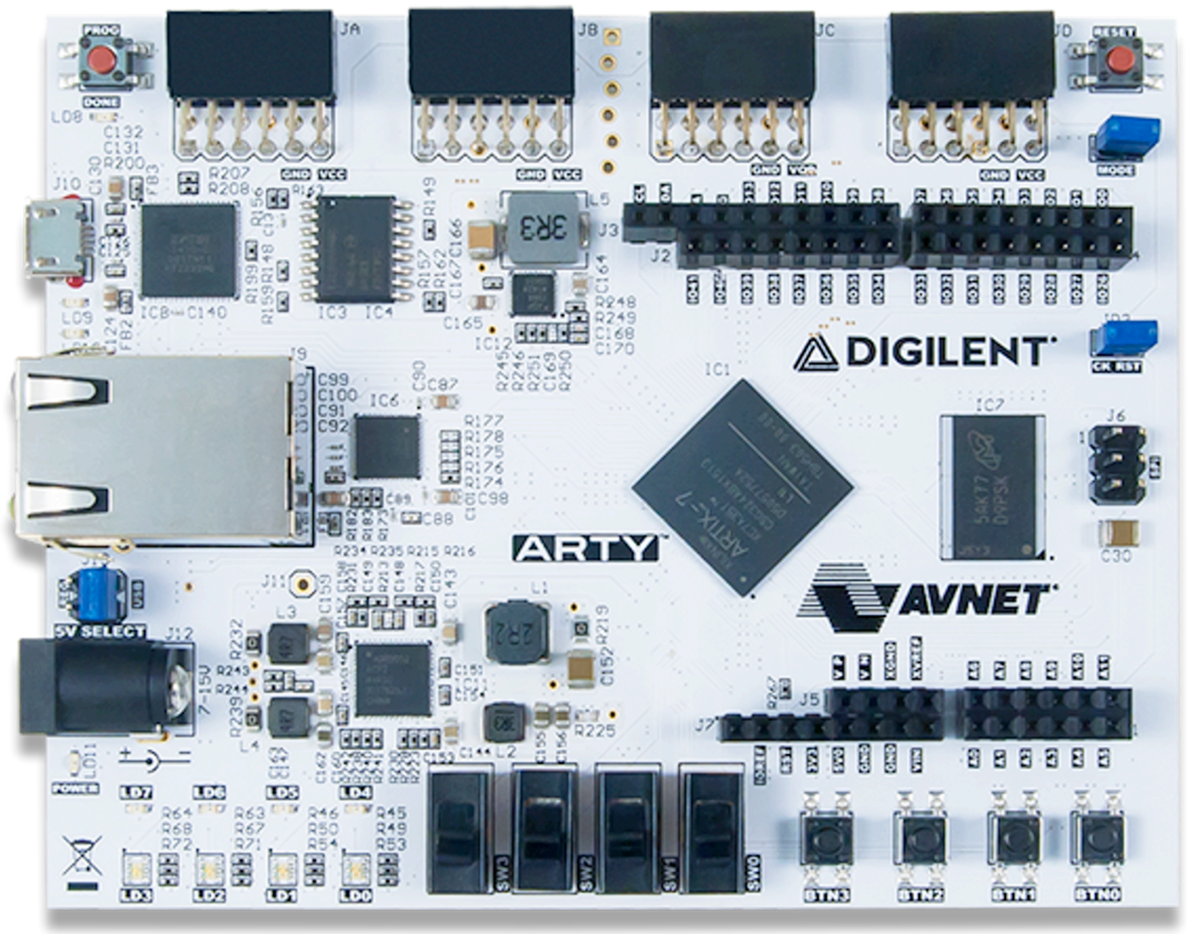
Releases 1

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

Packages

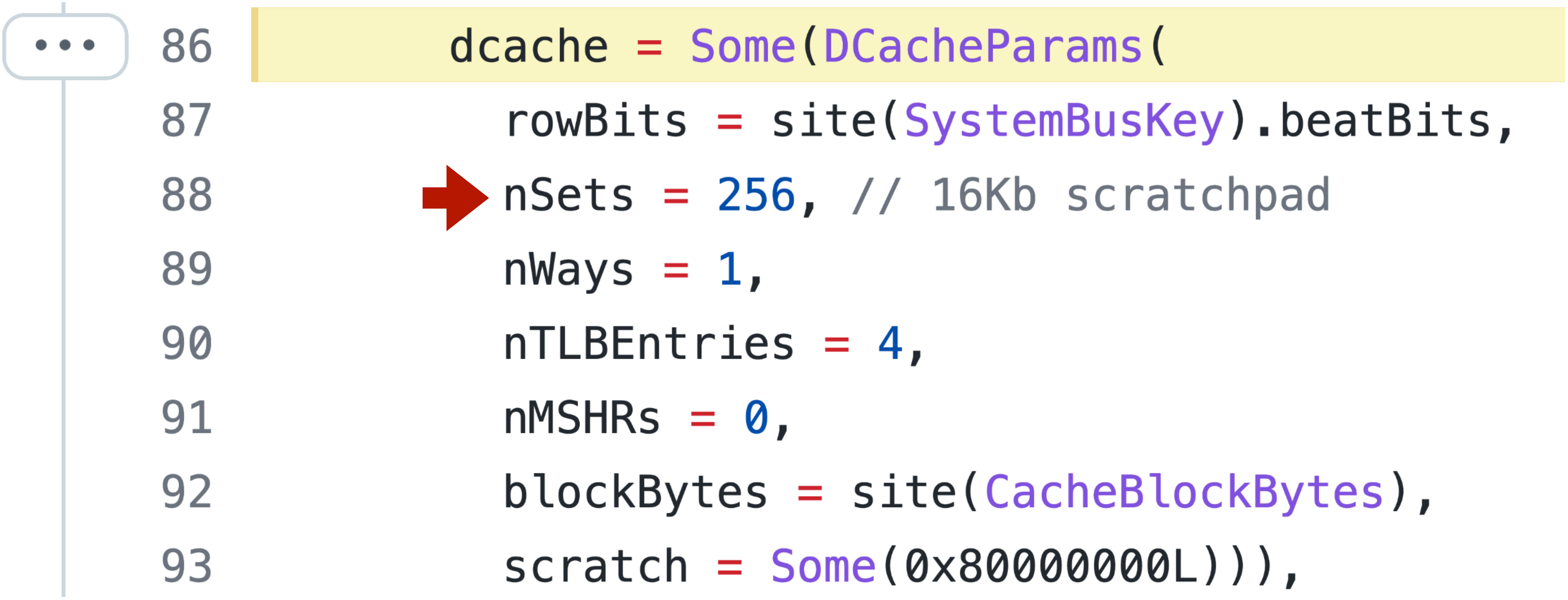
No packages published

A **binary file**
encoding the hardware design
(clocks, registers, circuits, etc.)



FPGA emulates the **hardware design**

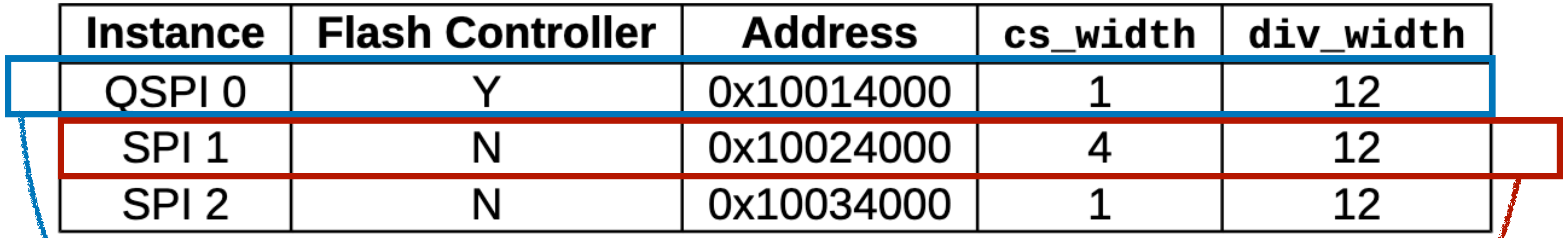
Idea #1: Increase the memory size



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

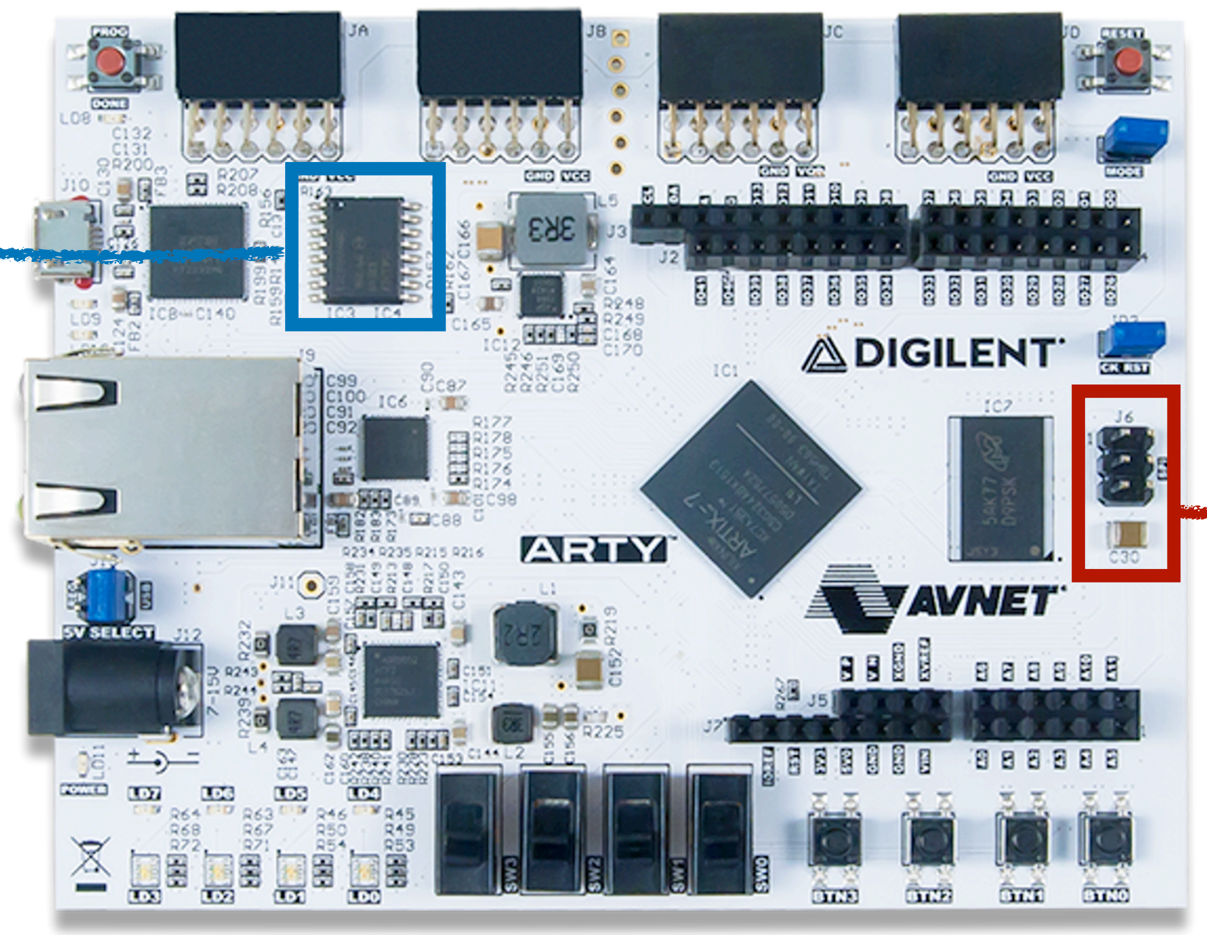
https://github.com/chipsalliance/rocket-chip/blob/b21c7879b3ea22f69cb8457109561f37c225f8ea/src/main/scala/subsystem/Configs.scala#L78

Background: SPI (simpler than USB)



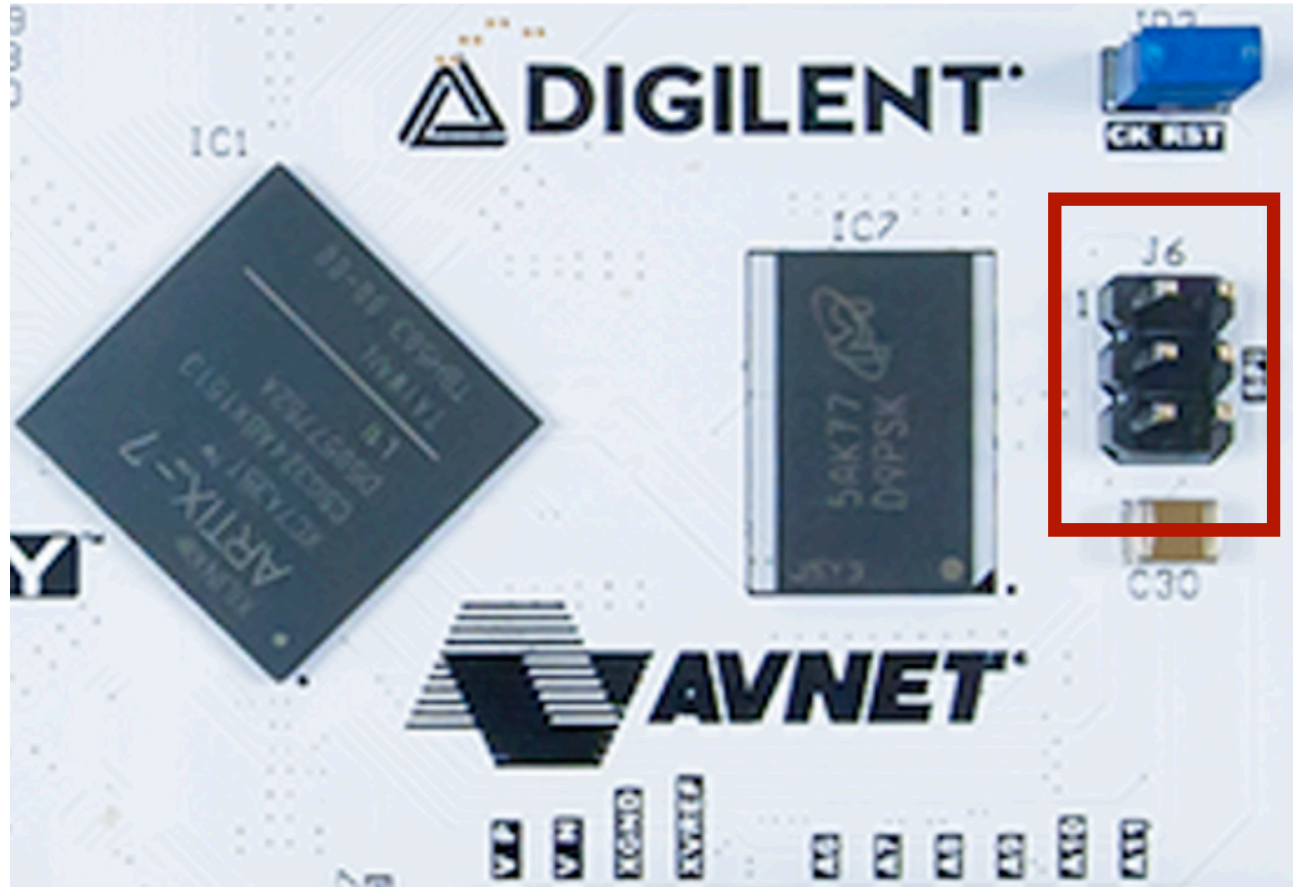
Instance	Flash Controller	Address	cs_width	div_width
QSPI 0	Y	0x10014000	1	12
SPI 1	N	0x10024000	4	12
SPI 2	N	0x10034000	1	12

Table 64: SPI Instances



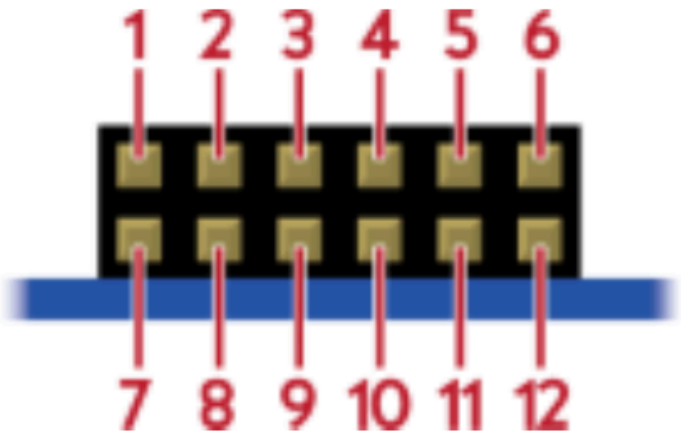
Chapter 19 of Sifive FE310 manual, v19p04
https://github.com/yhzhang0128/egos-2000/blob/main/references/sifive-fe310-v19p04.pdf

SPI: Serial Peripheral Interface

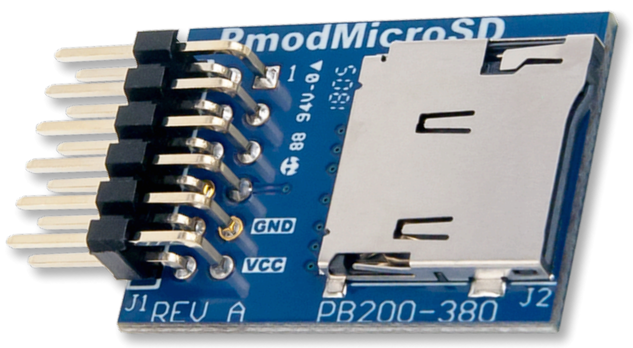


6 pins

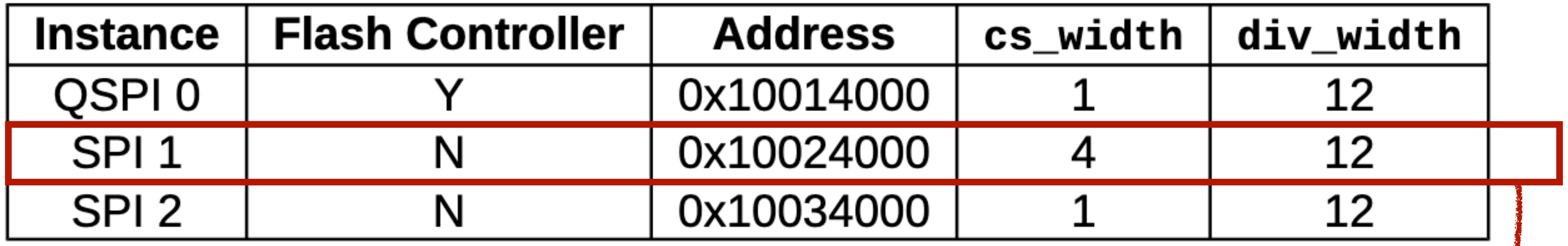
GND + VCC + SPI (4)



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

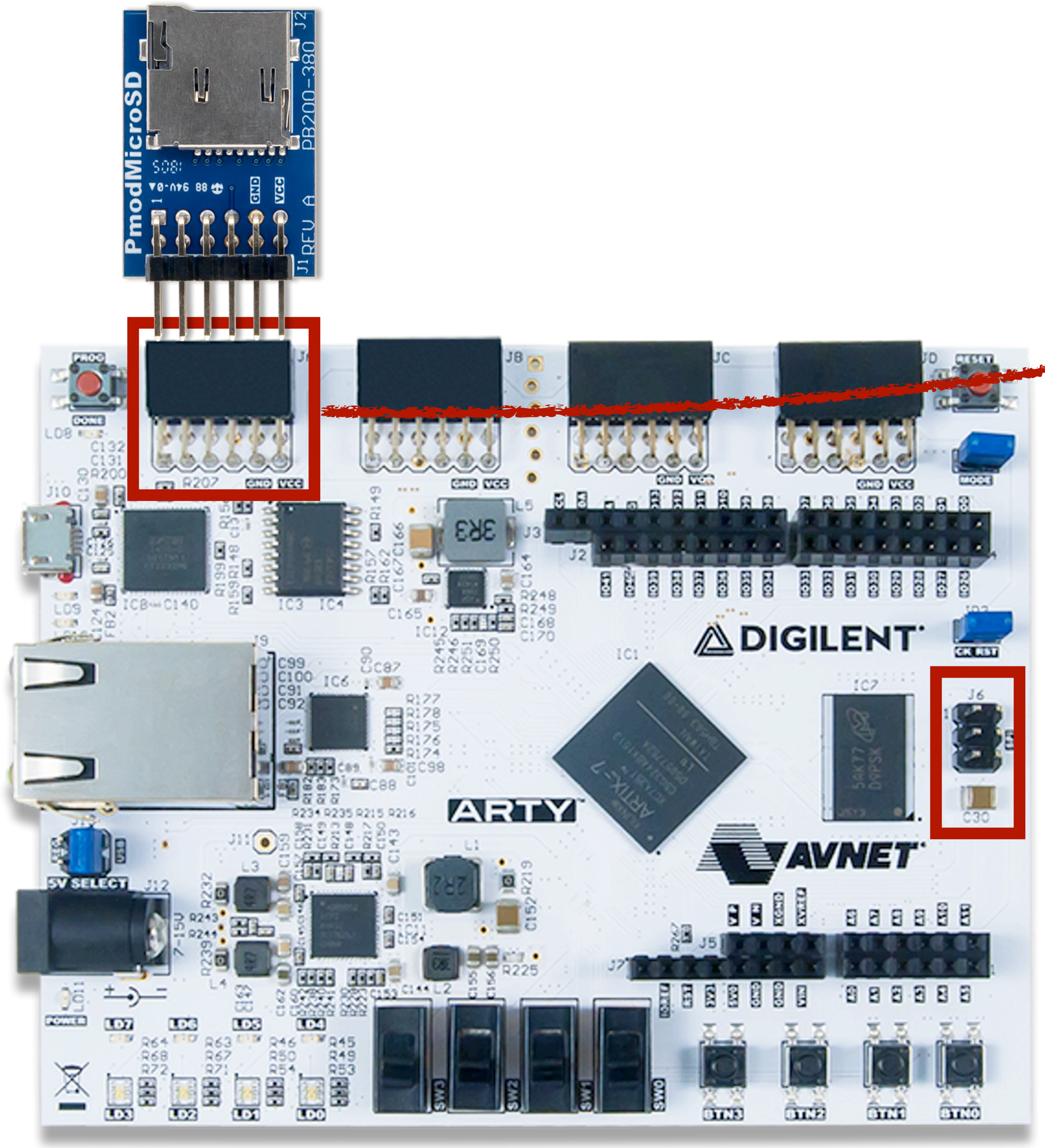


Idea #2: Remap SPI1 to a microSD card



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

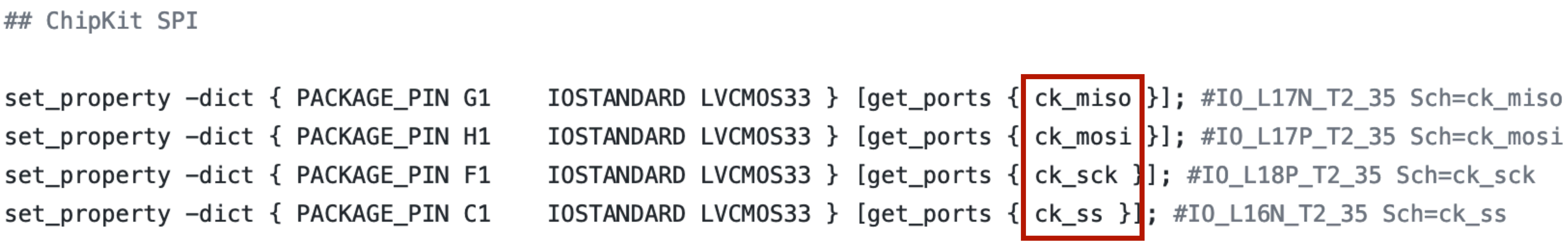
Table 64: SPI Instances



new SPI1

old SPI1

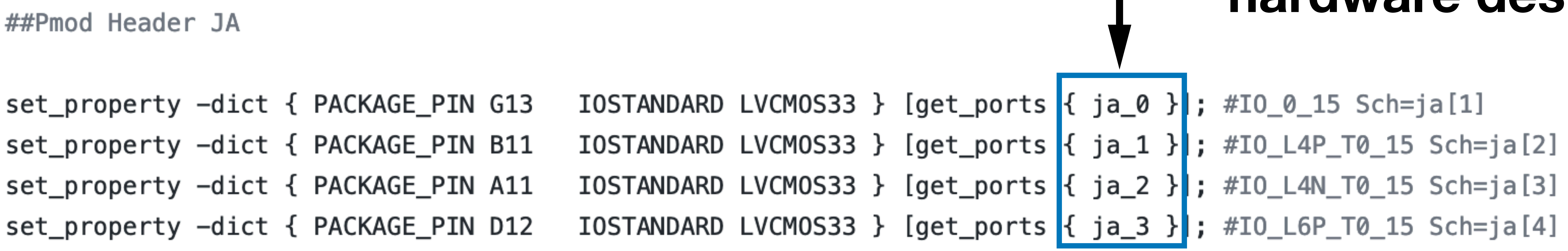
Idea #2: Remap SPI1 to a microSD card



```
## ChipKit SPI

set_property -dict { PACKAGE_PIN G1    IOSTANDARD LVCMOS33 } [get_ports { ck_miso }]; #IO_L17N_T2_35 Sch=ck_miso
set_property -dict { PACKAGE_PIN H1    IOSTANDARD LVCMOS33 } [get_ports { ck_mosi }]; #IO_L17P_T2_35 Sch=ck_mosi
set_property -dict { PACKAGE_PIN F1    IOSTANDARD LVCMOS33 } [get_ports { ck_sck }]; #IO_L18P_T2_35 Sch=ck_sck
set_property -dict { PACKAGE_PIN C1    IOSTANDARD LVCMOS33 } [get_ports { ck_ss }]; #IO_L16N_T2_35 Sch=ck_ss
```

Find and replace these 4 wires in the hardware design



```
##Pmod Header JA

set_property -dict { PACKAGE_PIN G13   IOSTANDARD LVCMOS33 } [get_ports { ja_0 }]; #IO_0_15 Sch=ja[1]
set_property -dict { PACKAGE_PIN B11   IOSTANDARD LVCMOS33 } [get_ports { ja_1 }]; #IO_L4P_T0_15 Sch=ja[2]
set_property -dict { PACKAGE_PIN A11   IOSTANDARD LVCMOS33 } [get_ports { ja_2 }]; #IO_L4N_T0_15 Sch=ja[3]
set_property -dict { PACKAGE_PIN D12   IOSTANDARD LVCMOS33 } [get_ports { ja_3 }]; #IO_L6P_T0_15 Sch=ja[4]
```

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

Coming up with ideas is difficult

No progress at all for more than a year.

Not sure whether it can work eventually.

Being the only person pushing this work.

Obstacles

Ideas * 2

Fall 2020

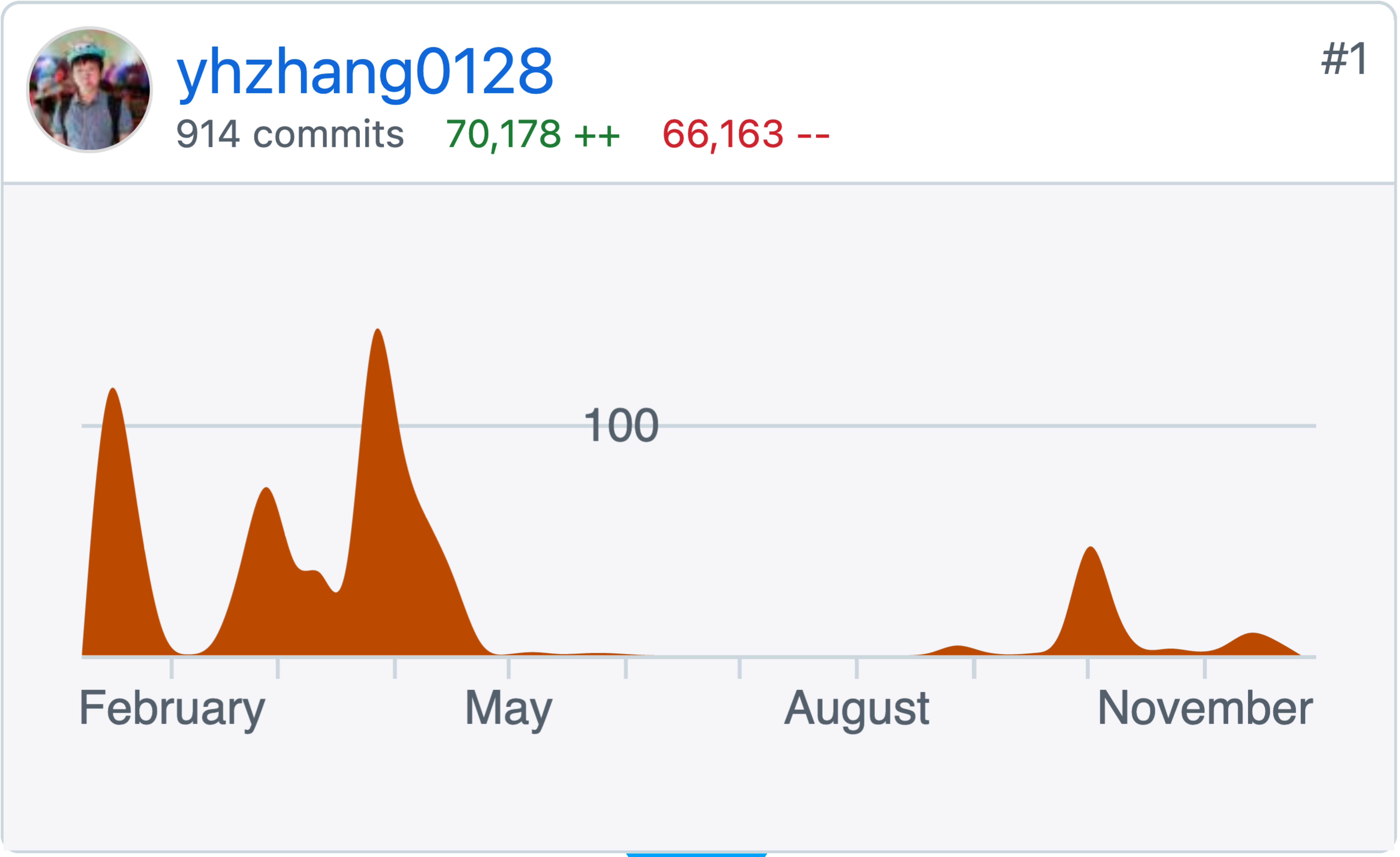
Fall 2021

Lesson

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

🥲

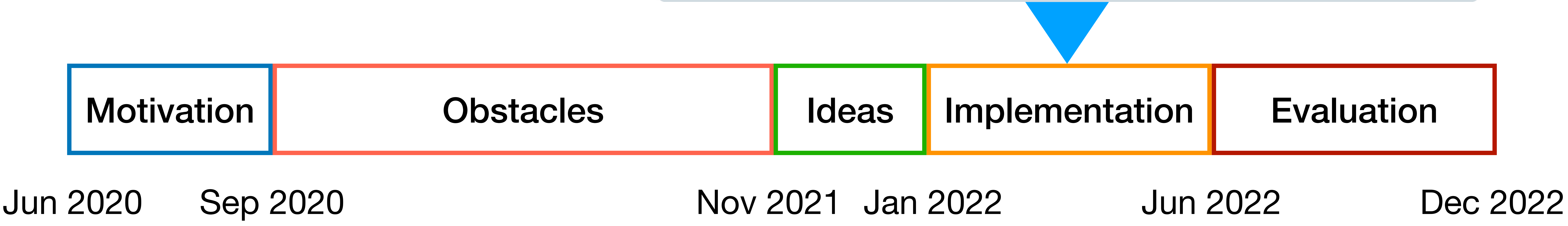
Motivation	Obstacles	Ideas	Implementation	Evaluation



yhzhang0128

#1

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

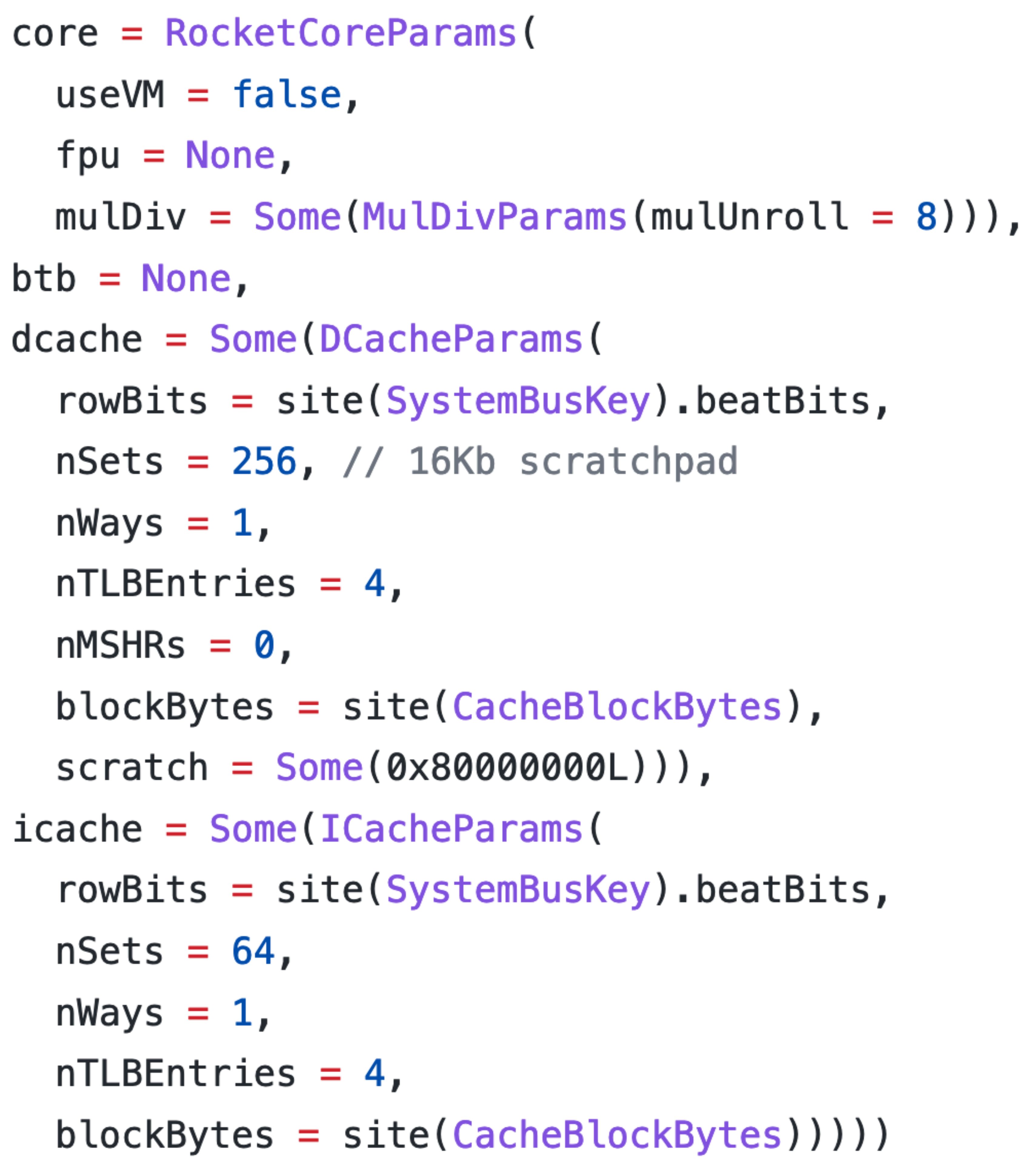


Motivation | Obstacles | Ideas | Implementation | Evaluation

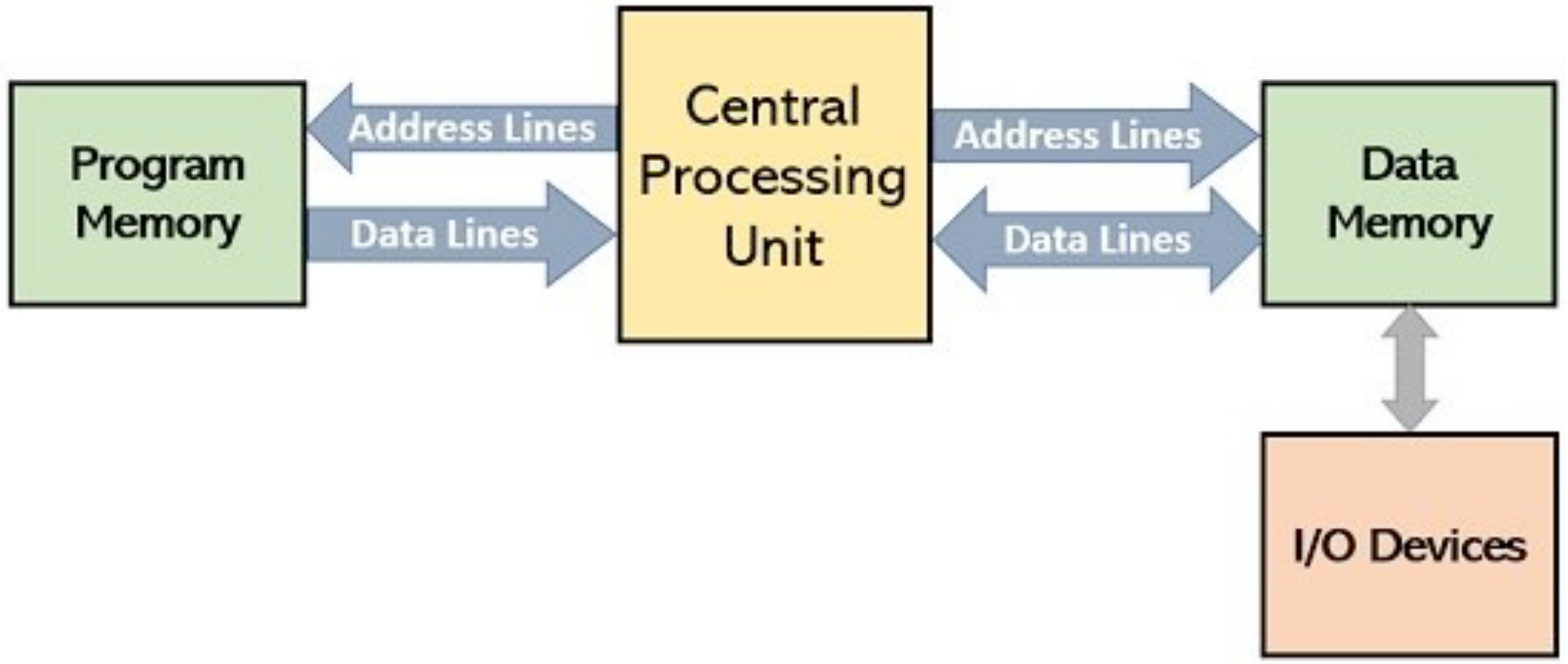
Jun 2020 | Sep 2020 | Nov 2021 | Jan 2022 | Jun 2022 | Dec 2022

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

A bug taking >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

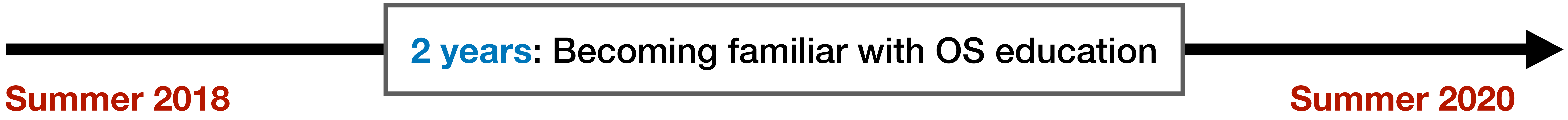
Lesson

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

Lessons about doing research

- Good motivations should convince non-experts why the work is valuable.
- Ideas are difficult to come up with and there is no guarantee of success.
- Implementing a system is non-trivial, taking hard work and determination.

The full 4.5-year research process

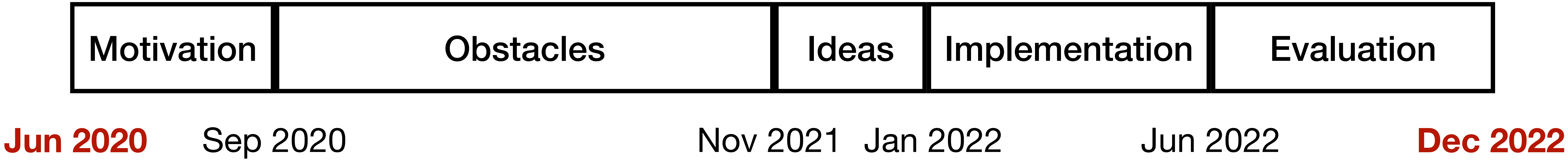


Summer 2018

2 years: Becoming familiar with OS education

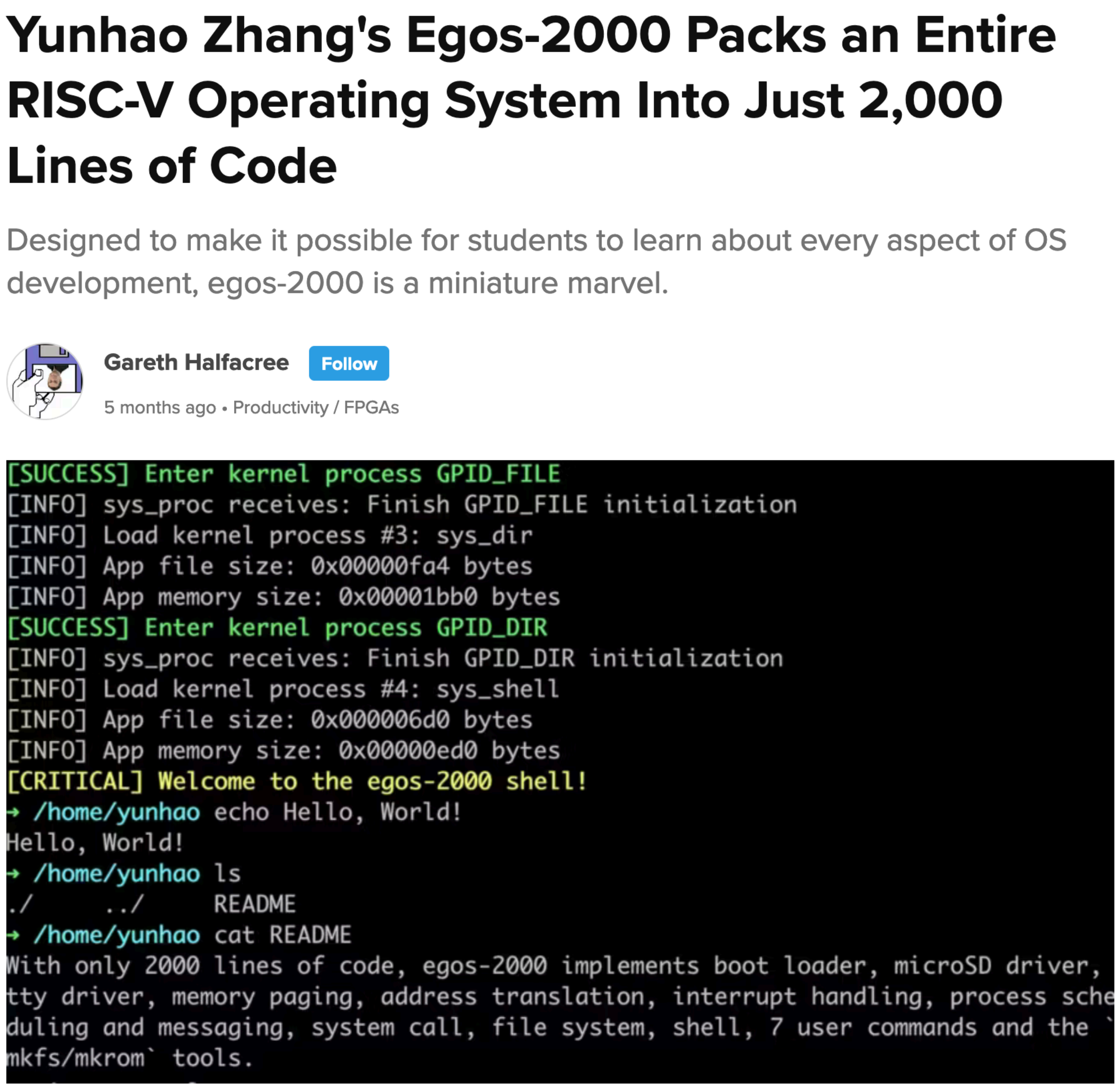
Summer 2020

Then, challenge the state-of-the-art



Jun 2020 | Motivation | Sep 2020 | Obstacles | Nov 2021 | Ideas | Jan 2022 | Implementation | Jun 2022 | Evaluation | **Dec 2022**

Research in the news



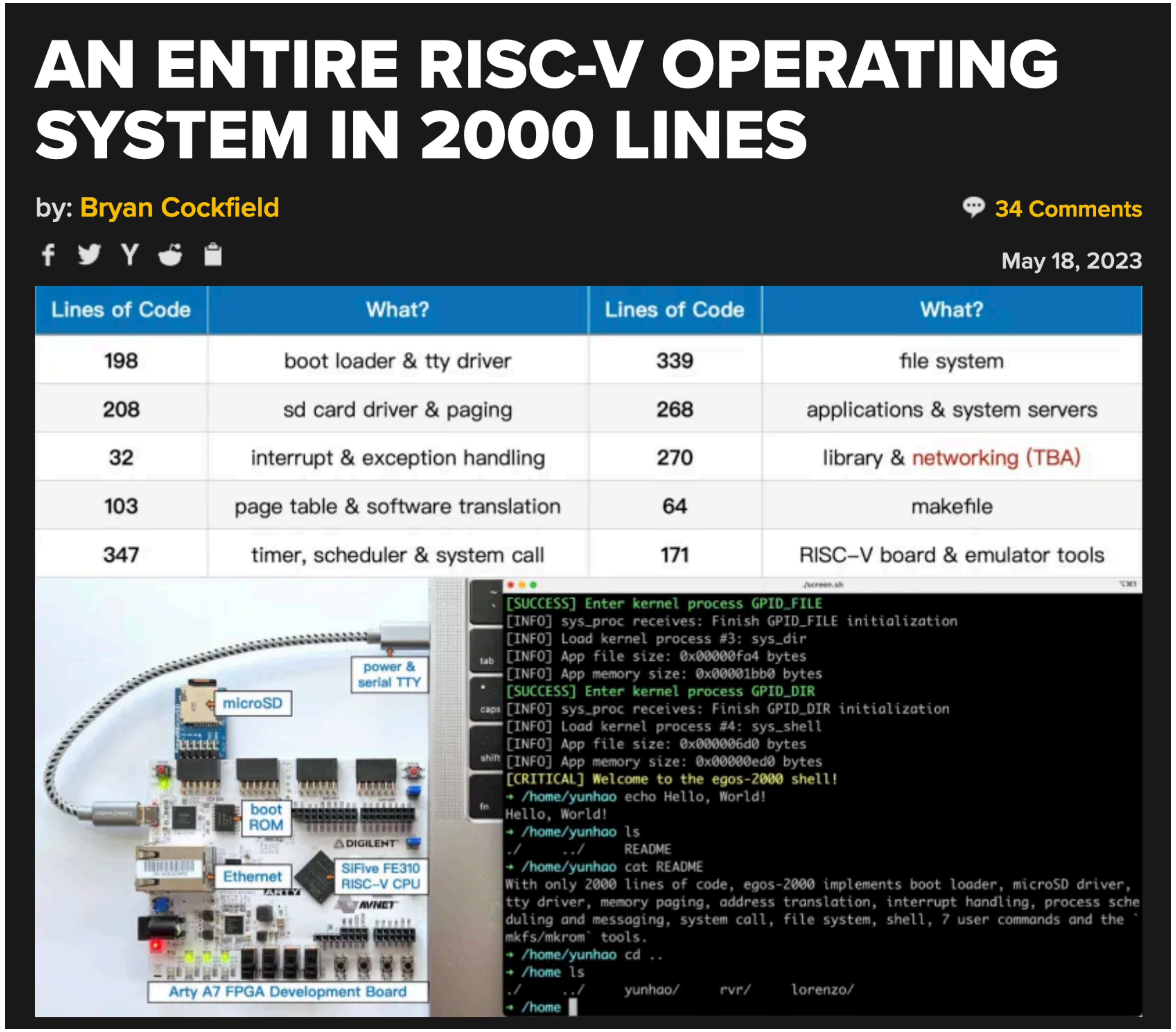
Yunhao Zhang's Egos-2000 Packs an Entire RISC-V Operating System Into Just 2,000 Lines of Code

Designed to make it possible for students to learn about every aspect of OS development, egos-2000 is a miniature marvel.

Gareth Halfacree Follow

5 months ago • Productivity / FPGAs

```
[SUCCESS] Enter kernel process GPID_FILE
[INFO] sys_proc receives: Finish GPID_FILE initialization
[INFO] Load kernel process #3: sys_dir
[INFO] App file size: 0x00000fa4 bytes
[INFO] App memory size: 0x00001bb0 bytes
[SUCCESS] Enter kernel process GPID_DIR
[INFO] sys_proc receives: Finish GPID_DIR initialization
[INFO] Load kernel process #4: sys_shell
[INFO] App file size: 0x000006d0 bytes
[INFO] App memory size: 0x00000ed0 bytes
[CRITICAL] Welcome to the egos-2000 shell!
→ /home/yunhao echo Hello, World!
Hello, World!
→ /home/yunhao ls
./      ../      README
→ /home/yunhao cat README
With only 2000 lines of code, egos-2000 implements boot loader, microSD driver,
tty driver, memory paging, address translation, interrupt handling, process sche
duling and messaging, system call, file system, shell, 7 user commands and the `
mkfs/mkrom` tools.
```



AN ENTIRE RISC-V OPERATING SYSTEM IN 2000 LINES

by: Bryan Cockfield

34 Comments

May 18, 2023

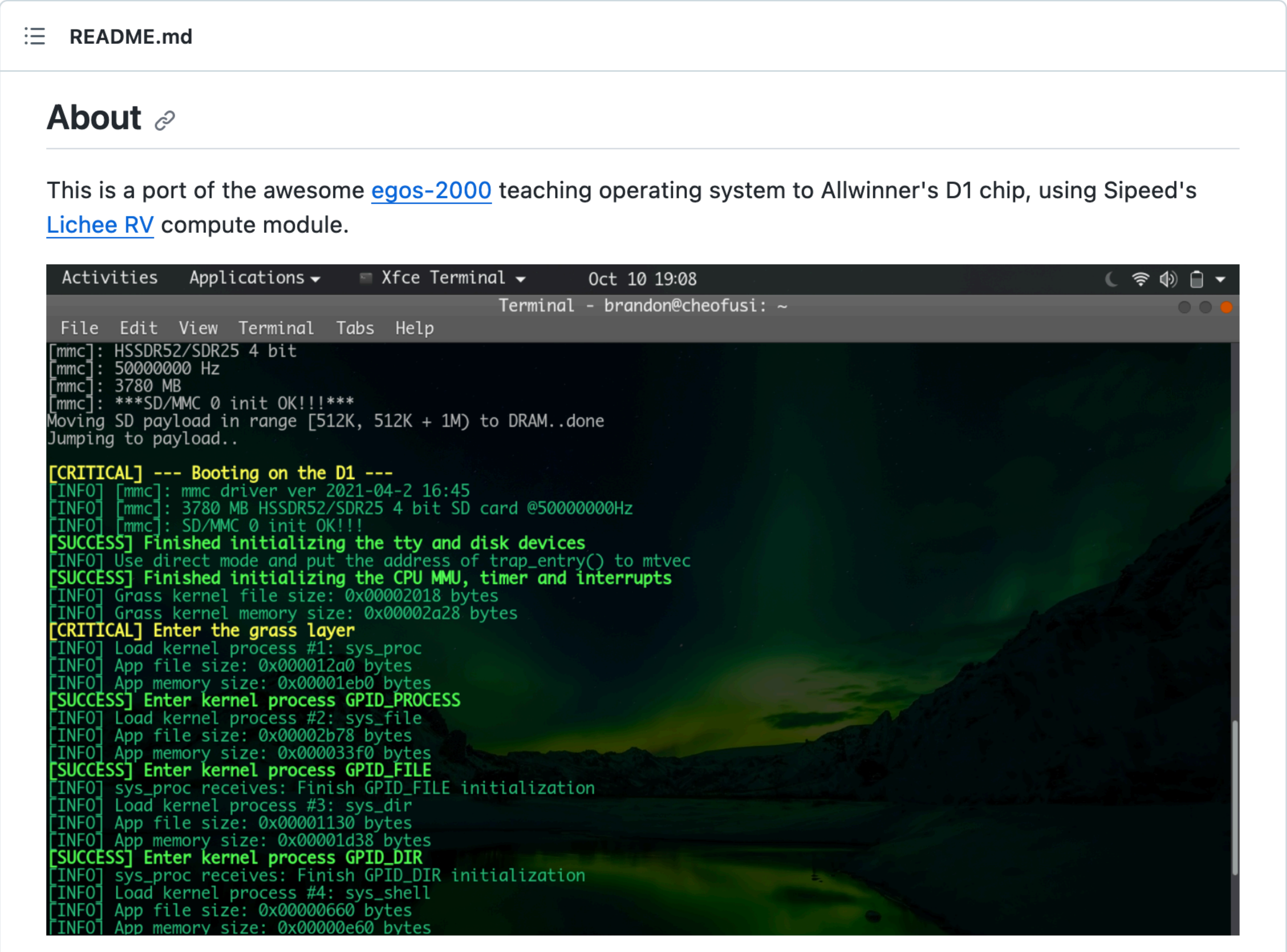
Lines of Code	What?	Lines of Code	What?
198	boot loader & tty driver	339	file system
208	sd card driver & paging	268	applications & system servers
32	interrupt & exception handling	270	library & networking (TBA)
103	page table & software translation	64	makefile
347	timer, scheduler & system call	171	RISC–V board & emulator tools

```
[SUCCESS] Enter kernel process GPID_FILE
[INFO] sys_proc receives: Finish GPID_FILE initialization
[INFO] Load kernel process #3: sys_dir
[INFO] App file size: 0x00000fa4 bytes
[INFO] App memory size: 0x00001bb0 bytes
[SUCCESS] Enter kernel process GPID_DIR
[INFO] sys_proc receives: Finish GPID_DIR initialization
[INFO] Load kernel process #4: sys_shell
[INFO] App file size: 0x000006d0 bytes
[INFO] App memory size: 0x00000ed0 bytes
[CRITICAL] Welcome to the egos-2000 shell!
→ /home/yunhao echo Hello, World!
Hello, World!
→ /home/yunhao ls
./      ../      README
→ /home/yunhao cat README
With only 2000 lines of code, egos-2000 implements boot loader, microSD driver,
tty driver, memory paging, address translation, interrupt handling, process sche
duling and messaging, system call, file system, shell, 7 user commands and the `
mkfs/mkrom` tools.
→ /home/yunhao cd ..
→ /home ls
./      ../      yunhao/      rvr/      lorenzo/
→ /home
```

https://hackaday.com/2023/05/18/an-entire-risc-v-operating-system-in-2000-lines/

https://www.hackster.io/news/yunhao-zhang-s-egos-2000-packs-an-entire-risc-v-operating-system-into-just-2-000-lines-of-code-2ba9875524a7

Follow-up from OS hobbyists



README.md

About

This is a port of the awesome egos-2000 teaching operating system to Allwinner's D1 chip, using Sipeed's Lichee RV compute module.

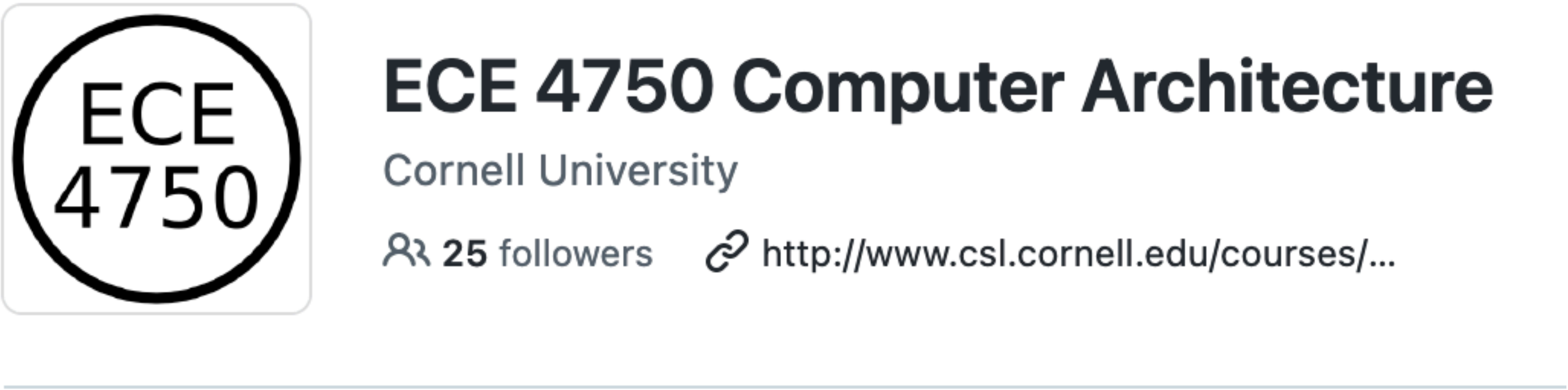
```
[mmc]: HSSDR52/SDR25 4 bit
[mmc]: 50000000 Hz
[mmc]: 3780 MB
[mmc]: ***SD/MMC 0 init OK!!!***
Moving SD payload in range [512K, 512K + 1M) to DRAM..done
Jumping to payload..

[CRITICAL] --- Booting on the D1 ---
[INFO] [mmc]: mmc driver ver 2021-04-2 16:45
[INFO] [mmc]: 3780 MB HSSDR52/SDR25 4 bit SD card @50000000Hz
[INFO] [mmc]: SD/MMC 0 init OK!!!
[SUCCESS] Finished initializing the tty and disk devices
[INFO] Use direct mode and put the address of trap_entry() to mtvec
[SUCCESS] Finished initializing the CPU MMU, timer and interrupts
[INFO] Grass kernel file size: 0x00002018 bytes
[INFO] Grass kernel memory size: 0x00002a28 bytes
[CRITICAL] Enter the grass layer
[INFO] Load kernel process #1: sys_proc
[INFO] App file size: 0x000012a0 bytes
[INFO] App memory size: 0x00001eb0 bytes
[SUCCESS] Enter kernel process GPID_PROCESS
[INFO] Load kernel process #2: sys_file
[INFO] App file size: 0x00002b78 bytes
[INFO] App memory size: 0x000033f0 bytes
[SUCCESS] Enter kernel process GPID_FILE
[INFO] sys_proc receives: Finish GPID_FILE initialization
[INFO] Load kernel process #3: sys_dir
[INFO] App file size: 0x00001130 bytes
[INFO] App memory size: 0x00001d38 bytes
[SUCCESS] Enter kernel process GPID_DIR
[INFO] sys_proc receives: Finish GPID_DIR initialization
[INFO] Load kernel process #4: sys_shell
[INFO] App file size: 0x00000660 bytes
[INFO] App memory size: 0x00000e60 bytes
```

Sipeed's Lichee RV64 board

https://github.com/cheofusi/egos-2000-d1

Future work

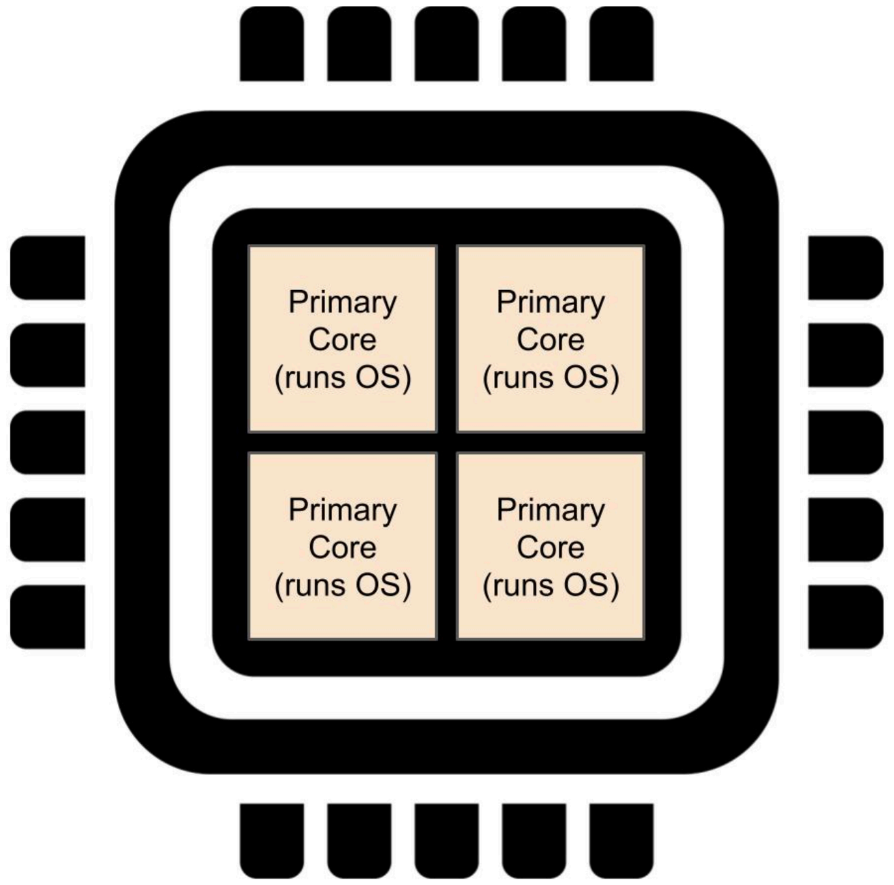


ECE 4750 Computer Architecture

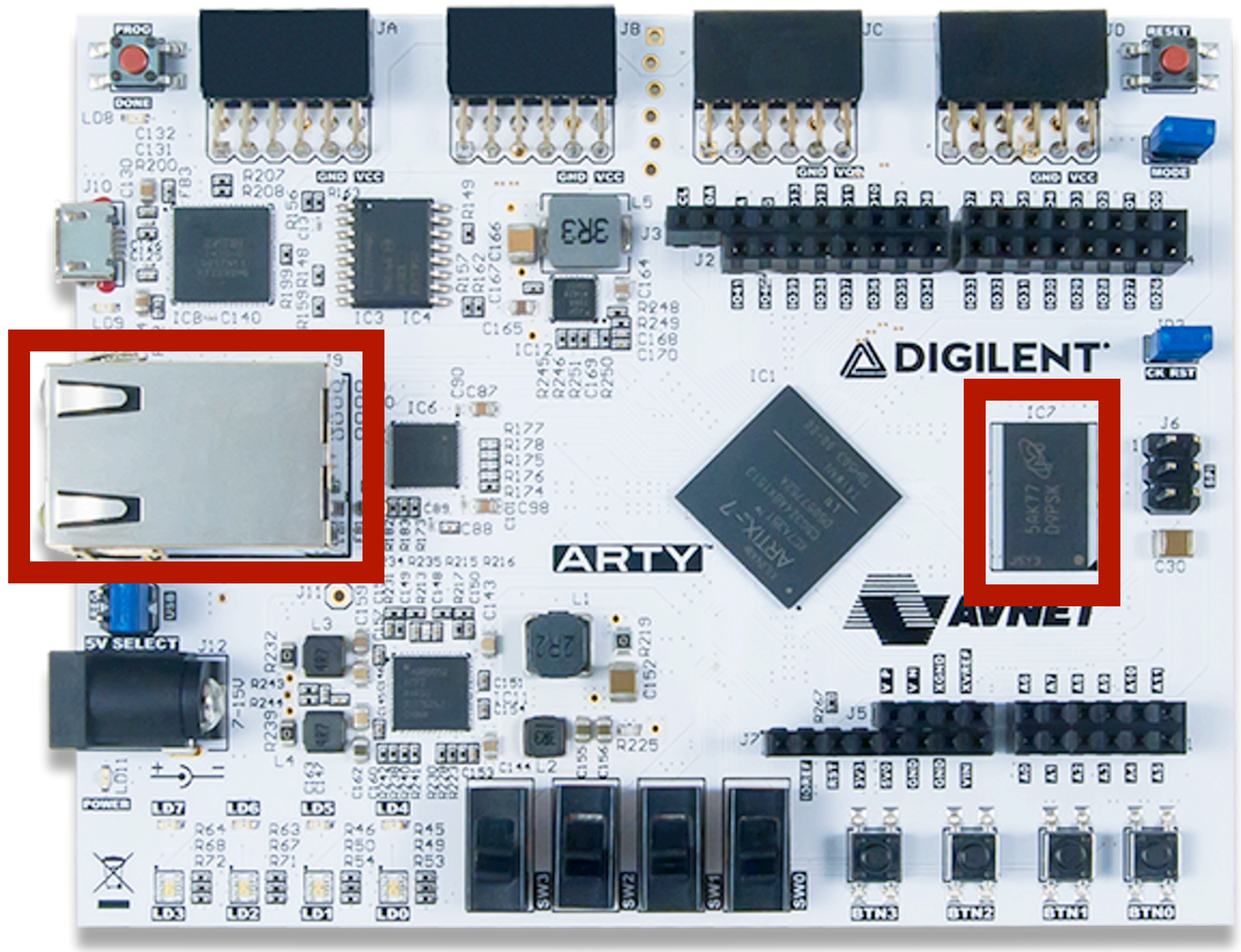
Cornell University

25 followers http://www.csl.cornell.edu/courses/...

Connect with our **ECE4750**



Enable **multi-core** in QEMU
and implement **locks** in egos-2000



Leverage the **256MB** DDR memory
and the **Ethernet** port on the Arty board

Vision

This project's vision is to help **every** college student read **all** the code of an operating system.

Lines of Code	What?	Lines of Code	What?
153	boot loader & tty driver	**336**	file system
240	sd card driver & paging	**320**	applications & system servers
32	interrupt & exception handling	**272**	library & networking (TBA)
108	page table & software translation	**64**	makefile
341	timer, scheduler & system call	**134**	RISC–V board & emulator tools