



# VLSI Lab @ UEC: RISC-V Research

Trong-Thuc HOANG and Cong-Kha PHAM



# OUTLINE

1. Introduction
2. Achievement highlight
3. What is RISC-V?
4. Working with RISC-V & open community
5. Main research @ UEC
6. RISC-V courses @ UEC



# OUTLINE

1. Introduction
2. Achievement highlight
3. What is RISC-V?
4. Working with RISC-V & open community
5. Main research @ UEC
6. RISC-V courses @ UEC

# 1. Introduction (1/4)

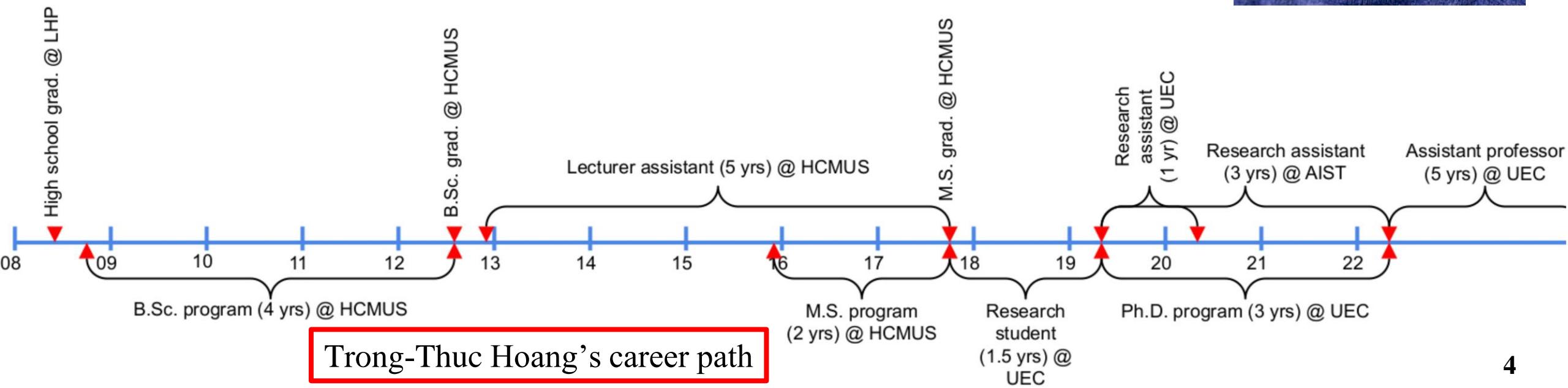


Trong-Thuc Hoang  
Assistant Professor (UEC)  
[hoangtt@uec.ac.jp](mailto:hoangtt@uec.ac.jp)

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



Cong-Kha Pham  
Professor (UEC)  
[phamck@uec.ac.jp](mailto:phamck@uec.ac.jp)



# 1. Introduction (2/4) University



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

<https://www.uec.ac.jp/>

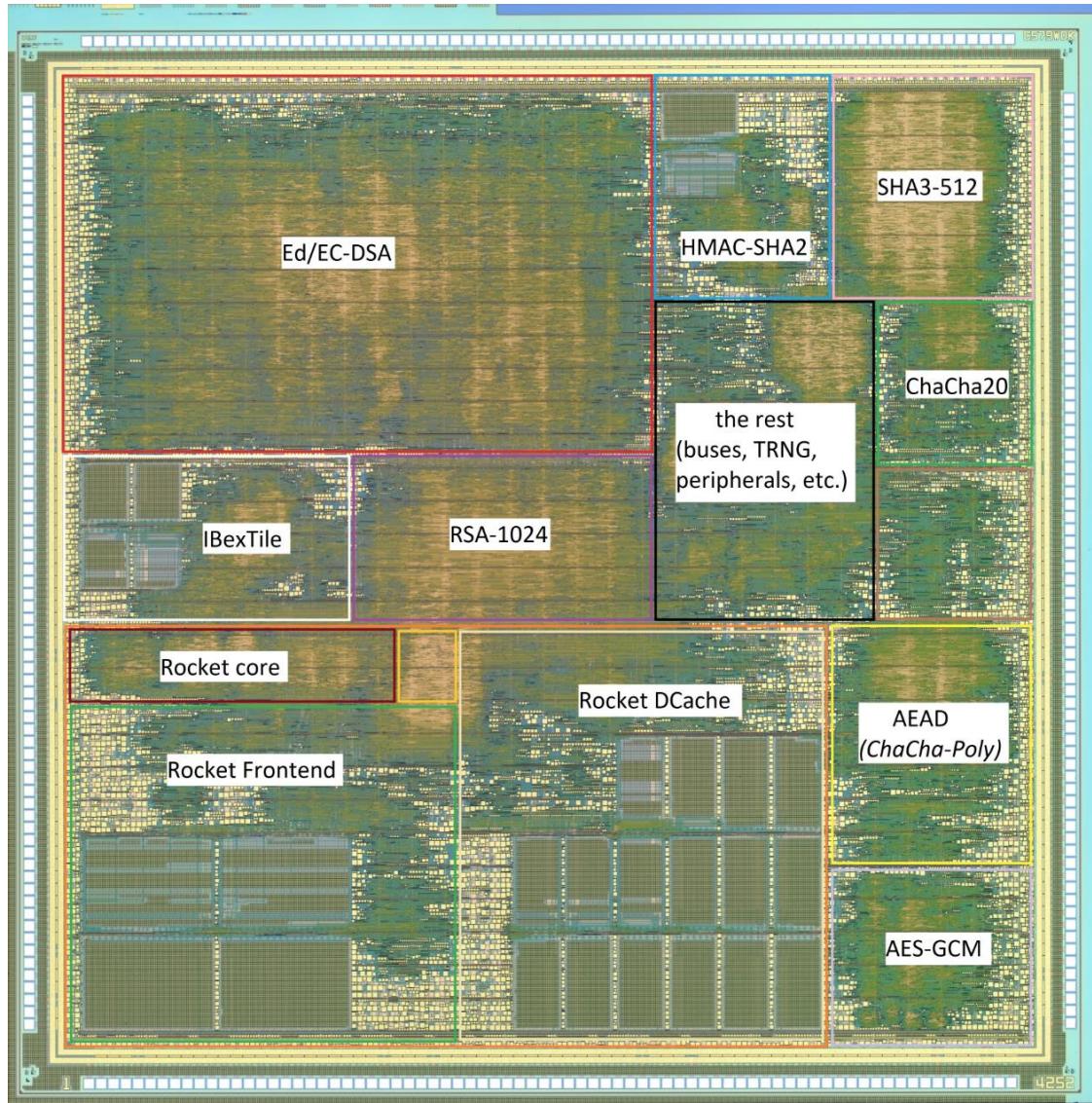


# 1. Introduction (3/4) Laboratory



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

**Pham Laboratory**  
Integrated circuit design laboratory



# 1. Introduction (4/4) Member



**Member**  
*(as in 2022)*

- Ph.D. students: 4
  - Master students: 6
  - Bachelor students: 3
  - Researchers: 3
- 

Total: 16

**From**

- Japan: 8
- Vietnam: 6
- Columbia: 2



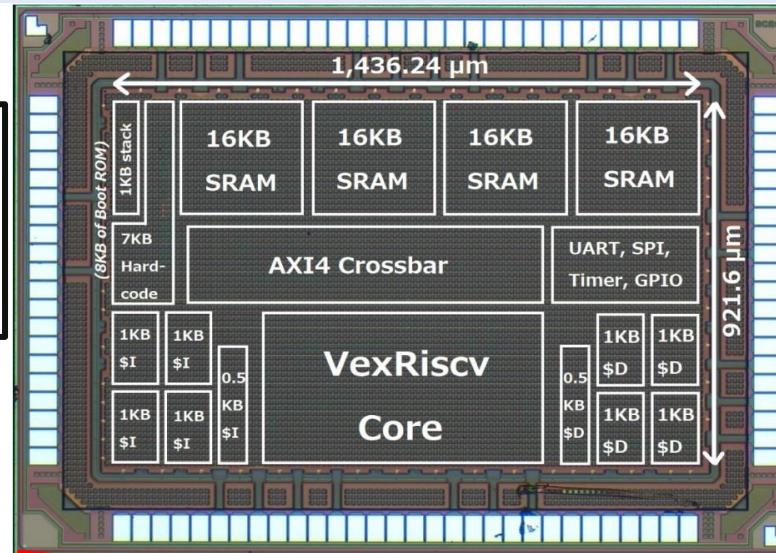
# OUTLINE

1. Introduction
2. Achievement highlight
3. What is RISC-V?
4. Working with RISC-V & open community
5. Main research @ UEC
6. RISC-V courses @ UEC

## 2. Achievement highlight (1/8) Crypto-SoC timeline

Crypto-SoC

VexRiscv32( $\times 1$ )  
SOTB-65nm  
 $2.0 \times 1.5\text{-mm}^2$



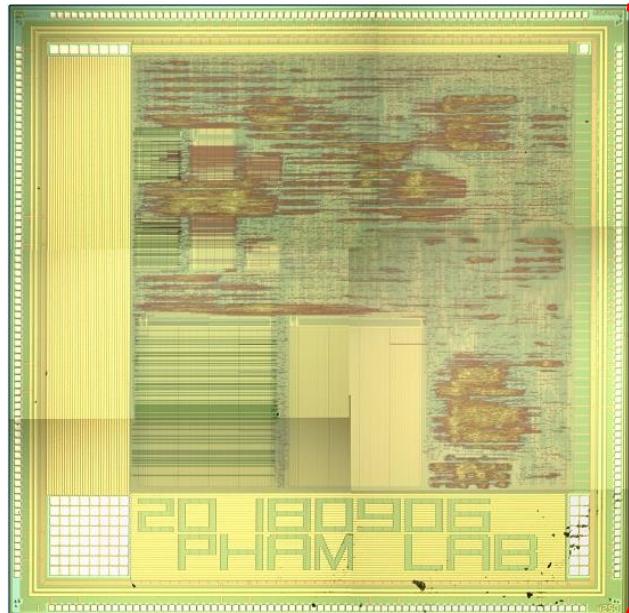
2018

2019

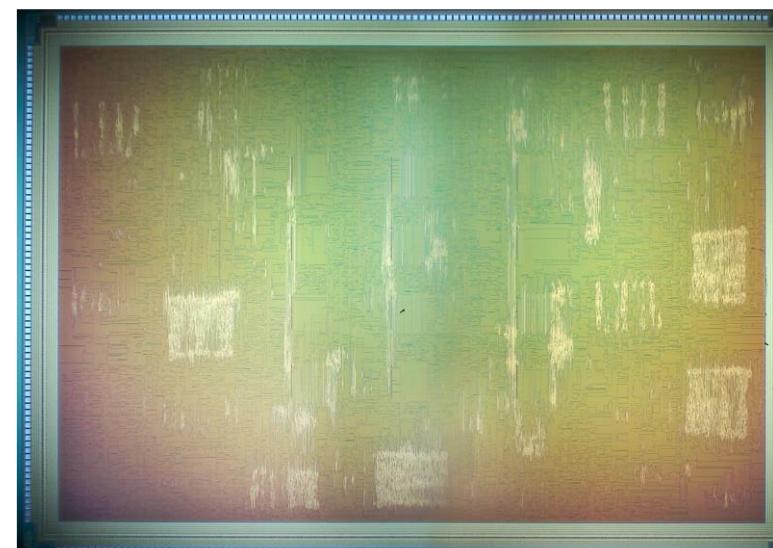
2020

09

10

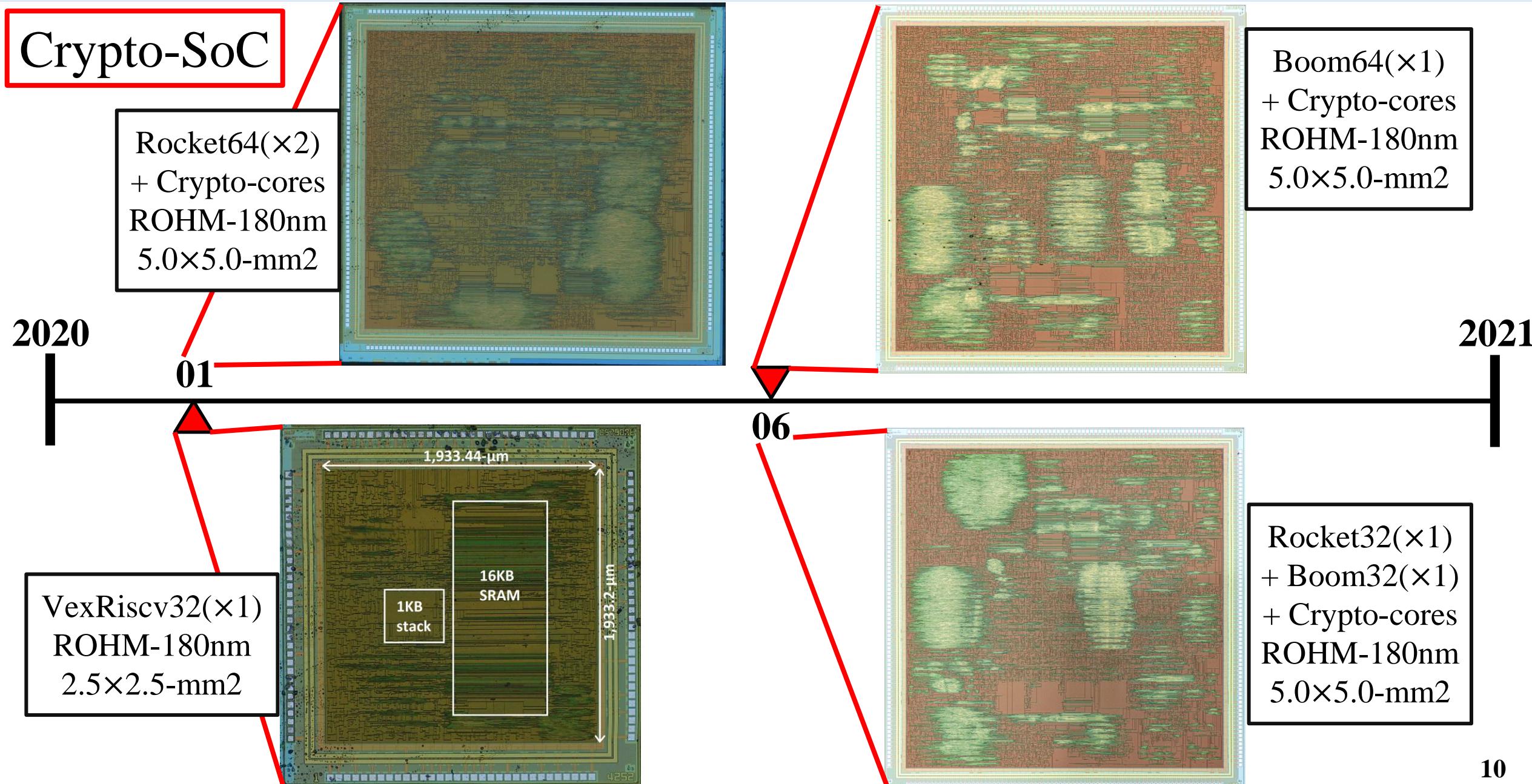


Rocket64( $\times 1$ )  
ROHM-180nm  
 $5.0 \times 5.0\text{-mm}^2$



Rocket64( $\times 4$ )  
ROHM-180nm  
 $5.0 \times 7.5\text{-mm}^2$

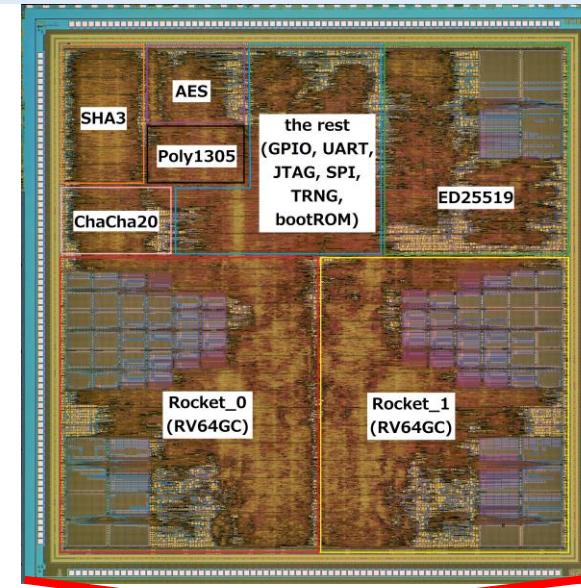
## 2. Achievement highlight (2/8) Crypto-SoC timeline



## 2. Achievement highlight (3/8) Crypto-SoC timeline

### Crypto-SoC

Rocket64( $\times 2$ )  
+ Crypto-cores+TRNG  
+ Secure boot  
ROHM-180nm, 5.0×5.0-mm<sup>2</sup>



Rocket64( $\times 2$ )  
+ Crypto-cores  
+ Rnd Freq  
ROHM-180nm  
5.0×5.0-mm<sup>2</sup>

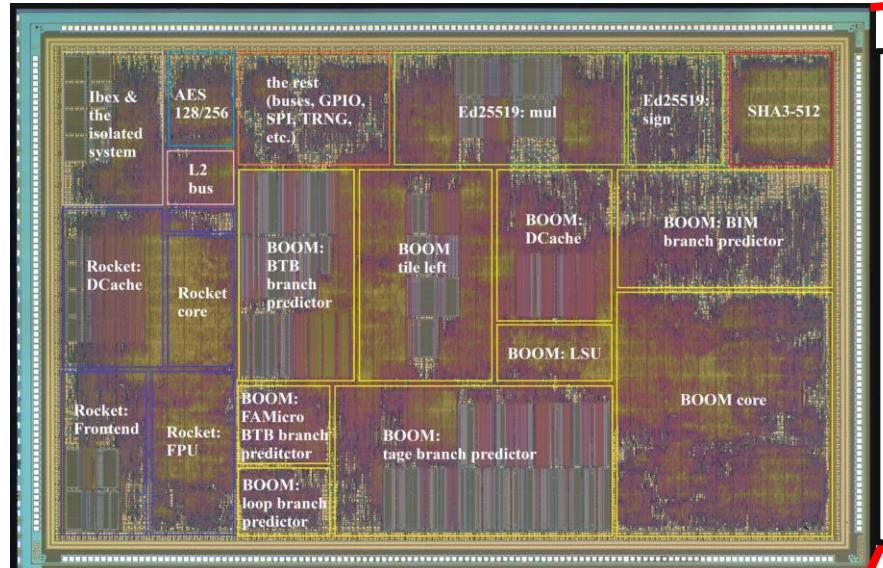
2021

02

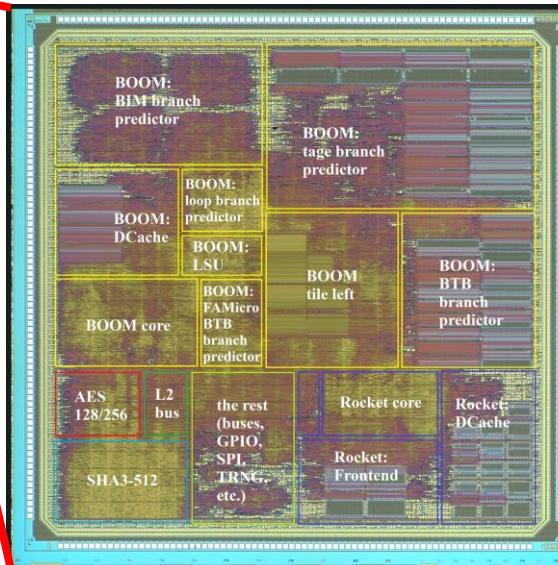
06

09

2022



Rocket64( $\times 1$ )  
+ Boom64( $\times 1$ )  
+ Crypto-cores  
+ TRNG  
+ Secure boot  
ROHM-180nm  
5.0×7.5-mm<sup>2</sup>



Rocket32( $\times 1$ )  
+ Boom32( $\times 1$ )  
+ Crypto-cores  
+ TRNG  
+ Secure boot  
ROHM-180nm  
5.0×5.0-mm<sup>2</sup>

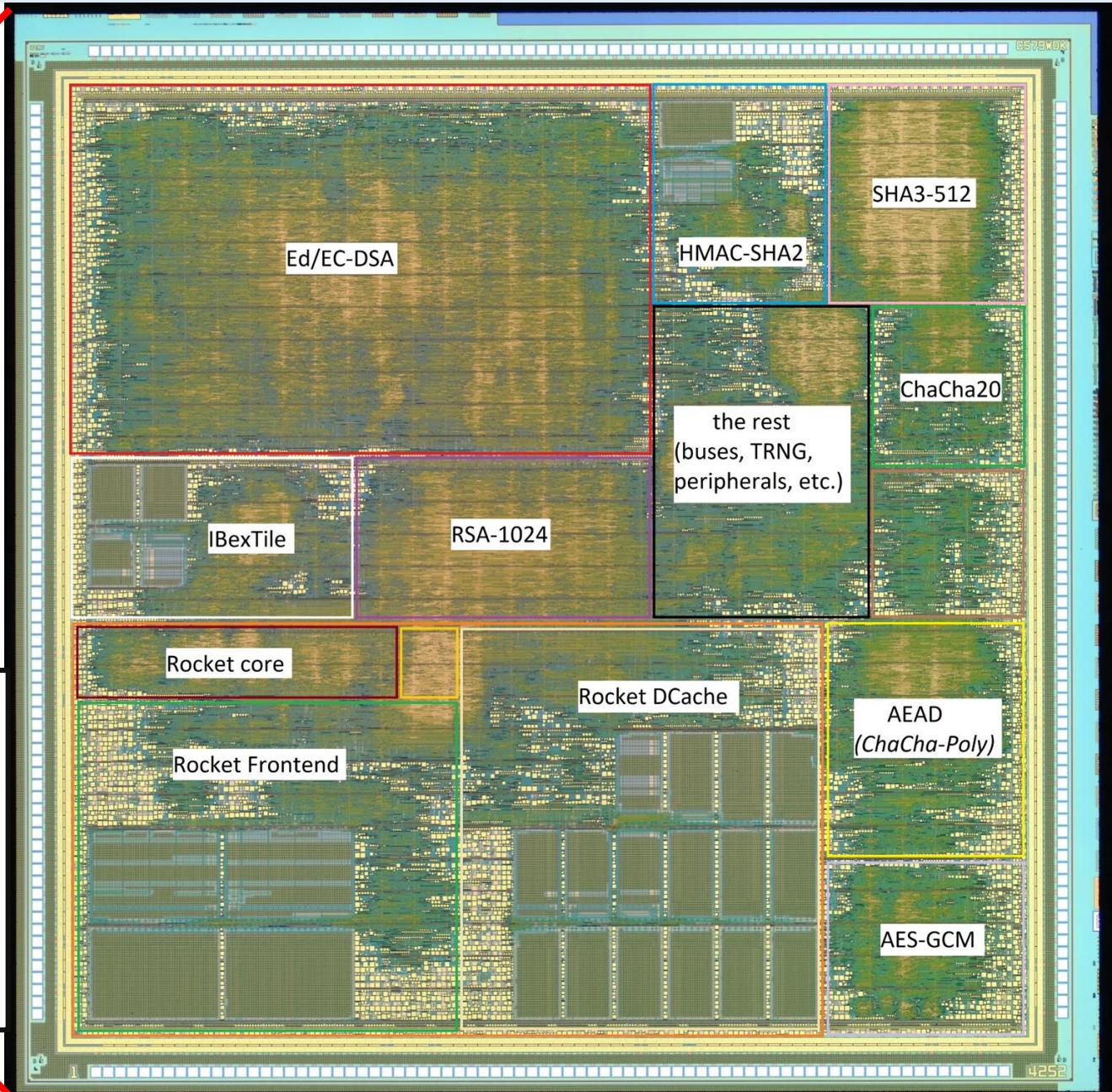
## 2. Achievement highlight (4/8) Crypto-SoC timeline

Crypto-SoC

2022

02

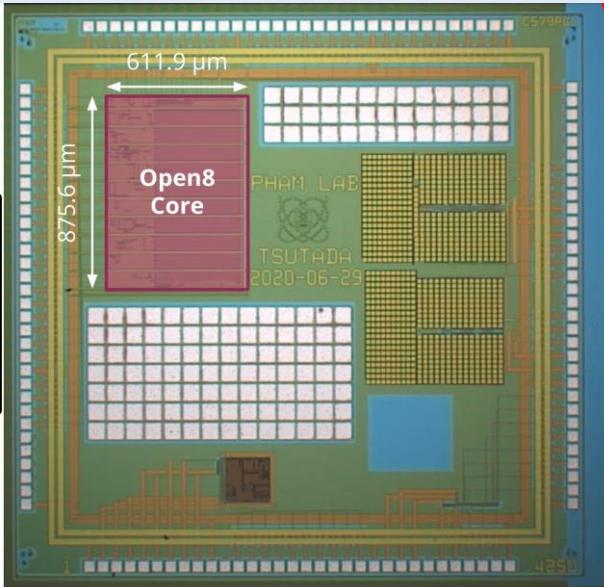
Rocket32( $\times 1$ )  
+ TLS-1.3 Crypto-cores  
+ TRNG  
+ Secure boot  
ROHM-180nm  
 $5.0 \times 5.0\text{-mm}^2$



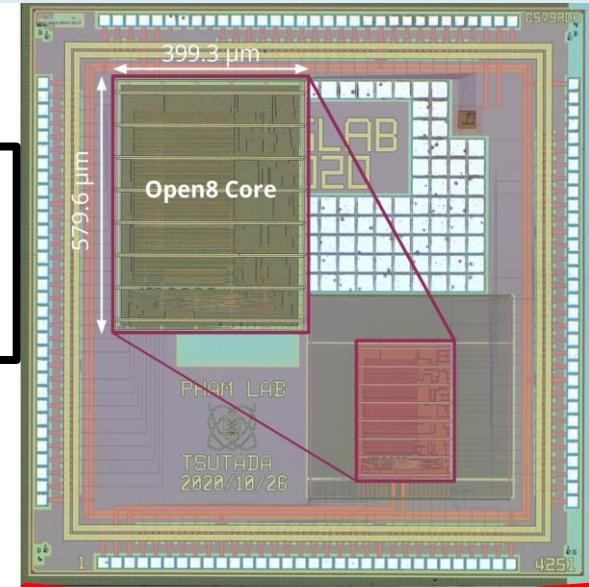
## 2. Achievement highlight (5/8) ULP-SoC timeline

ULP-SoC

Open8: 8-bit  
ROHM-180nm  
 $2.5 \times 2.5\text{-mm}^2$



Open8: 8-bit  
ROHM-180nm  
 $2.5 \times 2.5\text{-mm}^2$



2020

2021

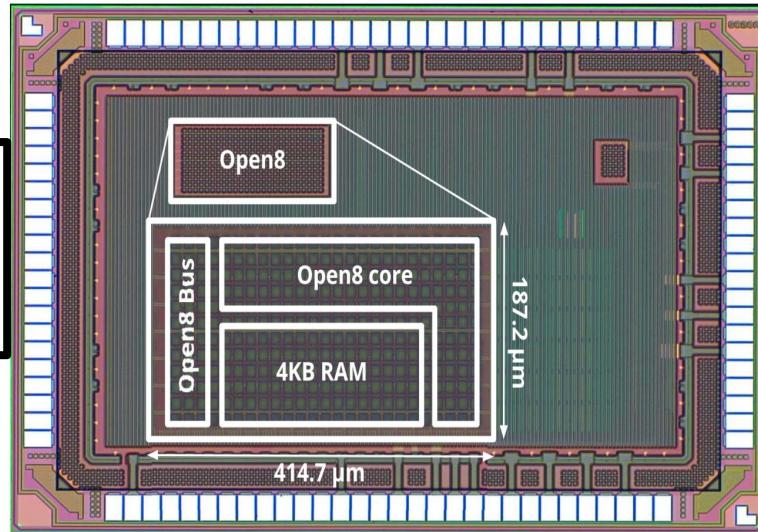
06

08

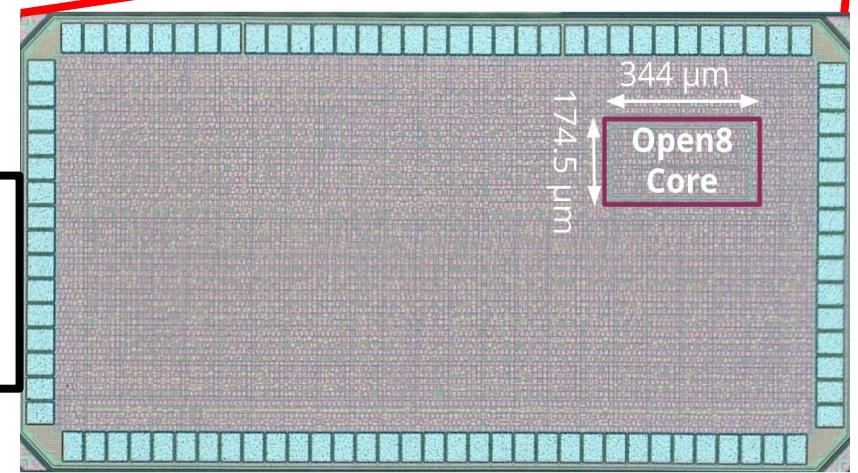
10

12

Open8: 8-bit  
SOTB-65nm  
 $1.5 \times 2.0\text{-mm}^2$



Open8: 8-bit  
TSMC-65nm  
 $1.0 \times 2.0\text{-mm}^2$

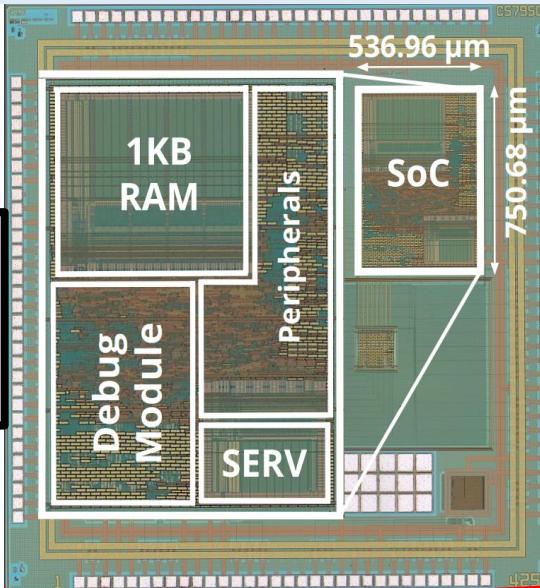


13

## 2. Achievement highlight (6/8) ULP-SoC timeline

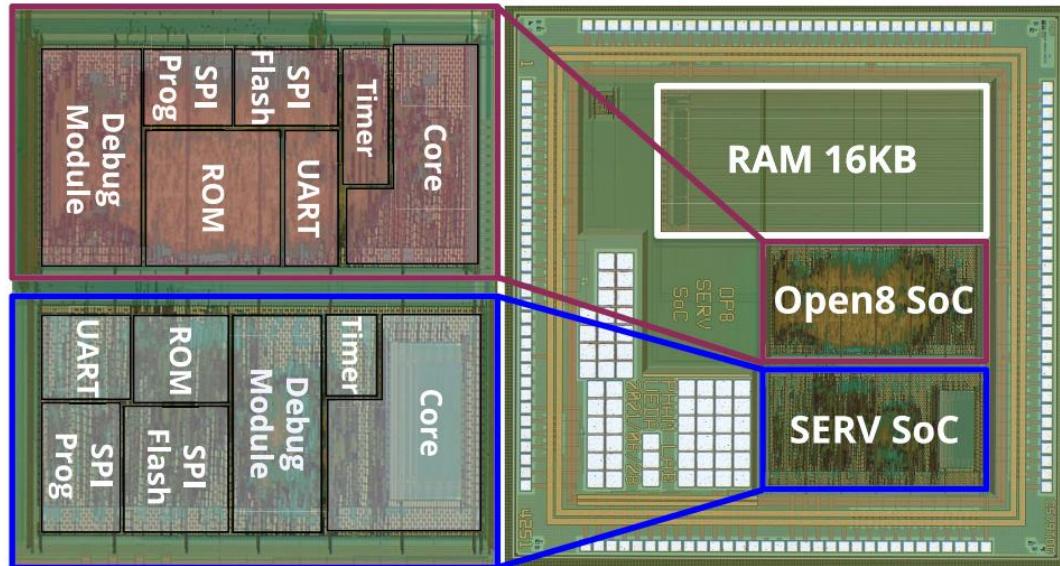
ULP-SoC

SERV: 32-bit  
ROHM-180nm  
 $2.5 \times 2.5\text{-mm}^2$

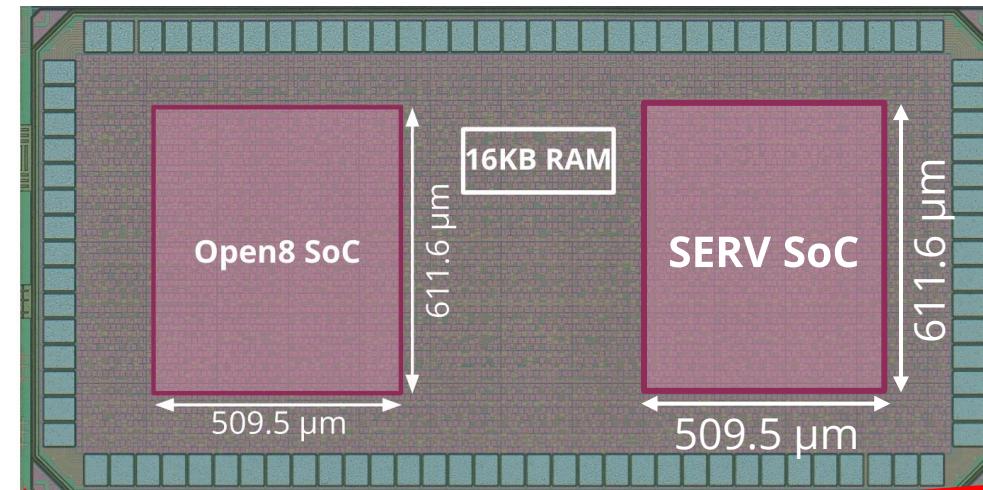


2021

02



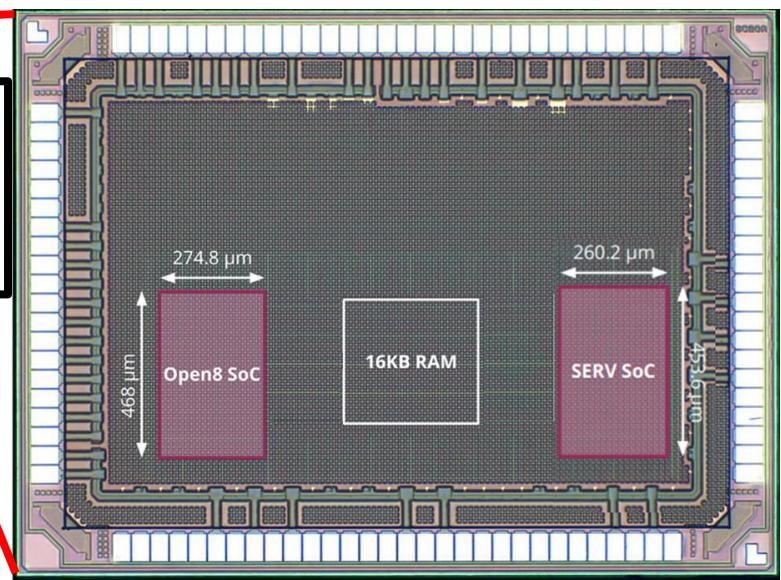
06 07



2022

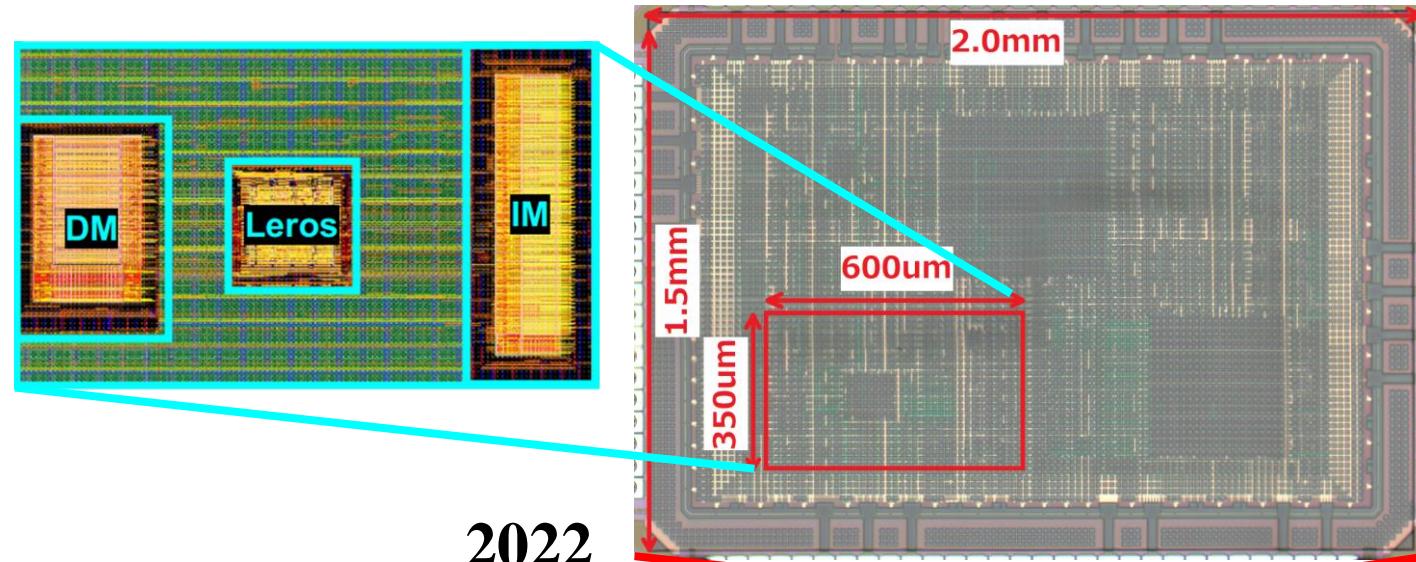
Open8+SERV  
SOTB-65nm  
 $1.5 \times 2.0\text{-mm}^2$

Open8+SERV  
ROHM-180nm  
 $2.5 \times 2.5\text{-mm}^2$



## 2. Achievement highlight (7/8) ULP-SoC timeline

ULP-SoC



Leros: 16-bit  
SOTB-65nm  
 $1.5 \times 2.0\text{-mm}^2$

2022

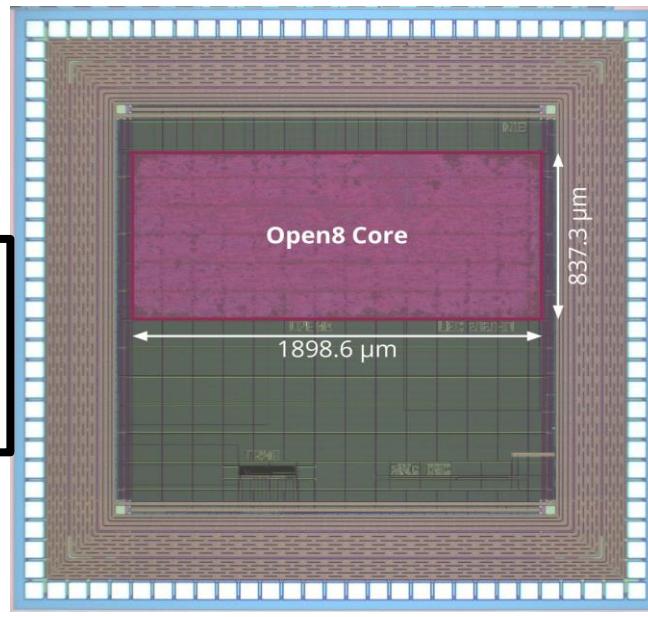
▼  
06

11

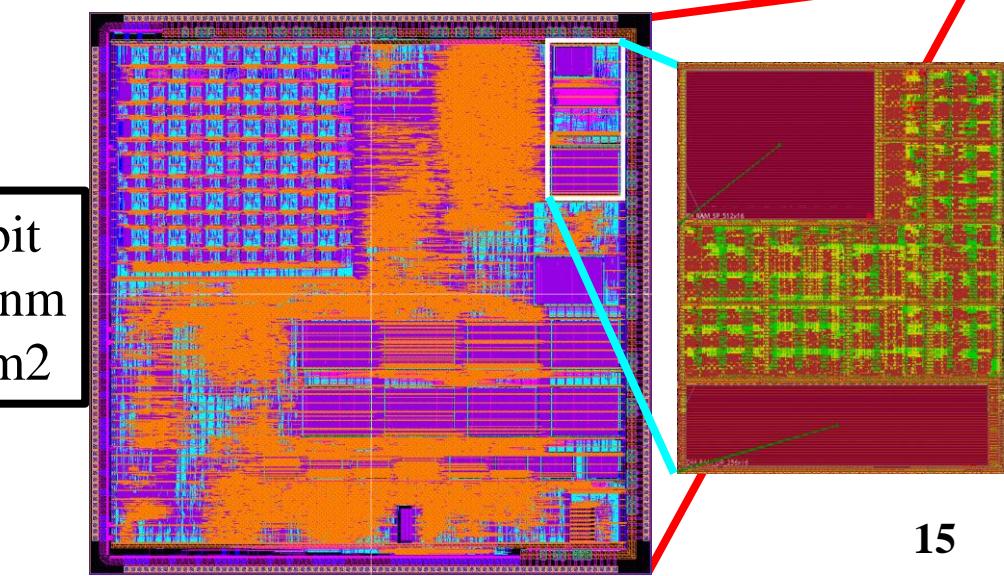
06

03

Open8: 8-bit  
LAPIS-220nm  
 $2.9 \times 2.9\text{-mm}^2$



Leros: 16-bit  
ROHM-180nm  
 $2.5 \times 2.5\text{-mm}^2$



15

## 2. Achievement highlight (8/8) Summary

Since 2019 to now: **4 years**

A core team of **five members**

**20+** chips: FullChip designs

**Linux** bootable SoC

Two big *past* projects: **Crypto-SoC** and **ULP-SoC** (*and other small projects*)

All these achievements are thanks to **RISC-V**.

**16** Journals and **11** Conferences



# OUTLINE

1. Introduction
2. Achievement highlight
3. What is RISC-V?
4. Working with RISC-V & open community
5. Main research @ UEC
6. RISC-V courses @ UEC

### 3. What is RISC-V (1/10) CISC vs. RISC

#### CISC

(Complex Instruction Set Computer)

- 1) Emphasis on hardware
- 2) Includes multi-clock complex instructions
- 3) Memory-to-memory mindset
- 4) Small code sizes, high cycles/s
- 5) Transistors used for storing complex instructions

#### RISC

(Reduced Instruction Set Computer)

- 1) Emphasis on software
- 2) Single-clock reduced instructions only
- 3) Register-to-register mindset
- 4) Large code sizes, low cycles/s
- 5) Spends more transistors, and most of them are used for storing data

$$\text{Performance} = \frac{\text{time}}{\text{program}} = \frac{\text{time}}{\text{cycle}} \times \frac{\text{cycle}}{\text{instruction}} \times \frac{\text{instruction}}{\text{program}}$$

RISC-V simply means RISC architecture version *five*

Nowadays, almost all processors in the market are RISCs.

*Economic reason:* memory price is way down ↓↓↓

### 3. What is RISC-V (2/10) 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

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

### 3. What is RISC-V (3/10) ISA?

ISA means Instruction Set Architecture.

It is the layer between software and hardware developers.

**Software tools:** assembler, compilers, debugger, linker, etc.

---

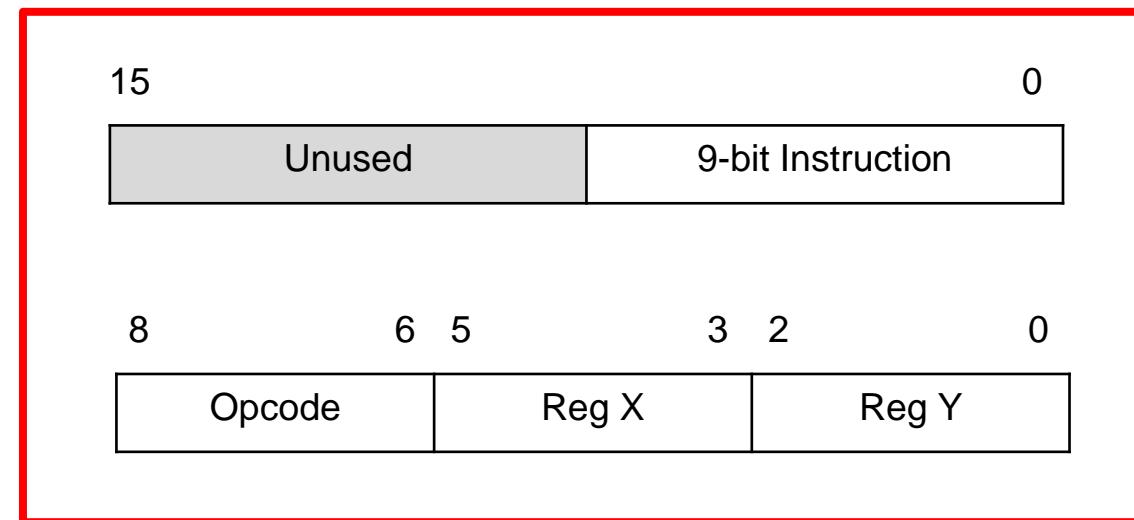
**ISA:** the interface between software & hardware architects

---

**Processor:** ALU, FPU, registers, CSRs, branch predictor, caches, etc.

**ISA has to define all these kinds of stuff:**

- 1) How many instructions, and which is which?
- 2) In an instruction, what field means what?
- 3) Addressing & data-path (8/16/32/64/128-bit)?
- 4) What is supported and what is not?
- 5) etc.



### 3. What is RISC-V (4/10) RISC-V ISA

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

*(modular architecture helps fine-tune the performance based on the developer's needs [1])*

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	Version	Status
L	2.0	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

### 3. What is RISC-V (5/10) RISC-V OS stack

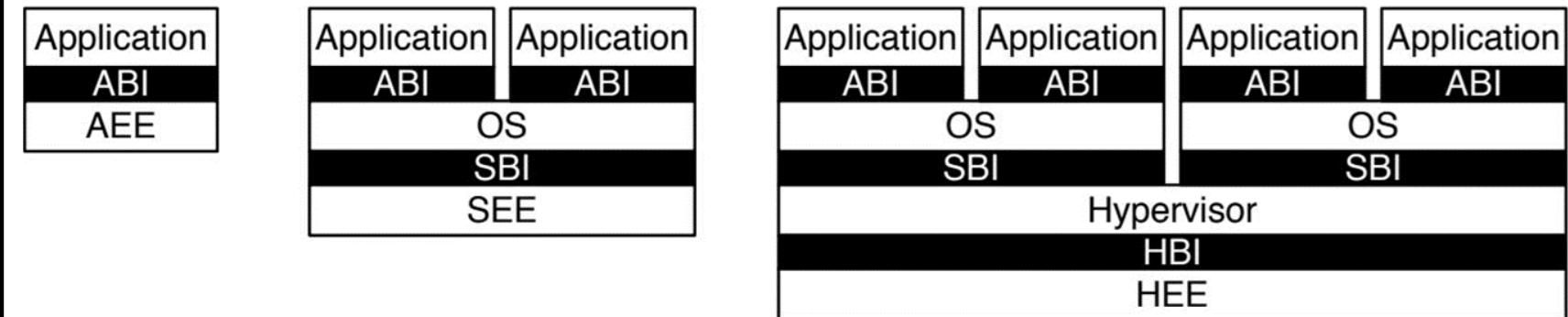
To support an Operating System (OS), the ISA has to support the OS stack or the **M-/S-/U-mode**.

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:

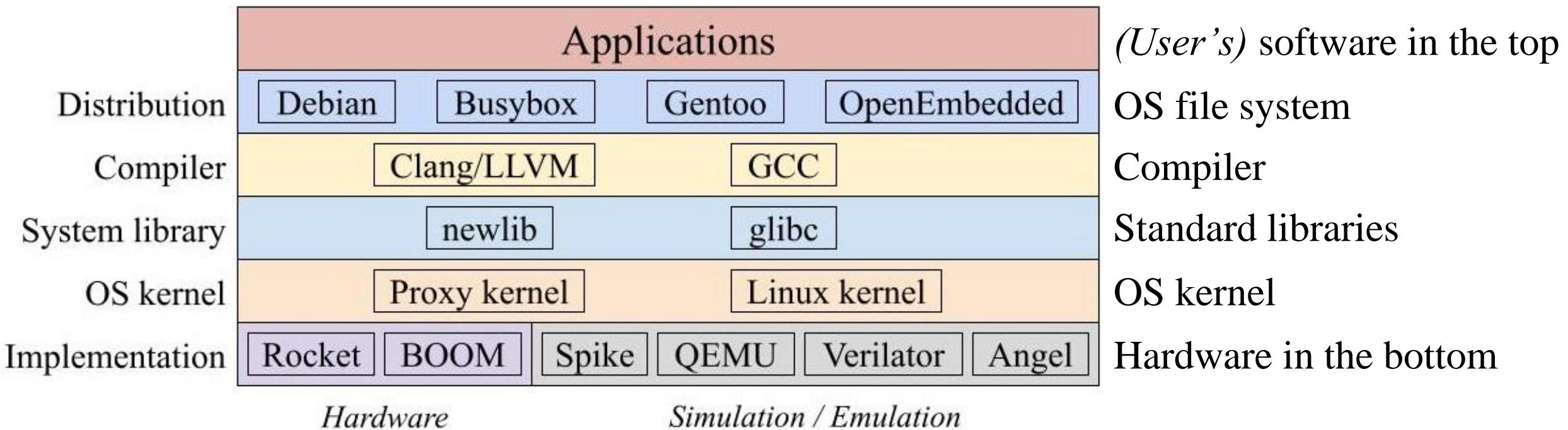


**RISC-V ISA** not only supports the OS stack, but also provides a **privileged architecture** [2].

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

### 3. What is RISC-V (6/10) RISC-V ecosystem

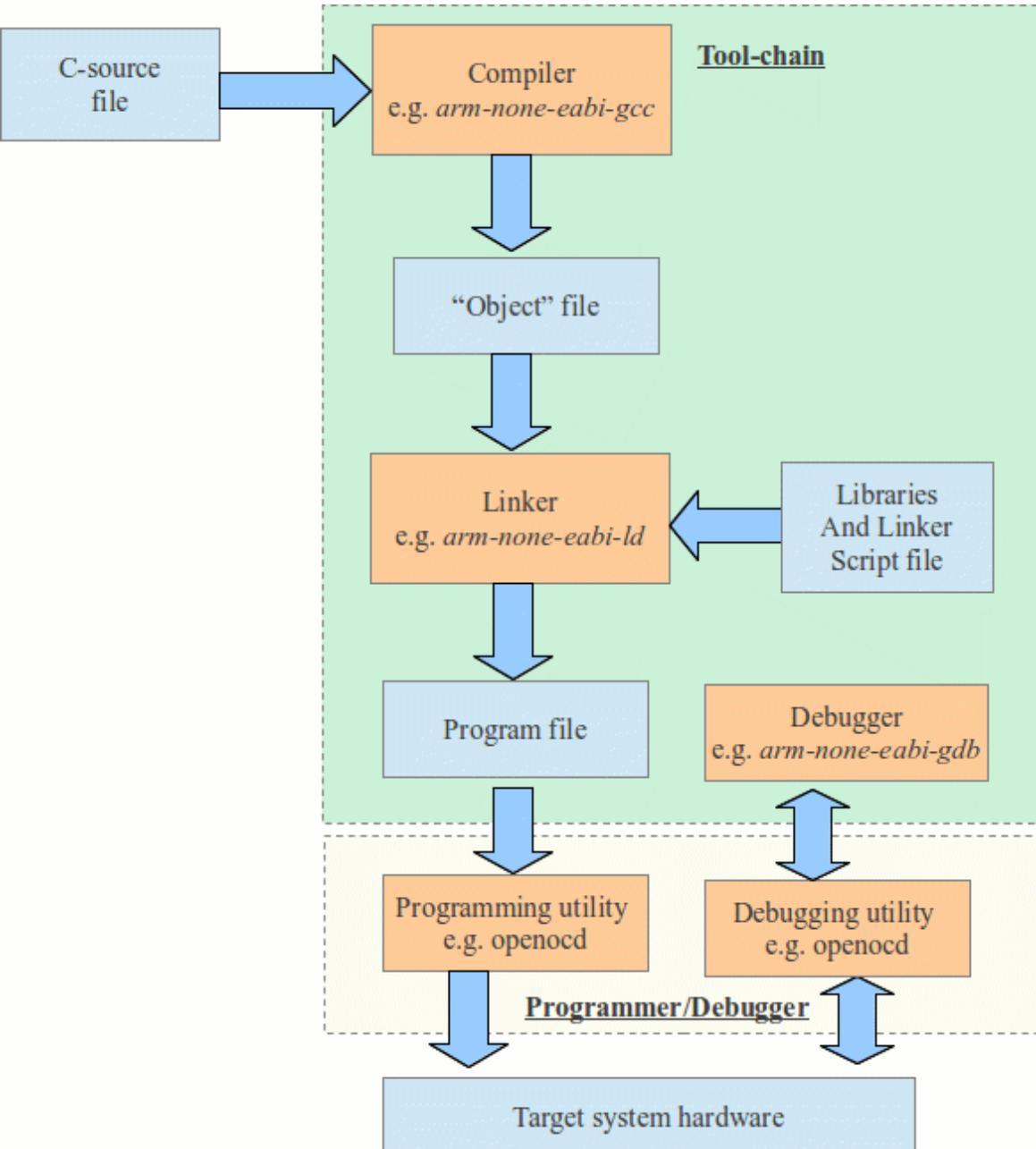
#### RISC-V toolchain and its ecosystem



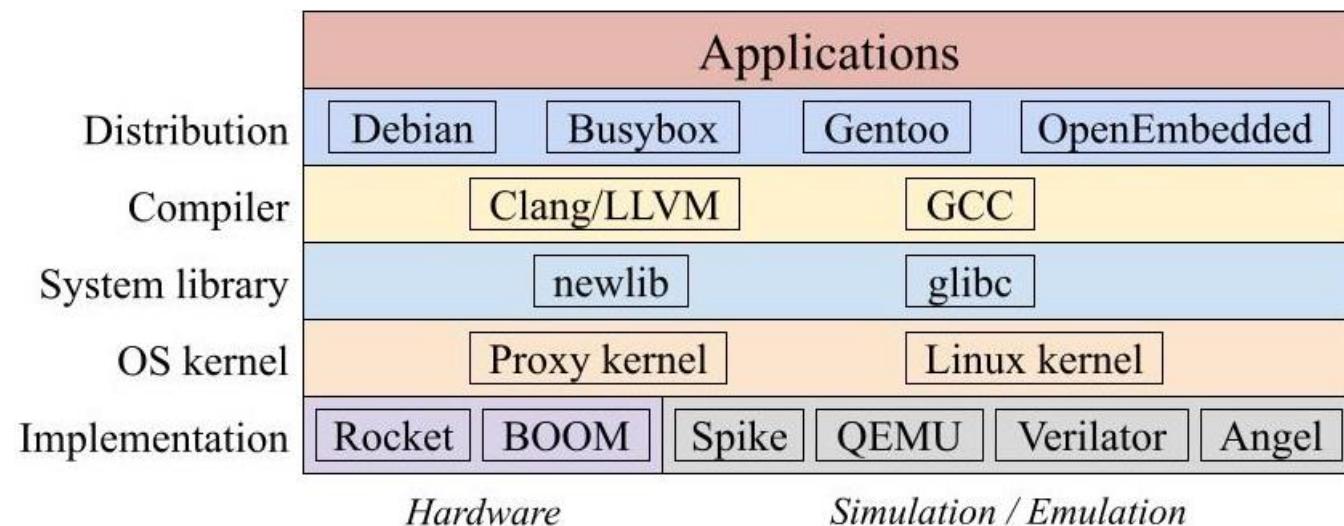
#### Top-down explanation:

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.

### 3. What is RISC-V (7/10) RISC-V toolchain



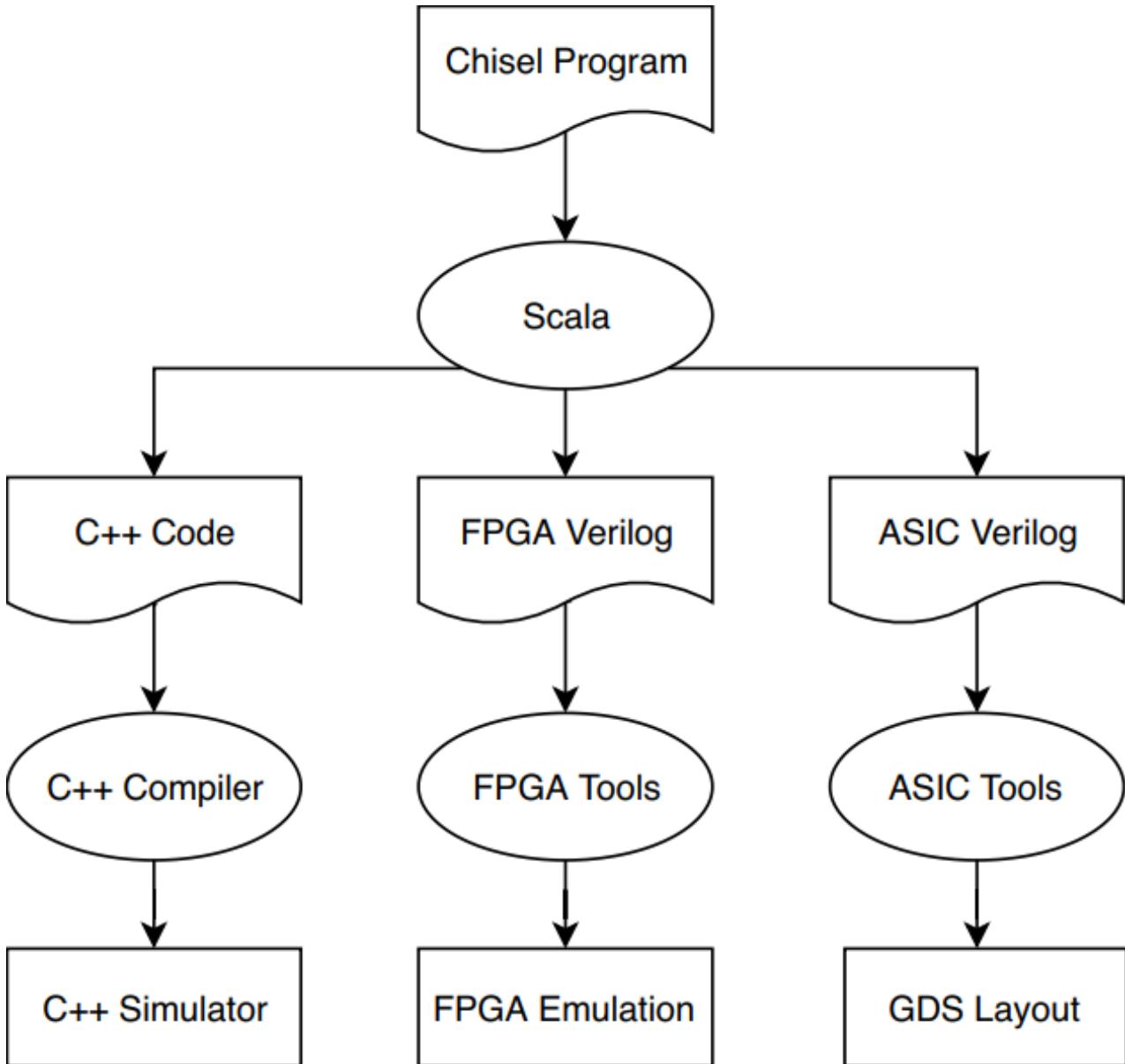
### RISC-V toolchain and its ecosystem



### Three most important tools

- **GCC:** (*cross C compiler*) makes a C code into assembly code
- **LD:** (*linker*) links standard libraries into the build; also links between multiple C files
- **GDB:** (*debugger*) debug the hardware/simulator/emulator

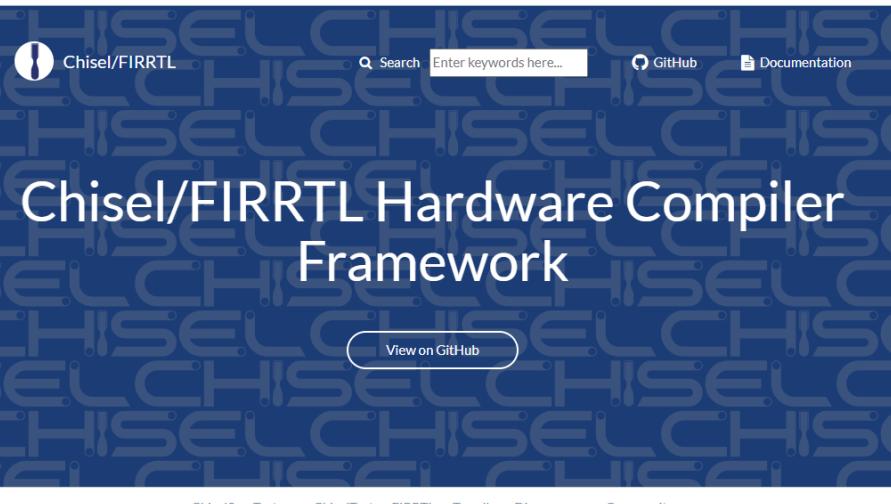
### 3. What is RISC-V (8/10) CHISEL: a new way to code



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  
    1<sup>st</sup> arrow: Scala compiler named SBT  
    2<sup>nd</sup> arrow: executing Java  
    3<sup>rd</sup> arrow: FIRRTL compiler

### 3. What is RISC-V (9/10) CHISEL: a new way to code



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

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

ucb-bar / chisel-tutorial Public

Code Issues 29 Pull requests 3 Actions Projects Wiki Security

release 20 branches 12 tags Go file Add file Code

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)

ucb-bar / chisel-release Public

Code Issues 2 Pull requests 3 Actions Projects Wiki Security Insights

master 39 branches 91 tags Go to file Add file Code

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>

### 3. What is RISC-V (10/10) RISC-V key factors

#### 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.

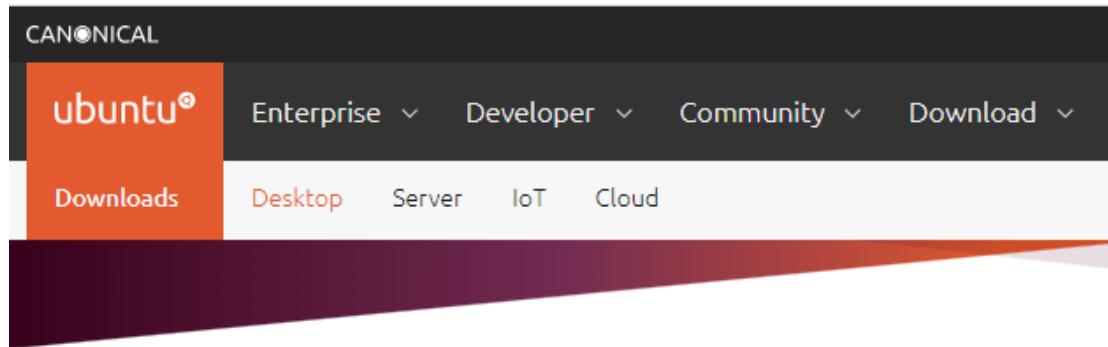


# OUTLINE

1. Introduction
2. Achievement highlight
3. What is RISC-V?
4. Working with RISC-V & open community
5. Main research @ UEC
6. RISC-V courses @ UEC

# 4. Working with RISC-V (1/21) Unix machine

First of all, you'll need an Ubuntu machine (recommend version 20.04-LTS)



## Download Ubuntu Desktop

The open-source desktop operating system that powers millions of PCs and laptops around the world. Find out more about Ubuntu's features and how we support developers and organisations below.

[Ubuntu Desktop homepage](#)

[Visit the Ubuntu Desktop blog ›](#)

You can follow my guide to set up a new Ubuntu machine for working with RISC-V:  
<https://thuchoang90.github.io/tutorial/2022/09/30/Fresh-Ubuntu-setup>

Thuctorial

HOME TUTORIAL PROJECT TAGS ABOUT

Light Dark

## Fresh Ubuntu machine setup for working with RISC-V

Sep 30, 2022 | About 13 mins

#RISC-V

### I. Dependencies & Proxy

To make `vi` more comfortable:

`vi`

**SHELL**

**Content**

- I. Dependencies & Proxy
- II. RISC-V Tools
  - II. a) Github
  - II. b) Scala & sbt:
  - II. c) Verilator
  - II. d) QEMU
  - II. e) Idea IntelliJ
  - II. f) Eclipse
  - II. g) OpenOCD
  - II. h) Vivado
  - II. i) Quartus
- III. RISC-V Toolchain
  - III. a) Git clone
  - III. b) Config & Make

# 4. Working with RISC-V (2/21) RISC-V toolchain

The next step, preparing the RISC-V GNU toolchain, is the essential part.

Official GitHub link for compiling from source:

<https://github.com/riscv-collab/riscv-gnu-toolchain>

A screenshot of a GitHub repository page. At the top, there's a navigation bar with links for Product, Solutions, Open Source, Pricing, and a Search bar. Below the navigation bar, the repository name 'riscv-collab / riscv-gnu-toolchain' is shown as public. There are buttons for Notifications, Code, Issues (45), Pull requests (5), Actions, Projects, Wiki, and Security. Below these, there are dropdowns for master branch, 5 branches, and 43 tags, along with 'Go to file' and 'Code' buttons. The main content area shows a file named 'README.md'. A section titled 'RISC-V GNU Compiler Toolchain' describes it as a cross-compiler for RISC-V C and C++ that supports two build modes: generic ELF/Newlib and Linux-ELF/glibc.

README.md

## RISC-V GNU Compiler Toolchain

This is the RISC-V C and C++ cross-compiler. It supports two build modes: a generic ELF/Newlib toolchain and a more sophisticated Linux-ELF/glibc toolchain.

You can also download a pre-built toolchain on the SiFive webpage:

[\(however, I recommend compiling from source\)](https://www.sifive.com/software)

Prebuilt RISC-V  
GCC Toolchain and Emulator

Save time by using one of our prebuilt toolchains which contain all the tools necessary to compile and debug programs on SiFive products. No hardware, no problem as the QEMU emulator packages can be used to test software applications without hardware. Our toolchain and emulator distributions have been carefully packaged to support both 32-bit & 64-bit ISAs. All the available packages are built using our [Freedom Tools GitHub repository](#), where previous released versions are available.

GNU Embedded Toolchain –  
v2020.12.8

OpenOCD – v2020.12.1

Windows

Windows

macOS

macOS

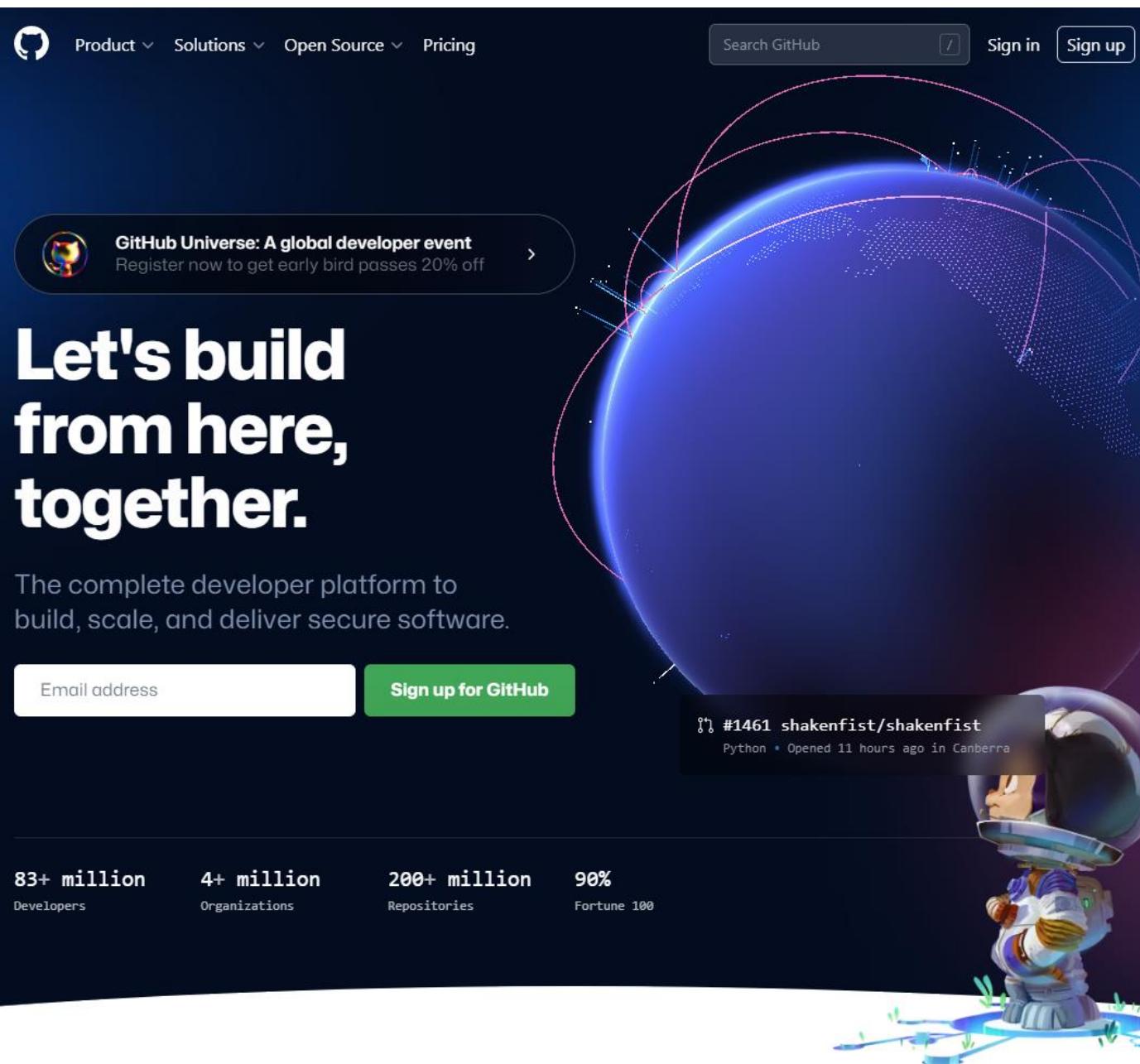
CentOS

CentOS

Ubuntu

Ubuntu

# 4. Working with RISC-V (3/21) Github



When working with the RISC-V open-source community, you'll be using GitHub a lot.

So it's better to set up a GitHub account for personal use.

<https://github.com/>

# 4. Working with RISC-V (4/21) Hardware simulate/emulate

## Verilator

- A simulator
- Simulate a Verilog code at each clock cycle
- A tool for hardware developers
- Web: <https://www.veripool.org/verilator/>

The screenshot shows the Verilator website. At the top, there's a navigation bar with links for Home, Verilator (which is highlighted), Verilog-Mode, Other Tools, Papers, and About. Below the navigation, a large "Welcome to Verilator" header is followed by a sub-header: "Welcome to Verilator, the fastest Verilog/SystemVerilog simulator." It lists several features:

- Accepts Verilog or SystemVerilog
- Performs lint code-quality checks
- Compiles into multithreaded C++, or SystemC
- Creates XML to front-end your own tools

A large blue "V" logo is centered below the sub-header. To the right of the logo, the word "VERILATOR" is written in bold capital letters. Below the logo, there's a bar chart titled "Fast" comparing simulation speeds. The chart has two sets of bars: one for "Big 3" (Vendor A and Vendor B) and one for Verilator. The Y-axis represents threads, ranging from 1 to 12. The X-axis categories are "Big 3", "Big 3", "1 thread", "2 threads", "4 threads", "6 threads", and "12 threads". The Verilator bars are significantly taller than the vendor bars across all thread counts. Below the chart, there's a bulleted list:

- Outperforms many closed-source commercial simulators
- Single- and multi-threaded output models

## QEMU

- An emulator
- Emulate an ideal RISC-V processor for software to run (*exactly like running a virtual machine; in this case, RISC-V on Linux*)
- A tool for software developers
- Web: <https://www.qemu.org/>

The screenshot shows the QEMU website. At the top, there's a dark navigation bar with links for DOWNLOAD, SUPPORT, CONTRIBUTE, DOCS, WIKI, and BLOG. Below the navigation, the word "QEMU" is prominently displayed in large white letters. Underneath it, the tagline "A generic and open source machine emulator and virtualizer" is written in a smaller white font.

# 4. Working with RISC-V (5/21) Working with CHISEL

Remind the Chisel-to-Verilog compile flow:  
Scala (+Chisel) → Java → FIRRTL → Verilog

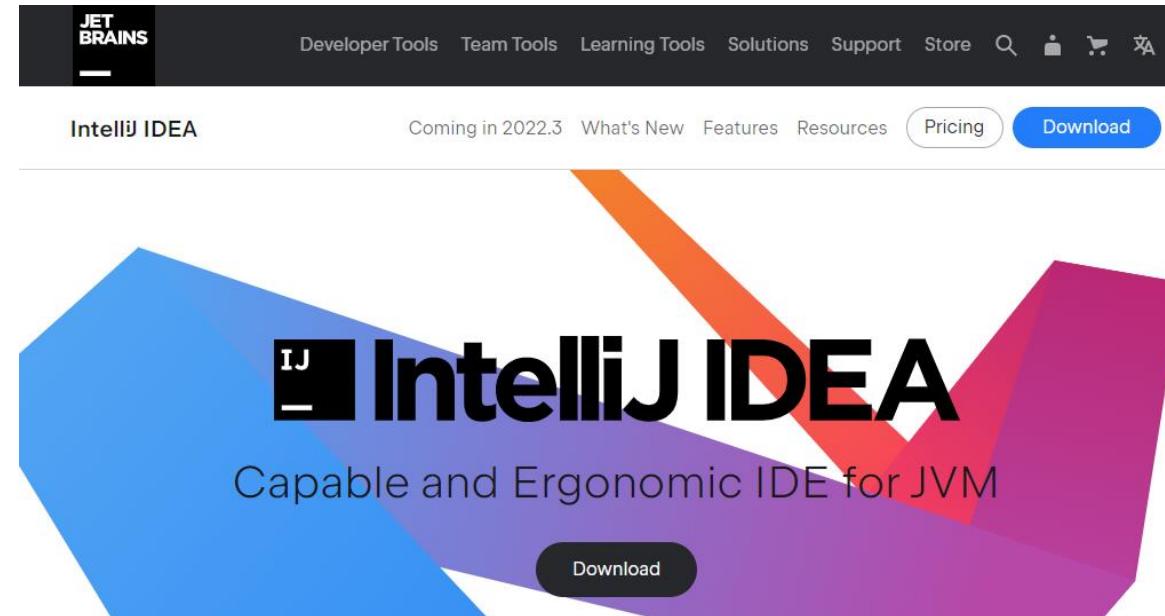
- Scala: a programming language
- Chisel: a library (*attached to Scala*)
- Sbt: a Scala compiler

So, we need to install the **Sbt** tool.

Webpage: <https://www.scala-sbt.org/>

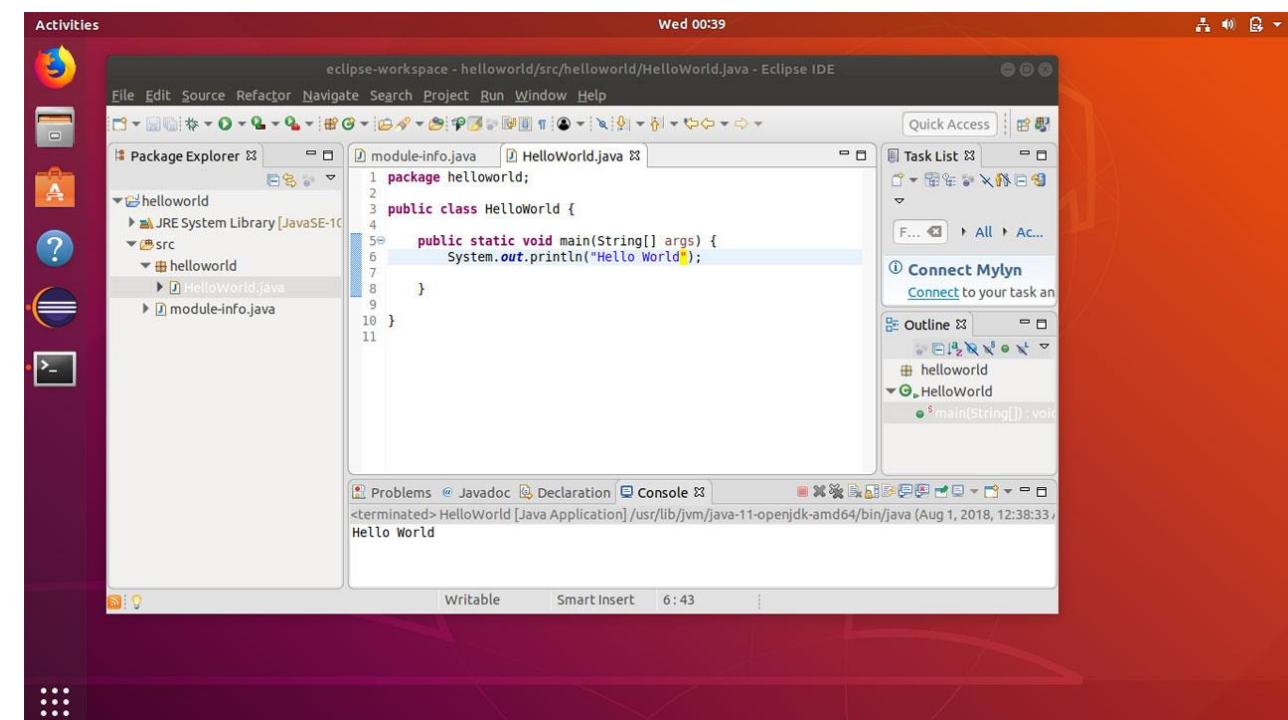


The **Sbt** tool itself is a terminal command line.  
Recommend installing **IntelliJ IDEA**, a GUI for  
Sbt (*like a Visual Studio for C/C++*).  
Webpage: <https://www.jetbrains.com/idea/>



# 4. Working with RISC-V (6/21) Software (C/C++) coding

For software coding, especially C/C++, recommend installing **Eclipse**.  
Webpage: <https://www.eclipse.org/>



# 4. Working with RISC-V (7/21) Debug tools

OpenOCD is the tool used for debugging  
*(together with RISC-V GDB in the toolchain):*

Webpage: <https://openocd.org/>

GitHub: <https://github.com/riscv/riscv-openocd>

## Open On-Chip Debugger

About Bug Tracker Discussion Documentation Donations

Getting OpenOCD IRC Mailing lists Repository Supported JTAG interfaces

### OpenOCD 0.12.0-rc1 release candidate is out

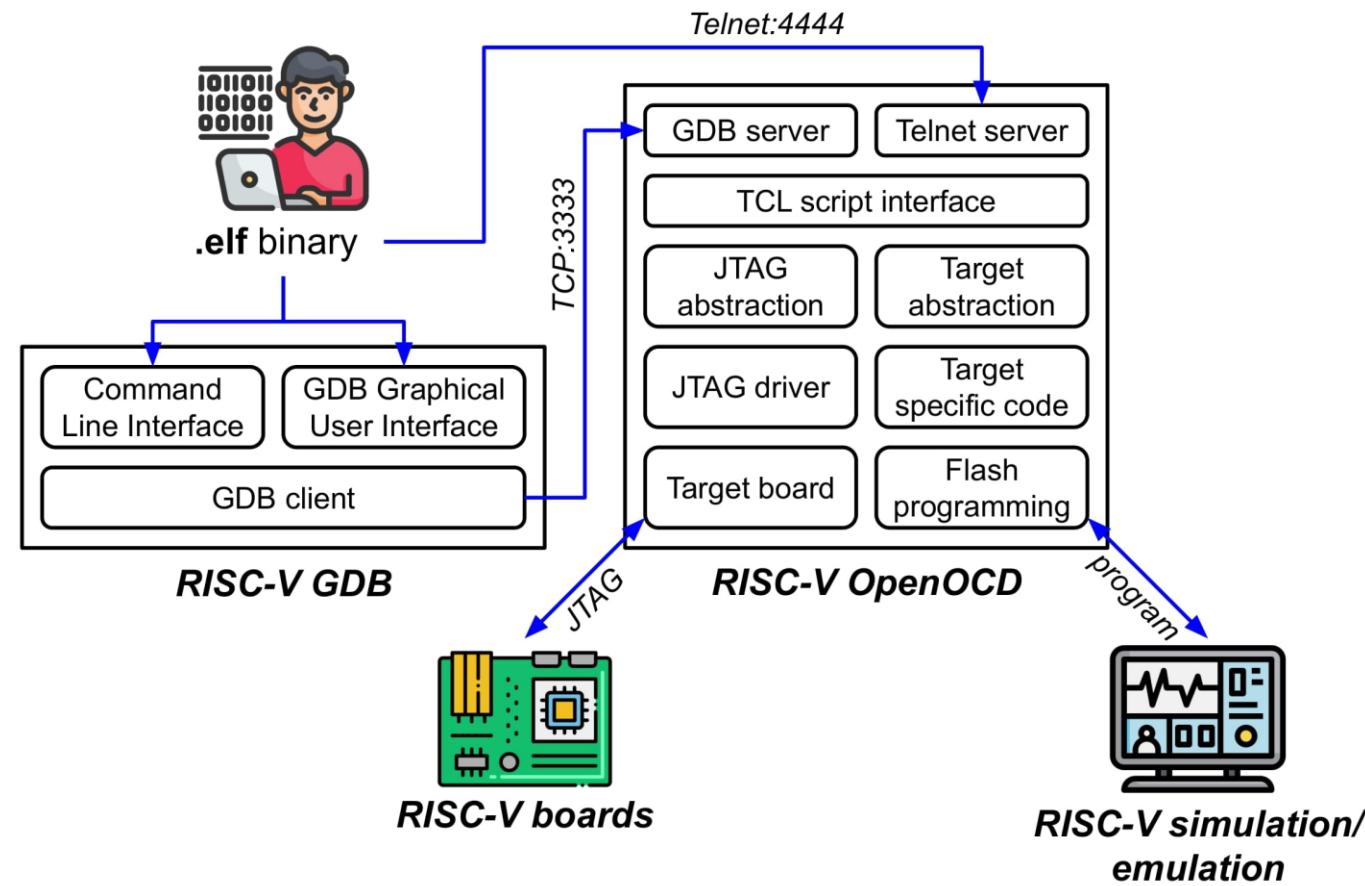
We are pleased to announce the first release candidate of the upcoming OpenOCD version.

Sun 18 September 2022  
By [fercerpav](#)

The source archives and release notes are available from [the usual SF download locations](#).

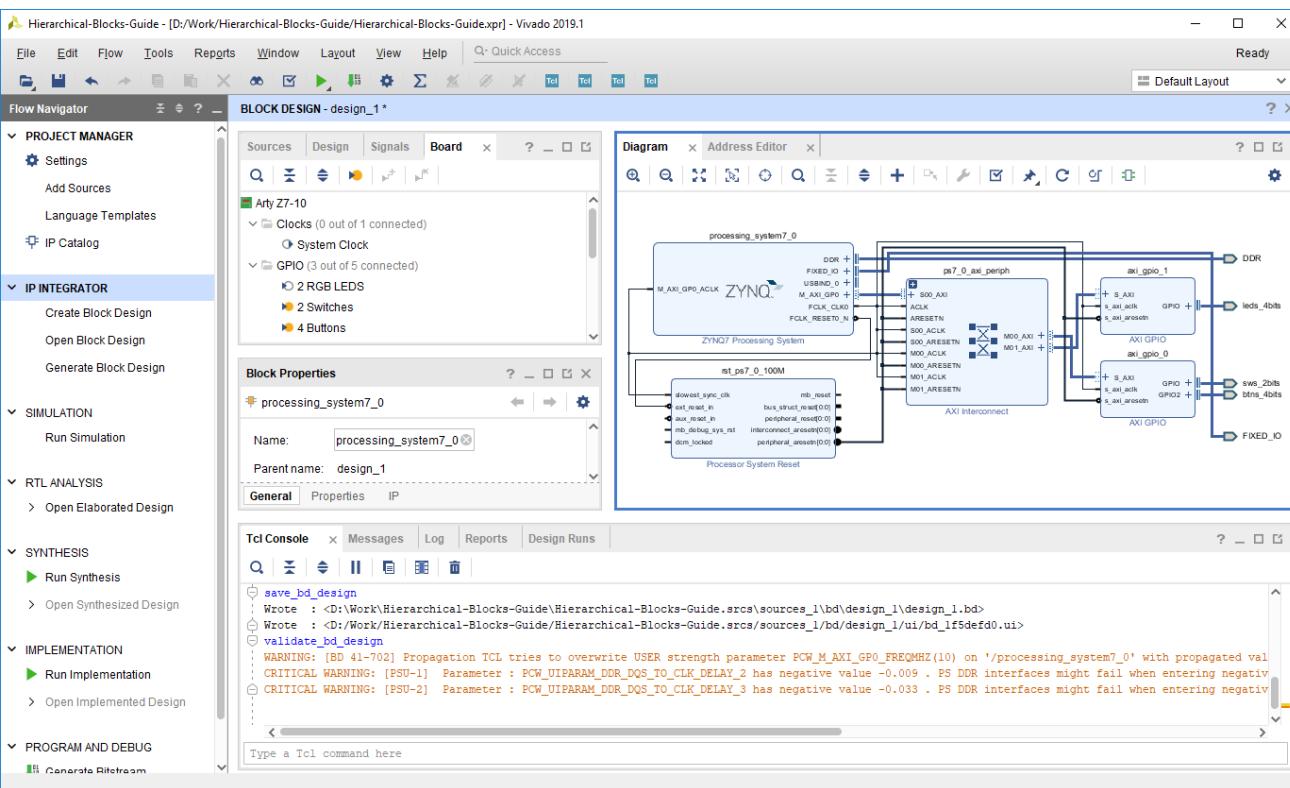
Please post all your feedback to the [openocd-devel mailing list](#).

A typical debug flow with GDB and OpenOCD:



# 4. Working with RISC-V (8/21) Working with FPGA

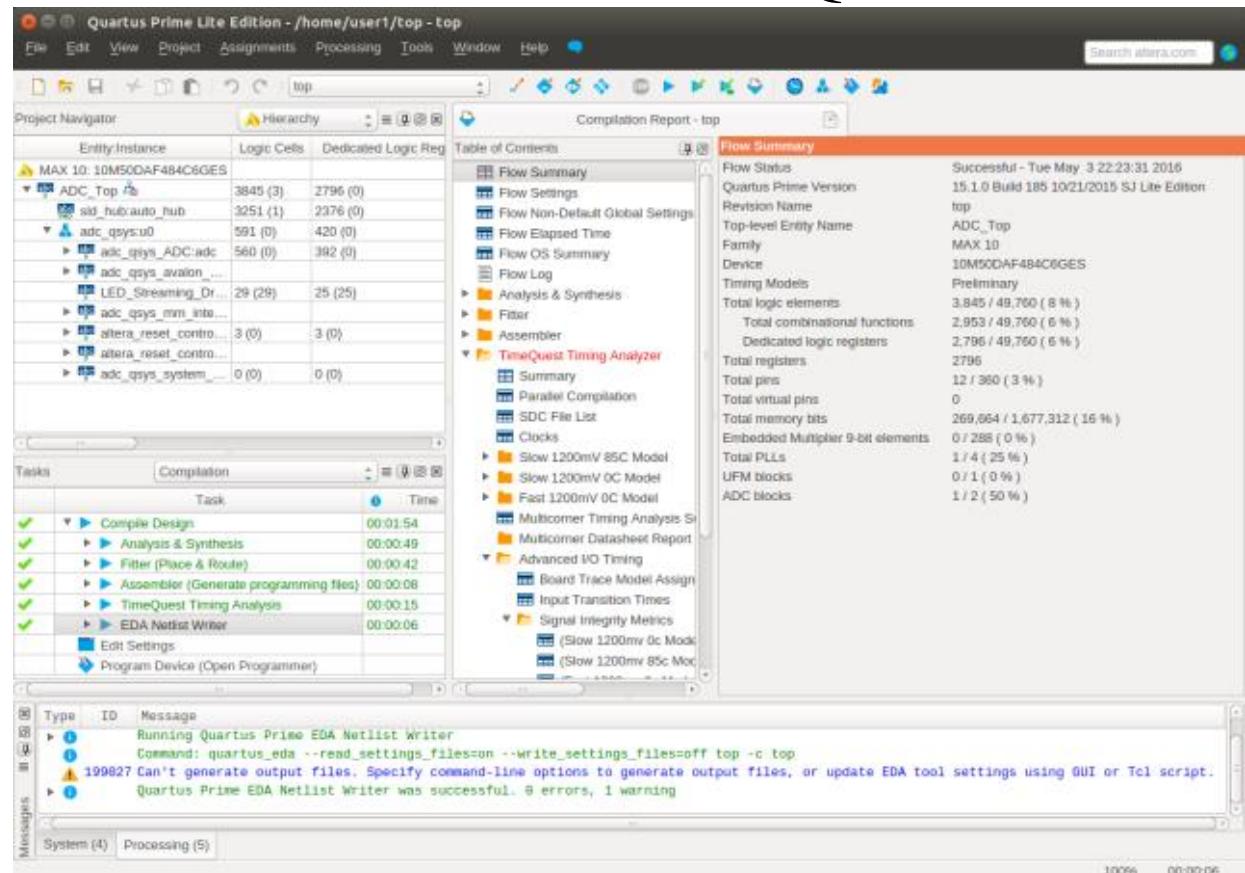
Finally, if you are working with FPGAs, you must install Vivado for Xilinx or Quartus for Altera boards.



Vivado ML standard

Web:

<https://www.xilinx.com/support/download.html>



Quartus Prime Standard

Web:

<https://www.intel.com/content/www/us/en/collections/products/fpga/software/downloads.html>

# 4. Open community (9/21) Open libraries

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

**CHIPIYARD**

Chipyard Framework chipyard-ci-process passing

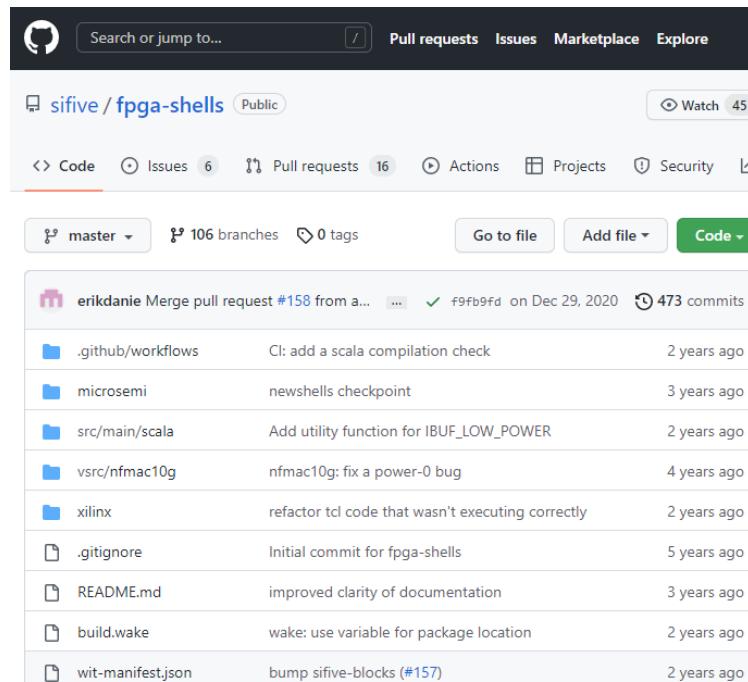
Quick Links

- 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



sifive / fpga-shells Public Watch 45

Code Issues 6 Pull requests 16 Actions Projects Security

master 106 branches 0 tags Go to file Add file Code

erikdanie Merge pull request #158 from a... f9fb9fd on Dec 29, 2020 473 commits

File	Description	Time Ago
.github/workflows	CI: add a scala compilation check	2 years ago
microsemi	newshells checkpoint	3 years ago
src/main/scala	Add utility function for IBUF_LOW_POWER	2 years ago
vsrc/nfmac10g	nfmac10g: fix a power-0 bug	4 years ago
xilinx	refactor tcl code that wasn't executing correctly	2 years ago
.gitignore	Initial commit for fpga-shells	5 years ago
README.md	improved clarity of documentation	3 years ago
build.wake	wake: use variable for package location	2 years ago
wit-manifest.json	bump sifive-blocks (#157)	2 years ago

README.md

fpga-shells

An FPGA shell is a Chisel module designed to wrap any SiFive core configuration. 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>

# 4. Open community (10/21) 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)

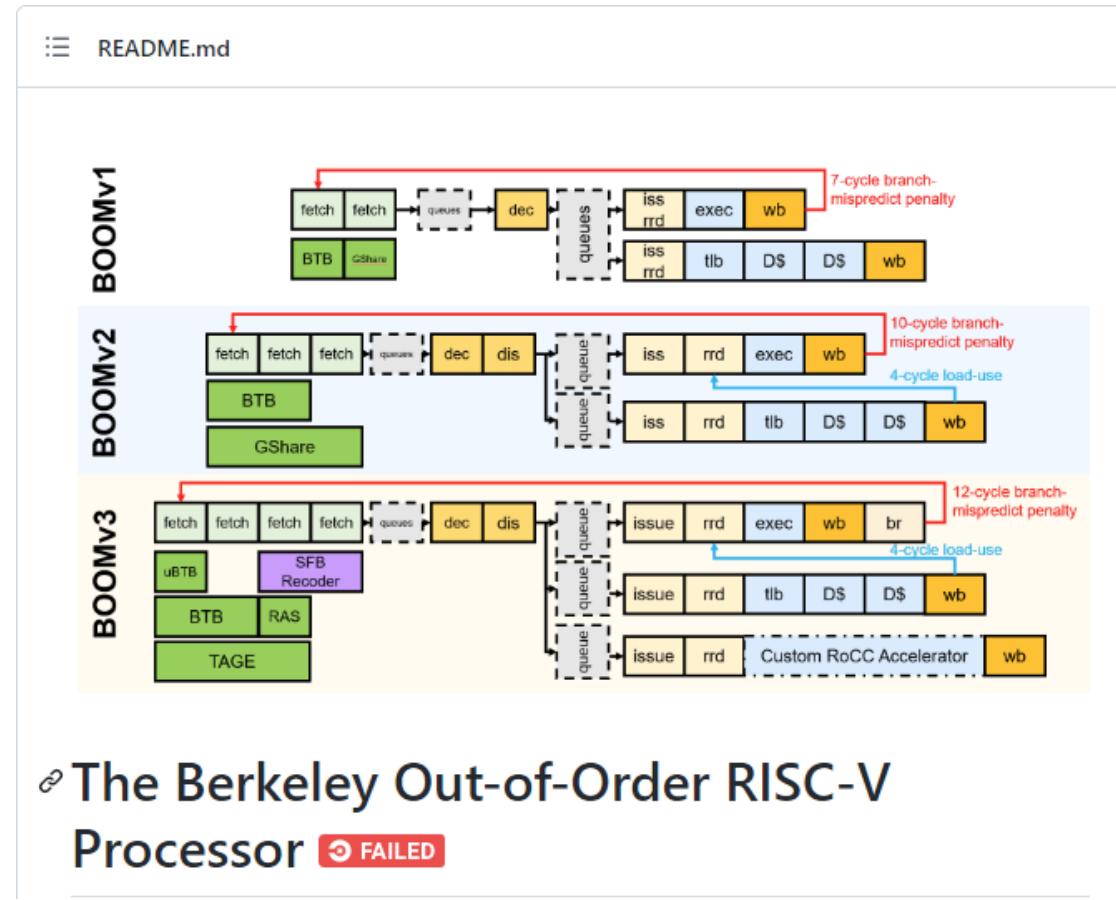
The screenshot shows the README.md page of the Rocket Chip Generator repository. It features a header with a logo and a 'Continuous Integration' status badge. Below the header, there's a brief description of the repository's purpose and a 'Table of Contents' section with several links to various documentation and usage guides.

**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>



# 4. Open community (11/21) Open processors

## Some famous RISC-V processors

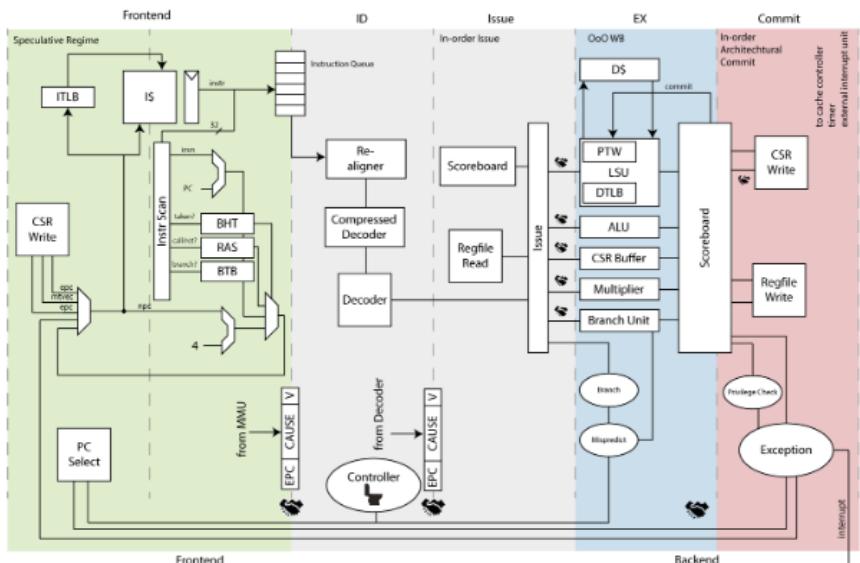
Ariane is a high-performance 64-bit in-of-order processor:

<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.

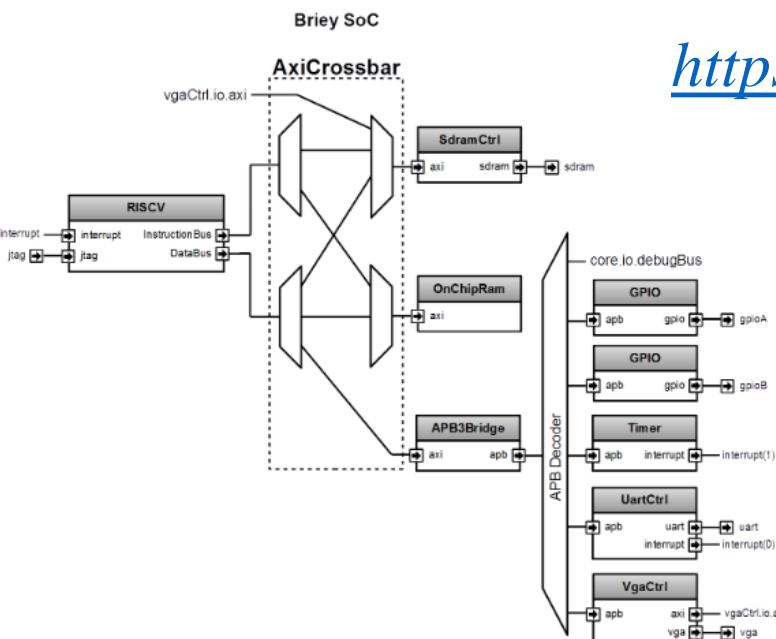
It has configurable size, separate TLBs, a hardware PTW and branch-prediction (branch target buffer and branch history table). The primary design goal was on reducing critical path length.



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>

# 4. Open community (12/21) Open processors

## Some famous RISC-V processors

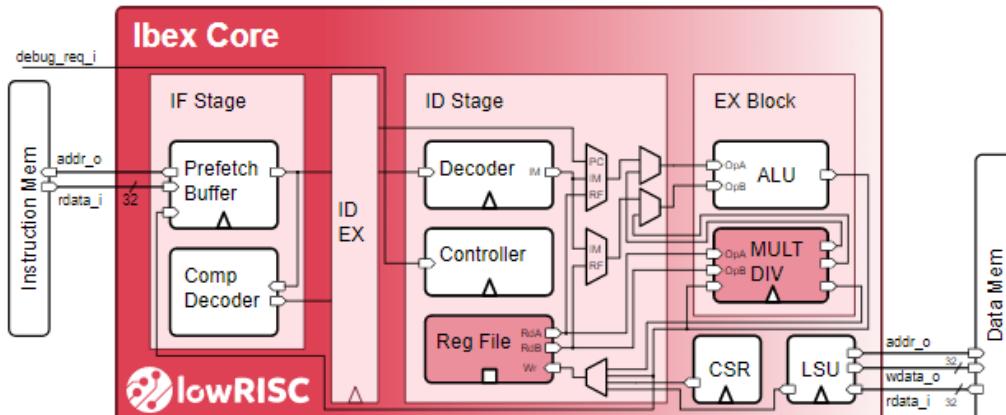
Ibex is an in-order 32-bit processor that focuses on security applications:  
<https://github.com/lowRISC/ibex>



### Ibex RISC-V Core

Ibex is a production-quality open source 32-bit RISC-V CPU core written in SystemVerilog. The CPU core is heavily parametrizable and well suited for embedded control applications. Ibex is being extensively verified and has seen multiple tape-outs. Ibex supports the Integer (I) or Embedded (E), Integer Multiplication and Division (M), Compressed (C), and B (Bit Manipulation) extensions.

The block diagram below shows the *small* parametrization with a 2-stage pipeline.

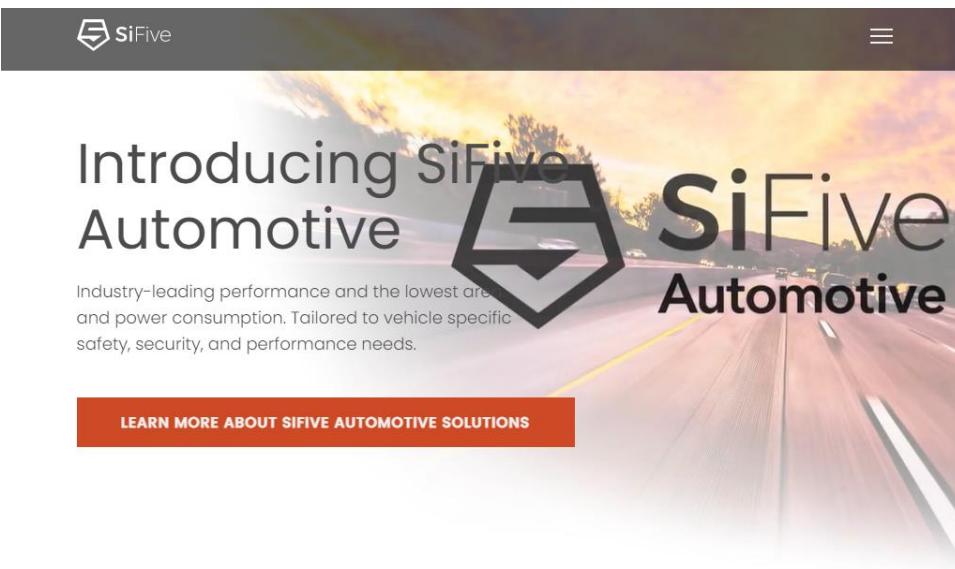


Ibex was initially developed as part of the [PULP platform](#) under the name "Zero-riscy", and has been contributed to [lowRISC](#) who maintains it and develops it further. It is under active development.

# 4. Open community (13/21) Groups & companies

## Some famous RISC-V groups & companies

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



A screenshot showing a news feed from the SiFive website. It includes two articles: one about the SiFive X280 as a coprocessor in Google's datacenter, dated September 21, 2022, and another about SiFive expanding its operations to the UK, dated June 28, 2022. Both articles have "Read More" links.

- 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>

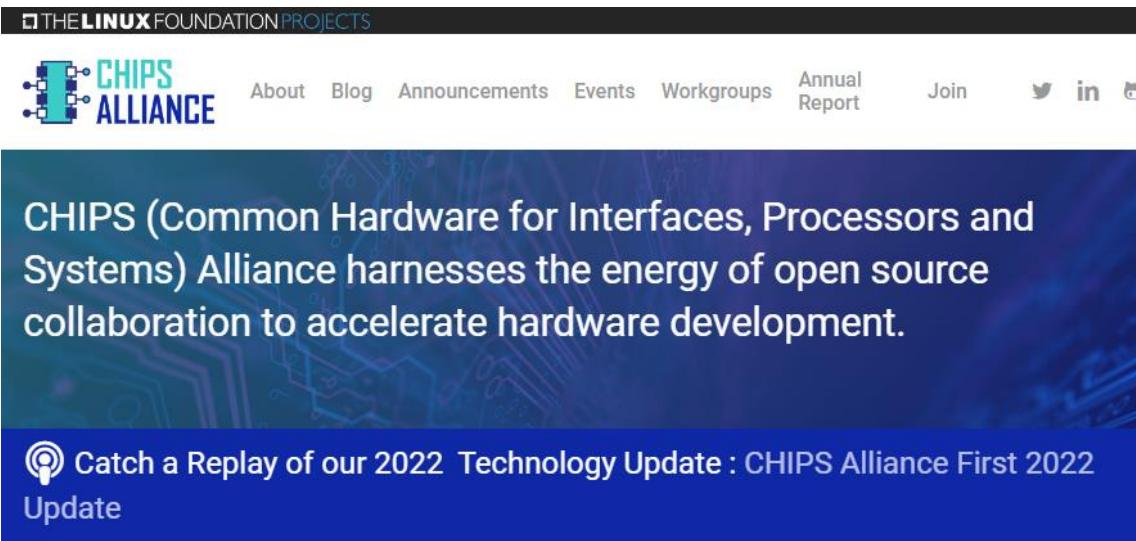
0x5 HEX-Five Security Products Partners Company Downloads Contact FAQ



# 4. Open community (14/21) Groups & companies

## Some famous RISC-V groups & companies

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



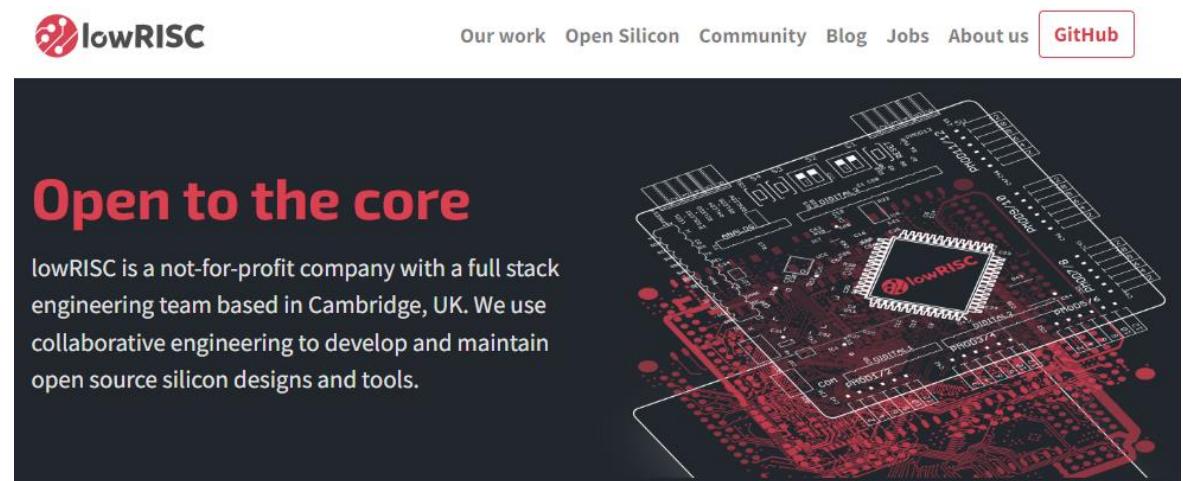
The screenshot shows the CHIPS Alliance website under THE LINUX FOUNDATION PROJECTS. The header includes the Linux Foundation logo, the CHIPS Alliance logo, and links for About, Blog, Announcements, Events, Workgroups, Annual Report, Join, and social media icons for Twitter, LinkedIn, and GitHub. The main content area features a blue background with white text: "CHIPS (Common Hardware for Interfaces, Processors and Systems) Alliance harnesses the energy of open source collaboration to accelerate hardware development." Below this is a call-to-action button: "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>

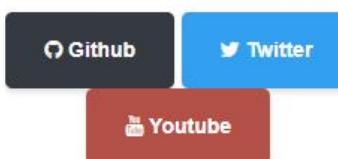
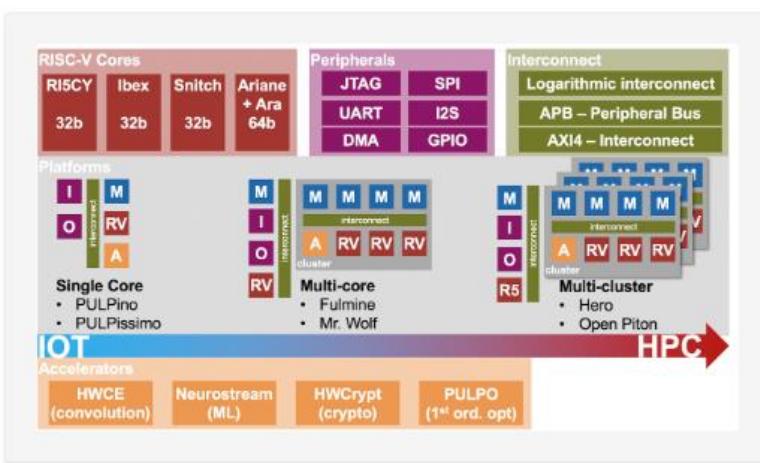
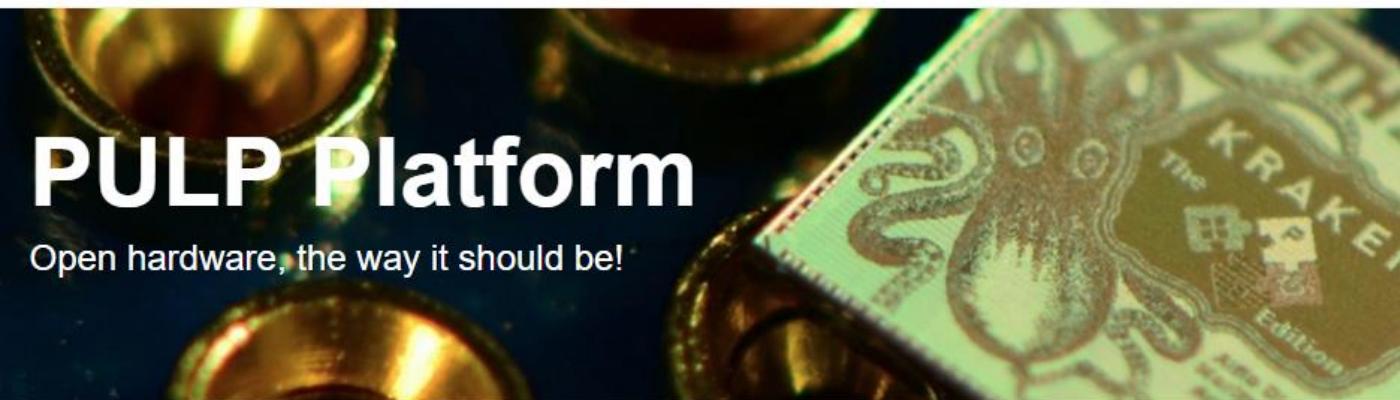


The screenshot shows the lowRISC website. The header includes the lowRISC logo and links for Our work, Open Silicon, Community, Blog, Jobs, About us, and GitHub. The main content area features a dark background with red text: "Open to the core". Below this is a paragraph: "lowRISC is a not-for-profit 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." To the right is an image of a printed circuit board (PCB) with a central square component labeled "lowRISC".

# 4. Open community (15/21) Groups & companies

## Some famous RISC-V groups & companies

PULP Platform Resources ▾ About ▾ FAQ Privacy Policy Contact



### 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:

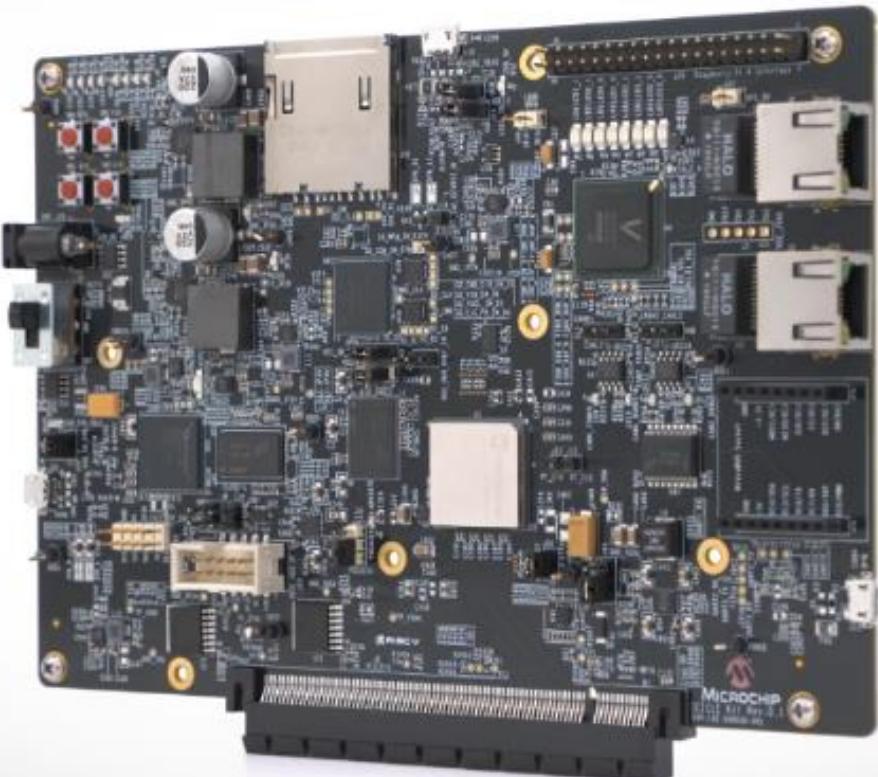
- Their website: <https://pulp-platform.org/>
- Their github: <https://github.com/pulp-platform>

## 4. Open community (16/21) Products

Some RISC-V-related products

**PolarFire SoC by Microchip** is the first FPGA with a dedicated hard IP for RISC-V processors (Rocket core).

### PolarFire® SoC Icicle Kit

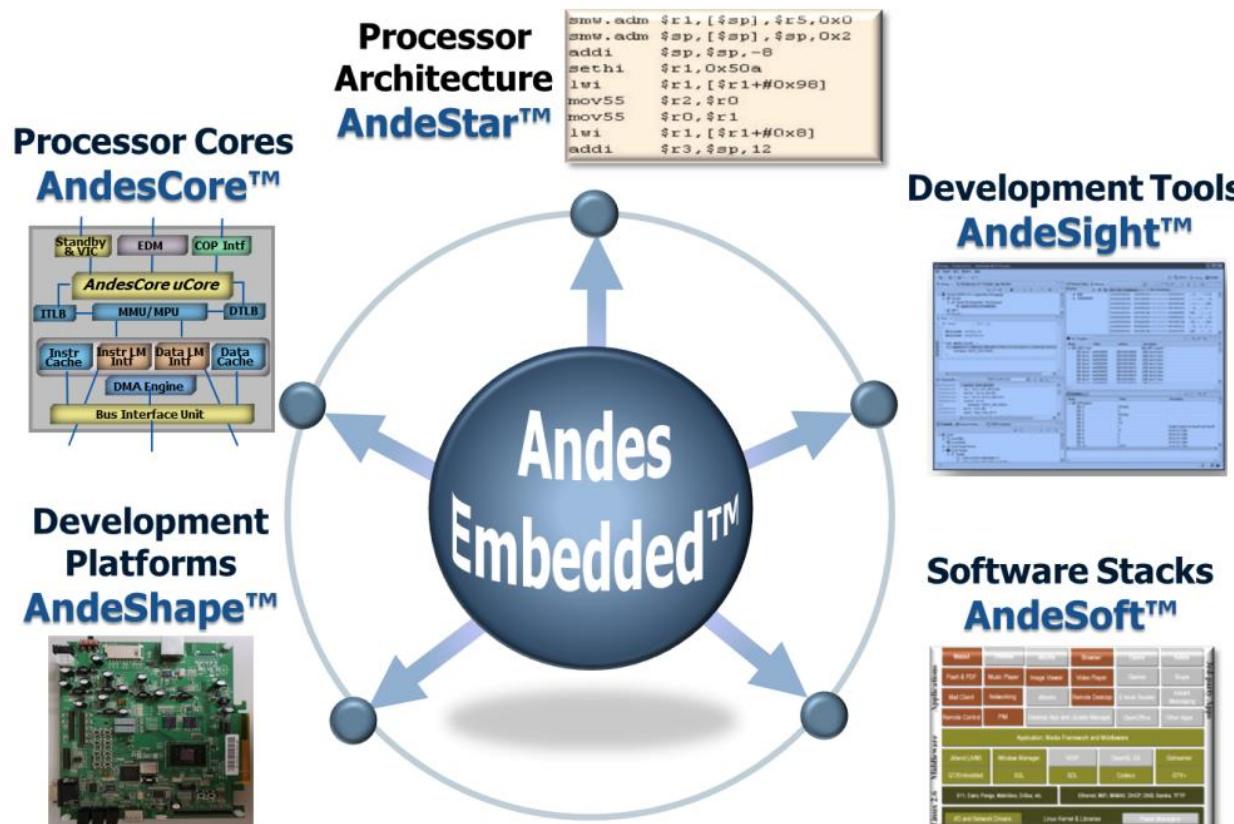


- FPGA: <https://www.microchip.com/en-us/products/fpgas-and-plds/system-on-chip-fpgas/polarfire-soc-fpgas>
- Ecosystem (Mi-V):  
<https://www.microchip.com/en-us/products/fpgas-and-plds/fpga-and-soc-design-tools/mi-v>
- Github page: <https://github.com/polarfire-soc>

# 4. Open community (17/21) Products

Some RISC-V-related products

**Andes Technology** is a company that focuses on delivering commercial RISC-V products:



- Their website:  
<https://www.andestech.com/en/risc-v-andes/>
- Their Github page:  
<https://github.com/andestech>

# 4. Open community (18/21) Products

## Some RISC-V-related products



The screenshot shows the Western Digital RISC-V landing 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 green glowing lights. 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. At the bottom, there's a "Innovations" section with the headline "RISC-V Enables a System on a Chip".

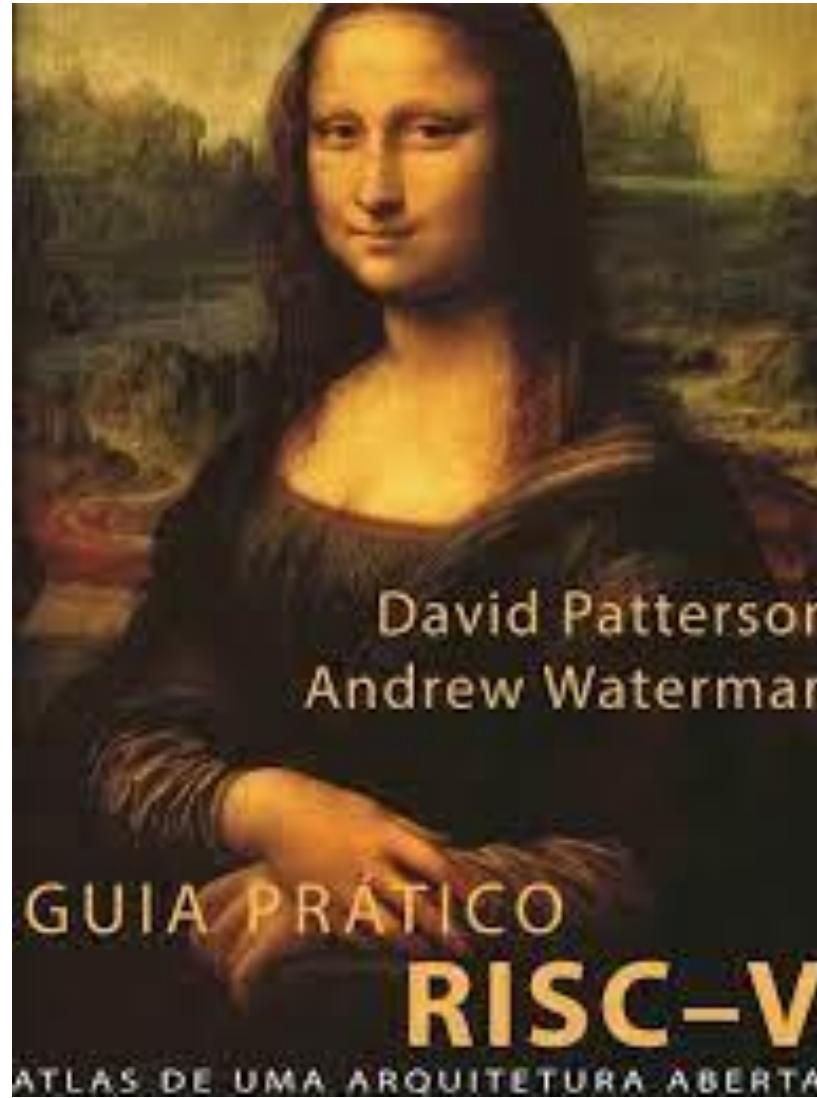
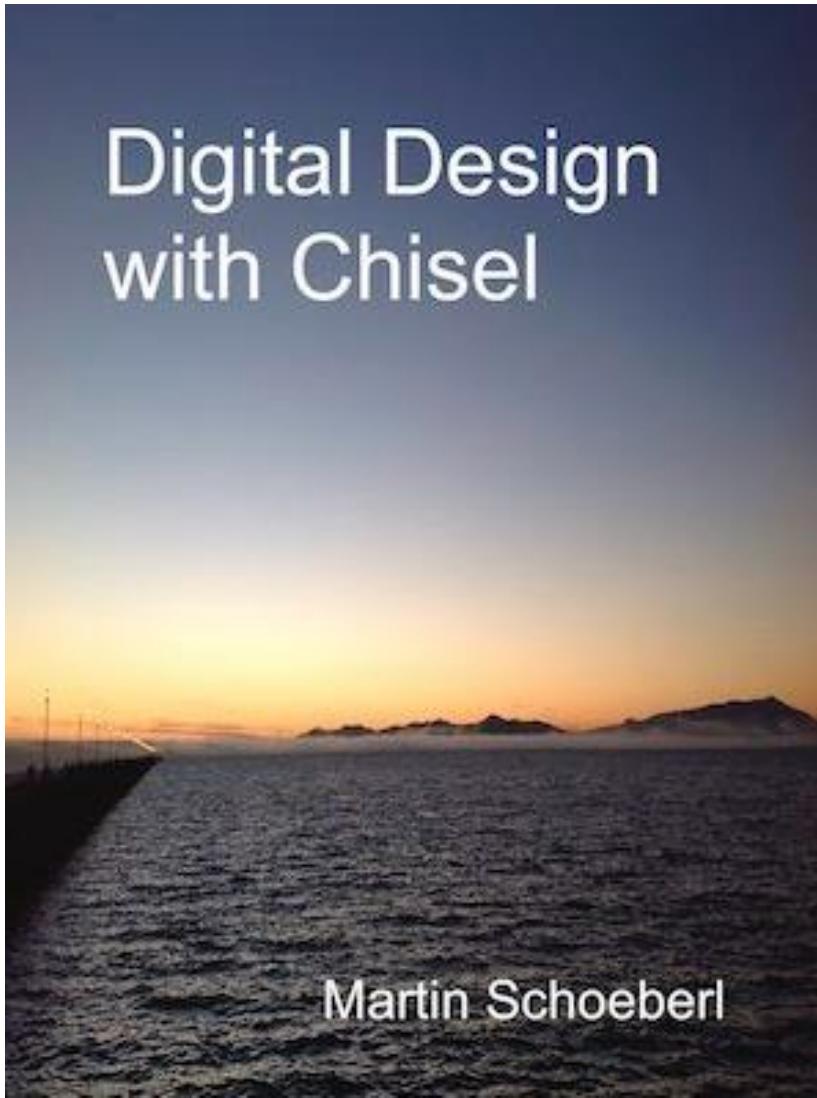
Promising upcoming products in  
**Western Digital and Seagate**

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>

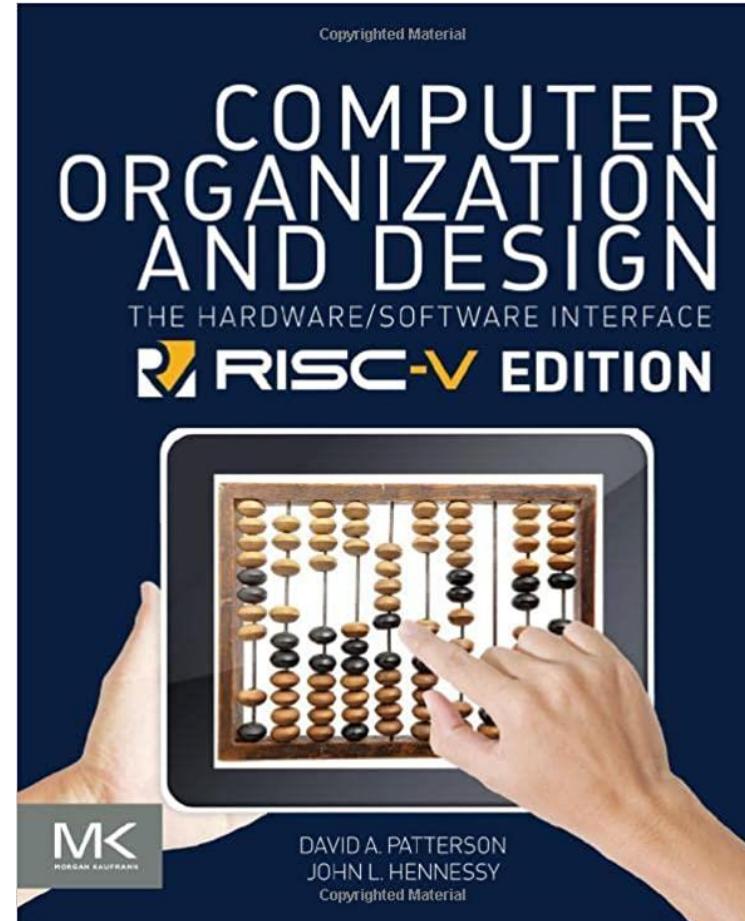
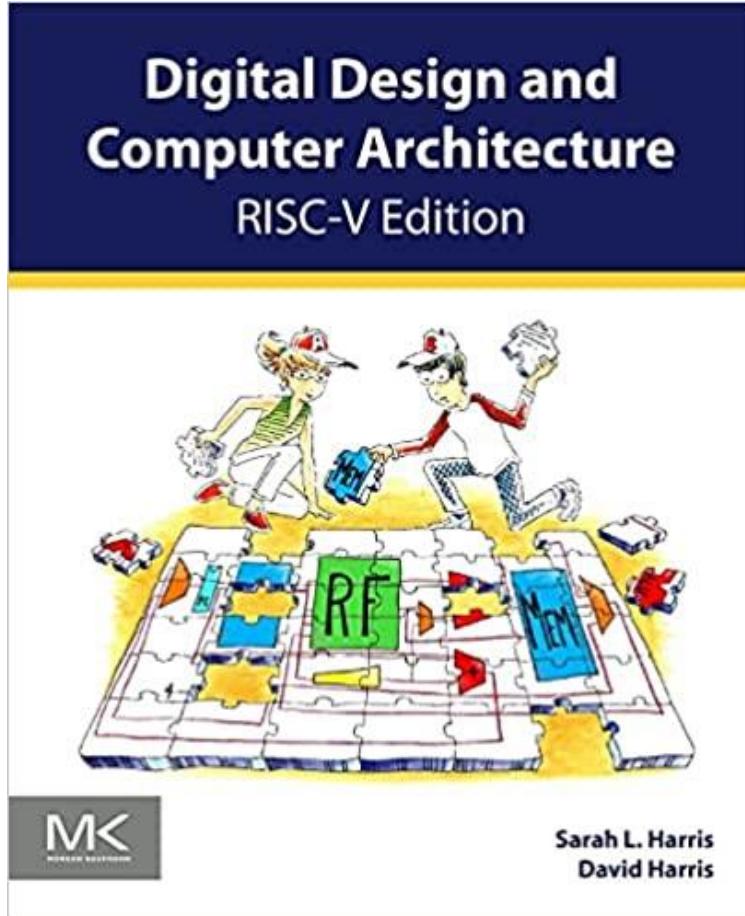
## 4. Open community (19/21) Books

Two “must-have” books for RISC-V developers,  
from beginners to experts



## 4. Open community (20/21) Books

RISC-V books that often used in universities for teaching



I have a book/material collection on google drive:

<https://drive.google.com/drive/folders/1bAfqALmKGJLOcOlqiE51WB-SudU6-EY?usp=sharing>

# 4. Open community (21/21) Recommend for beginners

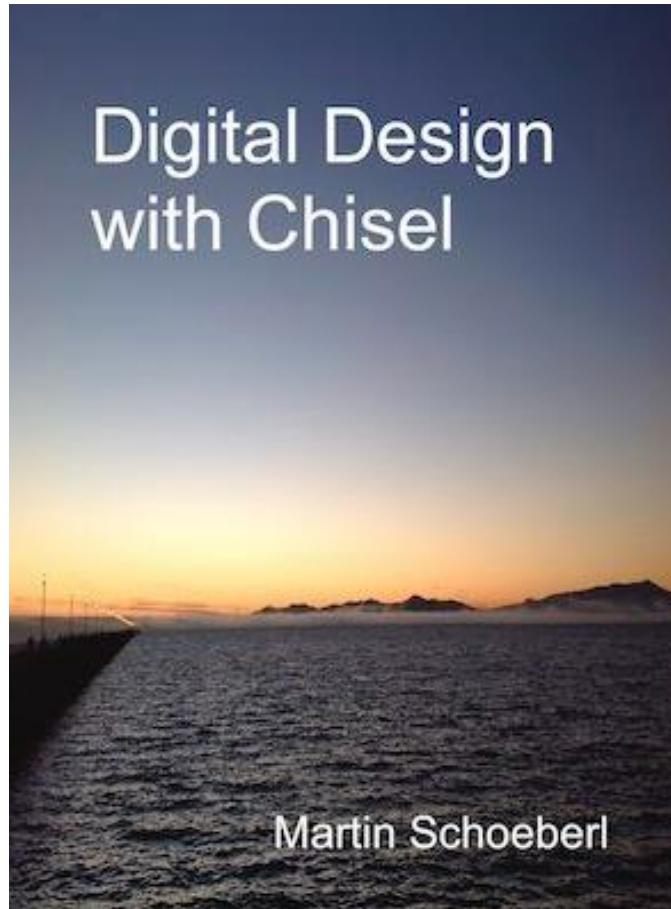
Beginners often started with CHISEL tutorial and specification.

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 changes to files like doc, project, src, .gitignore, LICENSE.txt, README.md, build.sbt, run-examples.sh, run-problem.sh, and run-solution.sh. The repository also includes a README.md file and a Chisel Tutorials (Release branch) section.

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

CHISEL coding:



RISC-V specification:

A screenshot of the RISC-V Specifications page on riscv.org. The page features the RISC-V logo and a large "Specifications" heading. It explains that the instruction set architecture (ISA) and related specifications are developed, ratified, and maintained by Technical Working Groups. Work on the specification is performed on GitHub, and the GitHub issue mechanics are mentioned. A "ISA Specification" section lists three items: Volume 1, Unprivileged Spec v. 20191213 [PDF], Volume 2, Privileged Spec v. 20211203 [PDF], and Recently ratified, but not yet integrated, extension specifications. A note states that past ratified releases include the term "ratified" in the release tag.

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

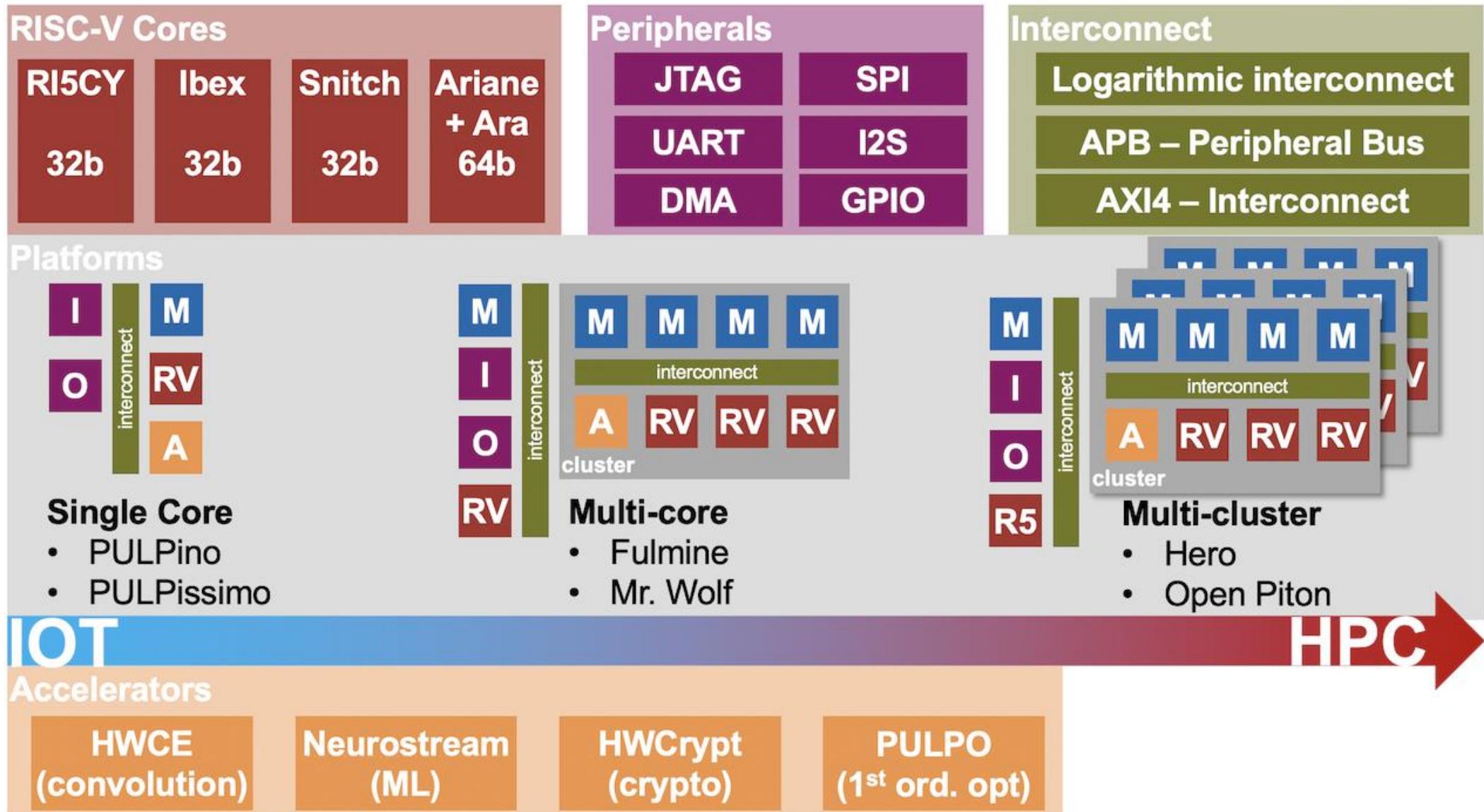


# OUTLINE

1. Introduction
2. Achievement highlight
3. What is RISC-V?
4. Working with RISC-V & open community
5. Main research @ UEC
6. RISC-V courses @ UEC

# 5. Main research @ UEC (1/30) Summary

- **Going low:**  
aim for low-power IoT
- **Going high:**  
aim for high-level cyber-security



([https://pulp-platform.org/img/main/pulp\\_family.png](https://pulp-platform.org/img/main/pulp_family.png))

# 5. Main research @ UEC (2/30) Cyber-security overview

Power and EM Analysis Attacks

Branch Prediction

Timing Channels

Intra-core Side-channel

Detection Techniques

## Side-channel Prevention

Lightweight Crypto

Symmetric/Asymmetric

SIKE

Elliptic Curves

TRNG

DICE

## Cryptographic Primitives

Reduce Attack Surface

SMPC

CFI

Cryptography

Side-channel Resist

## ISA Security Extensions

Tagged Memory

Memory Isolation

Memory Encryption and Authentication

## Memory Protection

Covert Channels

Physical Access

Logic-locking

EM Fault Injection

RTL Bugs

Hardware Trojans

## Hardware and Physical Security

Program Obfuscator and Churn Units

Memory Protection

Crypto Engines

## Hardware-assisted Security Units

Cyber-security topics attract attention in the RISC-V community.

### Main reasons:

- Opportunity to re-design from the bottom
- Price-sensitive applications
- Open approach
- Open community

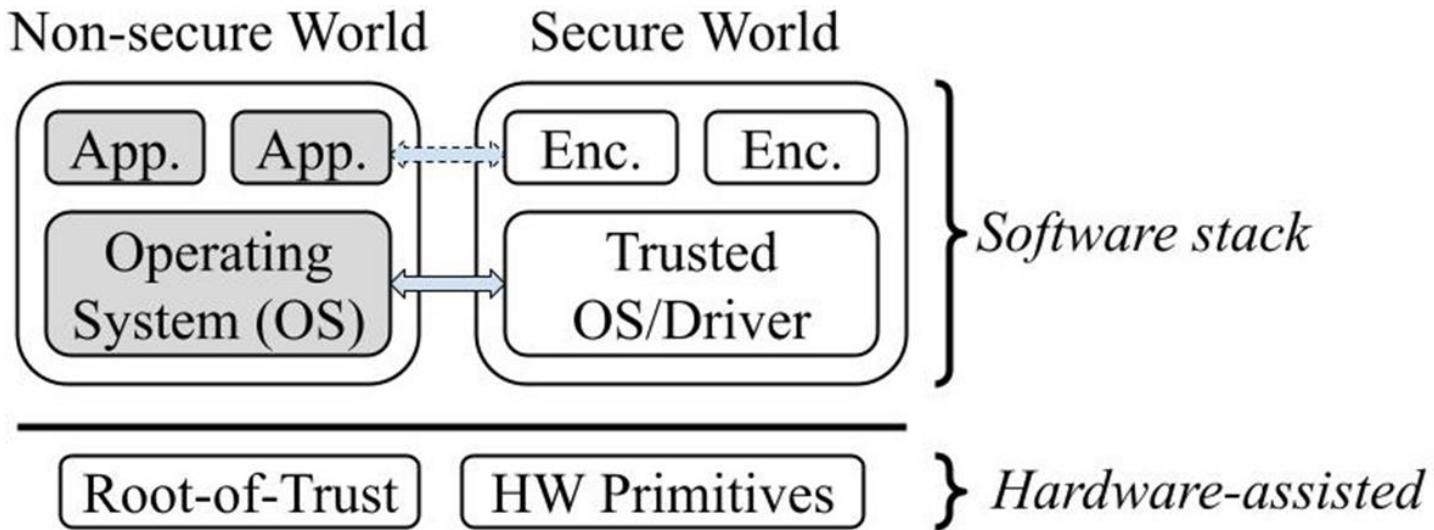
# 5. Main research @ UEC (3/30) TEE?

**Trusted Execution Environment (TEE) provides:**

1. *Integrity*: the code and data cannot be tampered.
2. *Confidentiality*: the application's content cannot be read.
3. *Attestation*: proof to a remote party that the system is safe.

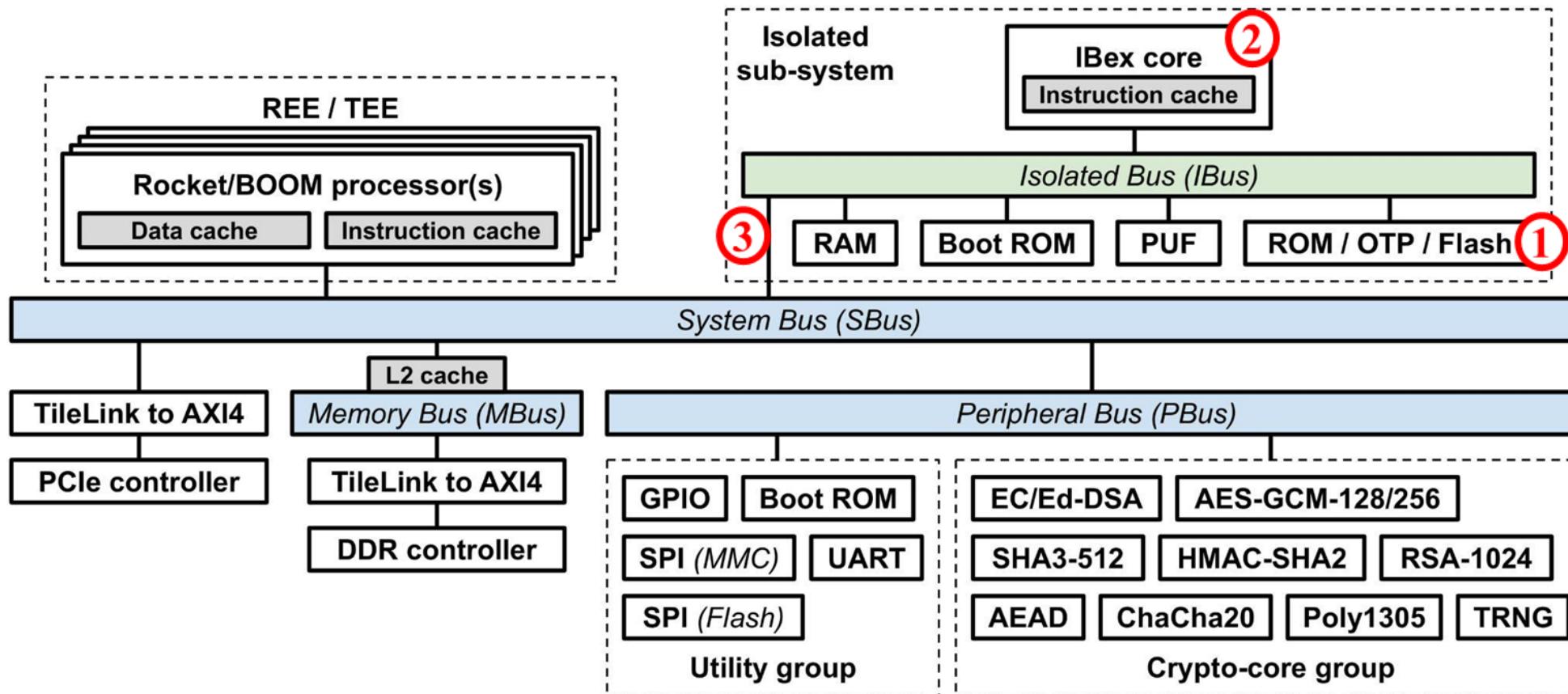
**A typical TEE setup:**

- Secure (trusted) vs. non-secure (untrusted) worlds.
- Barrier enforcer by: software *AND* hardware.
- All TEEs need some sort of hardware-assisted modules: Root-of-Trust (RoT) and primitives.
- HW primitives (*examples*): cache flushing, cache partitioning, memory isolation, memory encryption, keys management, bus access controller, enclave encryption, and so on.



# 5. Main research @ UEC (4/30) RoT for TEE

A secure boot process with Root-of-Trust (RoT) for TEE



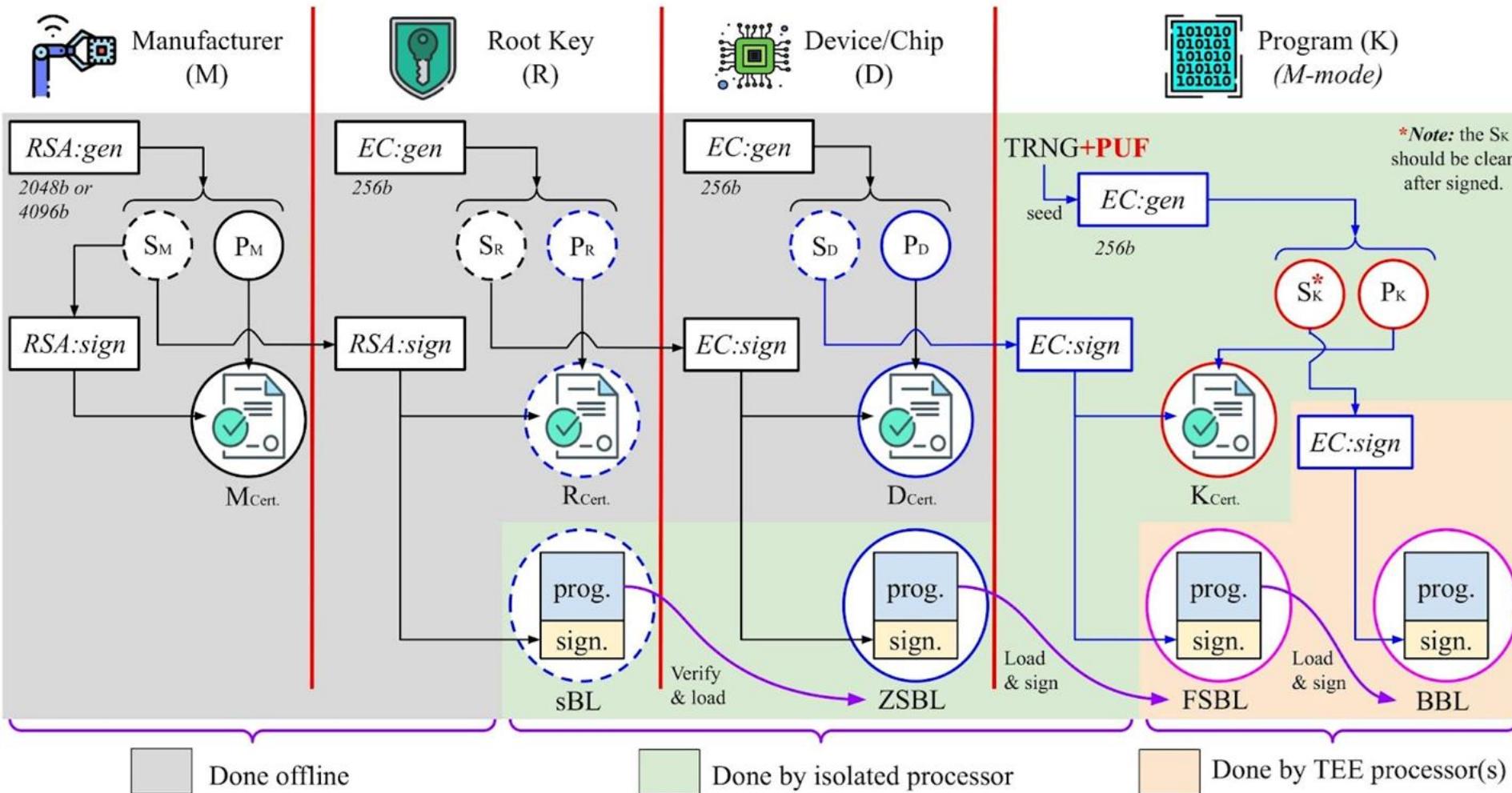
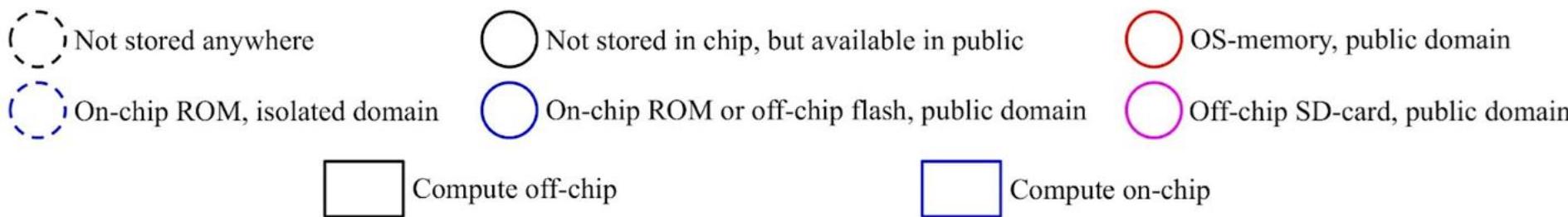
Main points of the proposed architecture:

1. Root key installed at the time manufactured.

2. Hidden MCU for the flexible boot program

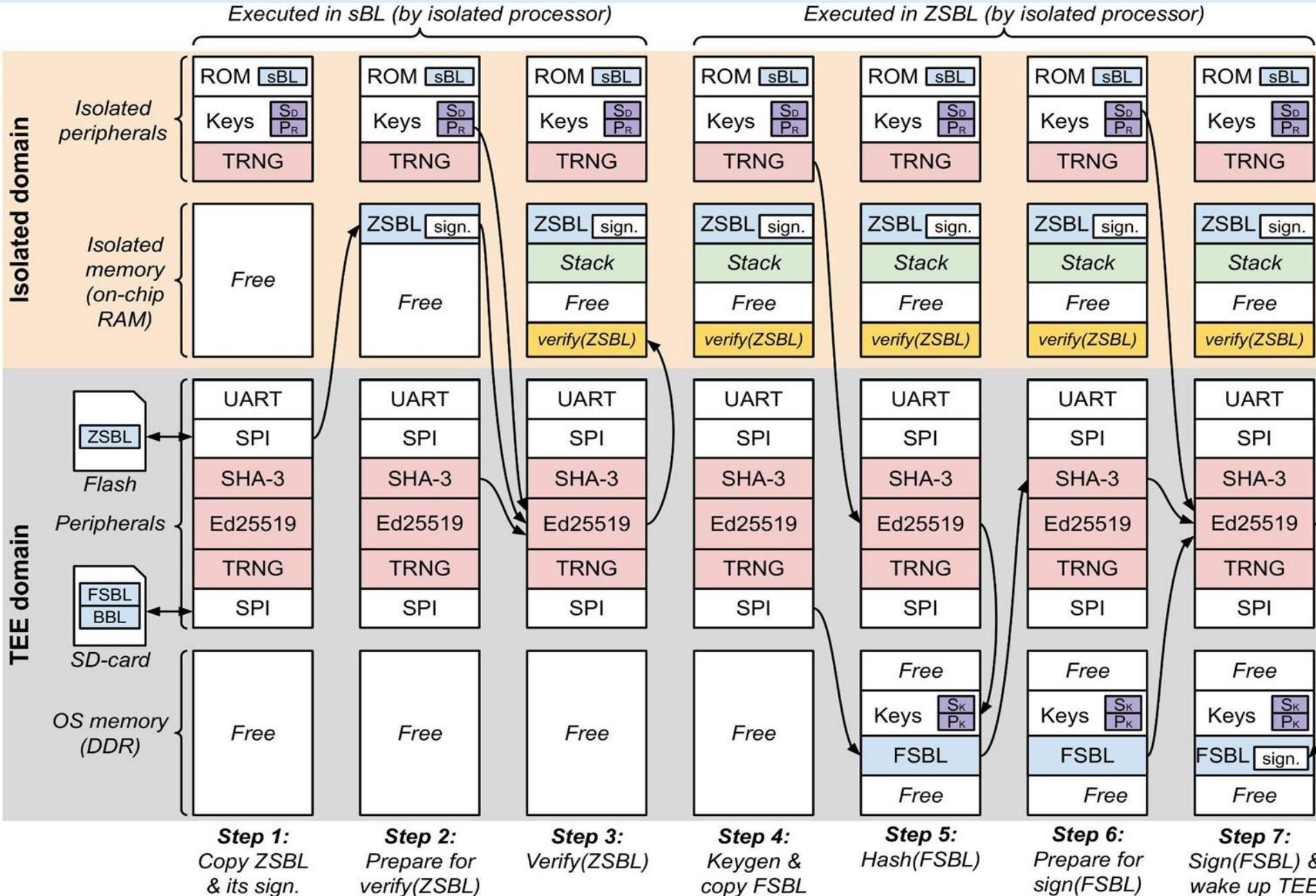
3. Hierarchy-bus: TEE processors cannot access RAM/ROMs in the isolated domain (*BUT the isolated core can access ALL*)

# 5. Main research @ UEC (5/30) RoT for TEE



The proposed keys scheduling scheme.

# 5. Main research @ UEC (6/30) RoT for TEE

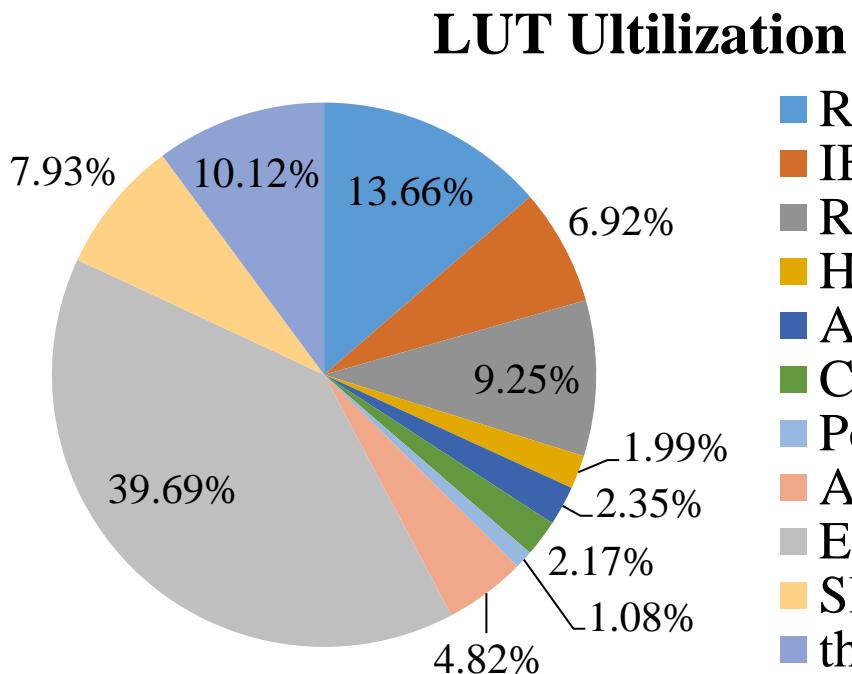


The proposed boot sequence.

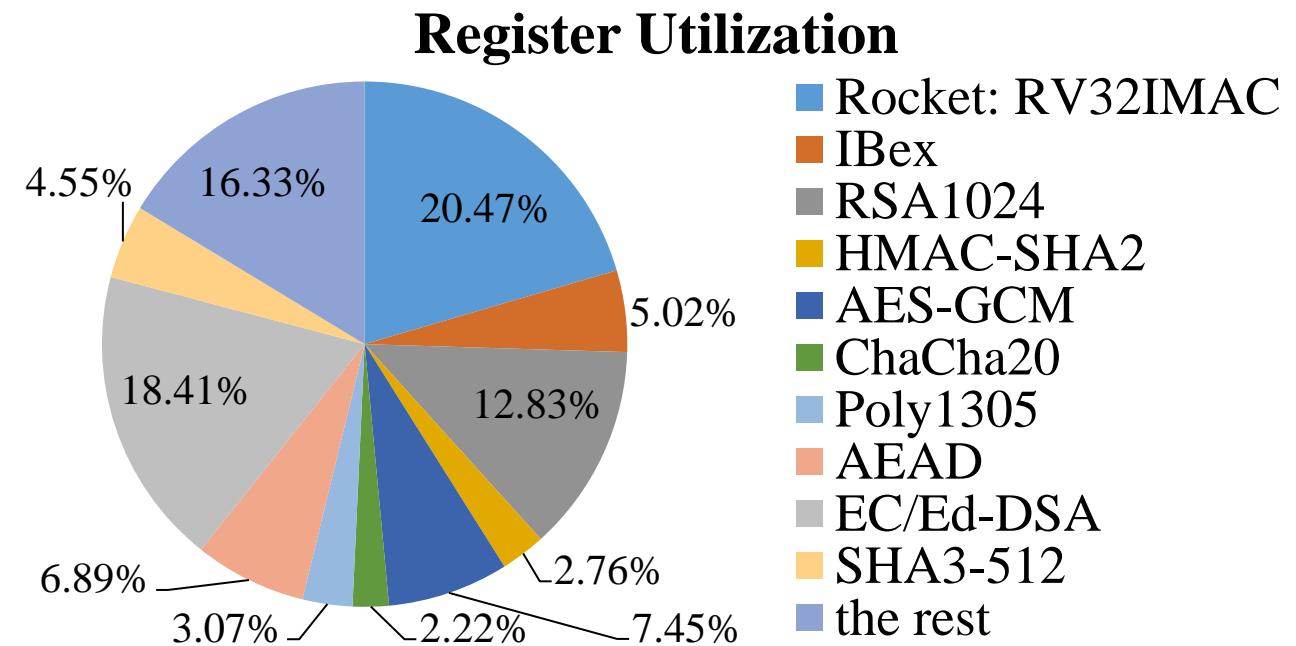
# 5. Main research @ UEC (7/30) RoT for TEE

SoC resources utilization pie chart:  
All crypto-cores + Single-core RV32IMAC Rocket

VC707 FPGA (Virtex-7) build report



- Rocket: RV32IMAC
- IBEx
- RSA1024
- HMAC-SHA2
- AES-GCM
- ChaCha20
- Poly1305
- AEAD
- EC/Ed-DSA
- SHA3-512
- the rest

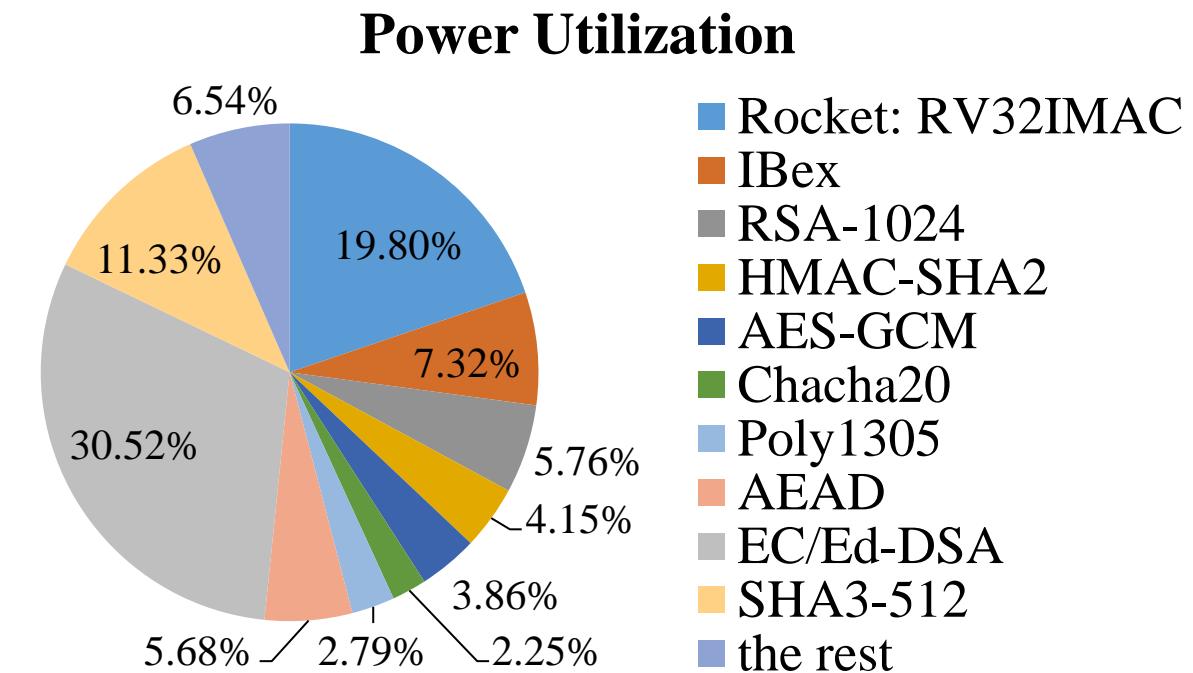
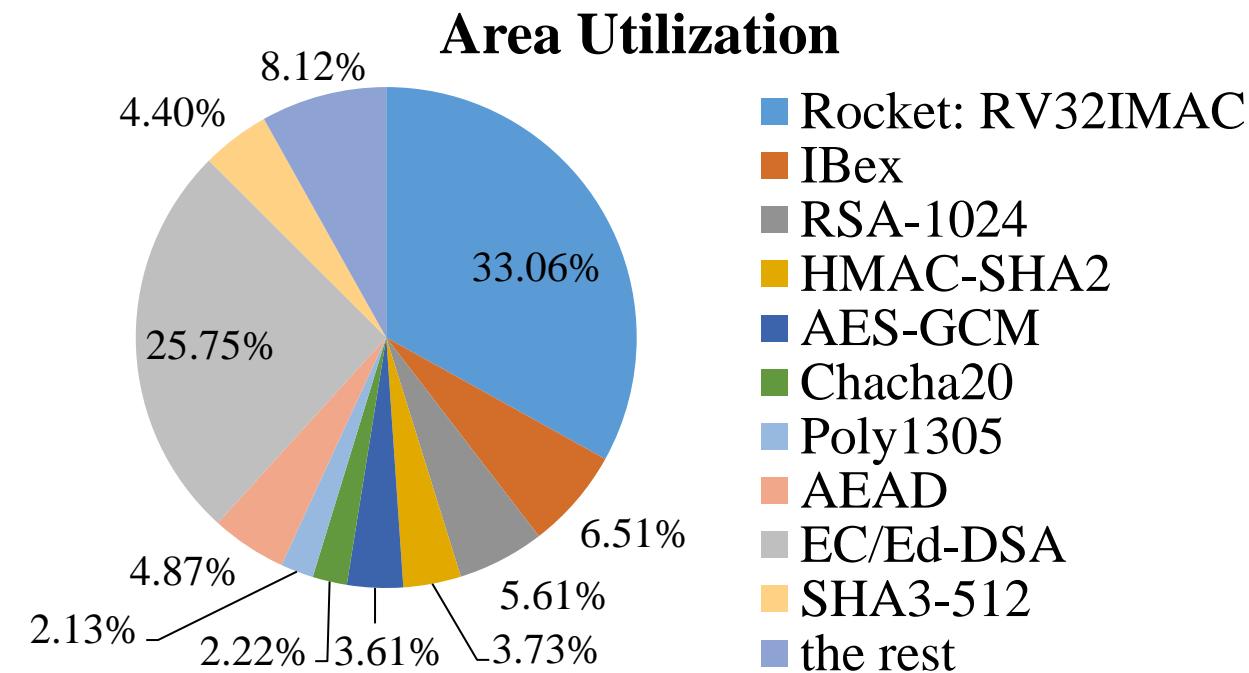


**\*Note:** “*the rest*” means all the buses, TRNG  
and utility-group peripherals such as GPIO, SPI, boot ROM, etc.

# 5. Main research @ UEC (8/30) RoT for TEE

SoC resources utilization pie chart:  
All crypto-cores + Single-core RV32IMAC Rocket

ROHM-180nm chip build report



**\*Note:** “*the rest*” means all the buses, TRNG  
and utility-group peripherals such as GPIO, SPI, boot ROM, etc.

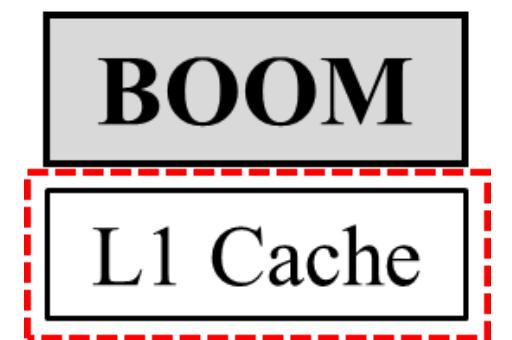
# 5. Main research @ UEC (9/30) Spectre attack?

Spectre: a Cache side-channel attack

Target: RISC-V Out-of-order (ex: BOOM)

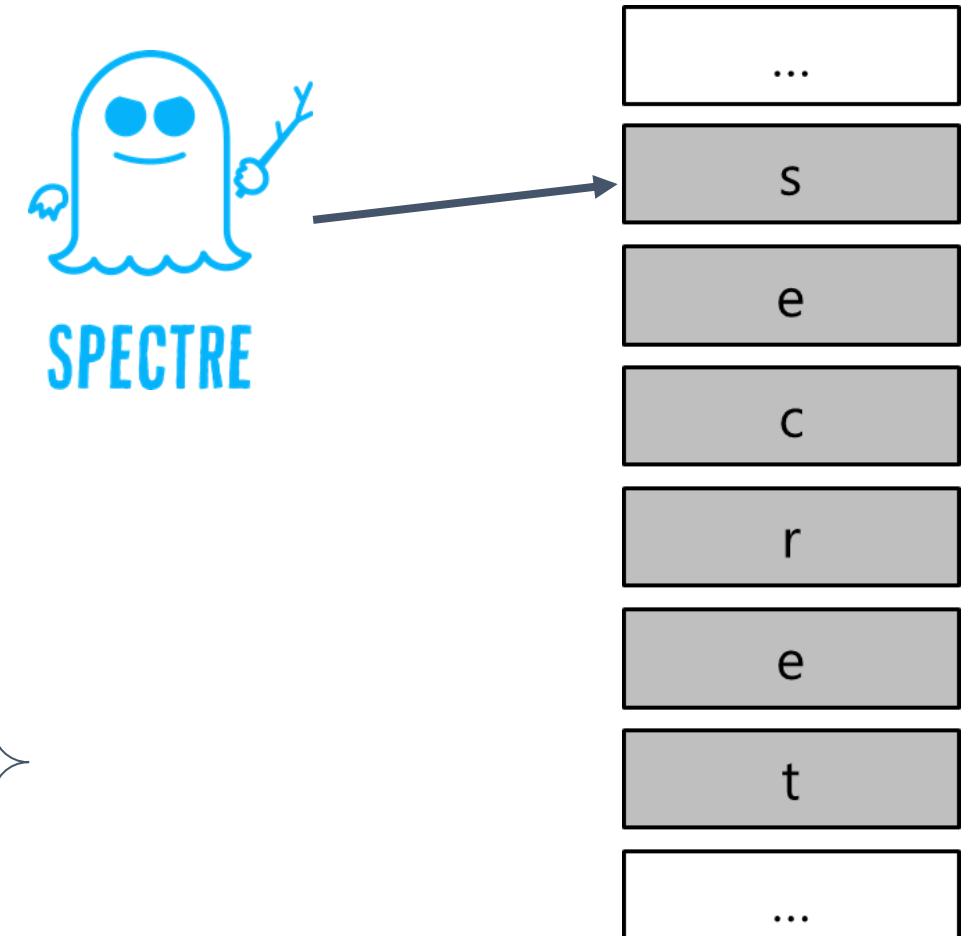
First variants:

- Spectre v1: Bound Check Bypass
- Spectre v2: Branch Target Injection



BOOM suitable for Spectre

- Branch Predictor Unit
- **Speculative Execution**
- Caching
- ...



Cache memory

# 5. Main research @ UEC (10/30) Spectre attack?

Speculative execution example:

User input

Process

a = 1

**TRUE** => Execute B

a = 2

**TRUE** => Execute B

a = 3

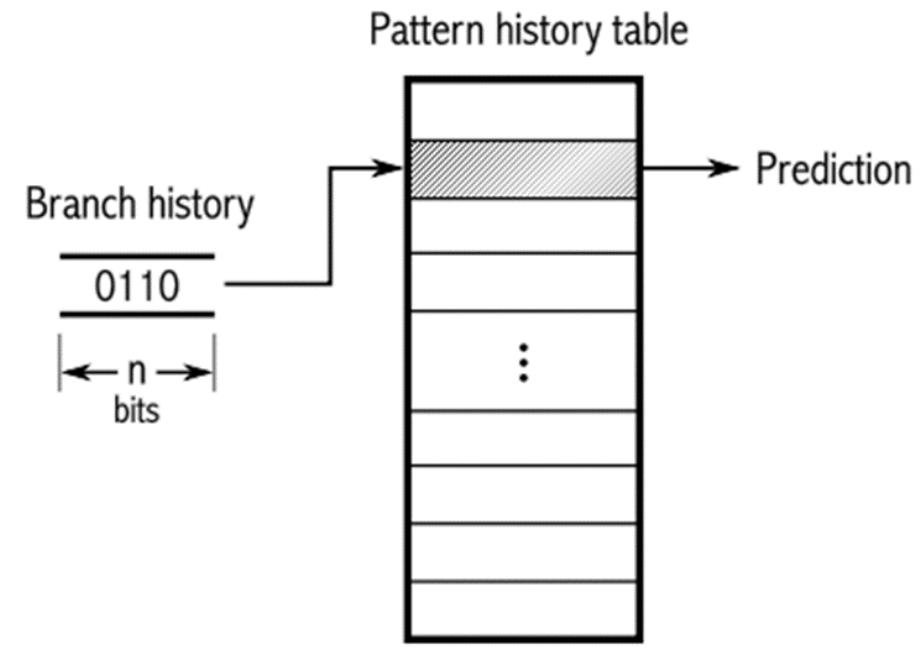
**TRUE** => Execute B

...

a = X

*Guess as* **TRUE** => Execute B

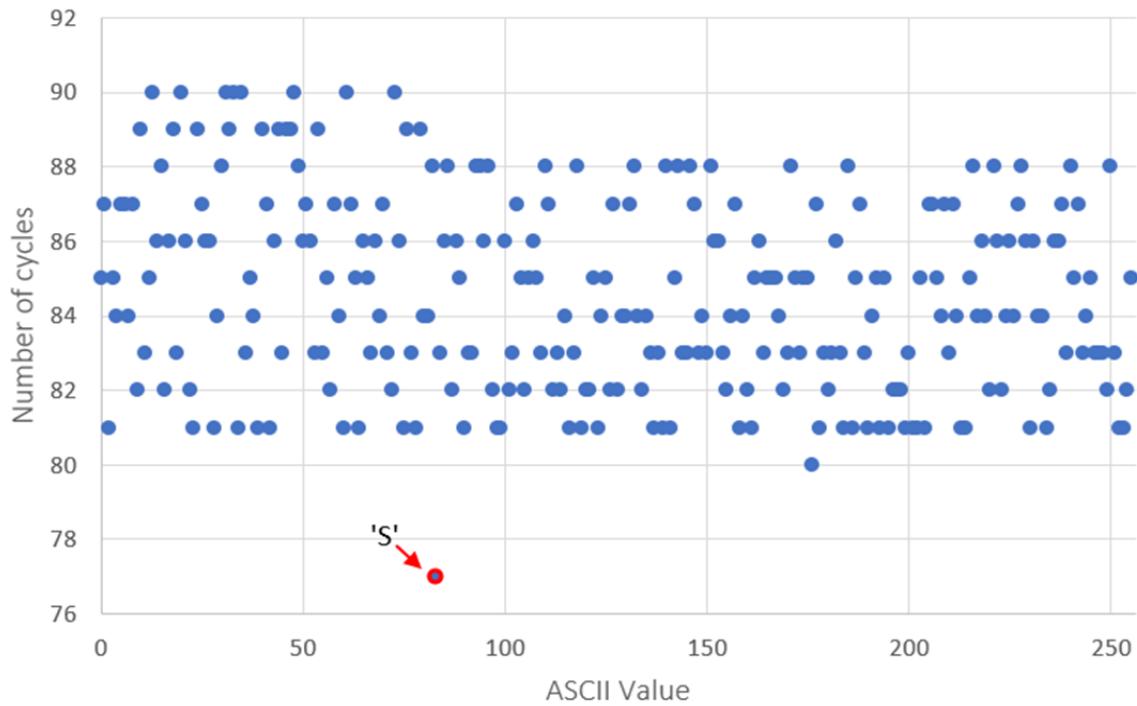
IF (a < 10)  
Run B



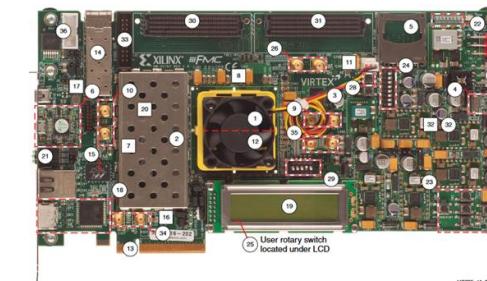
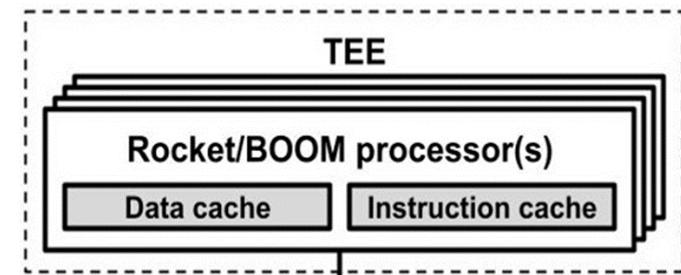
# 5. Main research @ UEC (11/30) Spectre attack?

Implement RISC-V processor

- BOOM (out-of-order) core: **exploited**
- Rocket (in-of-order) core: **cannot exploited**



Observe cache accessing time after  
an attack attempt

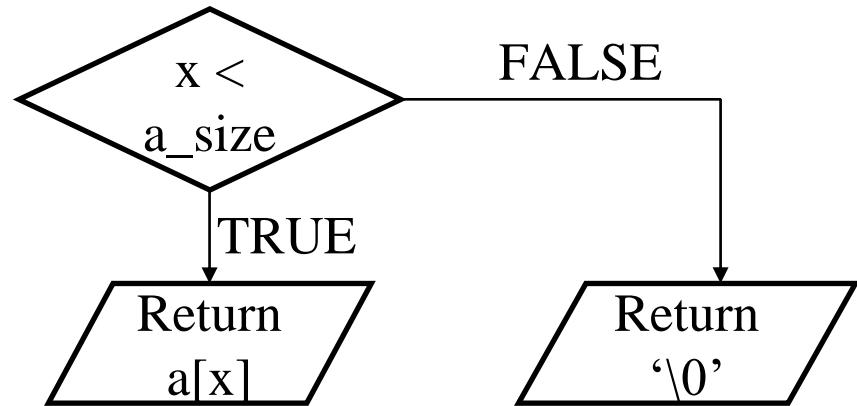


FPGA VC707

char(S)	guess_char(hits, score, value)	1.(3, 83, S)
char(e)	guess_char(hits, score, value)	1.(9, 101, e)
char(c)	guess_char(hits, score, value)	1.(7, 99, c)
char(r)	guess_char(hits, score, value)	1.(8, 114, r)
char(e)	guess_char(hits, score, value)	1.(8, 101, e)
char(t)	guess_char(hits, score, value)	1.(9, 116, t)
char( )	guess_char(hits, score, value)	1.(10, 32, )
char(K)	guess_char(hits, score, value)	1.(8, 75, K)
char(e)	guess_char(hits, score, value)	1.(8, 101, e)
char(y)	guess_char(hits, score, value)	1.(8, 121, y)

Attack log (success case)

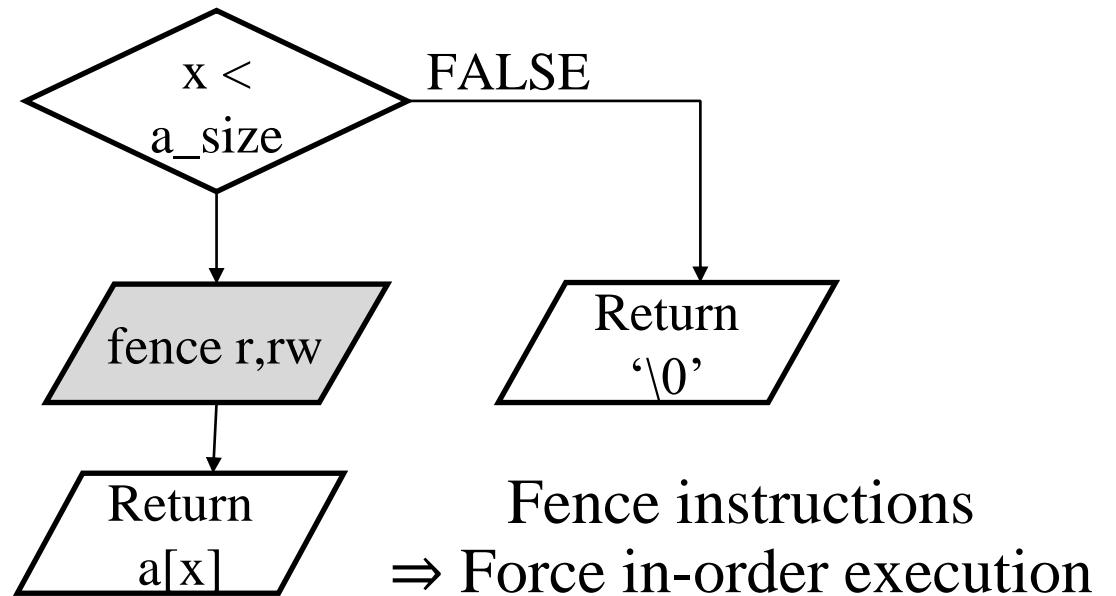
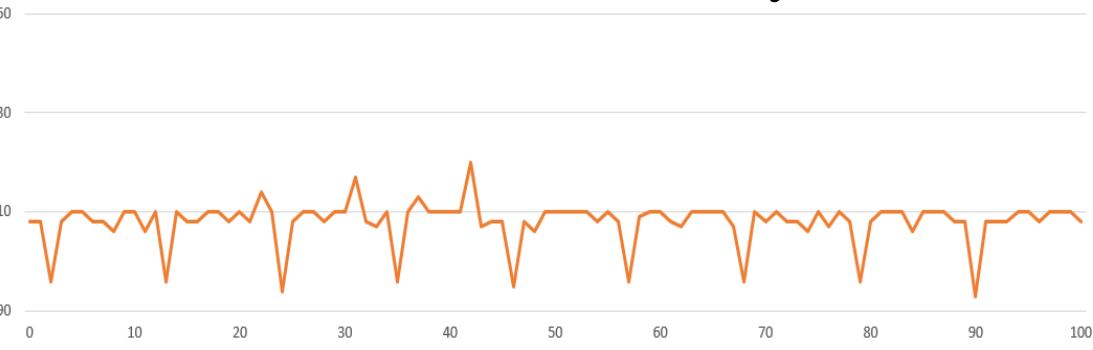
# 5. Main research @ UEC (12/30) Spectre software mitigate



Original code

No mitigation

- Normal execution cycle: 210



Fence instructions  
⇒ Force in-order execution



- Performance loss: 15 – 43% (242 – 290 cycles) 62

# 5. Main research @ UEC (13/30) Spectre hardware mitigate

## Hardware mitigation method: modifying MSHRs

- MSHRs: miss status holding registers
  - Located in The Load/Store Unit (LSU)
  - Handling data forwarding when mis-speculative events
- ⇒ Delay the data forwarding when mis-speculative

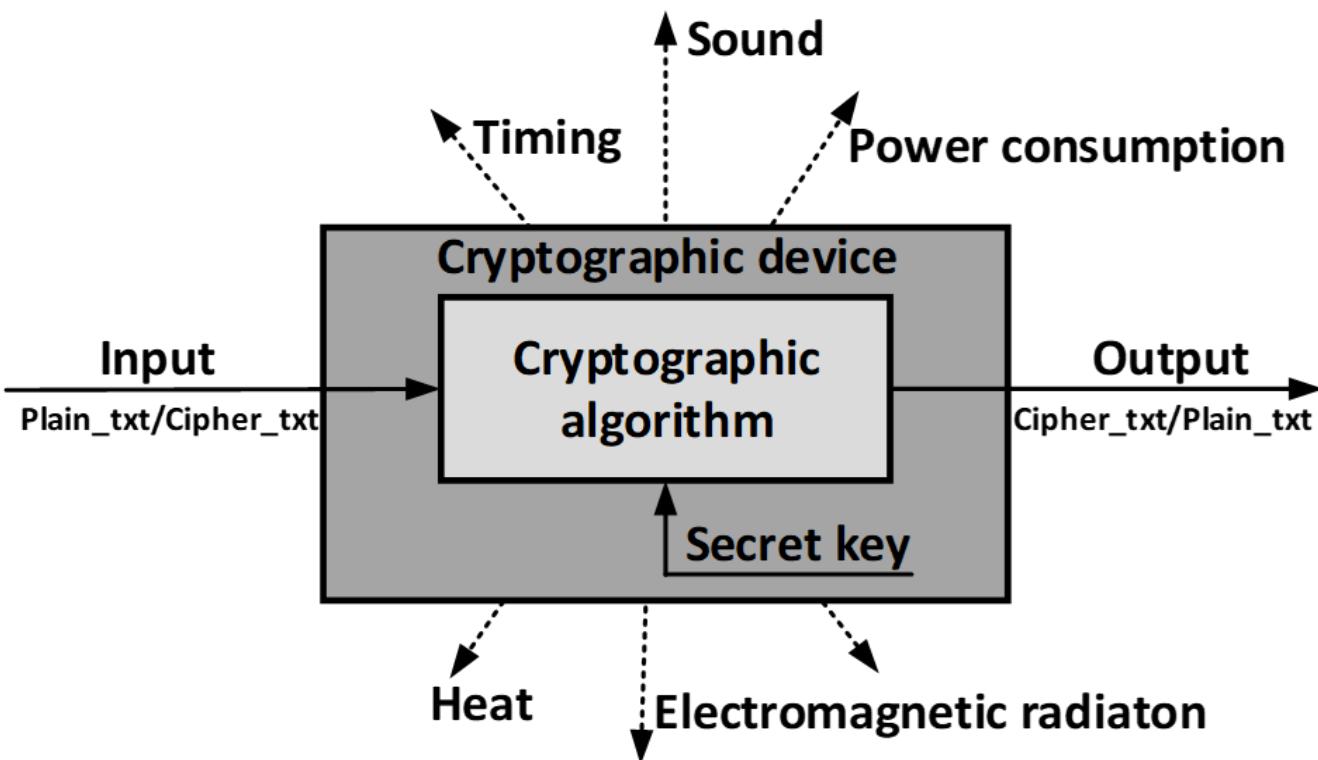


MSHRs resources utilization

Configuration	LUT	FF
Normal MSHR	1926	1120
Secure MSHR	1980 (2.8%)	1124 (0.4%)

# 5. Main research @ UEC (14/30) CPA?

A cryptographic device leaks side-channel information



## Side-channel attacks:

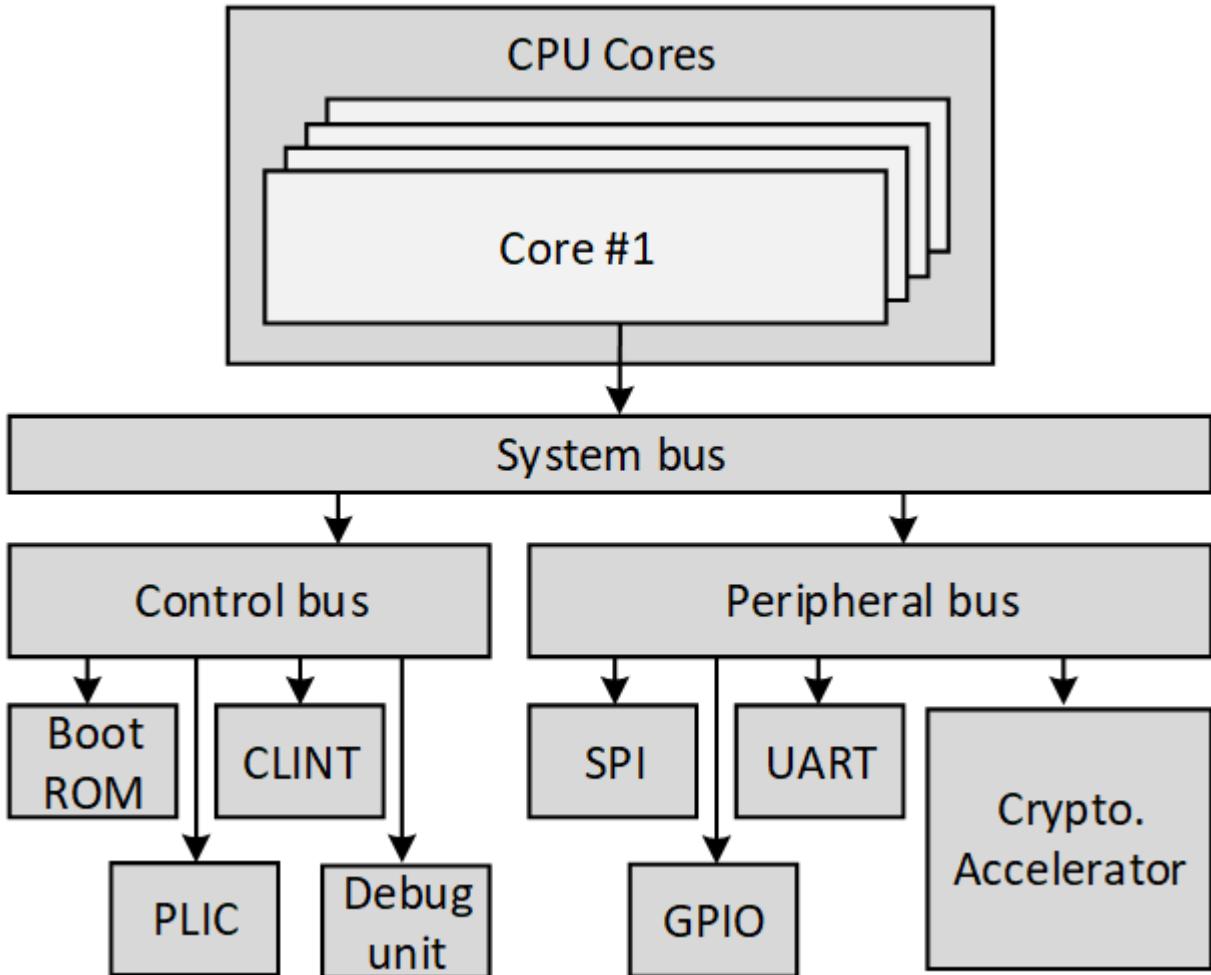
Exploit unavoidable side-channel information in cryptanalysis.

## Correlation Power Analysis (CPA) attacks:

Using Power consumption or Electromagnetic radiation.

# 5. Main research @ UEC (15/30) CPA countermeasure

Example of Cryptographic SoC



## Countermeasures for Cryptographic SoC:

Existing techniques are not suitable:

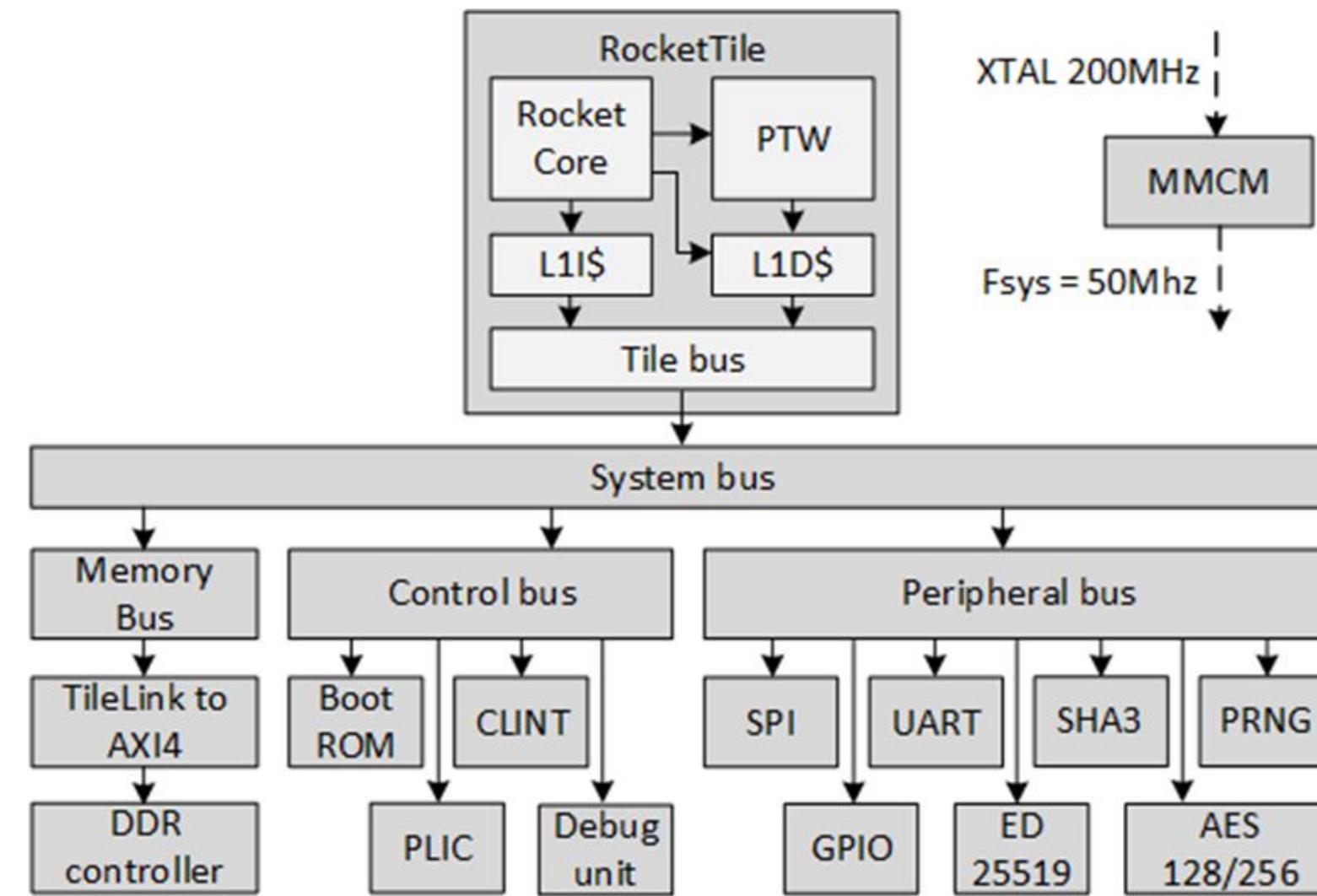
- Masking: Reduce performance, Increase power, area.
- Hiding: Huge hardware overheads.

## ⇒ Proposed Ideas:

- Randomly scale the clock freq. of Crypto.Acc. after each encryption/decryption.
- Only applied to the Crypto.Acc.
- Create as many Clock frequencies as possible.

# 5. Main research @ UEC (16/30) CPA countermeasure

Unprotected Cryptographic SoC (TEE-Hardware)

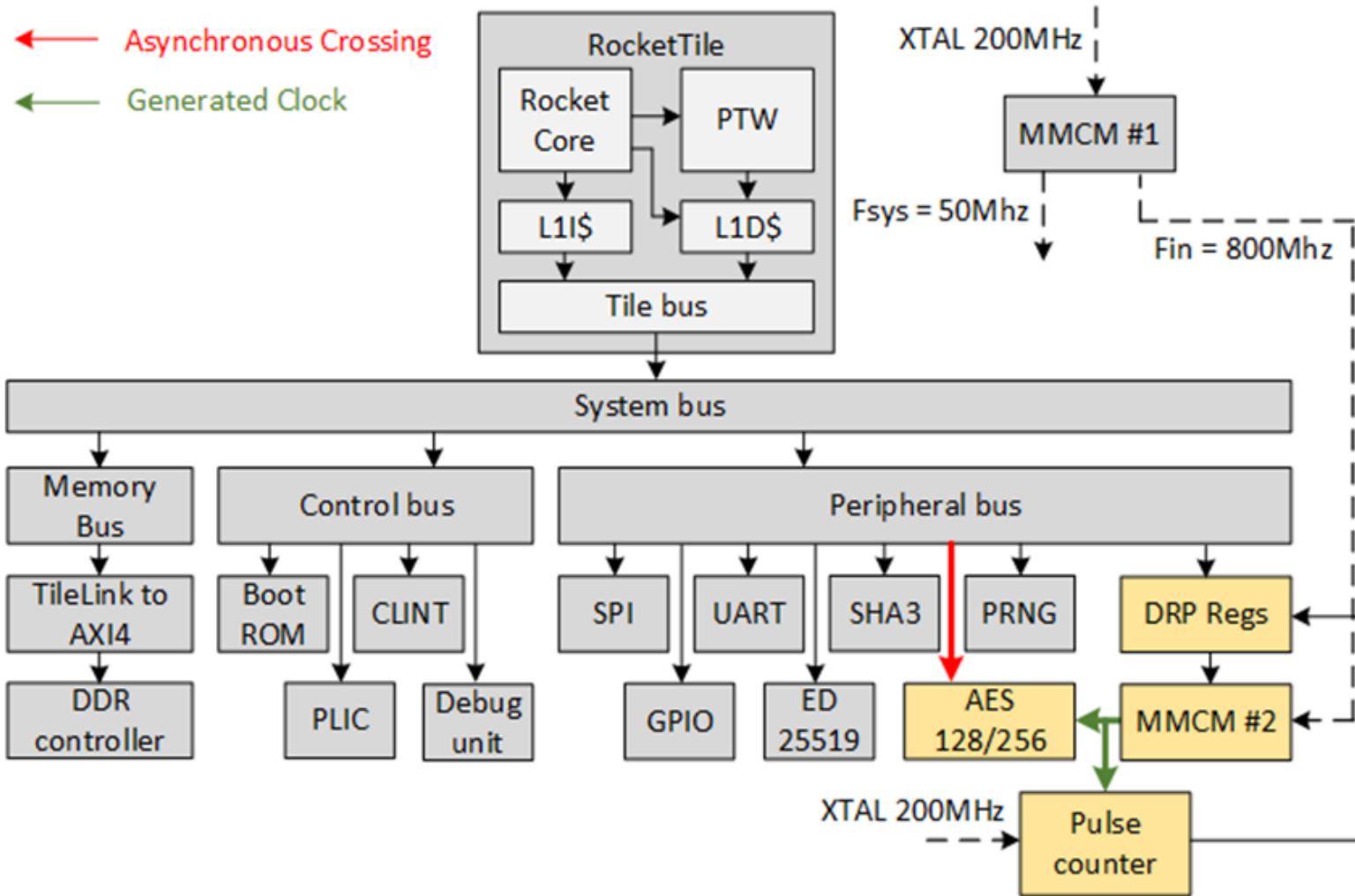


**Unprotected SoC (TEE-hardware):**

- 32-bit RISC-V SoC
- DDR Controller ⇒ support **Linux OS**
- Crypto. Accelerator:
  - **AES-128/256**
  - **SHA3**
  - **ED25519**
  - **PRNG**
- Fixed system Clock:  
 **$F_{sys} = 50\text{MHz}$**

# 5. Main research @ UEC (17/30) CPA countermeasure

## Protected Cryptographic SoC with RDFS



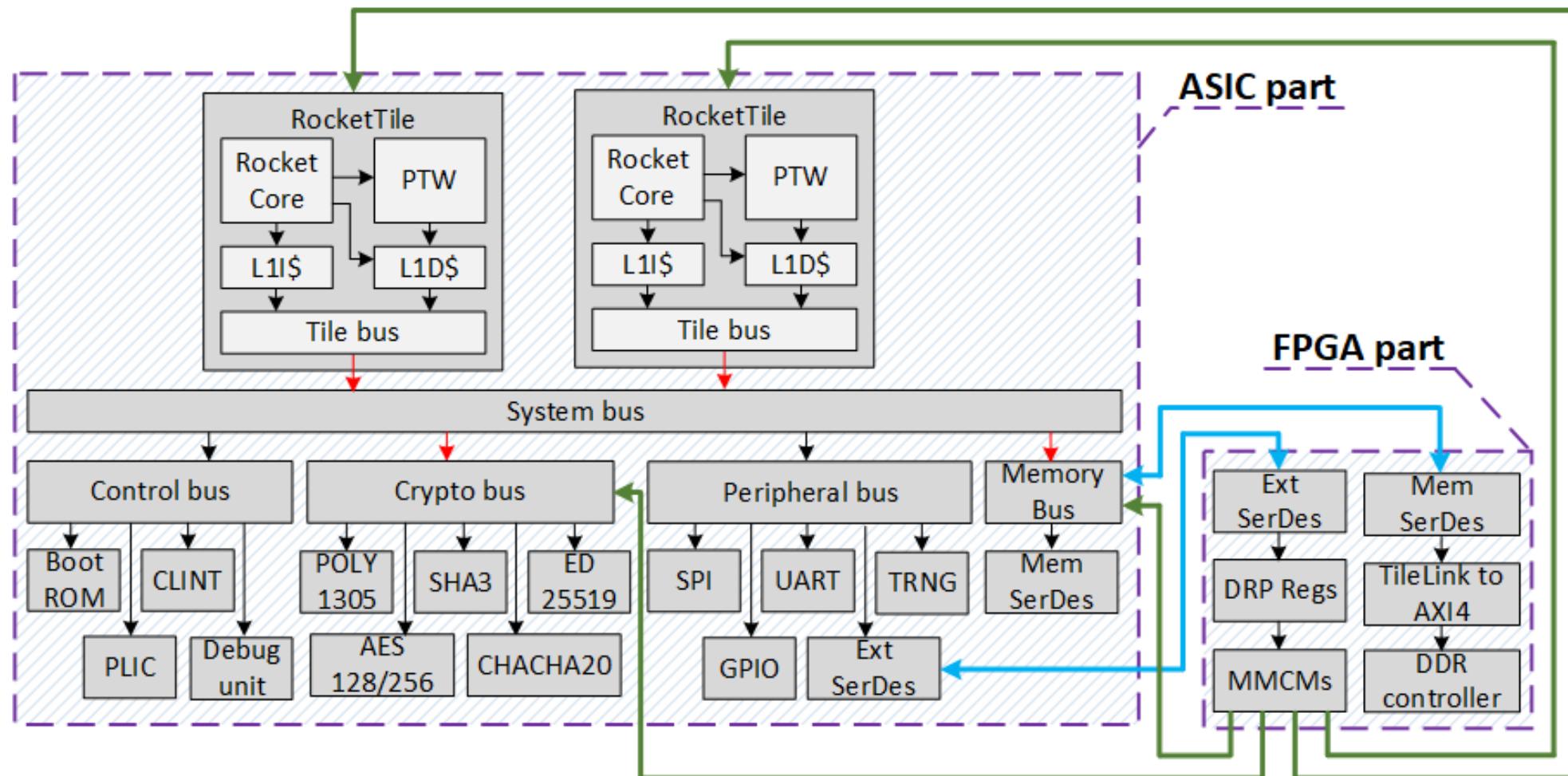
## Protected SoC with RDFS:

- Add Clock Generation peripherals (use Xilinx's Clock Manager IP)
- Create **> 219.000 frequencies** (in range from **50MHz** to **100MHz**)
- Verify accuracy by **Pulse counter**
- Only applied to AES-128 module
- Scale AES's CLK **after each encryption**

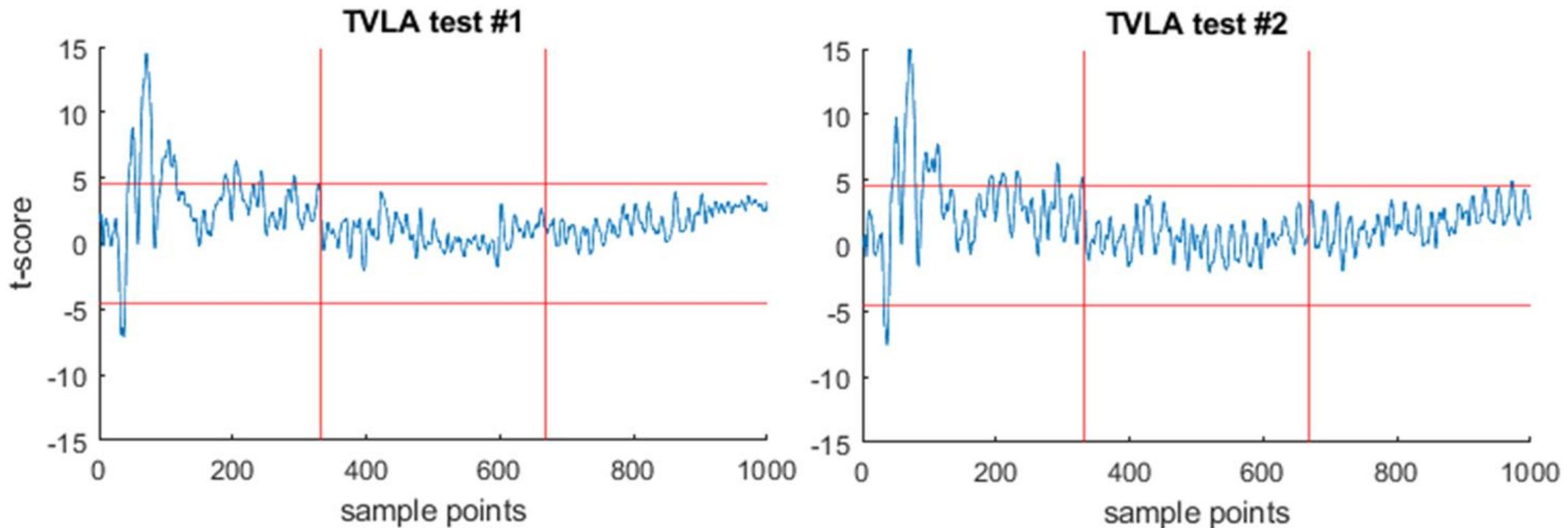
# 5. Main research @ UEC (18/30) CPA countermeasure

- ↔ Serializer/Deserializer data bus
- ← Asynchronous crossing
- ← Synchronous crossing
- ← Generated Clock signal

Chip version



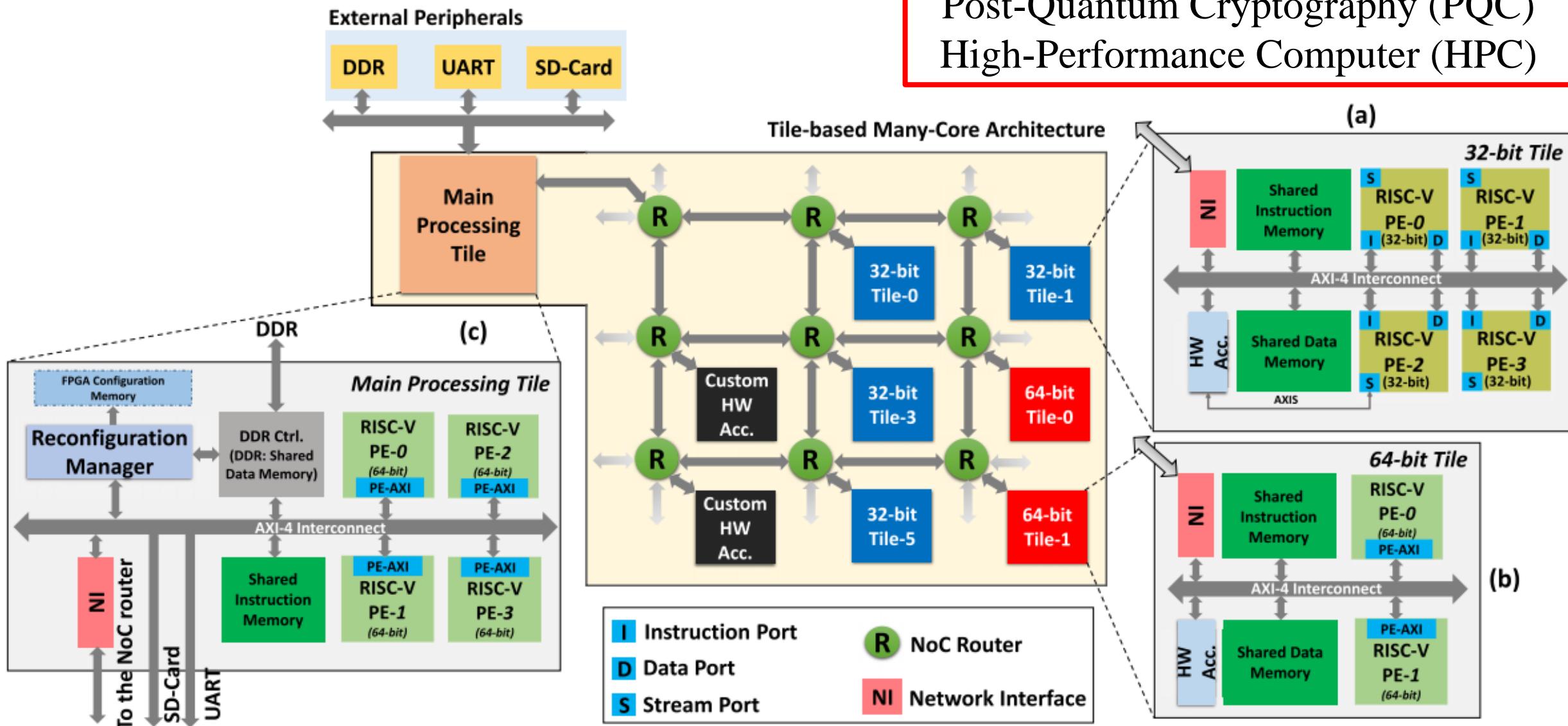
## 5. Main research @ UEC (19/30) CPA countermeasure



### Test Vector Leakage Assessment (TVLA) results:

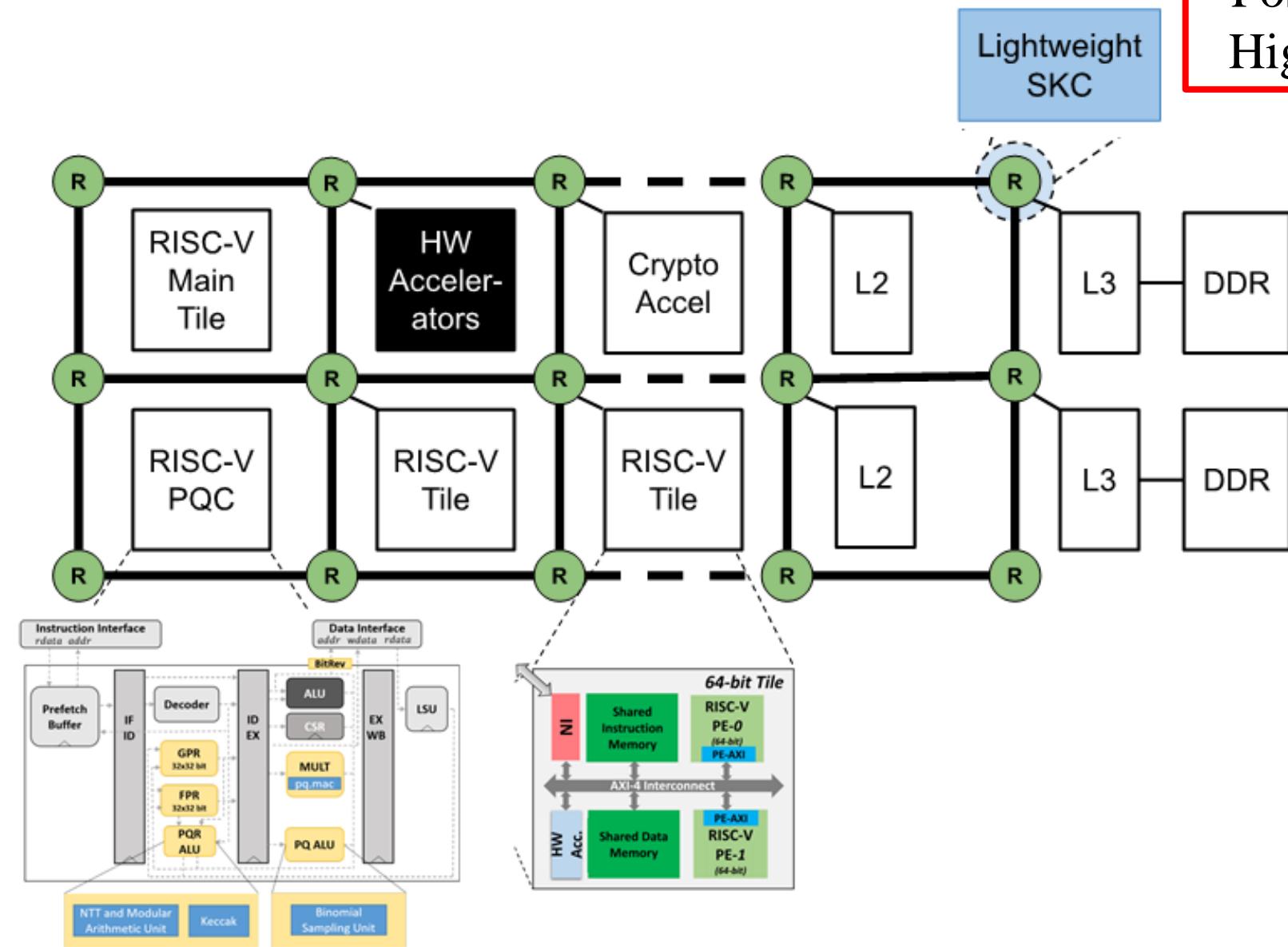
- RDFS with **219,412** clk freq. (**50MHz - 100MHz**)
- Does not detect any leakage in **5 million** power traces

# 5. Main research @ UEC (20/30) PQC-HPC (*future project*)



# 5. Main research @ UEC (21/30) PQC-HPC (*future project*)

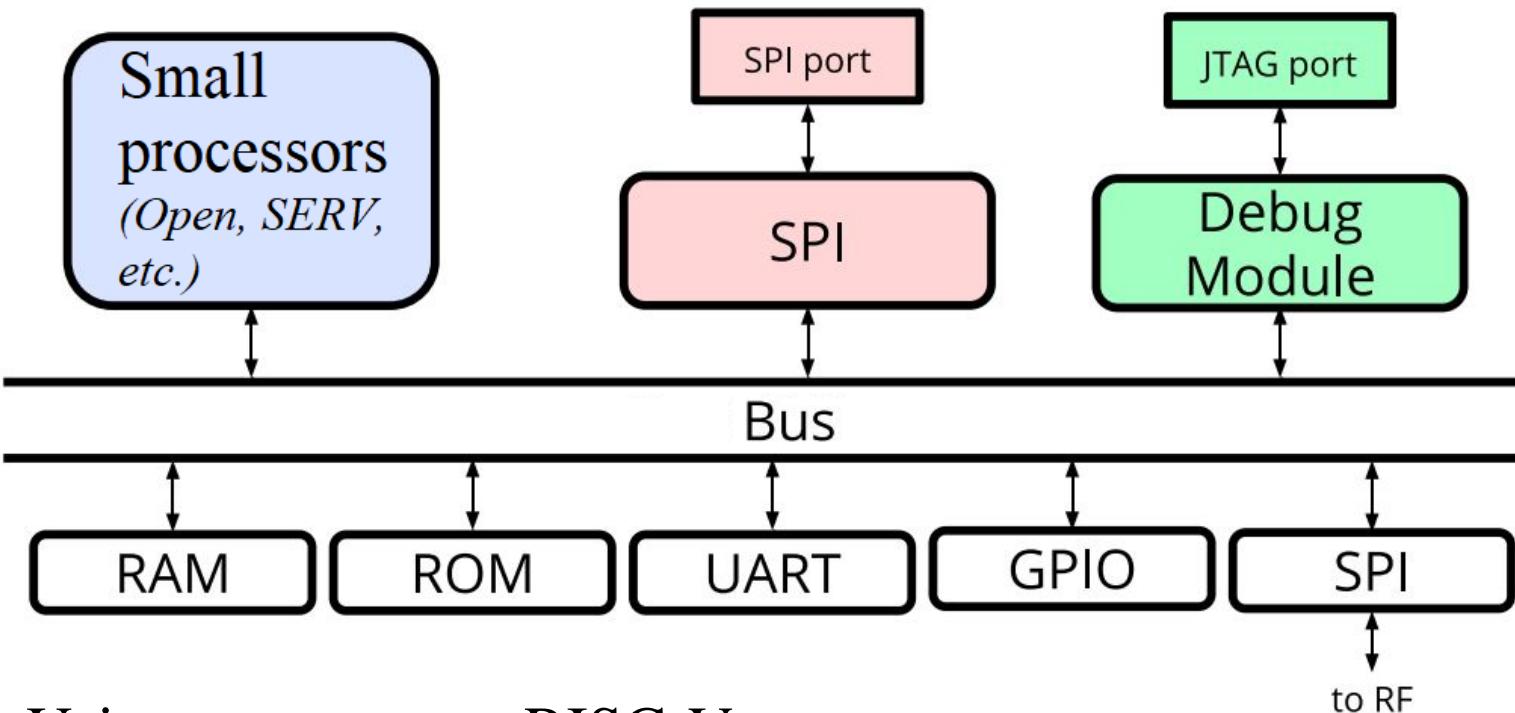
Post-Quantum Cryptography (PQC)  
High-Performance Computer (HPC)



- RISC-V main tile including a Network Interface (NI) to connect with NoC.
- Lightweight Symmetric Key (SKC) is applied to Router/NI.
- PQCs integrated inside RISC-V cores.
- Hardware accelerators
- Exploit special RISC-V ISA extensions for PQC.

# 5. Main research @ UEC (22/30) ULP-SoC

Modular for Ultra-Low Power (ULP) IoT-SoC architecture based on RISC-V



Using open-source RISC-V, we can:

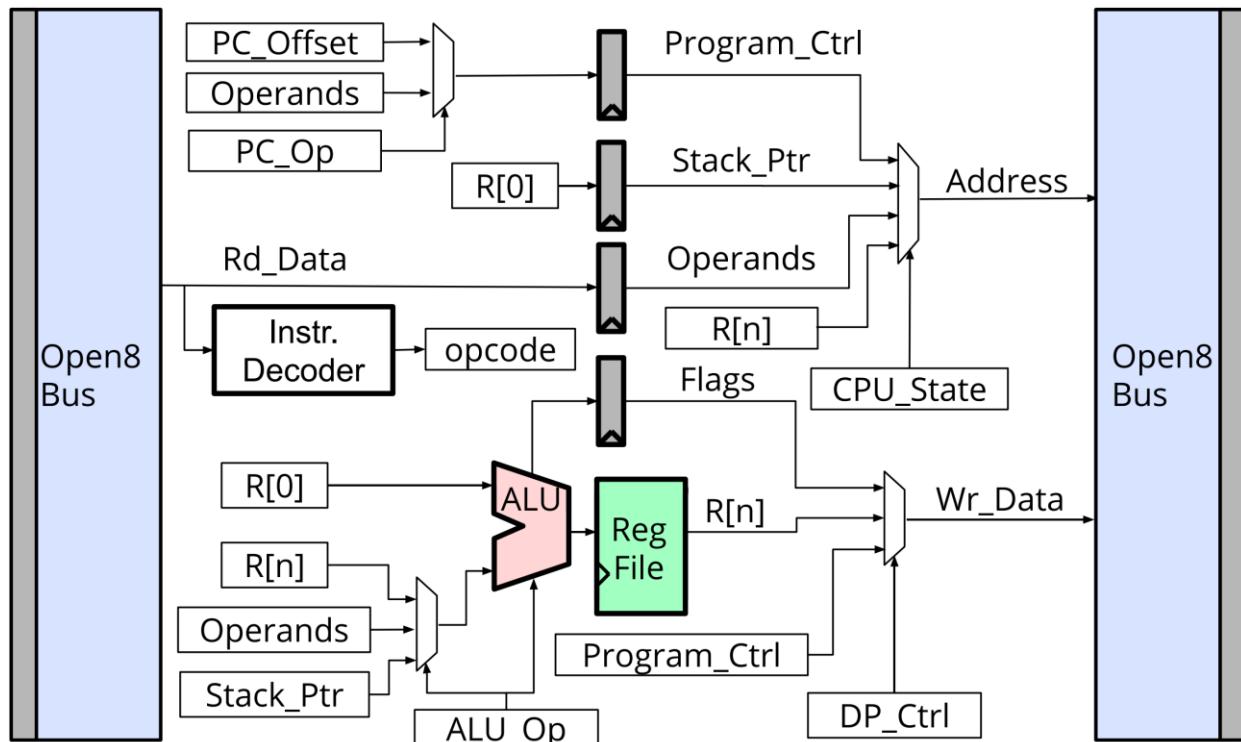
- Cherry-pick the wanted ISA extensions.
- Multiple options for the core processor(s) with wanted features (*i.e., low-power, high-performance, and security*).
- Easy to develop & debug software.
- Vast options of peripherals: open sources, commercial IPs, in-house developments. 72

Open-source modular framework aiming for:

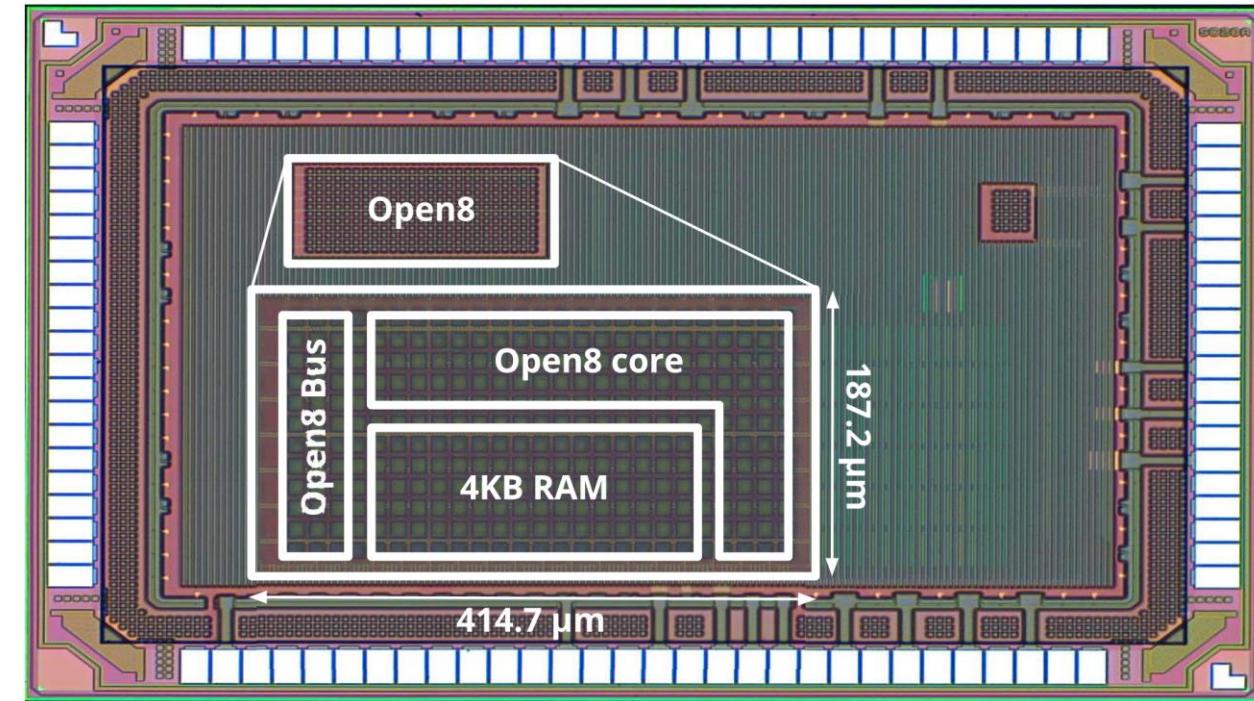
- Highly customizable
- Flexible and portable
- Wide range of applications

# 5. Main research @ UEC (23/30) ULP-SoC

Experiment with small MCUs:  
*try to achieve sub- $\mu$ W power consumption*



Open8 core data-path

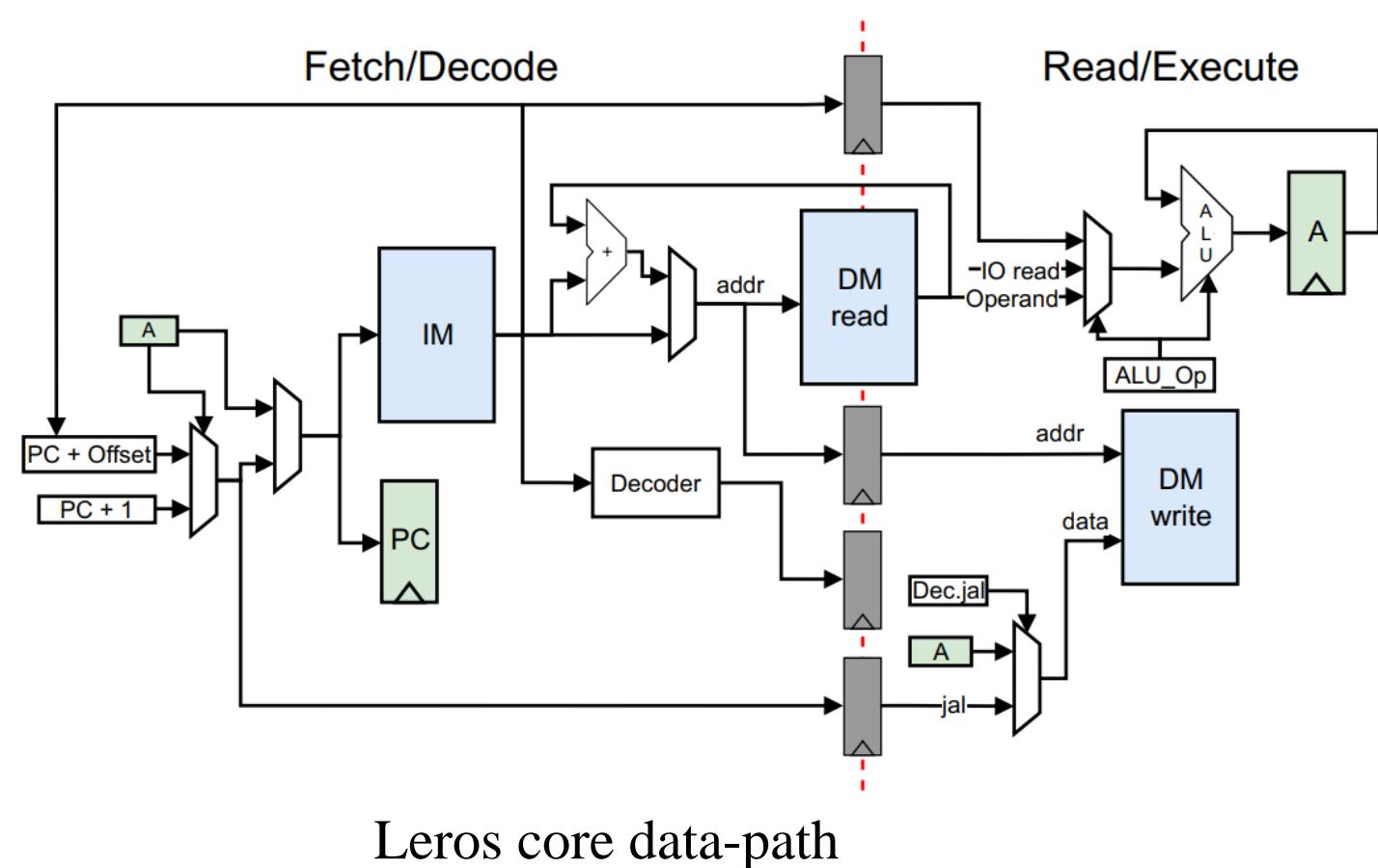


Open8 in SOTB-65nm

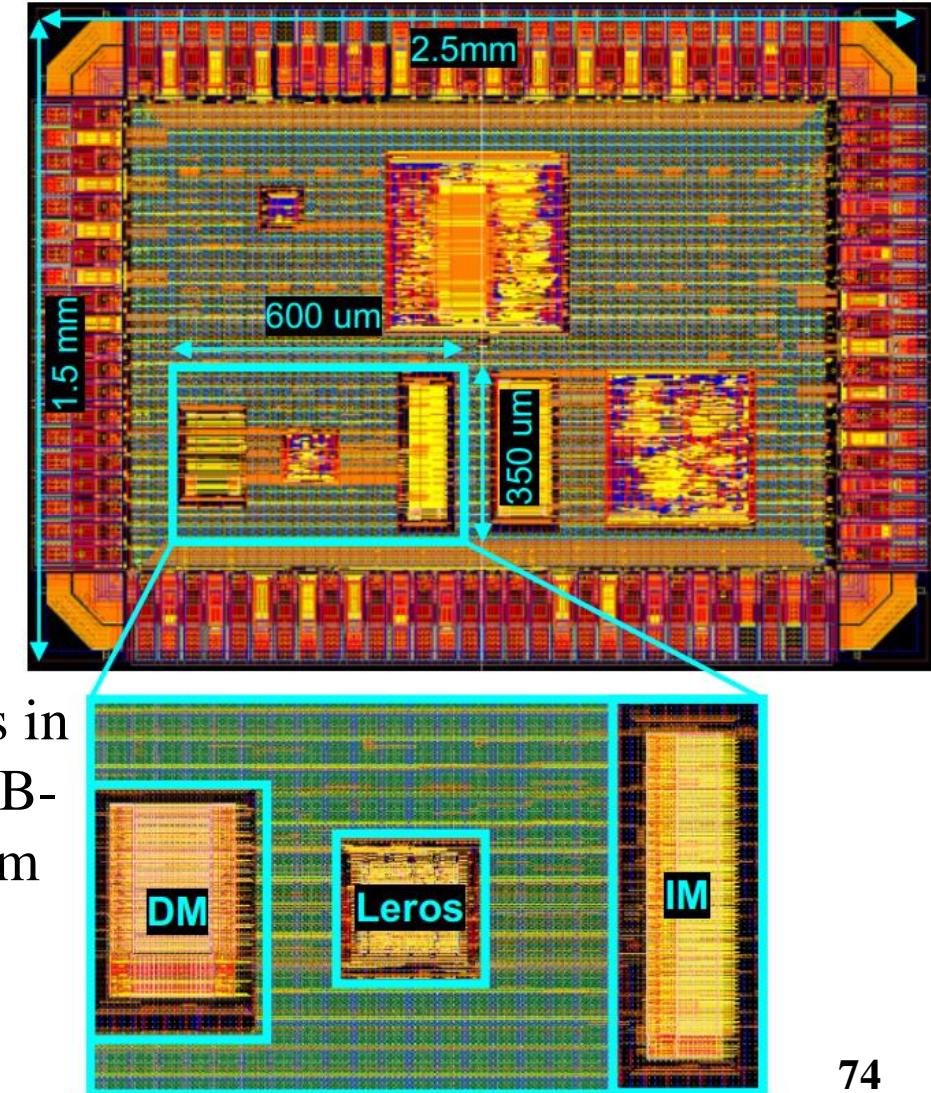
*8-bit RISC processor*

# 5. Main research @ UEC (24/30) ULP-SoC

Experiment with small MCUs:  
*try to achieve sub- $\mu$ W power consumption*

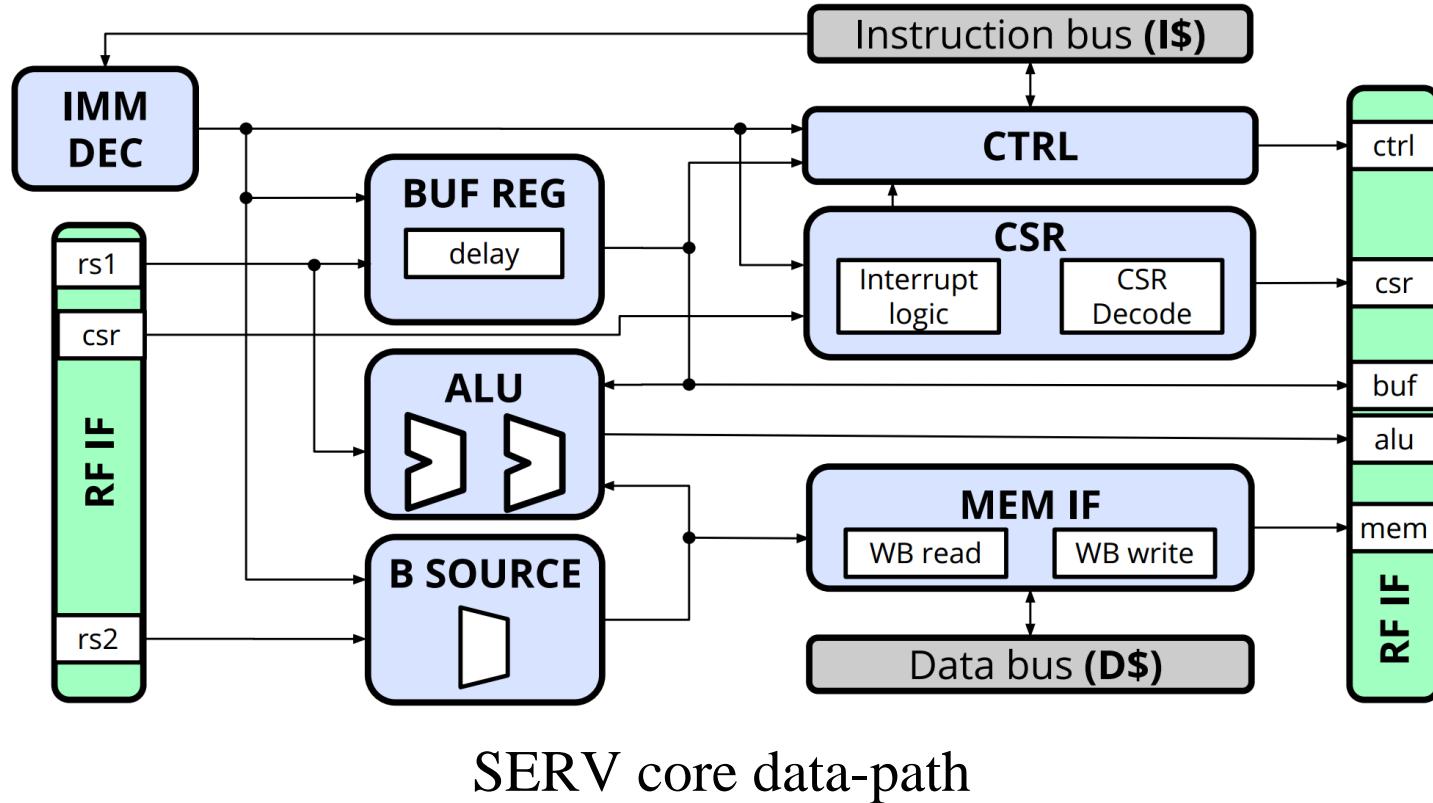


16-bit RISC processor

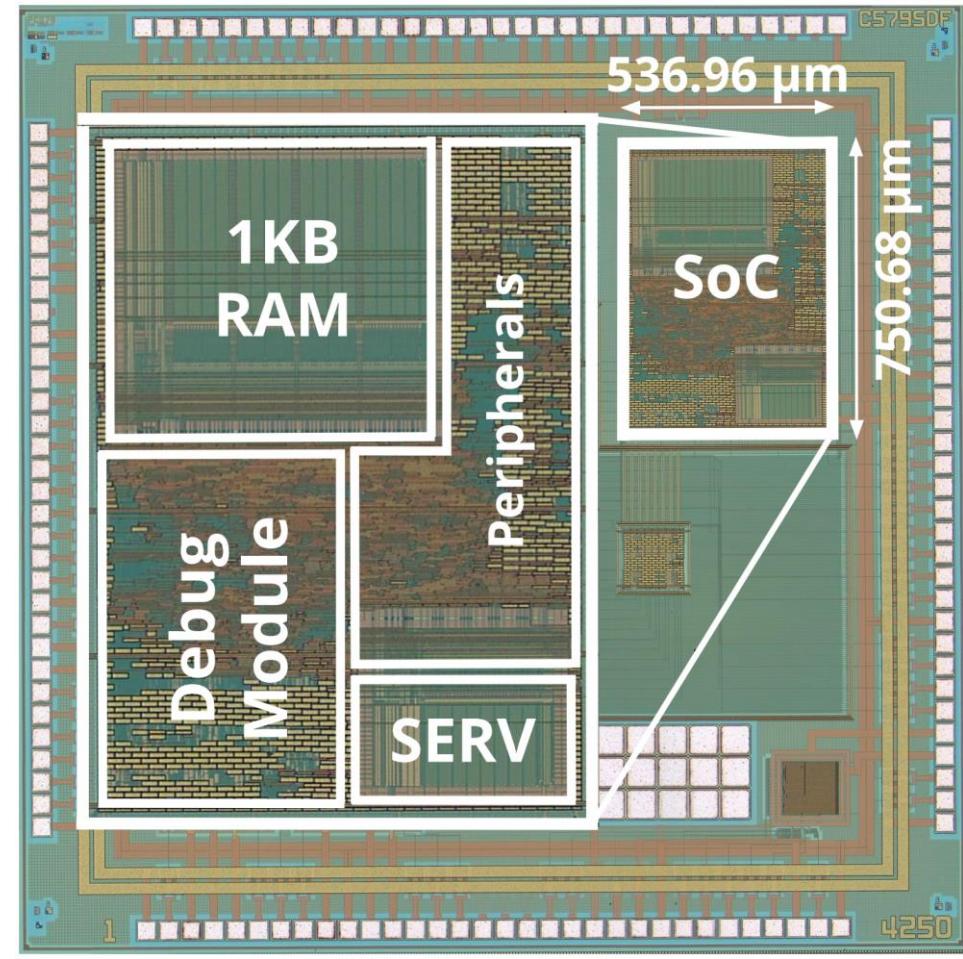


# 5. Main research @ UEC (25/30) ULP-SoC

Experiment with small MCUs:  
*try to achieve sub- $\mu$ W power consumption*



32-bit RISC-V serial processor

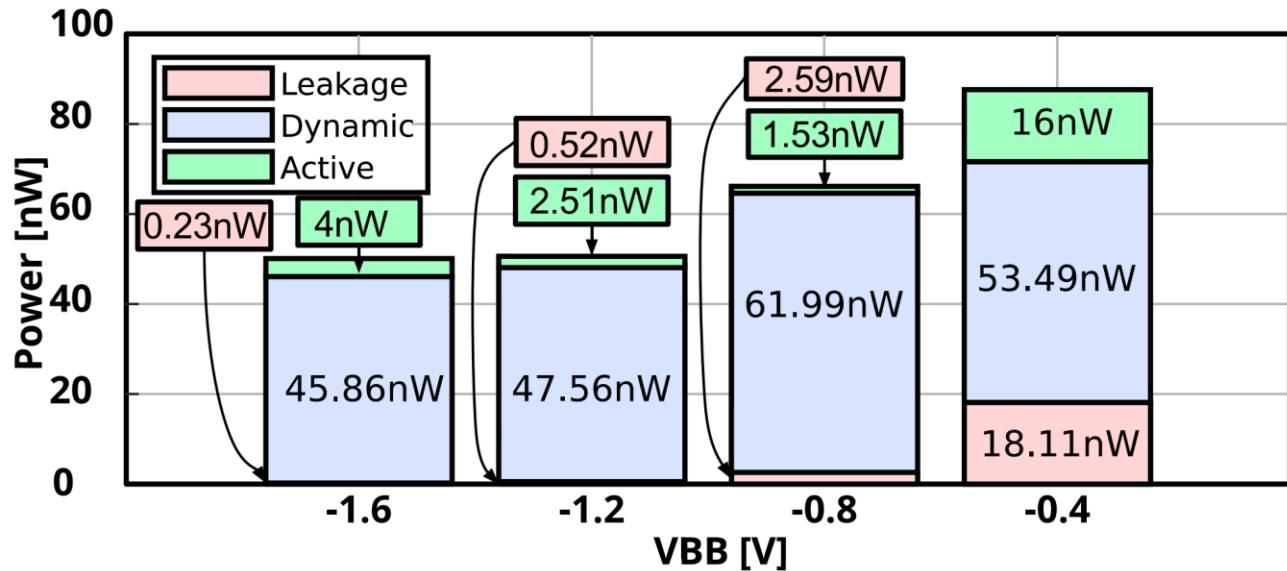


SERV in ROHM-180nm

# 5. Main research @ UEC (26/30) ULP-SoC

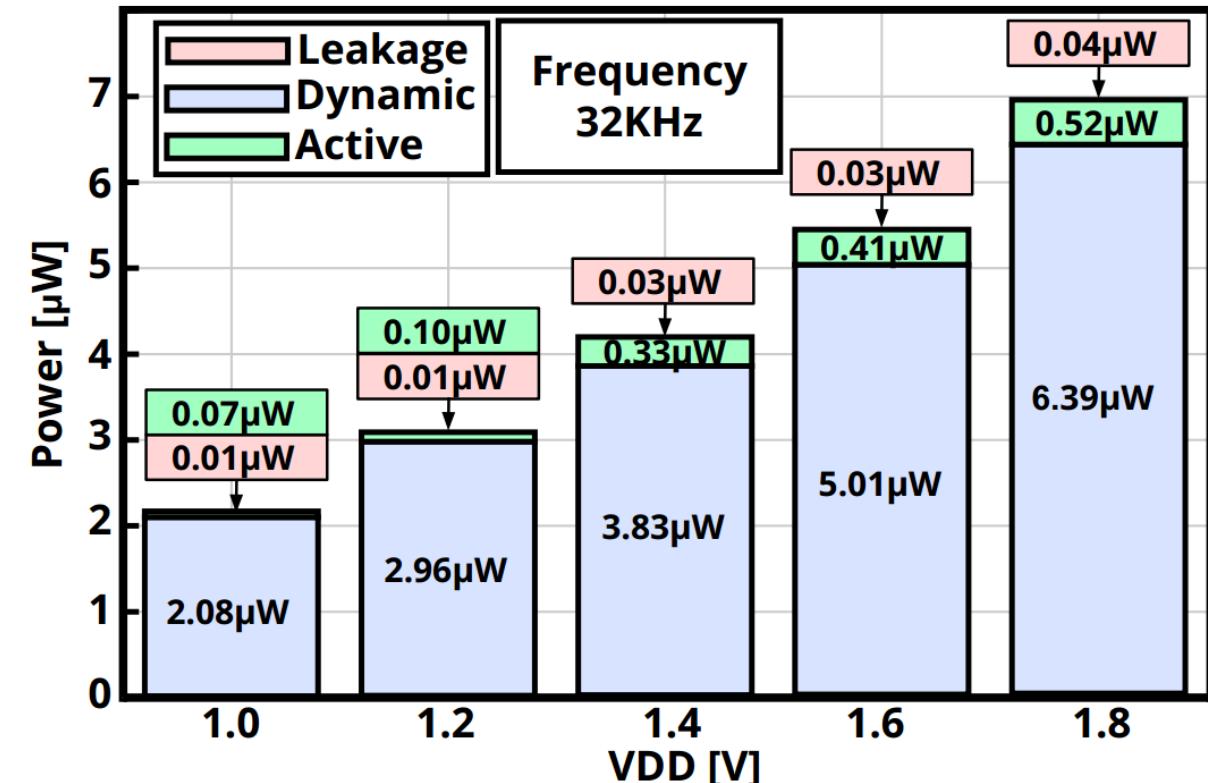
Experiment with small MCUs:

*try to achieve sub- $\mu$ W power consumption*



Open8 in SOTB65  
(fixed  $V_{DD} = 0.5\text{-V}$  &  $F = 32\text{-KHz}$ )

Best P = 46.13nW @  $V_{DD} = 0.5\text{-V}$ ,  $V_{BB} = -1.6\text{-V}$  &  $F = 32\text{-KHz}$

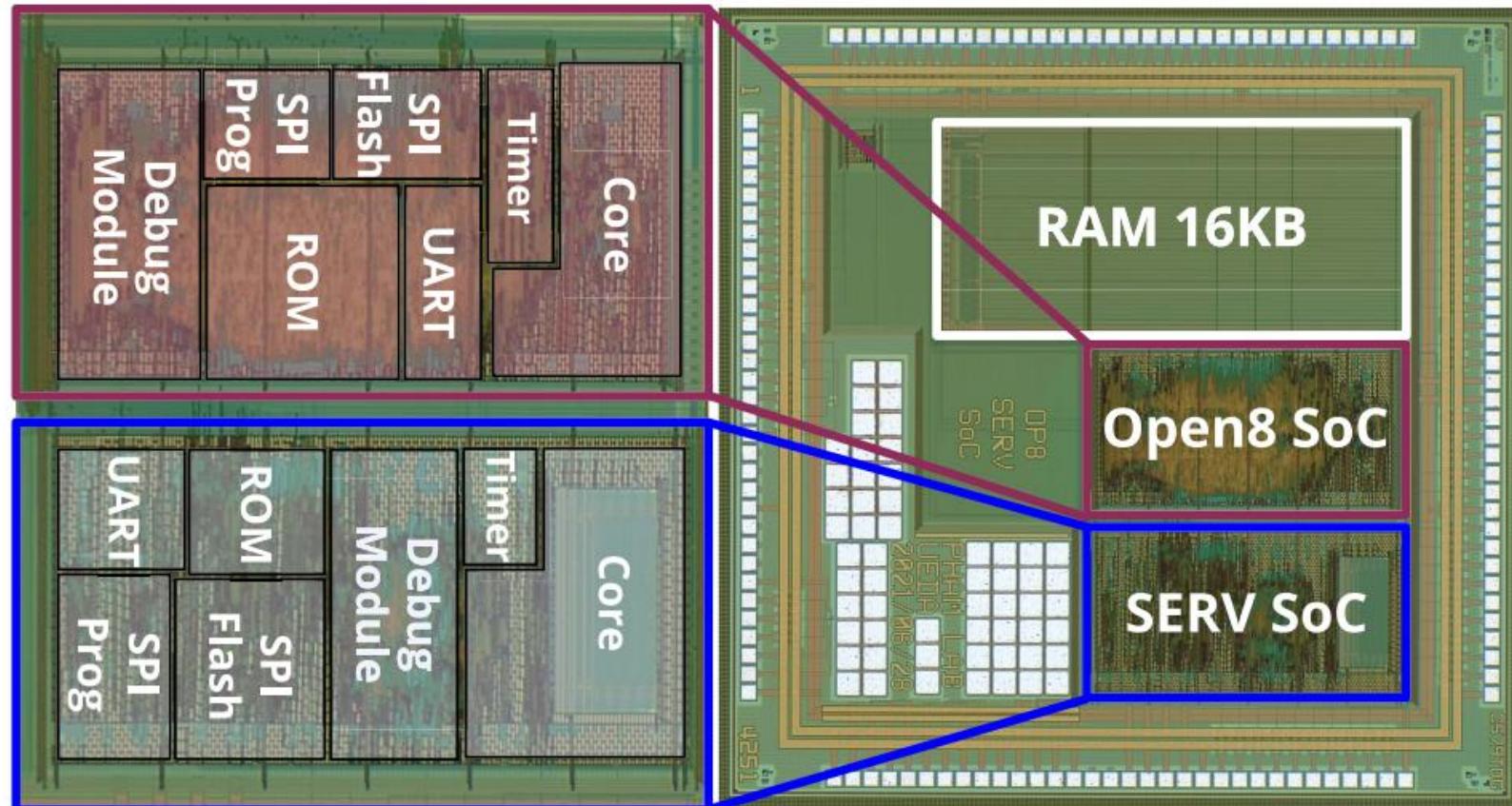
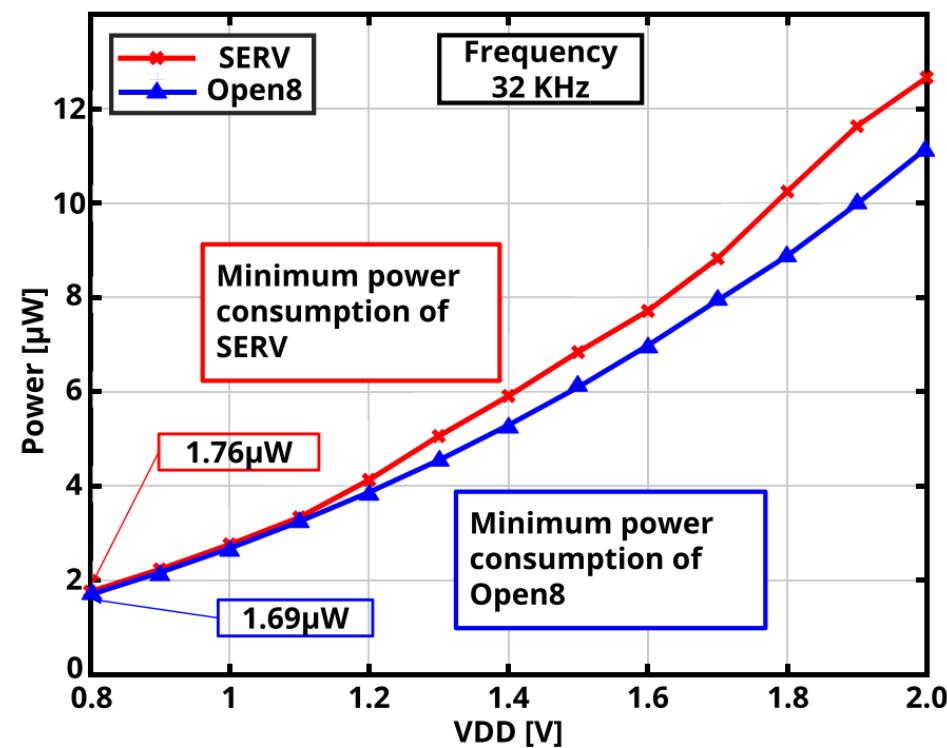


SERV in ROHM180nm  
Best P = 2.16 $\mu$ W  
@  $V_{DD} = 1.0\text{-V}$  &  $F = 32\text{-KHz}$

# 5. Main research @ UEC (27/30) ULP-SoC

Experiment with small MCUs:  
*try to achieve sub- $\mu$ W power consumption*

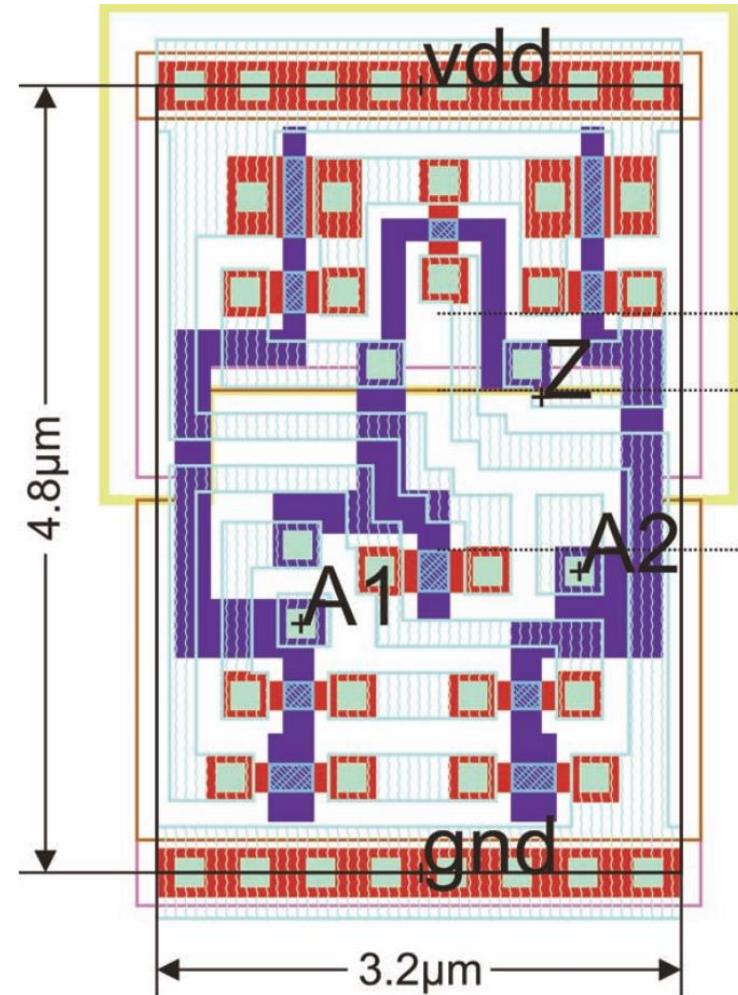
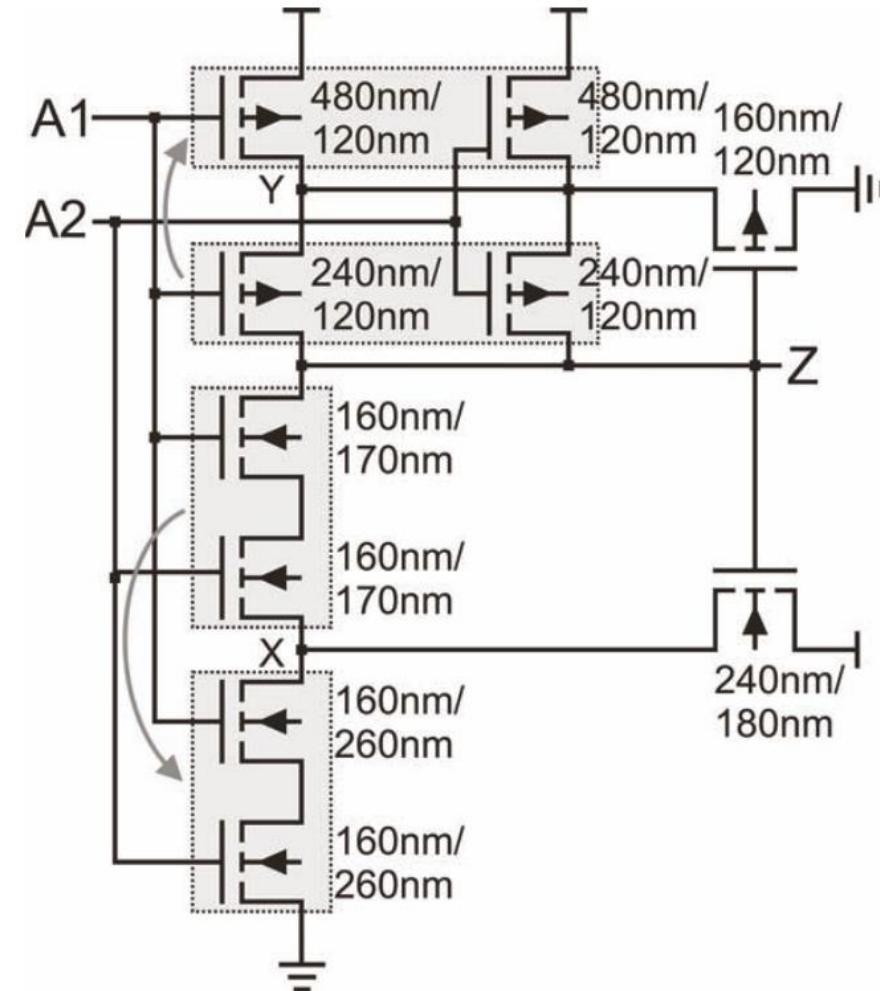
The dual-core of Open8 & SERV in this chip is just for comparison fairness.



# 5. Main research @ UEC (28/30) ULP-SoC (*future project*)

Develop Ultra-Low-Power (ULP) standard cell

Design a ULP standard cells library: intentionally trade-off  $F_{Max}$  for low power



Schematic of the NAND2 gate

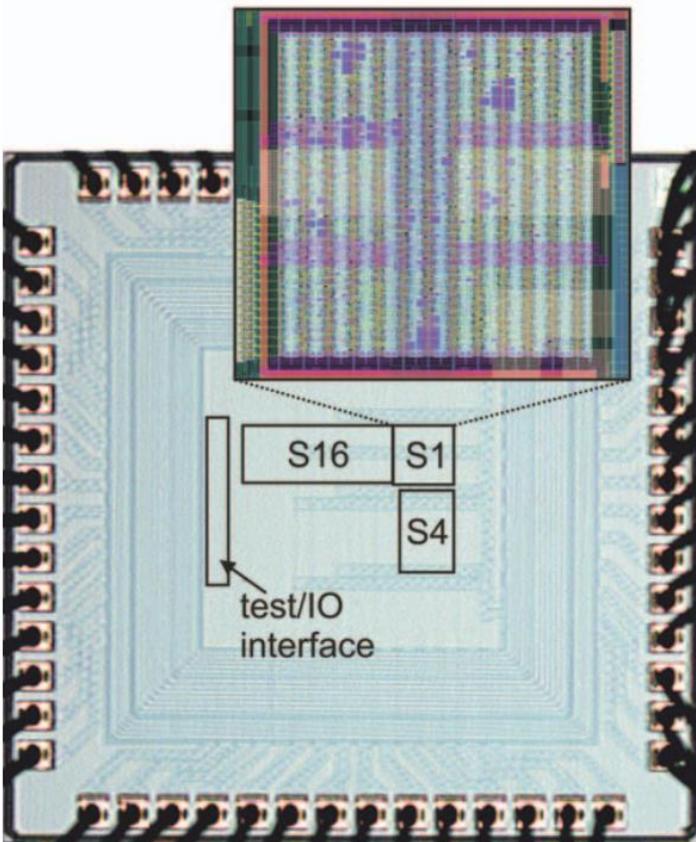
Layout of the NAND2 gate

Process: 0.13μm standard CMOS

spacings to limit PMOS/NMOS well proximity effect

Design idea: standard CMOS with Schmitt-trigger-like design

# 5. Main research @ UEC (29/30) ULP-SoC (*future project*)



Process	0.13µm 8-metal CMOS	
Design	8x8 Multiplier	
Area (S1/S4/S16)	19176/23978/45002µm <sup>2</sup>	
Supply Voltage (S1/S4/S16)	min. 84/68/62mV	max. 1.2V
Minimum Energy/Operation Point		
Frequency	S1	544.8kHz@260mV
	S4	328.3kHz@233mV
	S16	210.1kHz@226mV
Power	S1	346.6nW@260mV
	S4	243.8nW@233mV
	S16	275.6nW@226mV
Energy/ Operation	S1	0.63pJ@260mV
	S4	0.74pJ@233mV
	S16	1.31pJ@226mV
Minimum Supply Voltage		
Frequency	S1	15.20kHz@84mV
	S4	10.35kHz@68mV
	S16	5.15kHz@62mV
Power	S1	29.9nW@84mV
	S4	21.1nW@68mV
	S16	17.9nW@62mV
Energy/ Operation	S1	1.97pJ@84mV
	S4	2.04pJ@68mV
	S16	3.48pJ@62mV

Source: [ISSCC](#) (2011)

509.6pW/MHz-GE@0.26-V

Scaling equations in [Integration](#) (2017)

345.14pW/MHz-GE  
@0.5-V (*SOTB65 no bias*)

Open8 [26]  
(9kGE)

SOTB-65nm: 99.4nW  
@0.5-V no bias & 32-KHz

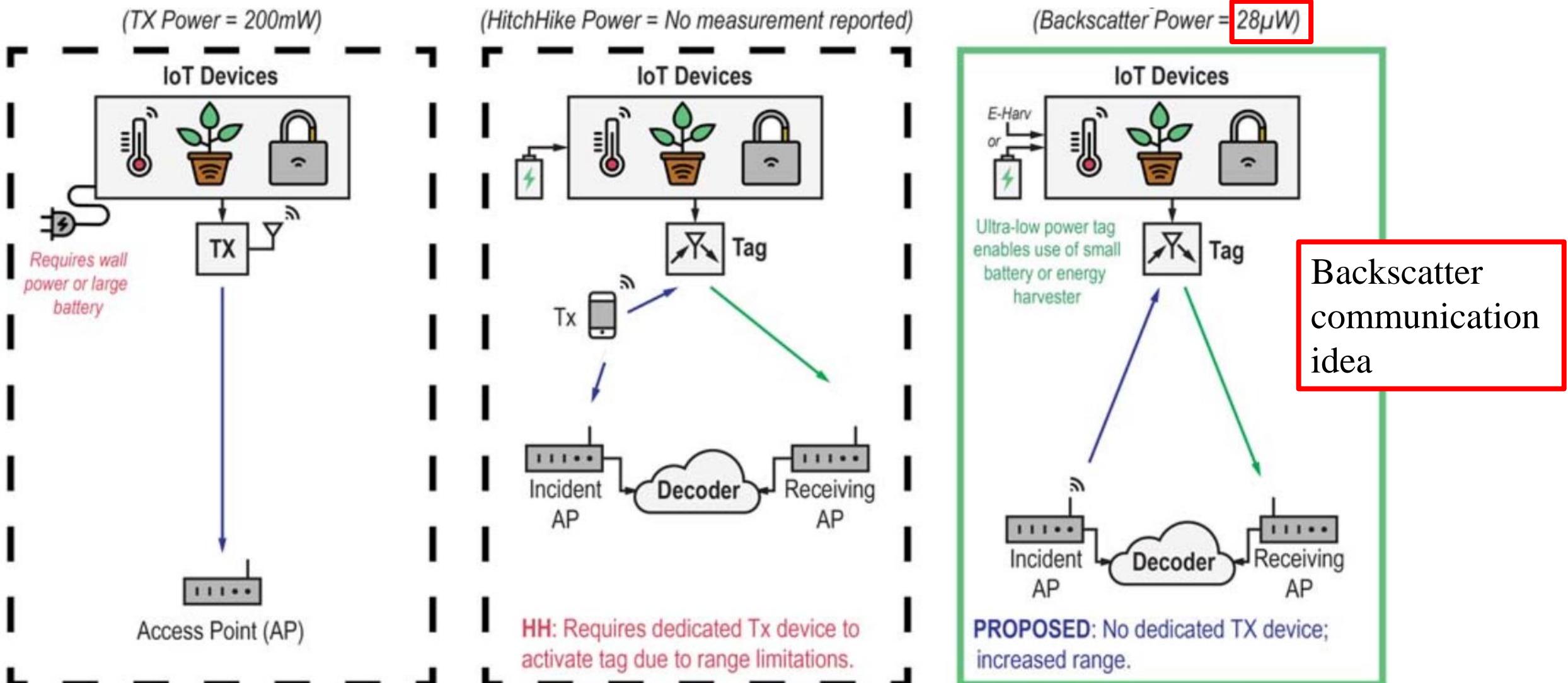
SERV [27]  
(2kGE)

SOTB-65nm: 22.09nW  
@0.5-V no bias & 32-KHz

\*Note: ~10× more power reduction when a reverse bias is applied.

# 5. Main research @ UEC (30/30) ULP-SoC (*future project*)

Develop ULP RF transceiver & communication protocol





# OUTLINE

1. Introduction
2. Achievement highlight
3. What is RISC-V?
4. Working with RISC-V & open community
5. Main research @ UEC
6. RISC-V courses @ UEC

# 6. RISC-V courses @ UEC (1/6) Five courses

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

No.	Name	Core knowledge
1	RISC-V Computer System Integration	<ul style="list-style-type: none"><li>• 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</li></ul>
2	Hardware Design with Chisel/Scala	<ul style="list-style-type: none"><li>• Learn how to design a hardware circuit using Scala/Chisel</li><li>• Learn how to create a new RISC-V computer system using open Scala/Chisel libraries</li></ul>
3	Linux on RISC-V	<ul style="list-style-type: none"><li>• Learn how to boot Linux on a RISC-V computer system → control your custom hardware after Linux boot</li></ul>
4	VLSI Design with RISC-V System-on-Chip (SoC)	<ul style="list-style-type: none"><li>• Make a RISC-V chip: flat layout for a simple system, hierarchy layout for a complex system</li></ul>
5	Cyber-security with RISC-V Computer System	<ul style="list-style-type: none"><li>• Learn how to modify a RISC-V computer system for security purposes: secure boot, cryptographic acceleration, prevent side-channel attacks, etc.</li></ul>

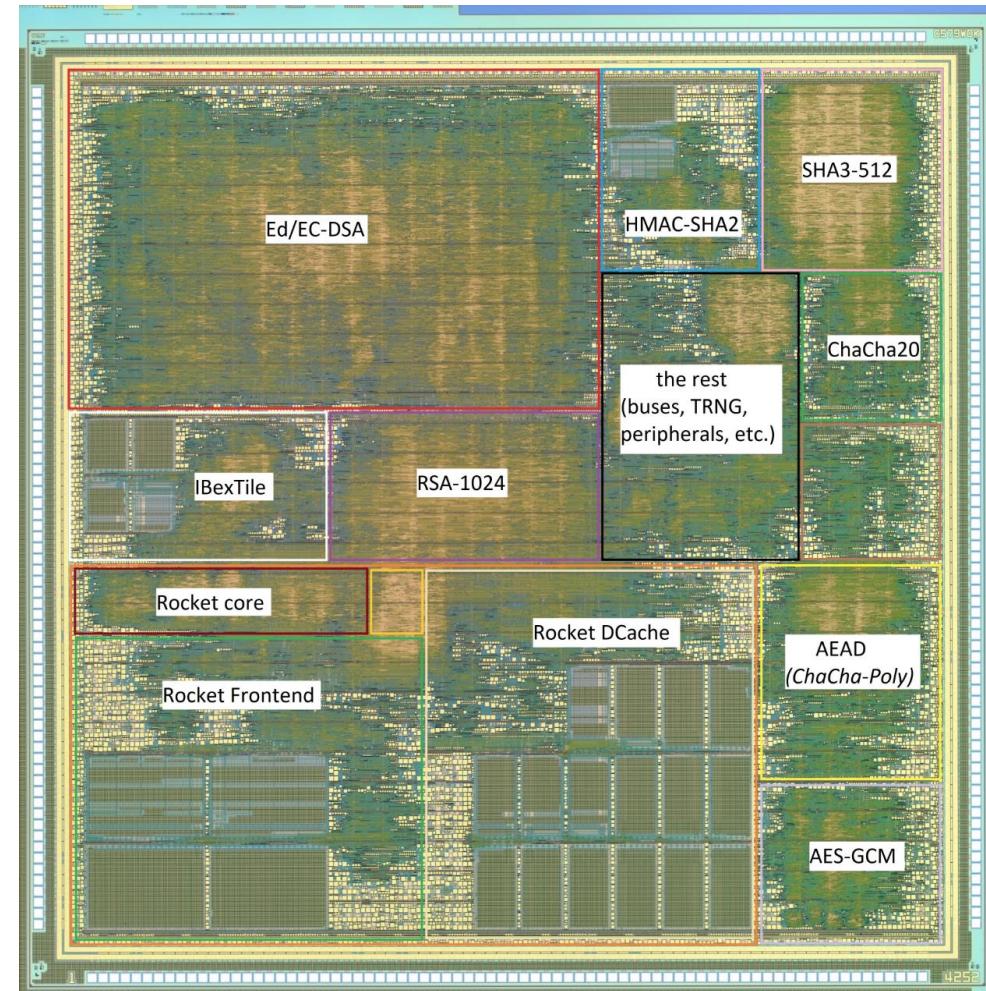
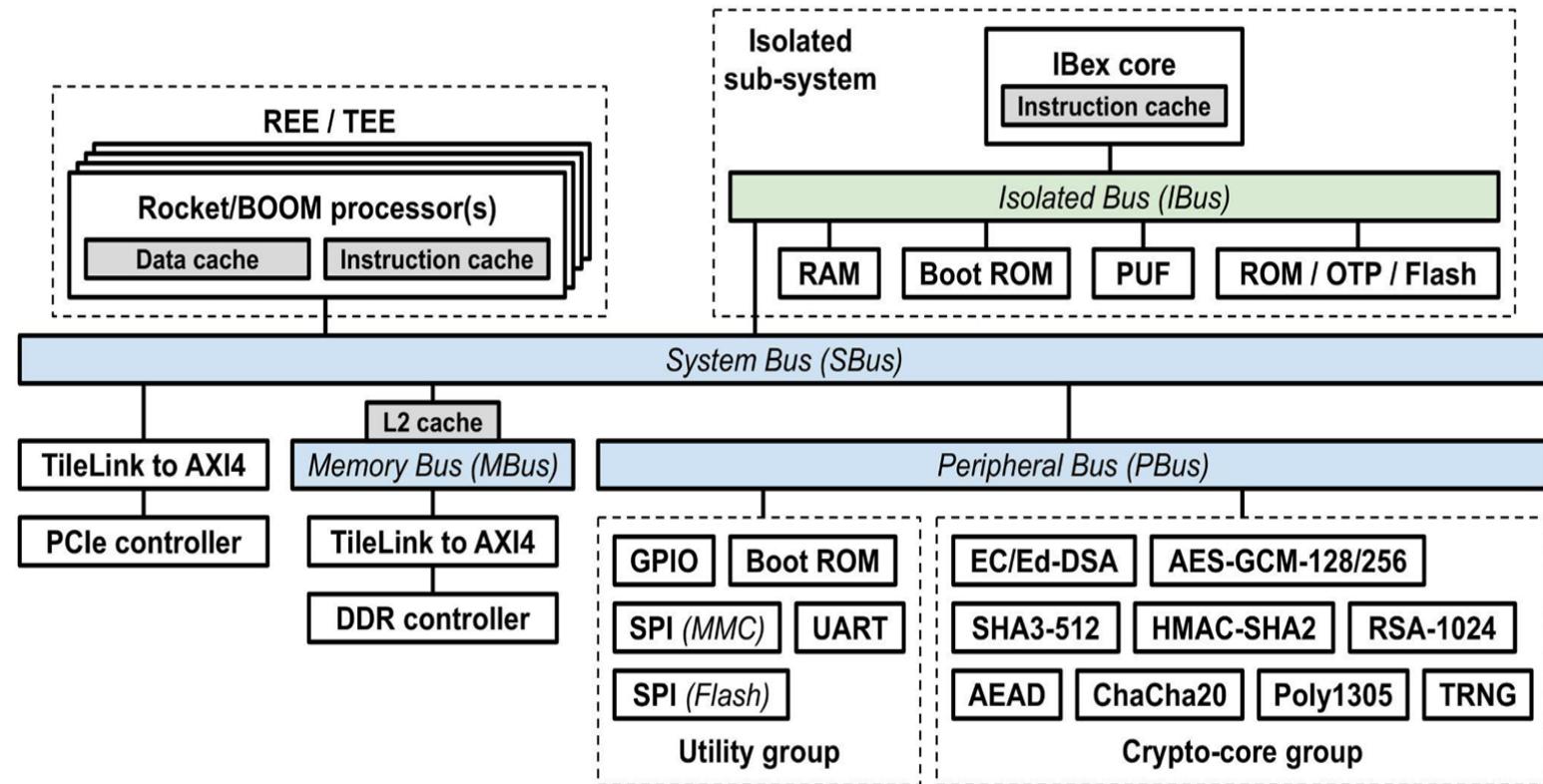
# 6. RISC-V courses @ UEC (2/6) Classes example

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*)

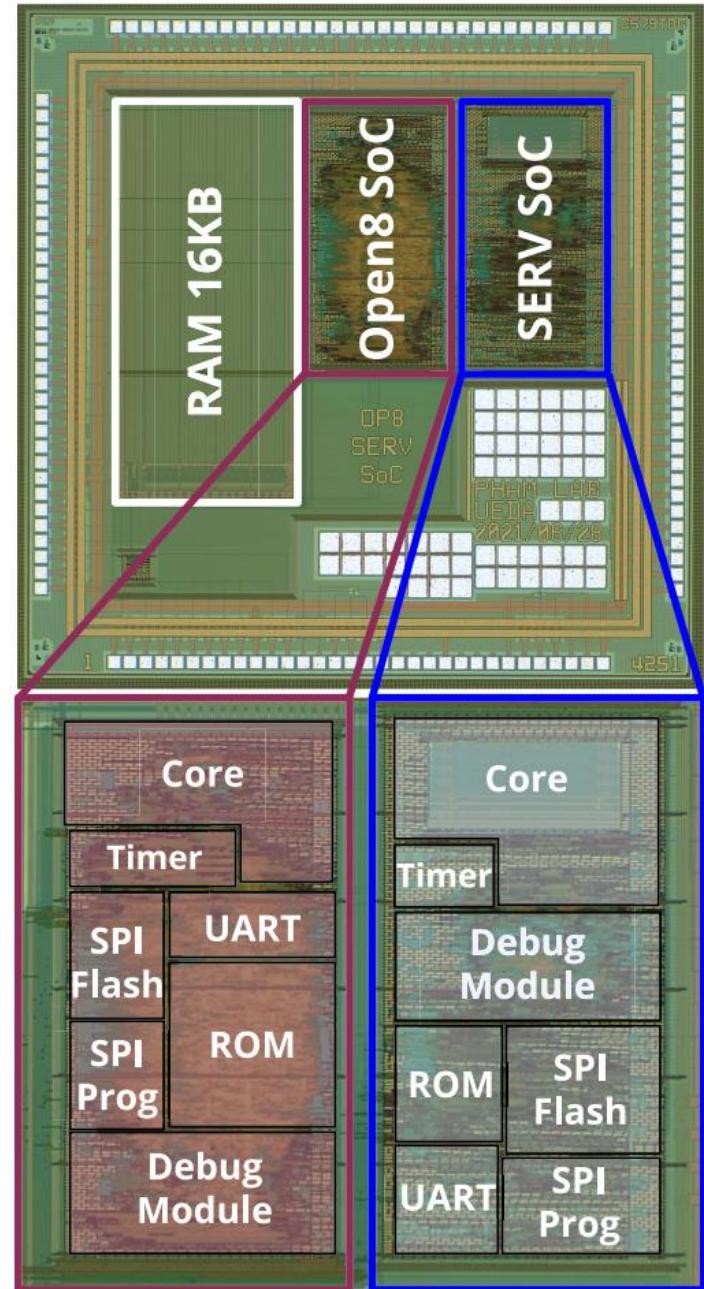
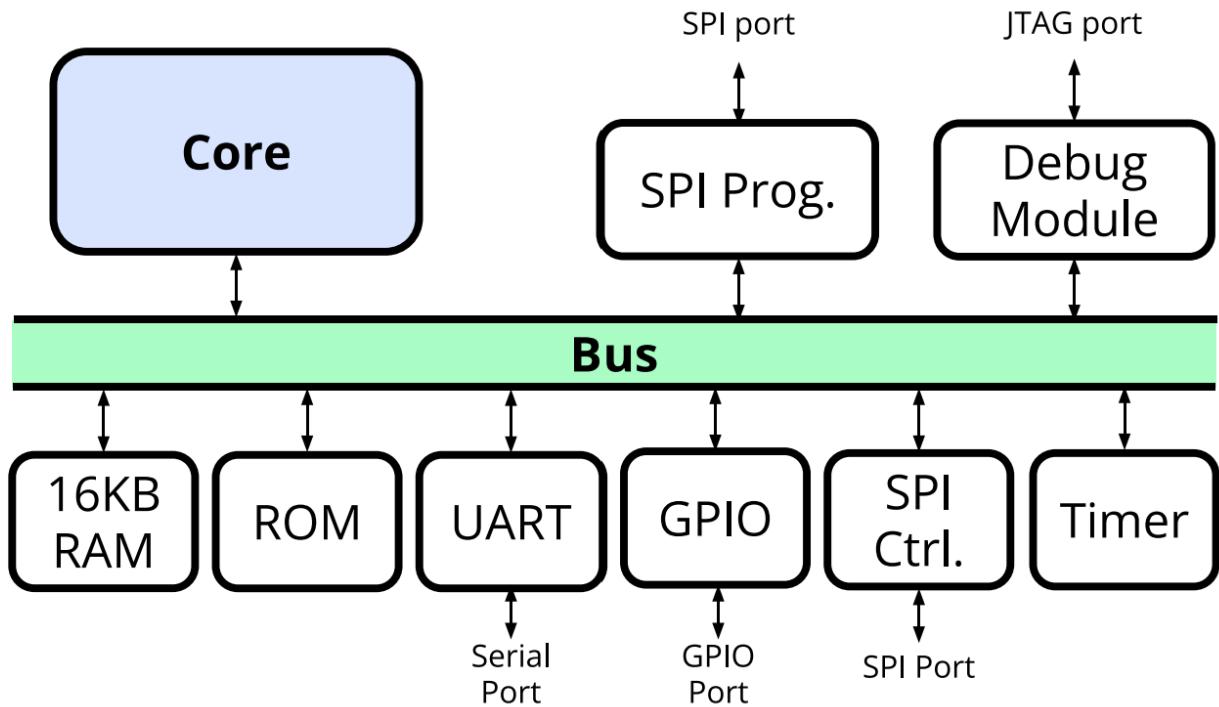
# 6. RISC-V courses @ UEC (3/6) Example design

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



# 6. RISC-V courses @ UEC (4/6) Example design

- ULP-SoC with small processor(s) and simple peripherals to achieve sub- $\mu$ W power consumption.



# 6. RISC-V courses @ UEC (5/6) Oversea courses

Sep. 2022



## *Place*

Academy of Cryptography  
Techniques (ACT), Hanoi, Vietnam

## *Content*

Course 1: *RISC-V Computer System Integration*

## 6. RISC-V courses @ UEC (6/6) Oversea courses

Potential future classes

Date	Place	Content
Aug. 2023	University of Electro-Communications (UEC) <i>Tokyo, Japan</i>	Course 1: <i>RISC-V Computer System Integration</i> Course 5: <i>Cyber-security with RISC-V Computer System</i>
Summer/Fall 2023	Le Quy Don Technical University (LQDTU) <i>Hanoi, Vietnam</i>	N/A
Summer/Fall 2023	Ho Chi Minh University of Technology (HCMUT) <i>HCM city, Vietnam</i>	N/A



**THANK YOU**