RISC-V Day Tokyo 2024 Winter

An Implementation Of SHA3 Accelerator Using RISC-V RoCC Coprocessor On ArtyA7 100T Board

Duc-Thuan Dam, Trong-Thuc Hoang, and Cong-Kha Pham
University of Electro-Communications (UEC), Tokyo, Japan
Corresponding author: ducthuan@vlsilab.ee.uec.ac.jp

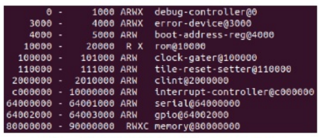
UEC

Pham Laboratory
Integrated circuit design laboratory

ABSTRACT

In 2010, the RISC-V instruction set architecture (ISA) marked a milestone in the design of microprocessor systems that could be customized following a modular mindset. The RISC-V is a high-potential solution to implement encryption algorithms most efficiently. In 2015, SHA3 was chosen as the standard for hash functions by NIST [1]; SHA3 generates hash values and uses various post-quantum cryptography (PQC) algorithms. Rocket chip [2], the most common RISC-V SoC, also supports a coprocessor connected directly to the CPU to perform specific tasks, known as an accelerator. It has been introduced to the coprocessor via the RoCC interface, and the CPU will control operations via custom commands. Following the instructions in [3, 4], we implemented this design on a Xilinx ArtyA7 100T board. We evaluated the SHA3 accelerator's performance through the number of clock cycles required for the computation by running pure software on the SoC and running soft hardware which calls the coprocessor. The results show that using SHA3 RoCC helps calculate the hash value much faster (~2000x) than performing by pure software.

I. PROPOSED ARCHITECTURE



```
       0 -      1000 ARWX  debug-controller@0
    3000 -      4000 ARWX  error-device@3000
    4000 -      5000 ARW   boot-address-reg@4000
   10000 -     20000  R X  rom@10000
  100000 -    101000 ARW   clock-gater@100000
  110000 -    111000 ARW   tile-reset-setter@110000
 2000000 -   2010000 ARW   clint@2000000
 c000000 -  10000000 ARW   interrupt-controller@c000000
64000000 -  64001000 ARW   serial@64000000
64002000 -  64003000 ARW   gpio@64002000
80000000 -  90000000  RWXC memory@80000000
```

Fig. 1: SoC Address Map

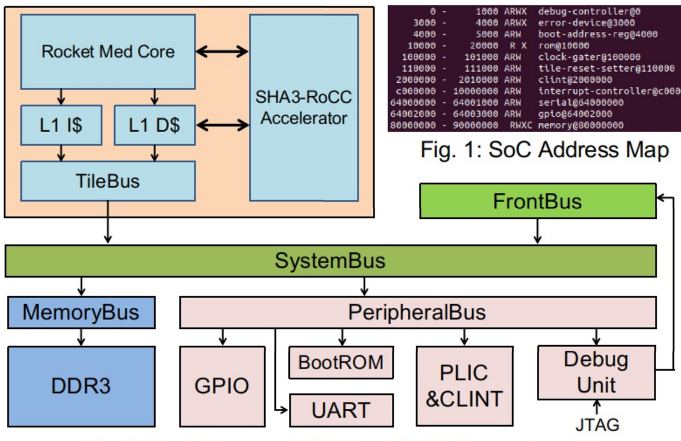


Fig. 2: Architectrure of SoC on ArtyA7 target board

- The SoC system includes a single medium Rocket core, which integrates the SHA3 coprocessor via the RoCC Interface.
- A UART is attached to print the result to the serial terminal, and a Debug unit is used for debugging using RISC-V GDB tools [5].

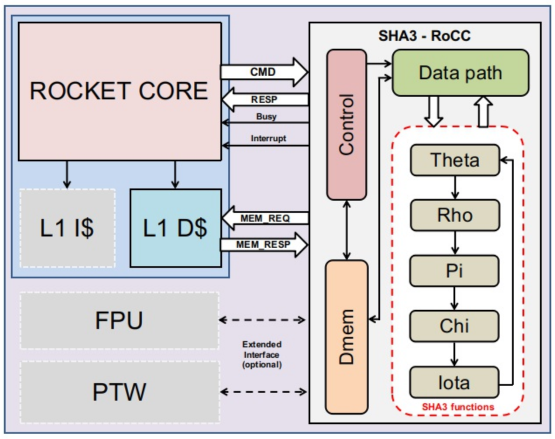


Fig. 3: The SHA3-RoCC accelerator architecture

- SHA3-RoCC includes SHA3 function blocks, data, and control modules.
- The core sends a customized instruction to the accelerator and waits for the response.
- The accelerator performs the data from L1D following the instruction, then pushes back the result to L1D.

II. TESTING SYSTEM

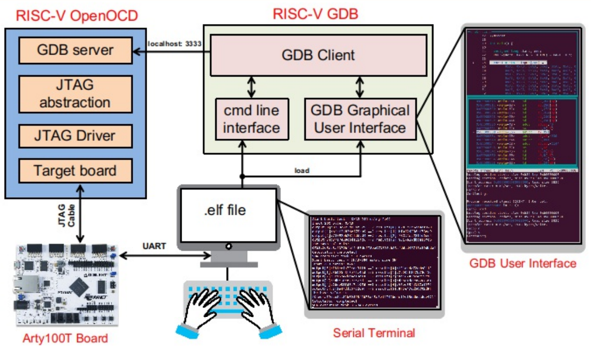


Fig. 4: The demo system connection diagram

- The software is run on SoC bare metal.
- The C codes are compiled to .elf file, then loaded to CPU by GDB using OpenOCD and results are printed in minicom terminal through UART.

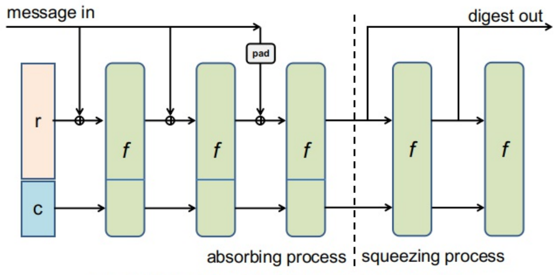


Fig. 5: The Keccak Sponge construction

III. EXPERIMENTAL RESULTS

Table 1: The scenario of practical testing

Parameter	Specification
CPU	rv64imaczicsr_zifencei_zihpm_xrocket
Digest size	224, 256, 384, 512 (bit)
Input	1600 bit (0xA3)**
Padding*	SHA3, Keccak

Table 2: The experimental execution cycle counts

Standard	Block Length		No. of round	Pad length	Execution cycles	
	r	c			SW	RoCC
Keccak/SHA3-224	1152	448	2	704	285507	-
Keccak/SHA3-256	1088	512	2	576	279635	139
Keccak/SHA3-384	832	768	2	64	268048	-
Keccak/SHA3-512	576	1024	3	128	399710	-

* Padding rule:
- Keccak: 0x01 0x00 .... 0x00 0x80
- SHA3: 0x06 0x00 .... 0x00 0x80

** The input and output of the test is following example values published by NIST [4]

REFERENCES

[1] NIST FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable – Output Functions, https://doi.org/10.6028/NIST.FIPS.202
[2] https://chipyard.readthedocs.io/en/stable/Generators/Rocket-Chip.html
[3] https://chipyard.readthedocs.io/en/stable/Customization/RoCC-Accelerators.html
[4] NIST, Cryptographic Standards and Guidelines https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/example-values
[5] GDB: The GNU Project Debugger, https://www.sourceware.org/gdb/documentation/