Lightweight security accelerators for RISC-V

19th CryptArchi workshop

Alberto Josué Ortiz Rosales
Brisbane Ovilla Martínez
Cuauhtemoc Mancillas López

June 13 2023

Cinvestav
DEPARTAMENTO DE COMPUTACIÓN

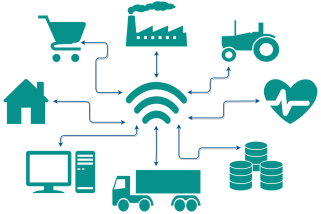
Table of Contents

- Open Standard ISA.
- Modular sets of instructions.
- Implemented from high-performance systems to embedded devices.
- Highly customizable according to the end use.

Embedded devices security

1 Introduction

In recent years, there is a widespread adoption of embedded devices in various sectors. Even with restricted resources, these devices must offer security.



Our approach

Design of lightweight (LW) acceleration modules to provide the services of confidentiality, integrity, secure storage, and randomness. Focusing on the LW algorithm ASCON.

Related work

Cryptographic Accelerators for Trusted Execution Environment in RISC-V Processors (Hoang, 2020)

- SHA-3
- PRNG
- EC 25519
- AES 128, 256

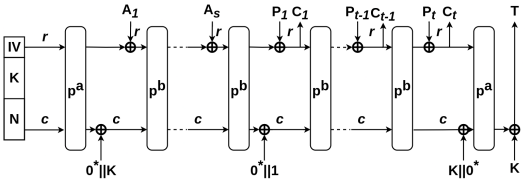
Lightweight Secure-Boot Architecture for RISC-V System-on-Chip (Haj-Yahya, 2019)

- Key management unit
- ECDSA
- SHA-3

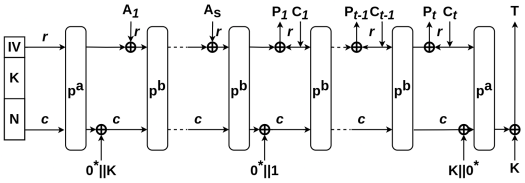
Ascon Modes

1 Introduction

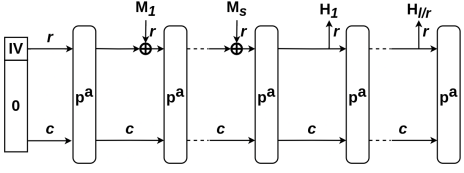
Encryption



Decryption

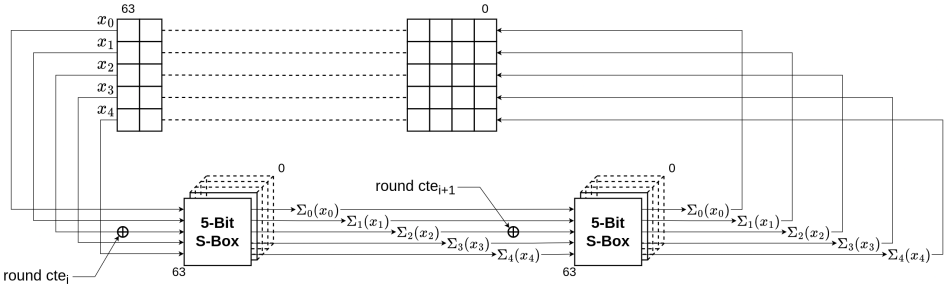


Hash



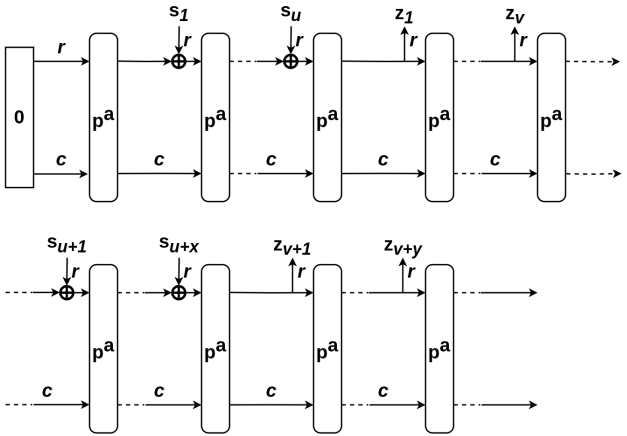
- $a = 12, b = 6$
- $|r| = 64$
- $|c| = 256$

Two unrolled round version



PRNG Construction

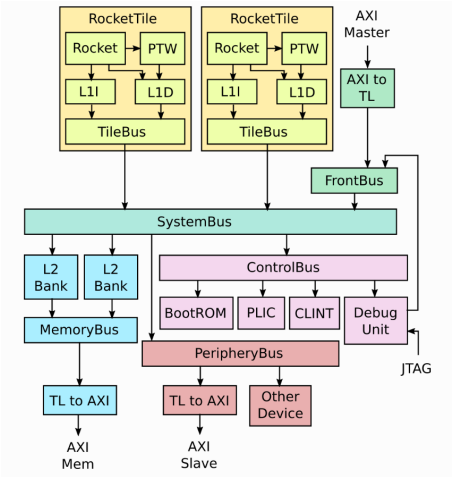
- Using the reseedable the sponge construction[a].
- $a = 12$ $r = 64$



[a] Guido Bertoni et al. (2010). "Sponge-Based Pseudo-Random Number Generators.". In: *CHES*. vol. 6225. Springer, pp. 33–47.

RocketChip

- Open-source RISC-V processor generator.
- Agile development of **Chisel** based System-on-Chip.
- Software development co-design.



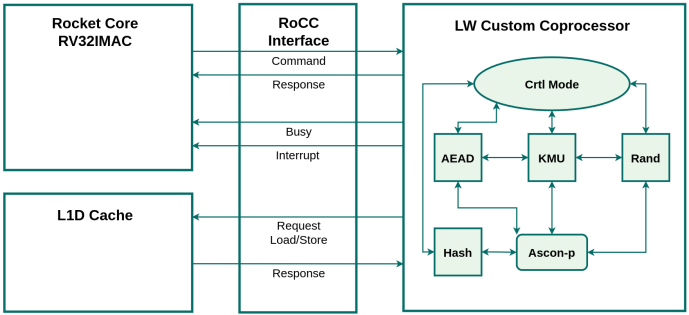
Development framework

For the prototyping and testing of the modules, two open-source development kits by SiFive were used:

- freedom: RTL for prototyping E300 (Everywhere) platforms to be mapped onto an Arty FPGA.
- freedom-e-sdk: SDK to develop software for the Freedom E platforms.

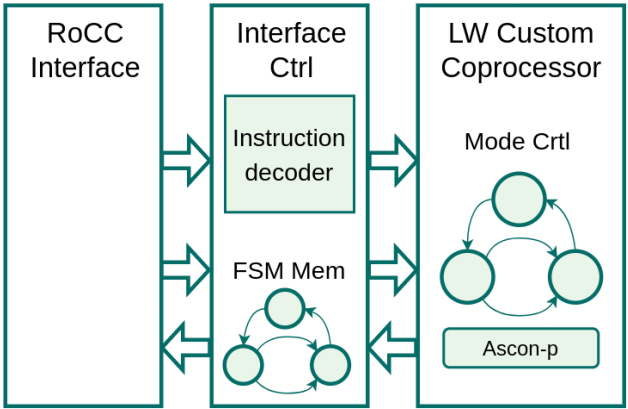
Rocket Custom Coprocessor

2 Hardware Development



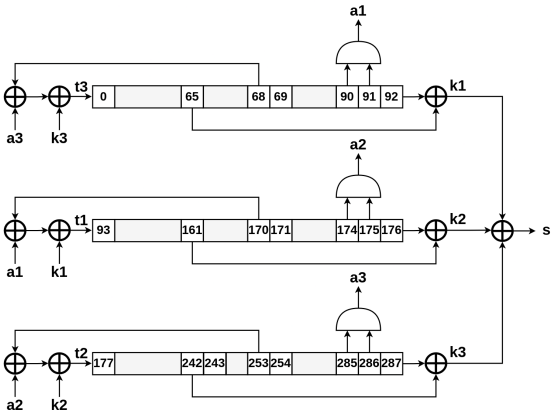
- **AEAD:** Implementation of ASCON-128.
- **Hash:** ASCON-Hash algorithm.
- **Rand:** PRNG uses Trivium as a seed generator.
- **KMU:** Generation, storage, and distribution of symmetric keys.
- **Ascon-p:** Ascon's permutation.

Interface


RoCC Interface
Interface Ctrl
Instruction decoder
FSM Mem
LW Custom Coprocessor
Mode Crtl
Ascon-p

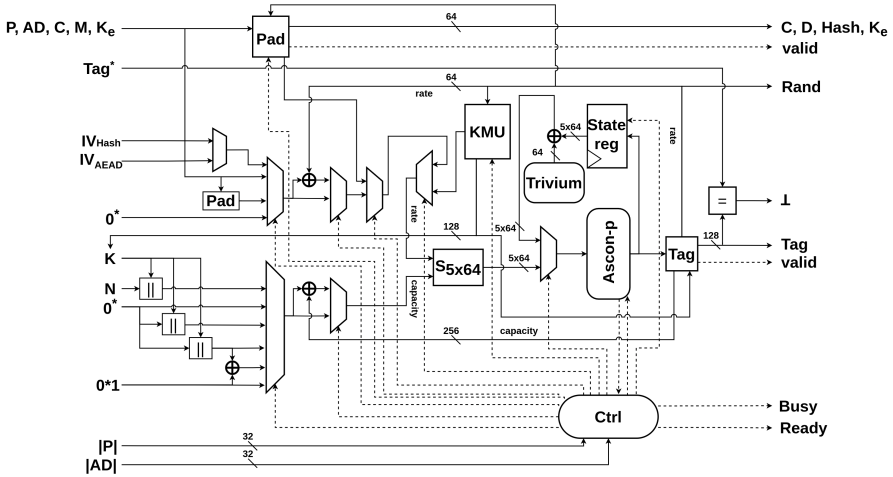
PRNG Seed

- As a proof of concept the stream cipher is used for the seed generation[a].
- 64 bit version of trivium.
- Will be replaced by a TRNG.



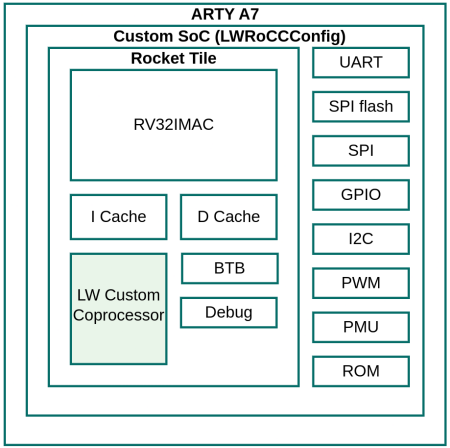
[a] Christophe De Canniere and Bart Preneel (2008). "Trivium". In: *New Stream Cipher Designs: The eSTREAM Finalists*, pp. 244–266.

- **C** Cipher text
- **P** Plain text
- **AD** Associated data
- **D** Deciphered text
- **M** Input message
- **K** Key
- **N** Nonce
- **IV** Initialization vector
- **Ke** Encrypted key



Custom RocketChip SoC

2 Hardware Development



SoC Configuration

```
class LWRoCCConfig extends Config(
    new WithNBreakpoints(2) ++
    new WithNExtTopInterrupts(0) ++
    new WithJtagDTM ++
    new WithL1ICacheWays(2) ++
    new WithL1ICacheSets(128) ++
    new WithDefaultBtb ++
    new WithLWCustom ++
    new TinyConfig)
```

Instructions

Module ID

Module	funct7 ID Module
AEAD E	001xxxx
AEAD D	010xxxx
Hash	011xxxx
Rand	100xxxx
KMU	101xxxx

AEAD E Instructions (funct7(6:4) = 1)

operation	funct7 (3:0)	rd	rs_1	rs_2
Set P	1	-	[P]	P length
Set AD	2	-	[AD]	AD length
Set C Tag	3	-	[C]	[Tag]
Set Nonce	4	-	[Nonce]	-
Use Key	5	-	Key ID	-
Init Enc	6	finish	-	-

AEAD D Instructions (funct7(6:4) = 2)

operation	funct7 (3:0)	rd	rs_1	rs_2
Set C	1	-	[C]	C length
Set AD	2	-	[AD]	AD length
Set D Tag	3	-	[Dec]	[Tag*]
Set Nonce Dec	4	-	[Nonce]	-
Use Key	5	-	Key ID	-
Init Enc	6	valid	-	-

Hash instructions (funct7(6:4) = 3)

operation	funct7 (3:0)	rd	rs_1	rs_2
Set M	1	-	[M]	M length
Set Hash	2	-	[Hash]	-
Init Hash	3	valid	-	-

PRNG instructions (funct7(6:4) = 4)

operation	funct7 (3:0)	rd	rs_1	rs_2
Seed	1	-	-	-
Get Rand	2	counter	[Rand]	-

KMU instructions (funct7(6:4) = 5)

operation	funct7 (3:0)	rd	rs_1	rs_2
Set New key	1	-	ID	-
Get key	2	-	ID	[Key]
Send key	3	-	ID	[Key]
Delete key	4	-	ID	-

Utilization

3 Results

Proposed module utilization

Module	Total LUTs	Logic LUTs	LUTRAMs	SRLs	FFs	RAMB36	RAMB18	DSP
LWCoprocessor	2180	2180	0	0	1101	0	0	0
Ascon-p	1354	1354	0	0	647	0	0	0

SoC utilization

Module	Total LUTs	Logic LUTs	LUTRAMs	SRLs	FFs	RAMB36	RAMB18	DSP
Default SoC	15040	14394	586	60	9873	8	2	2
Custom SoC	17288	16616	612	60	11215	8	2	2
Increase %	14.94	15.43	4.43	0	13.59	0	0	0

Conclusions

4 Conclusions

- By effectively managing the data flow into Ascon's permutation, the security services such as: confidentiality, integrity, authentication, and randomness can be provided.
- Accessing the coprocessor through the same instruction format as RISC-V ISA, make its control transparent at the software level.
- Adding the custom coprocessor increases the LUT's utilization by 14.94 % and the FF's by 13.59 % against the default SoC.

Future Work

5 Future Work

- Change the permutation to offer protection against side channel attacks.
- Implement a TRNG instead of the PRNG.
- This construction is meant to be integrated in a TEE system.

Thank you for listening!
Any questions?