Lightweight Secure Processor Prototype on FPGA

Yao Liu Ray C.C. Cheung and Hei Wong
City University of Hong Kong, Kowloon Tong, Hong Kong
Email: liu.yao@my.cityu.edu.hk, {r.cheung, h.wong}@cityu.edu.hk

***Abstract*—Lightweight devices usually come with an additional cryptographic co-processor for enabling the secrecy, in contrast, the master processor is typically a commercial processor where the required protection mechanism is missing. In this paper, an on-going effort in secured architecture named S-RISC-V based on RISC-V core is introduced. The mechanism of key generation used for memory protection is supported together with the joint efforts in the following perspectives, including ISA extension, compiler improvement, and hardware implementation. The architecture has been verified on Zedboard running at 25MHz, driven by the host ARM core. The area overhead is less than 10%, compared with the original RISC-V core.**

***Index Terms*—RISC-V; key generation; secure processor**

I. Introduction

The security of lightweight devices has become a crucial issue alongside the emerging trend of IoT (Internet of Things). Although the virtualized systems offer extensive software and hardware measures to protect the secrets, those lightweight systems rely on physical modules. A ubiquitous solution is to apply a cryptographic coprocessor to deal with the complicate cryptographic operations and to store the secrets in the non-volatile memory. However, the main processor itself is usually commercially authorized, lack of security extensions.

RISC-V promotes the concept of open Instruction Set Architecture (ISA), which provides the vast flexibility in the design of the processor and its ecosystem. A Scala embedded language, Chisel, offers a new design perspective to design and customize a processor, together with the open source RISC-V [1]. Chisel provides the hardware design with properties of higher-level program language, while the traditional Verilog design based on Register-Transition-Level (RTL) descriptions focuses too much on the distractible timing behavior. This shows great superiority in the complex system design, such as microprocessor. It is convenient for the user to customize a RISC-V core, and to translate it into Verilog code, with the open-source project framework and tools. Since Chisel is incompatible with Electronic Design Automation (EDA) tools, hybrid design is preferable, if some user-defined modules or IPs need to be integrated into the RISC-V core.

In this work, we present S-RISC-V, a preliminary architecture of the secure processor that allows architecture extensions such as key generation mechanism. It applies the framework of a single Rocket RISC-V core [2], with ISA extension, compiler supplement and hardware modification. The key generation process is verified on Zedboard with the frequency of 25MHz.

TABLE I
RESOURCES FOR DIFFERENT TYPES OF COMPUTING DEVICES

	Hypervisor	OS	Processor
Severs, PCs	complex	complex	complex
Mobile devices	limited	compact	compact
Lightweight devices	none	limited	limited

II. Protection of microprocessors

Computing devices cover a wide range of devices, from systems as complex as a cloud server, to cheap devices as compact as RFID tags in an IoT network. Table I briefly compares the availability of resources in different types of computing devices. Due to the constrained architectural support, lightweight devices lack of visualization and only support the simplified Operation Systems (OS). The hierarchical protection from software and hardware efforts, which is the preferable choice for computers and mobile phones, is beyond the ability of a lightweight computing system. To the contrary, lightweight devices can only apply hardware modules to safeguard the computing system.

Typically, a main general-purposed microprocessor and a cryptographic coprocessor form a lightweight security system. Because of the limited computing capability of lightweight devices, the cryptographic coprocessor usually undertakes the complicate cryptographic operations, together with other cryptographic primitives, such as True Random Number Generators (TRNGs) and Physical Unclonable Functions (PUFs). The main processor is merely a task handler, running the pre-installed programs in the non-volatile memory. Unlike the well-protected coprocessor, the main processor is vulnerable to external attacks.

III. Preliminary architecture of S-RISC-V

Aiming to construct a secure main processor prototype for lightweight applications, we propose S-RISC-V based on the Rocket Chip implementation of RISC-V ISA [2]. The preliminary architecture of S-RISC-V is shown in Fig. 1.

Once a transaction outside is initiated, a new session key should be generated to protect the internal secrets from the adversaries. A new instruction *keygen* is extended for internal secure execution. A True Random Number Generator (TRNG) is embedded in the processor to perform the key generation instruction, which applies the staged-running Self-timed Ring (STR) architecture proposed by Liu et al [3]. This architecture is able to generate high-quality unpredictable random bitstreams on purely digital circuits with low area overhead,

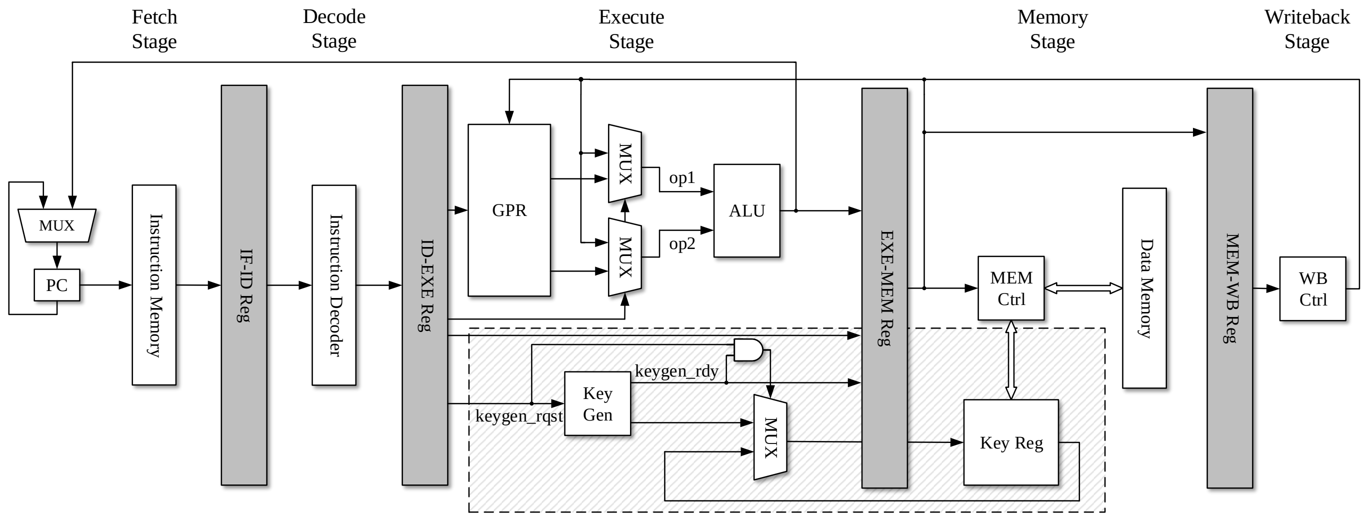

Fig. 1. The brief architecture of S-RISC-V based on the 5-stage pipeline of Rocket RISC-V core. The dashed block contains the main modification. PC: Program Counter; GPR: General-Purpose Registers; ALU: Arithmetic Logic Unit; MUX: Multiplexer.

which has been proven to pass the NIST test. LUT6 (6-input Look-up Table) primitives are utilized to construct STR nodes with the staged-running mechanism, the truth tables of which are directly set in the Verilog code. This method helps to form the nodes with identical timing characteristics. Additionally, a proper manual placement of these nodes is able to form identical timing delay between neighboring nodes. Since the recent versions of Vivado are able to optimize the feedback of LUT primitives, the STR should be set with *DONT_TOUCH* attribute. Besides, the timing path of the STR should be set as *asynchronous*, to avoid the interference with other synchronous modules. All the above measures effectively improve the performance of TRNG.

An internal register *KeyReg* is added to carry the generated session key during a transaction, and is inaccessible by other instructions. *keygen* can be added without interrupting the normal operation of the classic 5-stage pipeline, because *KeyReg* is an individual register. Execute stage and Memory stage are mainly affected, as shown in the dashed block in Fig. 1. The improvement of the architecture is implemented in Chisel, and then translated into Verilog.

A cross verification platform is built on Zedboard to test the functionality of the secure processor. It is convenient to apply the hardcore ARM Cortex-A9 as the driver, which plays the role of coordinator between the host computer and the RISC-V core, since the embedded system on ARM is well-supported from the tool vendor. After powering up, Zedboard configures the PL (Program Logic) part and boots up the PS (Processing System) part. When the ARM core loads the Linux image, the device-tree and the filesystem from the pre-installed SD card, the user is able to access the ARM core with UART from a host computer. The RISC-V core is implemented in the PL part, driven by the ARM core through the AXI Bus in the debug mode. The cache for the RISC-V utilizes the resources in PL, which is automatically generated by the tool.

TABLE II
RESOURCE UTILIZATION ON XILINX XC7Z020CLG484-1

	Slice	LUT	DFF	BRAM	DSP
Rocket [2]	9541	34508	15572	13	24
S-RISC-V	10439	34793	15804	13	24

The RISC-V core, cache and AXI Bus all run at 25MHz.

Table II shows the hardware utilization of the RISC-V core with the key generation extension after Place and Route. The DFF overhead for the key generation mainly contributes to the area overhead.

IV. Future work

The extended instruction *keygen* is essential for the secure operation of S-RISC-V. Due to the lack of the powerful supervision from the OS, it can be possibly skipped by a malicious user. In the future work, we plan to further improve the privileged instructions of the secure processor from the architectural level, in order to guarantee the hardware protection mechanism can not be bypassed by the malicious code. Afterwards, the security of the lightweight secure processor prototype will be verified with IoT protocols, such as 6LoWPAN and so on.

References

[1] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avižienis, J. Wawrzynek, and K. Asanović, “Chisel: constructing hardware in a Scala embedded language,” in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 1216–1225.

[2] K. Asanović, R. Avizienis, J. Bachrach, *et al.*, “The Rocket chip generator,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17, 2016. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html

[3] Y. Liu, R. C. Cheung, and H. Wong, “A bias-bounded digital true random number generator architecture,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 1, pp. 133–144, 2017.