



Body Bias Injection on the FLASH Memory Accelerator of a 32-bit Microcontroller

Ziling LIAO Florent BRUGUIER Philippe MAURINE

20/05/2025



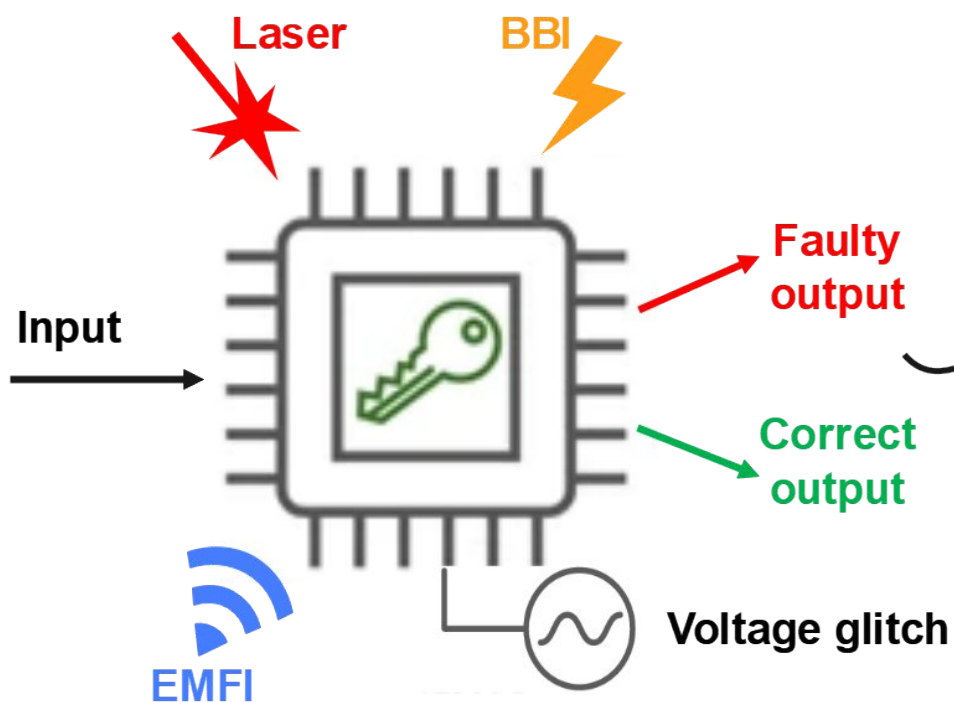
UNIVERSITÉ DE
MONTPELLIER





Background - Hardware security

❑ Fault Attack Techniques



- ❑ Force incorrect computations
- ❑ Bypass security checks
- ❑ Extract cryptographic key

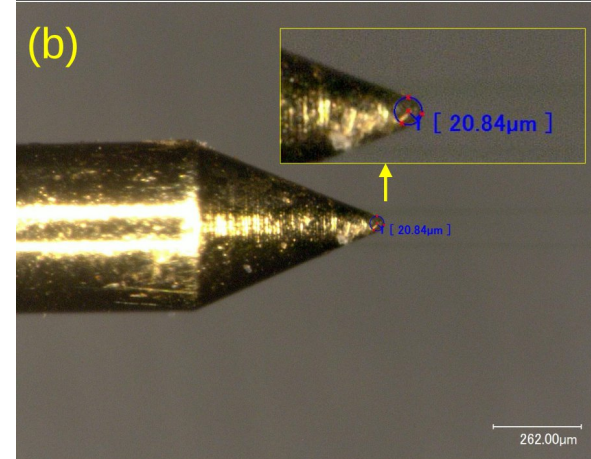
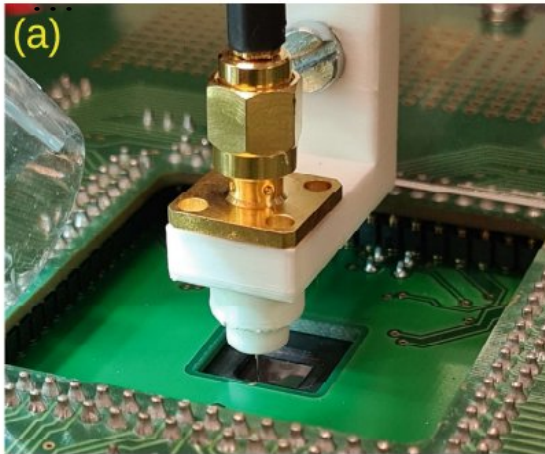


Body Bias Injection (BBI)

- ❑ **Principle: Apply voltage pulse onto the substrate of integrated circuits**

Disruptions on:

- Power supply networks
- Clock
- Control signals & data





The state of the art of Body Bias Injection

- 2011 **Yet Another Fault Injection Technique : by Forward Body Biasing Injection**
Invention de cette technique - Attaque de Bellcore sur RSA
- 2012 **Voltage spikes on the substrate to obtain timing faults**
Fault nature on digital circuits
- 2016 **Body biasing injection attacks in practice (Lumped model for dual-well substrates)**
First physical model
- 2020 **Low-cost body biasing injection (BBI) attacks on WLCSP devices**
Attaque on AES
- 2022 **Breaking a Recent SoC's Hardware AES Accelerator Using Body Biasing Injection**
Attaque on AES
- 2023 **A better practice for Body Biasing Injection**
Injection practice improvement

BBI's effects on program control flow remains largely unexplored ...



Program flow fault attacks

❑ **Program flow attack : alter the normal execution path of embedded programs**

❑ **Faults inside the processor**

Example :

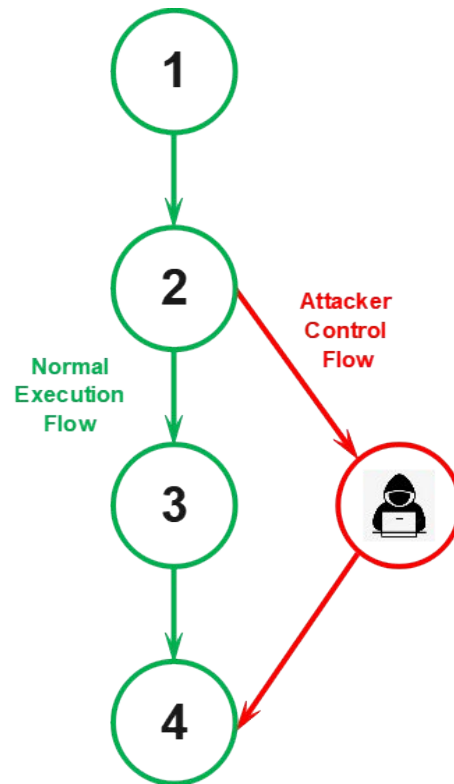
- Load attacker-controlled value into the program counter
- Instruction skipping/replay

❑ **Faults between program memory and the processor**

Example :

- Corrupt instruction cache/buffer contents
- Disrupt instruction cache/buffer updates

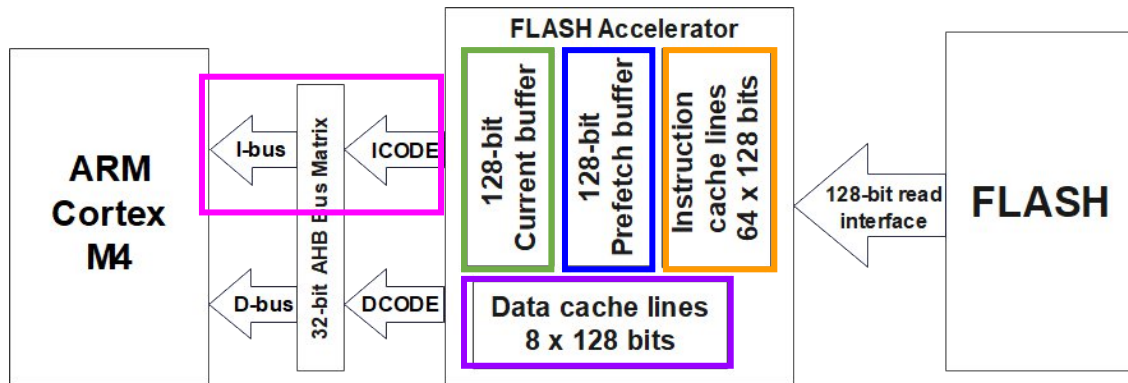
What about BBI ?





Target Architecture

- Target microcontroller : STM32F439 , ARM Cortex-M4 (3-stage pipeline)



**Hardware foundation of the program flow
=> potentially vulnerable points**

- Current buffer stores the instruction line currently requested by the CPU.
- AHB bus matrix fetches instructions (32-bits by 32 bits) from the current buffer
- Prefetch buffer stores the next consecutive instruction line.
- I-cache holds code segments that the CPU is likely to request soon based on update rules.
- D-cache holds program data.



Test Code - Sequential

Listing 1. Test Code

```
2-  ADD.W R5, R5, #0x0  //I0
3-  ...

32- ADD.W R5, R5, #0x17 //IA
33- ADD.W R2, R2, #0x27 //IB
34- ADD.W R3, R3, #0xa  //IC
35- ADD.W R3, R3, #0x0  //I0
...

65- ADD.W R3, R3, #0x0  //I0
```

Instructions I0

Final values in R5,
R2, R3 serve as error
indicators

Instructions I0

Listing 2. Memory Alignment

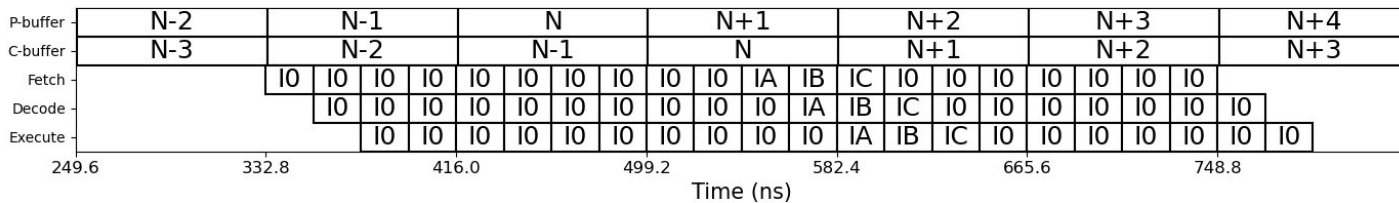
```
Line N-8 -  Ix Ix Ix Itrig
Line N-7 -  I0 I0 I0 I0
...
Line N-1 -  I0 I0 I0 I0
Line N    -  I0 I0 IA IB
Line N+1 -  IC I0 I0 I0
Line N+2 -  I0 I0 I0 I0
...
Line N+7 -  I0 I0 I0 I0
Line N+8 -  I0 I0 I0 Ix
```

- ❑ Each ADD instruction is 32-bit => 4 instructions form a 128-bit line.
- ❑ Place IA to IC within different instruction lines => Determine whether the attack affects a single instruction or the entire line

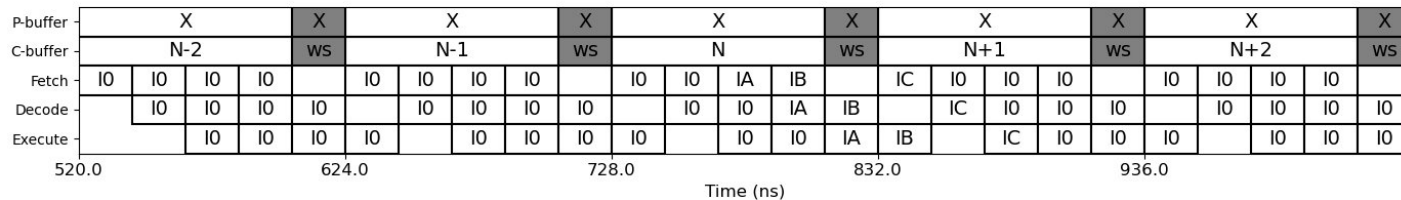


Timing Model

❑ Prefetch enabled



❑ Prefetch disabled





Experiments

- ❑ **4 different FLASH accelerator configurations**
 - ❑ Prefetch buffer enabled + instruction cache enabled
 - ❑ Prefetch buffer enabled + instruction cache disabled
 - ❑ Prefetch buffer disabled + instruction cache enabled
 - ❑ Prefetch buffer disabled + instruction cache disabled
- ❑ **Approach**
 1. Quick spatial scan => identify sensitive positions
 2. Time-sweeping attack at a fix position => establish time distribution patterns of fault



Results - Prefetch enabled

Instruction cache lines enabled



Line N-8	-	Ix	Ix	Ix	Itrig
Line N-7	-	I0	I0	I0	I0
...					
Line N-1	-	I0	I0	I0	I0
Line N	-	I0	I0	IA	IB
Line N+1	-	IC	I0	I0	I0
Line N+2	-	I0	I0	I0	I0
...					
Line N+7	-	I0	I0	I0	I0
Line N+8	-	I0	I0	I0	Ix

Fault Classification

- ☐ **Skipping**: The faulty register contains 0.
- ☐ **Replay**: The faulty register contains a value equal to 2 times the correct value.

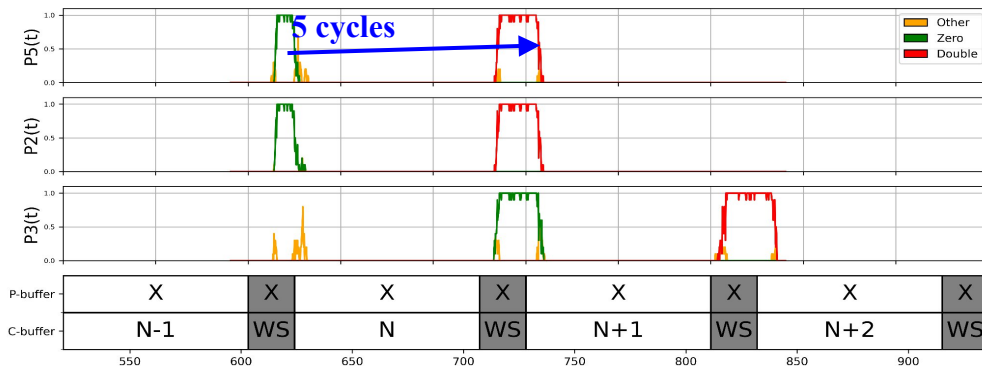
Hypotheses :

- ☐ Faults may result from prefetch buffer update failures
=> Instruction replay or skip



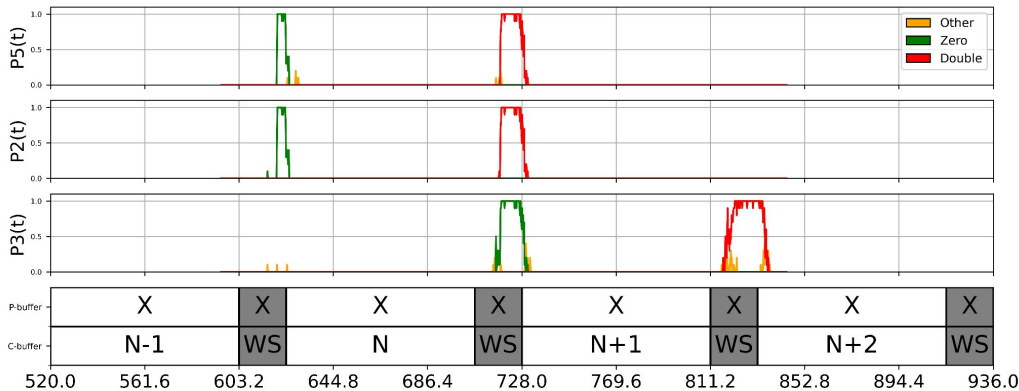
Results - Prefetch disabled

Instruction cache lines enabled



Current buffer update failure can explain the experimental results.

Instruction cache lines disabled



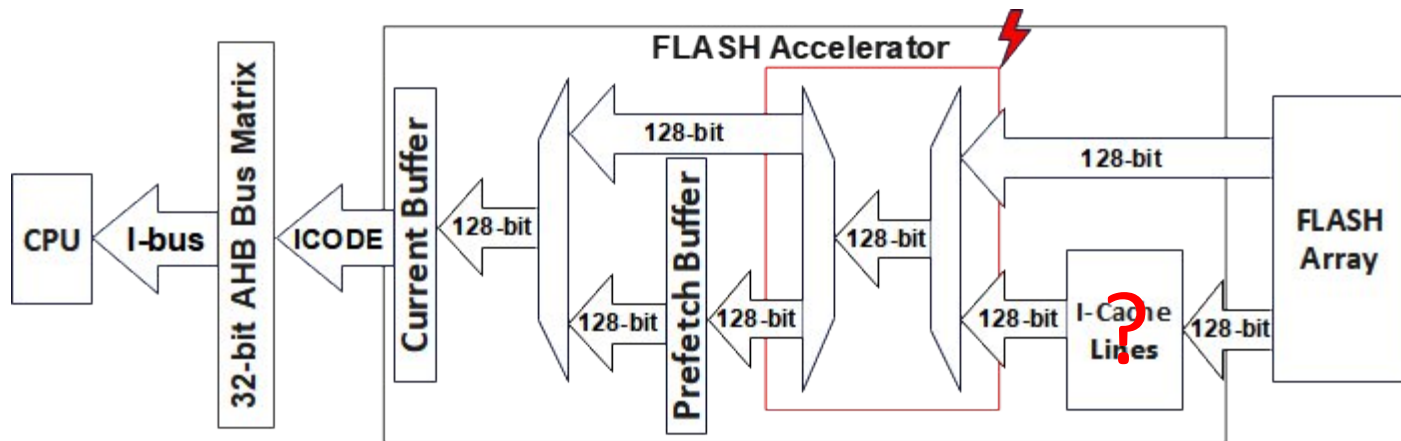


Fault Model Conclusion

- ❑ When the **P-buffer is enabled** (regardless of whether the I cache is enabled or not), the most **vulnerable** program flow stage is the update of **prefetch buffer**.
- ❑ When the **P-buffer is disabled** (regardless of whether the I-cache is enabled or not), the most **vulnerable** program flow stage is the update of **current buffer**.



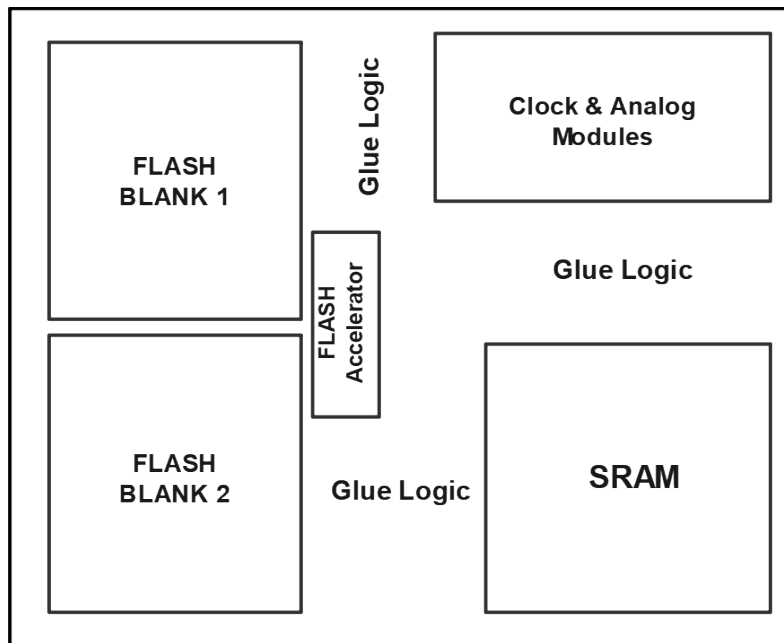
Body Bias Injection affects a common part of the IC that delivers instructions from the FLASH or I-cache lines to the prefetch and current buffer.





New Question

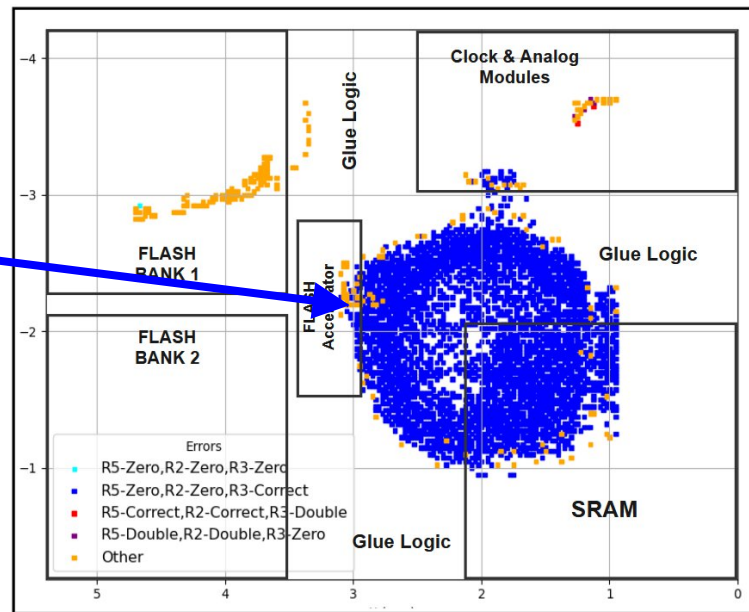
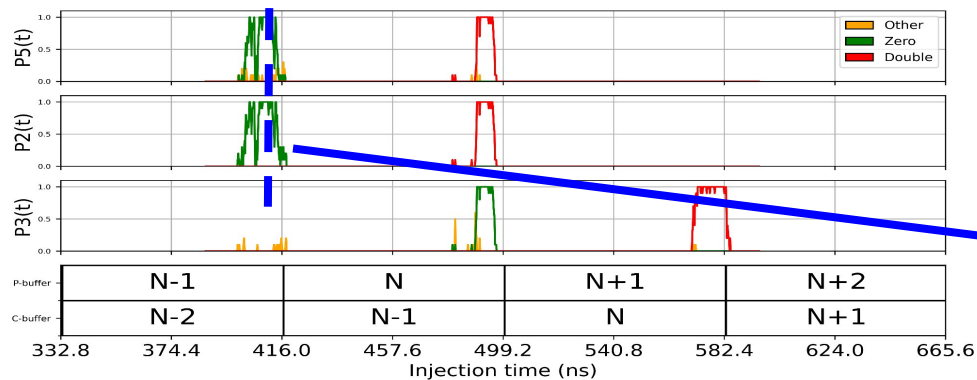
Is this behavior spatially general or spatially specific?





Spatial genericity of obtained results

- ❑ The fault is easy to generate in a large region on the chip





Conclusion

- ❑ Body Bias Injection can disrupt instruction buffer updates in the FLASH accelerator of an ARM-based MCU, which cause instruction line repetition or skipping.
=> Potentially be used to skip security checks / critical steps in cryptographic algorithms.
- ❑ In existing literature, other fault attack techniques (EMFI/LFI) can also induce similar phenomenon on the same architecture / on other architectures.
=> Common fault model behind ?
- ❑ Existing results are limited to the case of sequential code.
=> Extend to non-sequential code (branch instructions / for loop) ?