

# Fault attacks on RISC-V processor

09/11/2022

Thales Silicon Security (TSS) - Jean-Roch Coulon

# CVA6 core



## Open-source RISC-V application core

➤ Supports rich OSes like Linux

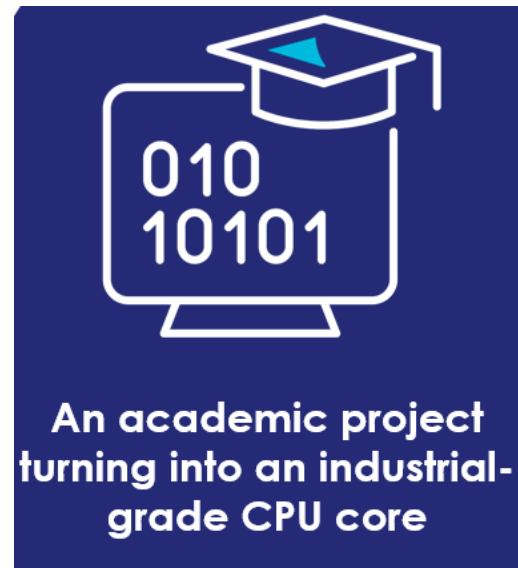
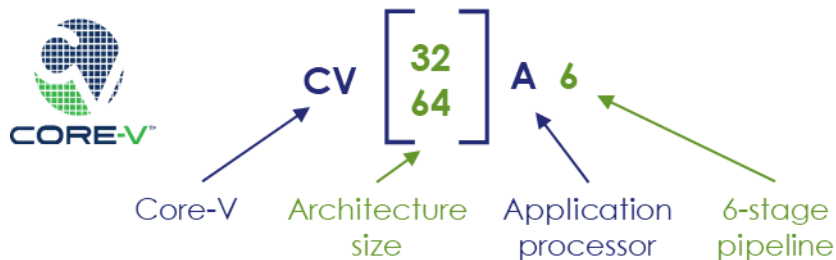
## Common source code, two flavors:

➤ CV64A6

- 64-bit
- ARIANE donated by ETH Zürich to OpenHWGroup

➤ CV32A6

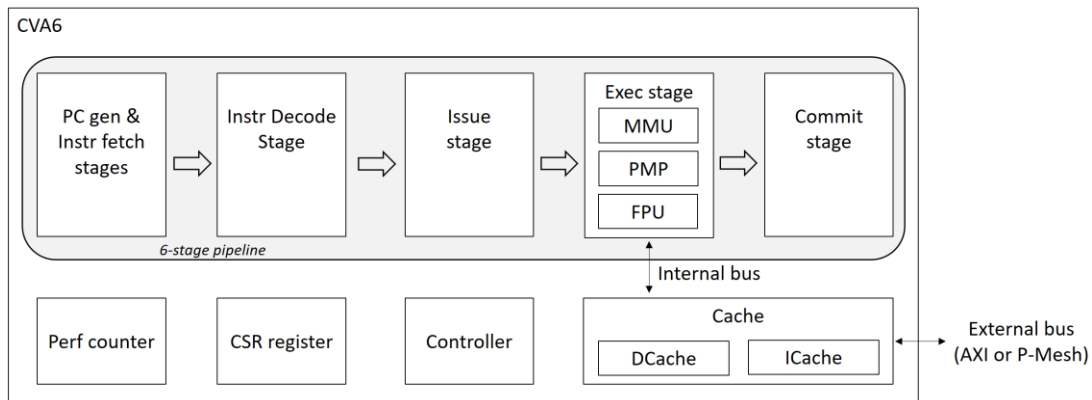
- 32-bit
- Compact version designed by Thales



# CVA6 architecture



- Fully compatible with the RISC-V open ISA
- 6-stage pipeline, single issue, branch prediction
- L1 data and instruction caches
- MMU and memory protection
- M/S/U privileges
- AXI4 interfaces
- CV-X-IF interface
- Safe and secure features
- Ready for multi/many-core CPUs



# CVA6 security vulnerabilities




# CVA6 threats













## Start from Smart Card PP84 Protection Profile (aka PP0084)

- Well recognized and established in the smartcard world
- Smartcards still remain the most secure civilian electronic devices

## Enhance with missing threats and associated counter measures

- Micro architectural attacks
- Software attacks

-  PP0084 Fault Injection
-  PP0084 Others
-  Not in PP0084

-  Leak Inherent
-  Physical probing
-  Malfunction due to environmental stress
-  Physical manipulation
-  Forced Information leakage (linked to Malfunction and Manipulation)
-  Abuse of functionality (out of scope)
-  Deficiency of Random Numbers
-  TOE identification (out of scope)
-  Authentication to external entities (out of scope)
-  Protection of the confidentiality of the TSF (linked to Manipulation)
-  Area based Memory Access Control
-  Micro architectural exploits

# CVA6 Malfunctions: Memory and registers

## ■ Corruption of data at rest in memory caches

- Implement integrity checking on both code and data caches

## ■ Corruption of data at rest in general purpose registers

- Implement random physical register location cycling (e.g. per reset)

## ■ Corruption of data in motion between registers and internal memories

- Implement redundancy field for each memory chunk in internal memories
  - Coherency checked when register is used

## ■ Data out-of-bounds by physical attack

- Check data boundaries before use



# CVA6 Malfunctions: Pipeline

## ■ Corruption of control registers

- Implement redundancy

## ■ Corruption of pipeline

- Implement integrity pipeline protection

## ■ Specific Corruption of Execute Stage

- Implement secure arithmetic operations
  - Duplicate operations or compute redundancy with data
  - Associate redundancy for possible operations (load, store, Boolean operations...)



# CVA6 Malfunctions: Control Flow

- **Address both Physical Attacks and Software Attacks**
- **Control flow corruption by physical attack or Return Oriented Programming**
  - Implement Control Flow Integrity (CFI) counter measure
- **Address corruption by physical attack**
  - Protect address integrity
  - Prohibit data accesses outside allowed address zones
  - Data integrity shall depend on address





# Fault injection before tape-out

# Objectives

## Inject faults before tape out

- To reduce the development loop duration
- To attack element one by one
- To improve the design

## Injection done by digital simulation



# Where to start?

## | Where to inject?

- Memory code or data
- Flip-flops: 24339 in CVA6
- Combinatory

## | How to inject?

- Injection duration
- Permanent or transient
- Set to 0 or 1

## | Which test?



# Cut branches!

## Unit injection first

- Step-by-step to better understand
- Then multi-faults with CAD vendor tool

## Reduce simulation time

- Single bit flip in memory and register
- One injection by register
- 24339 flip-flops => less than 200 simulations ( x 30'' = 100' )



# Methodology

- ▮ **Activate the fault with good test to avoid “not propagated”**
- ▮ **Do not overwrite the fault**
  - Flip memory or register bit with simulation force option
- ▮ **Detect attack by exception**
- ▮ **Compare executed instruction trace:**  
**SUCCEED, NOT PROPAGATED, DETECTED**



# Results

Fault type	CVA6		
	Detected	Succeeded	Not propagated
System Memory #30 fault injections	6	17	7
Register #148 fault injections	37	93	18

Fault type	SPRITZ		
	Detected	Succeeded	Not propagated
System Memory #30 fault injections	30	0	0
Register #148 fault injections	145	0	3

RISC-V and open source  
better for security?

# Open Source

- | ISA is well known
- | Implementation can be simulated
- | Silicon parts can be accessed

**BUT...**





# Open Source

## | Open source community

- Smart security Working Group
- CFI, fence.t, integrity, confidentiality, authentication, crypto,...

## | RISC-V specifies ISA

- Possible custom instructions and registers
- No implementation constraints

## | Dual version

- Public for functionality
- Private for security add-on



THALES



**ANY QUESTIONS?!?**

