

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

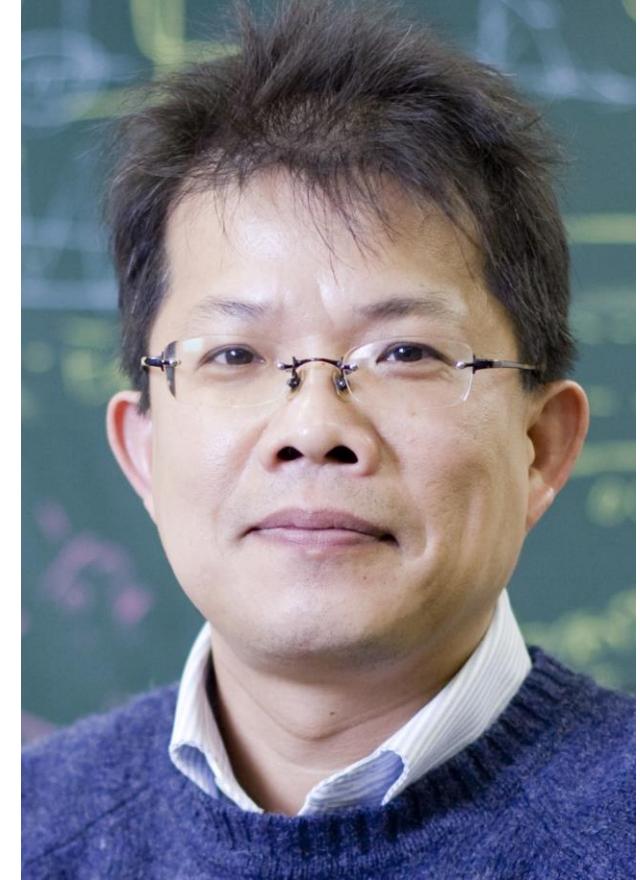
Outline

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

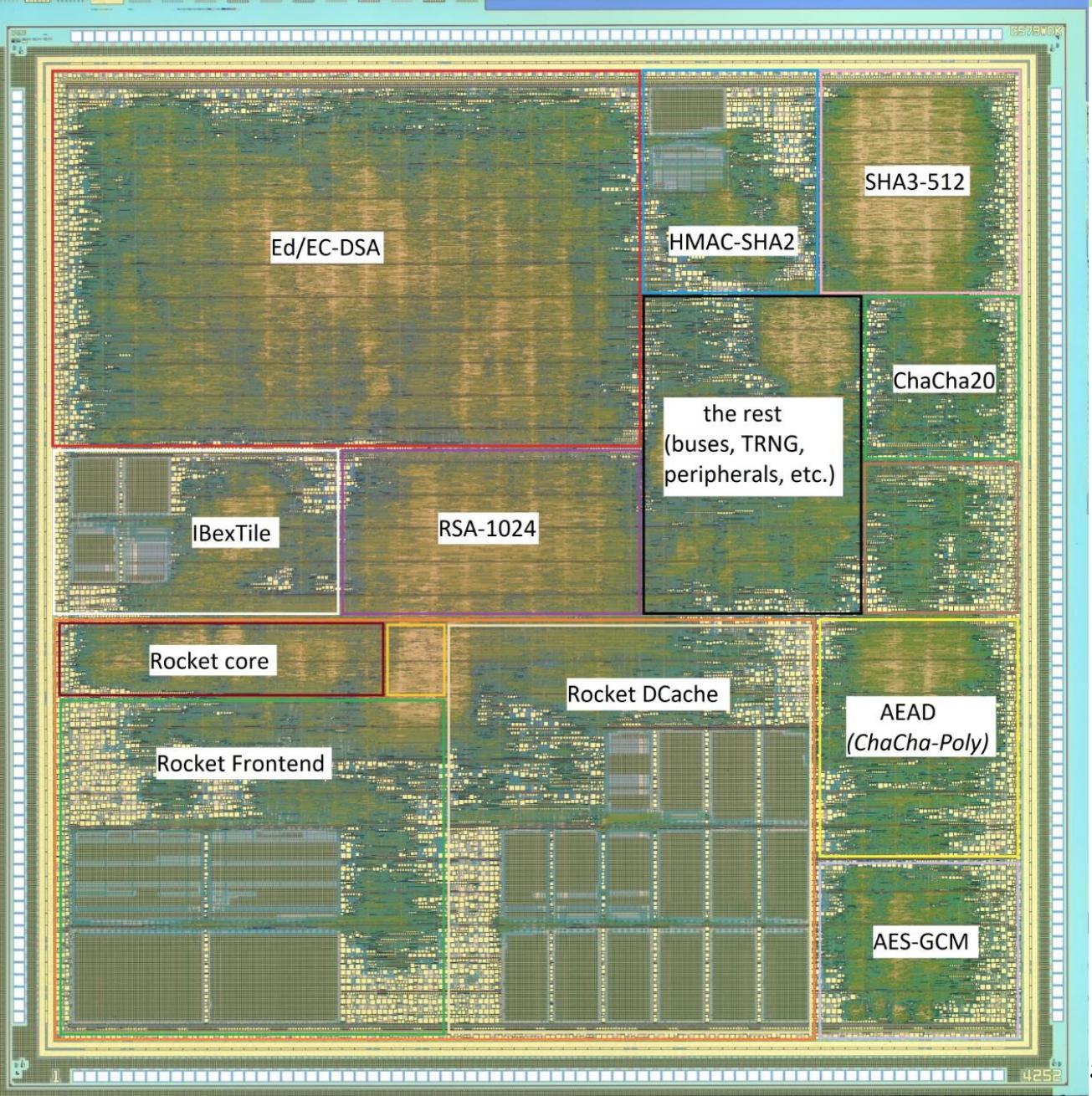
<https://thuchoang90.github.io/>

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

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

(VLSI lab website)

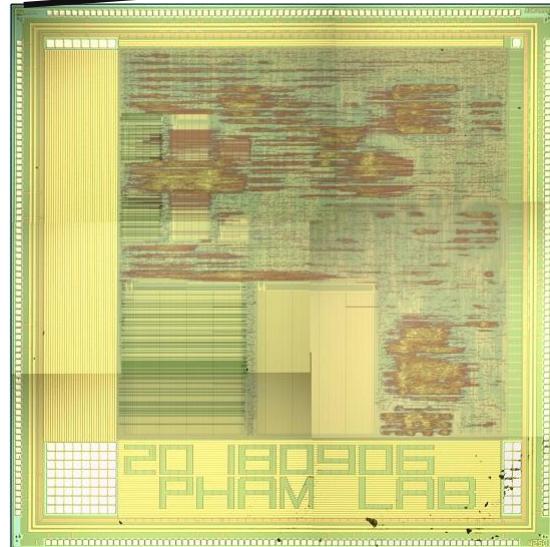
1. Introduction (2/9) Laboratory



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

Crypto-SoC

2018

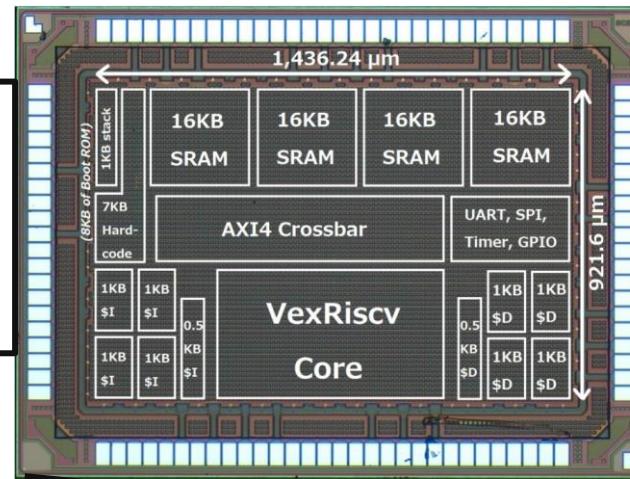


Rocket64(1)
ROHM180nm
5x5-mm²

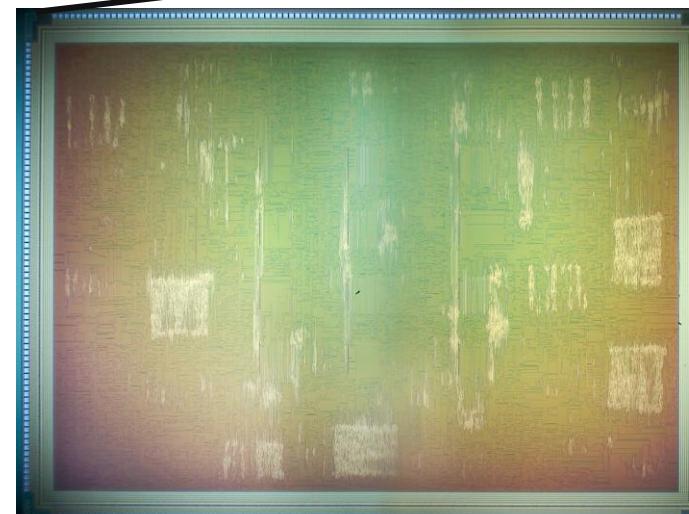
VexRiscv32(1)
SOTB65nm
2x1.5-mm²

09

2019



08



Rocket64(4)
ROHM180nm
5x7.5-mm²

2020

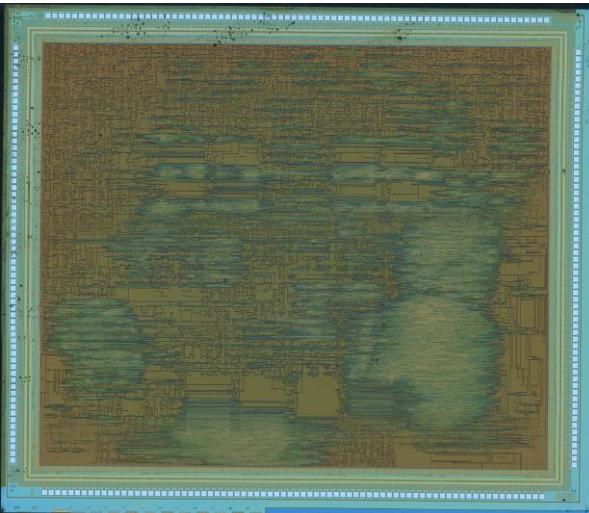
1. Introduction (4/9) Crypto-SoC series

Crypto-SoC

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

2020

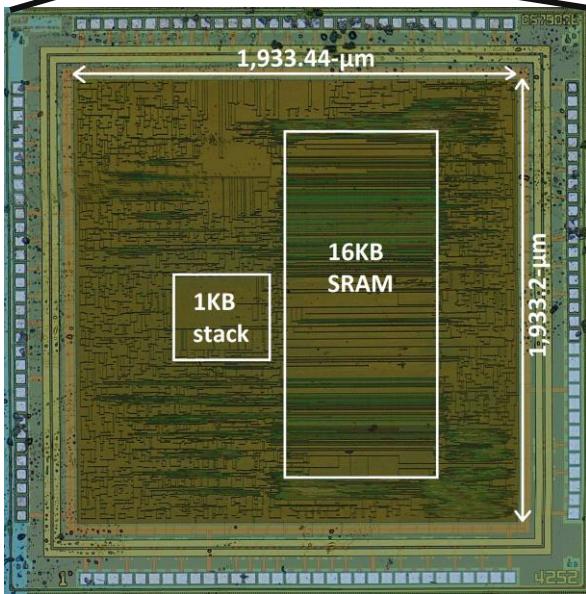
01



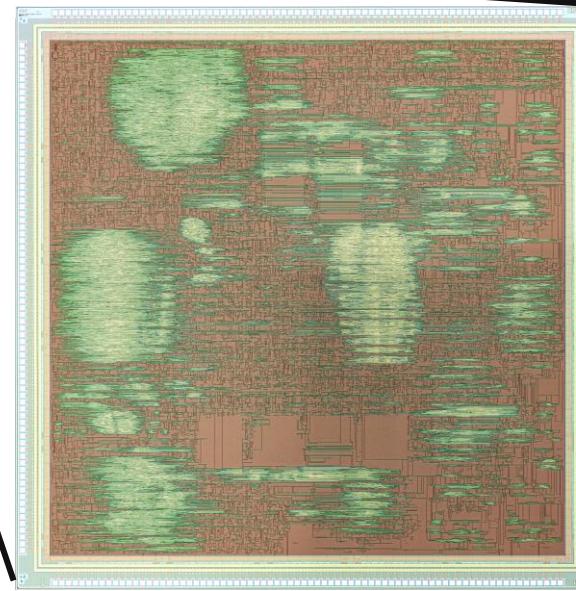
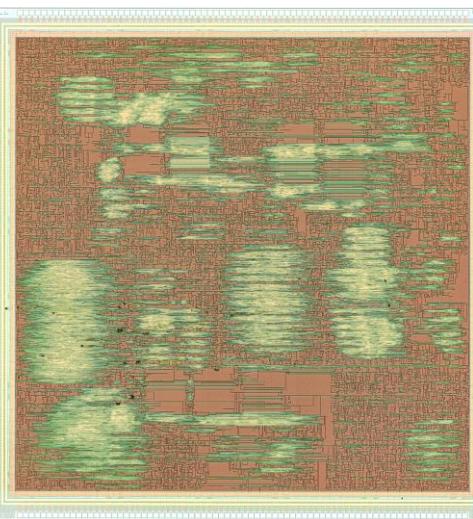
Boom64(1) + Crypto-cores ROHM180nm
5x5-mm2

2021

06



VexRiscv32(1)
ROHM180nm
2.5x2.5-mm2



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

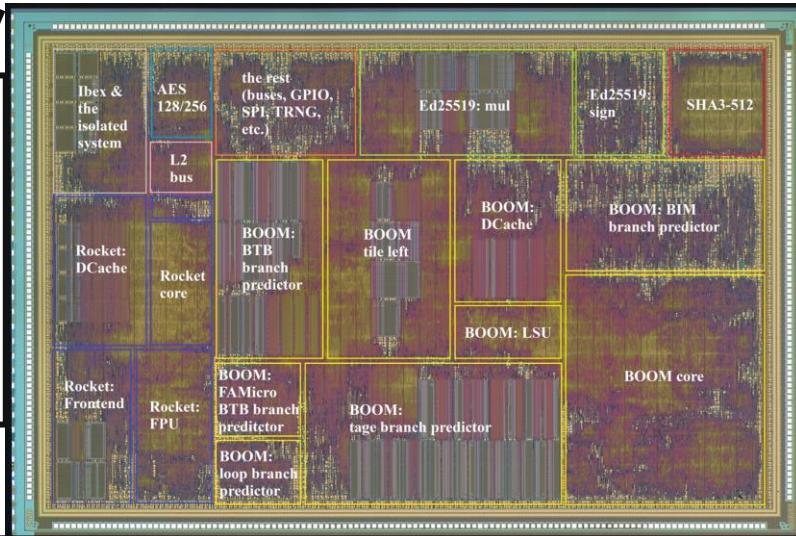
1. Introduction (5/9) Crypto-SoC series

Crypto-SoC

2021

02

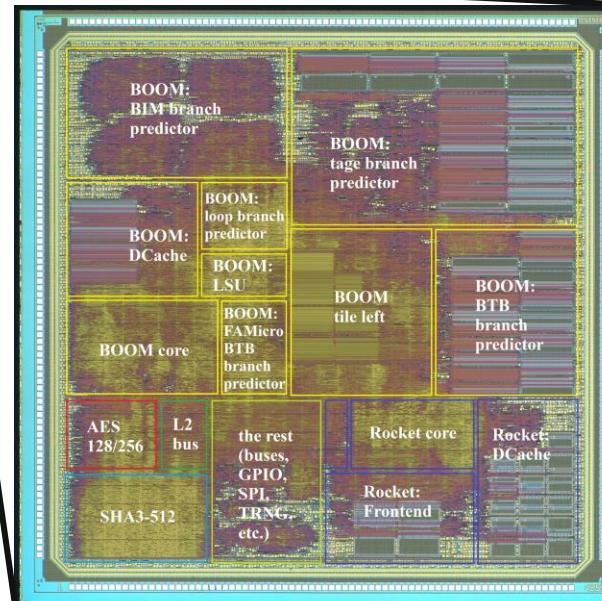
Rocket64(1) + Boom64(1)
+ Crypto-cores & TRNG
+ Secure boot
ROHM180nm : 5x7.5-mm²



2022

06

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



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

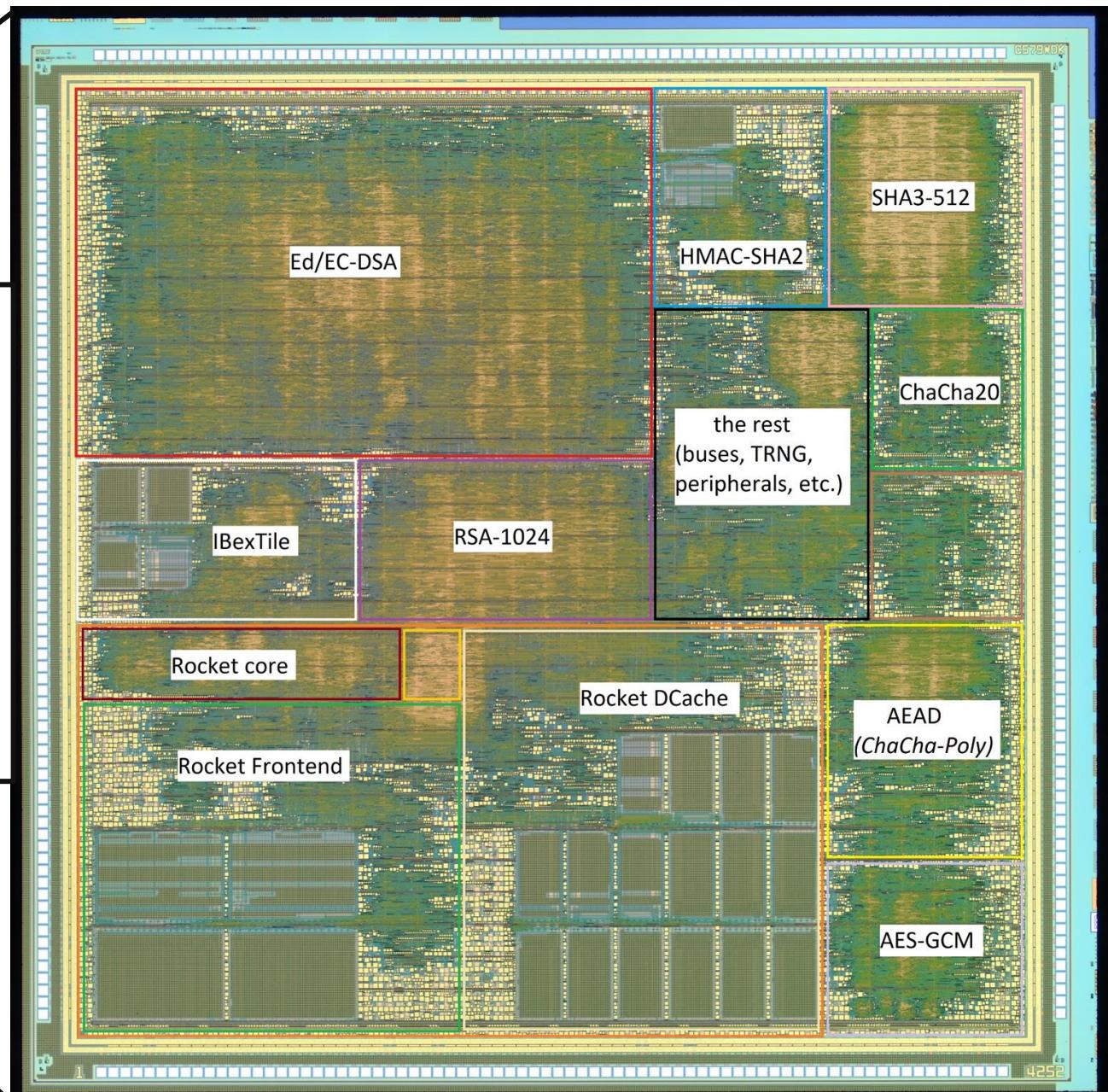
1. Introduction (6/9) Crypto-SoC series

Crypto-SoC

2022

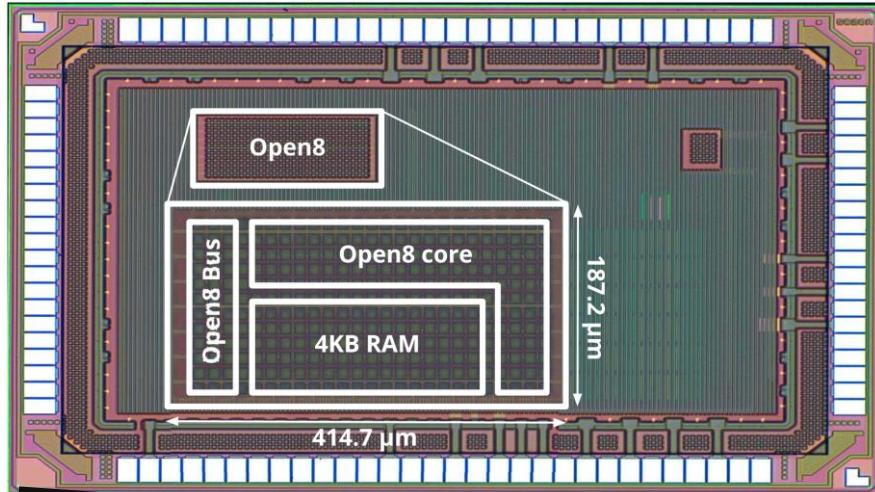
02

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

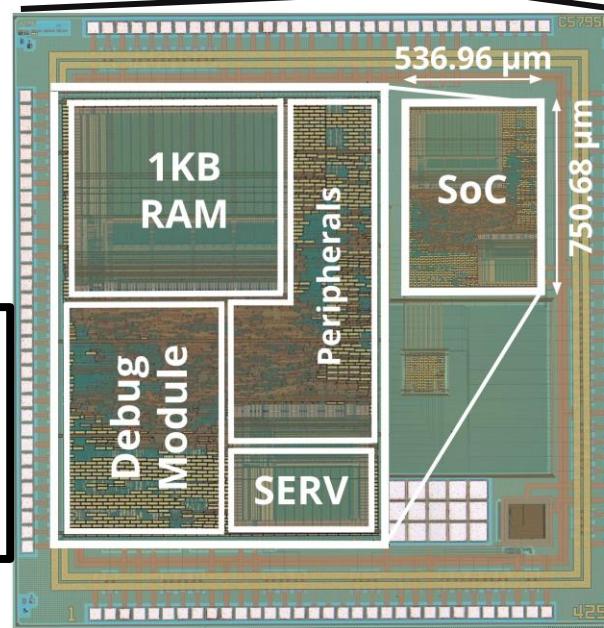
2021

07

2022

10

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



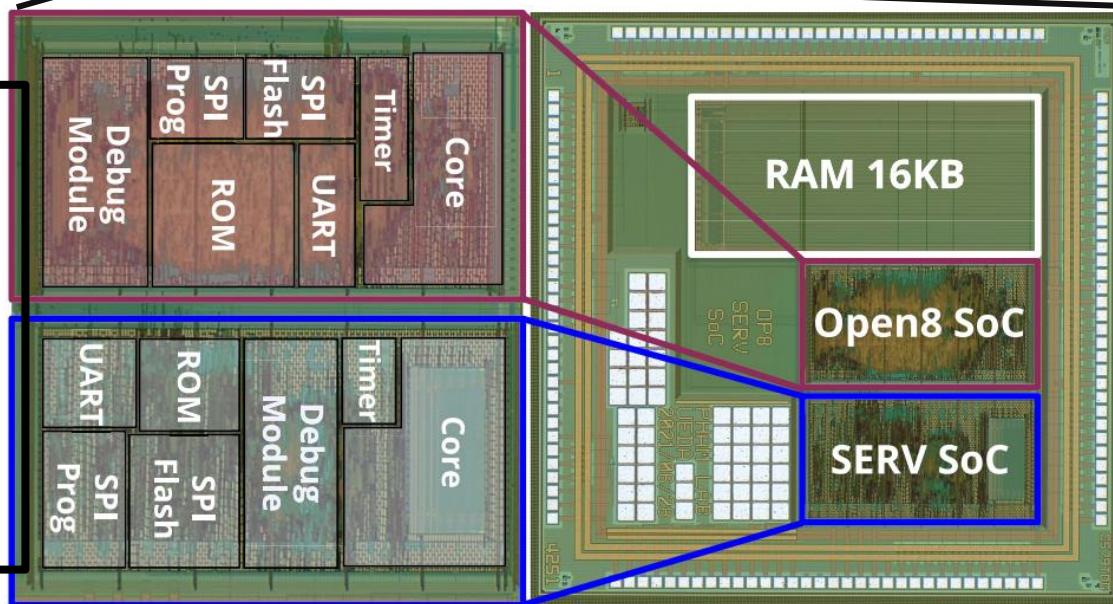
1. Introduction (8/9) Ultra-Low-Power (ULP) SoC

ULP-SoC

2022

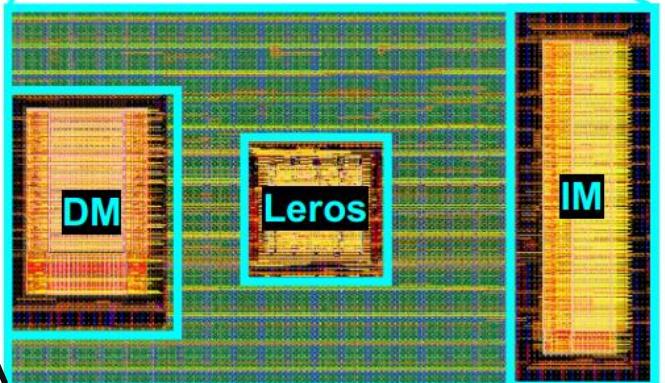
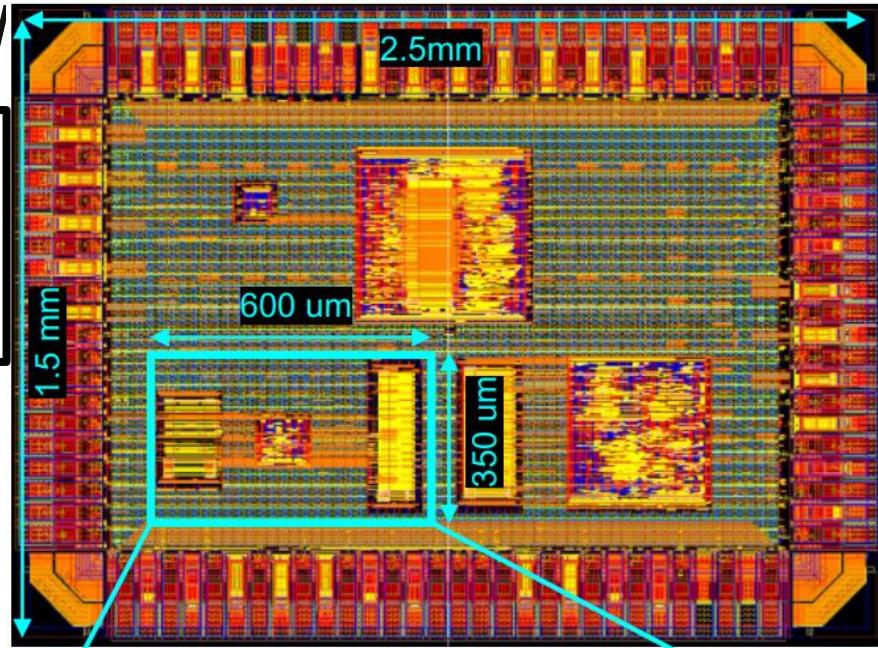
03

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



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

09



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 Exchange: Available Software

- Simulators
- Object Toolchain
- Debugging
- C Compilers & Libraries

- Bootloaders & Monitors
- Hypervisors
- OS & Kernels

- Non-C Compilers/Runtimes
- IDEs & SDKs
- Security
- Machine Learning & AI

- Configuration
- Verification Tools
- Accelerated Libraries

RISC-V Exchange: Cores & SoCs

| Name | Supplier | Links | Capability | Priv. spec | User spec |
|-----------|--------------|---------|------------|------------|-----------------|
| RV32EC_P2 | IQonIC Works | Website | RV32 | 1.11 | RV32E[M]C/RV32I |

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: Integer

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 |
| | 0.0 | Draft |
| | 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

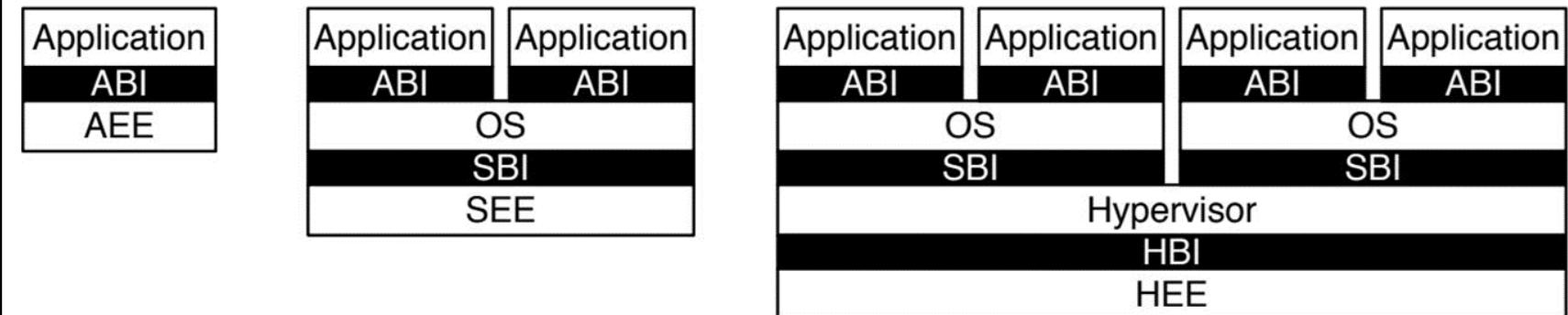
To support an Operating System (OS), the ISA has to support the OS stack or the **M-/S-/U-mode**.

RISC-V privileged architecture:

| RISC-V Modes | | |
|--------------|------------------|-------|
| Level | Name | Abbr. |
| 0 | User/Application | U |
| 1 | Supervisor | S |
| 2 | 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:

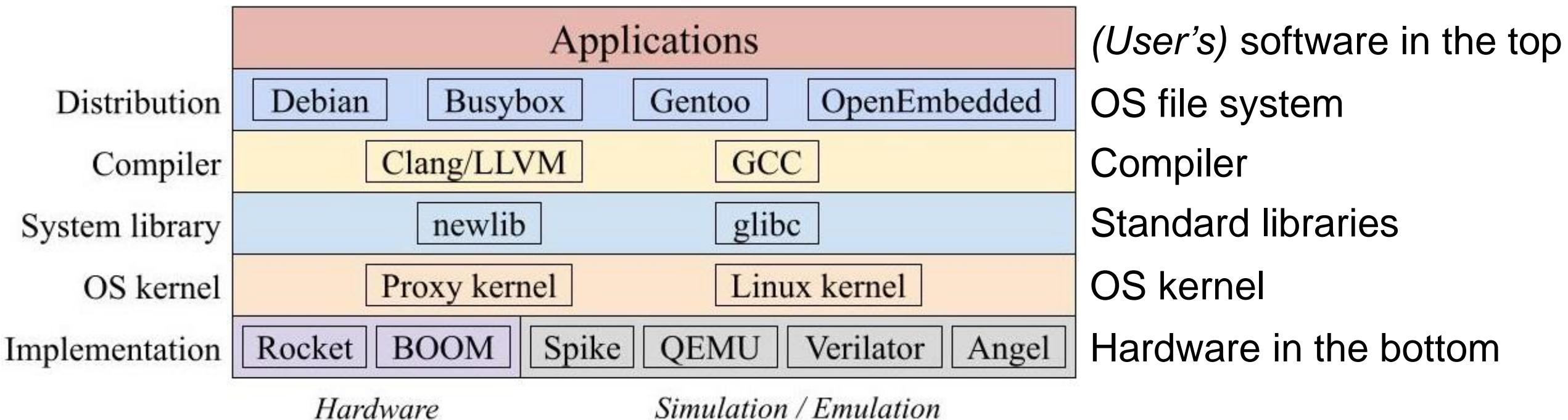


RISC-V ISA not only supports the OS stack, but also provides a **privileged architecture**.

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

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

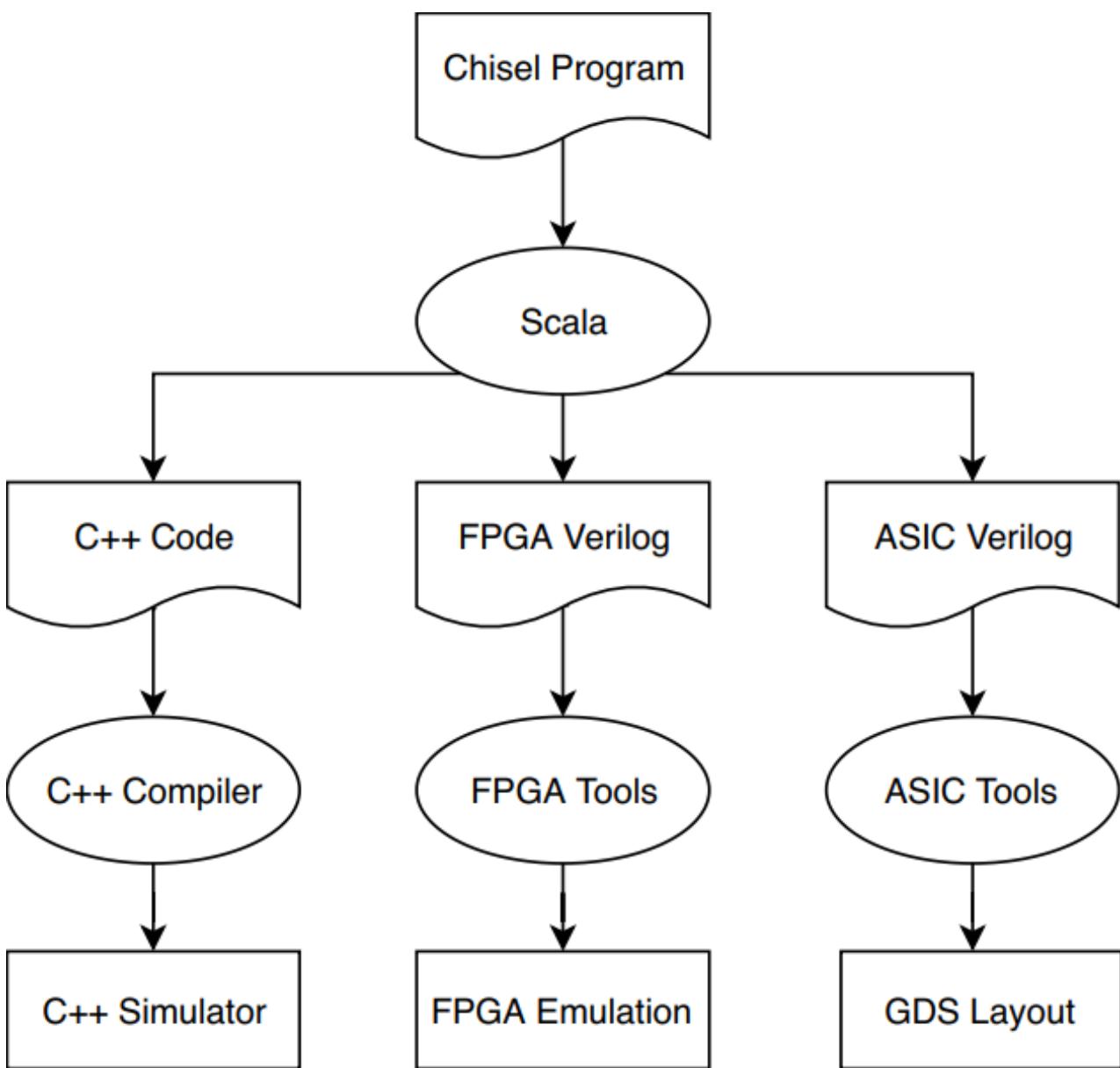
RISC-V toolchain and its ecosystem



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.

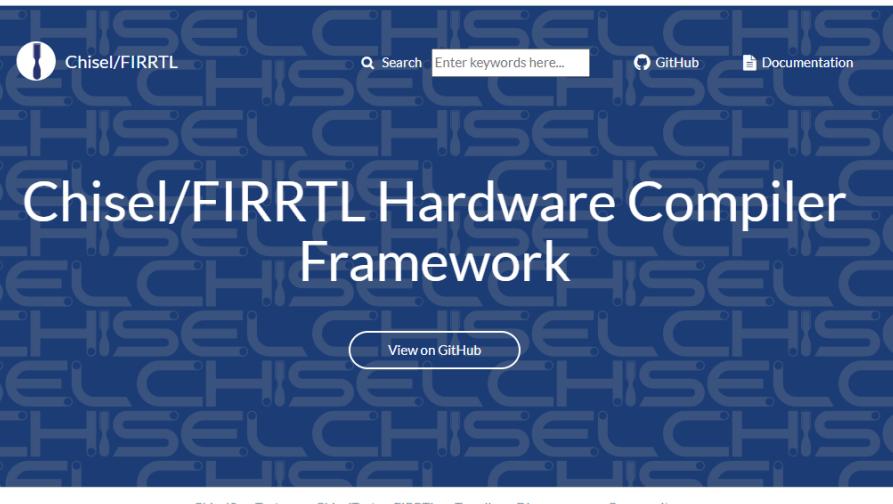
2. Classes necessity (5/18) CHISEL: a new way to code



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>

4 authors Bump for 3.3.1 (#162) e567a5f on Jun 2, 2020 426 commits

- doc Bump for 3.3.1 (#162) 2 years ago
- project Bump for 3.3.1 (#162) 2 years ago
- src Bump for 3.3.1 (#162) 2 years ago
- .gitignore ignore files created when viewing doc/tutorial/*.te... 4 years ago
- LICENSE.txt Add LICENSE.txt and reference to it in all .scala files. 6 years ago
- README.md Merge branch 'master' into release 4 years ago
- build.sbt Bump for 3.3.1 (#162) 2 years ago
- run-examples.sh Bump Scala compiler versions, sbt version to 1.1.1 ... 5 years ago
- run-problem.sh Bump Scala compiler versions, sbt version to 1.1.1 ... 5 years ago
- run-solution.sh Bump Scala compiler versions, sbt version to 1.1.1 ... 5 years ago

README.md

Chisel Tutorials (Release branch)

jackkoenig Remove chisel-template and chisel-tutorial c2cd7e7 on Jan 12 275 commits

- chisel-testers @ ce4e027 Bump .x branches 2 years ago
- chisel3 @ 688c86a Bump .x branches 2 years ago
- chiseltest @ ca3d881 Rename chisel-testers2 -> chiseltest 9 months ago
- diagrammer @ cfe9d1e Bump .x branches 2 years ago
- doc Update documentation. 2 years ago
- dsptools @ 0131a4e Bump .x branches 2 years ago
- firrtl @ 15013df Bump .x branches 2 years ago
- firrtl-interpreter @ de59... Bump .x branches 2 years ago
- treadle @ 109ea9b Bump .x branches 2 years ago
- .gitignore Update Makefile with individual project dependen... 3 years ago
- .gitmodules Remove chisel-template and chisel-tutorial 9 months ago
- LICENSE Create LICENSE 4 years ago
- README-OLD.md Fixing up 2 years ago
- README.md Rename chisel-testers2 -> chiseltest 9 months ago
- deps.bare Remove chisel-template and chisel-tutorial 9 months ago
- version.yml Remove chisel-template and chisel-tutorial 9 months ago

README.md

The chisel-release Repository

Released CHISEL
sources:
<https://github.com/ucb-bar/chisel-release> 19

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>

The screenshot shows the GitHub repository for Chipyard. At the top, there is a file tree with 'README.md'. Below it is the Chipyard logo, which features a stylized 'Y' shape composed of circuit boards and cranes, with the word 'CHIPIYARD' in large blue and yellow letters. Under the logo, there is a section titled 'Chipyard Framework' with a 'chipyard-ci-process' badge showing 'passing'. Below this is a 'Quick Links' section with three items: 'Stable Documentation' (link to https://chipyard.readthedocs.io/), 'User Question Forum' (link to https://groups.google.com/forum/#!forum/chipyard), and 'Bugs and Feature Requests' (link to https://github.com/ucb-bar/chipyard/issues). There is also a 'Using Chipyard' section with a note about documentation and a 'What is Chipyard' section.

The screenshot shows the GitHub repository for sifive/fpga-shells. It displays a list of commits from erikdanie, including: 'Merge pull request #158 from a...' (f9fb9fd on Dec 29, 2020), 'Cl: add a scala compilation check' (2 years ago), 'newshells checkpoint' (3 years ago), 'Add utility function for IBUF_LOW_POWER' (2 years ago), 'nfmac10g: fix a power-0 bug' (4 years ago), 'refactor tcl code that wasn't executing correctly' (2 years ago), 'Initial commit for fpga-shells' (5 years ago), 'improved clarity of documentation' (3 years ago), 'wake: use variable for package location' (2 years ago), and 'bump sifive-blocks (#157)' (2 years ago).

The screenshot shows the 'fpga-shells' section of the README.md file. It defines an FPGA shell as a Chisel module designed to wrap any SiFive core configuration. It states that the goal of the fpga-shell system is to reduce the number of wrappers to have only one for each physical device rather than one for every combination of physical device and core configuration.

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)

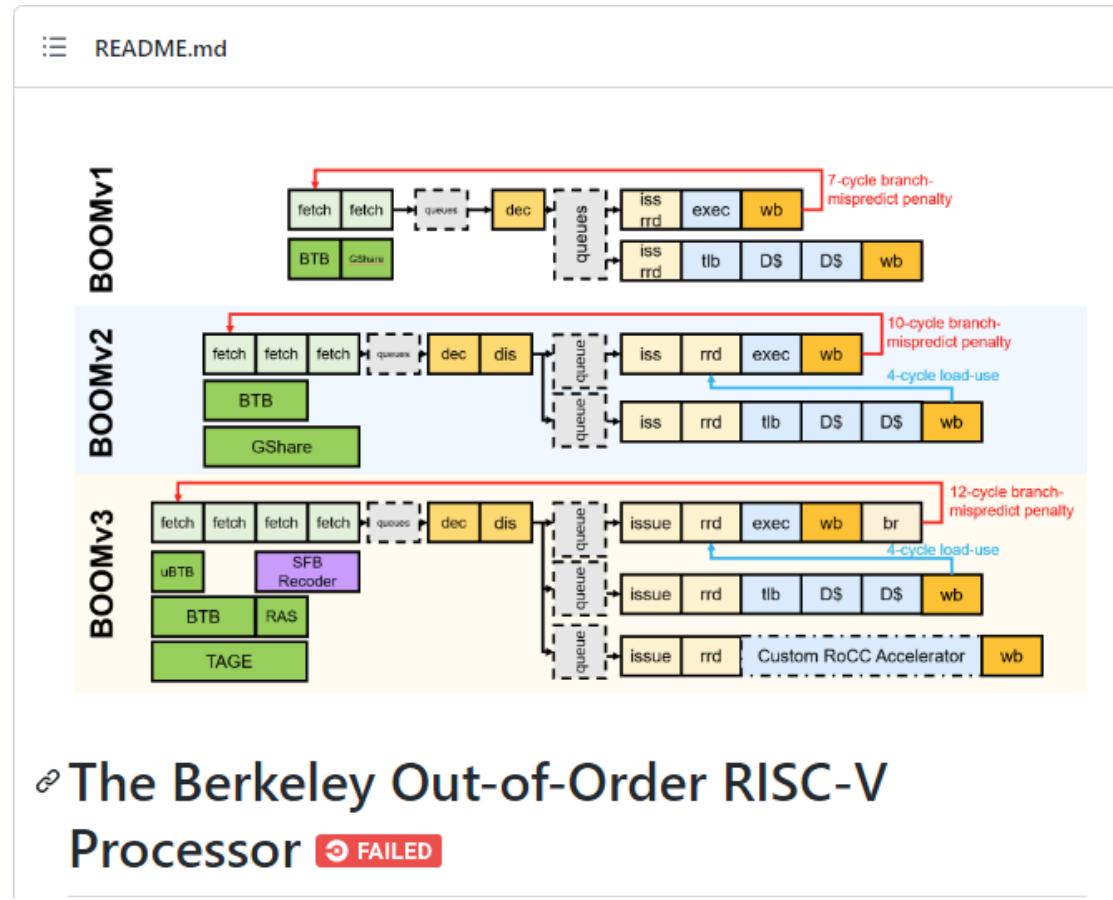
The screenshot shows the README.md page of the Rocket Chip Generator repository. It features a title "Rocket Chip Generator" with a rocket icon, a "Continuous Integration" badge showing "passing", and a table of contents. The main text describes the repository as containing the generator for the RISC-V Rocket Core. The table of contents lists various sections for using the generator, including quick instructions, what's in the repository, how to use it, how to parameterize, and debugging.

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>



2. Classes necessity (9/18) Open processors

Some famous RISC-V processors

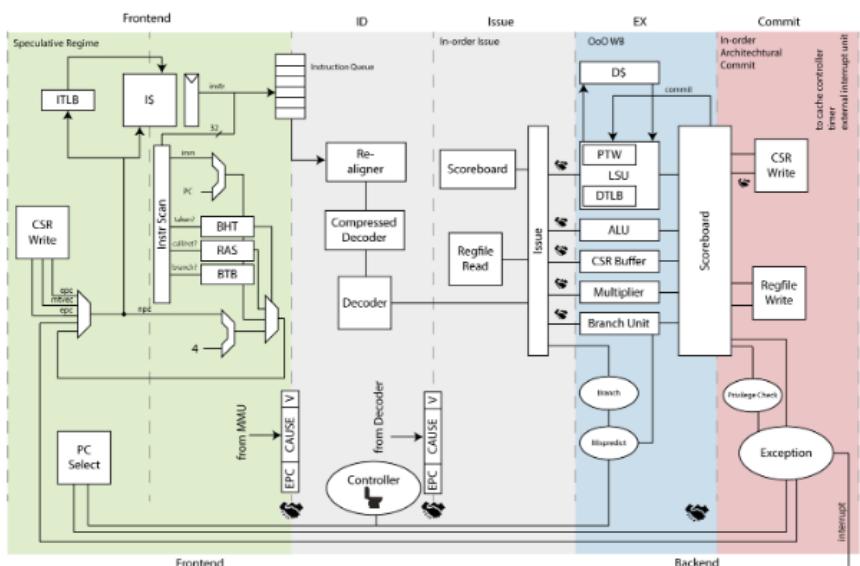
Ariane is a high-performance 64-bit in-of-order processor:

<https://github.com/lowRISC/ariane>

❖ 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.

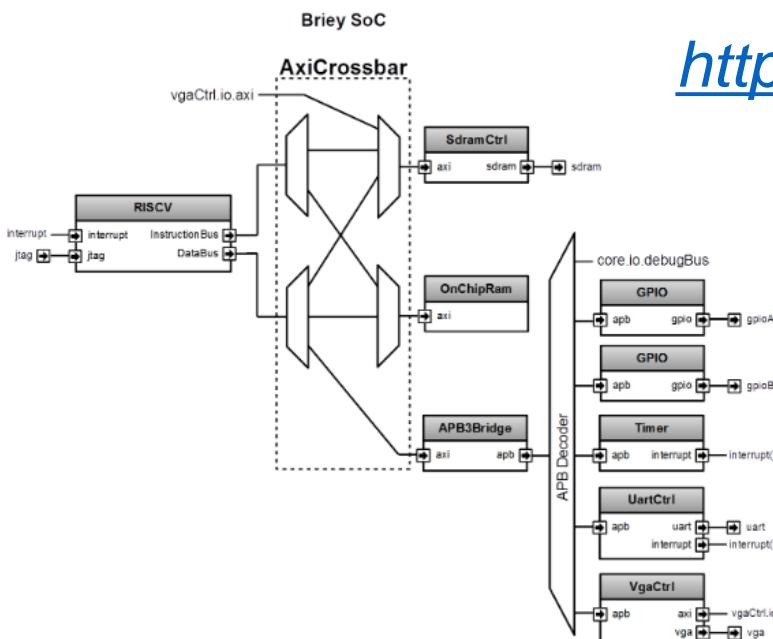
It has configurable size, separate TLBs, a hardware PTW and branch-prediction (branch target buffer and branch history table). The primary design goal was on reducing critical path length.



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:



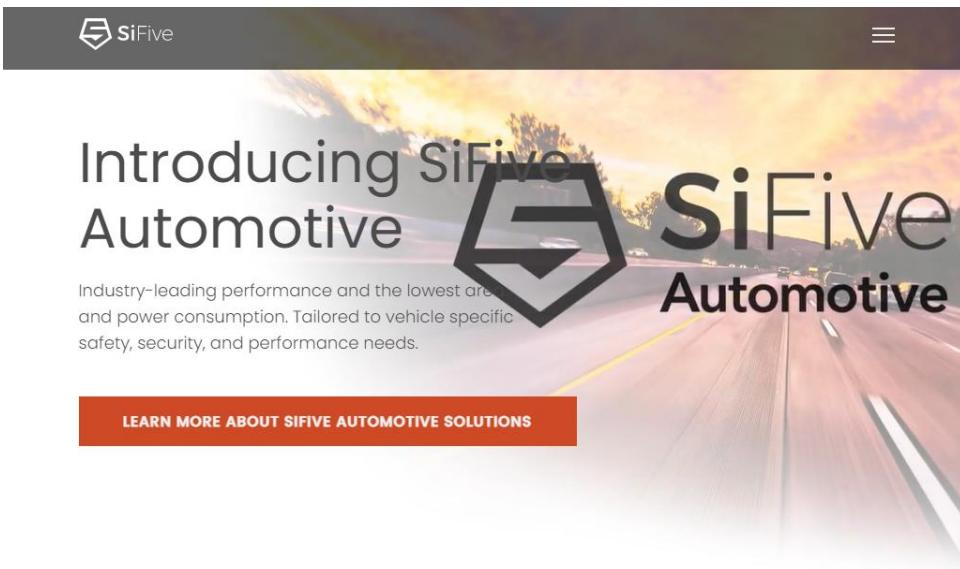
<https://github.com/SpinalHDL/VexRiscv>

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>



A screenshot of the SiFive news feed. It shows two articles: one about the SiFive X280 as Coprocessor in Google Datacenter (September 21, 2022) and another about SiFive Expanding Global Operations (June 28, 2022). Both articles have "Read More" buttons.

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>

0x5 HEX-Five Security Products Partners Company Downloads Contact FAQ



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:



CHIPS (Common Hardware for Interfaces, Processors and Systems) Alliance harnesses the energy of open source collaboration to accelerate hardware development.

Catch a Replay of our 2022 Technology Update : CHIPS Alliance First 2022 Update

The CHIPS Alliance develops high-quality, open source hardware designs relevant to silicon devices and FPGAs. By creating an open and collaborative environment, CHIPS Alliance shares resources to lower the cost of development. Companies and individuals can work together to develop open source CPUs, various peripherals, and complex IP blocks. CHIPS Alliance is open to all organizations who are interested in collaborating on open source hardware or software tools to accelerate the creation of more efficient and innovative chip designs.

- 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>

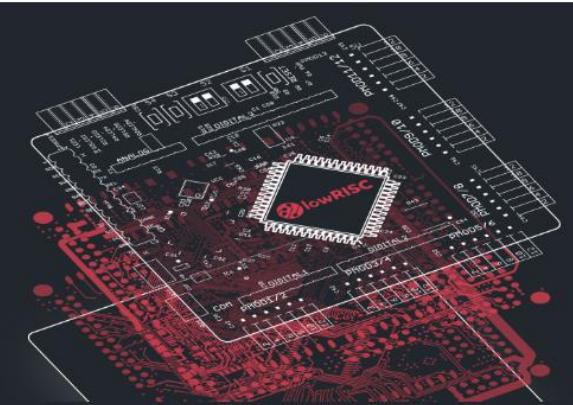


Our work Open Silicon Community Blog Jobs About us

[GitHub](#)

Open to the core

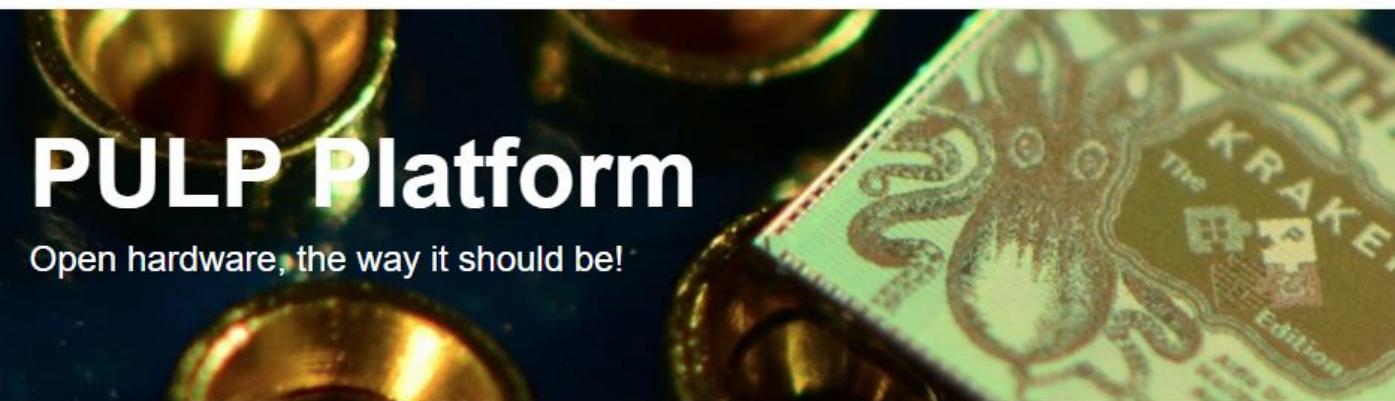
lowRISC is a not-for-profit company with a full stack engineering team based in Cambridge, UK. We use collaborative engineering to develop and maintain open source silicon designs and tools.



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

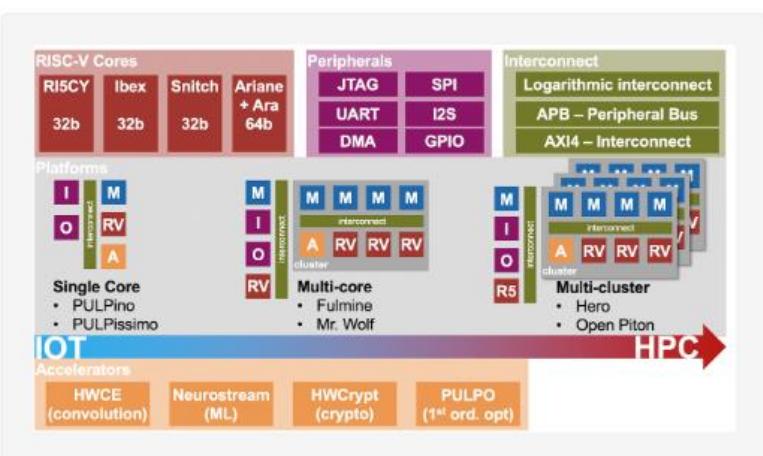
Some famous RISC-V groups

PULP Platform Resources ▾ About ▾ FAQ Privacy Policy Contact



PULP Platform

Open hardware, the way it should be!



[Github](#)

[Twitter](#)

[Youtube](#)

Latest news New

16 September 2022

Our team headed by Lorenzo Lamberti won
won the 1st prize in the Nanocopter AI
challenge at **IMAV2022** TU Delft. as well as
a special award in the Greenhouse

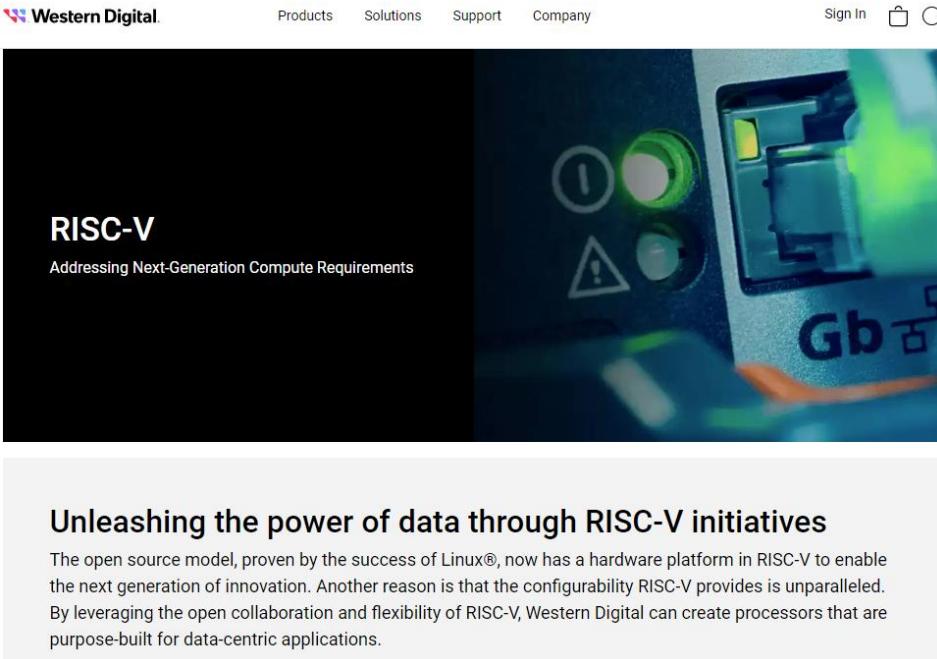
PULP is an open group started from a RISC-V project published by two universities.

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



The screenshot shows the Western Digital RISC-V page. At the top, there's a navigation bar with links for Products, Solutions, Support, Company, Sign In, and a search icon. Below the navigation is a large image of a circuit board component with a green LED and a blue metal case labeled "Gb". To the left of the image, the text "RISC-V" and "Addressing Next-Generation Compute Requirements" is displayed. A section titled "Unleashing the power of data through RISC-V initiatives" contains text about the open source model and its benefits for innovation.

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

Read more:

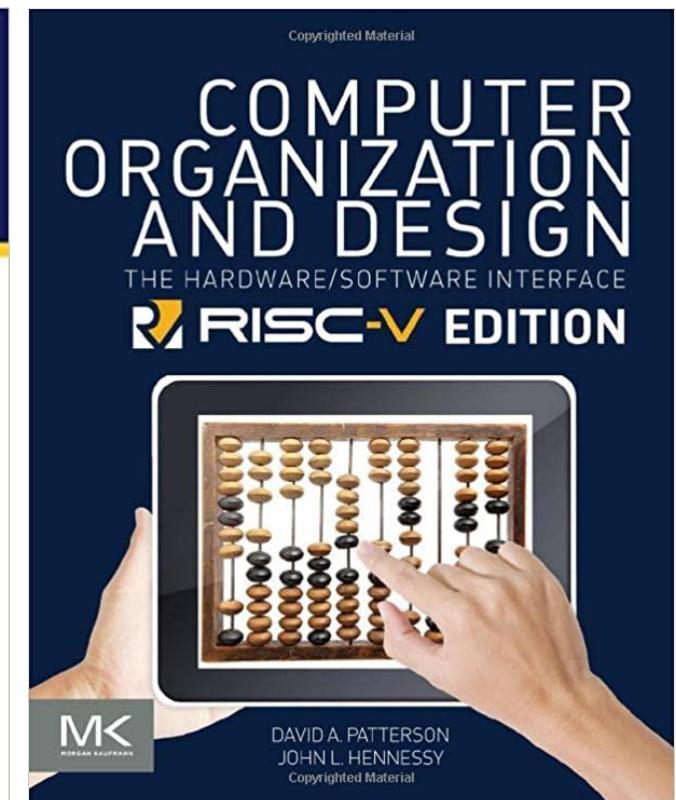
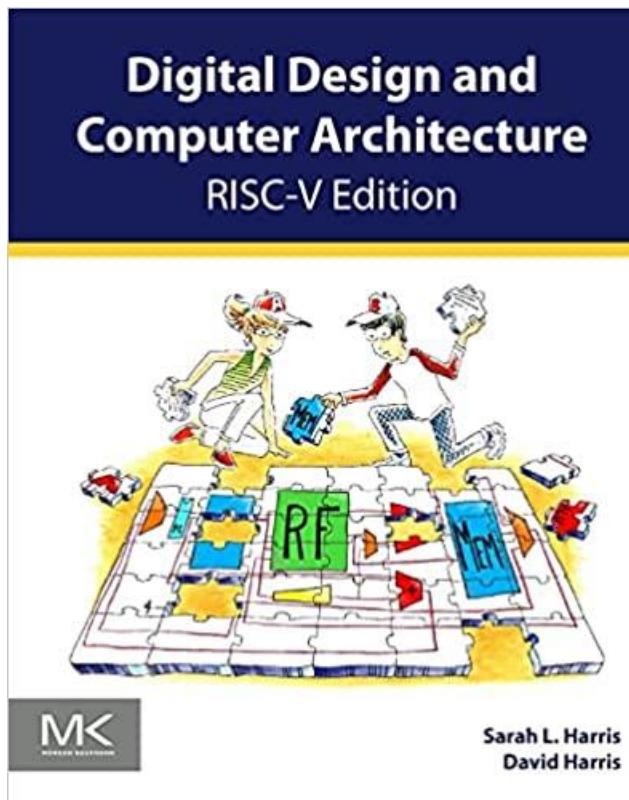
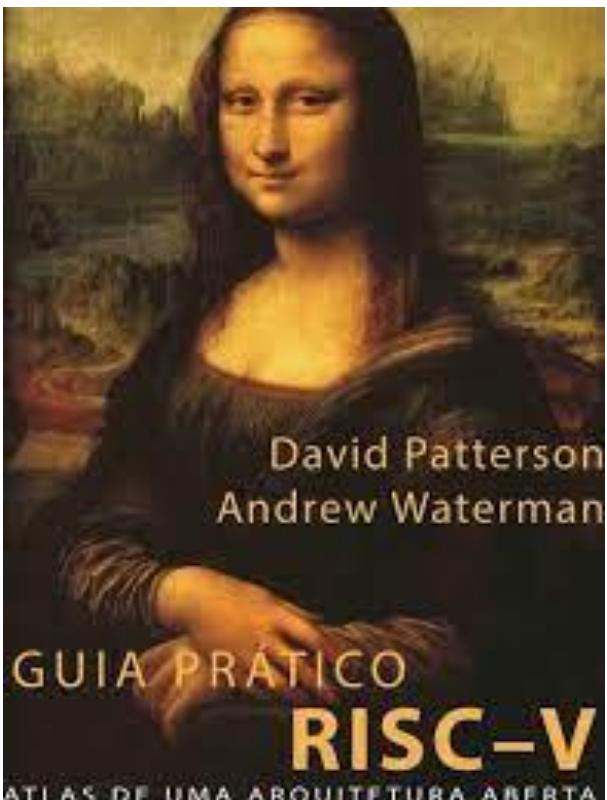
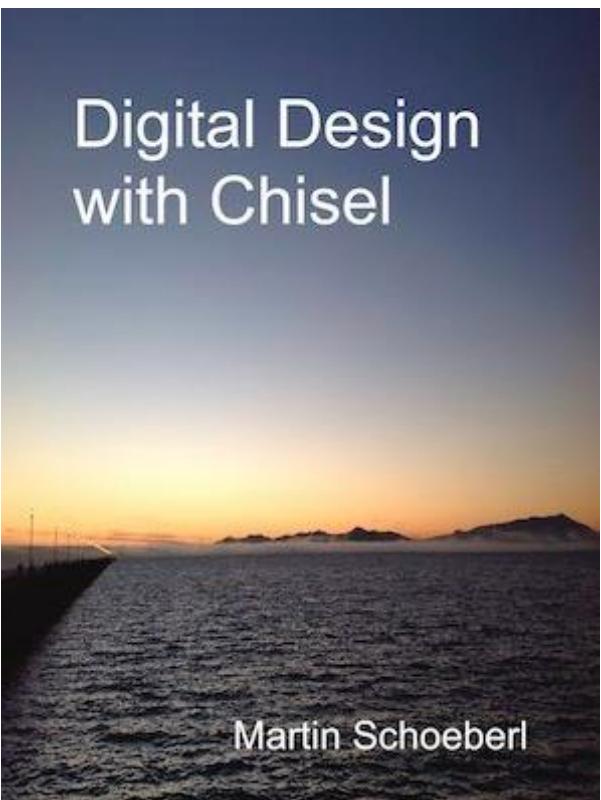
- At RISC-V Foundation website:
<https://riscv.org/news/2020/12/seagate-western-digital-outline-progress-on-risc-v-designs-carol-sliwa-searchstorage-techttarget-com/>
- At Seagate website:
<https://www.seagate.com/jp/ja/innovation/risc-v/>
- At WD website:
<https://www.westerndigital.com/company/innovation/open-source/risc-v>



The screenshot shows the Seagate RISC-V page. The top navigation bar includes links for 製品 (Products), ソリューション (Solutions), イノベーション (Innovation), サポート (Support), and 詳細 (Details). Below the navigation is a dark background with a bokeh light effect. The word "Innovations" is written in white. The main headline reads "RISC-V Enables a System on a Chip". The Seagate logo is at the bottom left, and the RISC-V logo is at the bottom right. A small note at the bottom states "New silicon design from Seagate powers data mobility and trust."

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

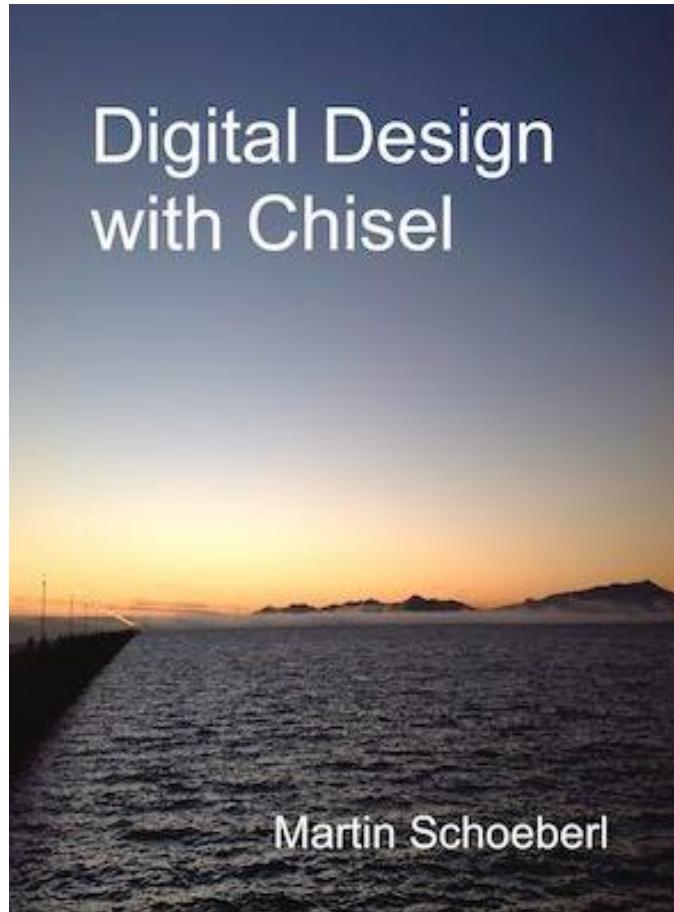
Beginners often started with CHISEL tutorial and specification.

CHISEL tutorials:

A screenshot of a GitHub repository page for "chisel-tutorial". The repository is public and has 89 stars. It contains 20 branches and 12 tags. The code tab is selected. A list of recent commits shows updates from June 2020, including bumping versions for 3.3.1 and adding LICENSE.txt. The README.md file is also visible at the bottom.

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

CHISEL coding:



RISC-V specification:

A screenshot of the RISC-V Specifications page on the official website. The page features the RISC-V logo and a large heading "Specifications". Below the heading, a section explains the development of the instruction set architecture (ISA) and related specifications. It mentions "Technical Working Groups" and work on GitHub. A "ISA Specification" section lists three ratified volumes: Volume 1 (Unprivileged Spec v. 20191213), Volume 2 (Privileged Spec v. 20211203), and recently ratified extension specifications. A note at the bottom states that past ratified releases include the term "ratified" in the release tag.

<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/>

The screenshot shows a web browser displaying the course homepage for CS 250: VLSI Systems Design at UC Berkeley. The page has a black header with yellow text. It includes the course title, professor (Prof. John Wawrzynek), and the spring 2016 schedule (Tuesday and Thursday, 12:30-2:00PM, 320 Soda). It also lists the section time (Tuesday 4:00-5:00PM, 212 Cory). Below the header, there are links for detailed course info, instructional machine setup, and discussion session slides. A course calendar table is shown, listing three lectures: First Class (Lecture 1) on Jan 19, Lecture 2 on Jan 21, and Lecture 3 on Jan 26. Each lecture entry includes a description and a link to the corresponding slides.

| Date | Type | Lecture | Description | Links |
|--------|-------------|-----------|---|---|
| 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 |
| 21-Jan | | Lecture 2 | Intro to Chisel. - Revised Slides , Original Slides | Lab 0 out : Chisel Tutorial |
| 26-Jan | | Lecture 3 | VLSI Introduction. Chip-level alternatives. IC Fabrication. Tool flow overview. RTL and other design representations. HLS. - Slides | Lab 1 out : Intro to SHA3 + VCS |

The screenshot shows a web browser displaying the ETH Zurich D-INFK website. The page has a dark header with the ETH Zurich logo and the text "Department of Computer Science". Below the header, there is a navigation bar with links for "ETH Zurich" and "D-INFK". The main content area is mostly blank or obscured by a large black rectangle.

ETH zürich

Department of Computer Science

*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:



WEBシラバス WEB syllabus 法政大学 2022

- [日本語 English](#)
- [MYクラス My class](#)
- [検索 Search](#)

< >

メニュー Menu

情報科学部 Faculty of Computer and Information Sciences

一覧に戻る Back to List

[削除 Delete](#) [追加 Add](#) [Del](#) [Add](#)

COT211KA-CS-212
コンピュータ構成と設計 Computer Organization and Design
李 亜民 Yamin LI

Finally, from our University of UEC:



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.)*

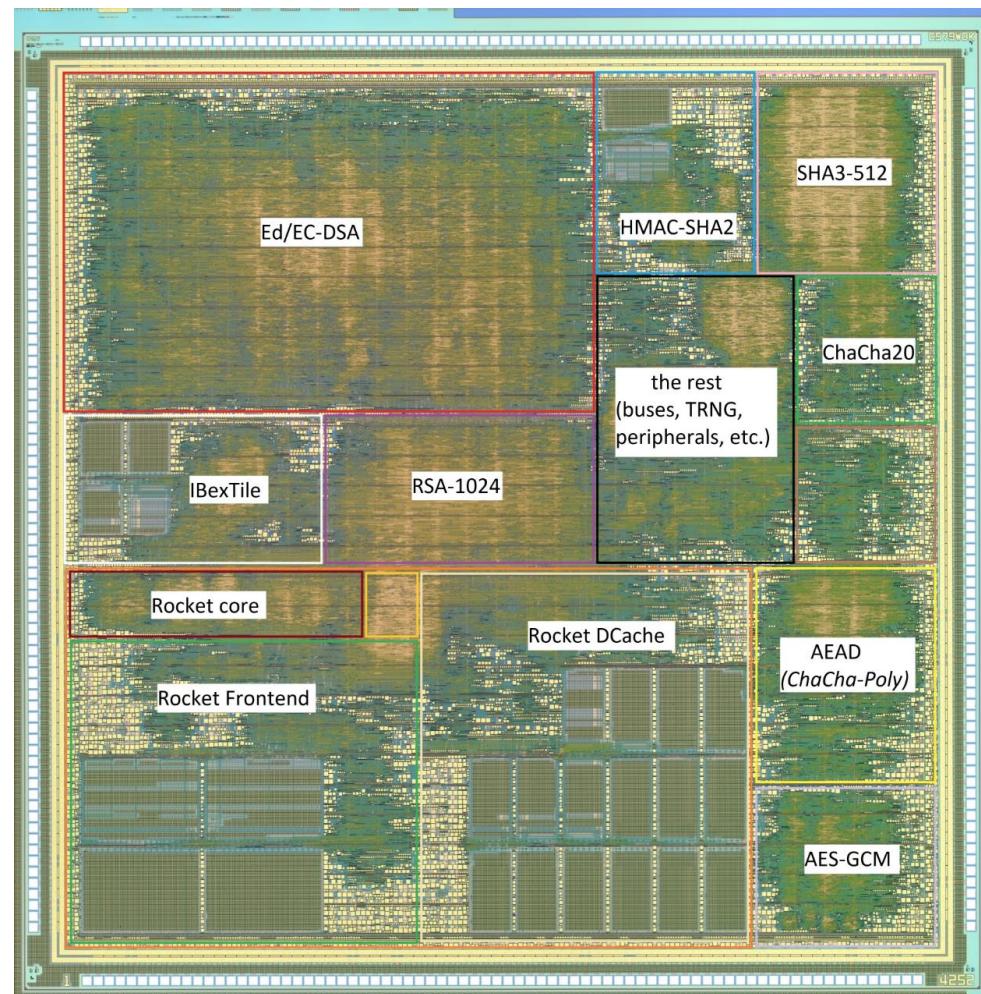
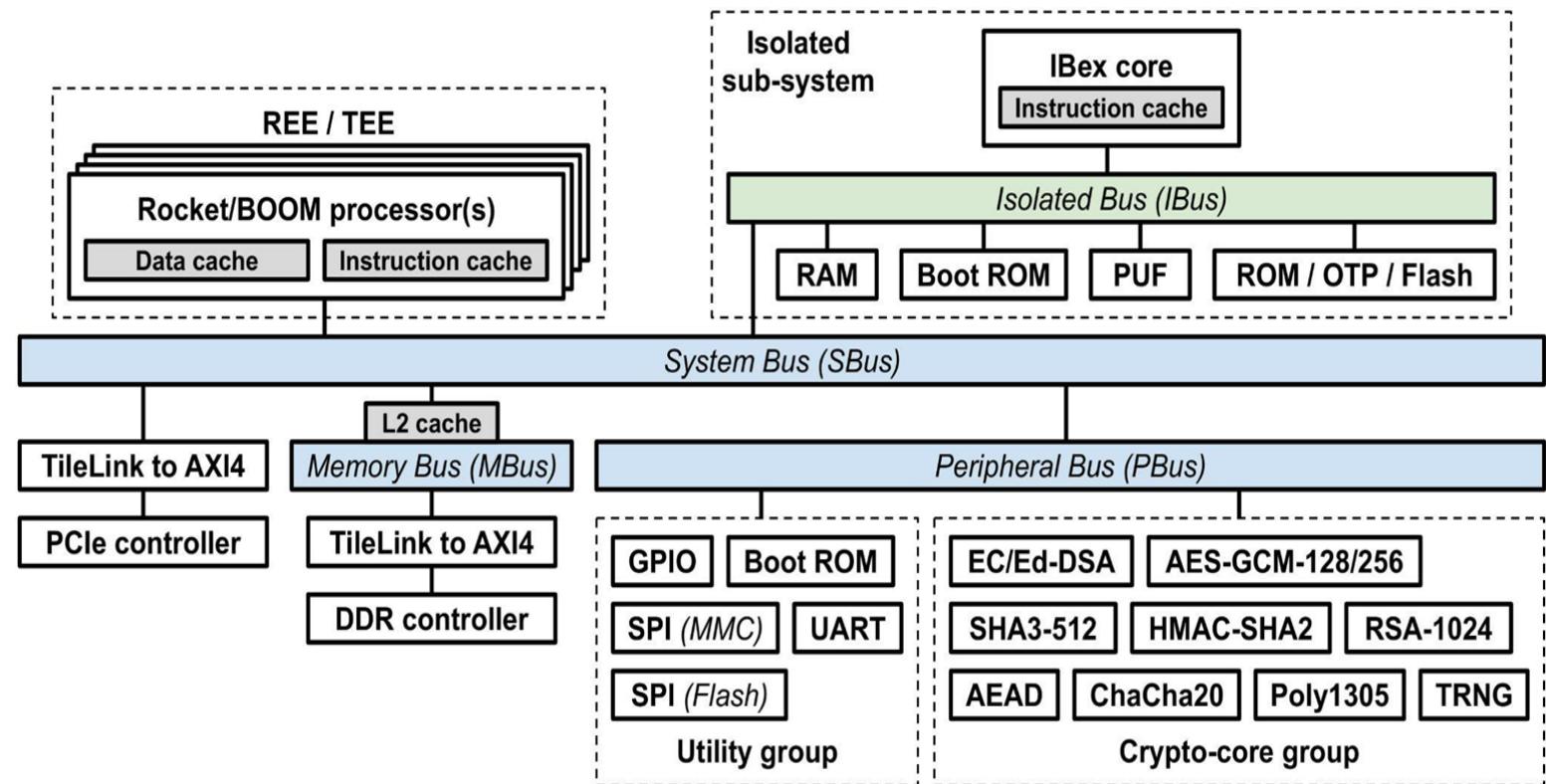
3. Classes content (4/6) Example: VLSI design class at UEC

| 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*)

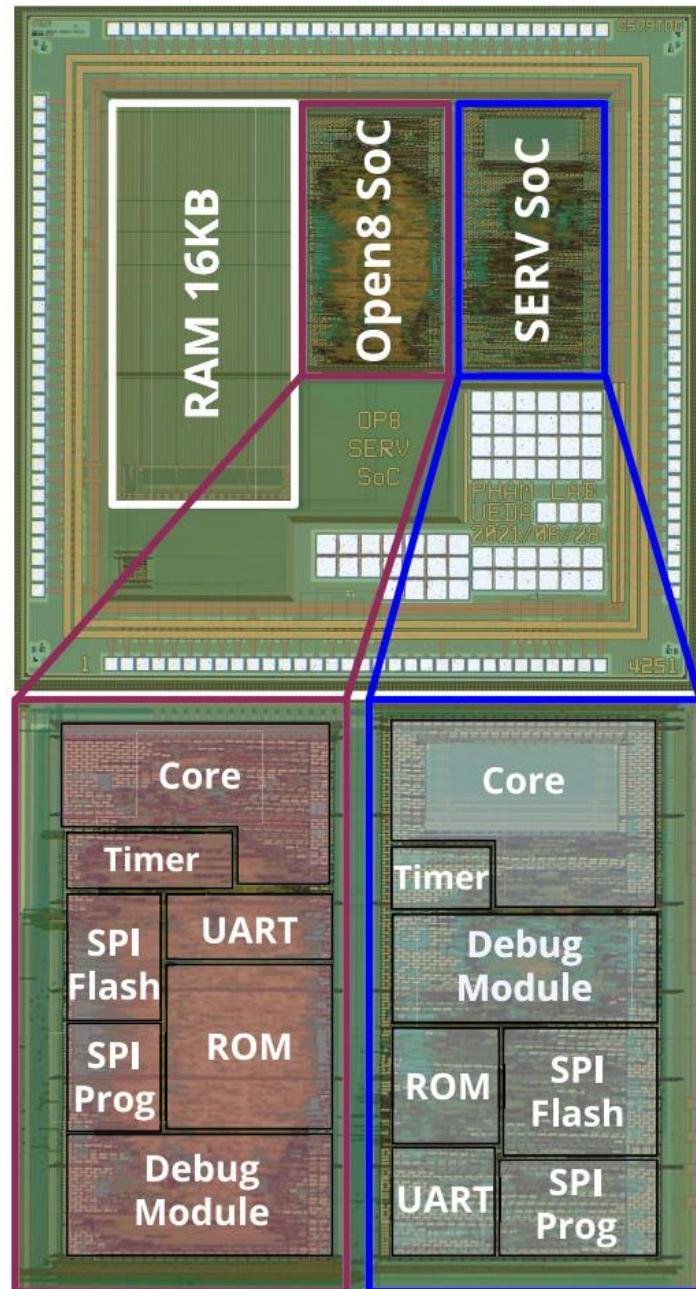
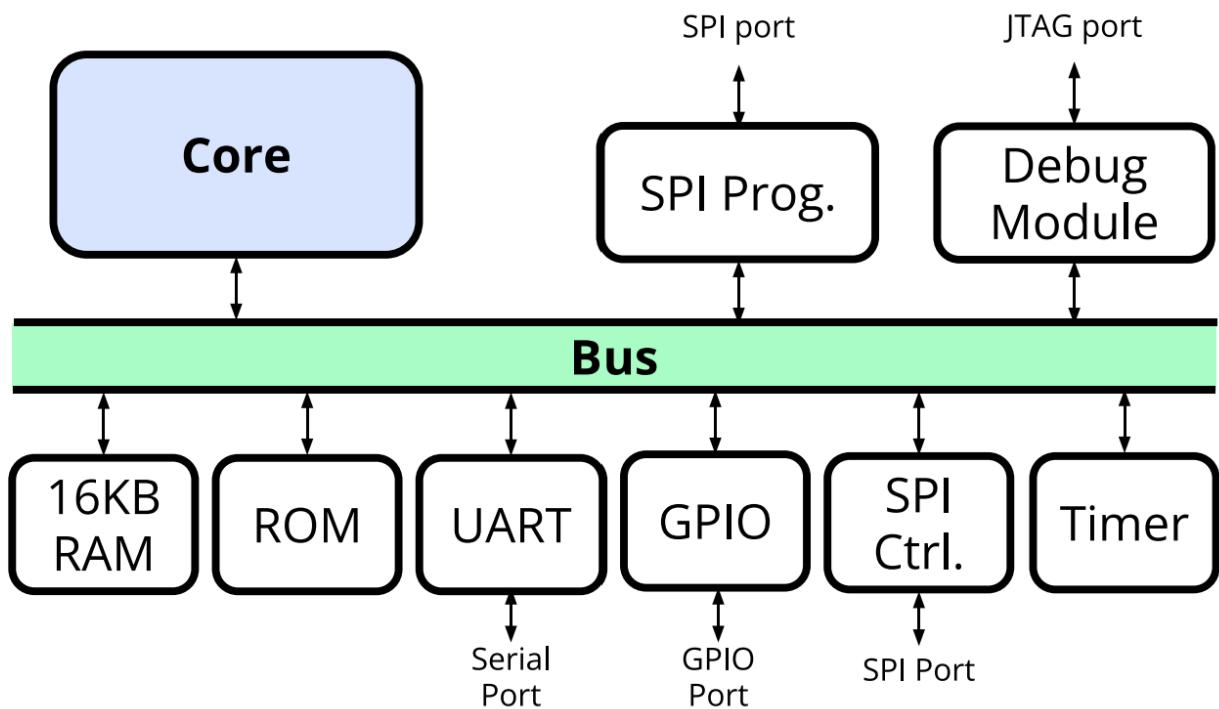
3. Classes content (5/6) Example: Crypto-SoC

- 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 achieve sub- μ W 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 | <ul style="list-style-type: none">• How to add your custom hardware (<i>Verilog or VHDL</i>) to an existing RISC-V computer system → generate a new → control/debug it in software |
| 2 | Hardware Design with Chisel/Scala | <ul style="list-style-type: none">• 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 | <ul style="list-style-type: none">• Learn 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) | <ul style="list-style-type: none">• 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 | <ul style="list-style-type: none">• 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