



国立大学法人  
電気通信大学



# TEE Hardware for RISC-V Implementation

**Authors:** Ckristian Duran, Trong-Thuc Hoang, Akira Tsukamoto,  
Kuniyasu Suzuki, and Cong-Kha Pham

# Outline

1. Introduction
2. Trusted Execution Environment
3. TEE-Hardware System
4. Crypto-cores Accelerators
5. Other Hardware Modules
6. Chip Results & Conclusion

# Outline

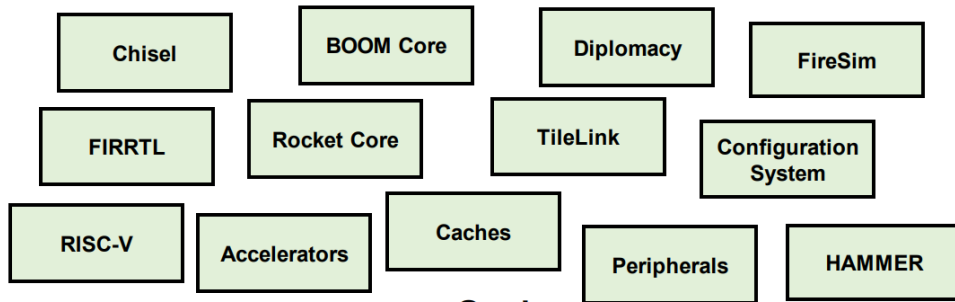
1. Introduction
2. Trusted Execution Environment
3. TEE-Hardware System
4. Crypto-cores Accelerators
5. Other Hardware Modules
6. Chip Results & Conclusion

# 1. Introduction (1/3)



Open-sources framework for agile development of Chisel-based System-on-Chip

Berkeley Architecture Research has developed and open-sourced:



**Goal:**

Make it easy for small teams to  
**design, integrate, simulate, and tape-out** a custom SoC



Berkeley Architecture Research

Fully customize hardware



VLSI

FPGA


Simulation




## Perks:








- Most common RISC-V cores: Rocket-chip, BOOM, Arianne (*and updated frequently with the mainstream of those cores*)
- FPGA accelerators included (*Hwacha, Gemmini, NVDLA*)
- Simulation supported (*RTL: Verilator, FPGA: FireSim, VLSI: Hammer*)

# 1. Introduction (2/3)






Based on Chipyard, a TEE-Hardware system is developed: <https://github.com/ckdur/tee-hardware>

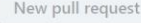


 **ckdur / tee-hardware**











 Watch **1**  Star **1**  Fork **0**

 Code  Issues **0**  Pull requests **0**  Actions  Projects **0**  Security **0**  Insights

TEE hardware - based on the chipyard repository - hardware to accelerate TEE

 **141** commits  **3** branches  **0** packages  **0** releases  **2** contributors

Branch: **master**   

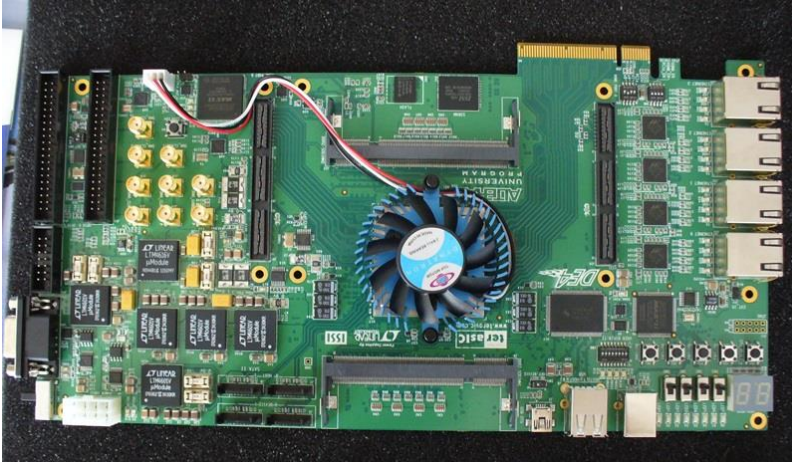
 <b>ckdur</b>	Update of the XIP bootloader form vlsi	Latest commit f4d6f85 3 days ago
 <b>fpga</b>	Fixes on variables. Changed the order of configurations. Added the UA...	3 days ago
 <b>hardware</b>	Fixes on variables. Changed the order of configurations. Added the UA...	3 days ago
 <b>patches</b>	Move the bootrom/ dir to inside the hardware/chipyard/ dir to avoid t...	2 months ago
 <b>project</b>	Initial configuration for sbt. Some empty files to do adaptation	7 months ago
 <b>simulator/verilator</b>	Fixes on variables. Changed the order of configurations. Added the UA...	3 days ago
 <b>software</b>	Update of the XIP bootloader form vlsi	3 days ago
 <b>.gitignore</b>	First multi-top, chip and harness generator	13 days ago
 <b>.gitmodules</b>	Rename hardware/keystoneAcc folder to hardware/teehw	last month
 <b>README.md</b>	Update README.md & remove other unused README.md files	2 months ago

# 1. Introduction (3/3)

**TEE-HW has demos on:**



Xilinx: VC707



Altera: DE4



Altera: TR4

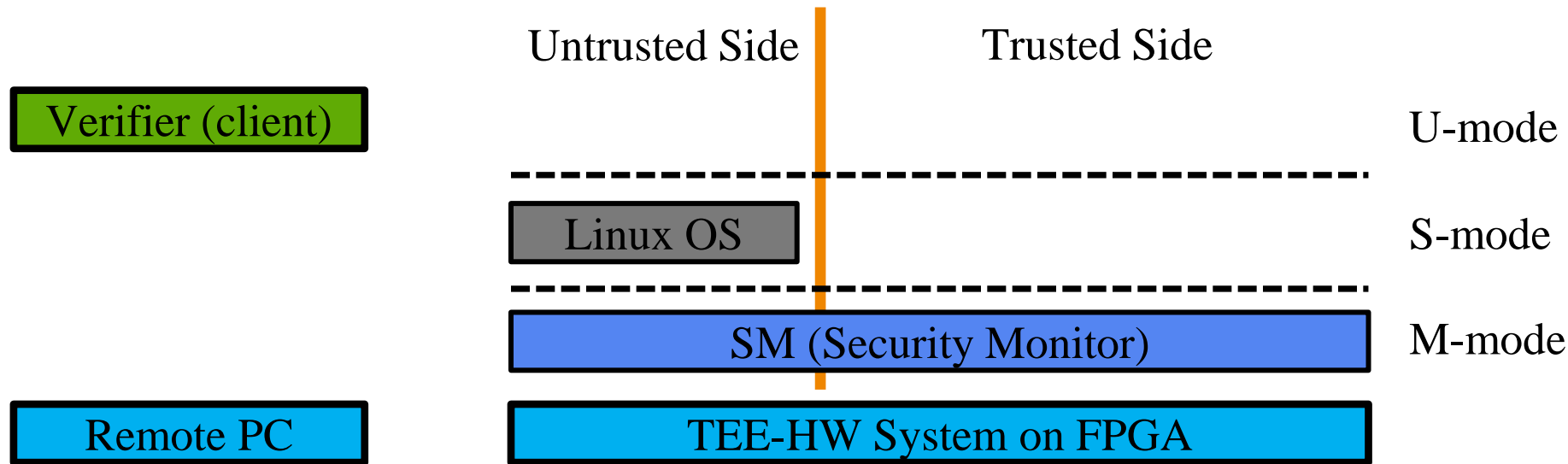
# Outline

1. Introduction
- 2. Trusted Execution Environment**
3. TEE-Hardware System
4. Crypto-cores Accelerators
5. Other Hardware Modules
6. Chip Results & Conclusion

## 2. Trusted Execution Environment (1/8)

### TEE in-action (*using Keystone: A TEE Framework*)

Remote PC connects to FPGA via Serial (*UART*) terminal or a TCP connection



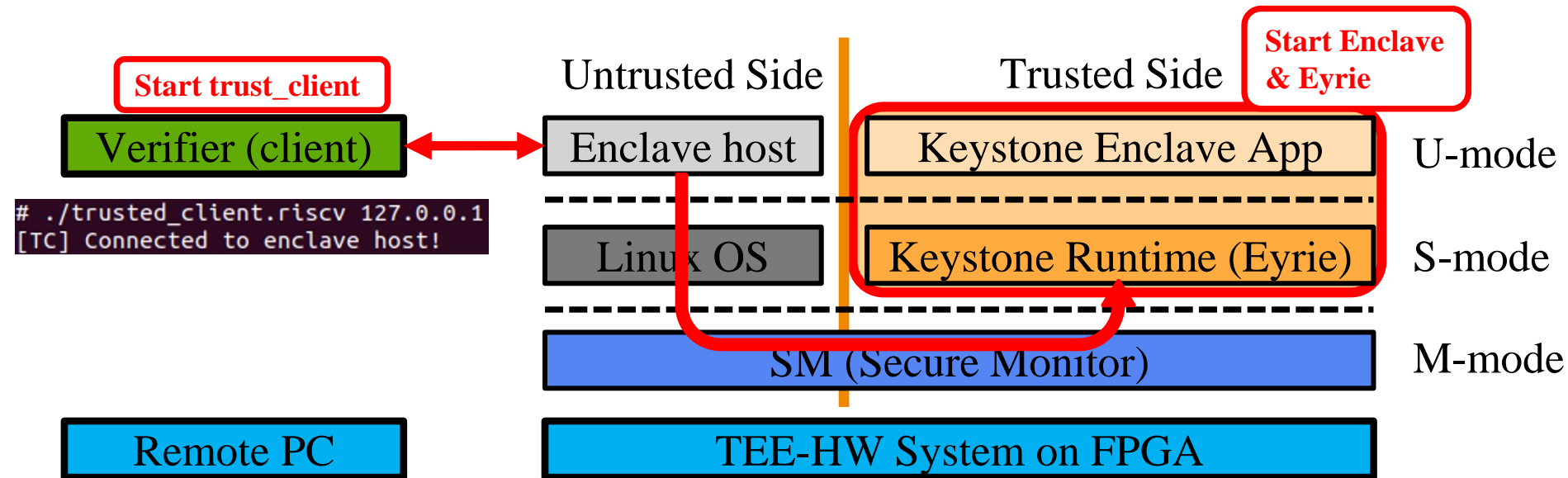
TEE (*Keystone in this case*) creates the Trusted-Side based on the chain-of-trust across multiple operating layers.

It allows client to create and operate an Enclave App in the Trusted Side.



## 2. Trusted Execution Environment (2/8)

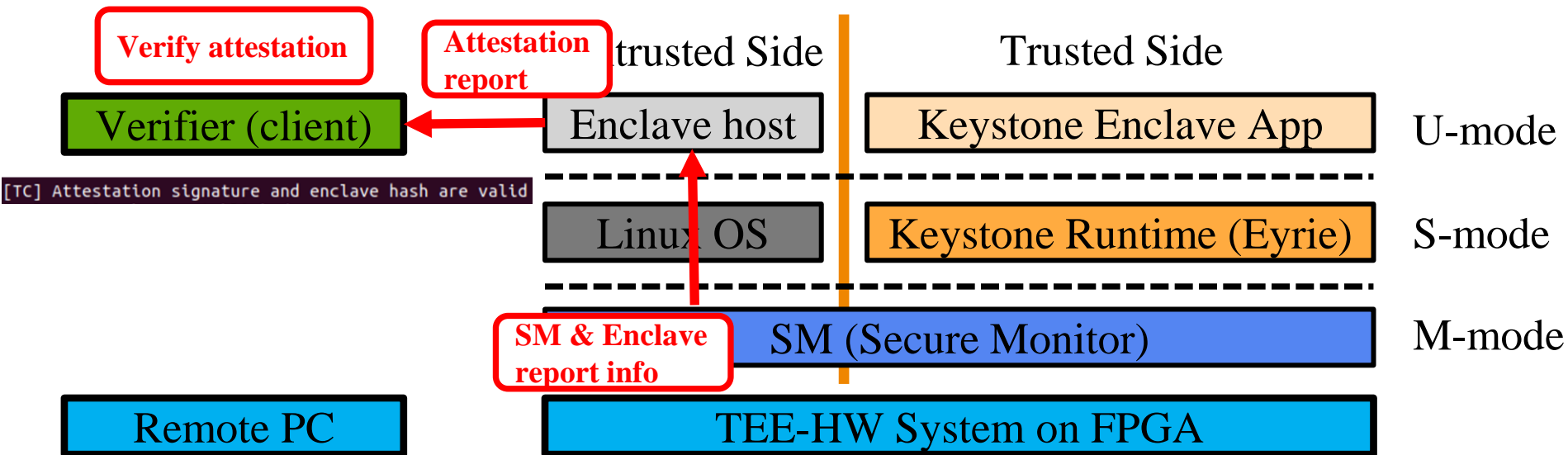
### TEE in-action (*using Keystone: A TEE Framework*)



1. Connection with the Enclave host

## 2. Trusted Execution Environment (3/8)

### TEE in-action (*using Keystone: A TEE Framework*)



1. Connection with the Enclave host
2. Verify attestation report

## 2. Trusted Execution Environment (4/8)

### TEE in-action (*using Keystone: A TEE Framework*)

An example of attestation report:

```
[TC] Connected to enclave host!  
    === Security Monitor ===  
Hash: e56168887f2d0748cf7cd57c73b46c6a60fd8bcf80a852e4c134326efa6259f5c8c4a38543  
514612d685baaf6de15edf330d4b74e7bf0f5405257029e79fcd20  
Pubkey: cd98f4a28a8523ba8ecd31175aa0e2330b2f46e7034545254660126a9f3b8cb9  
Signature: d5c4b1647d444d5b76bd5ecf24ad5e774cb761b4a7a864c754683b1a8db8340adcf06  
ebf7da099d35ef7aef26834e5ffbdd86ca7815a6a6602cc4ee721a14f01  
  
    === Enclave Application ===  
Hash: 84b2193e0f5ec391672d9f68415fbf8a928e1a25b89dfb88257d7a3becf310229d8bed1534  
5884f90f69792f6da237d8b6a9d55abe254f70b4181961989cbe7b  
Signature: c443ec369888c4065dbaaaaed9210f1d967bace5395cb3e679ca29a1aa8a8afa34ff2  
80e597ac229b1a87d29acaec5d5db2b93c2ccd415b5909042a19a181b0e  
Enclave Data: 8d571e0103e72ee6986407e67d5789dd8bc3318d663e519890f078f66b05ef57  
    -- Device pubkey --  
0faad4ff01178583baa588966f7c1ff32564dd17d7dc2b46cb50a84a69270b4c
```

→ SM hash

→ SM signature

→ Enclave App hash

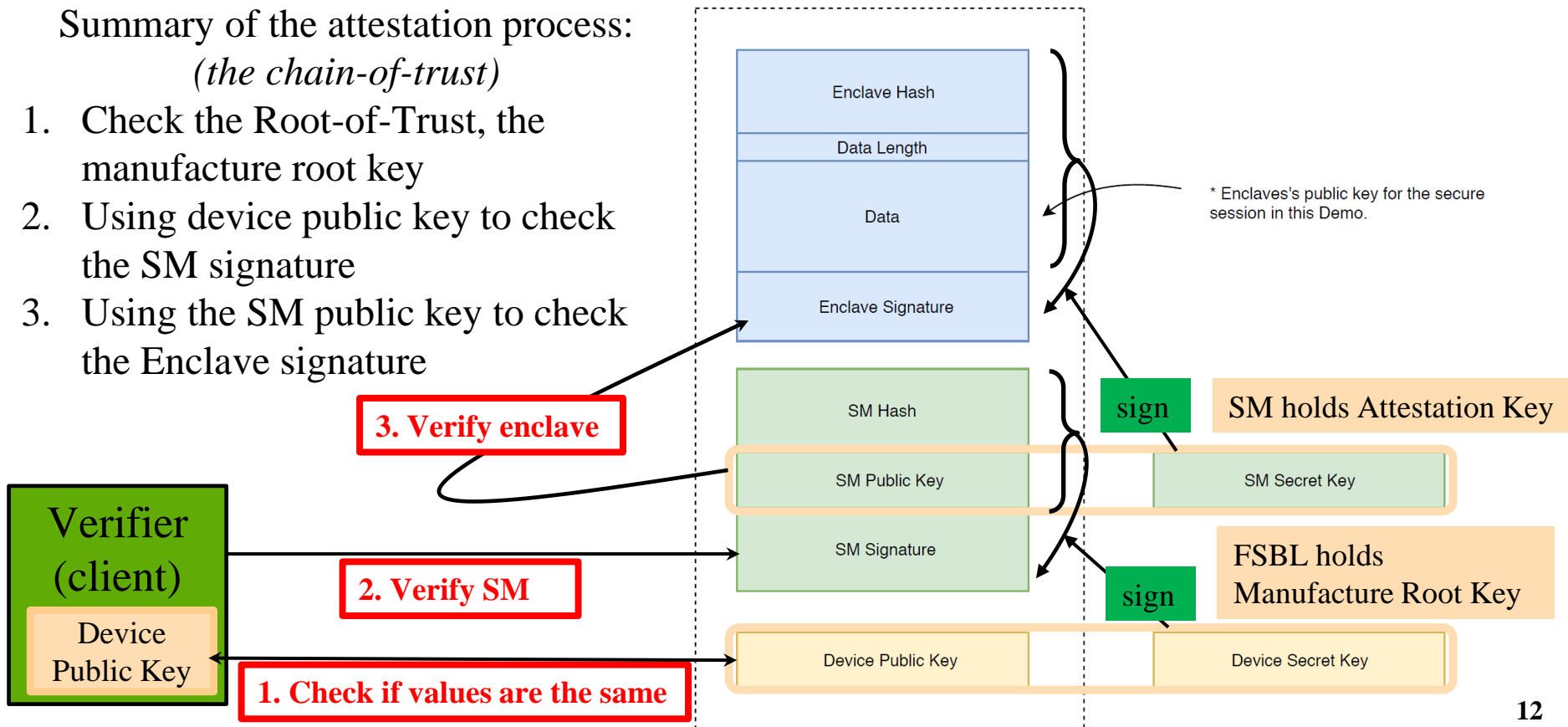
→ Manufacture Root key

## 2. Trusted Execution Environment (5/8)

### TEE in-action (*using Keystone: A TEE Framework*)

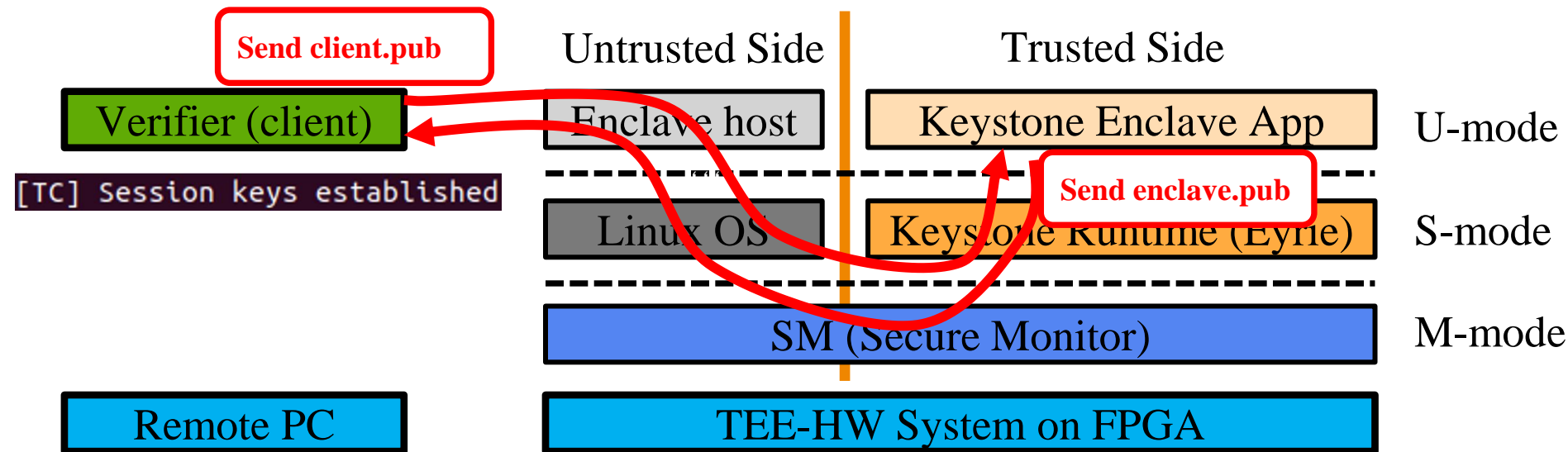
Summary of the attestation process:  
(*the chain-of-trust*)

1. Check the Root-of-Trust, the manufacture root key
2. Using device public key to check the SM signature
3. Using the SM public key to check the Enclave signature



## 2. Trusted Execution Environment (6/8)

### TEE in-action (*using Keystone: A TEE Framework*)

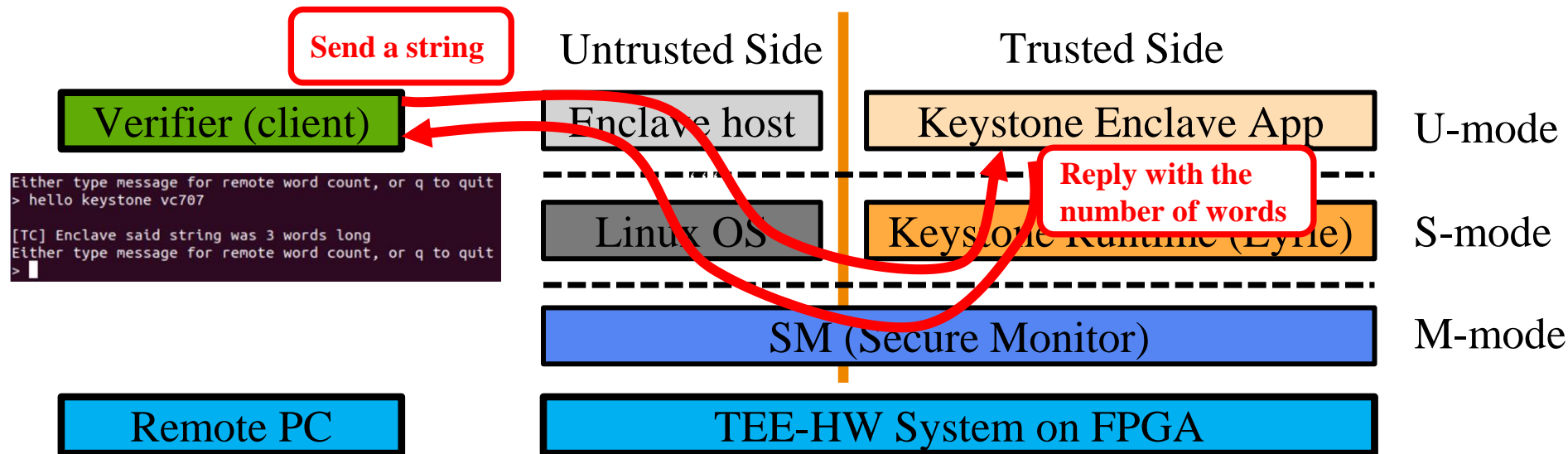


1. Connection with the Enclave host
2. Verify attestation report
3. Exchange communication keys

## 2. Trusted Execution Environment (7/8)

### TEE in-action (*using Keystone: A TEE Framework*)

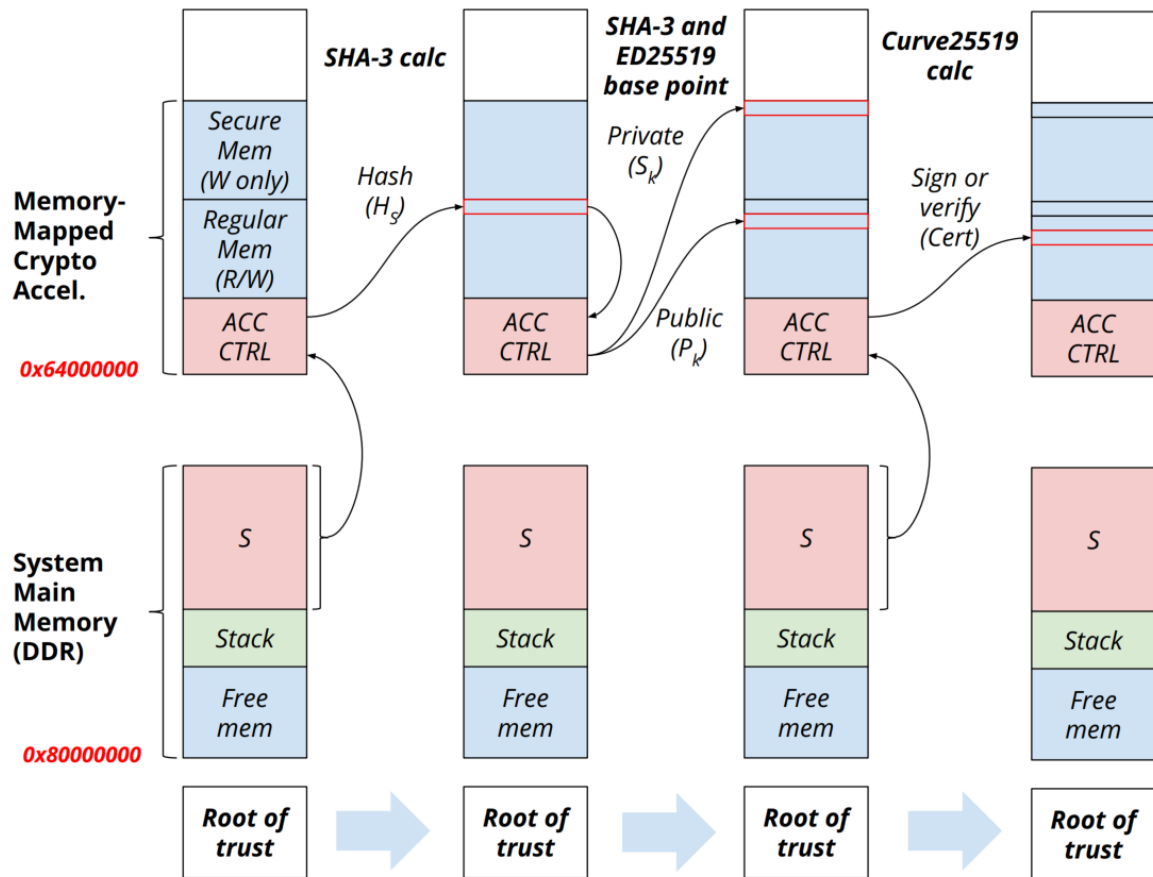
Keystone demo: (1) client sends strings, then (2) request calculation from the Enclave, finally (3) the Enclave replies with the number of words



1. Connection with the Enclave host
2. Verify attestation report
3. Exchange communication keys
4. Client's app runs on the established TEE

## 2. Trusted Execution Environment (8/8)

### TEE Secure Boot Sequence (*with HW Accelerators*)



- The  $H_S$  value is automatically transferred between acts, thus it is not exposed to the software.
- The data in W-only memory are also not exposed to the software.

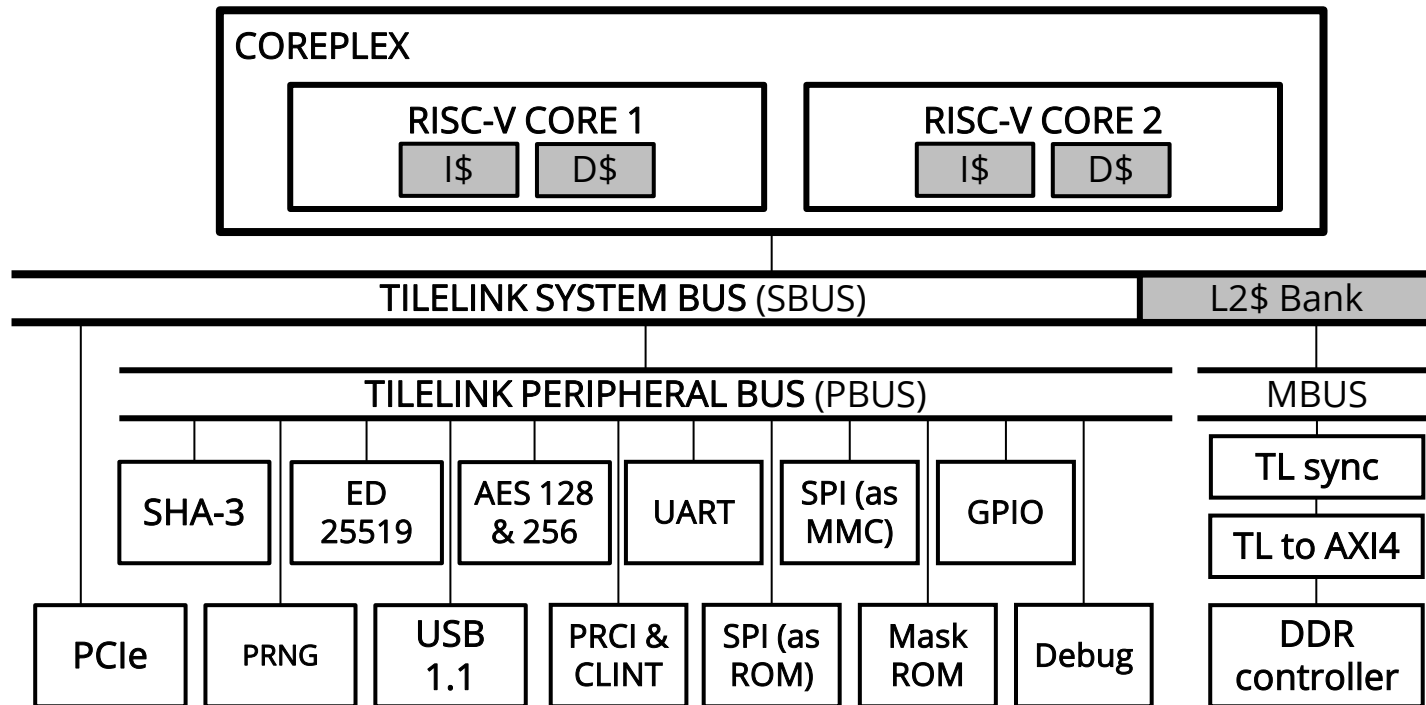
# Outline

1. Introduction
2. Trusted Execution Environment
- 3. TEE-Hardware System**
4. Crypto-cores Accelerators
5. Other Hardware Modules
6. Chip Results & Conclusion



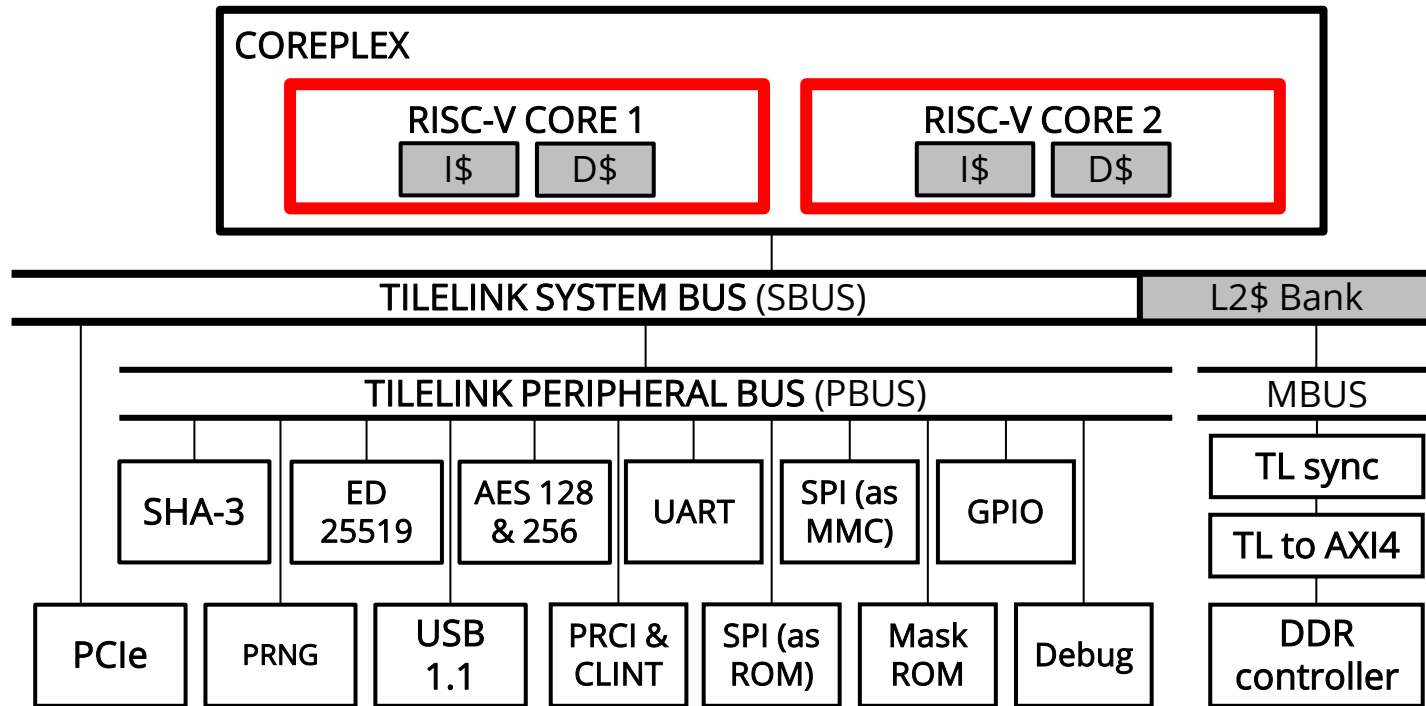
### 3. TEE-Hardware System (1/5)

#### System Architecture:



- Not fixed at dual-core, can increase/decrease the number of cores as you wanted (*as long as that fits the FPGA board*)
- Some hardware modules can be easily included/excluded to/from the system

### 3. TEE-Hardware System (2/5)



- Available cores in the system are Rocket-chip and BOOM
- Because BOOMv3 isn't stable yet, so both BOOMv2 and BOOMv3 are available on the GitHub with different branches.

### 3. TEE-Hardware System (3/5)

Variable	Available option	Description
BOARD	<ul style="list-style-type: none"> <li>- VC707</li> <li>- DE4</li> <li>- TR4</li> </ul>	Select the FPGA board
ISACONF	<ul style="list-style-type: none"> <li>- RV64GC</li> <li>- RV64IMAC</li> <li>- RV32GC</li> <li>- RV32IMAC</li> </ul>	Select the ISA
MBUS	<ul style="list-style-type: none"> <li>- MBus64</li> <li>- MBus32</li> </ul>	Select the bit-width for the memory bus
BOOTSRC	<ul style="list-style-type: none"> <li>- BOOTROM</li> <li>- QSPI</li> </ul>	Select the boot source
PCIE	<ul style="list-style-type: none"> <li>- WPCIE</li> <li>- WoPCIE</li> </ul>	<ul style="list-style-type: none"> <li>- Include PCIe module in the system</li> <li>- Remove PCIe module from the system</li> </ul>
DDRCLK	<ul style="list-style-type: none"> <li>- WSepaDDRCIk</li> <li>- WoSepaDDRCIk</li> </ul>	<ul style="list-style-type: none"> <li>- Separate DDR-clock with System-clock</li> <li>- Not separate DDR-clock with System-clock</li> </ul>
HYBRID	<ul style="list-style-type: none"> <li>- Rocket</li> <li>- Boom</li> <li>- RocketBoom</li> <li>- BoomRocket</li> </ul>	<ul style="list-style-type: none"> <li>- Two Rocket cores</li> <li>- Two Boom cores</li> <li>- Rocket core 1<sup>st</sup>, Boom core 2<sup>nd</sup></li> <li>- Boom core 1<sup>st</sup>, Rocket core 2<sup>nd</sup></li> </ul>

In the Makefile system, these variables are available.

Example usage:

```
BOARD=VC707
ISACONF=RV64GC
MBUS=MBus64
BOOTSRC=BOOTROM
PCIE=WoPCIE
DDRCLK=WoSepaDDRCIk
HYBRID=Rocket
```

### 3. TEE-Hardware System (4/5)

## TEE-HW with various core configurations

Boom

```
# cat /proc/cpuinfo
hart      : 0
isa       : rv64imafdc
mmu       : sv39
uarch     : ucb-bar,boom0
```

```
hart      : 1
isa       : rv64imafdc
mmu       : sv39
uarch     : ucb-bar,boom0
```

Rocket

```
# cat /proc/cpuinfo
hart      : 0
isa       : rv64imafdc
mmu       : sv39
uarch     : sifive,rocket0
```

```
hart      : 1
isa       : rv64imafdc
mmu       : sv39
uarch     : sifive,rocket0
```

BoomRocket

```
# cat /proc/cpuinfo
hart      : 0
isa       : rv64imafdc
mmu       : sv39
uarch     : ucb-bar,boom0
```

```
hart      : 1
isa       : rv64imafdc
mmu       : sv39
uarch     : sifive,rocket0
```

RocketBoom

```
# cat /proc/cpuinfo
hart      : 0
isa       : rv64imafdc
mmu       : sv39
uarch     : sifive,rocket0
```

```
hart      : 1
isa       : rv64imafdc
mmu       : sv39
uarch     : ucb-bar,boom0
```

RV64GC

```
# cat /proc/cpuinfo
hart      : 0
isa       : rv64imafdc
mmu       : sv39
uarch     : sifive,rocket0
```

```
hart      : 1
isa       : rv64imafdc
mmu       : sv39
uarch     : sifive,rocket0
```

RV64IMAC

```
# cat /proc/cpuinfo
hart      : 0
isa       : rv64imac
mmu       : sv39
uarch     : sifive,rocket0
```

```
hart      : 1
isa       : rv64imac
mmu       : sv39
uarch     : sifive,rocket0
```

RV32GC

```
# cat /proc/cpuinfo
hart      : 0
isa       : rv32imafdc
mmu       : sv32
uarch     : sifive,rocket0
```

```
hart      : 1
isa       : rv32imafdc
mmu       : sv32
uarch     : sifive,rocket0
```

RV32IMAC

```
# cat /proc/cpuinfo
hart      : 0
isa       : rv32imac
mmu       : sv32
uarch     : sifive,rocket0
```

```
hart      : 1
isa       : rv32imac
mmu       : sv32
uarch     : sifive,rocket0
```

### 3. TEE-Hardware System (5/5)

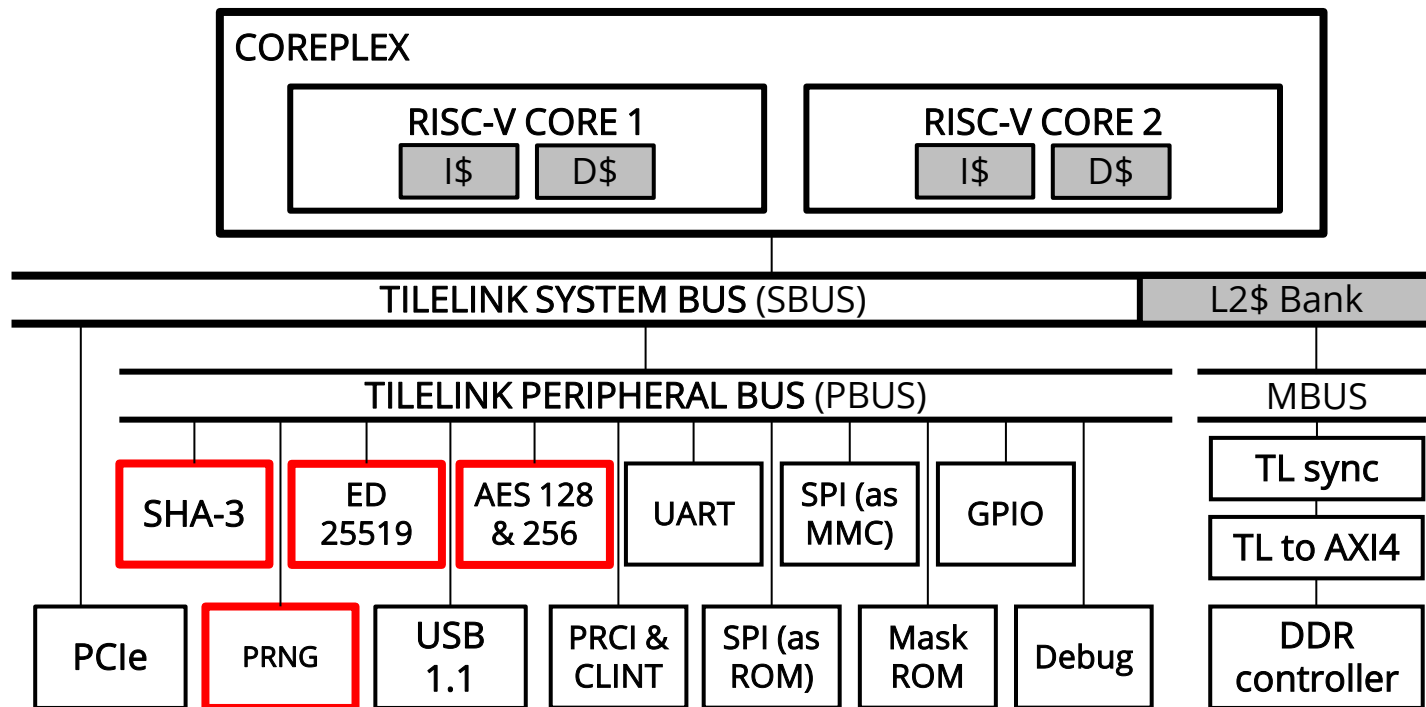
Summary table of FPGA logic utilization (*on VC707*) with various core configurations:

ISACONF	HYBRID		FPGA logic utilization (LUT) ( <i>on VC707</i> )	
	Core0	Core1		
RV64GC	Boom	Boom	160,873	52.99%
	Rocket	Rocket	96,571	31.81%
	Boom	Rocket	128,708	42.39%
	Rocket	Boom	128,719	42.40%
RV64GC	Rocket	Rocket	96,571	31.81%
RV64IMAC			72,007	23.72%
RV32GC			89,356	29.43%
RV32IMAC			65,899	21.71%

# Outline

1. Introduction
2. Trusted Execution Environment
3. TEE-Hardware System
- 4. Crypto-cores Accelerators**
5. Other Hardware Modules
6. Chip Results & Conclusion

## 4. Crypto-core Accelerators (1/6)



- Crypto-cores:**
- SHA-3 512
  - Ed25519 (*genkey and certification*)
  - AES-128/256
  - PRNG (*Pseudo-random generator*)

## 4. Crypto-core Accelerators (2/6)

### Some feature notes

- Any crypto-core can be brought from Peripheral Bus (PBus) to ROCC if needed.
- SHA-3 512 follows the NIST standard specification on Keccak
- AES supports both 128 and 256 bits, and can be changed on-the-fly
- Ed25519 contains two separated accelerators named Ed25519-Mult and Ed25519-Sign for pair-key generation and certificate verification, respectively.
- PRNG is done by utilizing the LFSR (*Linear-Feedback Shift Register*); we developed the PRNG based on ARM TrustZone RNG programming model



## 4. Crypto-core Accelerators (3/6)

### Crypto-cores on Stratix-IV FPGA

	SHA-3	AES-128/256	Ed25519	
			Mult	Sign
ALUT	8,108	3,195	2,737	3,969
Registers	2,790	2,854	4,778	4,617
Fmax (MHz)	100	100	100	100
Memory	0	0	8KB	0
DSP block	0	0	48	0

## 4. Crypto-core Accelerators (4/6)

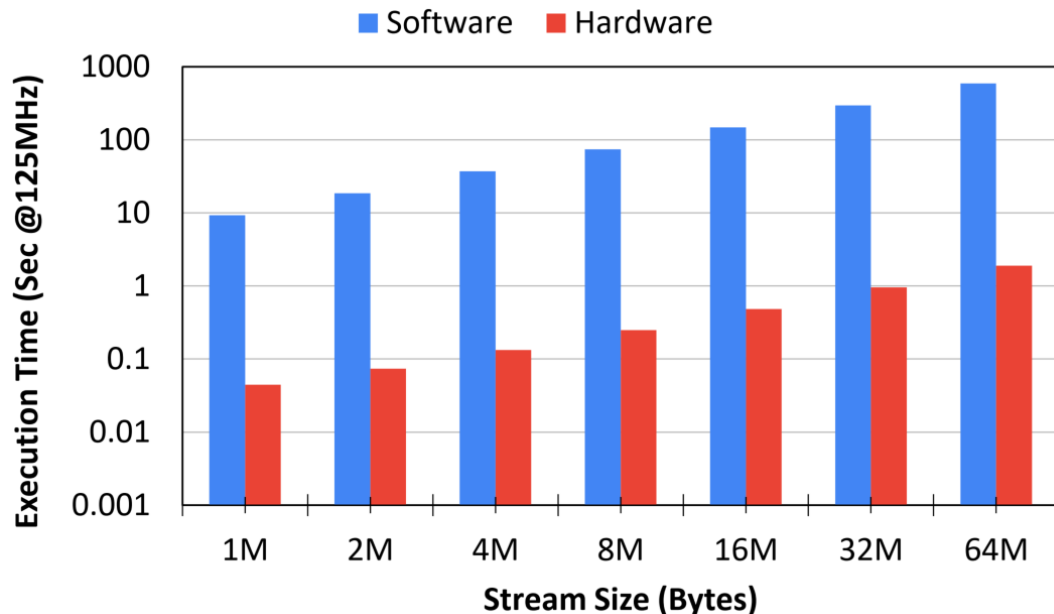
### Crypto-cores in ASIC (*ROHM-180nm*)

	SHA-3	AES-128/256	Ed25519	
			Mult	Sign
Size	1,150μm × 1,150μm	808.96μm × 806.4μm	1,694.72μm × 1,693.44μm	1,346.56μm × 1,345.68μm
Gate-count (NAND)	102,500	50,560	222,432	140,442
Fmax (MHz)	104	90	106	91
Power (mW)	42.745	37.566	53.061	80.894

## 4. Crypto-core Accelerators (5/6)

The result of using crypto-core hardware accelerators (*applying at boot stage*)

The test was done on Stratix-IV FPGA with Rocket-chip RV64GC core



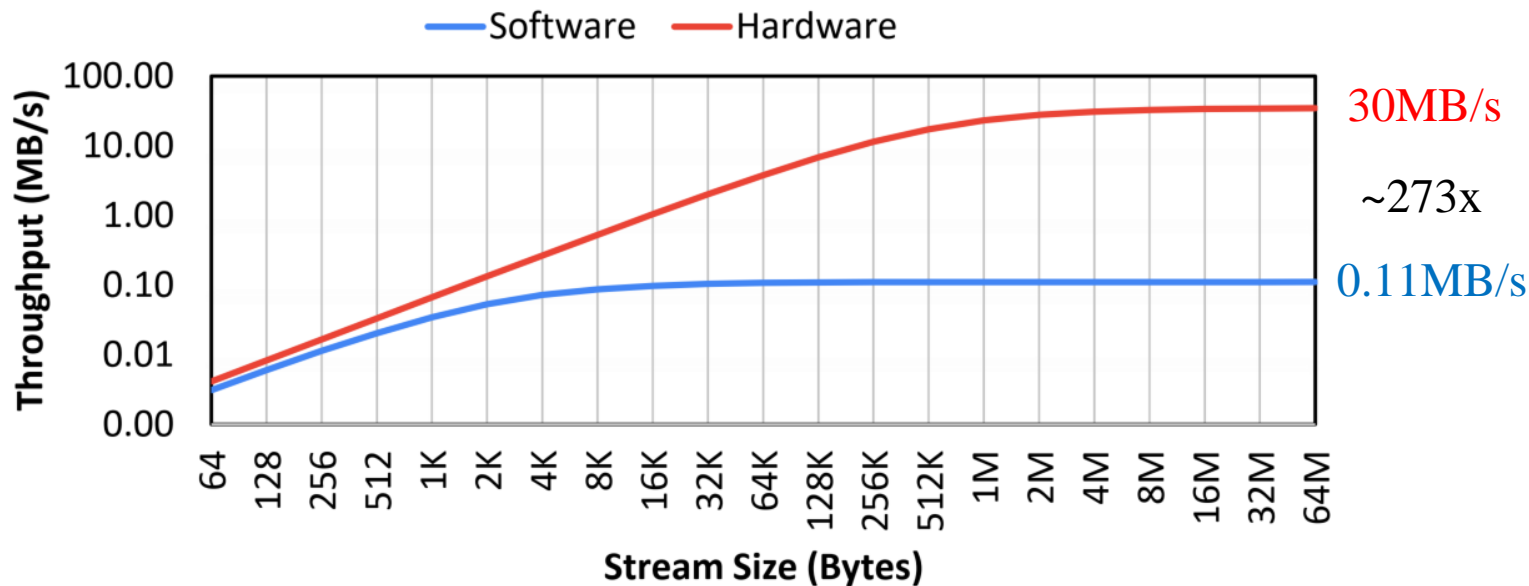
Software vs. hardware of SHA-3 execution times in the TEE framework.

*Hardware is faster about 2.5 decades*

## 4. Crypto-core Accelerators (6/6)

The result of using crypto-core hardware accelerators (*applying at boot stage*)

The test was done on Stratix-IV FPGA with Rocket-chip RV64GC core

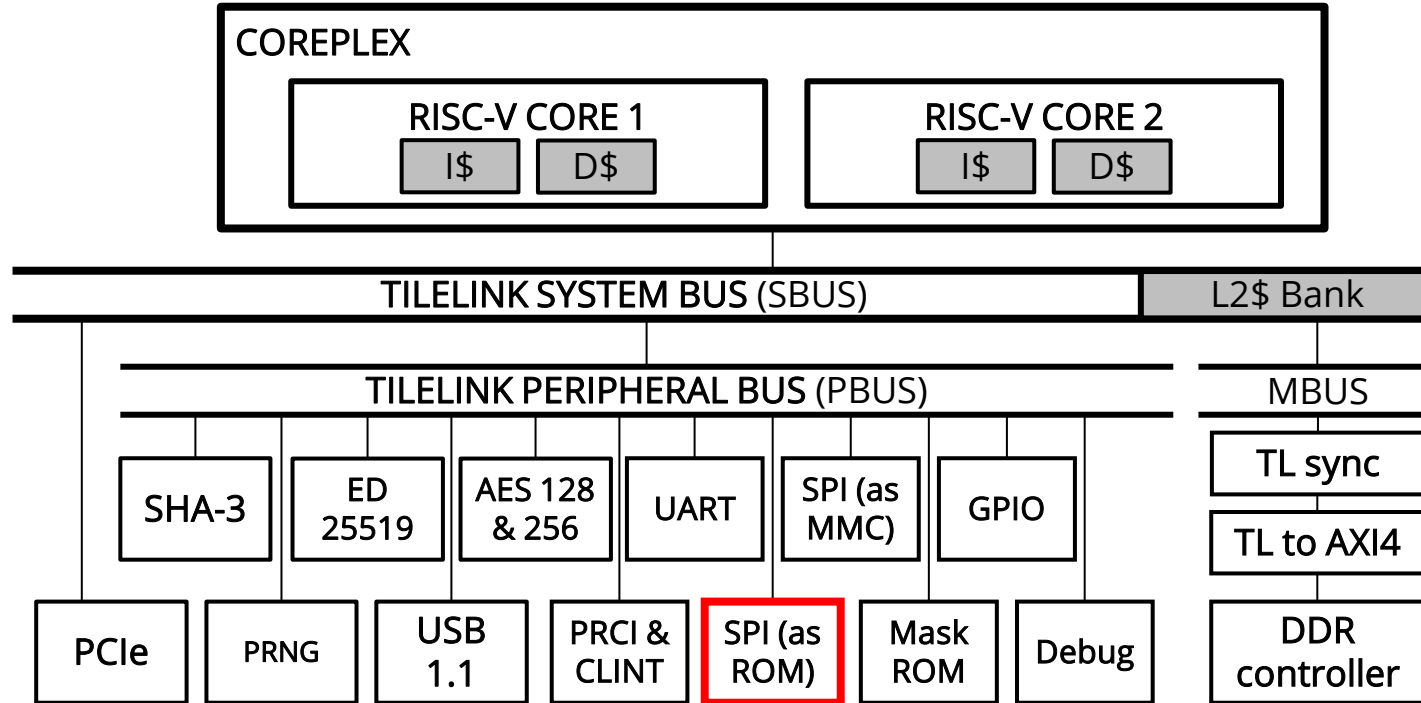


Software vs. hardware of SHA-3 operation throughput.

# Outline

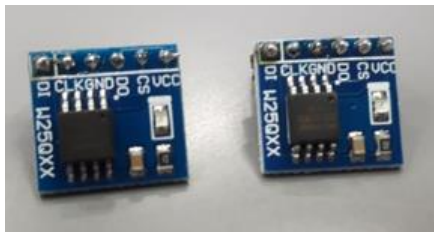
1. Introduction
2. Trusted Execution Environment
3. TEE-Hardware System
4. Crypto-cores Accelerators
- 5. Other Hardware Modules**
6. Chip Results & Conclusion

## 5. Other Hardware Modules (1/4)



**QSPI:** to use Flash outside

## 5. Other Hardware Modules (2/4)



Flash modules

*(cheap, bundle, and easy to plug-in with FPGA boards)*

```
class BOOTROM extends Config((site, here, up) => {  
  case PeripheryMaskROMKey => List(  
    MaskROMParams(address = BigInt(0x20000000), depth = 2048, name = "BootROM")  
  )  
  case PeripherySPIFlashKey => List() // disable SPIFlash  
})  
  
class QSPI extends Config((site, here, up) => {  
  case PeripheryMaskROMKey => List( //move BootROM back to 0x10000  
    MaskROMParams(address = 0x10000, depth = 16, name = "BootROM")) //smallest allowed depth is 16  
  case PeripherySPIFlashKey => List(  
    SPIFlashParams(fAddress = 0x20000000, rAddress = 0x64005000, defaultSampleDel = 3)  
  )  
})
```

- BOOTROM scenario:

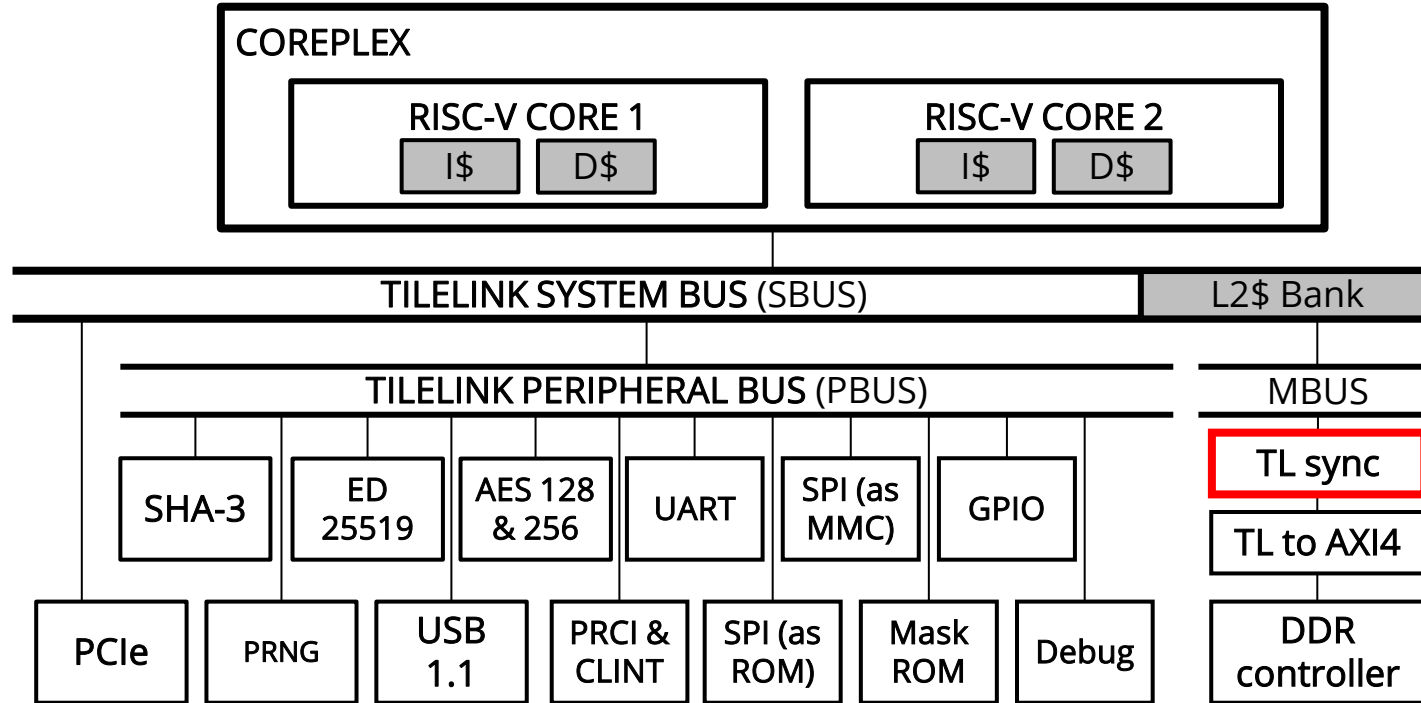
- Disable QSPI
- BootROM at 0x20000000, ZSBL in BootROM

- QSPI scenario:

- Enable QSPI at 0x20000000, ZSBL now in Flash
- BootROM moved back to 0x10000, in BootROM now just a simple instruction to jump directly to 0x20000000

Easy to on/off the using of QSPI

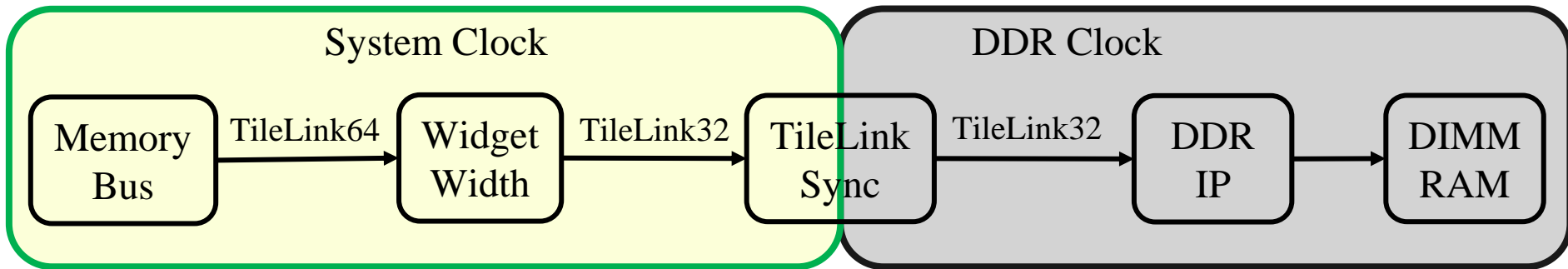
## 5. Other Hardware Modules (3/4)



**TileLink Sync:** synchronize between different clock domains



## 5. Other Hardware Modules (4/4)



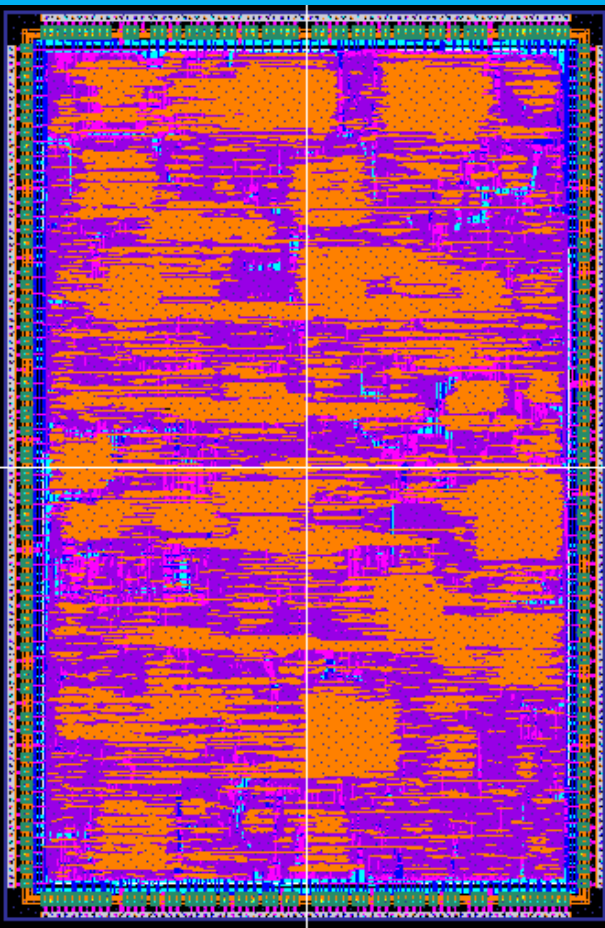
Separate the inner system clock with outer DDR clock:

- Sometime inner system cannot run at high-speed
  - System-clock < DDR-clock
  - Keep the DDR bandwidth still at high-speed
- Sometime (*depends on board*) DDR IP is fixed at lower clock rate (*for example, 100MHz*) than the CPU (*for example, 125MHz*)
  - System-clock > DDR-clock
  - Keep the CPU runs at higher clock rate

# Outline

1. Introduction
2. Trusted Execution Environment
3. TEE-Hardware System
4. Crypto-cores Accelerators
5. Other Hardware Modules
- 6. Chip Results & Conclusion**

## 6. Conclusion (1/4)



Layout

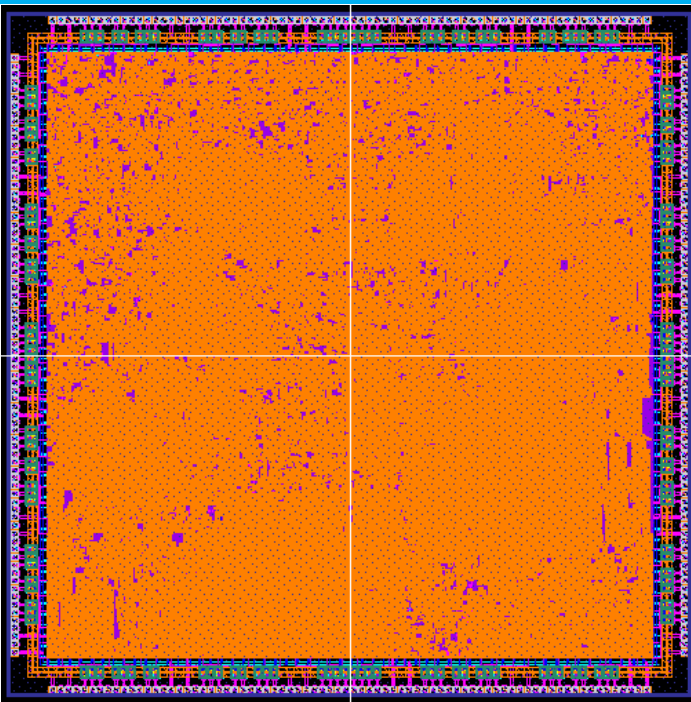


Barechip

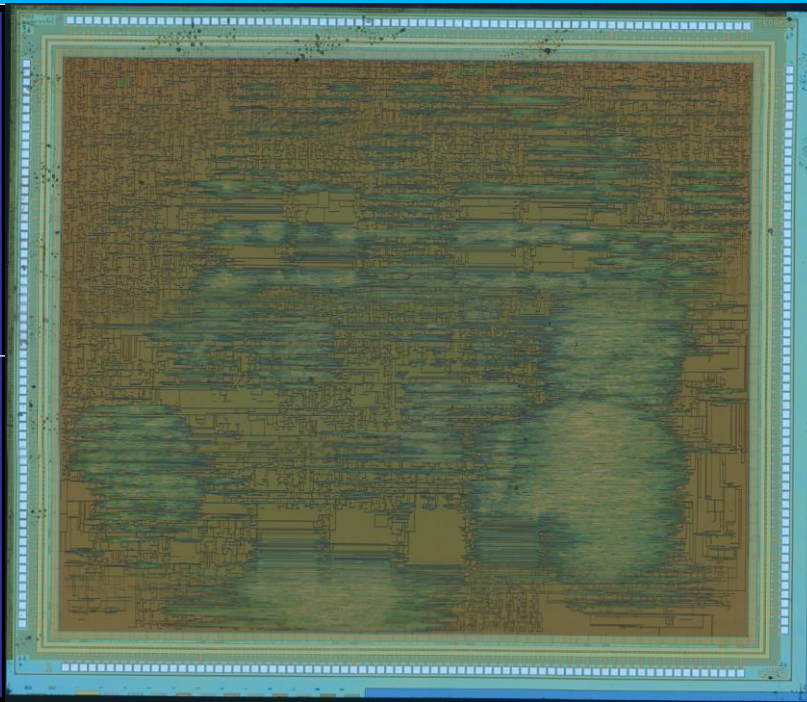
### Features

- Cores: Rocket-chip (**x4**)
- ISA: RV64GC  
*(crypto-cores aren't included)*
- Size:  $4,512\mu m \times 7,172\mu m$
- Fmax: 92 MHz
- Power: 391.125 mW
- Process: ROHM 180nm
- Fabricate: 10/2019

## 6. Conclusion (2/4)



Layout



Barechip

### Features

- Core: Rocket-chip (x2)
- ISA: RV64GC
- Crypto-cores: SHA3-512, AES-128/256, Ed25519 (*both Mult and Sign*)
- Other: QSPI (*for Flash*), USB1.1

- Size:  $4,573\mu m \times 4,578\mu m$
- Fmax: 98 MHz
- Power: 706.635 mW

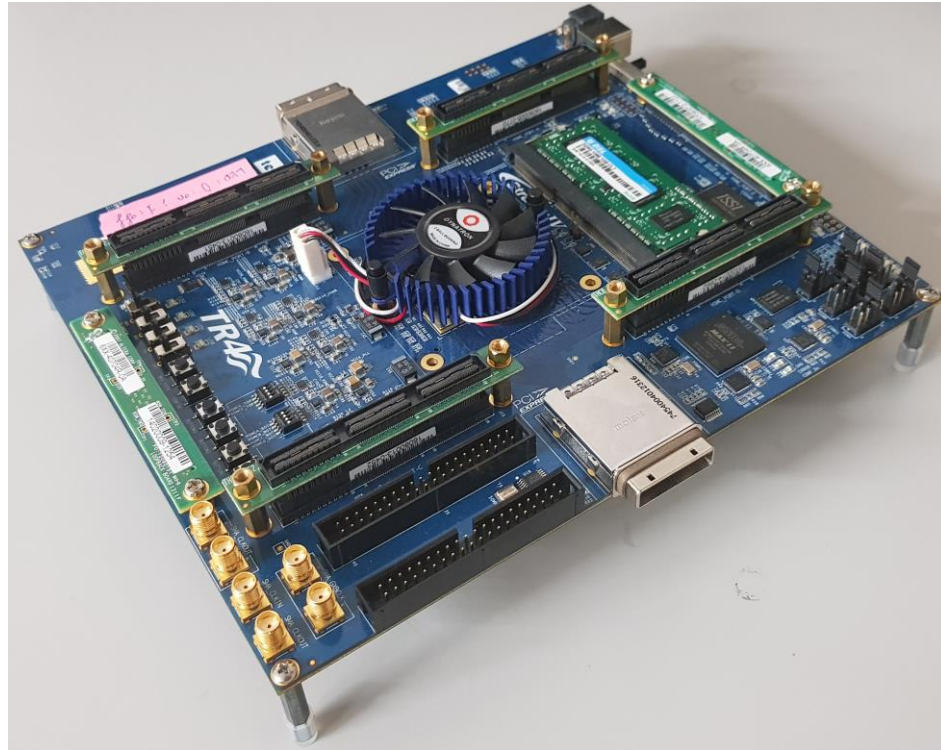
- Process: ROHM 180nm
- Fabricate: 01/2020



## 6. Conclusion (3/4)

Solving the DDR problem for the chip by:

1. Using the DIMM RAM in the TR4
2. Having the PCB (*with socket-chip*) mounted on the TR4



## 6. Conclusion (4/4)

- We presented a system platform for Trusted Execution Environment (TEE) featuring crypto-cores accelerators.
- Completed TEE-Hardware system was developed with various configurations to fit specific needs; such as core options, hybrid options, ISA options, etc.
- The system was implemented and tested on various FPGAs (*VC707*, *DE4*, *TR4*) and ASIC (*ROHM-180nm*).
- The execution time of the TEE with hardware accelerators dropped significantly compared to software.



国立大学法人  
電気通信大学



**THANK YOU FOR YOUR LISTENING**