



UNIVERSIDAD  
COMPLUTENSE  
MADRID

*Implementation of an electromagnetic fault  
injection platform for a RISC-V-based SoC*

# **Implementación de una plataforma para tests de inyección de fallos mediante electromagnetismo contra SoCs basados en RISC-V**

2022

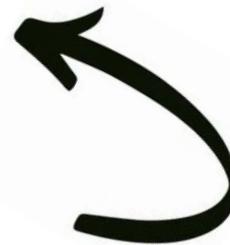
Realizado Por: Pedro Javier Fernández

Facultad de Informática

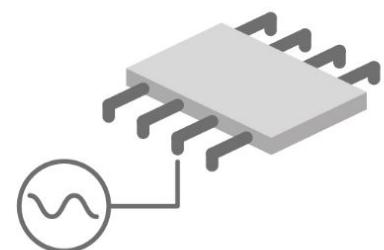
# INDEX OF CONTENTS

1. Objectives of this TFG
2. Previous research and state-of-the-art
3. Contributions to EMFI
  - Proposed EMFI solutions
4. Experiments analysis and discussion
5. Impacts on reliability and security
6. Future research topics and follow-up's
7. Conclusions

- **The Problem:** Spread of RISC-V. At the same time: increased need for reliable computers: in harmful conditions, but also due to rise in malicious uses of Fault Injection (FI).
- To evaluate the impact of Electromagnetic Fault Injection (EMFI) on several Systems-on-Chip (SoCs) that implement different versions of the RISC-V ISA.
- To develop an economically affordable EMFI device suited for academia research to do so.



## Power/Clock Glitch Fault Injection (PGFI)



**SAMPLE ACCESS**  
Only Pin access

### EQUIPMENT COST



### PRECISION

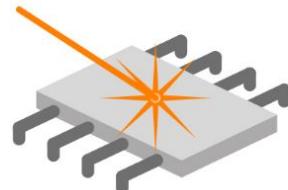


### EXPERTISE



Inject a fault by altering the clock's frequency or power supply in the corresponding pin of the chip.

## Light/Laser Fault Injection (LFI)



**SAMPLE ACCESS**  
Back side - Front side  
depackaging

### EQUIPMENT COST



### PRECISION



### EXPERTISE



Inject a fault by inducing a light pulse on the back side or front side surface of the silicon dice.

## Electromagnetic Fault Injection (EMFI)



**SAMPLE ACCESS**  
Front side-Back side  
(partial) depackaging

### EQUIPMENT COST



### PRECISION

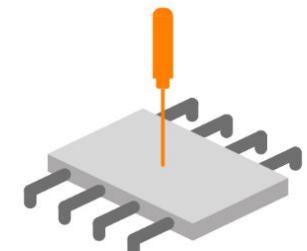


### EXPERTISE



Inject a fault by applying an electromagnetic pulse near the surface of the chip.

## Body Biased Fault Injection (BBFI)



**SAMPLE ACCESS**  
Back side  
depackaging

### EQUIPMENT COST



### PRECISION



### EXPERTISE



Inject a fault by applying a forward or reverse bias at hundred of mV to the substrate of the chip.



chipSHOUTER, 3000€

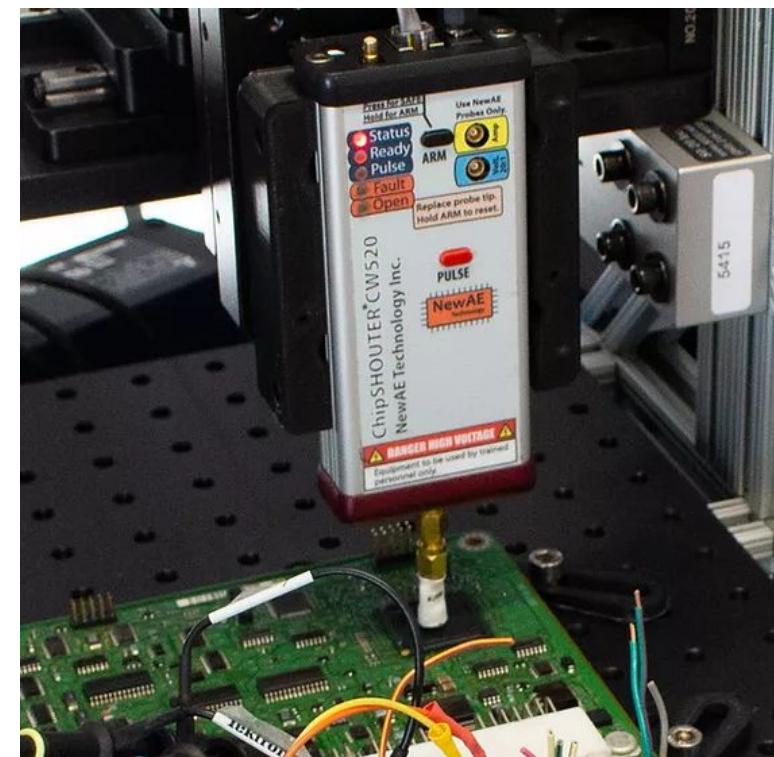
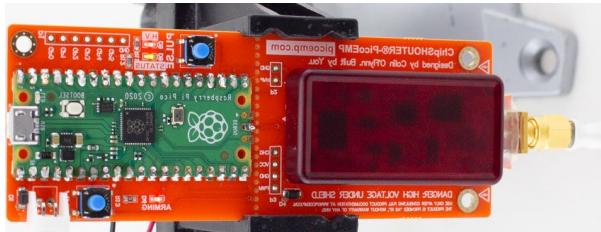
[1] Hajdu, N. Ivaki, I. Kocsis, "Using fault injection to assess blockchain systems in presence of faulty smart contracts," IEEE Access, vol. 8, pp. 190760–190783, 2020

[2] EM Fault Injection on ARM and RISC-V (Mahmoud Elmohr et al.).

[3] Fault Injection on Hidden Registers in a RISC-V Rocket Processor and Software Countermeasures

Johan Laurent, V. Berouille,

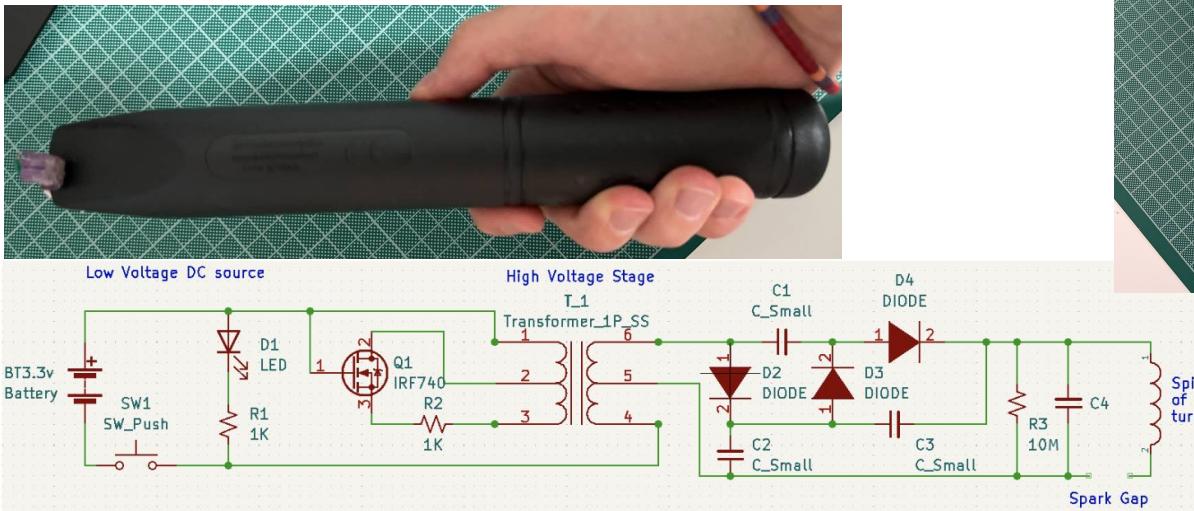
[4] PicoEMP: NewAE



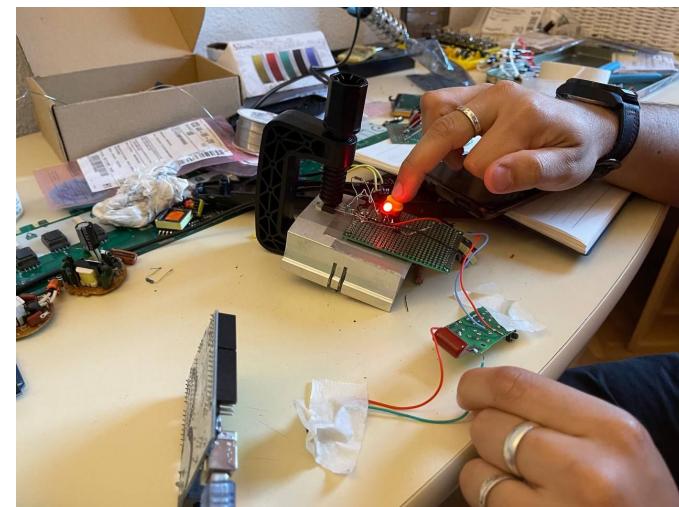
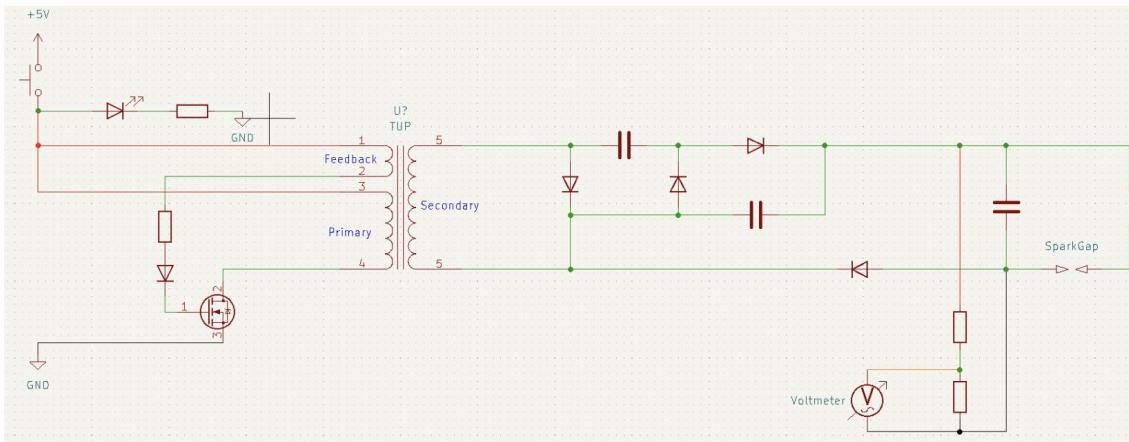
source: <https://www.newae.com/auto>

## Proposal of two devices

- **Device 1:** Repurposed mosquito zapper injector



- **Device 2:** cheapSHOUTER prototype injector



## Dynamic Experiments

```
#include <stdio.h>
int main(void){
    int i,j,k,cnt;
    k = 0;
    while(1){
        cnt = 0;
        for(i=0; i<5000; i++){
            for(j=0; j<5000; j++){
                cnt++;
            }
        }
        printf("%d %d %d %d\n", cnt, i, j,k++);
    }
}
```

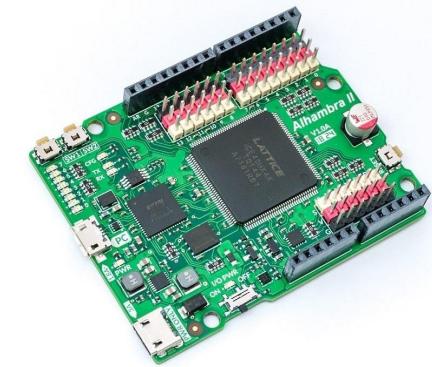
## Static Experiments

MAiX Bit with JTAG



MAiX Bit (RV64GC ISA +  
IMAFDC )

Alhambra FPGA (RV32I  
soft-core)



Longan Nano (RV32IMAC)



parameters: state of the DUT, injector used, distance to the DUT, n° of shots

Volume II: RISC-V Privileged Architectures V1.10

```
[MAIXPY]P110:freq:832000000  
[MAIXPY]P111:freq:398666666  
[MAIXPY]P112:freq:45066666  
[MAIXPY]cpu:freq:416000000  
[MAIXPY]kpu:freq:398666666  
[MAIXPY]flash:0x0-0x17
```

```
open second core...
mallocmallocmallocmallocgc h
[MaixPy] init end
```

core dump: illegal instruction  
Cause 0x0000000000000003 EB

cause 0x0000000000000002, EF reg[001(zero)]=0x00000000

**reg(02)(sp) = 0x0000000000**

[reg[04](tp )] = 0x000000000

**reg[06](t1 ) = 0x00000008**

`reg[08](s0/rp) = 0x0080000000  
reg[10](s0...) = 0x41d8cbfa80`

[reg[10](a0) ] = 0x41d8cda80  
[reg[12](a2) ] = 0x0000000000

[reg[14](a4)] = 0x00000000

reg[16](a6) = 0x00000000

reg[18](s2) = 0x00c65d400

`reg[20](s4) = 0x00000000  
reg[33](s6) = 0x00000000`

[reg[22](s0 )] = 0x00000000  
[reg[24](s8 )] = 0x00000008

[reg[26](s10 ) = 0x0000000000

reg[28](t3) = 0x0000000000

[reg[30](t5 ) = 0x0000000000  
fptr[20](fto ) = 0x0000000000

`freq[00](ft0) = 0x00000000  
freq[03](ft3) = 0x00000000`

freq[92](ft2) = 0x00000000

freq[06](ft6 ) = 0x0000000000

freq[08](fs0 ) = 0x00000000

```
[reg[10](fa0 ) = 0x0000000000
```

Interrupt	Exception Code	Description
1	0	User software interrupt
1	1	Supervisor software interrupt
1	2	<i>Reserved</i>
1	3	Machine software interrupt
1	4	User timer interrupt
1	5	Supervisor timer interrupt
1	6	<i>Reserved</i>
1	7	Machine timer interrupt
1	8	User external interrupt
1	9	Supervisor external interrupt
1	10	<i>Reserved</i>
1	11	Machine external interrupt
1	$\geq 12$	<i>Reserved</i>
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned
0	5	Load access fault
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode
0	9	Environment call from S-mode
0	10	<i>Reserved</i>
0	11	Environment call from M-mode
0	12	Instruction page fault
0	13	Load page fault
0	14	<i>Reserved</i>

uring ~~~~~DBUG\_STATUS\_BUSY~~~~~  
orting XXXXXXXX DBUG\_STATUS\_BUSY XXXXXXXXXX

-  
-  
-

ception handler triggered): misaligned store (see screenshots)  
(exception handler triggered): fault fetch (see screenshots)

ception handler triggered): illegal instruction (see screenshots)

-  
-  
-

ception handler triggered): fault load (see screenshots)

-  
-  
-

jal instruction (core 0) + illegal instruction (core 1)

-  
-

ception handler triggered): illegal instruction (see screenshots)  
(exception handler triggered): fault load (see screenshots)

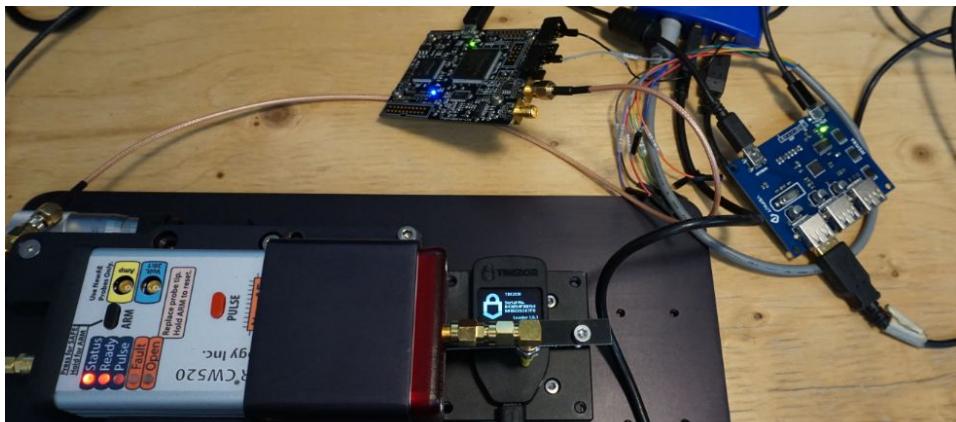
ception handler triggered): illegal instruction (see screenshots)

ception handler triggered): illegal instruction (see screenshots)  
(exception handler triggered): misaligned load (see screenshots)  
ception handler triggered): illegal instruction (see screenshots)

```
f-3.0_beta5\af30.bin  
00 00 00 00 .....  
03 00 00 00 BBBB .....  
00 00 00 00 .....  
00 00 00 00 .....  
00 00 00 00 .....  
FC AC AD C6 TΩΞΕ, W $>4 n%!  
C8 8F 6B B3 .oBT, .Q o. ™Ak  
A9 8D ED 4F H!..ΦΦCM =/H-ιΦΦ  
7D BF 5E FA F$.a.epu c█ύO)†^  
00 00 00 00 ..... xj.ç  
00 00 00 00 rj.ç .....  
00 00 00 00 z'ç ..... Éç.ç  
f-3.0_beta5\af30.bin
```

```
00 00 00 00 .....  
03 00 00 00 BBBB.....  
00 00 00 00 .....  
00 00 00 00 .....  
00 00 00 00 .....  
FC A4 AD C6 TÖDE..XW $~Φ||nñ  
C8 8F EB 93 .mBT.f..Φo.ΠΔøø  
A9 AD ED 4F H!.ΦΦCM =/H-ιΦΦ  
7D BF 5E FA Γ$.α.epu c|ΦΟΓΙ^  
00 00 00 00 ..... xj.Ç  
00 00 00 00 rj.Ç .....
```

# IMPACT OF THE PROPOSED SOLUTIONS



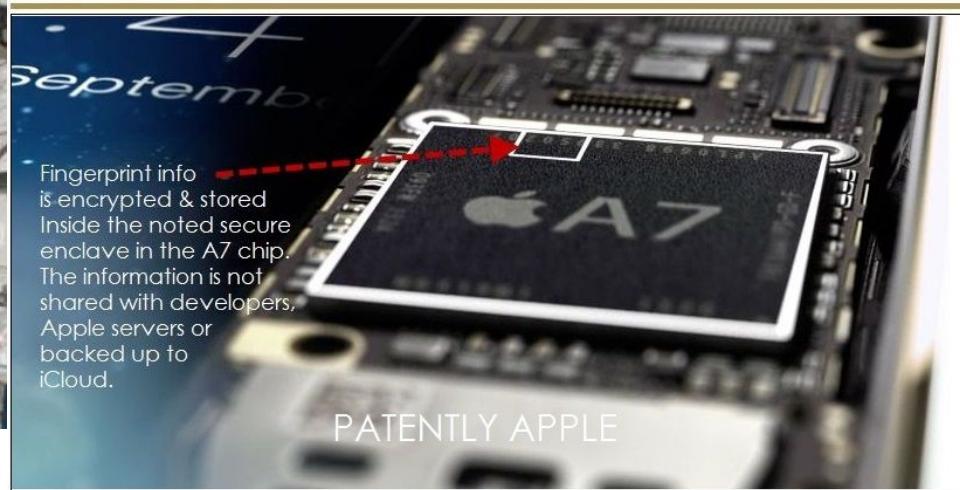
- MIN()imum Failure: EMFI Attacks against USB Stacks. Colin O'Flynn, *Dalhousie University*
- Expensive equipment requiring lots of specialization and previous research of the DUT. (i.e.: firmware reverse engineering, source code analysis, etc.)



- Low cost solution
- Enabling for less precise but much cheaper attack strategies.
- Repairable and easily accessible.
- Adequate for Devices Under Test (DUTs) and experiments in which we don't have the need to know internal microarchitecture details.

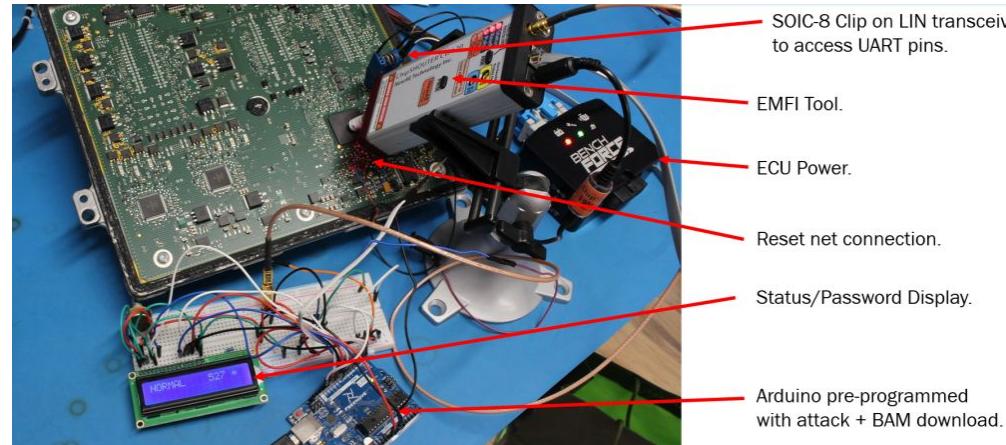
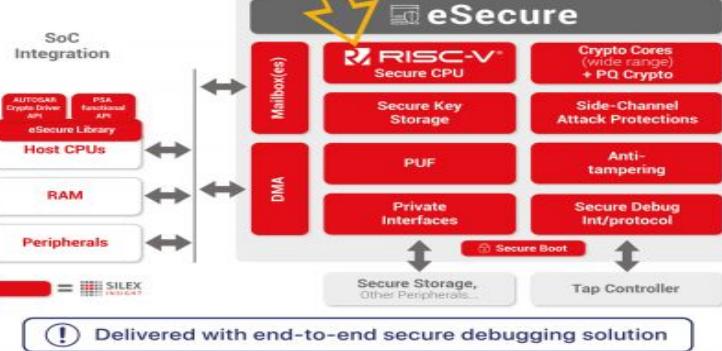


PATENT APPLICATION



## RISC-V SECURE CPU

- RISC-V 32-bit compliant CPU Core
- Configurable
- One-stop shop licensing
- Fully integrated and validated
- Software extensions available



source: <https://www.newae.com/auto>



Developed an economically affordable EMFI device suited for academia research.



Not one but two proposed designs



Evaluated the impact of EMFI attacks of several Systems-on-Chip (SoCs) that implement different versions of the RISC-V ISA. Causing exceptions on the DUTs. Reboots and crashes.



Positive results! - with room for more experimentation especially with FPGAs.



Interesting future research topics: space computers, offensive EMFI, hardware architecture improvements, fault models, etc.

# Implementación de una plataforma para tests de inyección de fallos mediante electromagnetismo contra SoCs basados en RISC-V

**THANK YOU**