国立大学法人
電気通信大学
The University of Electro-Communications

UEC Seminar
2022/12

Computer Design Classes Using Open-source RISC-V: Linux Bootable SoC on FPGA and VLSI Implementations

Trong-Thuc Hoang and Cong-Kha Pham

Outline

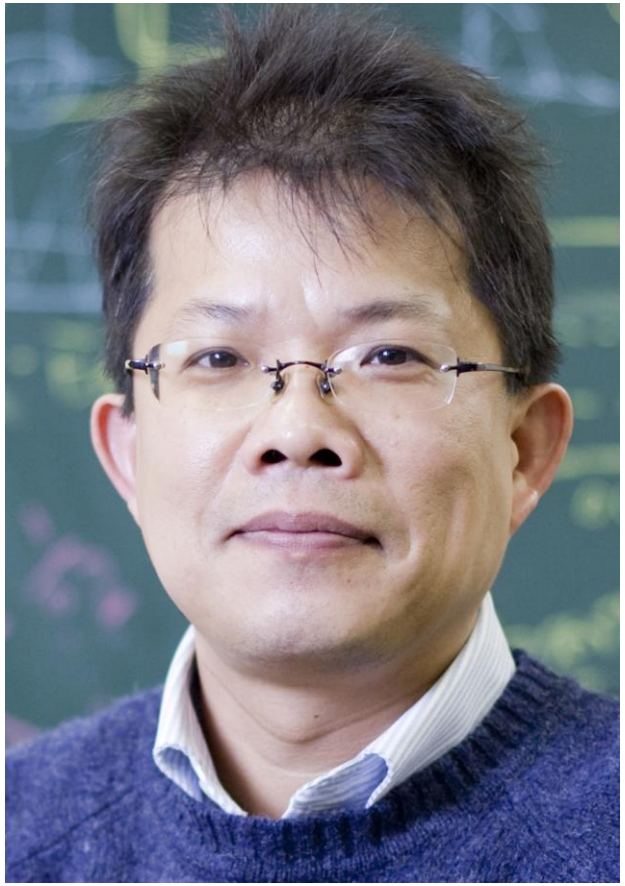
1. Introduction
2. Why are these classes needed?
3. Classes content
4. Conclusion

Outline

1. Introduction
2. Why are these classes needed?
3. Classes content
4. Conclusion

1. Introduction (1/9) Authors

Cong-Kha Pham
phamck@uec.ac.jp



Trong-Thuc Hoang
hoangtt@uec.ac.jp

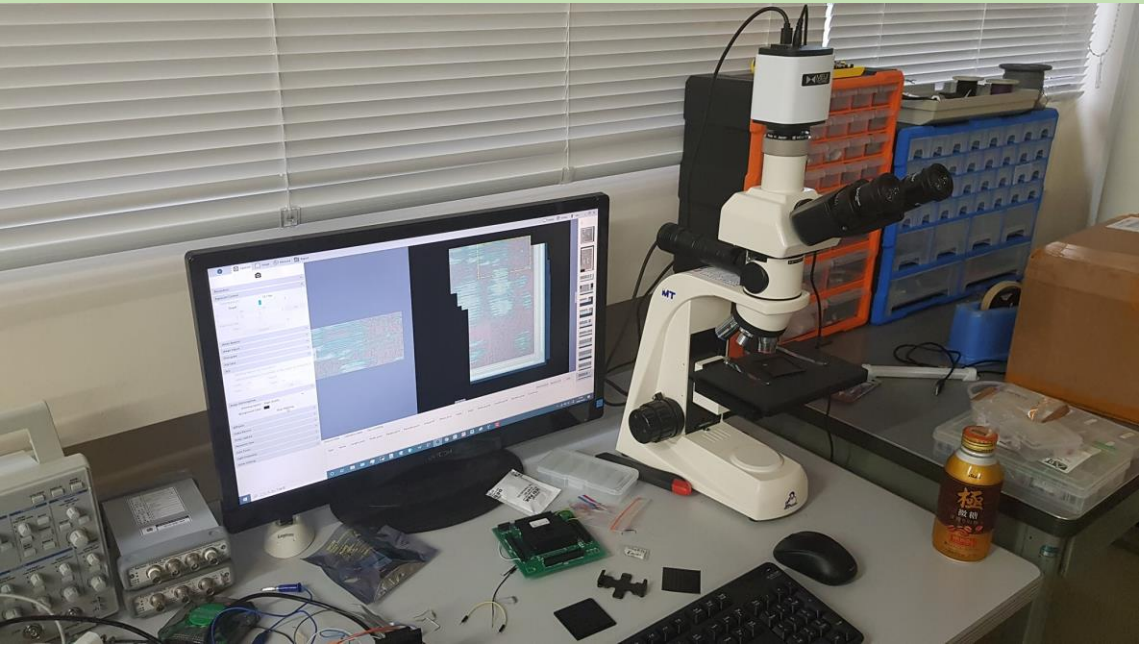
https://thuchoang90.github.io/

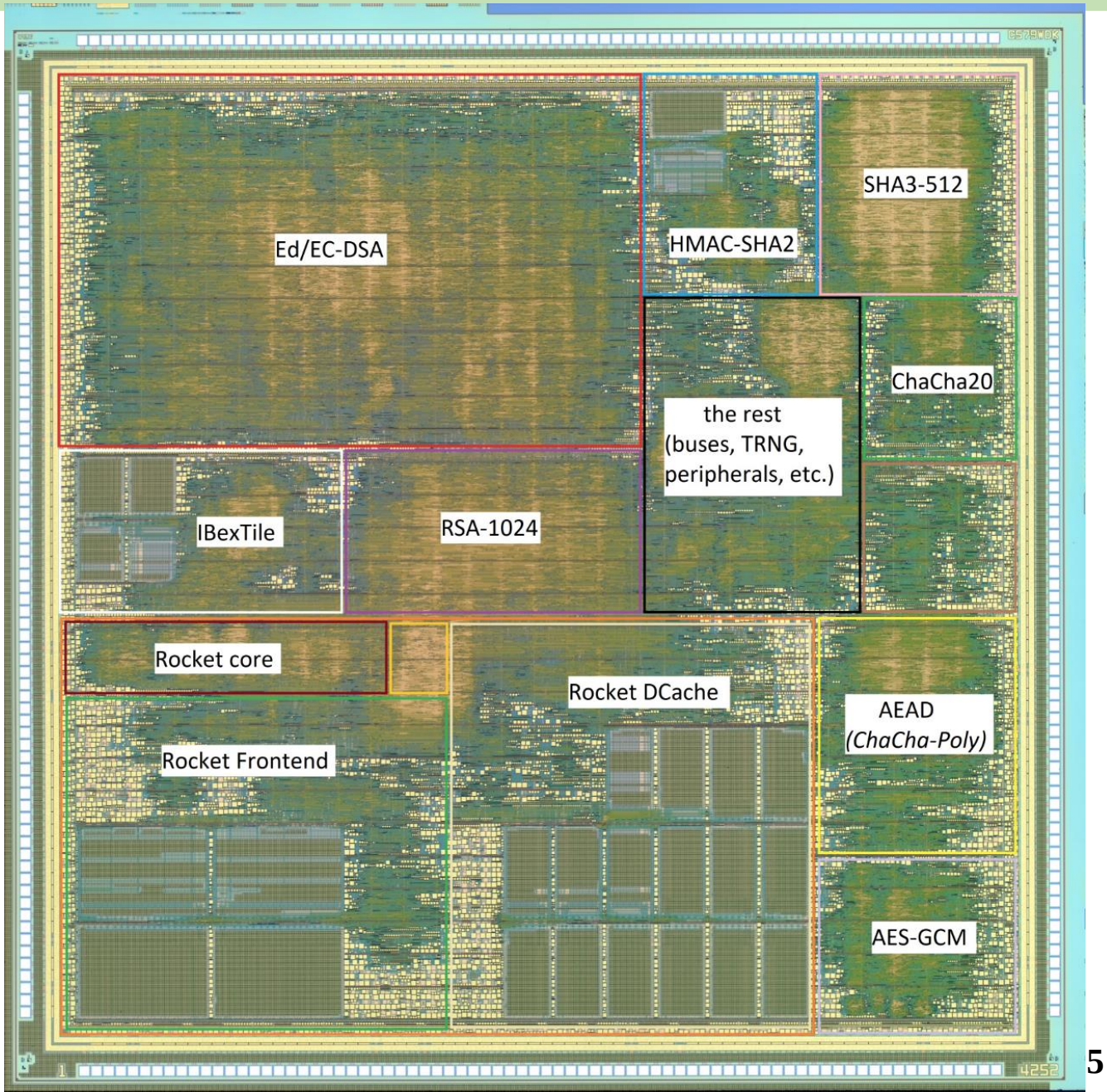
(you can find tutorials and project sources on the website)

http://vlsilab.ee.uec.ac.jp/

(VLSI lab website)

1. Introduction (2/9) Laboratory

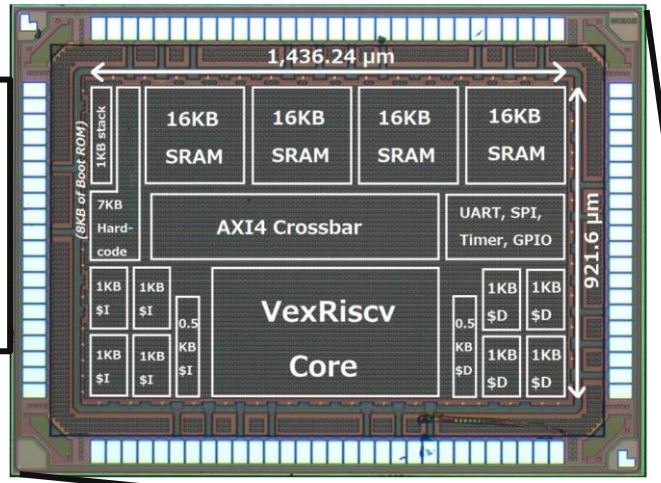




1. Introduction (3/9) Crypto-SoC series

Crypto-SoC

VexRiscv32(1)
SOTB65nm
2x1.5-mm2



2018

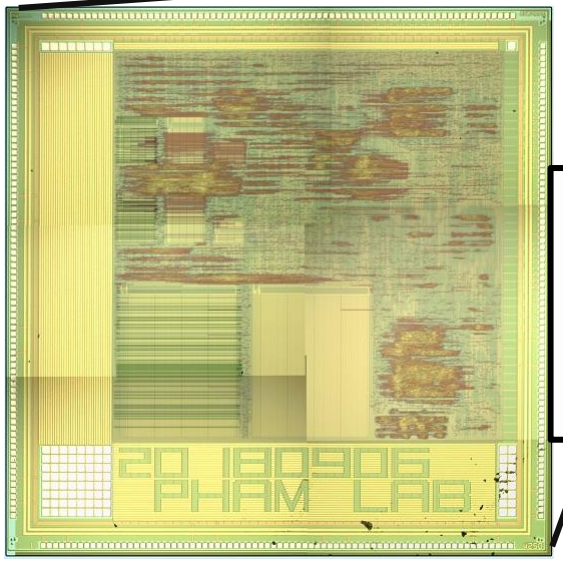
09

2019

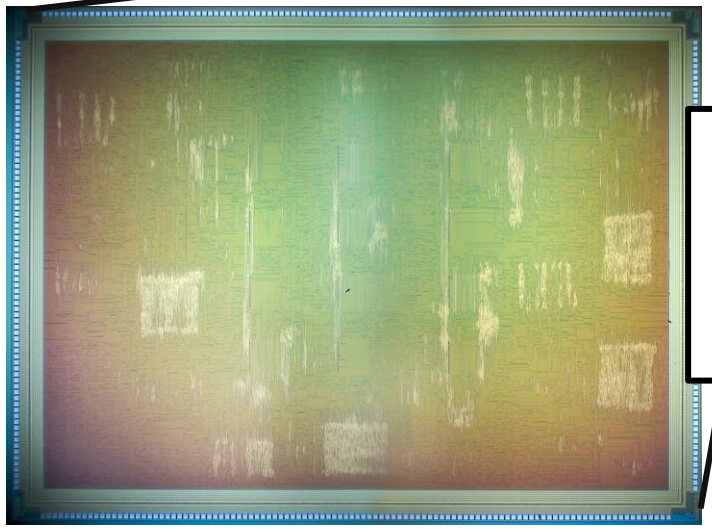
08

10

2020



Rocket64(1)
ROHM180nm
5x5-mm2

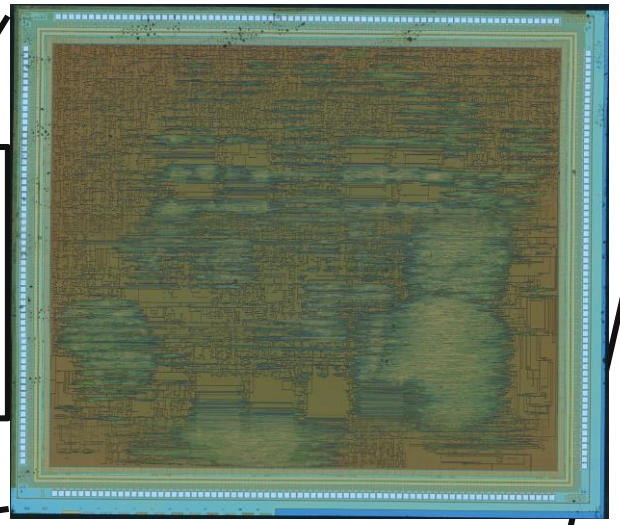


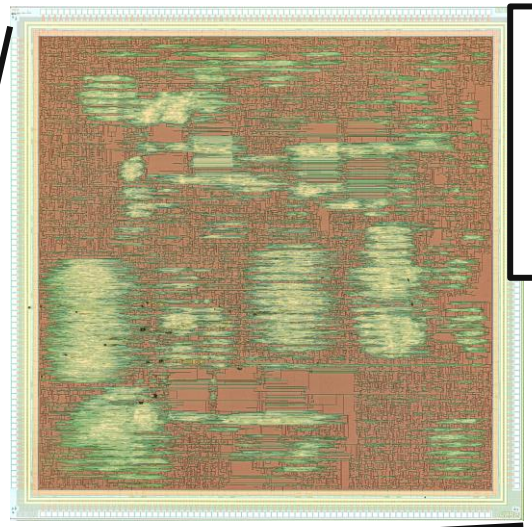
Rocket64(4)
ROHM180nm
5x7.5-mm2

1. Introduction (4/9) Crypto-SoC series

Crypto-SoC

Rocket64(2) + Crypto-cores ROHM180nm 5x5-mm2





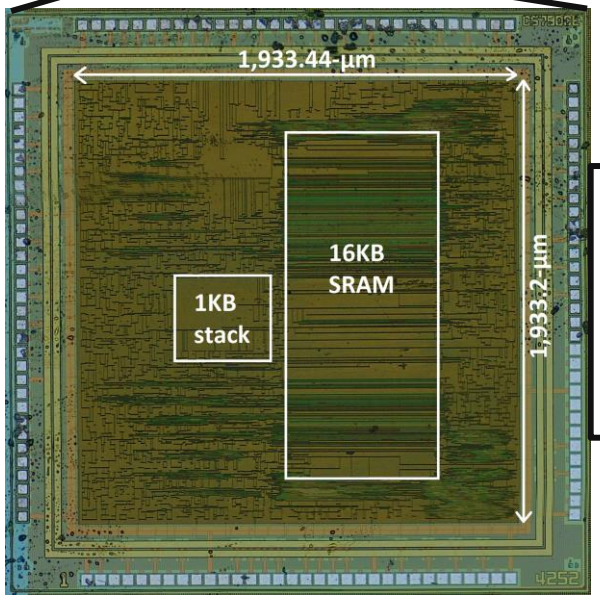
Boom64(1) + Crypto-cores ROHM180nm 5x5-mm2

2020

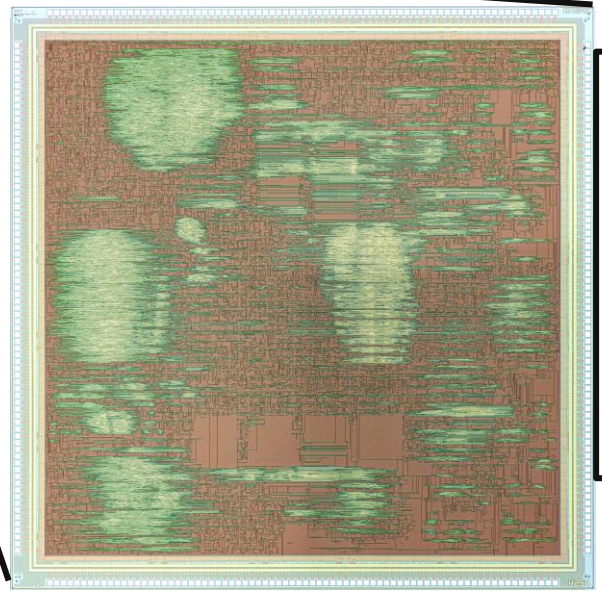
01

06

2021



VexRiscv32(1) ROHM180nm 2.5x2.5-mm2



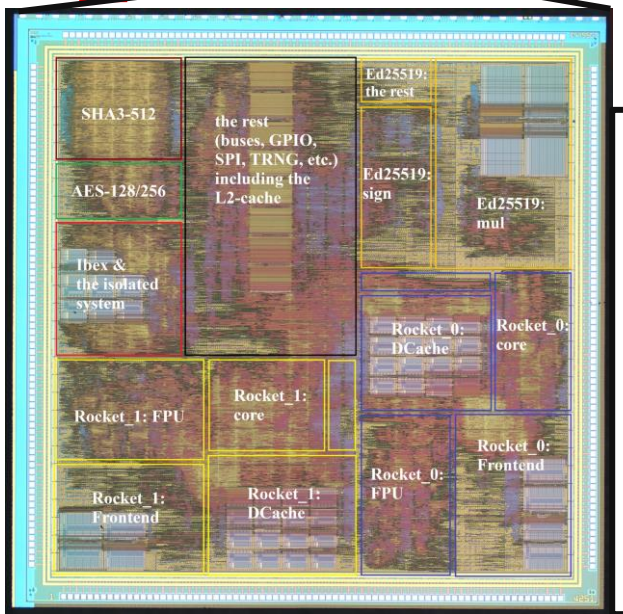
Rocket32(1) + Boom32(1) + Crypto-cores ROHM180nm 5x5-mm2

1. Introduction (5/9) Crypto-SoC series

Crypto-SoC

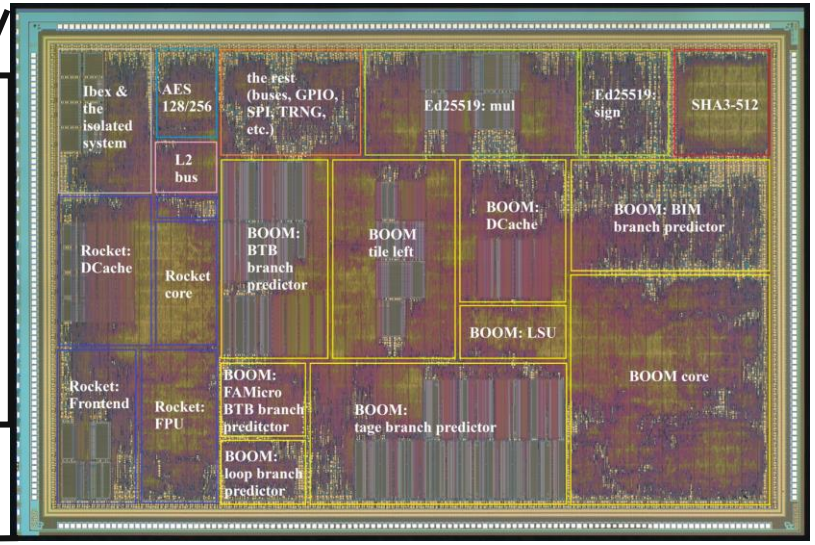
2021

02



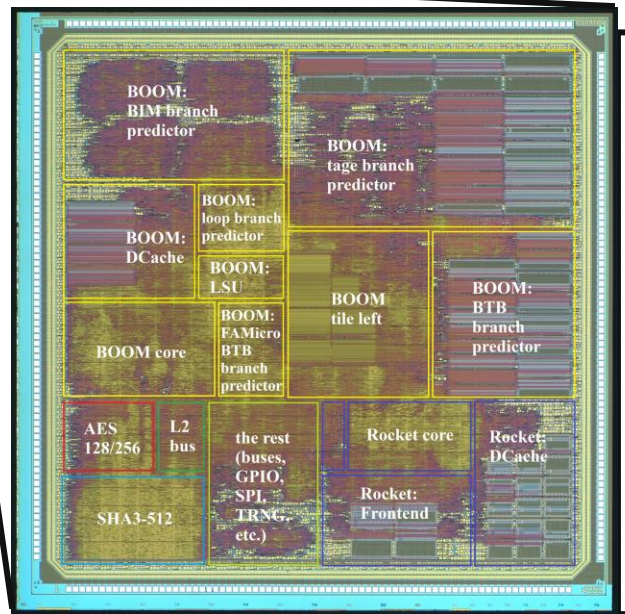
Rocket64(2)
+ Crypto-cores
+ TRNG
+ Secure boot
ROHM180nm
5x5-mm2

Rocket64(1) + Boom64(1)
+ Crypto-cores & TRNG
+ Secure boot
ROHM180nm : 5x7.5-mm2



2022

06



Rocket32(1)
+ Boom32(1)
+ Crypto-cores
+ TRNG
+ Secure boot
ROHM180nm
5x5-mm2

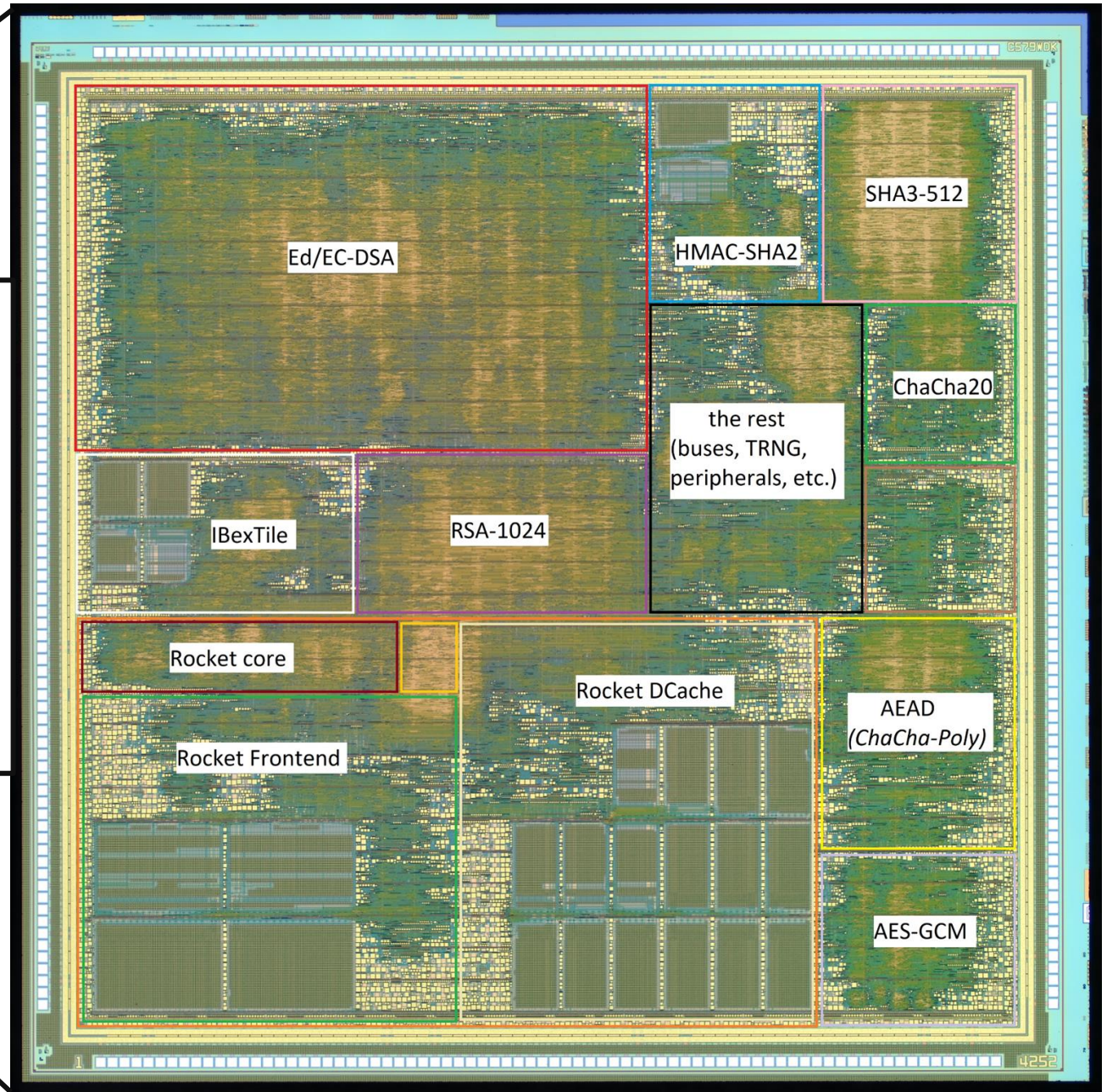
1. Introduction (6/9) Crypto-SoC series

Crypto-SoC

2022

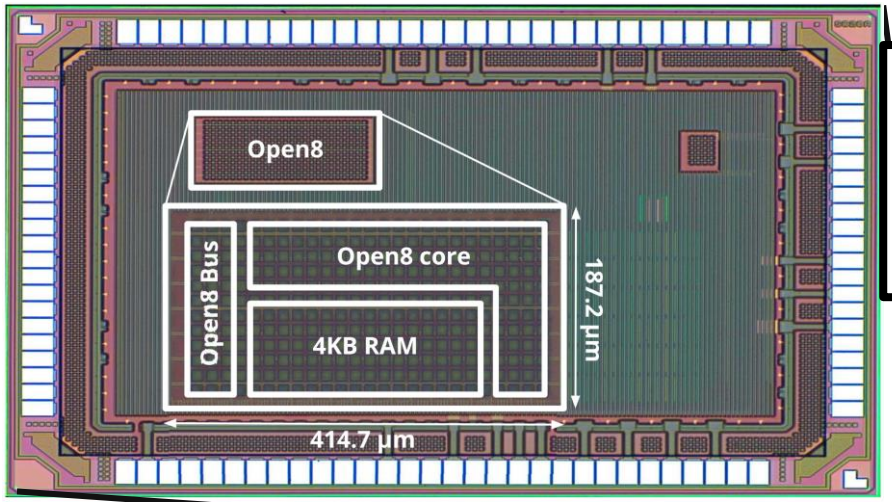
02

Rocket32(1)
+ TLS-1.3
Crypto-cores
+ TRNG
+ Secure boot
ROHM180nm
5x5-mm2



1. Introduction (7/9) Ultra-Low-Power (ULP) SoC

ULP-SoC



Open8 8-bit MCU
SOTB-65nm
1.5x2.0-mm2

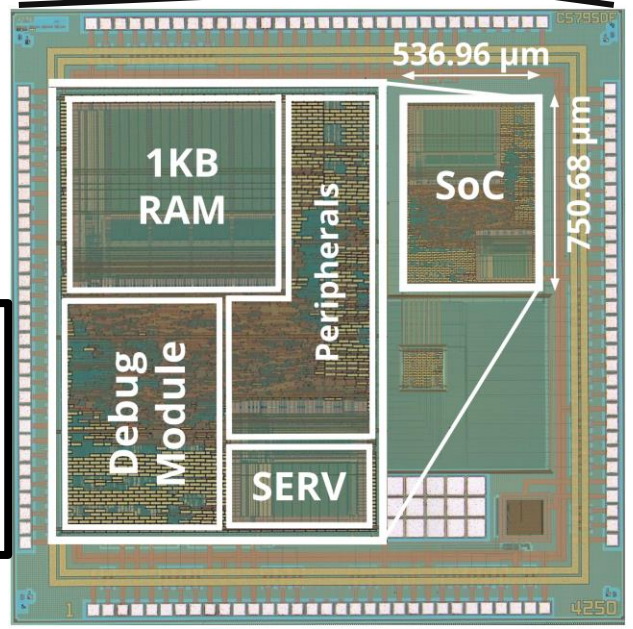
2021

07

10

2022

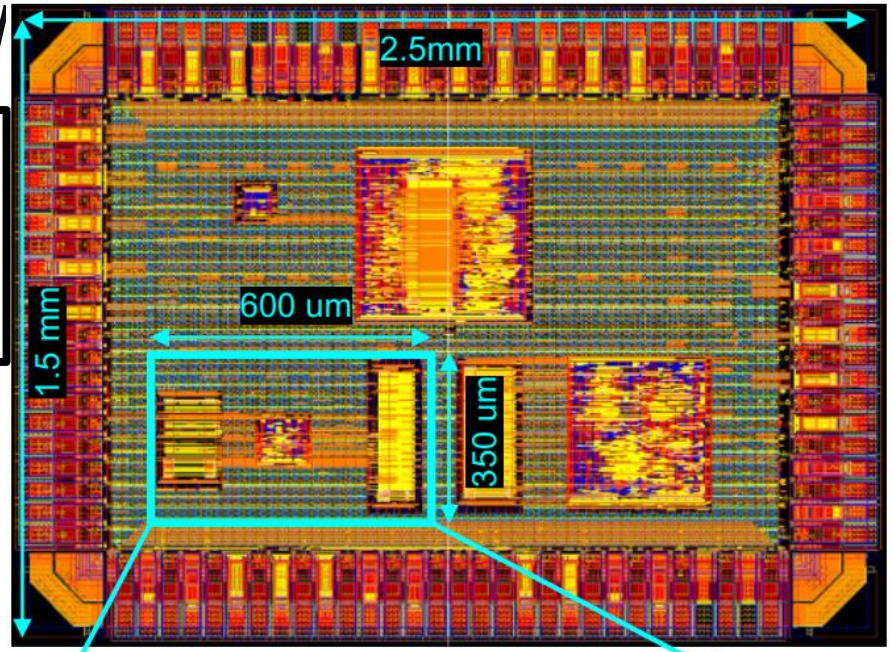
SERV 32-bit MCU
ROHM-180nm
2.5x2.5-mm2



1. Introduction (8/9) Ultra-Low-Power (ULP) SoC

ULP-SoC

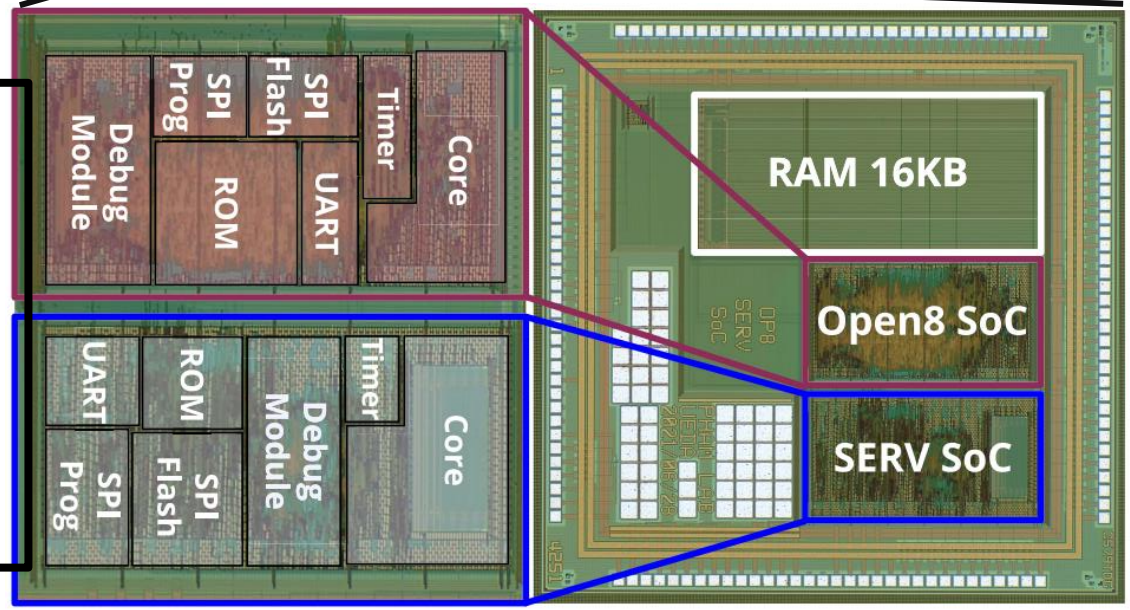
Leros 16-bit MCU
SOTB-65nm
1.5x2.5-mm2

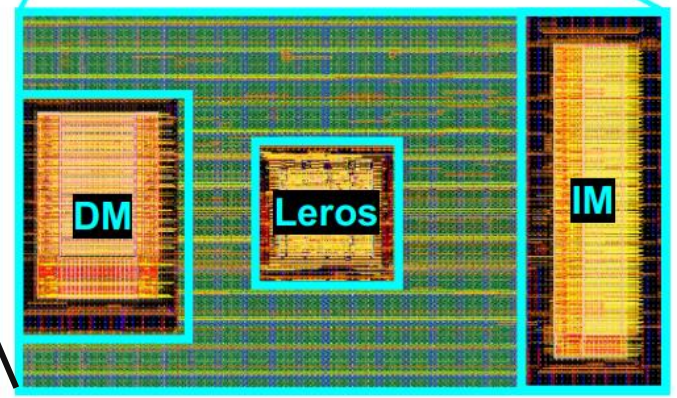


2022

03 **09**

Dual-core
Open8 &
SERV
ROHM-
180nm
2.5x2.5-mm2





1. Introduction (9/9) RISC-V-based SoC

- Since 2019 to now: **4 years**
- A core team of **five members**
- **20+** chips: FullChip designs
- **Linux** bootable SoC
- Two big projects: **Crypto-SoC** and **ULP-SoC** *(and other small projects)*
- All these achievements are thanks to **RISC-V**.
- **14** Journals and **10** Conferences

Our goal is to bring **Computer Design classes** based on RISC-V architecture.
Participants can make their own SoC: *explore ideas → develop → implement → evaluate*

Outline

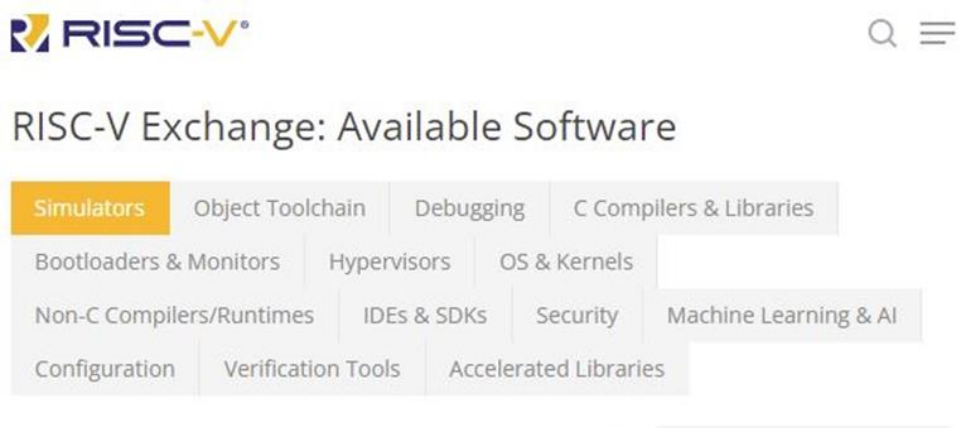
1. Introduction
2. Why are these classes needed?
3. Classes content
4. Conclusion

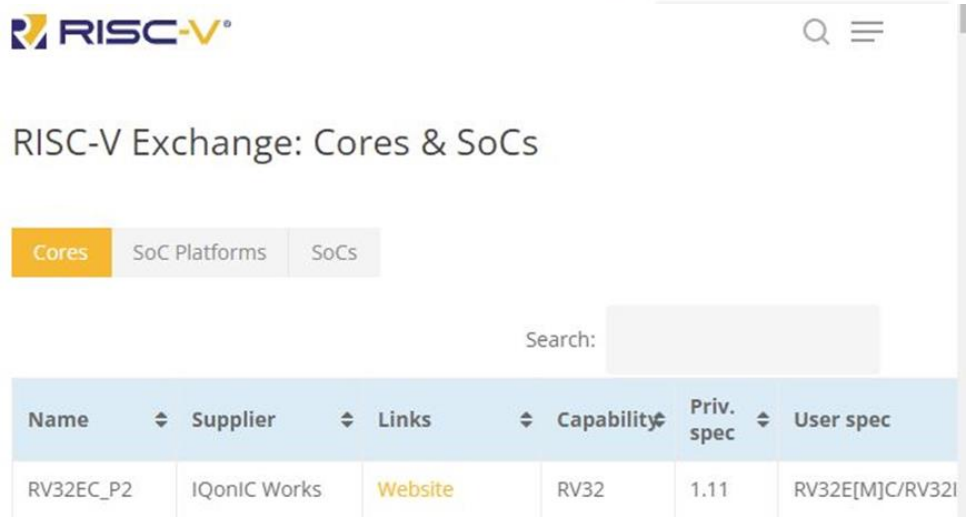
2. Classes necessity (1/18) What is RISC-V

Open-source **RISC-V** means open-source **ISA**, no more, no less.

(ISA: Instruction Set Architecture)

(some other common ISAs: i386, amd64, ARM 32/64, AVR, MIPS, NiosII, etc.)





RISC-V Foundation: https://riscv.org/

- Official released ISA specification
- Many cores, SoCs, & software are available
- Developers can reuse each other designs & tools
  → significantly reducing R&D time and effort

Licensed free:
- RISC-V ISA
- RISC-V toolchain

License depended on authors/developers:
- RISC-V processors
- RISC-V software applications
- RISC-V-related products

2. Classes necessity (2/18) RISC-V's ISA

What makes **RISC-V** different: its modular mindset

(modular architecture helps fine-tune the performance based on the developer's needs)

Base instruction set: **I**nteger

Extended instruction set: *the rest*

Extension	Description
I	Integer
M	Integer Multiplication and Division
A	Atomics
F	Single-Precision Floating Point
D	Double-Precision Floating Point
G	General Purpose = IMAFD
C	16-bit Compressed Instructions
Non-Standard User-Level Extensions	
Xext	Non-standard extension "ext"

The most common extensions: **IMAFDC** *(also known as **GC**)*

There are also a lot more than just **IMAFDC :**

Base	Version	Status
RVWMO	2.0	**Ratified**
RV32I	**2.1**	**Ratified**
RV64I	**2.1**	**Ratified**
RV32E	*1.9*	*Draft*
RV128I	*1.7*	*Draft*
Extension	**Version**	**Status**
M	**2.0**	**Ratified**
A	**2.1**	**Ratified**
F	**2.2**	**Ratified**
D	**2.2**	**Ratified**
Q	**2.2**	**Ratified**
C	**2.0**	**Ratified**
Counters	*2.0*	*Draft*
L	*0.0*	*Draft*
B	*0.0*	*Draft*
J	*0.0*	*Draft*
T	*0.0*	*Draft*
P	*0.2*	*Draft*
V	*0.7*	*Draft*
Zicsr	**2.0**	**Ratified**
Zifencei	**2.0**	**Ratified**
Zam	*0.1*	*Draft*
Ztso	*0.1*	*Frozen*

2. Classes necessity (3/18) RISC-V's OS-stack

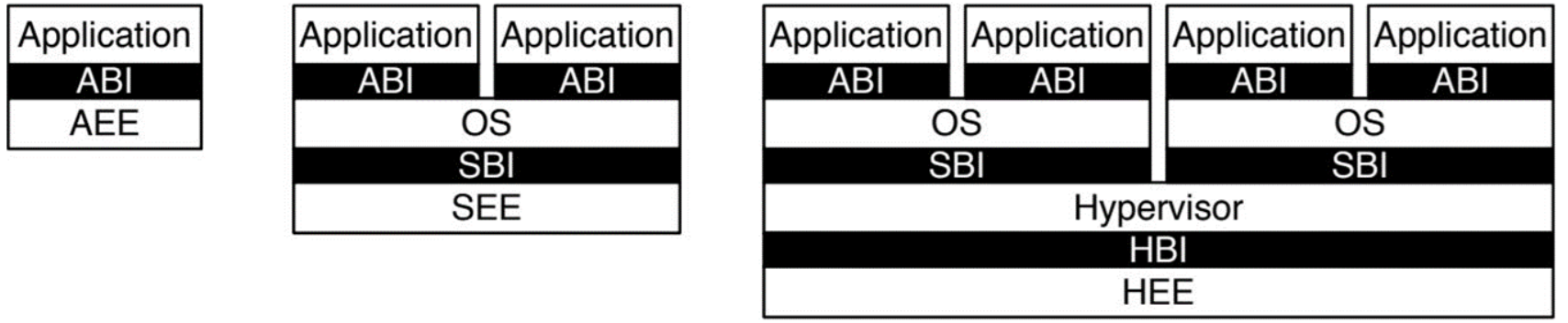
To support an Operating System (OS), the ISA has to support the OS stack *or the **M-/S-/U-mode***.

RISC-V privileged architecture:

RISC-V Modes		
Level	Name	Abbr.
0	User/Application	U
1	Supervisor	S
	Reserved	
3	Machine	M

Supported Combinations of Modes	
Supported Levels	Modes
1	M
2	M, U
3	M, S, U

Different scenarios of utilizing the OS stack:

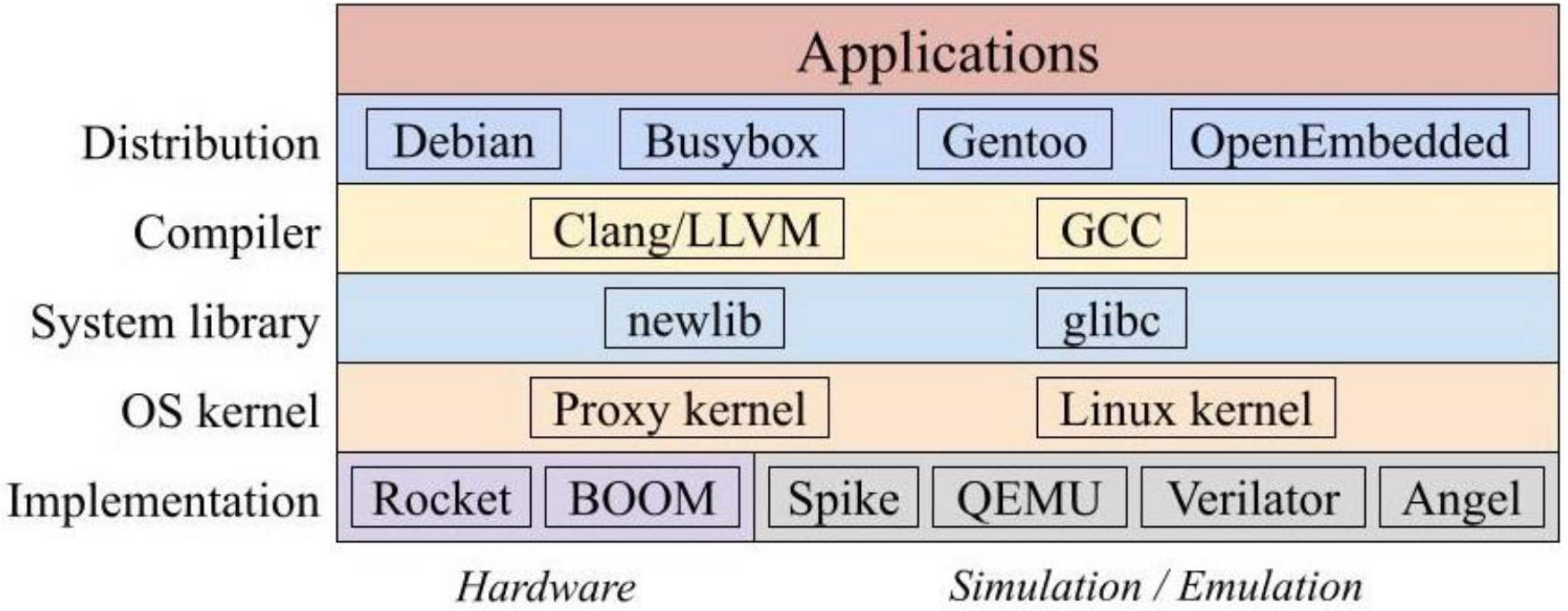


RISC-V ISA not only supports the OS stack, but also provides a **privileged architecture**.

→ Better security scheme by having the hardware recognize different codes executed at different modes.

2. Classes necessity (4/18) RISC-V's ecosystem

RISC-V toolchain and its ecosystem



(User's) software in the top

OS file system

Compiler

Standard libraries

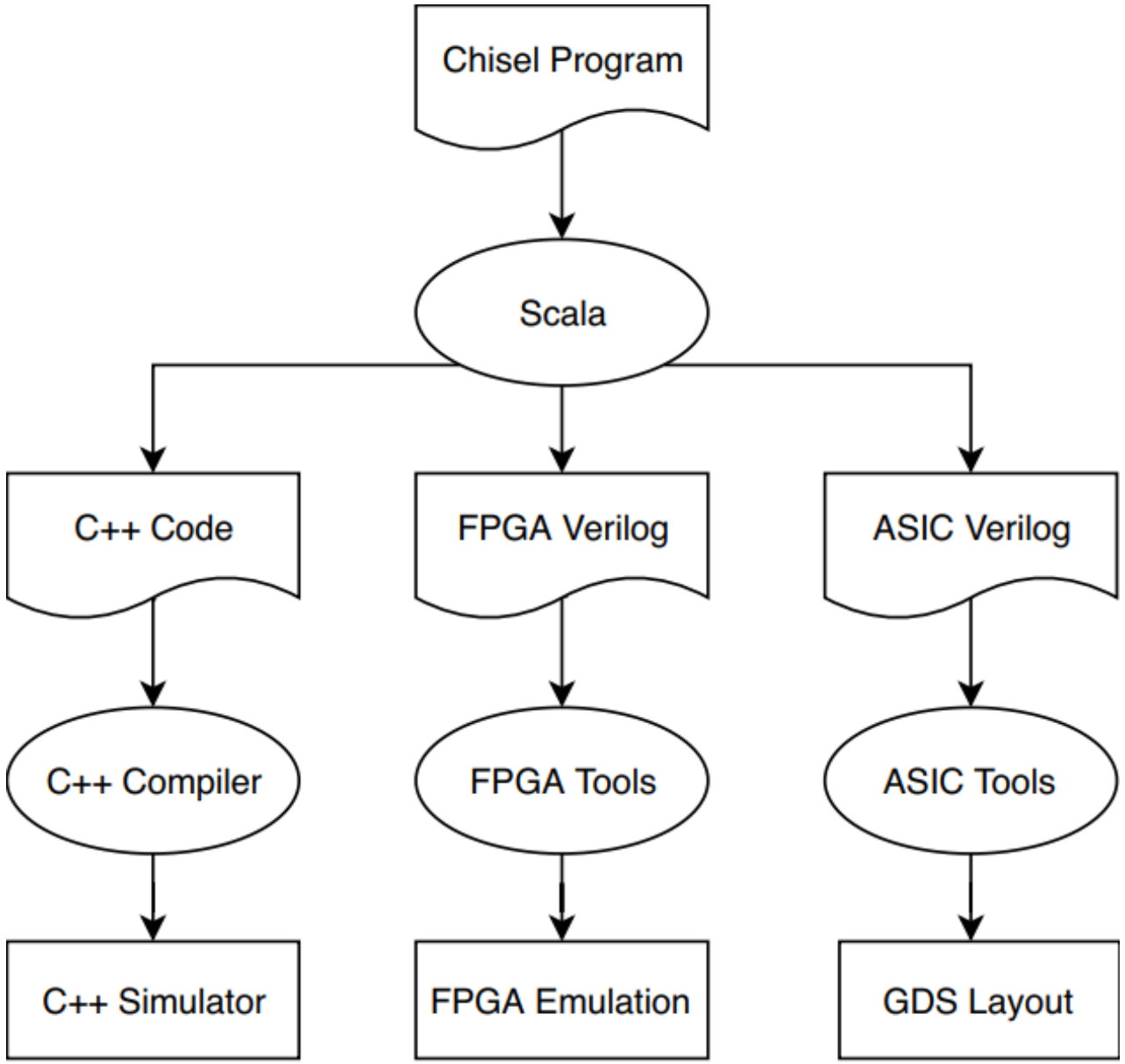
OS kernel

Hardware in the bottom

Top-down explanation:

User's applications on the top are operated in an OS file system, which then compiled by a compiler based on multiple standard libraries. After compiled, the execution file is run on the OS kernel that manages the hardware in the bottom.

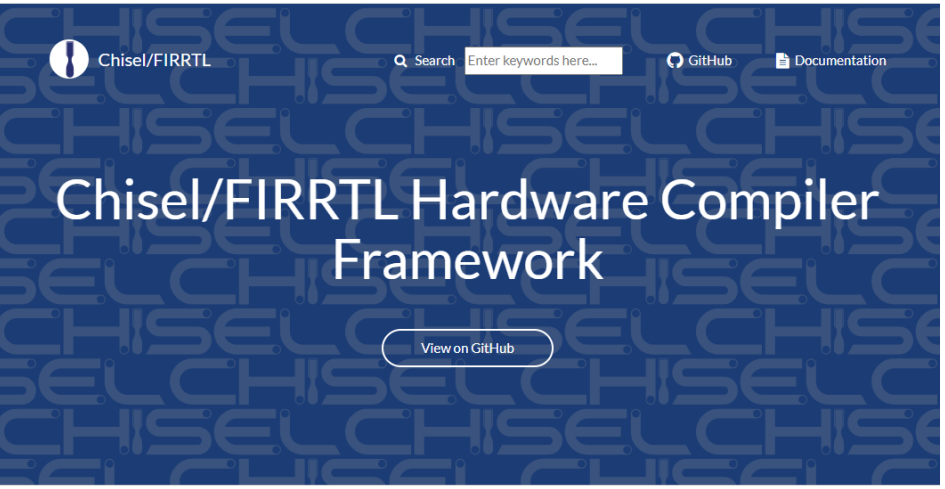
2. Classes necessity (5/18) CHISEL: a new way to code



Chisel is a library.
Scala is a language.

- **Scala** itself is a *high-level object-oriented programming language*
  → It is not designed for "hardware coding."
- **Chisel** is a library attached to Scala to define a set of coding rules.
  → It is designed for "hardware coding."
- From **Scala** to **Verilog**:
  Scala → Java → FIRRTL → Verilog
  1st arrow: Scala compiler named SBT
  2nd arrow: executing Java
  3rd arrow: FIRRTL compiler

2. Classes necessity (6/18) CHISEL: a new way to code

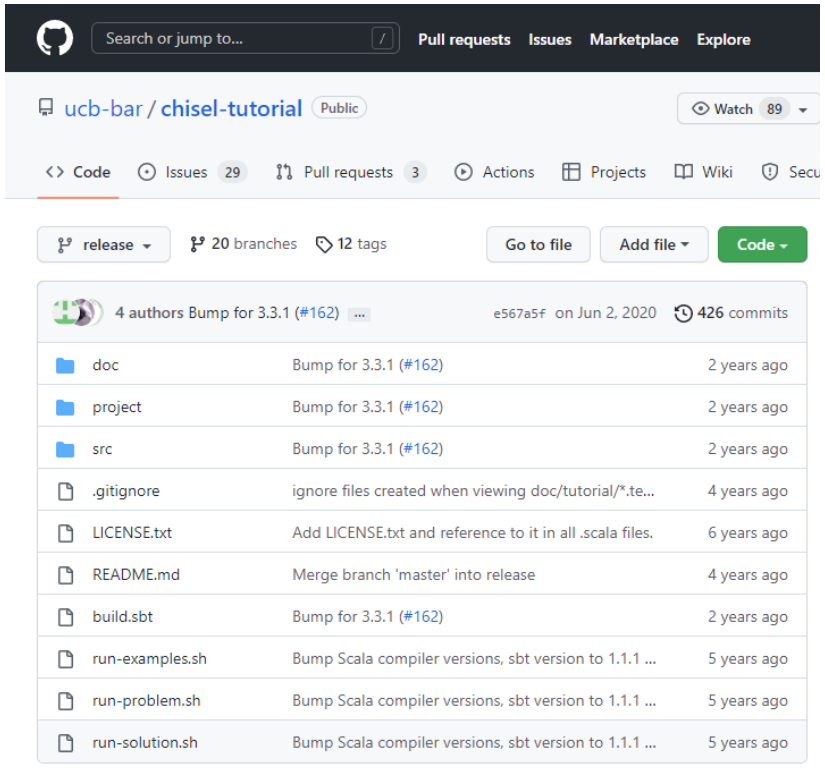


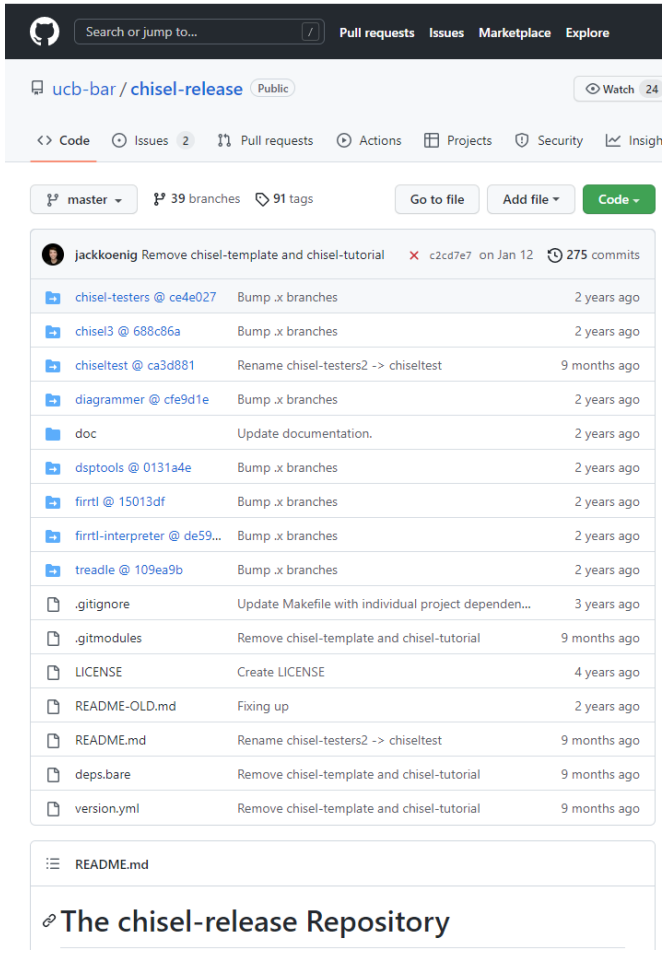
CHISEL

CCC 2022

The official CHISEL website:
https://www.chisel-lang.org/

CHISEL tutorials can be found at:
https://github.com/ucb-bar/chisel-tutorial





Released CHISEL sources:
https://github.com/ucb-bar/chisel-release

2. Classes necessity (7/18) Open library

The common open RISC-V libraries that you can use

Chipyard *(contains many common and frequently used open IPs, including RISC-V processors and other peripherals such as uart, spi, sd-card, etc.)*: https://github.com/ucb-bar/chipyard

README.md

CHIPYARD

Chipyard Framework

chipyard-ci-process passing

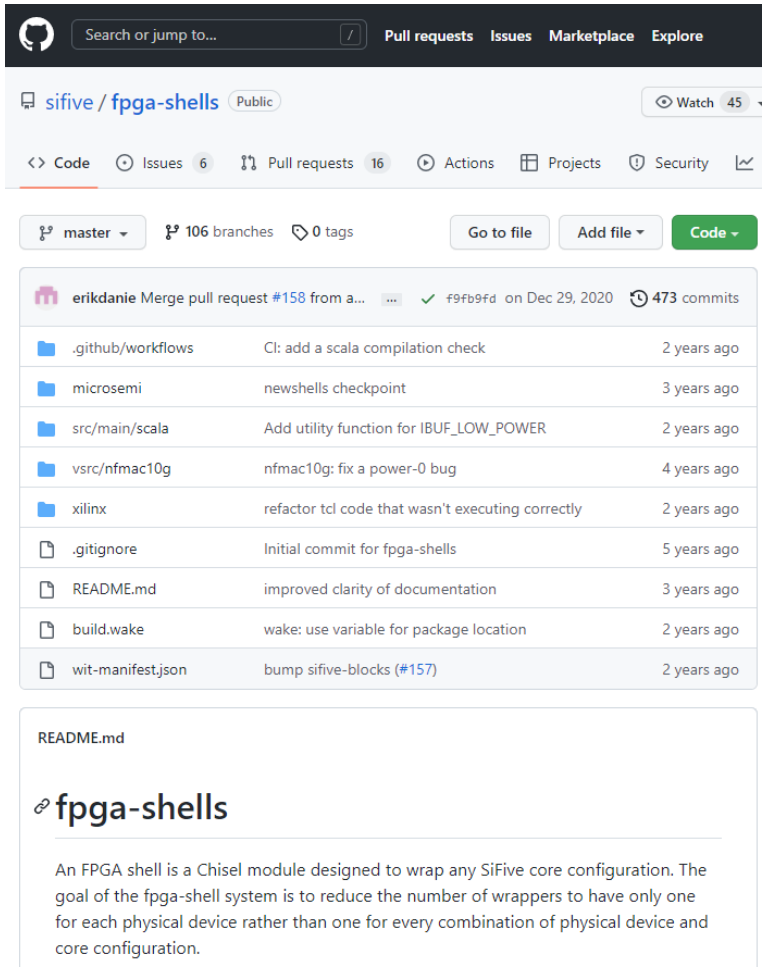
Quick Links

- **Stable Documentation:** https://chipyard.readthedocs.io/
- **User Question Forum:** https://groups.google.com/forum/#!forum/chipyard
- **Bugs and Feature Requests:** https://github.com/ucb-bar/chipyard/issues

Using Chipyard

To get started using Chipyard, see the stable documentation on the Chipyard documentation site: https://chipyard.readthedocs.io/

What is Chipyard



fpga-shells *(contains many common FPGA configurations)*: https://github.com/sifive/fpga-shells

2. Classes necessity (8/18) Open processors

Some famous RISC-V processors

Rocket is the most popular among RISC-V processors: https://github.com/chipsalliance/rocket-chip
(it is an in-of-order processor)

README.md

Rocket Chip Generator

Continuous Integration passing

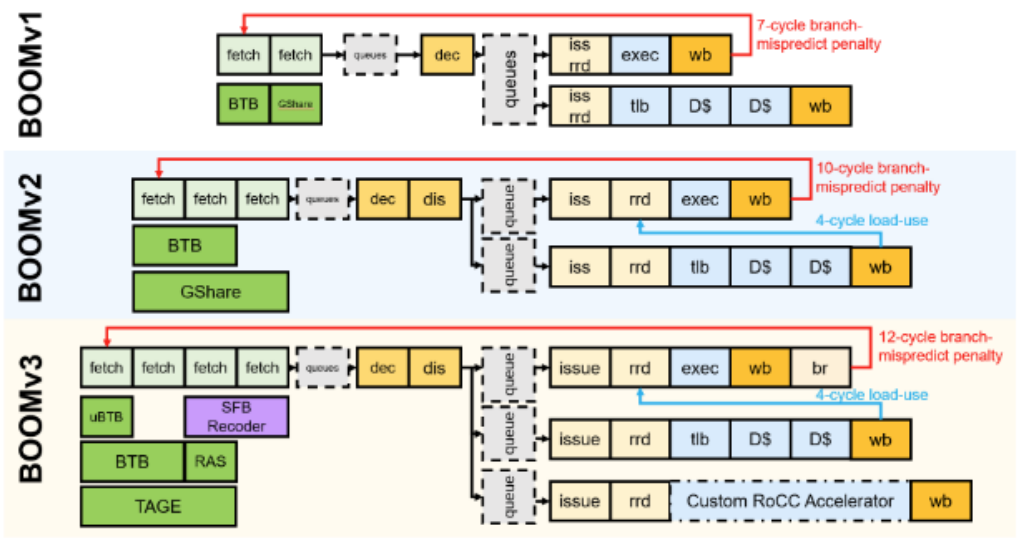
This repository contains the Rocket chip generator necessary to instantiate the RISC-V Rocket Core. For more information on Rocket Chip, please consult our technical report.

Table of Contents

- Quick instructions for those who want to dive directly into the details without knowing exactly what's in the repository.
- What's in the Rocket chip generator repository?
- How should I use the Rocket chip generator?
  - Using the cycle-accurate Verilator simulation
  - Mapping a Rocket core down to an FPGA
  - Pushing a Rocket core through the VLSI tools
- How can I parameterize my Rocket chip?
- Debugging with GDB
- Building Rocket Chip with an IDE
- Contributors

BOOM is an out-of-order processor that can rival ARM: https://github.com/riscv-boom/riscv-boom

README.md



The Berkeley Out-of-Order RISC-V Processor

2. Classes necessity (9/18) Open processors

Some famous RISC-V processors

README.md

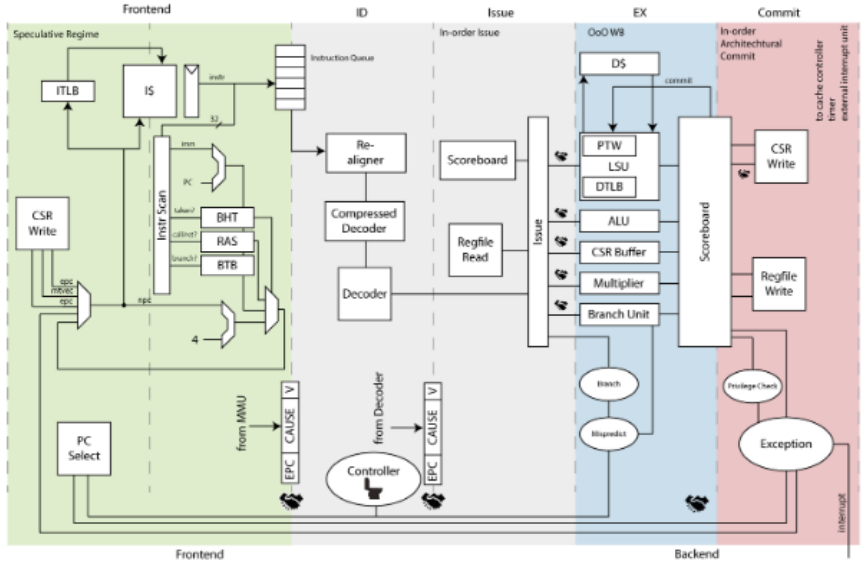
Ariane is a high-performance 64-bit in-of-order processor:
https://github.com/lowRISC/ariane

build unknown

Ariane RISC-V CPU

Ariane is a 6-stage, single issue, in-order CPU which implements the 64-bit RISC-V instruction set. It fully implements I, M, A and C extensions as specified in Volume I: User-Level ISA V 2.3 as well as the draft privilege extension 1.10. It implements three privilege levels M, S, U to fully support a Unix-like operating system. Furthermore it is compliant to the draft external debug spec 0.13.

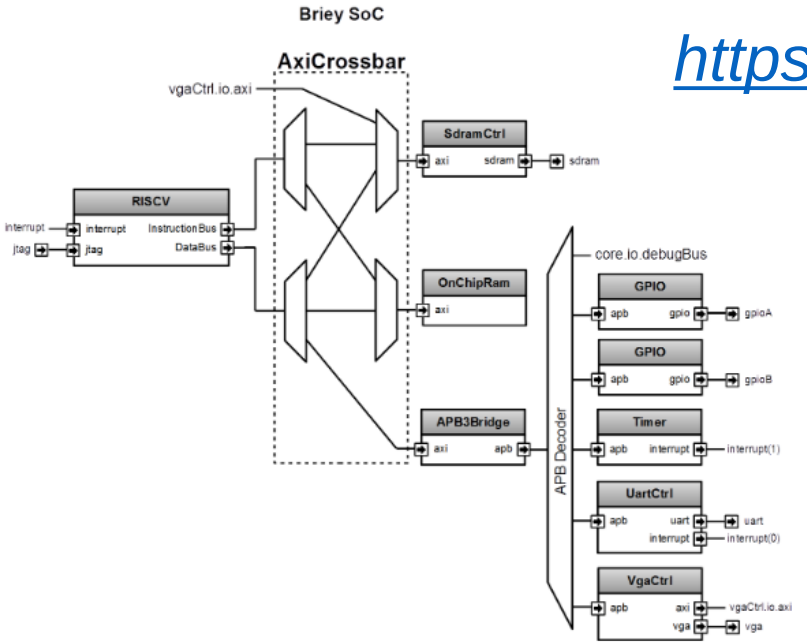
It has configurable size, separate TLBs, a hardware PTW and branch-prediction (branch target buffer and branch history table). The primary design goal was on reducing critical path length.



VexRiscv is a simple in-of-order 32-bit processor that is suited for an MCU:

Briey SoC

As a demonstration, a SoC named Briey is implemented in `src/main/scala/vexriscv/demo/Briey.scala`. This SoC is very similar to the Pinsec SoC:



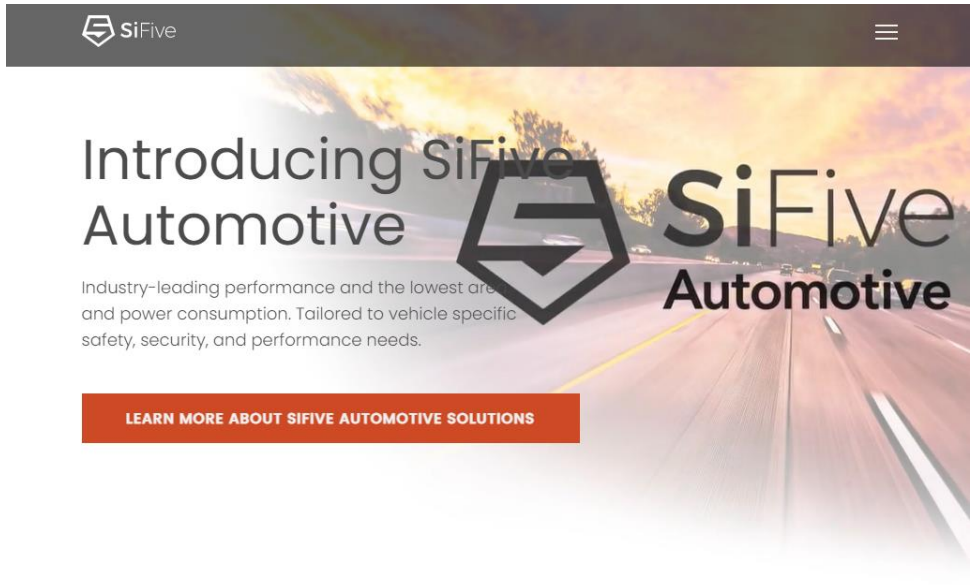
https://github.com/SpinalHDL/VexRiscv

2. Classes necessity (10/18) RISC-V groups

Some famous RISC-V groups

SiFive is the first and the most famous company that is doing RISC-V-related products:

- Their website: https://www.sifive.com/
- Their github: https://github.com/sifive



HexFive is a company that focuses on RISC-V-based security applications:

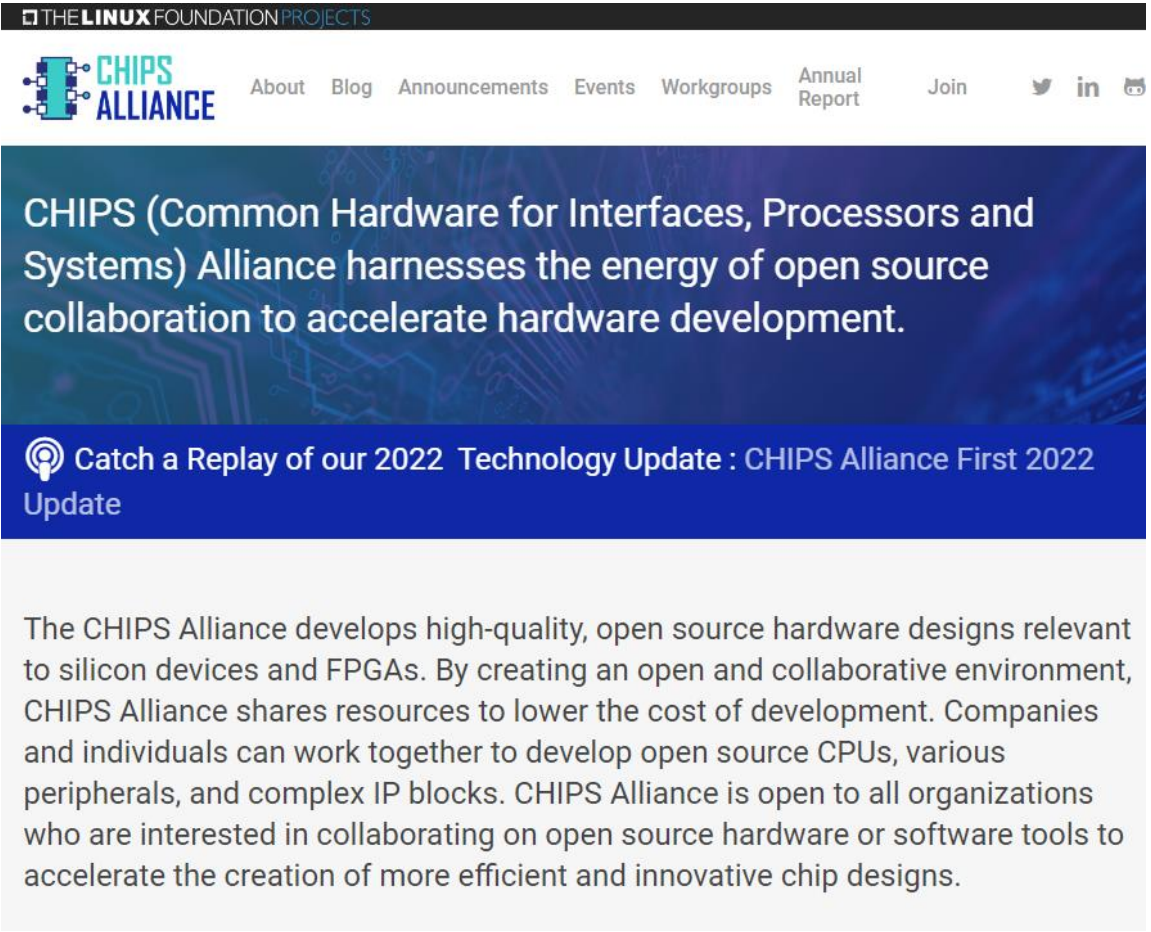
- Their website: https://hex-five.com/
- Their github: https://github.com/hex-five

2. Classes necessity (11/18) RISC-V groups

Some famous RISC-V groups

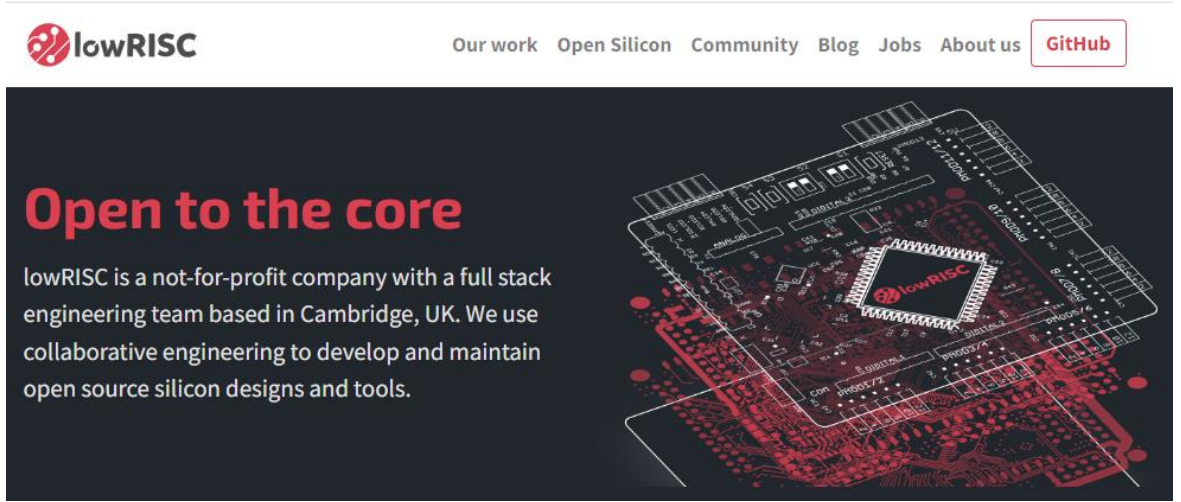
Chips Alliance is an organization that maintains many high-quality open-source IPs used in RISC-V systems:

- Their website: https://chipsalliance.org/
- Their github: https://github.com/chipsalliance



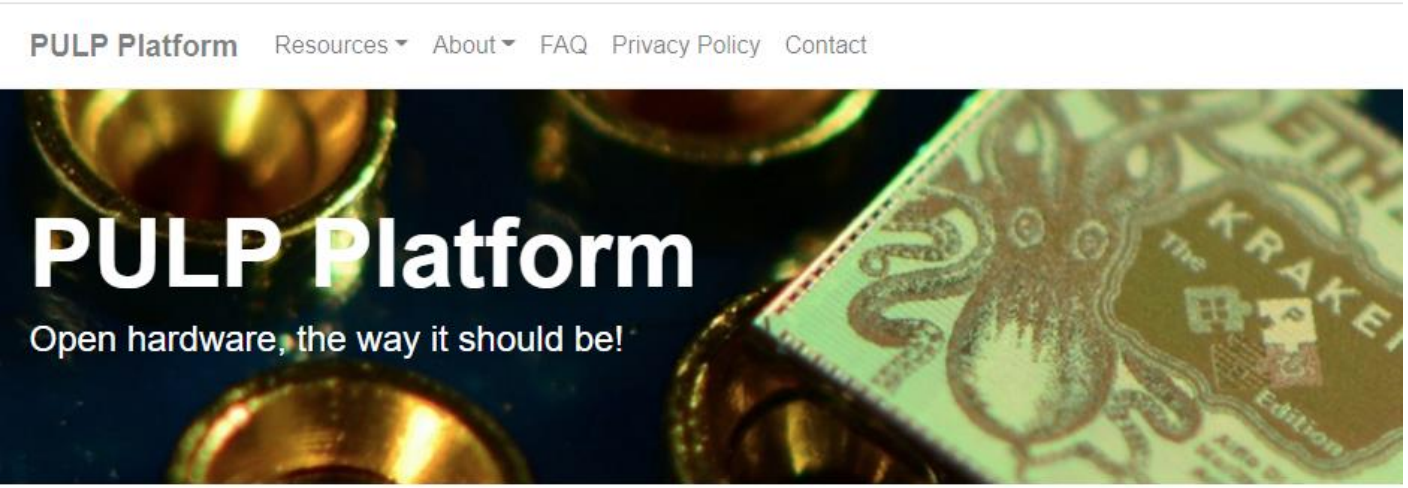
lowRISC is a non-profit company that has published many low-power and high-security IPs for RISC-V systems:

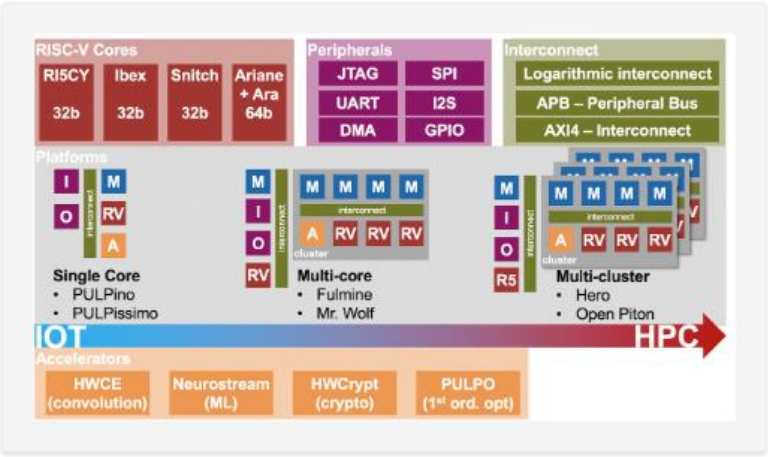
- Their website: https://lowrisc.org/
- Their github: https://github.com/lowRISC

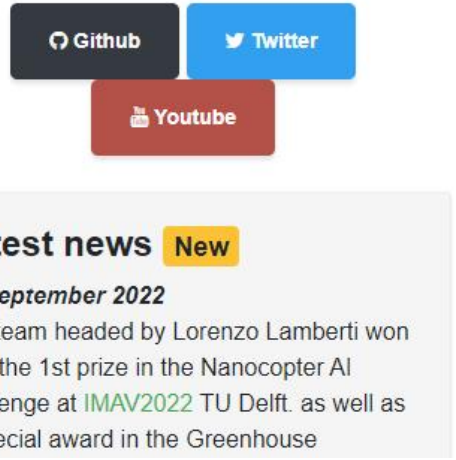


2. Classes necessity (12/18) RISC-V groups

Some famous RISC-V groups





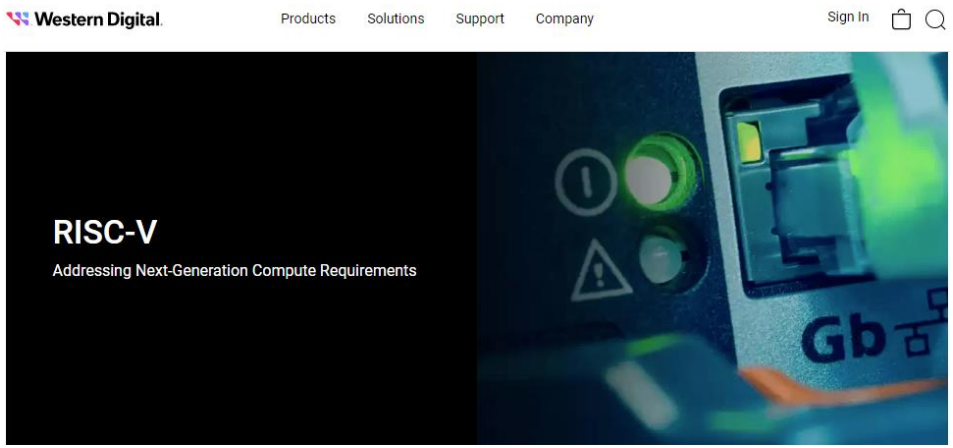


PULP is an open group started from a RISC-V project published by two universities.
It has many open-source processors and fabricated many chips:

- Their website: https://pulp-platform.org/
- Their github: https://github.com/pulp-platform

2. Classes necessity (13/18) RISC-V products

Some RISC-V-related products



Unleashing the power of data through RISC-V initiatives

The open source model, proven by the success of Linux®, now has a hardware platform in RISC-V to enable the next generation of innovation. Another reason is that the configurability RISC-V provides is unparalleled. By leveraging the open collaboration and flexibility of RISC-V, Western Digital can create processors that are purpose-built for data-centric applications.

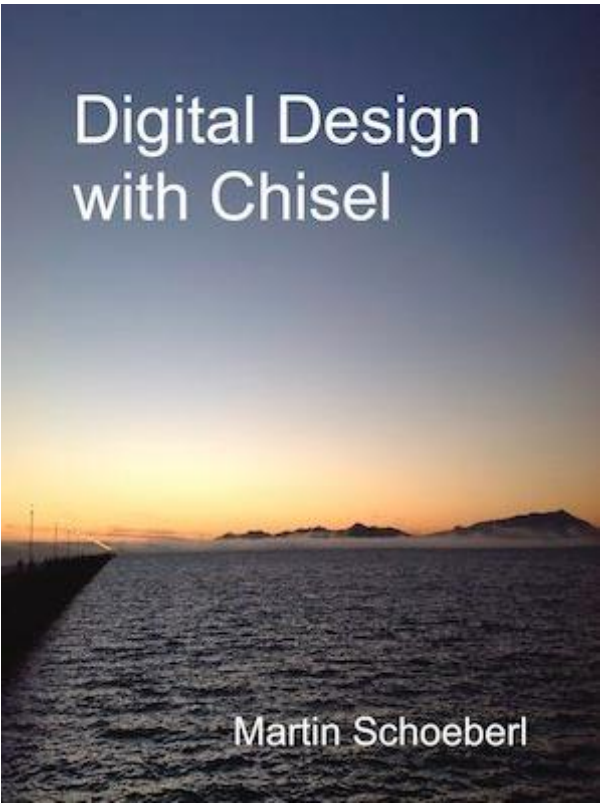
Promising upcoming products in **Western Digital** and **Seagate**

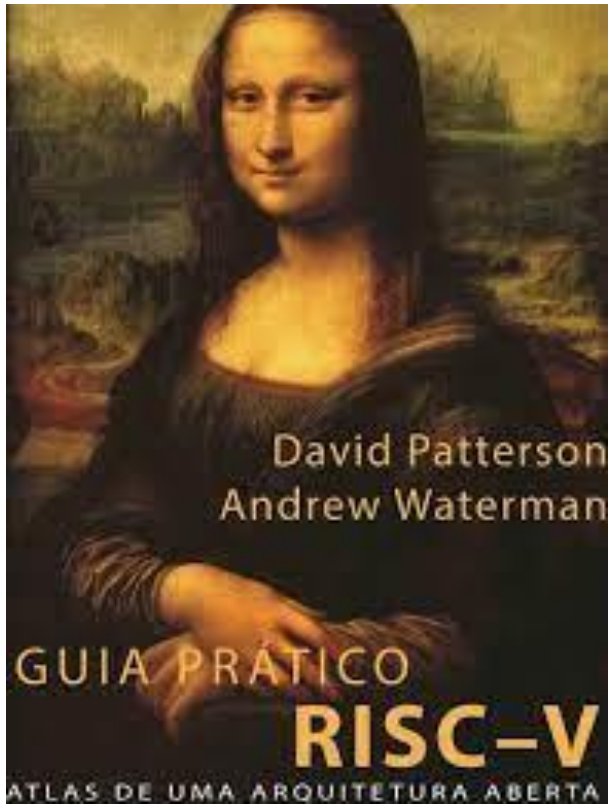
Read more:

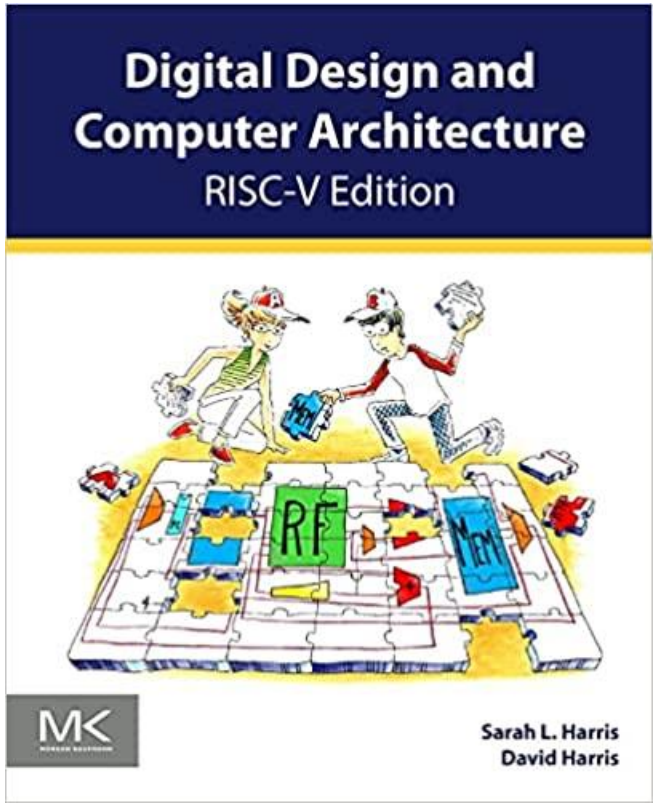
- At RISC-V Foundation website: https://riscv.org/news/2020/12/seagate-western-digital-outline-progress-on-risc-v-designs-carol-sliwa-searchstorage-techtarget-com/
- At Seagate website: https://www.seagate.com/jp/ja/innovation/risc-v/
- At WD website: https://www.westerndigital.com/company/innovation/open-source/risc-v

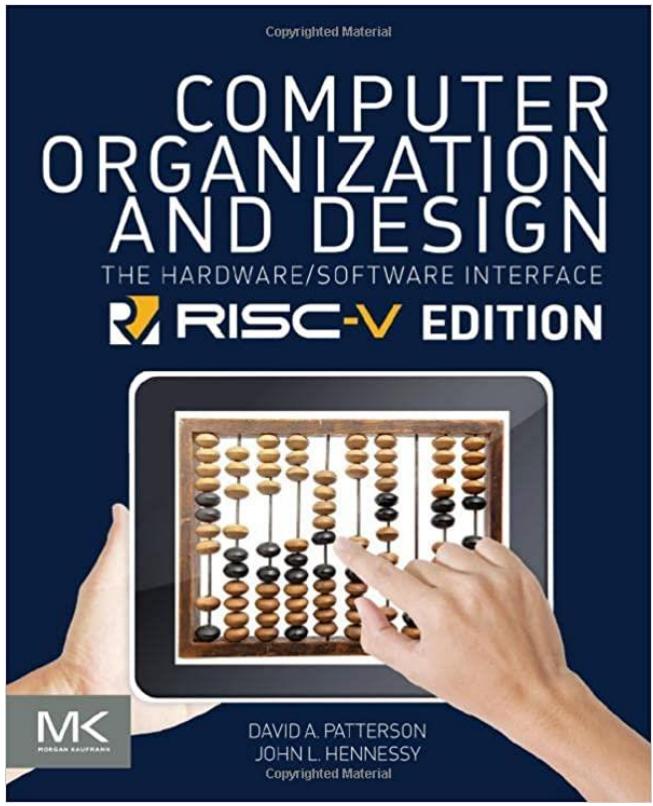
2. Classes necessity (14/18) RISC-V books

Several books for RISC-V learners









2. Classes necessity (15/18) RISC-V keys contribution

RISC-V revolutionizes Computer System Design

1. **Modular at heart:**
   customizable ISA and customizable hardware
   → fine-tune the system to your specific needs.

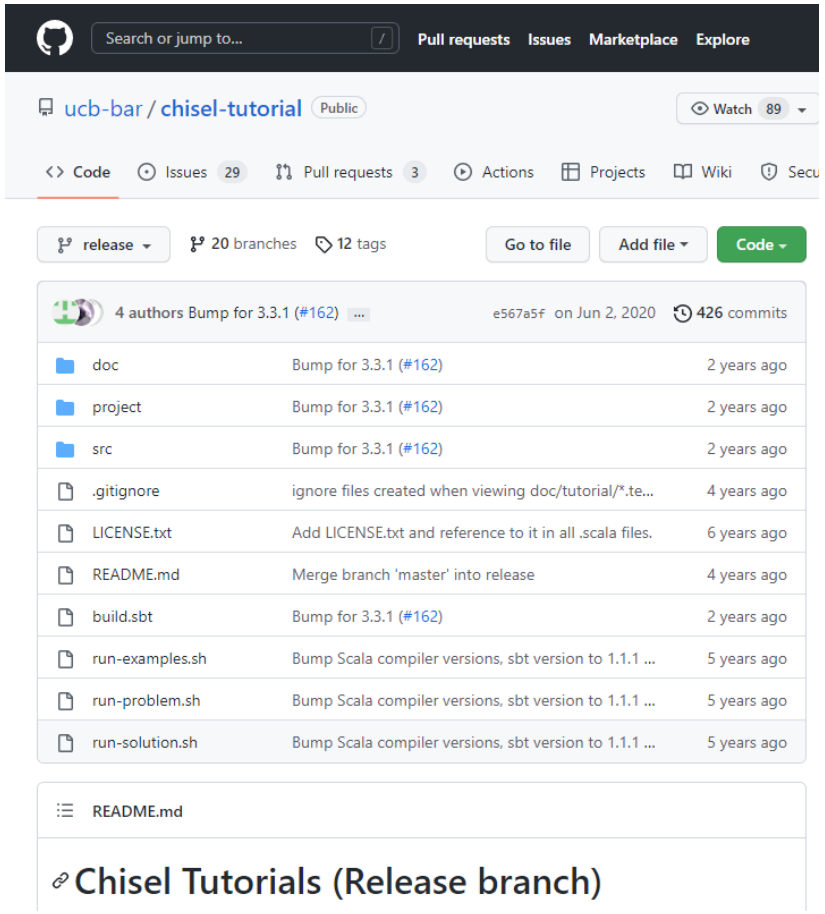
2. **Open-source community:**
   license-free ISA, open cores and SoCs, open-source libraries, open-source software, etc.
   → reuse other developers' designs → save time and effort for R&D

3. **CHISEL** ***(Constructing Hardware In Scala Embedded Language)*****:**
   a new way to "coding" hardware circuits. When compiled, it will generate a true RTL Verilog code.
   → a "meta-programming" language for hardware developers with parameters and sub-designs that can be overridden and extended.
   → easy to develop an "object-oriented" hardware library for reuse purposes.

2. Classes necessity (16/18) RISC-V courses around the world

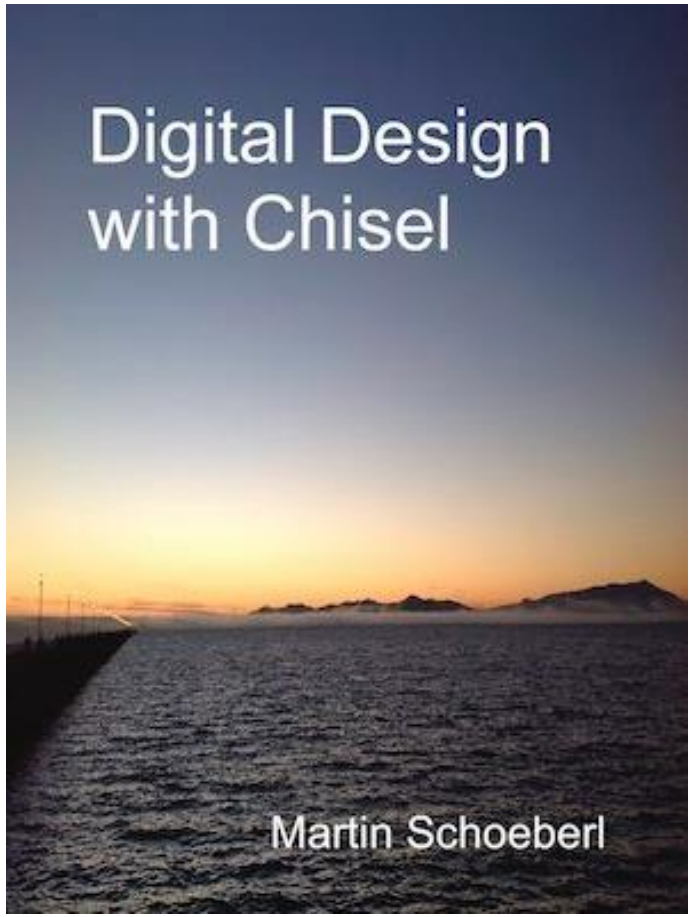
Beginners often started with CHISEL tutorial and specification.

CHISEL tutorials:

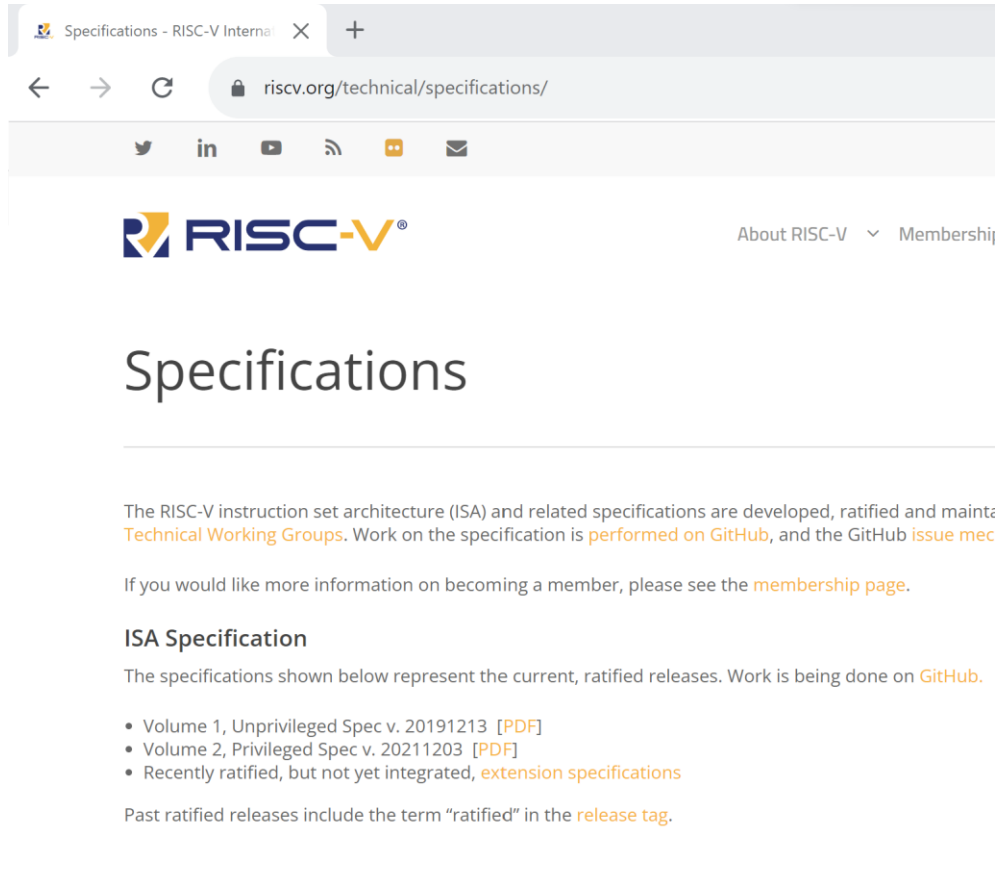


https://github.com/ucb-bar/chisel-tutorial

CHISEL coding:



RISC-V specification:



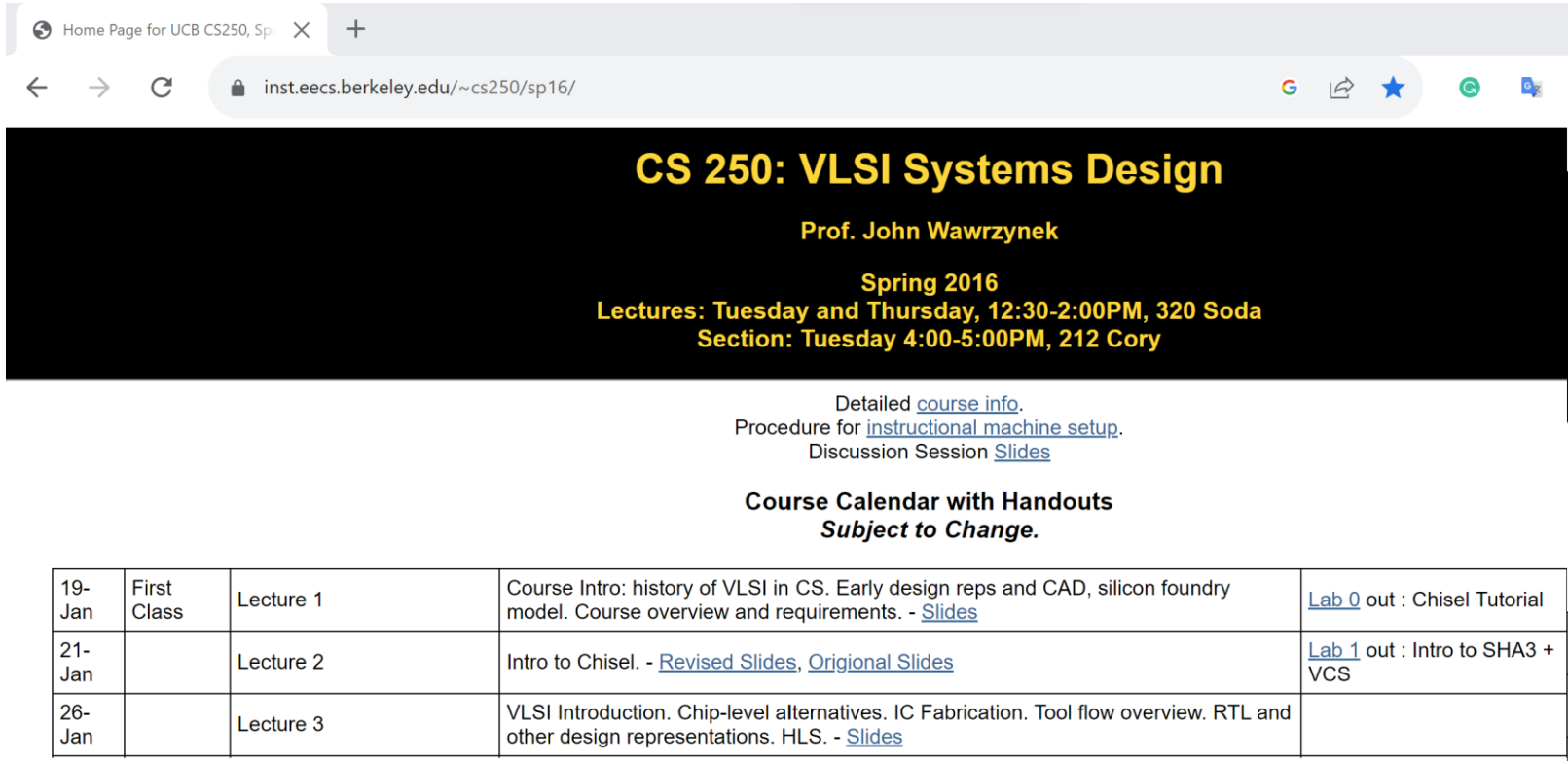
https://riscv.org/technical/specifications/

2. Classes necessity (17/18) RISC-V courses around the world

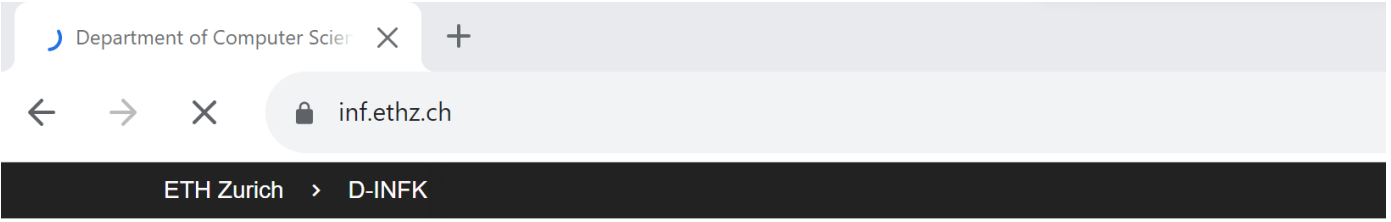
For University courses

From Berkeley University:

https://inst.eecs.berkeley.edu/~cs250/sp16/



19-Jan	First Class	Lecture 1	Course Intro: history of VLSI in CS. Early design reps and CAD, silicon foundry model. Course overview and requirements. - Slides	Lab 0 out : Chisel Tutorial
21-Jan		Lecture 2	Intro to Chisel. - Revised Slides, Origional Slides	Lab 1 out : Intro to SHA3 + VCS
26-Jan		Lecture 3	VLSI Introduction. Chip-level alternatives. IC Fabrication. Tool flow overview. RTL and other design representations. HLS. - Slides	



I know that ETH Zurich has a RISC-V course, but seem they didn't public it.

2. Classes necessity (18/18) RISC-V courses around the world

For University courses **in Japan**

Their course is to use Verilog to design a processor that can execute RISC-V instructions.

From Hosei University:

法政大学 HOSEI University

WEBシラバス ホーム

Finally, from our University of UEC:

国立大学法人 電気通信大学 The University of Electro-Communications

Here at **UEC**:

- We're the 3rd Uni. in the world, the 1st Uni. in Japan, that has a RISC-V course from ***idea*** to ***full chip***.

Outline

1. Introduction
2. Why are these classes needed?
3. Classes content
4. Conclusion

3. Classes content (1/6) Five main courses

Five main courses: from *basic* to *chip* and beyond

Number	Name
1	RISC-V Computer System Integration
2	Hardware Design with Chisel/Scala
3	Linux on RISC-V
4	VLSI Design with RISC-V System-on-Chip (SoC)
5	Cyber-security with RISC-V Computer System

- **「Course 1」：** RISC-V Computer System Integration

  Learn how to generate a RISC-V computer system. Make, compile, and debug the system.
  Learn how to add custom hardware to the integration.

3. Classes content (2/6) Five main courses

Number	Name
1	RISC-V Computer System Integration
2	Hardware Design with Chisel/Scala
3	Linux on RISC-V
4	VLSI Design with RISC-V System-on-Chip (SoC)
5	Cyber-security with RISC-V Computer System

- **「Course 2」：** Hardware Design with Chisel/Scala

  Learn how to code with Chisel/Scala. How to develop your circuit using Chisel/Scala.
  Learn how to reuse other works (system, IP, etc.) via an open-source library/repository.

- **「Course 3」：** Linux on RISC-V

  Learn how to boot Linux on the RISC-V computer system.
  Learn how to add your custom hardware, then have Linux recognizes it.

3. Classes content (3/6) Five main courses

Number	Name
1	RISC-V Computer System Integration
2	Hardware Design with Chisel/Scala
3	Linux on RISC-V
4	VLSI Design with RISC-V System-on-Chip (SoC)
5	Cyber-security with RISC-V Computer System

- **「Course 4」：** VLSI Design with RISC-V System-on-Chip (SoC)

  Learn how to make a chip, a big one!…like a fully Linux-compatible SoC

- **「Course 5」：** Cyber-security with RISC-V Computer System

  Learn how to customize a computer system to increase its security level.
  (adding crypto-cores, setup secure boot procedure, root-of-trust isolation, trusted execution environment, etc.)

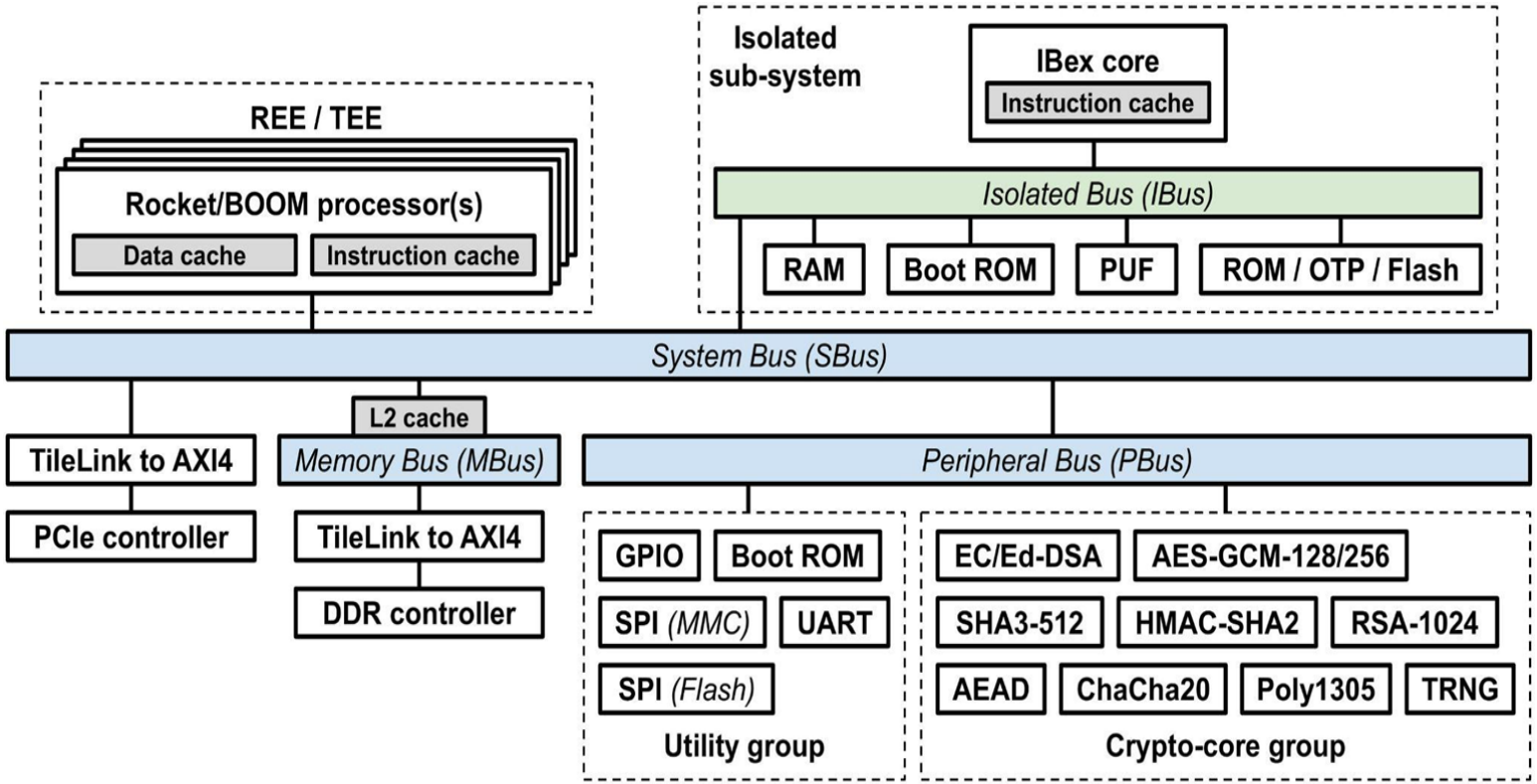
3. Classes content (4/6) Example: VLSI design class at UEC

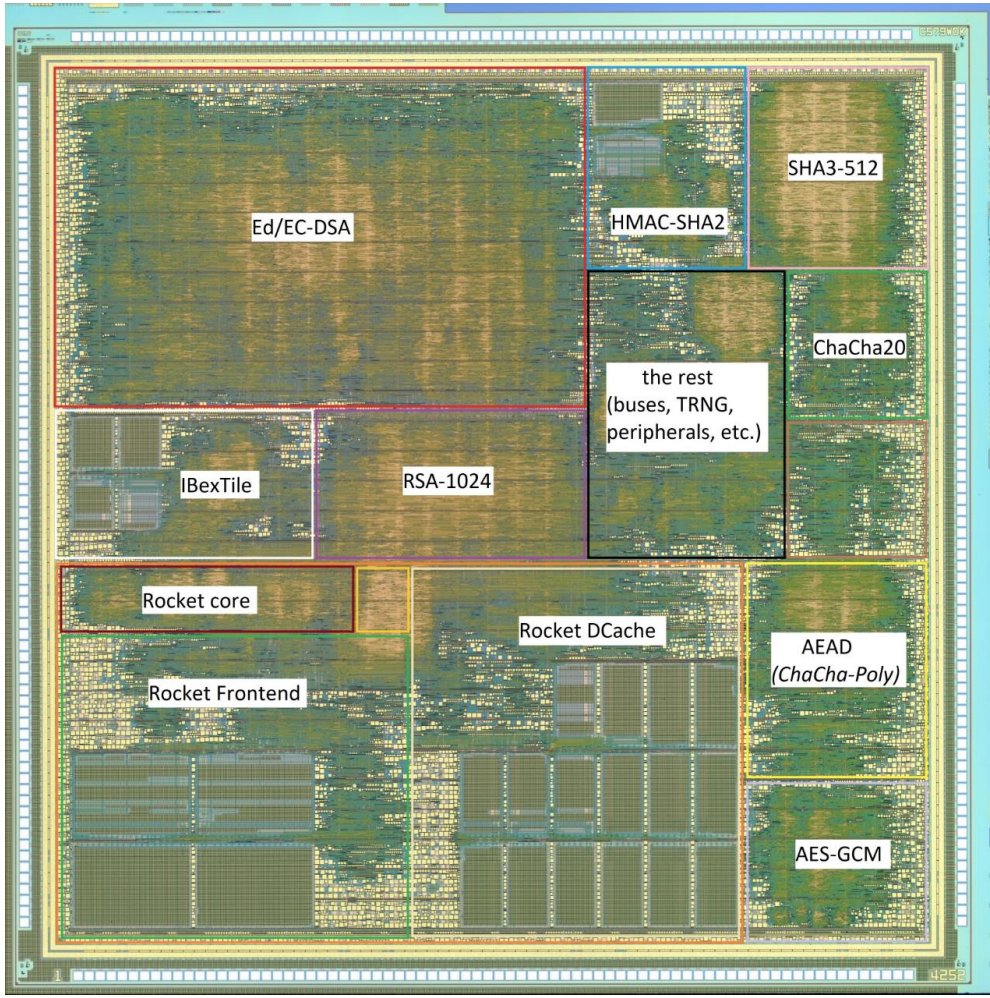
Week	Title
1	RISC-V Introduction
2	VexRiscv: a simple 32-bit MCU
3	Computer system with Rocket: Introduction, make, & program Arty-A7
4	Computer system with Rocket: System details & system modifications
5	Computer system with Rocket: Boot sequence & making your software
6	Midterm review
7	Computer system with Rocket: Make a custom hardware
8	Computer system with Rocket: Make a custom hardware (continue)
9	Computer system with Rocket: Custom hardware with external IOs
10	Make a chip in ROHM-180nm: Introduction & using templates
11	Make a chip in ROHM-180nm: Flat layout with VexRiscv
12	Make a chip in ROHM-180nm: Hierarchy layout with Rocket
13	Make a chip in ROHM-180nm: Frame integration
14	Class summary (total review)

- We're doing the VLSI design class at UEC.
- The content was the combined course **1** *(integration)* and **4** *(VLSI design)*

3. Classes content (5/6) Example: Crypto-SoC

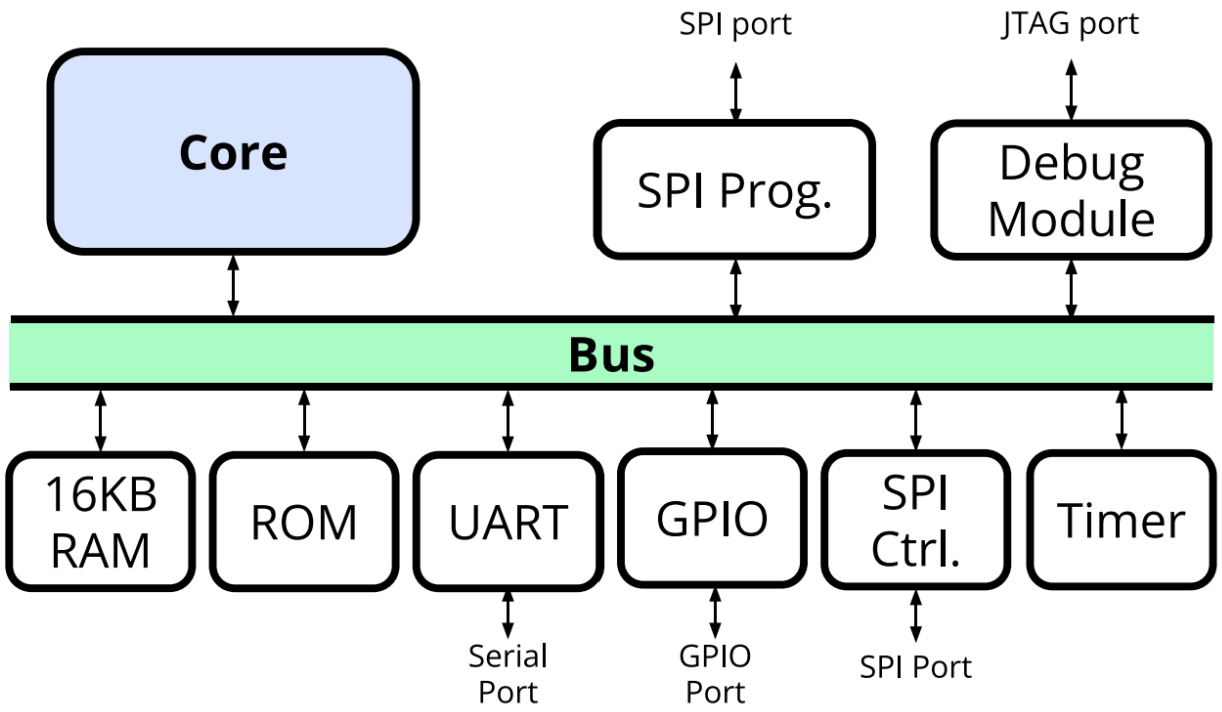
- Crypto-SoC with crypto-core accelerators, hidden MCU for secure boot, and complex key scheduling.

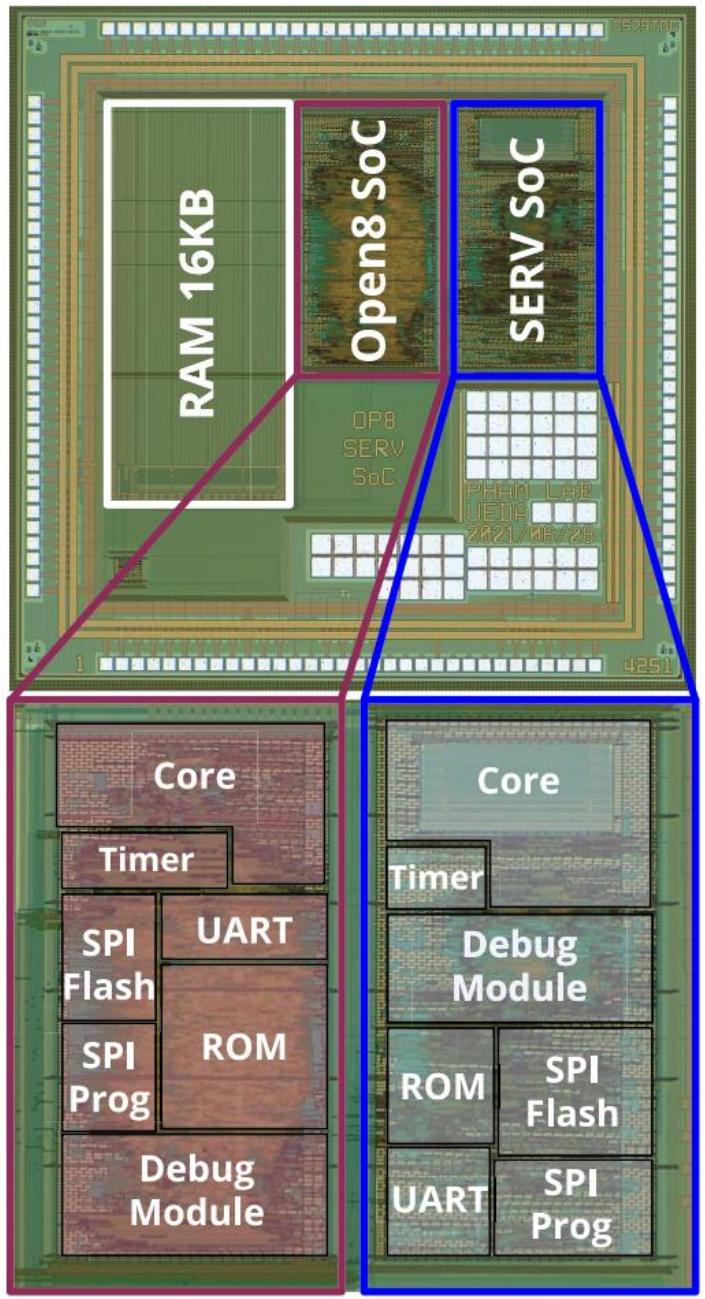




3. Classes content (6/6) Example: ULP-SoC

- ULP-SoC with small processor(s) and simple peripherals to achive sub-μW power consumption.





Outline

1. Introduction
2. Why are these classes needed?
3. Classes content
4. Conclusion

4. Conclusion (1/2) Keys takeaway for participants

No.	Name	Core knowledge
1	RISC-V Computer System Integration	• How to add your custom hardware *(Verilog or VHDL)* to an existing RISC-V computer system → generate a new → control/debug it in software
2	Hardware Design with Chisel/Scala	• Learn how to design a hardware circuit using Scala/Chisel • Learn how to create a new RISC-V computer system using open Scala/Chisel libraries
3	Linux on RISC-V	• Lean how to boot Linux on a RISC-V computer system → control your custom hardware after Linux boot
4	VLSI Design with RISC-V System-on-Chip (SoC)	• Make a RISC-V chip: flat layout for a simple system, or hierarchy layout for a complex system
5	Cyber-security with RISC-V Computer System	• Learn how to modify a RISC-V computer system for security purposes: secure boot, cryptographic acceleration, etc.

4. **Conclusion** (2/2) Past-Current-Future classes

Past classes:

- *Sep. 2022:*

Place Academy of Cryptography Techniques (ACT), Hanoi, Vietnam
Content Course **1**: *RISC-V Integration*

Current classes:

- *Oct. 2022 ~ Jan. 2023:*

Place University of Electro-Communications (UEC), Tokyo, Japan
Content Course **1**: *RISC-V Integration*
Course **4**: *VLSI Design with RISC-V SoCs*

Future classes:

- *~ Aug. 2023:*

Place Academy of Cryptography Techniques (ACT), Hanoi, Vietnam
Content Course **1**: *RISC-V Integration*
Course **5**: *Cyber-security with RISC-V*

THANK YOU

2022/12