Technology Research Association of Secure IoT Edge application based on RISC-V Open architecture

**TEE Hardware for RISC-V Implementation**

**Authors:** Ckristian Duran (UEC), Trong-Thuc Hoang (UEC/AIST), Akira Tsukamoto (AIST), Kuniyasu Suzaki (TRASIO/AIST), and Cong-Kha Pham (UEC)
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:

- Chisel
- BOOM Core
- Diplomacy
- FireSim
- FIRRTL
- Rocket Core
- TileLink
- Configuration System
- RISC-V
- Accelerators
- Caches
- Peripherals
- HAMMER

Goal:

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

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)
Based on Chipyard, a TEE-Hardware system is developed: [https://github.com/uec-hanken/tee-hardware](https://github.com/uec-hanken/tee-hardware)
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 (1/10)

**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 (2/10)

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

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

**Verifier (client)**

**TCP or UART**

**Remote PC**

**Untrusted Side**

**Enclave host**

**Linux OS**

**SM (Security Monitor)**

**Trusted Side**

**U-mode**

**S-mode**

**M-mode**

**TEE-HW System on FPGA**

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 (3/10)

**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 (4/10)

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

1. Connection with the Enclave host

```
# ./trusted_client.riscv 127.0.0.1
[TC] Connected to enclave host!
```
2. Trusted Execution Environment (5/10)

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

1. Connection with the Enclave host
2. Verify attestation report
2. Trusted Execution Environment (6/10)

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

An example of attestation report:

```
[TC] Connected to enclave host!
   === Security Monitor ===
Hash: e56168887f2d0748cf7cd57c3b46c6a60fd8b8c80a852e4c134326efaaa6259f5c8c4a38543
514612d685baaf6de15edf330d4b74e7bf0f5405257029e79fcd20
Pubkey: cd98f4a28a8523ba8edcd31175aa0e2330b246e7034545254660126a9f3b8cb9
Signature: d5c4b1647d444d5b76bd5ecf24ad5e774cb761b4a7a864c754683b1a8db8340adcf4d
6ebf7da099d35ef7aeef26834e5ffbdd86ca7815a6a6602cc4ee721a14f01

=== Enclave Application ===
Hash: 84b2193e0f5ec391672d9f68415f5bf8a928e1a25b89dfb88257d7a3becf310229d8bed1534
5884f90f69792f6da237d8b6a9d55abe254f70b4181961989cbe7b
Signature: c443ec369888c4065dbaaaad9210f1d967bae5395cb3e679ca29a1aa8aaf34ff2
80e597ace229b1a87d29acaec5d5db2b93c2cc4d15b5990902a19a181b0e
Enclave Data: 8d571e0103e72ee6986407e67d5789dd8bc3318d663e519890f078f66b05ef57
   -- Device pubkey --
0faad4ff01178583bba588966f7c1ff32564dd17d7dc2b46cb50a84a69270b4c
```
2. Trusted Execution Environment (7/10)

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

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

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

---

Verifier (client)

Device Public Key

1. Check if values are the same

2. Verify SM

3. Verify enclave
2. Trusted Execution Environment (8/10)

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

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

![Diagram showing the process of TEE in-action using Keystone](image-url)
2. Trusted Execution Environment (9/10)

**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
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 (1/6)

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
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 (3/6)

- System Bus (SBUS), Memory Bus (MBUS), and Peripheral Bus (PBUS) hierarchy.
- Several Peripheral devices for IO (GPIO, MMC, UART, PCIe, USB), memory (DDR, SPI ROM, Mask ROM), and Crypto-cores (SHA-3, ED25519, AES, PRNG)
### 3. TEE-Hardware System (4/6)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Available option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOARD</td>
<td>- VC707</td>
<td>Select the FPGA board</td>
</tr>
<tr>
<td></td>
<td>- DE4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- TR4</td>
<td></td>
</tr>
<tr>
<td>ISA CONF</td>
<td>- RV64GC</td>
<td>Select the ISA</td>
</tr>
<tr>
<td></td>
<td>- RV64IMAC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- RV32GC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- RV32IMAC</td>
<td></td>
</tr>
<tr>
<td>MBUS</td>
<td>- MBus64</td>
<td>Select the bit-width for the memory bus</td>
</tr>
<tr>
<td></td>
<td>- MBus32</td>
<td></td>
</tr>
<tr>
<td>BOOTSRC</td>
<td>- BOOTROM</td>
<td>Select the boot source</td>
</tr>
<tr>
<td></td>
<td>- QSPI</td>
<td></td>
</tr>
<tr>
<td>PCIE</td>
<td>- WPCIe</td>
<td>Include PCIe module in the system</td>
</tr>
<tr>
<td></td>
<td>- WoPCIe</td>
<td>Remove PCIe module from the system</td>
</tr>
<tr>
<td>DDRCLK</td>
<td>- WSepaDDRClk</td>
<td>Separate DDR-clock with System-clock</td>
</tr>
<tr>
<td></td>
<td>- WoSepaDDRClk</td>
<td>Not separate DDR-clock with System-clock</td>
</tr>
<tr>
<td>HYBRID</td>
<td>- Rocket</td>
<td>Two Rocket cores</td>
</tr>
<tr>
<td></td>
<td>- Boom</td>
<td>Two Boom cores</td>
</tr>
<tr>
<td></td>
<td>- RocketBoom</td>
<td>Rocket core 1&lt;sup&gt;st&lt;/sup&gt;, Boom core 2&lt;sup&gt;nd&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td>- BoomRocket</td>
<td>Boom core 1&lt;sup&gt;st&lt;/sup&gt;, Rocket core 2&lt;sup&gt;nd&lt;/sup&gt;</td>
</tr>
</tbody>
</table>

In the Makefile system, these variables are available.

Example usage:

```makefile
BOARD=VC707
ISA CONF=RV64GC
MBUS=MBus64
BOOTSRC=BOOTROM
PCIE=WoPCIe
DDRCLK=WoSepaDDRClk
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 : sifive,rocket0
```

### 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 : rv64imafdc
mmu : sv39
uarch : sifive,rocket0

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

### RV32GC

```
# cat /proc/cpuinfo
hart : 0
isa : rv64imac
mmu : sv32
uarch : sifive,rocket0

hart : 1
isa : rv64imac
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:

<table>
<thead>
<tr>
<th>ISACONF</th>
<th>HYBRID</th>
<th>FPGA logic utilization (LUT) <em>(on VC707)</em></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Core0</td>
<td>Core1</td>
</tr>
<tr>
<td>RV64GC</td>
<td>Boom</td>
<td>Boeing</td>
</tr>
<tr>
<td></td>
<td>Rocket</td>
<td>Rocket</td>
</tr>
<tr>
<td></td>
<td>Boom</td>
<td>Rocket</td>
</tr>
<tr>
<td></td>
<td>Rocket</td>
<td>Boeing</td>
</tr>
<tr>
<td>RV64GC</td>
<td>Rocket</td>
<td>Rocket</td>
</tr>
<tr>
<td>RV64IMAC</td>
<td>Rocket</td>
<td>Rocket</td>
</tr>
<tr>
<td>RV32GC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RV32IMAC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
- AES-128/256
- Ed25519 (*genkey and signature*)
- PRNG (*Pseudo-random generator*)
Some feature notes

• Crypto-Core can be implemented as a custom instruction (ROCC)
• AES supports on-the-fly 128 and 256 bits, and can be changed
• Ed25519 contains:
  Ed25519-Mult for pair-key generation
  Ed25519-Sign for signature verification
• PRNG uses LFSR (*Linear-Feedback Shift Register*); and is based on ARM TrustZone RNG register model
<table>
<thead>
<tr>
<th></th>
<th>SHA-3</th>
<th>AES-128/256</th>
<th>Ed25519</th>
<th>Ed25519</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mult</td>
<td>Sign</td>
</tr>
<tr>
<td>ALUT</td>
<td>8,108</td>
<td>3,195</td>
<td>2,737</td>
<td>3,969</td>
</tr>
<tr>
<td>Registers</td>
<td>2,790</td>
<td>2,854</td>
<td>4,778</td>
<td>4,617</td>
</tr>
<tr>
<td>Fmax (MHz)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Memory</td>
<td>0</td>
<td>0</td>
<td>8KB</td>
<td>0</td>
</tr>
<tr>
<td>DSP block</td>
<td>0</td>
<td>0</td>
<td>48</td>
<td>130</td>
</tr>
<tr>
<td>Total (%)</td>
<td>1.1</td>
<td>0.6</td>
<td>3.3</td>
<td>5.9</td>
</tr>
</tbody>
</table>
# 4. Crypto-core Accelerators (4/6)

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

<table>
<thead>
<tr>
<th></th>
<th>SHA-3</th>
<th>AES-128/256</th>
<th>Ed25519</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Size</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1,150µm ×</td>
<td>808.96µm ×</td>
<td>1,694.72µm ×</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1,150µm</td>
<td>806.4µm</td>
<td>1,693.44µm</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1,346.56µm ×</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1,345.68µm</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Gate-count</strong> (NAND)</td>
<td><strong>Mult</strong></td>
<td><strong>Sign</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>102,500</td>
<td>50,560</td>
<td>222,432</td>
<td>140,442</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Fmax (MHz)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>104</td>
<td>90</td>
<td>106</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Power (mW)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>42.745</td>
<td>37.566</td>
<td>53.061</td>
<td>80.894</td>
<td></td>
</tr>
</tbody>
</table>
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 in SHA-3 operation throughput.

- 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)

COREPLEX

- RISC-V CORE 1
  - I$
  - D$

- RISC-V CORE 2
  - I$
  - D$

TILELINK SYSTEM BUS (SBUS)  L2$ Bank

TILELINK PERIPHERAL BUS (PBUS)

- SHA-3
- ED 25519
- AES 128 & 256
- UART
- SPI (as MMC)
- GPIO

- PCIe
- PRNG
- USB 1.1
- PRCI & CLINT
- SPI (as ROM)
- Mask ROM
- Debug

MBUS

- TL sync
- TL to AXI4
- DDR controller

QSPI: to use Flash outside
5. Other Hardware Modules (2/4)

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

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

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)

**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

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

**Barechip**
- 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.
The presented work is based on results obtained from a project (JPNP16007) commissioned by the New Energy and Industrial Technology Development Organization (NEDO), and Technology Research Association of Secure IoT Edge application based on RISC-V Open architecture.
THANK YOU FOR YOUR LISTENING