Laser Fault Injection in a 32-bit Microcontroller: from the Flash Interface to the Execution Pipeline

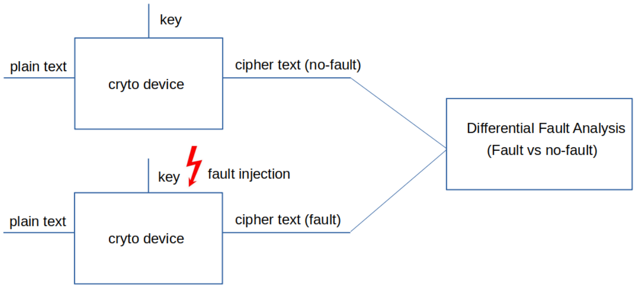
Vanthanh Khuat, Jean-Luc Danger, Jean-Max Dutertre

September 2021

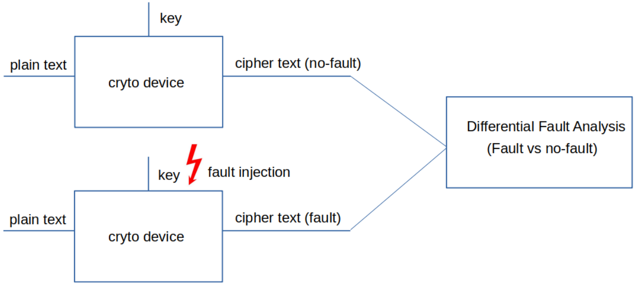
Table contents

Introduction

Fault injection

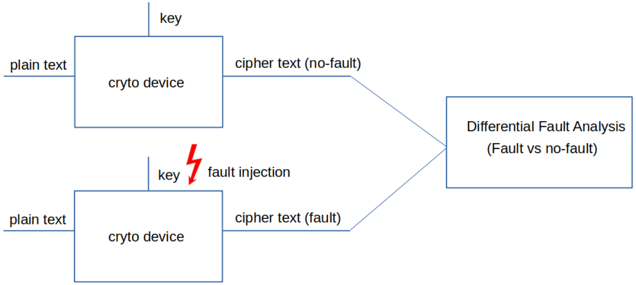


Introduction
Fault injection



- FI is an active side-channel attack in which the attacker induces stress to the target, forcing it to produce a fault result.

Introduction
Fault injection



- FI is an active side-channel attack in which the attacker induces stress to the target, forcing it to produce a fault result.
- The fault result is further used to extract secret information by differential fault analysis (fault vs no-fault).

Introduction
Fault injection techniques

There are several common techniques for FI.

[1] **barenghi2009low**; **balasch2011depth**.
[2] **riviere2015high**; **beckers2019characterization**.
[3] **skorobogatov2002optical**; **dutertre2019experimental**.

Introduction
Fault injection techniques

There are several common techniques for FI.

- Clock glitch or Voltage glitch[1]

[1] **barenghi2009low**; **balasch2011depth**.
[2] **riviere2015high**; **beckers2019characterization**.
[3] **skorobogatov2002optical**; **dutertre2019experimental**.

Introduction

Fault injection techniques

There are several common techniques for FI.

- Clock glitch or Voltage glitch[1]
- Electromagnetic fault injection[2]

[1] **barenghi2009low**; **balasch2011depth**.

[2] **riviere2015high**; **beckers2019characterization**.

[3] **skorobogatov2002optical**; **dutertre2019experimental**.

Introduction

Fault injection techniques

There are several common techniques for FI.

- Clock glitch or Voltage glitch[1]
- Electromagnetic fault injection[2]
- Laser fault injection (LFI)[3]

[1] **barenghi2009low**; **balasch2011depth**.
[2] **riviere2015high**; **beckers2019characterization**.
[3] **skorobogatov2002optical**; **dutertre2019experimental**.

Introduction

Fault injection techniques

There are several common techniques for FI.

- Clock glitch or Voltage glitch[1]
- Electromagnetic fault injection[2]
- Laser fault injection (LFI)[3]
  - The laser has a very high spacial and temporal resolution because the pulse can be confined a very small space and lasts for a very short time.

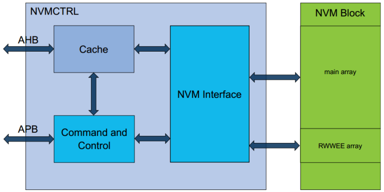
[1] **barenghi2009low**; **balasch2011depth**.

[2] **riviere2015high**; **beckers2019characterization**.

[3] **skorobogatov2002optical**; **dutertre2019experimental**.

Introduction

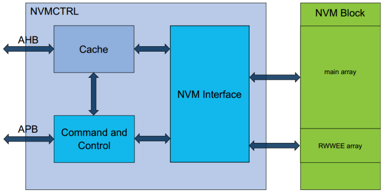
About the target: SAMD21G18A



[4] **SAMD21_datasheet_microchip**.

Introduction

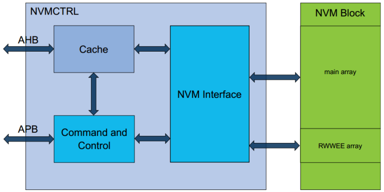
About the target: SAMD21G18A



Our target is SAMD21G18A, with the following features:[4]

[4] **SAMD21_datasheet_microchip**.

Introduction
About the target: SAMD21G18A



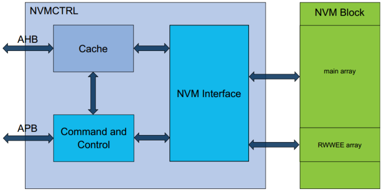
Our target is SAMD21G18A, with the following features:[4]

- is a 32-bit MCU;

[4]**SAMD21_datasheet_microchip**.

Introduction

About the target: SAMD21G18A

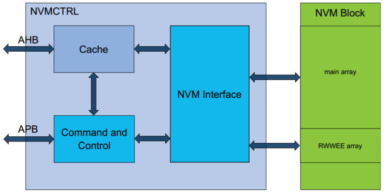


Our target is SAMD21G18A, with the following features:[4]

- is a 32-bit MCU;
- implements an ARM Cortex-M0+ (2-stage pipeline);

[4] **SAMD21_datasheet_microchip**.

Introduction
About the target: SAMD21G18A

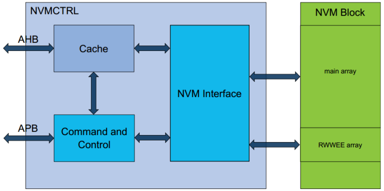


Our target is SAMD21G18A, with the following features:[4]

- is a 32-bit MCU;
- implements an ARM Cortex-M0+ (2-stage pipeline);
- has an 8 lines 64 bits cache;

[4]**SAMD21_datasheet_microchip**.

Introduction
About the target: SAMD21G18A



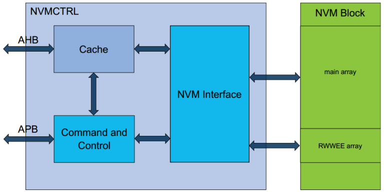
Our target is SAMD21G18A, with the following features:[4]

- is a 32-bit MCU;
- implements an ARM Cortex-M0+ (2-stage pipeline);
- has an 8 lines 64 bits cache;
- can operate at maximum frequency of 48 MHz;

[4]**SAMD21_datasheet_microchip**.

Introduction

About the target: SAMD21G18A



Our target is SAMD21G18A, with the following features:[4]

- is a 32-bit MCU;
- implements an ARM Cortex-M0+ (2-stage pipeline);
- has an 8 lines 64 bits cache;
- can operate at maximum frequency of 48 MHz;

[4]**SAMD21_datasheet_microchip**.

Introduction
Contributions of this paper

The main contributions of this paper are:

Introduction

Contributions of this paper

The main contributions of this paper are:

- being able to fault instructions from the Flash interface to the core pipeline (Flash interface buffer-> AHB bus-> core fetch -> core execution) in a 32-bit MCU;

Introduction

Contributions of this paper

The main contributions of this paper are:

- being able to fault instructions from the Flash interface to the core pipeline (Flash interface buffer-> AHB bus-> core fetch -> core execution) in a 32-bit MCU;
- identifying the faults in different stages and their behaviors, and characterizing the fault models at instruction level and bit level;

Introduction

Contributions of this paper

The main contributions of this paper are:

- being able to fault instructions from the Flash interface to the core pipeline (Flash interface buffer-> AHB bus-> core fetch -> core execution) in a 32-bit MCU;
- identifying the faults in different stages and their behaviors, and characterizing the fault models at instruction level and bit level;
- investigating the impact of LFI parameters such as the PW and the power on the faults;

Introduction

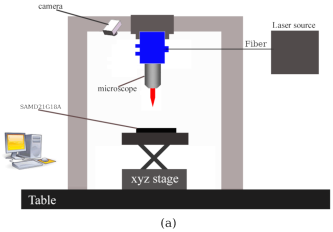
Contributions of this paper

The main contributions of this paper are:

- being able to fault instructions from the Flash interface to the core pipeline (Flash interface buffer-> AHB bus-> core fetch -> core execution) in a 32-bit MCU;
- identifying the faults in different stages and their behaviors, and characterizing the fault models at instruction level and bit level;
- investigating the impact of LFI parameters such as the PW and the power on the faults;
- comparing the instruction(s) skip fault models obtained with LFI at different positions.

Table contents

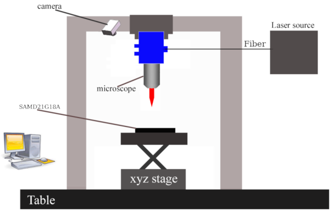
Experimental setup and methodology



(a) Laser bench

[5] **dutertre2019experimental**.

Experimental setup and methodology



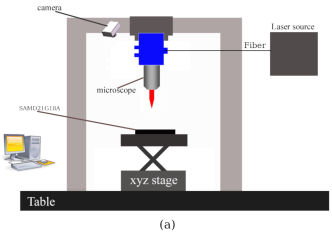
(a)

(a) Laser bench

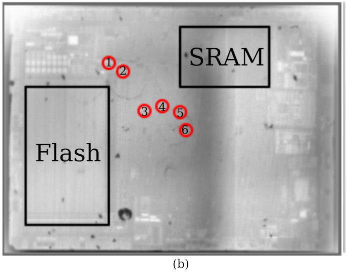
- Wavelength: 1064 nm, power: 0 - 3 W, PW: 5 ns - 1 s (more details can be found in[5]).

[5] **dutertre2019experimental**.

Experimental setup and methodology



(a) Laser bench

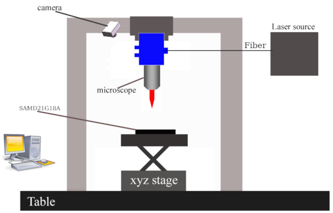


(b) Microchip back-side image

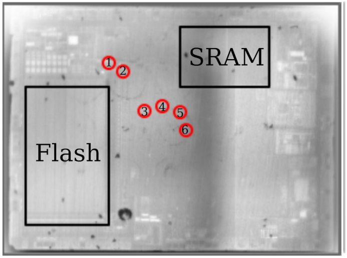
- Wavelength: 1064 nm, power: 0 - 3 W, PW: 5 ns - 1 s (more details can be found in[5]).

[5] **dutertre2019experimental**.

Experimental setup and methodology



(a)



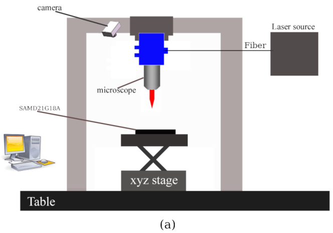
(b)

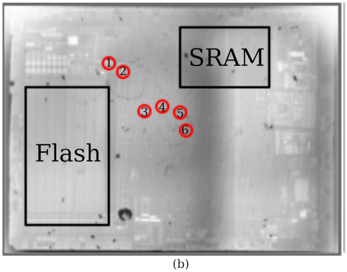
(a) Laser bench (b) Microchip back-side image

- Wavelength: 1064 nm, power: 0 - 3 W, PW: 5 ns - 1 s (more details can be found in[5]).
- The MCU was depackaged and the laser pulse was injected from the back side.

[5] **dutertre2019experimental**.

Experimental setup and methodology





(a) Laser bench (b) Microchip back-side image

- Wavelength: 1064 nm, power: 0 - 3 W, PW: 5 ns - 1 s (more details can be found in[5]).
- The MCU was depackaged and the laser pulse was injected from the back side.
- The MCU was configured to work at 12 MHz, with zero waitstate.

[5] **dutertre2019experimental**.

Experimental setup and methodology
Test procedures

A test follows three main steps:

- (1) the target is reset to initialize all systems including memories and registers;

Experimental setup and methodology

Test procedures

A test follows three main steps:

- (1) the target is reset to initialize all systems including memories and registers;
- (2) the trigger for laser pulse generator is set and the test code is then executed;

Experimental setup and methodology

Test procedures

A test follows three main steps:

- (1) the target is reset to initialize all systems including memories and registers;
- (2) the trigger for laser pulse generator is set and the test code is then executed;
- (3) registers content harvesting is performed as the program reaches the configured break-point.

Experimental setup and methodology

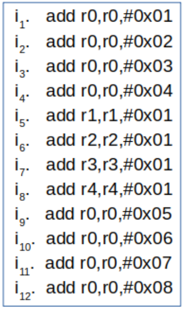
Test procedures

A test follows three main steps:

- (1) the target is reset to initialize all systems including memories and registers;
- (2) the trigger for laser pulse generator is set and the test code is then executed;
- (3) registers content harvesting is performed as the program reaches the configured break-point.

For each injection parameter, 100 tests are performed. At the beginning, a test without LFI was performed to make sure the program functions correctly and the data is used as the reference.

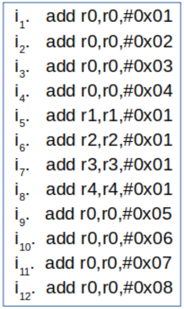
Test code and skip fault definitions



```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  add r1,r1,#0x01
i6.  add r2,r2,#0x01
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(a) test code

Test code and skip fault definitions

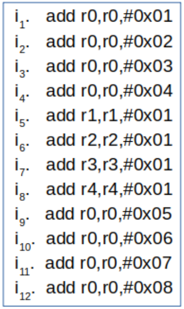


```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  add r1,r1,#0x01
i6.  add r2,r2,#0x01
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(a) test code

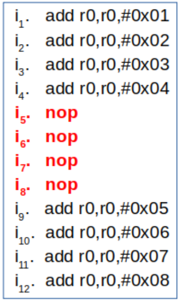
- (a) test code

Test code and skip fault definitions



```
i1. add r0,r0,#0x01
i2. add r0,r0,#0x02
i3. add r0,r0,#0x03
i4. add r0,r0,#0x04
i5. add r1,r1,#0x01
i6. add r2,r2,#0x01
i7. add r3,r3,#0x01
i8. add r4,r4,#0x01
i9. add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(a) test code

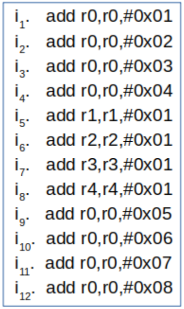


```
i1. add r0,r0,#0x01
i2. add r0,r0,#0x02
i3. add r0,r0,#0x03
i4. add r0,r0,#0x04
i5. nop
i6. nop
i7. nop
i8. nop
i9. add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(b) skip $i_5i_6i_7i_8$

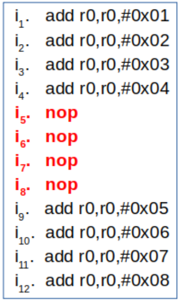
- (a) test code

Test code and skip fault definitions



```
i1. add r0,r0,#0x01
i2. add r0,r0,#0x02
i3. add r0,r0,#0x03
i4. add r0,r0,#0x04
i5. add r1,r1,#0x01
i6. add r2,r2,#0x01
i7. add r3,r3,#0x01
i8. add r4,r4,#0x01
i9. add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(a) test code

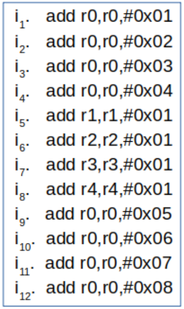


```
i1. add r0,r0,#0x01
i2. add r0,r0,#0x02
i3. add r0,r0,#0x03
i4. add r0,r0,#0x04
i5. nop
i6. nop
i7. nop
i8. nop
i9. add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(b) skip $i_5 i_6 i_7 i_8$

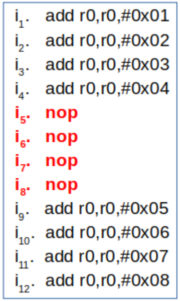
- (a) test code
- (b) skip $i_5 i_6 i_7 i_8$: instructions (i_5, i_6, i_7, i_8) are replaced by instructions equivalent to (*nop*, *nop*, *nop*, *nop*);

Test code and skip fault definitions



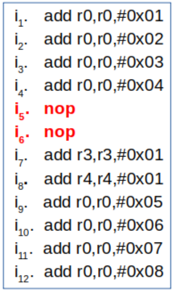
```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  add r1,r1,#0x01
i6.  add r2,r2,#0x01
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(a) test code



```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  nop
i6.  nop
i7.  nop
i8.  nop
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(b) skip $i_5i_6i_7i_8$

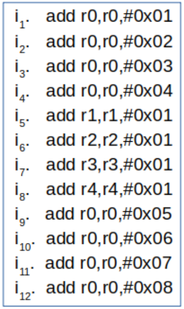


```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  nop
i6.  nop
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(c) skip i_5i_6

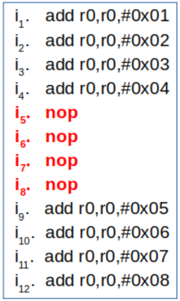
- (a) test code
- (b) skip $i_5i_6i_7i_8$: instructions (i_5, i_6, i_7, i_8) are replaced by instructions equivalent to (*nop*, *nop*, *nop*, *nop*);

Test code and skip fault definitions



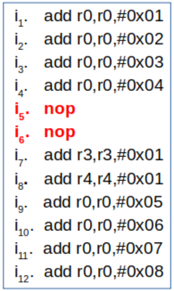
```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  add r1,r1,#0x01
i6.  add r2,r2,#0x01
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(a) test code



```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  nop
i6.  nop
i7.  nop
i8.  nop
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(b) skip $i_5 i_6 i_7 i_8$

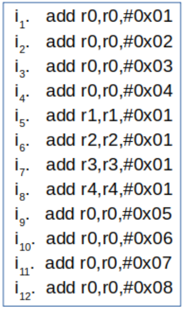


```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  nop
i6.  nop
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(c) skip $i_5 i_6$

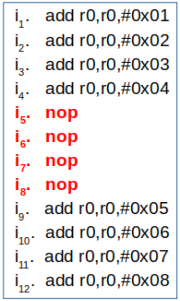
- (a) test code
- (b) skip $i_5 i_6 i_7 i_8$: instructions (i_5, i_6, i_7, i_8) are replaced by instructions equivalent to (*nop*, *nop*, *nop*, *nop*);
- (c) skip $i_5 i_6$: instructions (i_5, i_6) are replaced by (*nop*, *nop*);

Test code and skip fault definitions



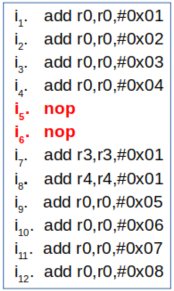
```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  add r1,r1,#0x01
i6.  add r2,r2,#0x01
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(a) test code



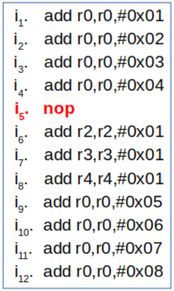
```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  nop
i6.  nop
i7.  nop
i8.  nop
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(b) skip $i_5i_6i_7i_8$



```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  nop
i6.  nop
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(c) skip i_5i_6

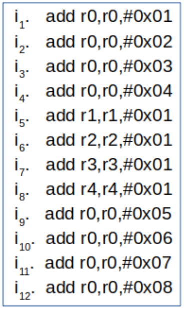


```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  nop
i6.  add r2,r2,#0x01
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(d) skip i_5

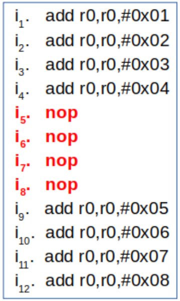
- (a) test code
- (b) skip $i_5i_6i_7i_8$: instructions (i_5, i_6, i_7, i_8) are replaced by instructions equivalent to (*nop*, *nop*, *nop*, *nop*);
- (c) skip i_5i_6: instructions (i_5, i_6) are replaced by (*nop*, *nop*);

Test code and skip fault definitions



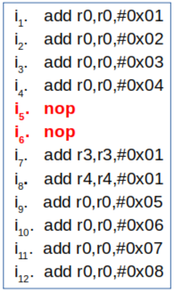
```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  add r1,r1,#0x01
i6.  add r2,r2,#0x01
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(a) test code



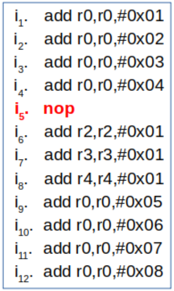
```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  nop
i6.  nop
i7.  nop
i8.  nop
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(b) skip $i_5 i_6 i_7 i_8$



```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  nop
i6.  nop
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(c) skip $i_5 i_6$

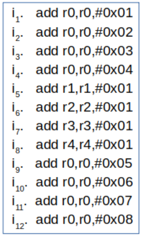


```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  nop
i6.  add r2,r2,#0x01
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(d) skip i_5

- (a) test code
- (b) skip $i_5 i_6 i_7 i_8$: instructions (i_5, i_6, i_7, i_8) are replaced by instructions equivalent to (*nop*, *nop*, *nop*, *nop*);
- (c) skip $i_5 i_6$: instructions (i_5, i_6) are replaced by (*nop*, *nop*);
- (d) skip i_5: instruction (i_5) is replaced by instruction (*nop*).

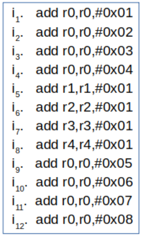
Test code and replay fault definitions



```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  add r1,r1,#0x01
i6.  add r2,r2,#0x01
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(a) test code

Test code and replay fault definitions

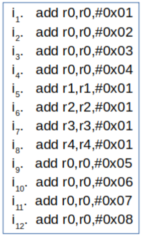


```
i1. add r0,r0,#0x01
i2. add r0,r0,#0x02
i3. add r0,r0,#0x03
i4. add r0,r0,#0x04
i5. add r1,r1,#0x01
i6. add r2,r2,#0x01
i7. add r3,r3,#0x01
i8. add r4,r4,#0x01
i9. add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(a) test code

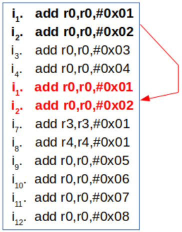
- (a) test code

Test code and replay fault definitions



```
i1. add r0,r0,#0x01
i2. add r0,r0,#0x02
i3. add r0,r0,#0x03
i4. add r0,r0,#0x04
i5. add r1,r1,#0x01
i6. add r2,r2,#0x01
i7. add r3,r3,#0x01
i8. add r4,r4,#0x01
i9. add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(a) test code

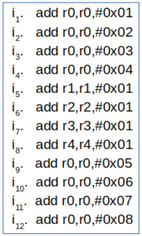


```
i1. add r0,r0,#0x01
i2. add r0,r0,#0x02
i3. add r0,r0,#0x03
i4. add r0,r0,#0x04
i1. add r0,r0,#0x01
i2. add r0,r0,#0x02
i7. add r3,r3,#0x01
i8. add r4,r4,#0x01
i9. add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(b) replay $i_1 i_2 (i_5 i_6)$

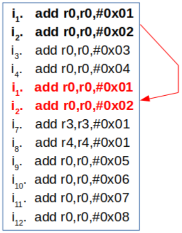
- (a) test code

Test code and replay fault definitions



```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  add r1,r1,#0x01
i6.  add r2,r2,#0x01
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(a) test code

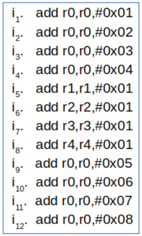


```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(b) replay $i_1 i_2 (i_5 i_6)$

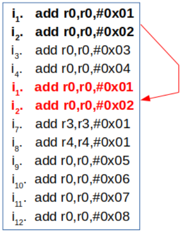
- (a) test code
- (b) replay $i_1 i_2 (i_5 i_6)$: instructions (i_5, i_6) are overwritten by instructions(i_1, i_2);

Test code and replay fault definitions



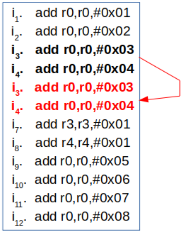
```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  add r1,r1,#0x01
i6.  add r2,r2,#0x01
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(a) test code



```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(b) replay $i_1 i_2 (i_5 i_6)$

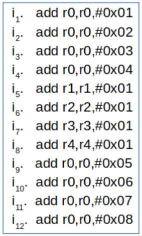


```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(c) replay $i_3 i_4 (i_5 i_6)$

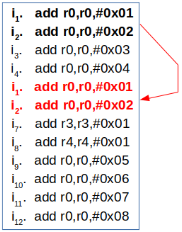
- (a) test code
- (b) replay $i_1 i_2 (i_5 i_6)$: instructions (i_5, i_6) are overwritten by instructions(i_1, i_2);

Test code and replay fault definitions



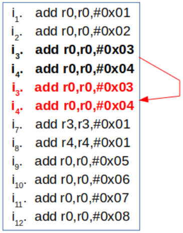
```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  add r1,r1,#0x01
i6.  add r2,r2,#0x01
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(a) test code



```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(b) replay $i_1 i_2 (i_5 i_6)$

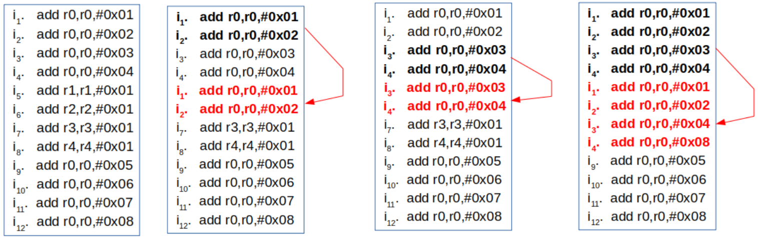


```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(c) replay $i_3 i_4 (i_5 i_6)$

- (a) test code
- (b) replay $i_1 i_2 (i_5 i_6)$: instructions (i_5, i_6) are overwritten by instructions(i_1, i_2);
- (c) replay $i_3 i_4 (i_5 i_6)$: instructions (i_5, i_6) are overwritten by instructions (i_3, i_4);

Test code and replay fault definitions



```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  add r1,r1,#0x01
i6.  add r2,r2,#0x01
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(a) test code

```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(b) replay $i_1 i_2 (i_5 i_6)$

```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

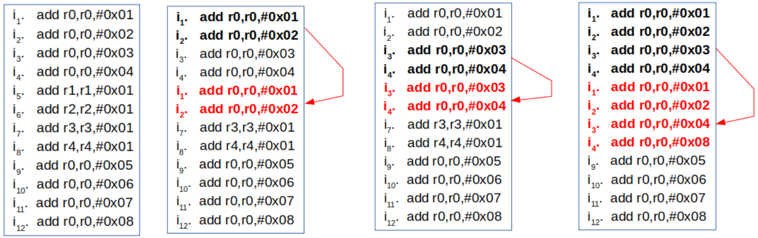
(c) replay $i_3 i_4 (i_5 i_6)$

```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x04
i4.  add r0,r0,#0x08
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(d) replay $i_1 i_2 i_3 i_4$

- (a) test code
- (b) replay $i_1 i_2 (i_5 i_6)$: instructions (i_5, i_6) are overwritten by instructions(i_1, i_2);
- (c) replay $i_3 i_4 (i_5 i_6)$: instructions (i_5, i_6) are overwritten by instructions (i_3, i_4);

Test code and replay fault definitions



```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i5.  add r1,r1,#0x01
i6.  add r2,r2,#0x01
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(a) test code

```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(b) replay $i_1 i_2 (i_5 i_6)$

```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i7.  add r3,r3,#0x01
i8.  add r4,r4,#0x01
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(c) replay $i_3 i_4 (i_5 i_6)$

```
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x03
i4.  add r0,r0,#0x04
i1.  add r0,r0,#0x01
i2.  add r0,r0,#0x02
i3.  add r0,r0,#0x04
i4.  add r0,r0,#0x08
i9.  add r0,r0,#0x05
i10. add r0,r0,#0x06
i11. add r0,r0,#0x07
i12. add r0,r0,#0x08
```

(d) replay $i_1 i_2 i_3 i_4$

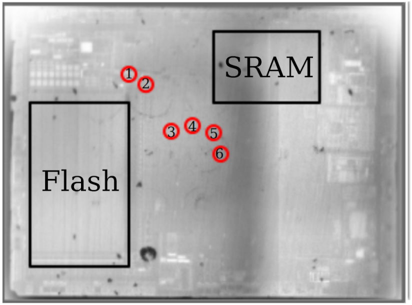
- (a) test code
- (b) replay $i_1 i_2 (i_5 i_6)$: instructions (i_5, i_6) are overwritten by instructions(i_1, i_2);
- (c) replay $i_3 i_4 (i_5 i_6)$: instructions (i_5, i_6) are overwritten by instructions (i_3, i_4);
- (d) replay $i_1 i_2 i_3 i_4$: instructions (i_5, i_6, i_7, i_8) are overwritten by instructions (i_1, i_2, i_3, i_4).

Table contents

Faults at six positions

- Laser power: 1.5 W, PW: 50 ns.

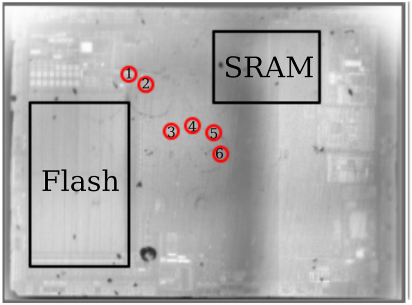
Faults at six positions



(b)

- Laser power: 1.5 W, PW: 50 ns.

Faults at six positions

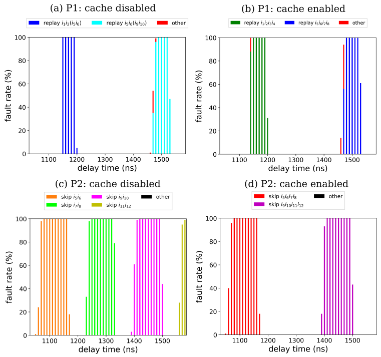


(b)

- Laser power: 1.5 W, PW: 50 ns.
- Six positions marked with red circular shapes with different fault behavior were found.

LFI-induced faults

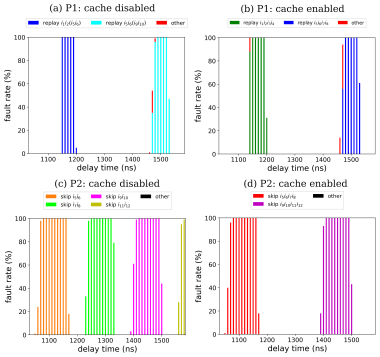
from The flash interface to the execution pipeline: P1 and P2



[6] **vkhuat_emc_europe_2021**; **vkhuat_dsd_2021**.

LFI-induced faults

from The flash interface to the execution pipeline: P1 and P2

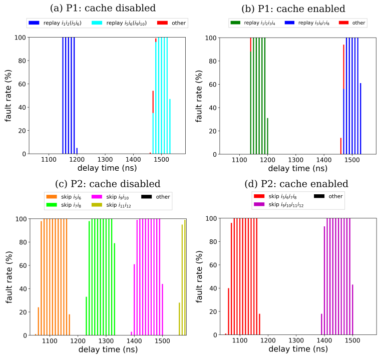


- The fault is related to block of two or four instructions depending on the cache operation mode;

[6] **vkhuat_emc_europe_2021**; **vkhuat_dsd_2021**.

LFI-induced faults

from The flash interface to the execution pipeline: P1 and P2

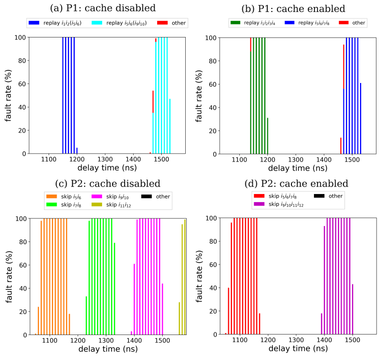


- The fault is related to block of two or four instructions depending on the cache operation mode;
- Two fault models: skip and replay of instruction block are observed;

[6] **vkhuat_emc_europe_2021**; **vkhuat_dsd_2021**.

LFI-induced faults

from The flash interface to the execution pipeline: P1 and P2



- The fault is related to block of two or four instructions depending on the cache operation mode;
- Two fault models: skip and replay of instruction block are observed;

- The fault behavior is the same with results obtained in[6], in which we ascribed the fault to impact of EMFI and LFI to the **Flash interface buffer**.

[6]**vkhuat_emc_europe_2021**; **vkhuat_dsd_2021**.

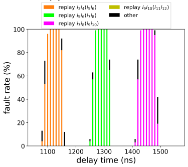
LFI-induced faults

from The flash interface to the execution pipeline: P3 and P4

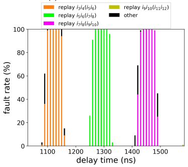
LFI-induced faults

from The flash interface to the execution pipeline: P3 and P4

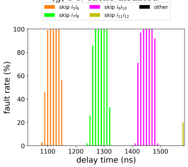
(e) P3: cache disabled



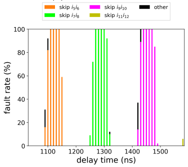
(f) P3: cache enabled



(g) P4: cache disabled

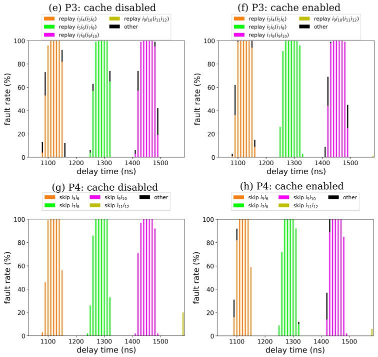


(h) P4: cache enabled



LFI-induced faults

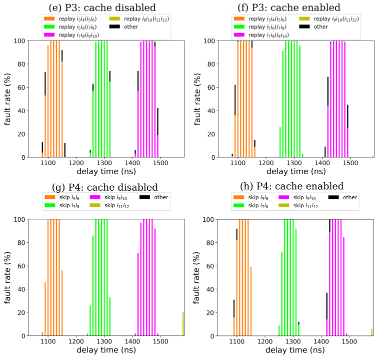
from The flash interface to the execution pipeline: P3 and P4



- The fault is related to a block of two instructions for both cache operation modes;

LFI-induced faults

from The flash interface to the execution pipeline: P3 and P4



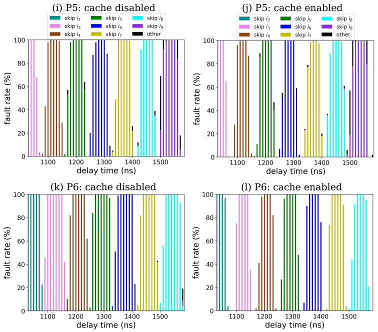
- The fault is related to a block of two instructions for both cache operation modes;
- Two fault models of skip and replay of a block of two instructions are observed.

LFI-induced faults

from The flash interface to the execution pipeline: P5 and P6

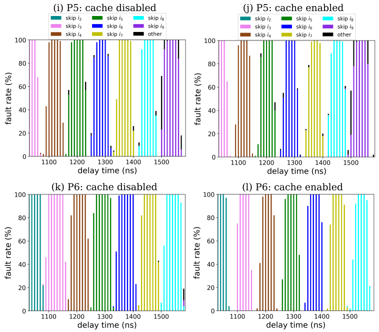
LFI-induced faults

from The flash interface to the execution pipeline: P5 and P6



LFI-induced faults

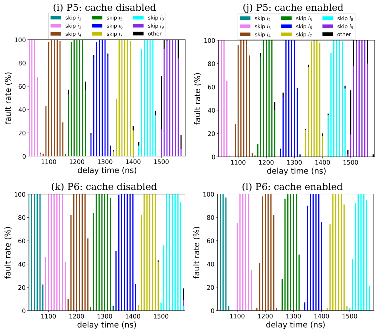
from The flash interface to the execution pipeline: P5 and P6



- The fault is related to a single instruction;

LFI-induced faults

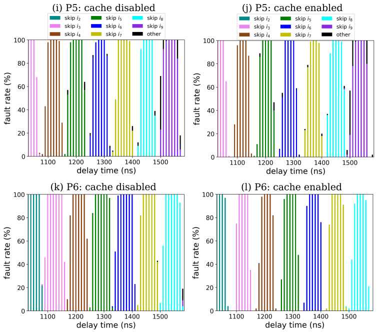
from The flash interface to the execution pipeline: P5 and P6



- The fault is related to a single instruction;
- Single instruction skip was obtained at position P5 and P6.

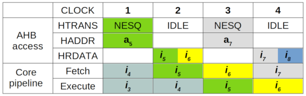
LFI-induced faults

from The flash interface to the execution pipeline: P5 and P6



- The fault is related to a single instruction;
- Single instruction skip was obtained at position P5 and P6.
- There is a phase shift of one clock cycle between the fault at position 5 and 6.

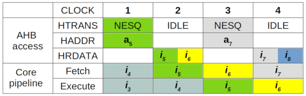
Fault mechanism hypothesis



	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_7 i_8
Core pipeline	Fetch	i_4	i_5	i_6	i_7
	Execute	i_3	i_4	i_5	i_6

(a) Normal execution

Fault mechanism hypothesis

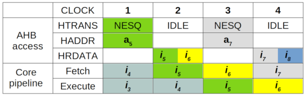


	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_7 i_8
Core pipeline	Fetch	i_4	i_5	i_6	i_7
	Execute	i_3	i_4	i_5	i_6

(a) Normal execution

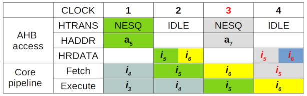
- (a) Normal execution process.

Fault mechanism hypothesis



	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_7 i_8
Core pipeline	Fetch	i_4	i_5	i_6	i_7
	Execute	i_3	i_4	i_5	i_6

(a) Normal execution

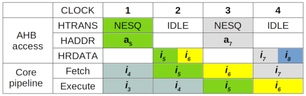


	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_5 i_6
Core pipeline	Fetch	i_4	i_5	i_6	i_5
	Execute	i_3	i_4	i_5	i_6

(b) Laser-induced replay of two instructions

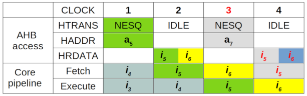
- (a) Normal execution process.

Fault mechanism hypothesis



	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_7 i_8
Core pipeline	Fetch	i_4	i_5	i_6	i_7
	Execute	i_3	i_4	i_5	i_6

(a) Normal execution

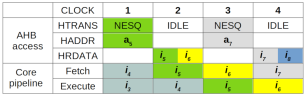


	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_5 i_6
Core pipeline	Fetch	i_4	i_5	i_6	i_5
	Execute	i_3	i_4	i_5	i_6

(b) Laser-induced replay of two instructions

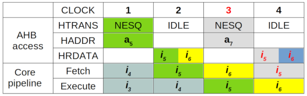
- (a) Normal execution process.
- (b) Laser-induced prevention of AHB bus update, resulting in replay of two instructions.

Fault mechanism hypothesis



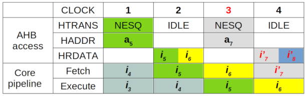
	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_7 i_8
Core pipeline	Fetch	i_4	i_5	i_6	i_7
	Execute	i_3	i_4	i_5	i_6

(a) Normal execution



	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_5 i_6
Core pipeline	Fetch	i_4	i_5	i_6	i_5
	Execute	i_3	i_4	i_5	i_6

(b) Laser-induced replay of two instructions

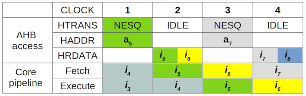


	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i'_7 i'_8
Core pipeline	Fetch	i_4	i_5	i_6	i'_7
	Execute	i_3	i_4	i_5	i_6

(c) Laser-induced modification of two instructions

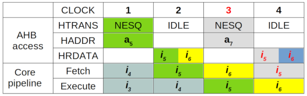
- (a) Normal execution process.
- (b) Laser-induced prevention of AHB bus update, resulting in replay of two instructions.

Fault mechanism hypothesis



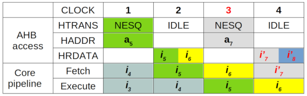
	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_7 i_8
Core pipeline	Fetch	i_4	i_5	i_6	i_7
	Execute	i_3	i_4	i_5	i_6

(a) Normal execution



	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_5 i_6
Core pipeline	Fetch	i_4	i_5	i_6	i_5
	Execute	i_3	i_4	i_5	i_6

(b) Laser-induced replay of two instructions

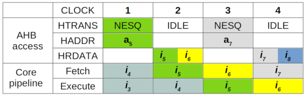


	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i'_7 i'_8
Core pipeline	Fetch	i_4	i_5	i_6	i'_7
	Execute	i_3	i_4	i_5	i_6

(c) Laser-induced modification of two instructions

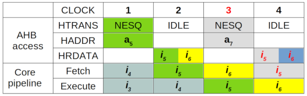
- (a) Normal execution process.
- (b) Laser-induced prevention of AHB bus update, resulting in replay of two instructions.
- (c) Laser-induced instructions corruption of data loaded into ABH bus, resulting in skip of two instructions.

Fault mechanism hypothesis



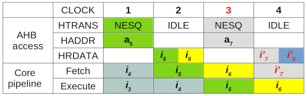
	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_7 i_8
Core pipeline	Fetch	i_4	i_5	i_6	i_7
	Execute	i_3	i_4	i_5	i_6

(a) Normal execution



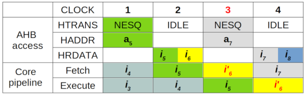
	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_5 i_6
Core pipeline	Fetch	i_4	i_5	i_6	i_5
	Execute	i_3	i_4	i_5	i_6

(b) Laser-induced replay of two instructions



	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i'_7 i'_8
Core pipeline	Fetch	i_4	i_5	i_6	i'_7
	Execute	i_3	i_4	i_5	i_6

(c) Laser-induced modification of two instructions

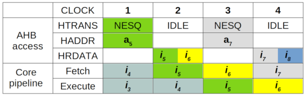


	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_7 i_8
Core pipeline	Fetch	i_4	i_5	i'_6	i_7
	Execute	i_3	i_4	i_5	i'_6

(d) Laser-induced fault on core pipeline fetch stage

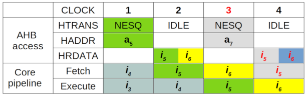
- (a) Normal execution process.
- (b) Laser-induced prevention of AHB bus update, resulting in replay of two instructions.
- (c) Laser-induced instructions corruption of data loaded into ABH bus, resulting in skip of two instructions.

Fault mechanism hypothesis



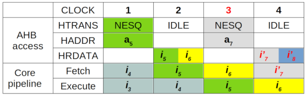
	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_7 i_8
Core pipeline	Fetch	i_4	i_5	i_6	i_7
	Execute	i_3	i_4	i_5	i_6

(a) Normal execution



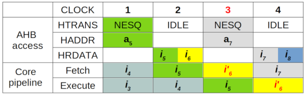
	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_5 i_6
Core pipeline	Fetch	i_4	i_5	i_6	i_5
	Execute	i_3	i_4	i_5	i_6

(b) Laser-induced replay of two instructions



	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i'_7 i'_8
Core pipeline	Fetch	i_4	i_5	i_6	i'_7
	Execute	i_3	i_4	i_5	i_6

(c) Laser-induced modification of two instructions

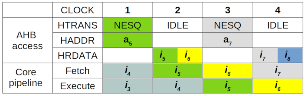


	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_7 i_8
Core pipeline	Fetch	i_4	i_5	i'_6	i_7
	Execute	i_3	i_4	i_5	i'_6

(d) Laser-induced fault on core pipeline fetch stage

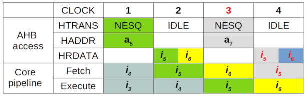
- (a) Normal execution process.
- (b) Laser-induced prevention of AHB bus update, resulting in replay of two instructions.
- (c) Laser-induced instructions corruption of data loaded into ABH bus, resulting in skip of two instructions.
- (d) Laser-induced fault on pipeline fetch.

Fault mechanism hypothesis



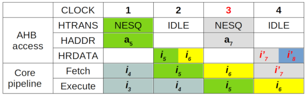
	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_7 i_8
Core pipeline	Fetch	i_4	i_5	i_6	i_7
	Execute	i_3	i_4	i_5	i_6

(a) Normal execution



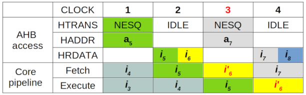
	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_5 i_6
Core pipeline	Fetch	i_4	i_5	i_6	i_5
	Execute	i_3	i_4	i_5	i_6

(b) Laser-induced replay of two instructions



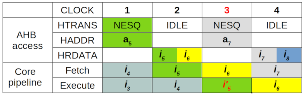
	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i'_7 i'_8
Core pipeline	Fetch	i_4	i_5	i_6	i'_7
	Execute	i_3	i_4	i_5	i_6

(c) Laser-induced modification of two instructions



	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_7 i_8
Core pipeline	Fetch	i_4	i_5	i'_6	i_7
	Execute	i_3	i_4	i_5	i'_6

(d) Laser-induced fault on core pipeline fetch stage

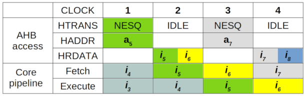


	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_7 i_8
Core pipeline	Fetch	i_4	i_5	i_6	i_7
	Execute	i_3	i_4	i'_5	i_6

(e) Laser-induced fault on core pipeline execution stage

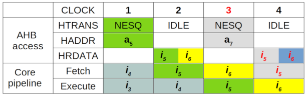
- (a) Normal execution process.
- (b) Laser-induced prevention of AHB bus update, resulting in replay of two instructions.
- (c) Laser-induced instructions corruption of data loaded into ABH bus, resulting in skip of two instructions.
- (d) Laser-induced fault on pipeline fetch.

Fault mechanism hypothesis



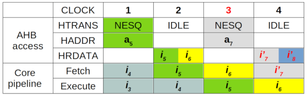
	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_7 i_8
Core pipeline	Fetch	i_4	i_5	i_6	i_7
	Execute	i_3	i_4	i_5	i_6

(a) Normal execution



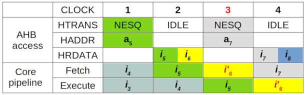
	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_5 i_6
Core pipeline	Fetch	i_4	i_5	i_6	i_5
	Execute	i_3	i_4	i_5	i_6

(b) Laser-induced replay of two instructions



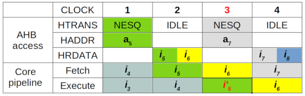
	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i'_7 i'_8
Core pipeline	Fetch	i_4	i_5	i_6	i'_7
	Execute	i_3	i_4	i_5	i_6

(c) Laser-induced modification of two instructions



	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_7 i_8
Core pipeline	Fetch	i_4	i_5	i'_6	i_7
	Execute	i_3	i_4	i_5	i'_6

(d) Laser-induced fault on core pipeline fetch stage



	CLOCK	1	2	3	4
AHB access	HTRANS	NESQ	IDLE	NESQ	IDLE
	HADDR	a_5		a_7	
	HRDATA		i_5 i_6		i_7 i_8
Core pipeline	Fetch	i_4	i_5	i_6	i_7
	Execute	i_3	i_4	i'_5	i_6

(e) Laser-induced fault on core pipeline execution stage

- (a) Normal execution process.
- (b) Laser-induced prevention of AHB bus update, resulting in replay of two instructions.
- (c) Laser-induced instructions corruption of data loaded into ABH bus, resulting in skip of two instructions.
- (d) Laser-induced fault on pipeline fetch.
- (e) Laser-induced fault on the pipeline execution.

Fault identification

- **Position P1**: the replay of a block of instructions due to laser-induced prevention of the Flash interface buffer updating process;

Fault identification

- **Position P1**: the replay of a block of instructions due to laser-induced prevention of the Flash interface buffer updating process;
- **Position P2**: the modification of a block instructions (including skip) due to laser-induced bit corruption of instruction's opcodes in the Flash interface buffer;

Fault identification

- **Position P1**: the replay of a block of instructions due to laser-induced prevention of the Flash interface buffer updating process;
- **Position P2**: the modification of a block instructions (including skip) due to laser-induced bit corruption of instruction's opcodes in the Flash interface buffer;
- **Position P3**: the replay of two instructions due to laser-induced prevention of loading data into the AHB bus;

Fault identification

- **Position P1**: the replay of a block of instructions due to laser-induced prevention of the Flash interface buffer updating process;
- **Position P2**: the modification of a block instructions (including skip) due to laser-induced bit corruption of instruction's opcodes in the Flash interface buffer;
- **Position P3**: the replay of two instructions due to laser-induced prevention of loading data into the AHB bus;
- **Position P4**: the modification of two instructions (including skip) due to laser-induced bit(s) corruption of instructions loaded into the AHB bus;

Fault identification

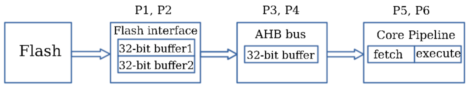
- **Position P1**: the replay of a block of instructions due to laser-induced prevention of the Flash interface buffer updating process;
- **Position P2**: the modification of a block instructions (including skip) due to laser-induced bit corruption of instruction's opcodes in the Flash interface buffer;
- **Position P3**: the replay of two instructions due to laser-induced prevention of loading data into the AHB bus;
- **Position P4**: the modification of two instructions (including skip) due to laser-induced bit(s) corruption of instructions loaded into the AHB bus;
- **Position P5**: the modification of a single instruction (including skip) due to laser-induced fault in the core pipeline fetch stage;

Fault identification

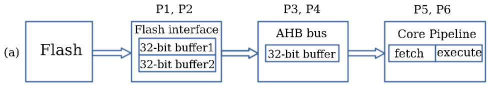
- **Position P1**: the replay of a block of instructions due to laser-induced prevention of the Flash interface buffer updating process;
- **Position P2**: the modification of a block instructions (including skip) due to laser-induced bit corruption of instruction's opcodes in the Flash interface buffer;
- **Position P3**: the replay of two instructions due to laser-induced prevention of loading data into the AHB bus;
- **Position P4**: the modification of two instructions (including skip) due to laser-induced bit(s) corruption of instructions loaded into the AHB bus;
- **Position P5**: the modification of a single instruction (including skip) due to laser-induced fault in the core pipeline fetch stage;
- **Position P6**: the modification of a single instruction (including skip) due to laser-induced fault in the core pipeline execution stage.

Proposed core architecture

(a)

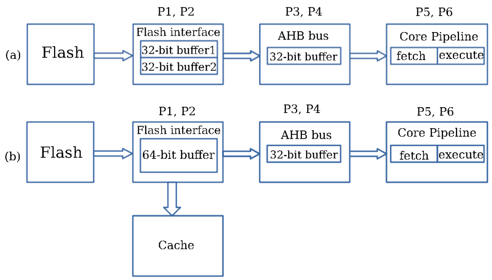


Proposed core architecture



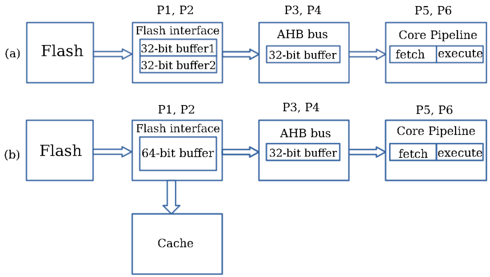
- (a) cache disabled;

Proposed core architecture



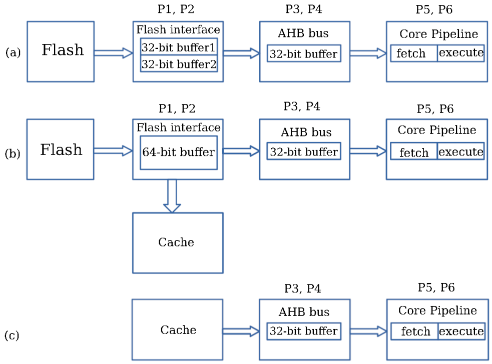
- (a) cache disabled;

Proposed core architecture



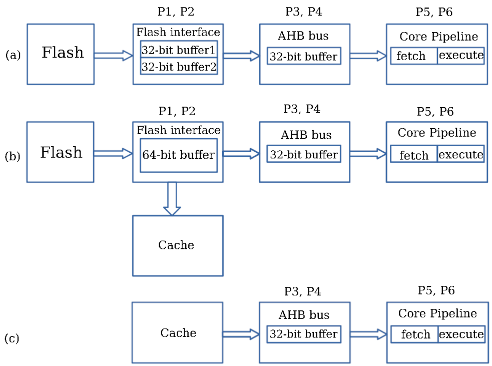
- (a) cache disabled;
- (b) cache enabled: cache miss;

Proposed core architecture



- (a) cache disabled;
- (b) cache enabled: cache miss;

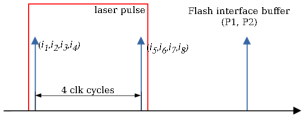
Proposed core architecture



- (a) cache disabled;
- (b) cache enabled: cache miss;
- (c) cache enabled: cache hit.

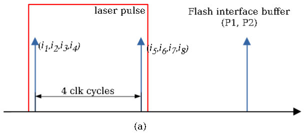
Table contents

Impact of the PW on the faults

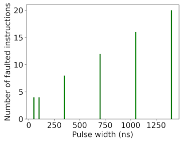


(a)

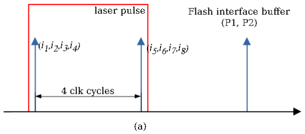
Impact of the PW on the faults

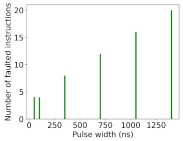


(a)



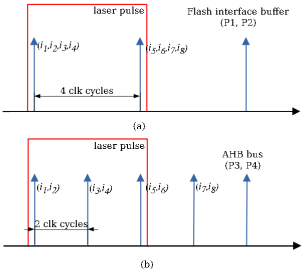
Impact of the **PW** on the faults

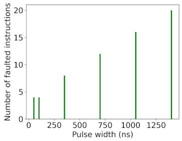




- (a) Flash interface buffer: 20 faulted instructions.

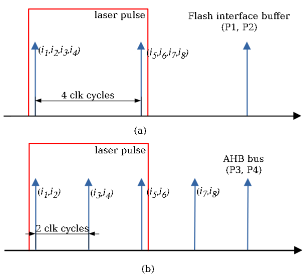
Impact of the PW on the faults

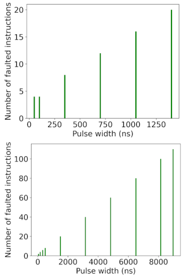




- (a) Flash interface buffer: 20 faulted instructions.

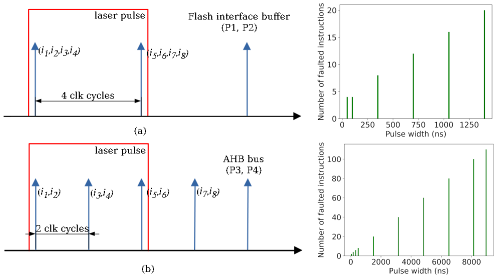
Impact of the PW on the faults





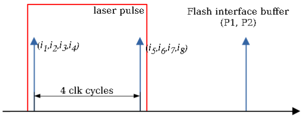
- (a) Flash interface buffer: 20 faulted instructions.

Impact of the PW on the faults

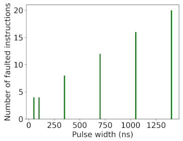


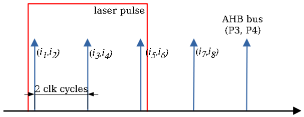
- (a) Flash interface buffer: 20 faulted instructions.
- (b) AHB bus: 110 faulted instructions

Impact of the PW on the faults

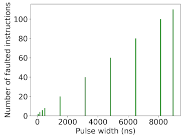


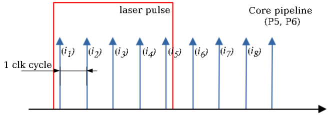
(a)





(b)

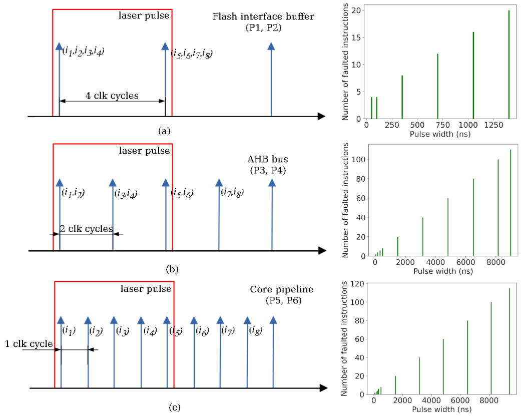




(c)

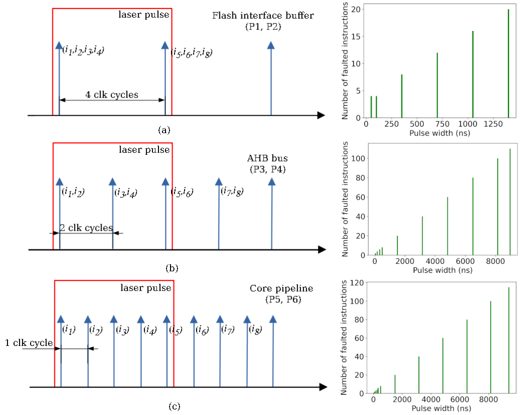
- (a) Flash interface buffer: 20 faulted instructions.
- (b) AHB bus: 110 faulted instructions

Impact of the PW on the faults



- (a) Flash interface buffer: 20 faulted instructions.
- (b) AHB bus: 110 faulted instructions

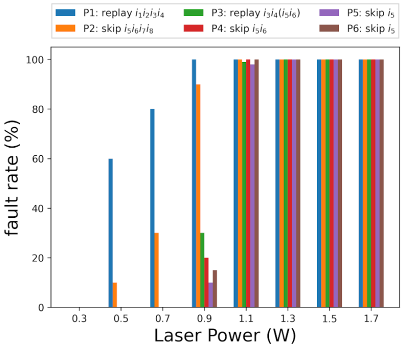
Impact of the PW on the faults



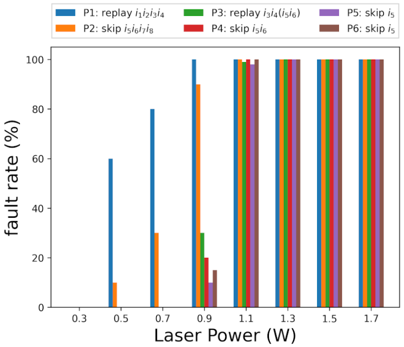
- (a) Flash interface buffer: 20 faulted instructions.
- (b) AHB bus: 110 faulted instructions
- (c) Pipeline (fetch or execution): 115 faulted instructions

Table contents

Impact of the laser power on the fault rates

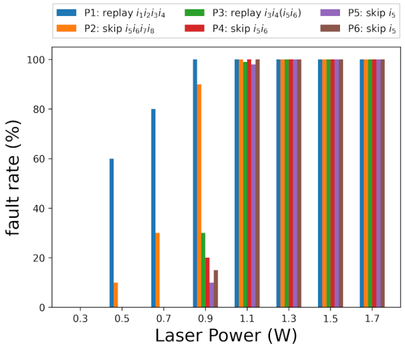


Impact of the laser power on the fault rates



- The laser power has a direct impact on the fault rates; as the power increases the fault rates increase accordingly.

Impact of the laser power on the fault rates



- The laser power has a direct impact on the fault rates; as the power increases the fault rates increase accordingly.
- The Flash interface buffer seems to be more sensitive to the laser pulse as compared to the AHB bus and the core pipeline.

Table contents

Fault at bit level characterization

test code

```
lsl r0,r0, #0x00
lsl r0,r0, #0x00
lsl r0,r0, #0x00
lsl r0,r0, #0x00
```

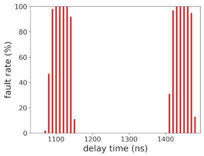
(a) bit-set detection

```
sub r7,r7, #0xff
sub r7,r7, #0xff
sub r7,r7, #0xff
sub r7,r7, #0xff
```

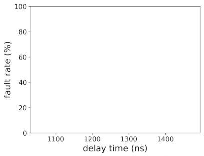
(b) bit-reset detection

- The opcode of lsl r0,r0,#0x00 is 0x0000 (all bits' values are 0)
- The opcode of sub r7,r7,#0xff is 0x3fff (most of the bits'values are 1)

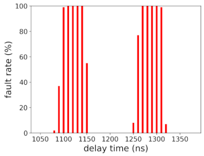
Fault at bit level characterization



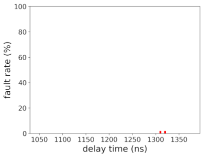
(a) P2: Flash interface buffer bit-reset fault rate



(b) P2: Flash interface buffer bit-set fault rate

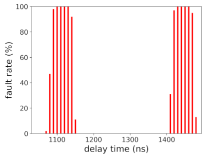


(c) P4: data loaded into AHB bus bit-reset fault rate

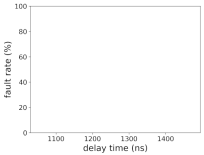


(d) P4: data loaded into AHB bus bit-set fault rate

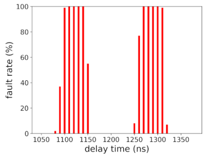
Fault at bit level characterization



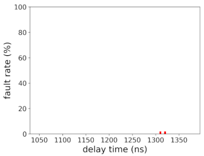
(a) P2: Flash interface buffer bit-reset fault rate



(b) P2: Flash interface buffer bit-set fault rate



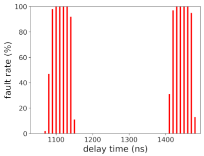
(c) P4: data loaded into AHB bus bit-reset fault rate



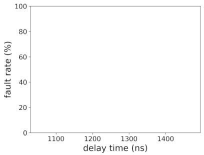
(d) P4: data loaded into AHB bus bit-set fault rate

- Many faults were detected when the buffers were filled with bits at 1.

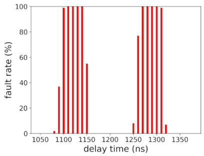
Fault at bit level characterization



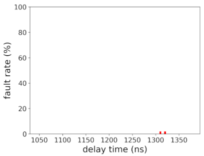
(a) P2: Flash interface buffer bit-reset fault rate



(b) P2: Flash interface buffer bit-set fault rate



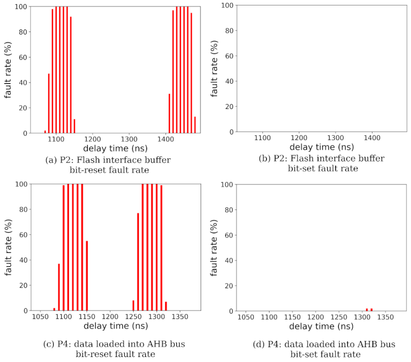
(c) P4: data loaded into AHB bus bit-reset fault rate



(d) P4: data loaded into AHB bus bit-set fault rate

- Many faults were detected when the buffers were filled with bits at 1.
- Almost no fault was detected when the buffers were filled with bits at 0.

Fault at bit level characterization



- Many faults were detected when the buffers were filled with bits at 1.
- Almost no fault was detected when the buffers were filled with bits at 0.
- At bit level the faults are **bit-reset** rather than **bit-set**.

Table contents

Test code

Test code

```
.........
ldr r1, #address
ldr r2, #address
ldr r3, #address
ldr r4, #address
.........
```

(a) test code

Test code

```
.........
ldr r1, #address
ldr r2, #address
ldr r3, #address
ldr r4, #address
.........
```

(a) test code

....

~~ldr r1, #address~~

~~ldr r2, #address~~

~~ldr r3, #address~~

~~ldr r4, #address~~

....

(b) skip fault

Test code

```
.........
ldr r1, #address
ldr r2, #address
ldr r3, #address
ldr r4, #address
.........
```

(a) test code

....

~~ldr r1, #address~~

~~ldr r2, #address~~

~~ldr r3, #address~~

~~ldr r4, #address~~

....

(b) skip fault

- **"skip"** fault models were obtained by faulting the Flash interface buffer, AHB bus, Pipeline: fetch, Pipeline: execution.

Test code

```
.........
ldr r1, #address
ldr r2, #address
ldr r3, #address
ldr r4, #address
.........
```

(a) test code

....

~~ldr r1, #address~~

~~ldr r2, #address~~

~~ldr r3, #address~~

~~ldr r4, #address~~

....

(b) skip fault

- **"skip"** fault models were obtained by faulting the Flash interface buffer, AHB bus, Pipeline: fetch, Pipeline: execution.
- The execution time of instruction **ldr rx, #address** is two clock cycles.

Test code

```
.........
ldr r1, #address
ldr r2, #address
ldr r3, #address
ldr r4, #address
.........
```

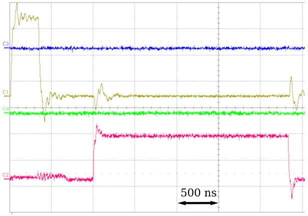
(a) test code

....
~~ldr r1, #address~~
~~ldr r2, #address~~
~~ldr r3, #address~~
~~ldr r4, #address~~
....

(b) skip fault

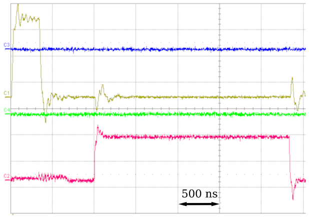
- **"skip"** fault models were obtained by faulting the Flash interface buffer, AHB bus, Pipeline: fetch, Pipeline: execution.
- The execution time of instruction **ldr rx, #address** is two clock cycles.
- The execution time of instruction **nop** is one clock cycles.

Signals taken from oscilloscope

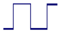


(a) No fault

Signals taken from oscilloscope



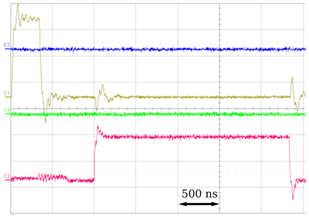
(a) No fault



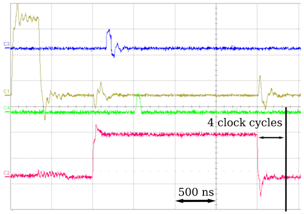
1 clock cycle= ~83.2 ns

🟦	pulse command
🟨	trigger
🟩	pulse image
🟥	test code execution window

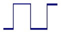
Signals taken from oscilloscope



(a) No fault



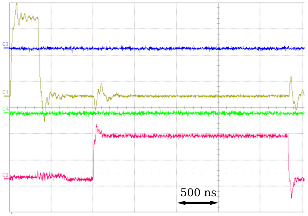
(b) P2: Fault on the Flash interface buffer



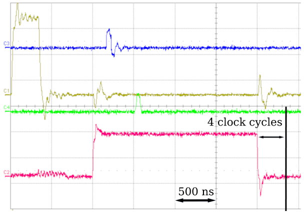
1 clock cycle= ~83.2 ns

🟦	pulse command
🟨	trigger
🟩	pulse image
🟥	test code execution window

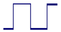
Signals taken from oscilloscope



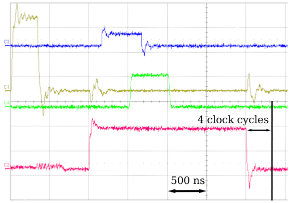
(a) No fault



(b) P2: Fault on the Flash interface buffer



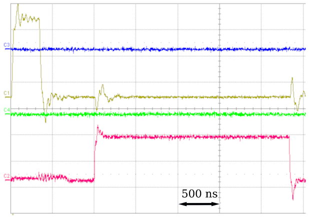
1 clock cycle= ~83.2 ns



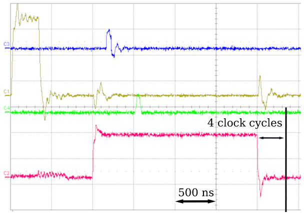
(c) P4: Fault on data loaded into the AHB bus

🟦	pulse command
🟨	trigger
🟩	pulse image
🟥	test code execution window

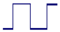
Signals taken from oscilloscope



(a) No fault

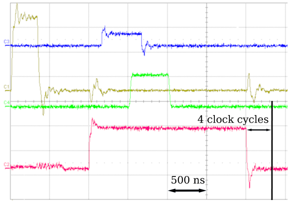


(b) P2: Fault on the Flash interface buffer



1 clock cycle= ~83.2 ns

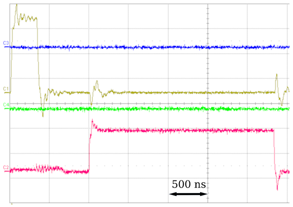
🟦	pulse command
🟨	trigger
🟩	pulse image
🟥	test code execution window



(c) P4: Fault on data loaded into the AHB bus

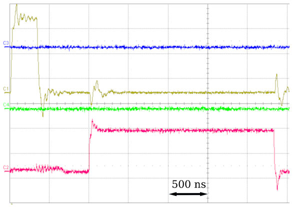
- Flash interface buffer, AHB bus: Reduction in the length of code execution windows by 4 clocks cycles. -> *ldr* instructions were replaced by *nop* operations.

Signals taken from oscilloscope



(a) No fault

Signals taken from oscilloscope

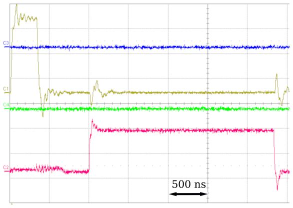


(a) No fault

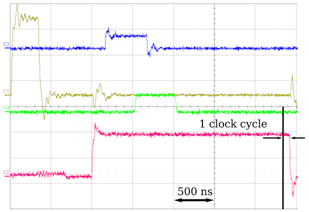
1 clock cycle= ~83.2 ns

- pulse command
- trigger
- pulse image
- test code execution window

Signals taken from oscilloscope



(a) No fault

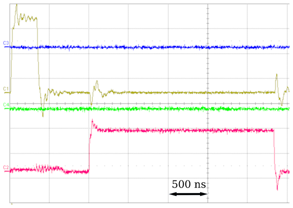


(d) P5: Fault on the core fetch

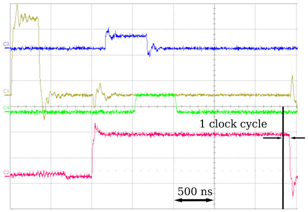
1 clock cycle= ~83.2 ns

- pulse command
- trigger
- pulse image
- test code execution window

Signals taken from oscilloscope



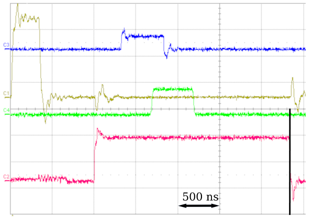
(a) No fault



(d) P5: Fault on the core fetch

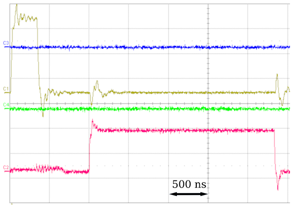
1 clock cycle= ~83.2 ns

🟦	pulse command
🟨	trigger
🟩	pulse image
🟥	test code execution window

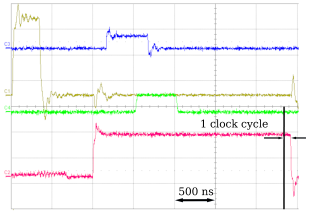


(e) P6: Fault on the core execution

Signals taken from oscilloscope

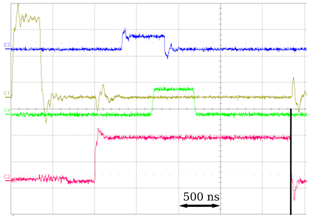


(a) No fault



(d) P5: Fault on the core fetch

🟦	pulse command
🟨	trigger
🟩	pulse image
🟥	test code execution window



(e) P6: Fault on the core execution

- Pipeline: No reduction in the length of code execution windows.
  -> *ldr* instructions were replaced by "unknown" operations.

Table contents

Conclusions

Conclusions

- By using LFI, we were able to fault the instructions in a 32-bit MCU from Flash interface buffer -> AHB bus -> fetch-> execution at six different positions.

Conclusions

- By using LFI, we were able to fault the instructions in a 32-bit MCU from Flash interface buffer -> AHB bus -> fetch-> execution at six different positions.
- Fault rate of 100% was obtained at all the positions.

Conclusions

- By using LFI, we were able to fault the instructions in a 32-bit MCU from Flash interface buffer -> AHB bus -> fetch-> execution at six different positions.
- Fault rate of 100% was obtained at all the positions.
- Skip and replay of block of instructions were obtained by faulting Flash interface buffer and the AHB bus.

Conclusions

- By using LFI, we were able to fault the instructions in a 32-bit MCU from Flash interface buffer -> AHB bus -> fetch-> execution at six different positions.
- Fault rate of 100% was obtained at all the positions.
- Skip and replay of block of instructions were obtained by faulting Flash interface buffer and the AHB bus.
- Replay fault was ascribed to laser-induced buffer update prevention.

Conclusions

- By using LFI, we were able to fault the instructions in a 32-bit MCU from Flash interface buffer -> AHB bus -> fetch-> execution at six different positions.
- Fault rate of 100% was obtained at all the positions.
- Skip and replay of block of instructions were obtained by faulting Flash interface buffer and the AHB bus.
- Replay fault was ascribed to laser-induced buffer update prevention.
- Skip fault was ascribed to laser-induced instruction modification.

Conclusions

- By using LFI, we were able to fault the instructions in a 32-bit MCU from Flash interface buffer -> AHB bus -> fetch-> execution at six different positions.
- Fault rate of 100% was obtained at all the positions.
- Skip and replay of block of instructions were obtained by faulting Flash interface buffer and the AHB bus.
- Replay fault was ascribed to laser-induced buffer update prevention.
- Skip fault was ascribed to laser-induced instruction modification.
- At the Flash interface: When the cache is disabled, the block size is 32 bits. When the cache is enabled the block size is 64 bits.

Conclusions

- By using LFI, we were able to fault the instructions in a 32-bit MCU from Flash interface buffer -> AHB bus -> fetch-> execution at six different positions.
- Fault rate of 100% was obtained at all the positions.
- Skip and replay of block of instructions were obtained by faulting Flash interface buffer and the AHB bus.
- Replay fault was ascribed to laser-induced buffer update prevention.
- Skip fault was ascribed to laser-induced instruction modification.
- At the Flash interface: When the cache is disabled, the block size is 32 bits. When the cache is enabled the block size is 64 bits.
- At the AHB bus, the faults is with a block of two instructions in both cache operation modes.

Conclusions

- By using LFI, we were able to fault the instructions in a 32-bit MCU from Flash interface buffer -> AHB bus -> fetch-> execution at six different positions.
- Fault rate of 100% was obtained at all the positions.
- Skip and replay of block of instructions were obtained by faulting Flash interface buffer and the AHB bus.
- Replay fault was ascribed to laser-induced buffer update prevention.
- Skip fault was ascribed to laser-induced instruction modification.
- At the Flash interface: When the cache is disabled, the block size is 32 bits. When the cache is enabled the block size is 64 bits.
- At the AHB bus, the faults is with a block of two instructions in both cache operation modes.
- The faults of pipeline fetch and execution are with a single instruction, and single instruction skip with fault rate of 100 % was obtained.

Conclusions

Conclusions

- There is a difference of one clock cycle between the fault of the two stages in the pipeline.

Conclusions

- There is a difference of one clock cycle between the fault of the two stages in the pipeline.
- The laser power has a direct impact on the fault rate. And the Flash interface buffer seems to be more sensitive to the laser pulse than the AHB bus and the Pipeline since smaller laser power is needed to induced fault on it.

Conclusions

- There is a difference of one clock cycle between the fault of the two stages in the pipeline.
- The laser power has a direct impact on the fault rate. And the Flash interface buffer seems to be more sensitive to the laser pulse than the AHB bus and the Pipeline since smaller laser power is needed to induced fault on it.
- Tens to more than one hundred of instructions were faulted by increasing the laser PW.

Conclusions

- There is a difference of one clock cycle between the fault of the two stages in the pipeline.
- The laser power has a direct impact on the fault rate. And the Flash interface buffer seems to be more sensitive to the laser pulse than the AHB bus and the Pipeline since smaller laser power is needed to induced fault on it.
- Tens to more than one hundred of instructions were faulted by increasing the laser PW.
- At bit level, the faults at Flash interface buffer and AHB bus were identified to to be bit-reset rather than bit-set.

Conclusions

- There is a difference of one clock cycle between the fault of the two stages in the pipeline.
- The laser power has a direct impact on the fault rate. And the Flash interface buffer seems to be more sensitive to the laser pulse than the AHB bus and the Pipeline since smaller laser power is needed to induced fault on it.
- Tens to more than one hundred of instructions were faulted by increasing the laser PW.
- At bit level, the faults at Flash interface buffer and AHB bus were identified to to be bit-reset rather than bit-set.
- The skips fault obtained at different positions were compared by comparing the related signals such as the pulse duration, the execution windows.

Future works

- Validation of the faults obtained in this work on other devices.

Thanks for your attention!