Writing an operating system in 2.5 years

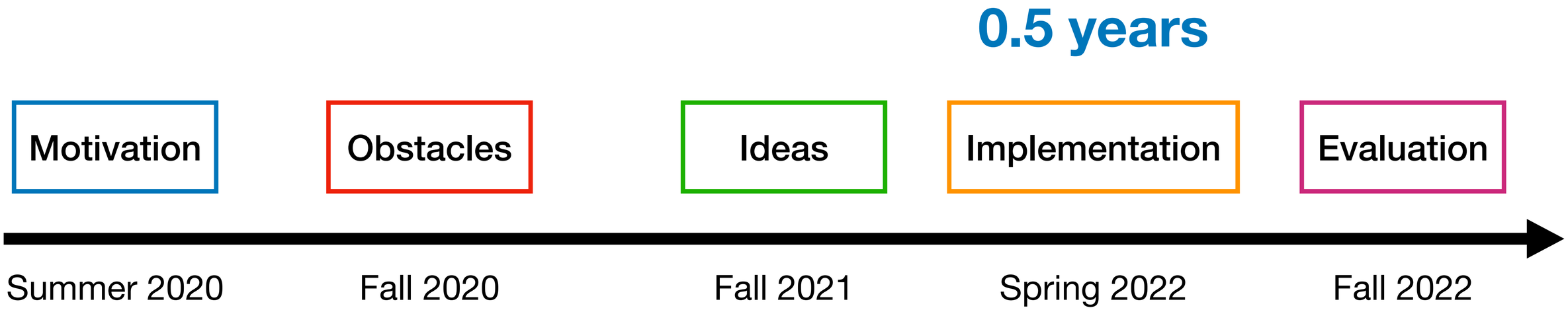
Yunhao Zhang

But first, writing an OS in 0.5 years

- P0: understand C and user-level instructions
- P1: understand context-switch and multi-threading
- P2: understand exception and privilege levels
- P3: understand file system abstractions
- P4: understand file system implementation
- P5: understand bus and I/O devices

https://github.com/yhzhang0128/egos-2000/blob/main/earth/sd/sd_rw.c#L13

Why 2.5 years? An overview



In June 2020, we created egos

~20K lines of code

run on Intel/Arm CPU

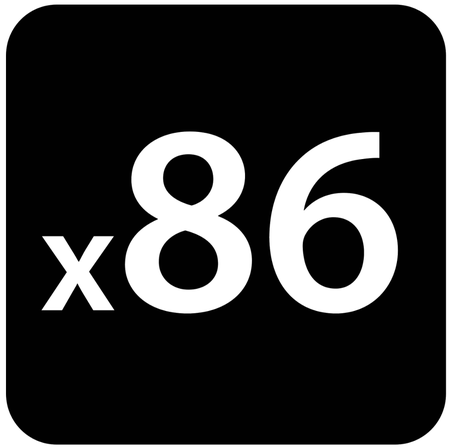
run as a Linux / MacOS process

20K lines of code

Students read a **very small** portion

2K lines of code

Students read a **large** portion



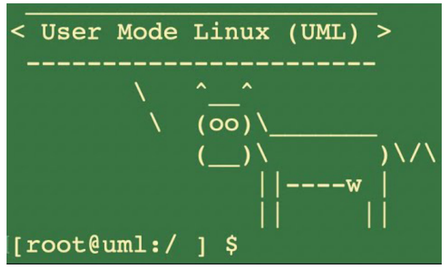
Intel x86 (1987)

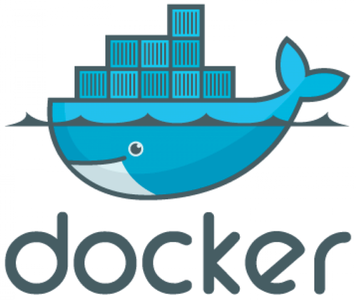
CPU document has
several thousands of pages

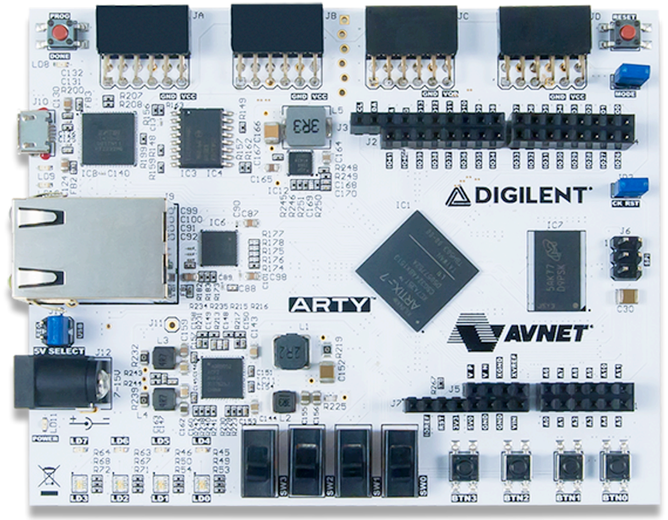
CS 3410
CS 4420

RISC-V (2010)

CPU document
has **<100** of pages







User-mode OS

Easier to deploy and run

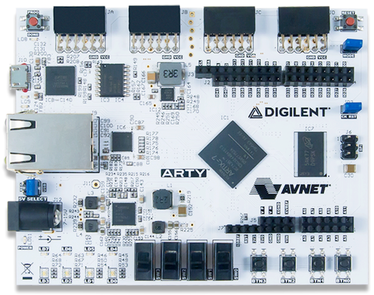
OS on hardware

More realistic and fun

The motivation

CS 3410
CS 4420

RISC-V®



~20K → ~2K

x86 → RISC-V

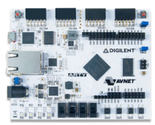
Linux / MacOS → hardware

Lesson

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

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

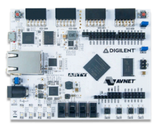
Yet, ideal ≠ possible



Obstacles

Summer 2020 Fall 2020

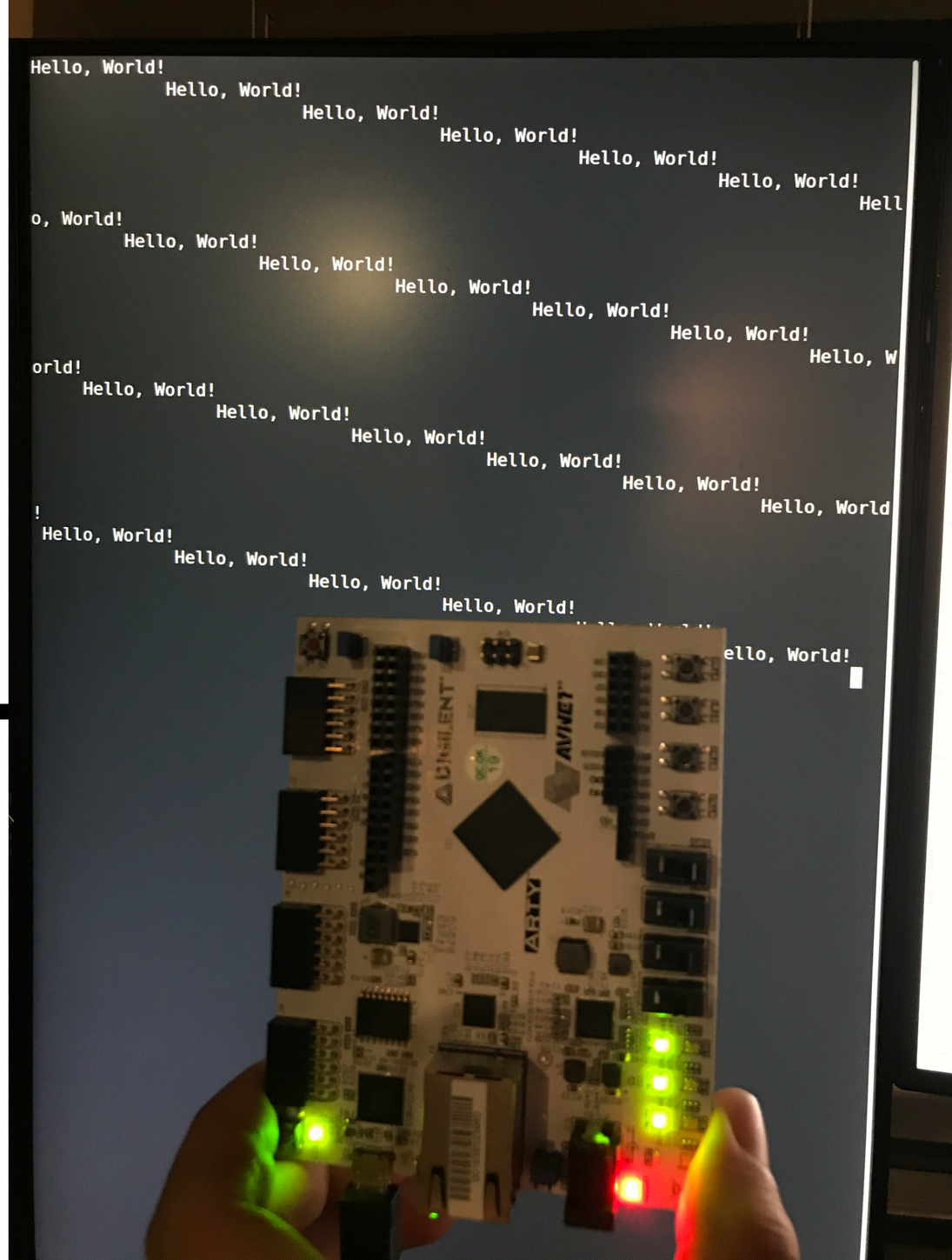
Hello World



Obstacles

Summer 2020 Fall 2020

Sep 12, 2020



Obstacles & Hope

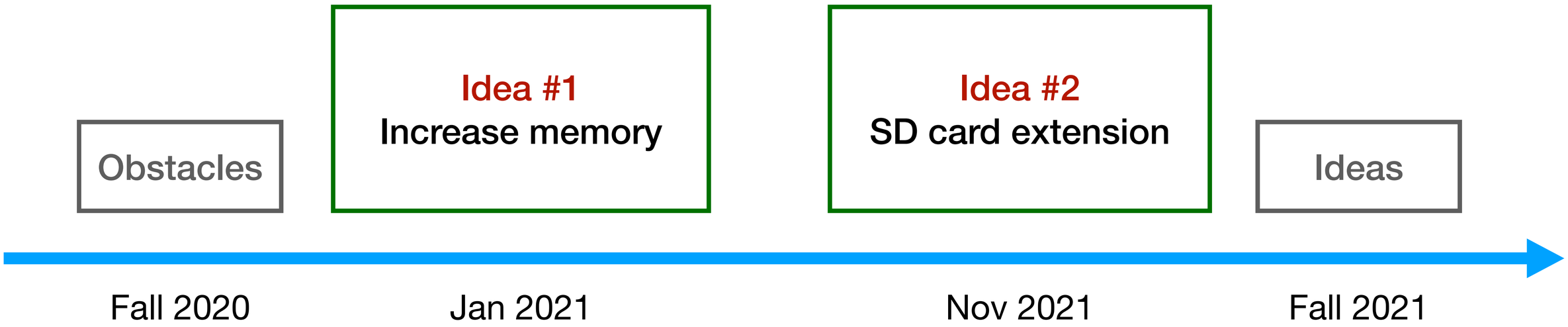

The hardware had only 24KB memory


There was no disk

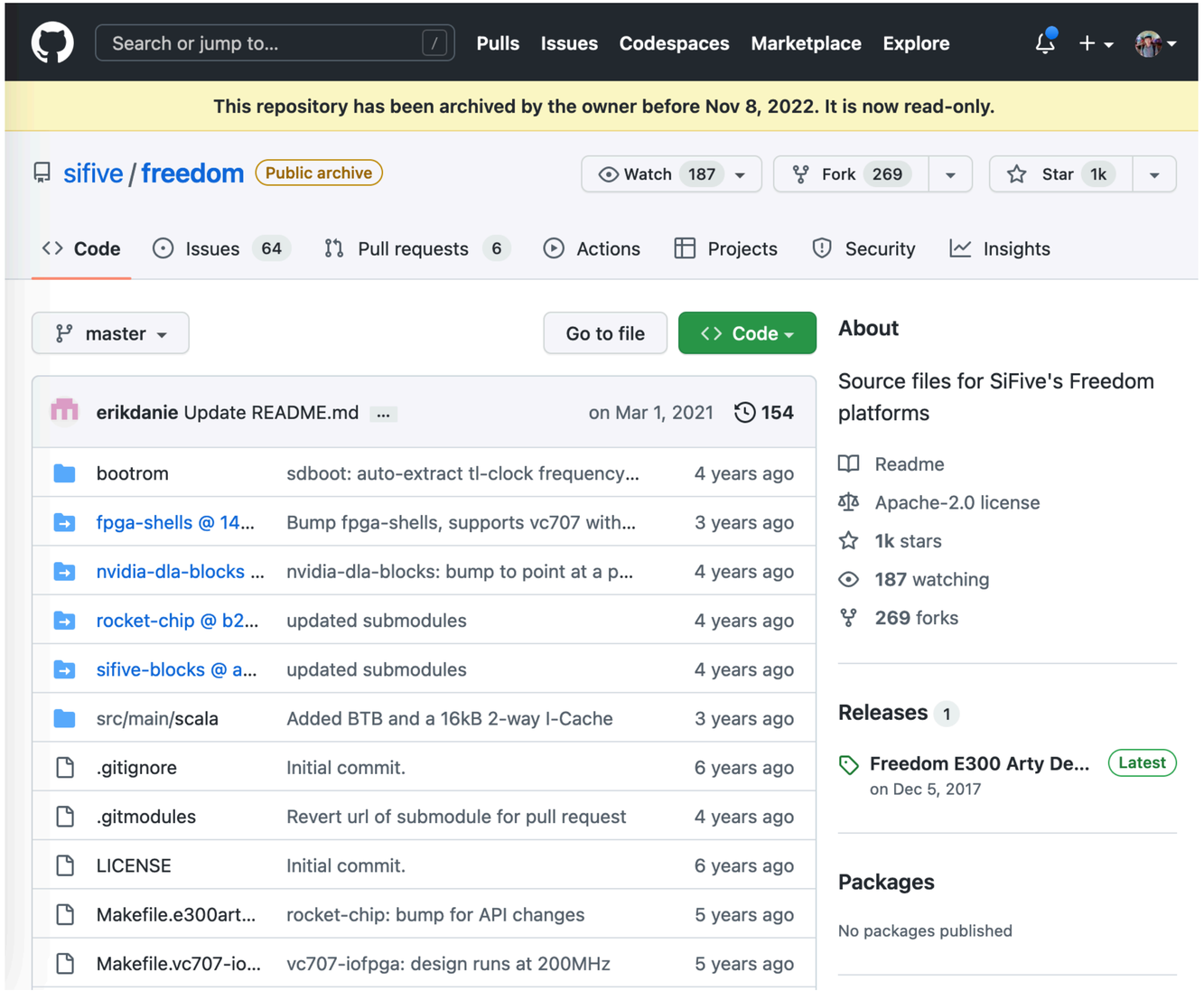

The hardware supports timer interrupt


The hardware supports privilege levels and exceptions

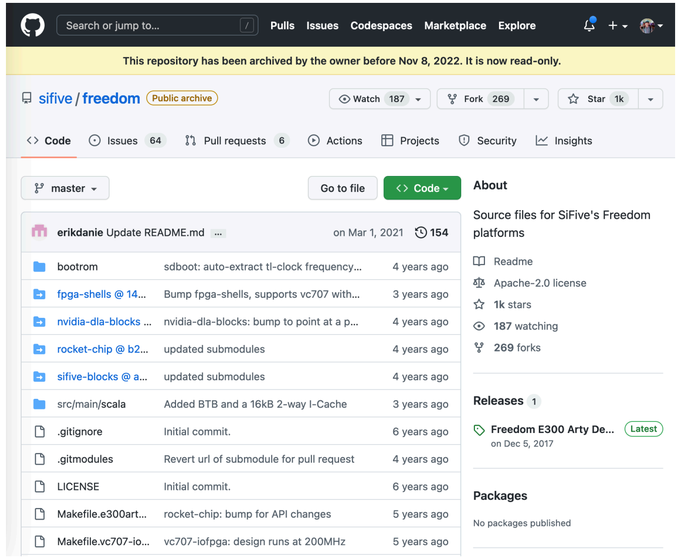
Overcome obstacles with ideas!



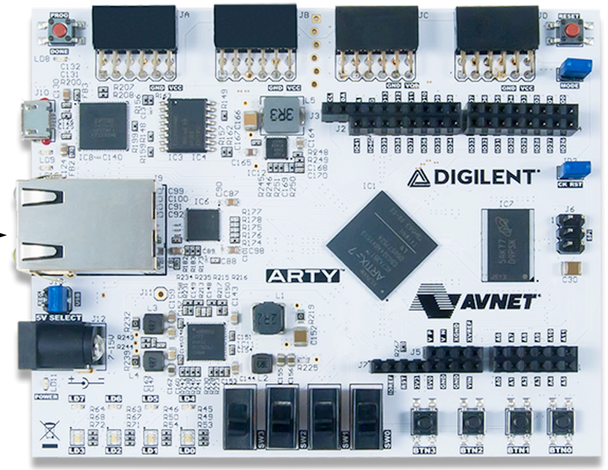
Open-source hardware



Running open-source hardware



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



FPGA: emulate the **hardware design**

Idea #1: Increase memory

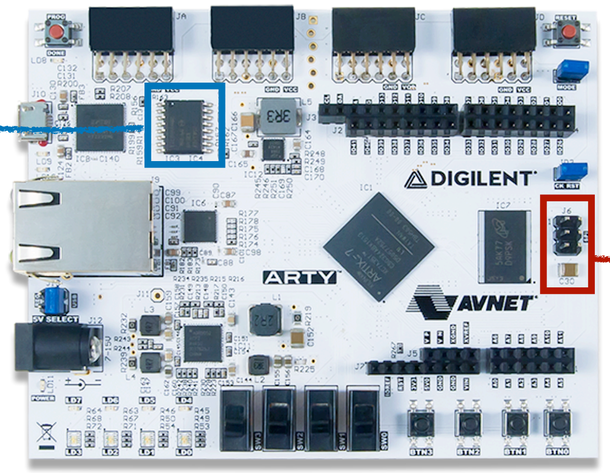
```
86        dcache = Some(DCacheParams(
87          rowBits = site(SystemBusKey).beatBits,
88          nSets = 256, // 16Kb scratchpad
89          nWays = 1,
90          nTLBEntries = 4,
91          nMSHRs = 0,
92          blockBytes = site(CacheBlockBytes),
93          scratch = Some(0x80000000L))),
```

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

Background of the SPI bus

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



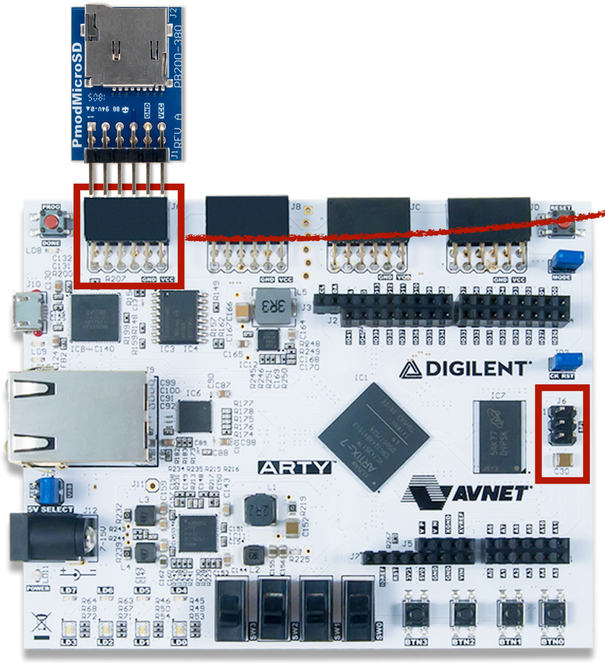
Chapter 19 of Sifive FE310 manual, v19p04
https://github.com/yhzhang0128/egos-2000/blob/main/references/sifive-fe310-v19p04.pdf

Remap SPI1 to 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

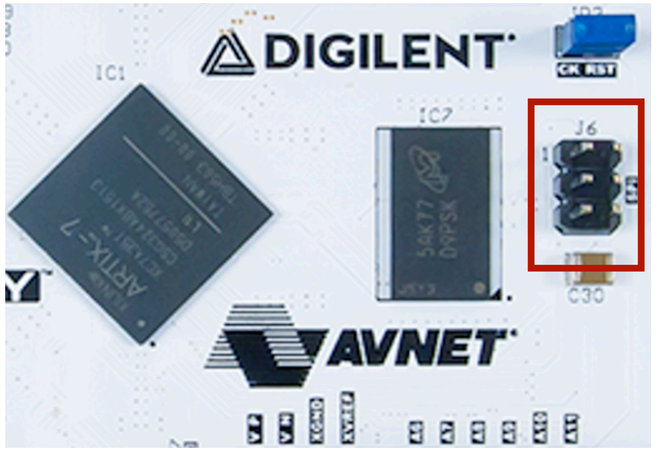
Table 64: SPI Instances

new SPI1



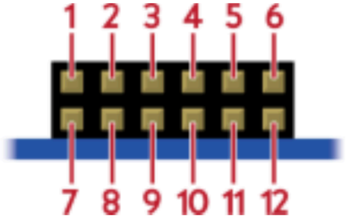
old SPI1

Remap SPI1 to microSD card

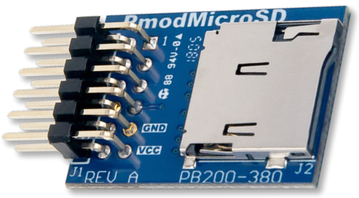


6 pins

GND + VCC + SPI (4)



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



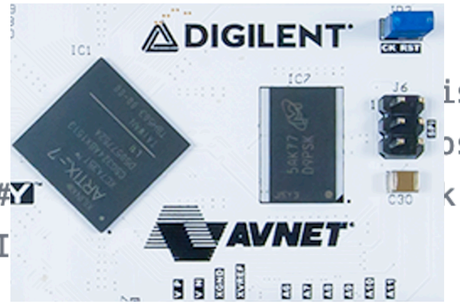
Old SPI1 mapping

New SPI1 mapping

Idea #2: Using microSD card as disk

```
## ChipKit SPI

set_property -dict { PACKAGE_PIN G1    IOSTANDARD LVCMOS33 } [get_ports { ck_miso }];
set_property -dict { PACKAGE_PIN H1    IOSTANDARD LVCMOS33 } [get_ports { ck_mosi }];
set_property -dict { PACKAGE_PIN F1    IOSTANDARD LVCMOS33 } [get_ports { ck_sck }]; #
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
set_property -dict { PACKAGE_PIN A11   IOSTANDARD LVCMOS33 } [get_ports { ja_2 }]; #I
set_property -dict { PACKAGE_PIN D12   IOSTANDARD LVCMOS33 } [get_ports { ja_3 }]; #IO_L
```

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

The road to 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

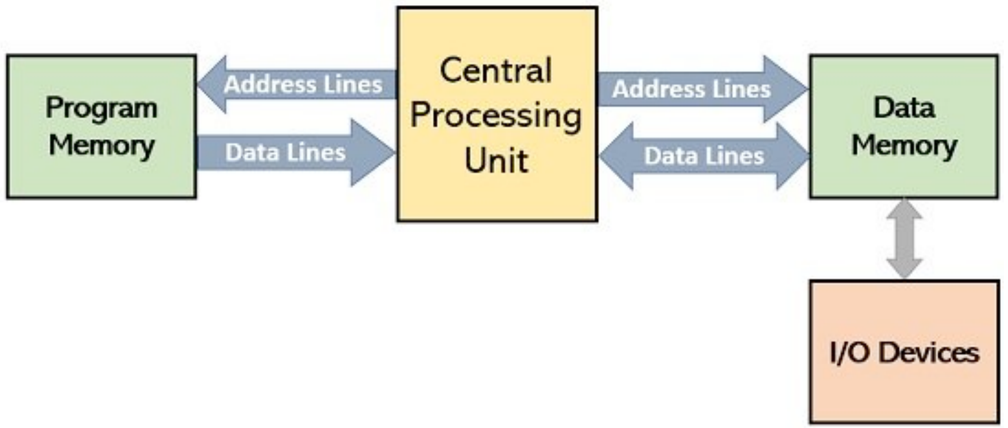
Motivation | Obstacles | Ideas | Implementation | Evaluation

Summer 2020 | Fall 2020 | Fall 2021 | Spring 2022 | Fall 2022

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

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

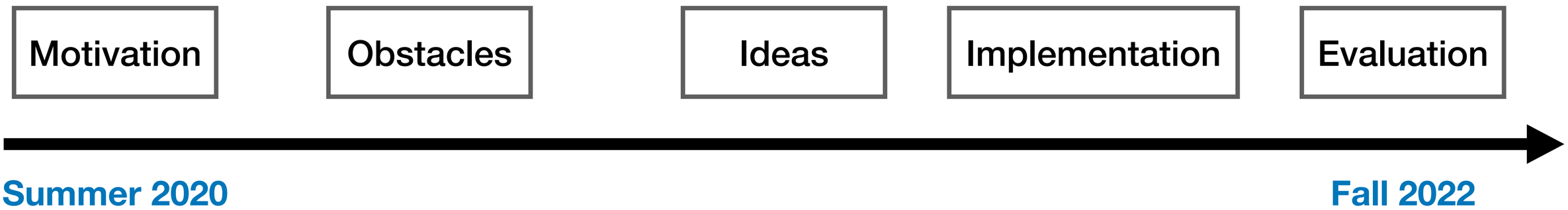
https://github.com/chipsalliance/rocket-chip/blob/b21c7879b3ea22f69cb8457109561f37c225f8ea/src/main/scala/subsystem/Configs.scala#L78

Lesson

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

A 4.5-year research process

Then, challenge state-of-the-art



Next step: 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