FDTC 2025
Fault Diagnosis and Tolerance in Cryptography

Fault Analysis through Body Bias Injection on the FLASH Memory Accelerator of a Microcontroller

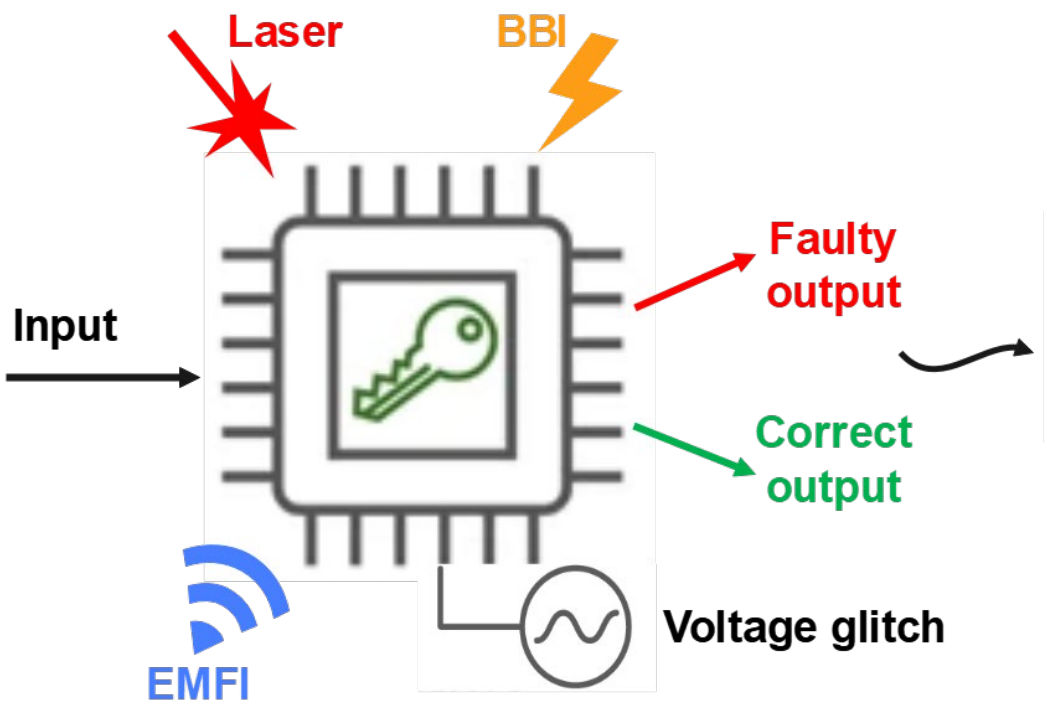
Ziling LIAO Florent BRUGUIER Philippe MAURINE

14/09/2025

LIRMM
UNIVERSITÉ DE MONTPELLIER
cnrs

Background - Hardware security

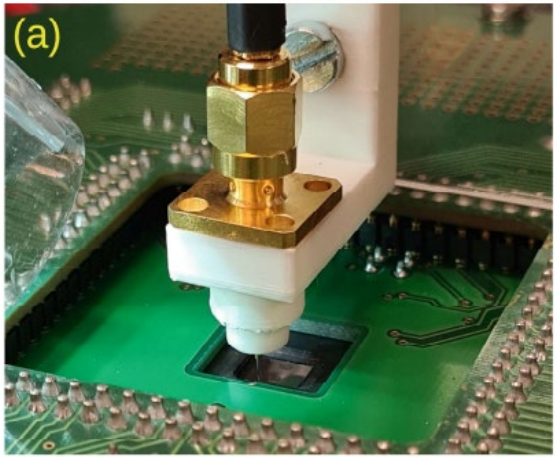
- Fault Attack Techniques

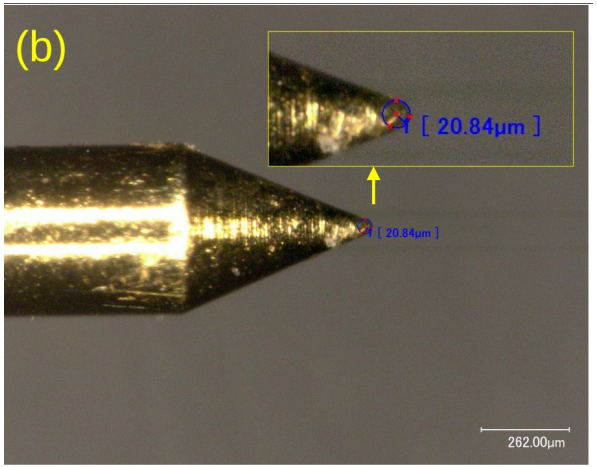


- Force incorrect computations
- Bypass security checks
- Extract cryptographic key

Body Bias Injection (BBI)

- **Principle: Voltage pulse injection on IC substrate**
- **Potential Disruptions :**
  - Power supply networks
  - Clock signals
  - Control & data signals
  - …





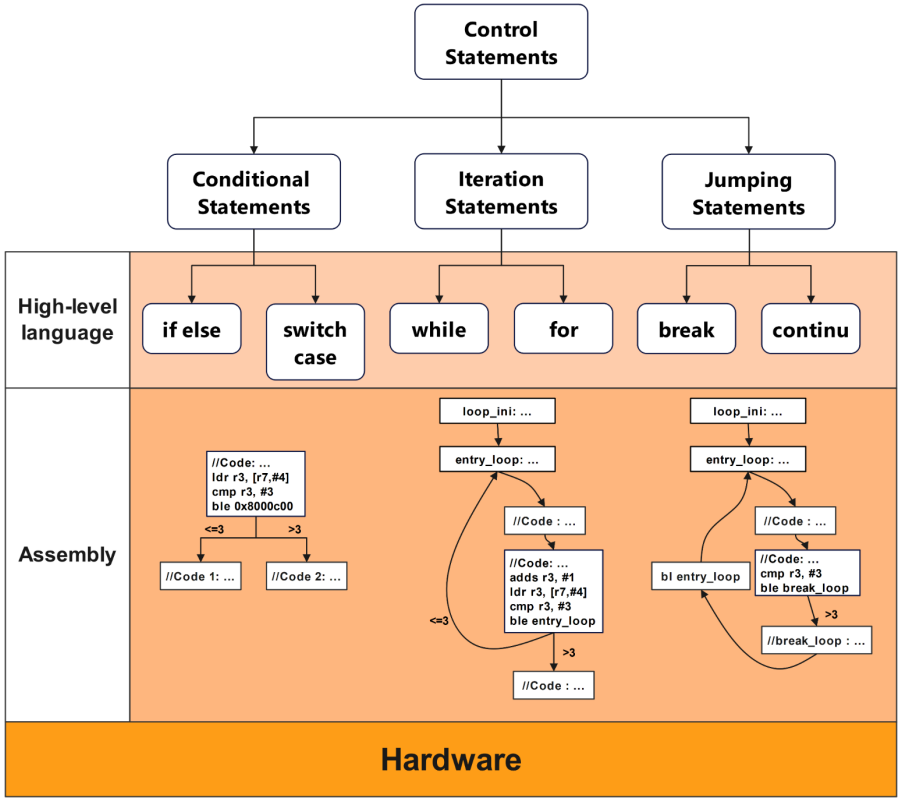
The state of the art of Body Bias Injection

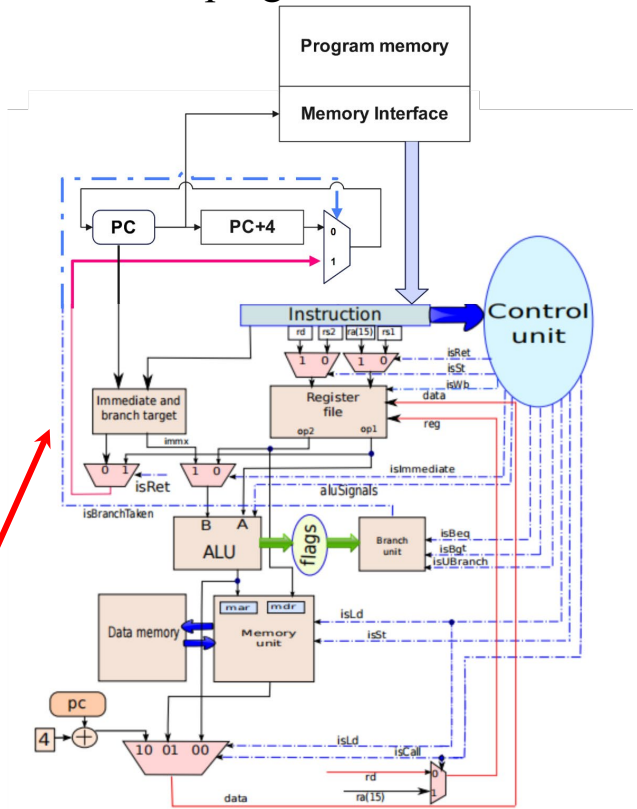
2011	**Yet Another Fault Injection Technique : by Forward Body Biasing Injection** **Invention de cette technique - Attaque de Bellcore sur RSA**
2012	**Voltage spikes on the substrate to obtain timing faults** **Fault nature on digital circuits**
2016	**Body biasing injection attacks in practice (Lumped 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 practive improvement**

My Objective: To establish a model that characterizes the impact of BBI on program control flow.

Program Control Flow

- Order in which statements and instructions are executed in a program.





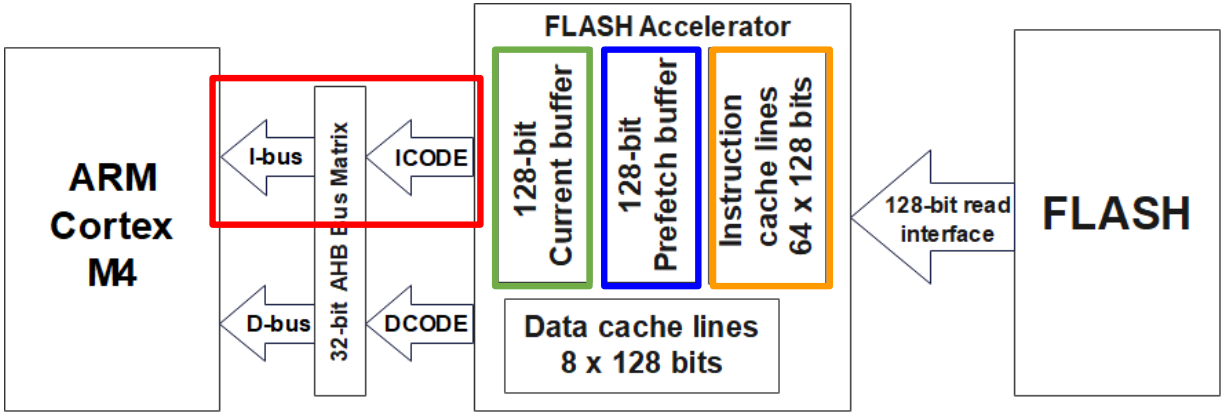
Program control flow attack

- **Program control flow attack : Alters the normal execution path of a program**
  - **Faults Inside the Processor**
    Examples:
    - Loading attacker-controlled values into the Program Counter (PC)
    - Instruction skipping / replay
  - **Faults between Program Memory and the Processor**
    Examples :
    - Corrupt instruction cache/buffer contents
    - Disrupt instruction cache/buffer updates

Can BBI also generate exploitable faults in program control flow ?

Target architecture

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



- **Current Buffer: Stores the instruction line currently requested by the CPU**
- **AHB Bus Matrix: Fetches instructions (32 bits at a time) from the Current Buffer**
- **Prefetch Buffer: Holds the next consecutive instruction line**
- **I-Cache: Stores likely branch destination lines based on update rules**

Study in 2 Phases

❑ Phase 1 : Sequential Test Code

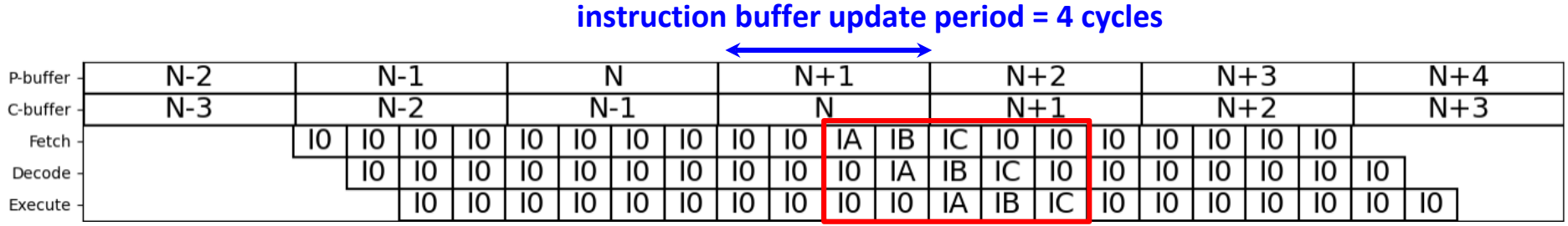
- How BBI affects the normal operation of the **CPU pipeline** and **FLASH accelerator** when there's no branch involved?
- Which stage of the sequential execution flow is most likely to be affected by BBI ?

❑ Phase 2 : Single Branch Test Code

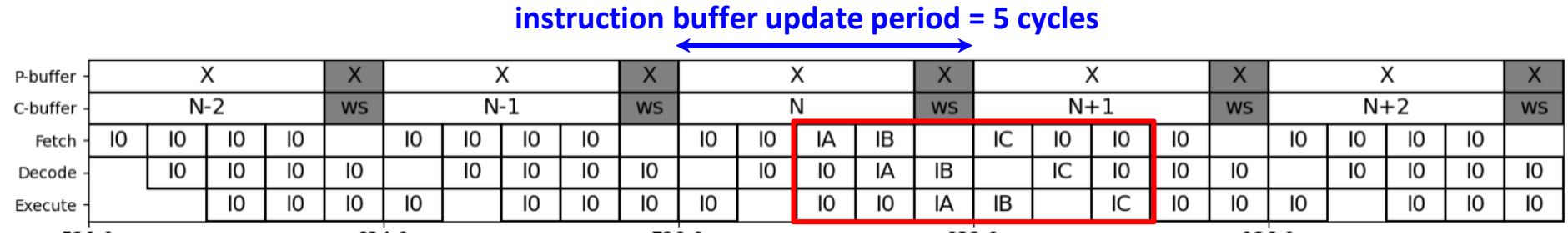
- What kind of phenomenon can be generated when there is a branch operation involved ?

Sequential Execution

- Prefetch enabled

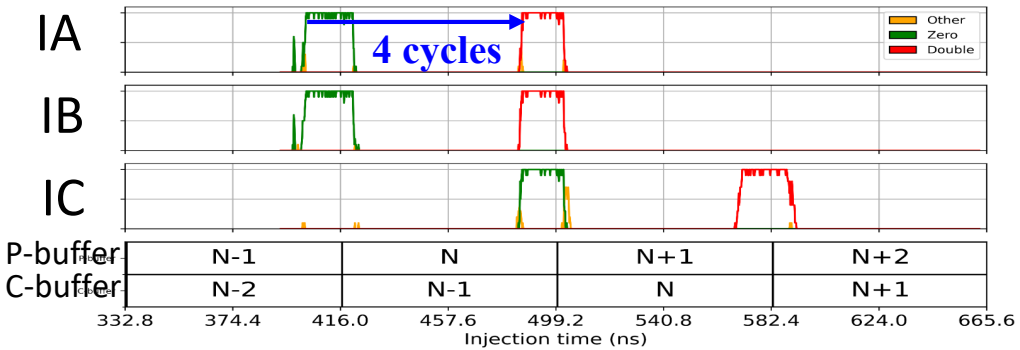


- Prefetch disabled



Sequential Execution

- Prefetch buffer enabled

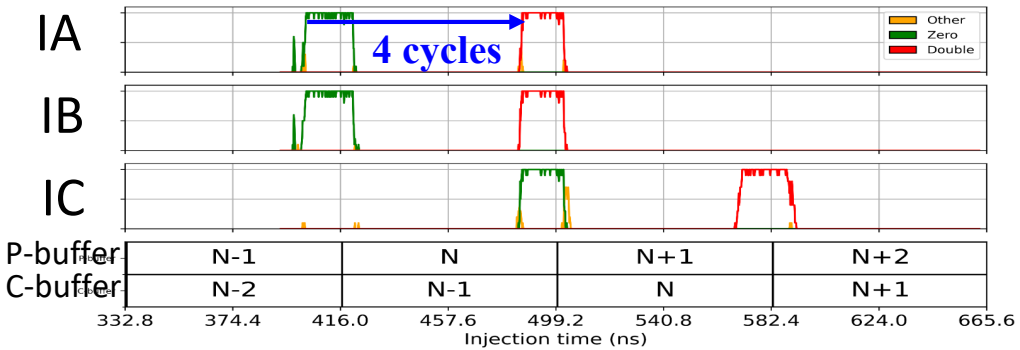


Fault Classification

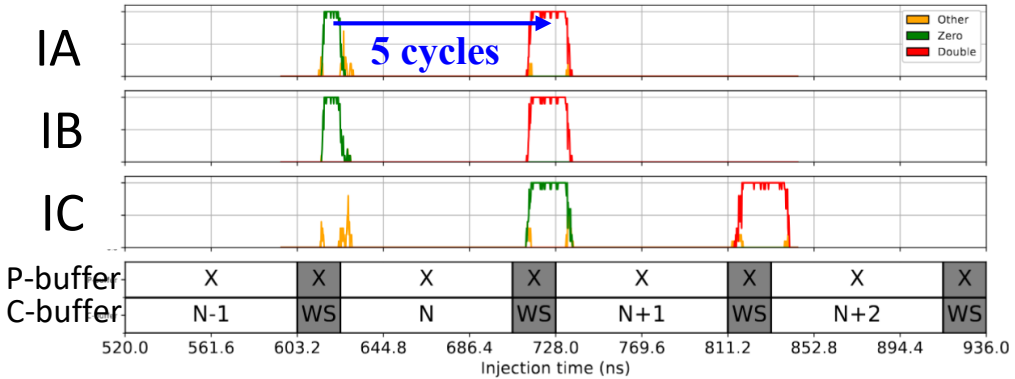
- **Skipping**: The instruction is skipped.
- **Replay**: The instruction is replayed

Sequential Execution

- Prefetch buffer enabled



- Prefetch buffer disabled



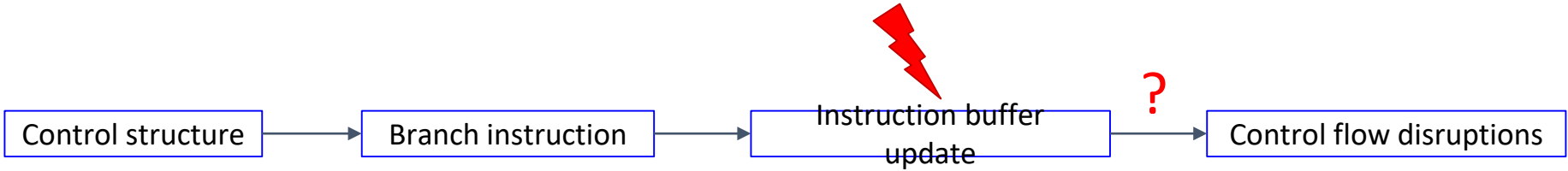
Fault Classification

- **Skipping**: The instruction is skipped.
- **Replay**: The instruction is replayed

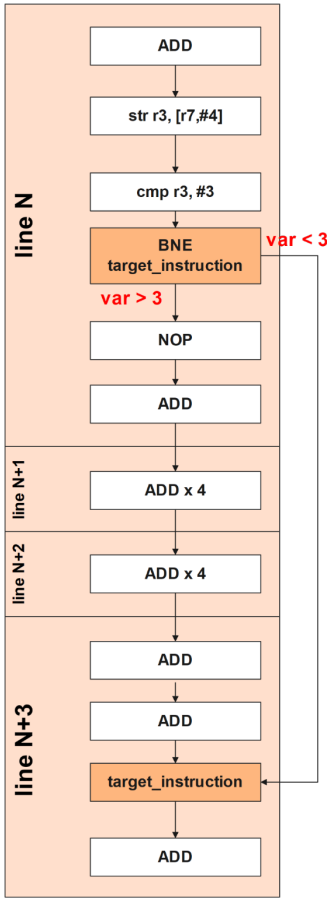
Sequential Execution

- **Conclusion for the first phase:** In sequential code execution, BBI-induced faults are observed, primarily attributable to update failures in the instruction buffers of the FLASH accelerator.
  - With the prefetch buffer enabled, the update of the prefetch buffer represents the most vulnerable stage, whereas the current buffer is seldom impacted.
  - Conversely, when the prefetch buffer is disabled, the update of the current buffer emerges as the most vulnerable stage of the program flow.

- **Question :** Can the same phenomenon happen to **non-sequential control structure** execution ?



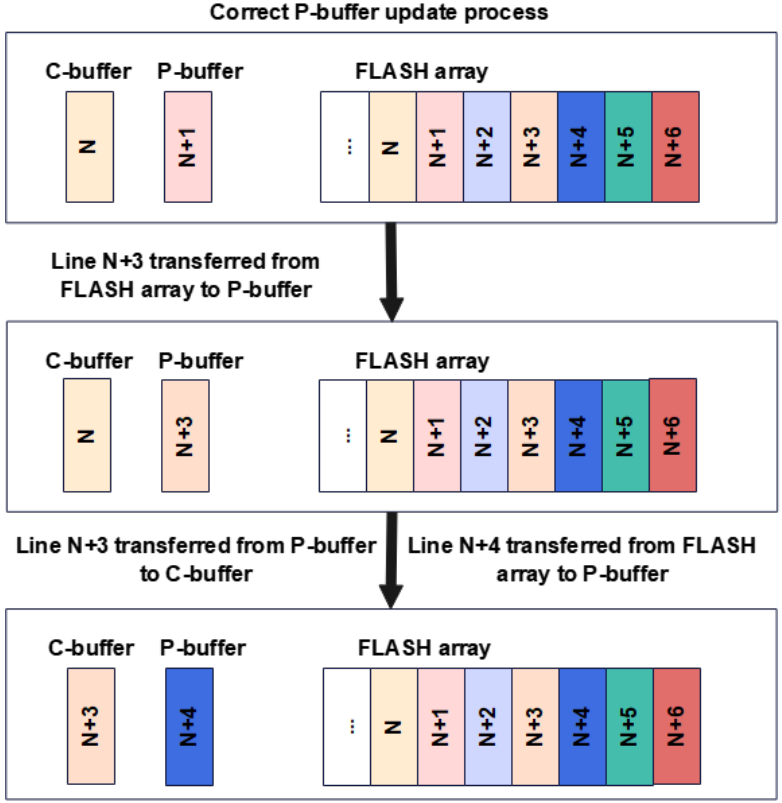
Single Branch Control Structure



- Target Conditional Branch Operation
  - The branch under consideration transfers control from line N to line N+3.
  - The fault injection is performed during the execution of this branch operation.

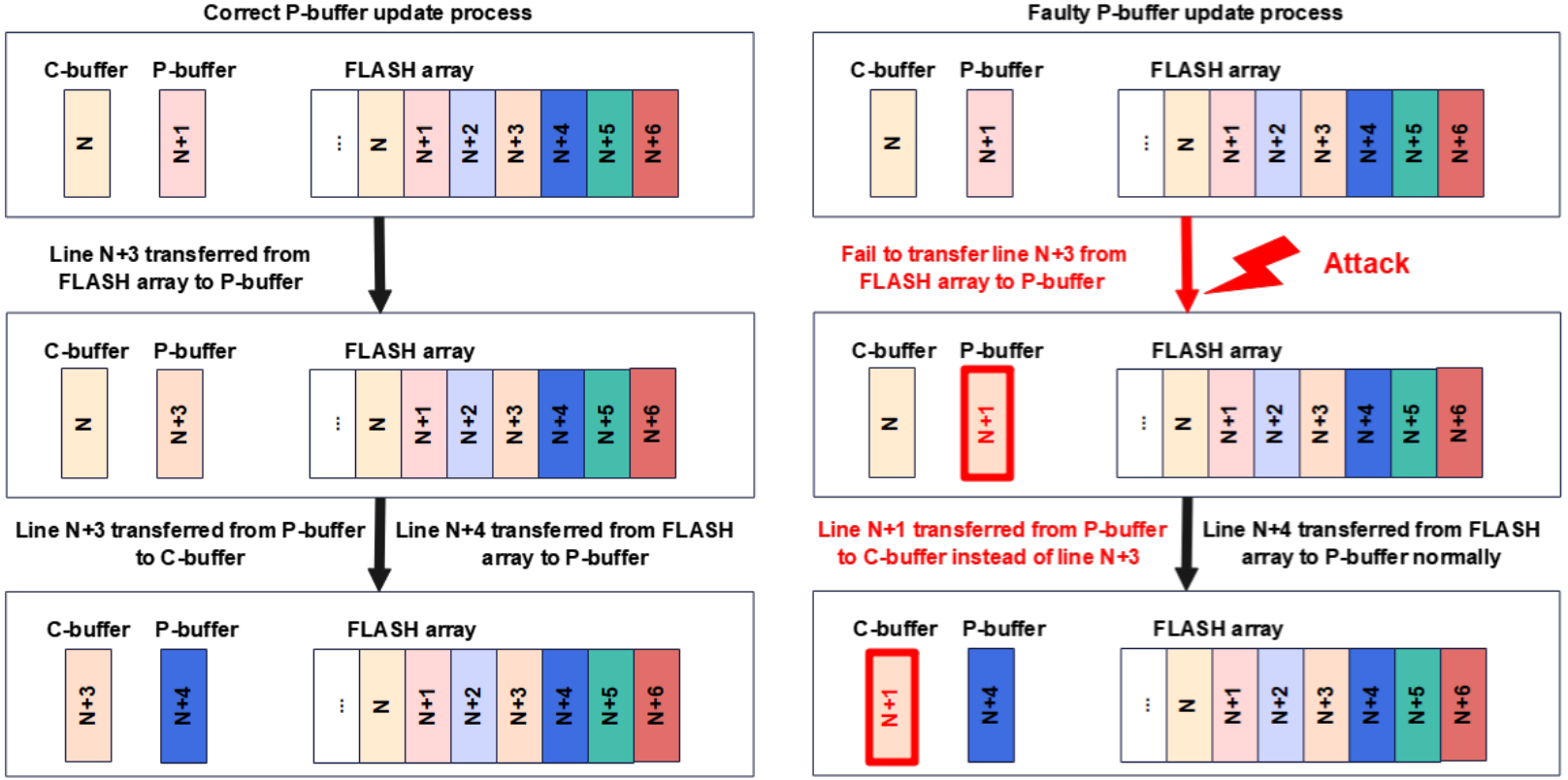
Single Branch Control Structure

- Prefetch buffer enabled



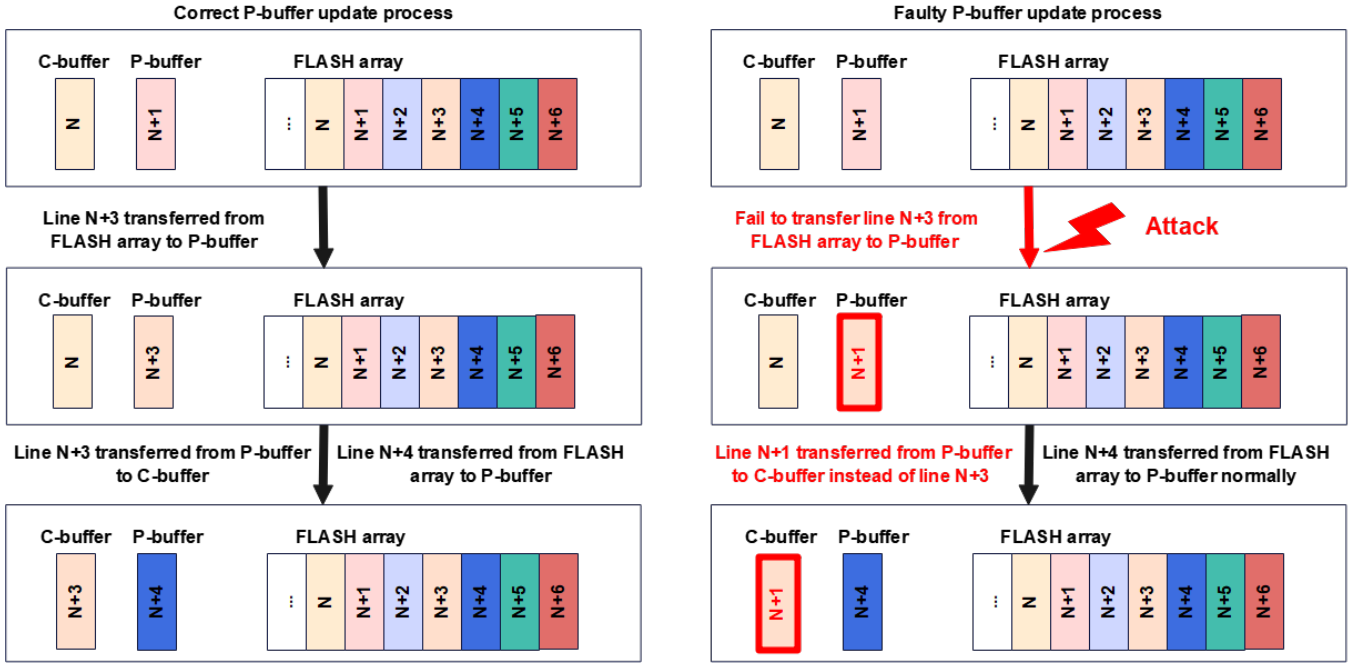
Single Branch Control Structure

- Prefetch buffer enabled



Single Branch Control Structure

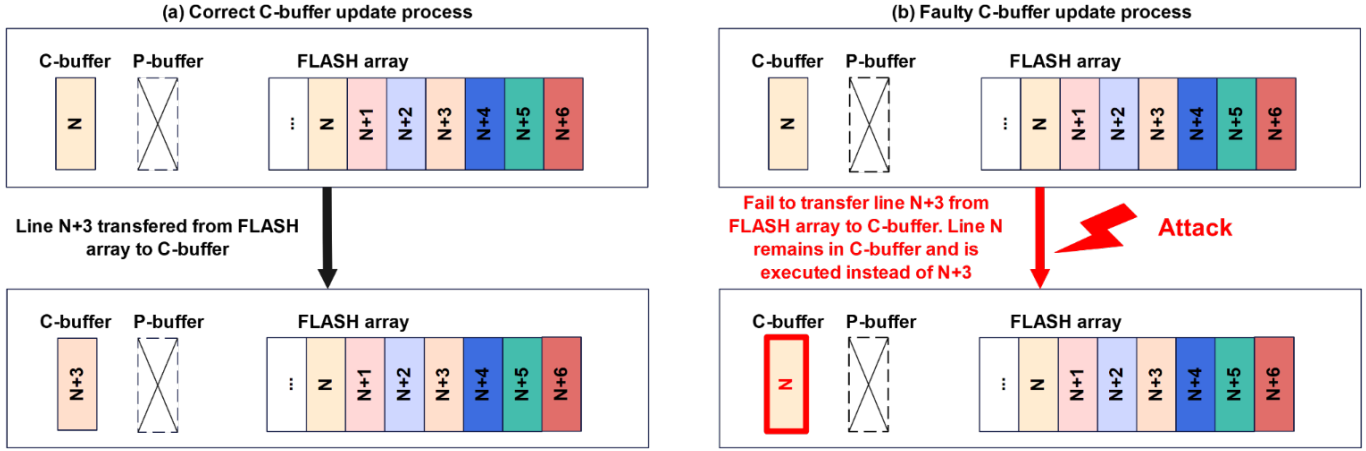
- Prefetch buffer enabled



Line N+1 is executed illegally instead of N+3, so N+3 is skipped. Both N+1 and N+3 contain arithmetic instructions; therefore, in this scenario, the primary impact of the fault is the generation of incorrect arithmetic results.

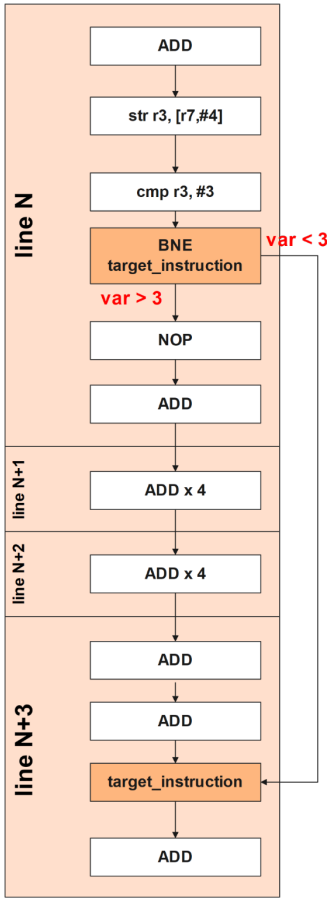
Single Branch Control Structure

- Prefetch buffer disabled



Line N is replayed, line N+3 is skipped.

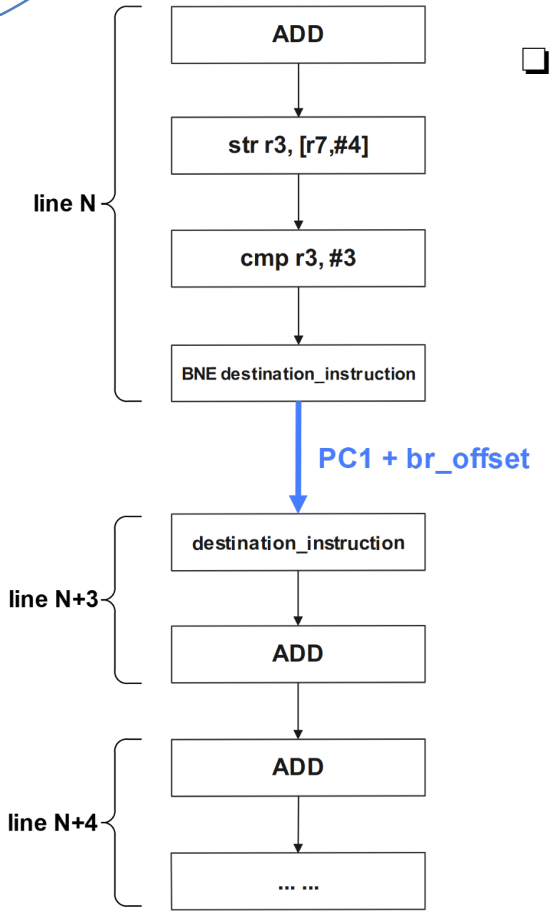
Single Branch Control Structure



- **Question:** What happens when a branch instruction is replayed illegally ?

  Line N includes not only arithmetic instructions but also the branch instruction that has juste been executed, which means the branch will also be replayed, prompting the question of how the system behaves in this case.

Single Branch Control Structure

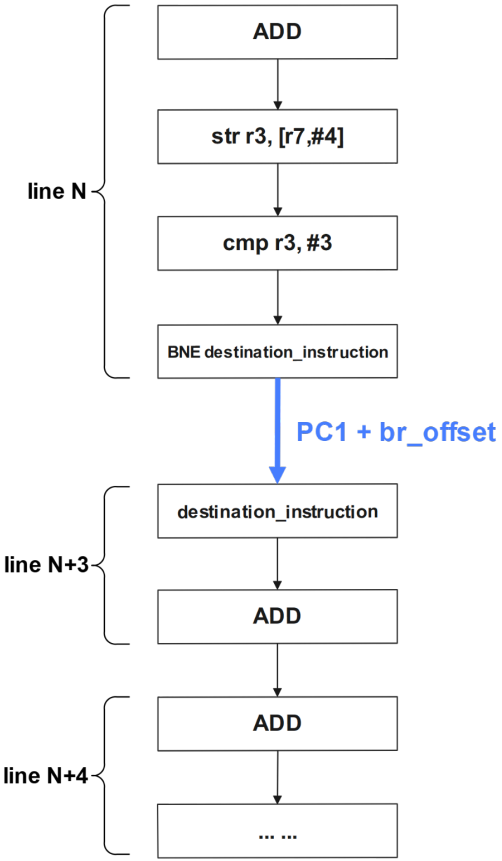


- **Branch instruction**
  - Add a fixed offset (br_offset) to the current Program Counter (PC).
  - "br_offset" is coded in the instruction "BNE destination_instruction".
  - The CPU only see br_offset but not the absolute destination address. The destination address is calculated by :

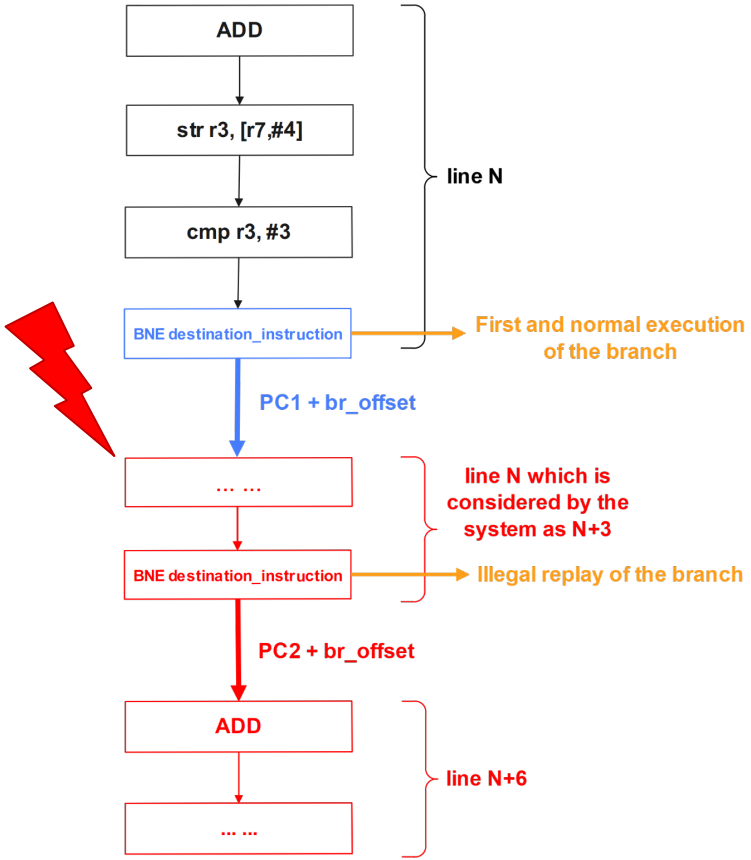
    Destination address = PC1 (current PC) + br_offset

Single Branch Control Structure

❑ Normal Execution



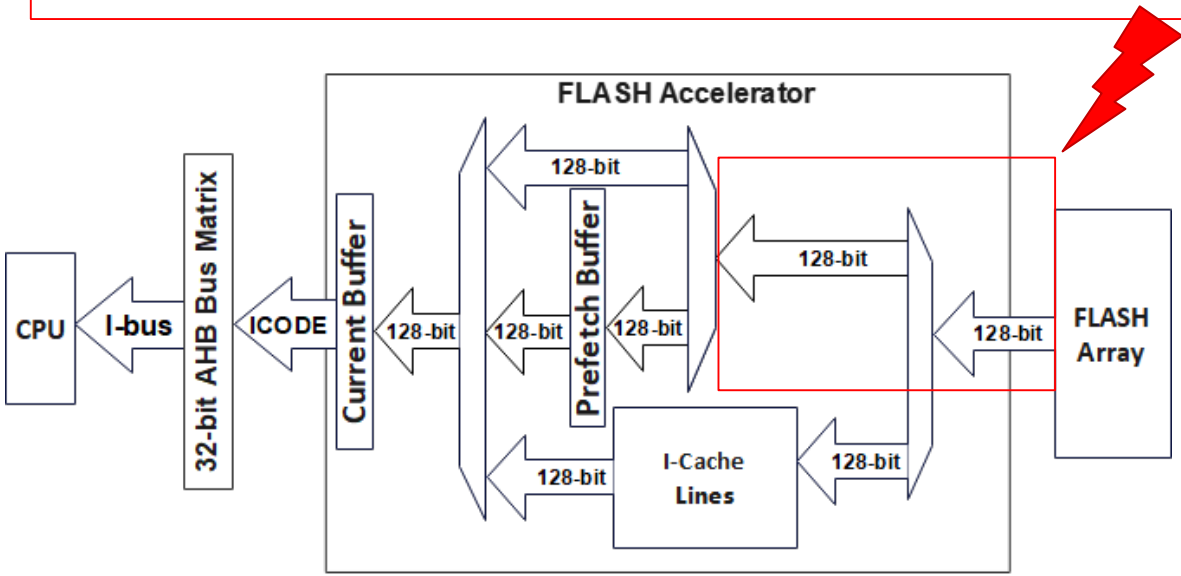
❑ Disrupted Execution



Fault Model Conclusion

- **Conclusion for the first phase (sequential test code) :**
  During the execution of sequential code, BBI can induce control flow faults, primarily caused by update failures in the instruction buffers within the FLASH accelerator.

Body Bias Injection affects a common path that delivers instructions from the FLASH to the instruction buffers.

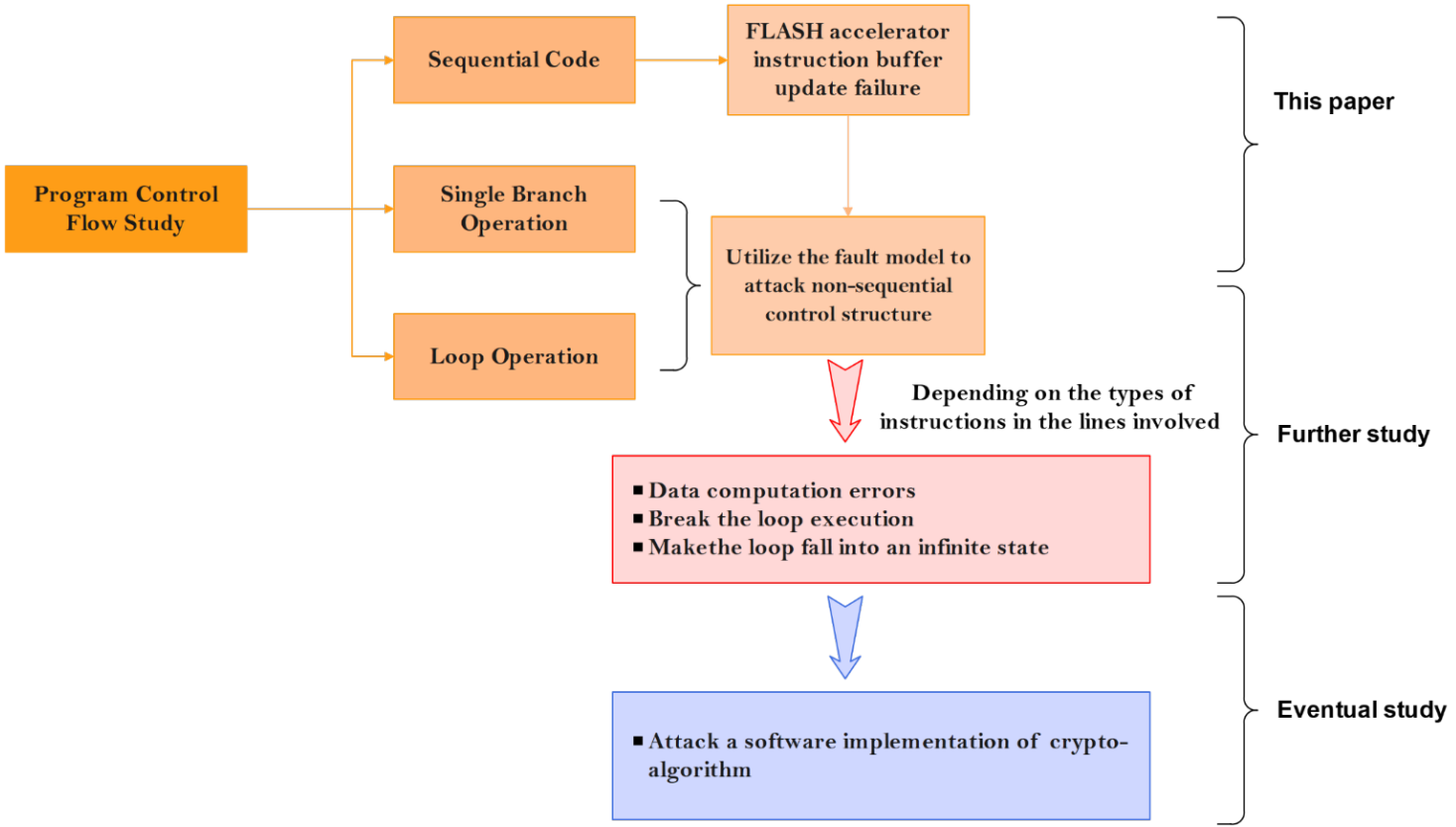


Fault Model Conclusion

- **Conclusion for the first phase (sequential test code) :**
  During the execution of sequential code, BBI can induce control flow faults, primarily caused by update failures in the instruction buffers within the FLASH accelerator.
  - When the **prefetch buffer is enabled**, the most **vulnerable** stage is the update of the **prefetch buffer**, the current buffer is rarely affected.
  - When the **prefetch buffer is disabled**, the most **vulnerable** stage is the update of the **current buffer**.

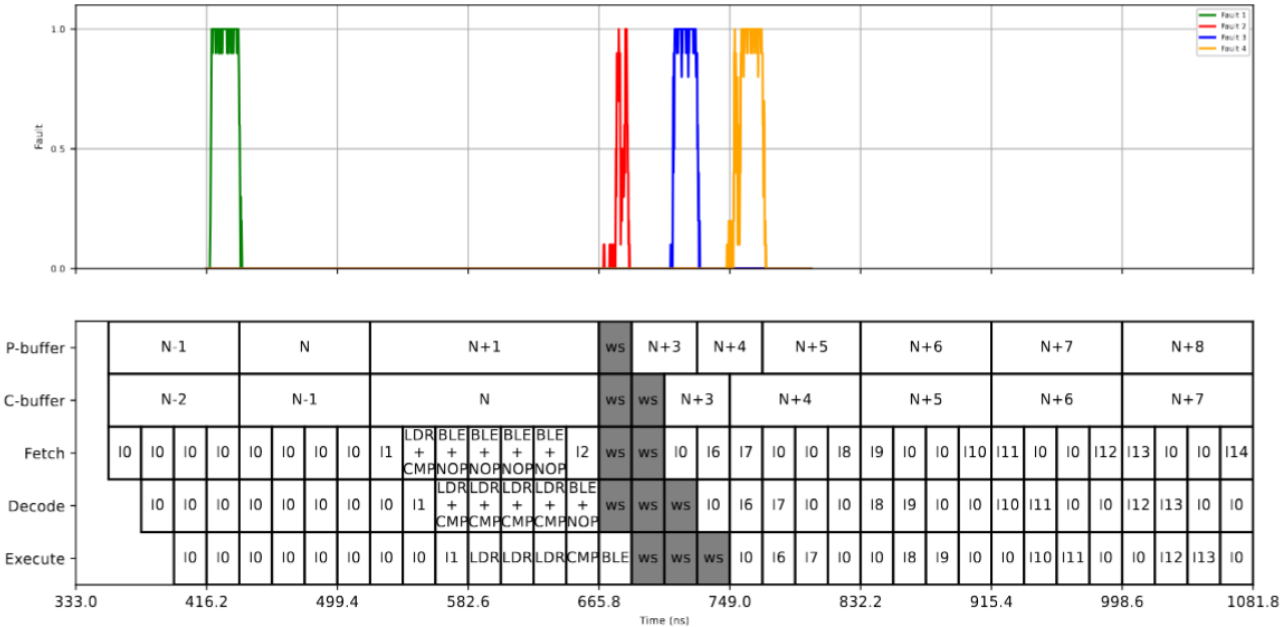
- **Conclusion for the second phase (test code with branch) :**
  - **Root cause of faults**: Failure to update instruction buffer
  - Impact depends on buffer content before/after update:
    Arithmetic instructions → Arithmetic errors
    Branch instructions → Illegal branch / Control flow deviation

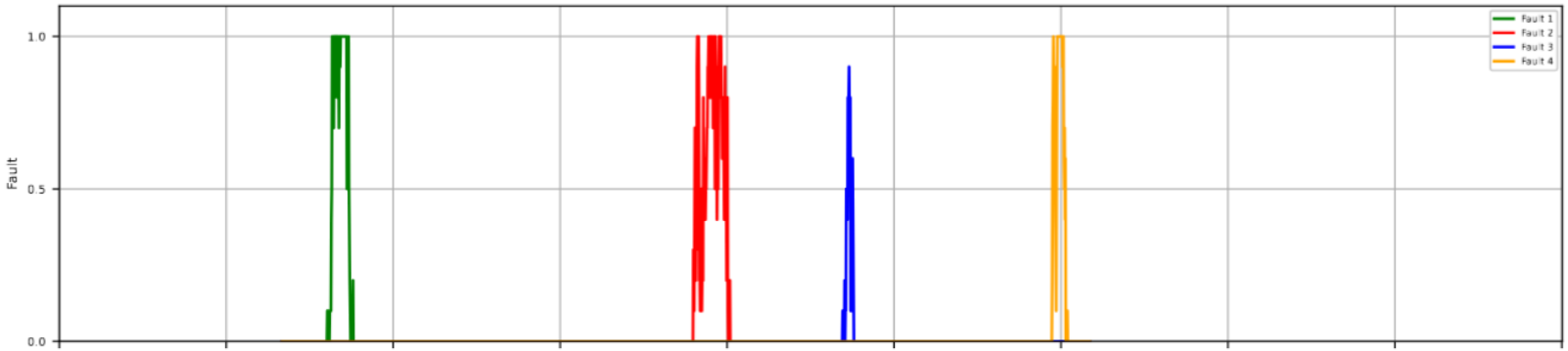
Conclusion

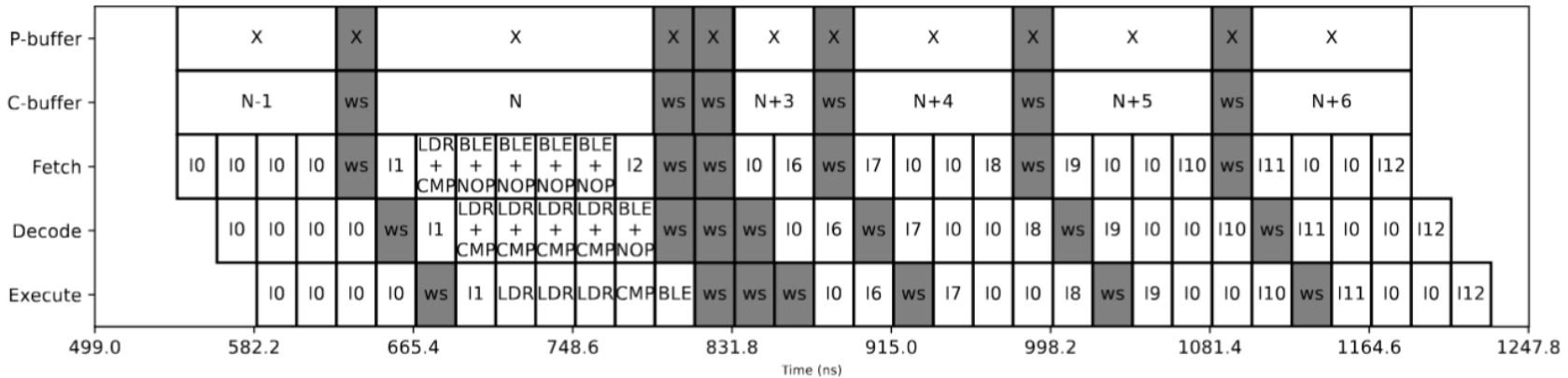

Program Control Flow Study
Sequential Code
Single Branch Operation
Loop Operation
FLASH accelerator instruction buffer update failure
Utilize the fault model to attack non-sequential control structure
Depending on the types of instructions in the lines involved
Data computation errors
Break the loop execution
Makethe loop fall into an infinite state
Attack a software implementation of crypto-algorithm
This paper
Further study
Eventual study

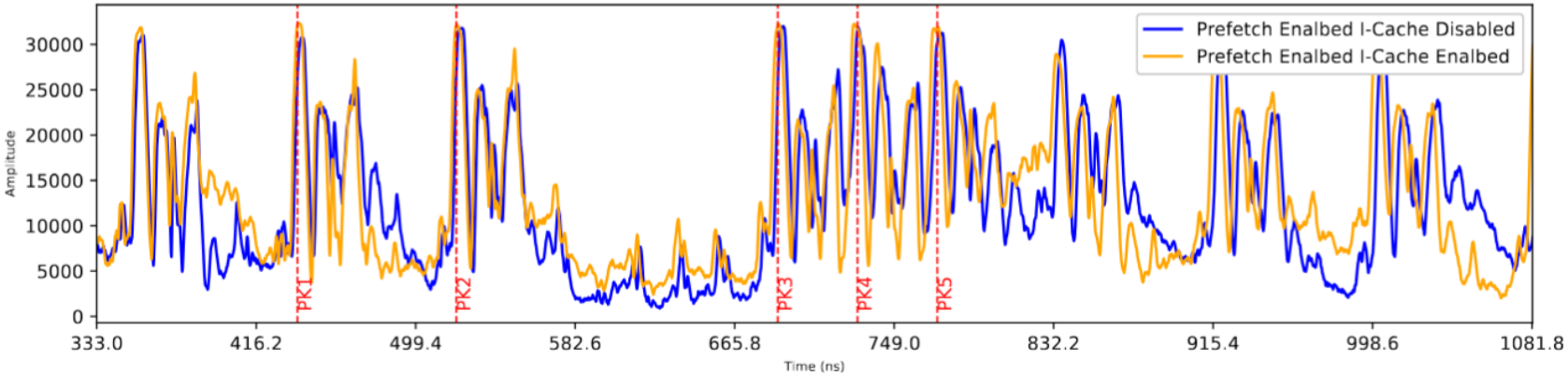

Thank you!

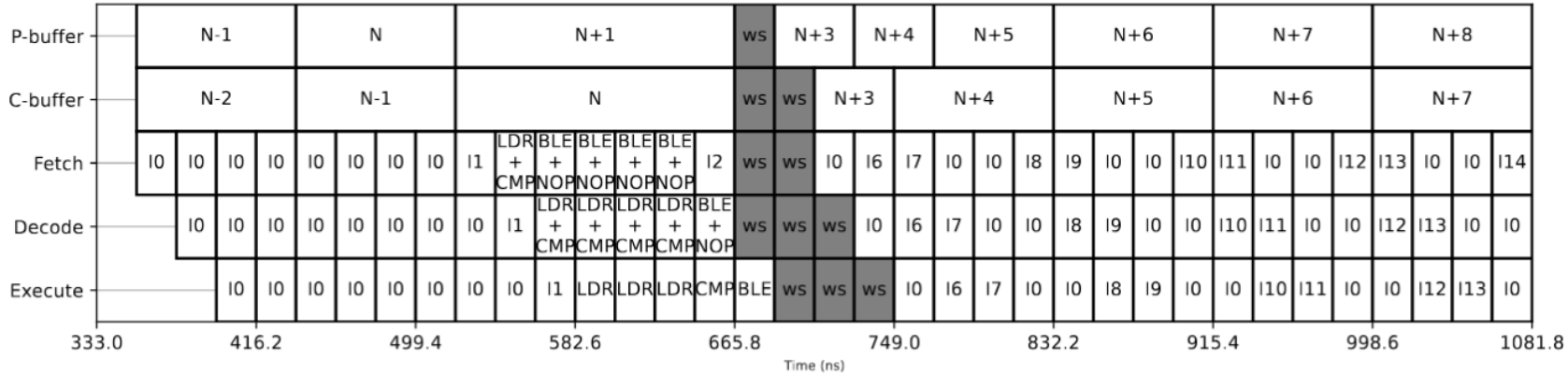
Annex

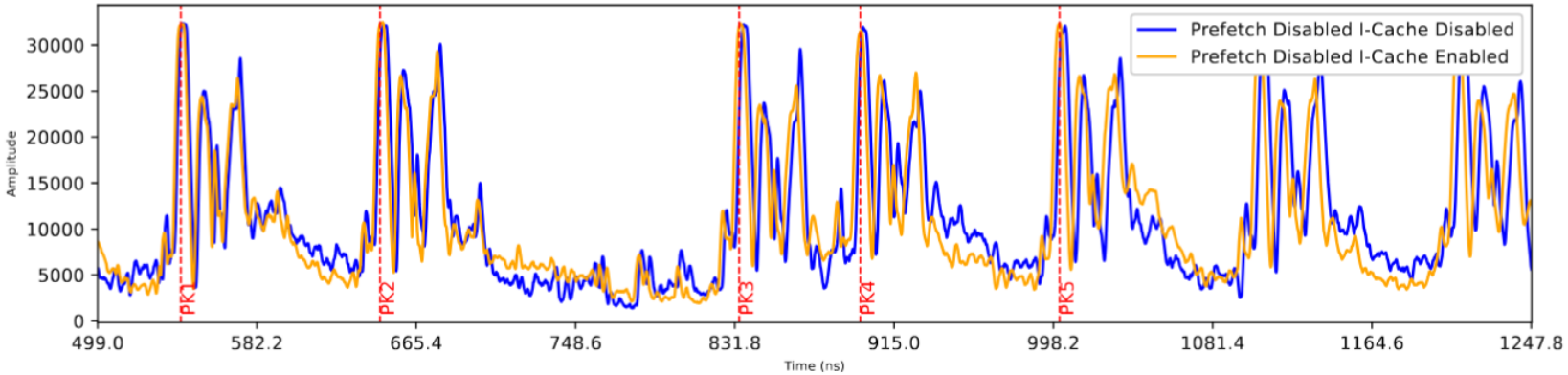

Fault 1
Fault 2
Fault 3
Fault 4
Fault
1.0
0.5
0.0
P-buffer
C-buffer
Fetch
Decode
Execute
N-1
N
N+1
ws
N+3
N+4
N+5
N+6
N+7
N+8
N-2
LDR + CMP
BLE + NOP
I0
I1
I2
I6
I7
I8
I9
I10
I11
I12
I13
I14
333.0
416.2
499.4
582.6
665.8
749.0
832.2
915.4
998.6
1081.8
Time (ns)

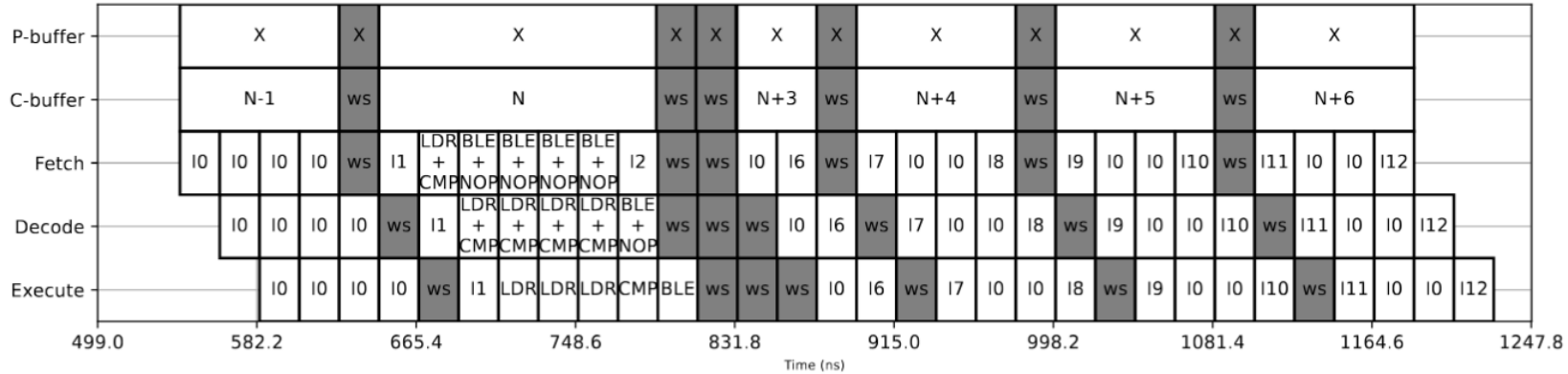

Fault 1
Fault 2
Fault 3
Fault 4
Fault
1.0
0.5
0.0


P-buffer
C-buffer
Fetch
Decode
Execute
499.0
582.2
665.4
748.6
831.8
915.0
998.2
1081.4
1164.6
1247.8
Time (ns)


Prefetch Enalbed I-Cache Disabled
Prefetch Enalbed I-Cache Enalbed
Amplitude
30000
25000
20000
15000
10000
5000
0
PK1
PK2
PK3
PK4
PK5
333.0
416.2
499.4
582.6
665.8
749.0
832.2
915.4
998.6
1081.8
Time (ns)


P-buffer
C-buffer
Fetch
Decode
Execute
N-1
N
N+1
ws
N+3
N+4
N+5
N+6
N+7
N+8
N-2
I0
I1
I2
I6
I7
I8
I9
I10
I11
I12
I13
I14
LDR + CMP
BLE + NOP
BLE + NOP
BLE + NOP
BLE + NOP
LDR + CMP
LDR + CMP
LDR + CMP
LDR + CMP
BLE + NOP
LDR
BLE
CMP
333.0
416.2
499.4
582.6
665.8
749.0
832.2
915.4
998.6
1081.8
Time (ns)


Prefetch Disabled I-Cache Disabled
Prefetch Disabled I-Cache Enabled
Amplitude
30000
25000
20000
15000
10000
5000
0
PR1
PR2
PR3
PR4
PR5
499.0
582.2
665.4
748.6
831.8
915.0
998.2
1081.4
1164.6
1247.8
Time (ns)


P-buffer
C-buffer
Fetch
Decode
Execute
X
N-1
N
N+3
N+4
N+5
N+6
ws
I0
I1
I2
I6
I7
I8
I9
I10
I11
I12
LDR + CMP
BLE + NOP
LDR + CMP
BLE
CMP
499.0
582.2
665.4
748.6
831.8
915.0
998.2
1081.4
1164.6
1247.8
Time (ns)

Spatial genericity of obtained results

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