

DE LA RECHERCHE À L'INDUSTRIE



ELECTROMAGNETIC FAULT INJECTION: TOWARDS A FAULT MODEL ON A 32-BIT MICROCONTROLLER

Nicolas Moro^{1,3}, Amine Dehbaoui², Karine Heydemann³,
Bruno Robisson¹, Emmanuelle Encrenaz³

¹ **CEA**

Commissariat à l'Energie Atomique et aux Energies Alternatives

² **ENSM.SE**

Ecole Nationale Supérieure des Mines de Saint-Etienne

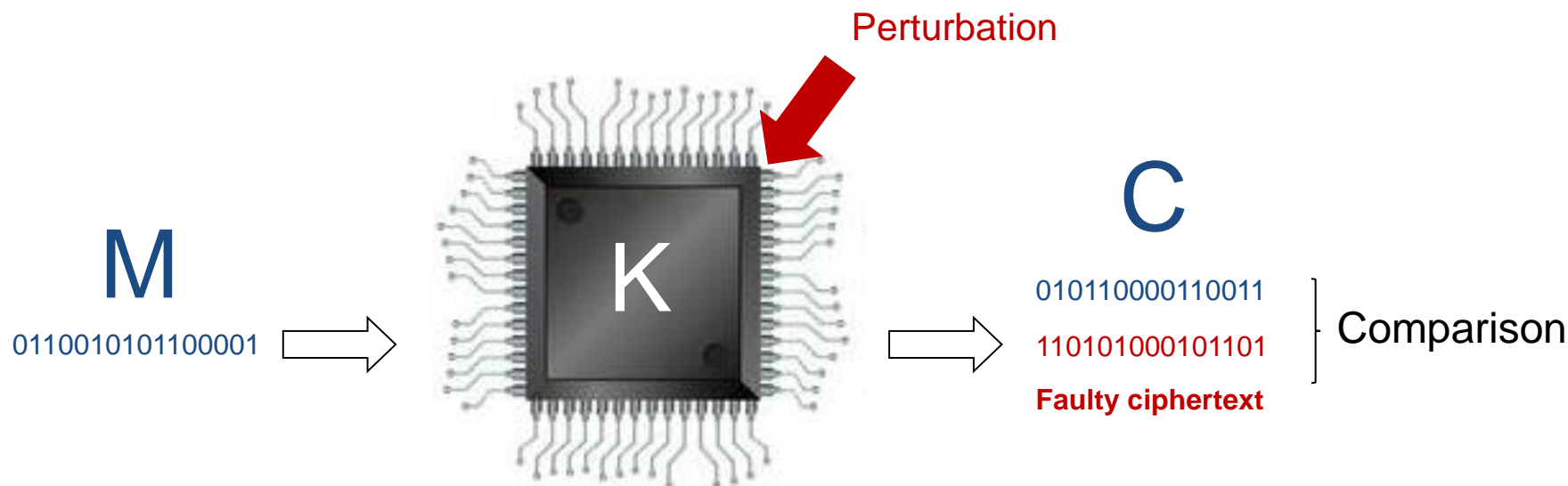
³ **LIP6 - UPMC**

Laboratoire d'Informatique de Paris 6
Université Pierre et Marie Curie

FDTC 2013 – AUGUST 20, SANTA BARBARA, USA

- **Security** of microcontroller-based embedded systems against fault injection attacks
 - **Target:** ARM Cortex-M3 microcontroller
 - **Fault injection means:** Pulsed electromagnetic fault injection
 - Theoretical attacks rely on an **attacker's fault model**
 - Electromagnetic fault injection is quite recent
 - Very few **in-depths studies** of the effects on complex systems
- ➔ Better understanding of the effects of **EM fault injection**
- ➔ Detailed fault model at a **register-transfer level**

- ➔ **I. Experimental setup**
- II. General approach**
- III. Study of the injection parameters**
- IV. Register-transfer level fault model**
- V. Conclusion**

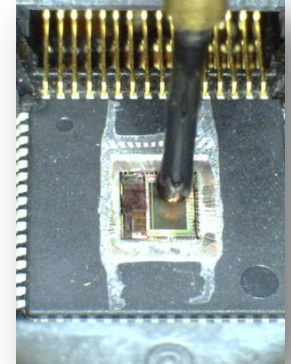


- Several **physical ways** to inject faults into a circuit's computation
- Necessary for an attacker to **know the type of injected faults**

Fault target	Data, instructions
Fault type	Bit flip, reset at 0, set at 1, stuck
Granularity	Bit, byte, word
Determinism	Deterministic, metastable, random
Temporal aspect	Single piece of data/instruction, multiple

Pulsed electromagnetic fault injection

- **Transient and local** effect of the fault injection
- **Standard circuits not protected** against this technique
- **Solenoid** used as an injection antenna
- Up to **200V** sent on the injection antenna, pulses width longer than **10ns**



Microcontroller based on an ARM Cortex-M3

- Frequency 56 MHz
- 16/32 bits **Thumb2** RISC instruction set
- ARMv7-M modified Harvard architecture
- SWD link to **debug the microcontroller**

- Experiment **driven by the computer**
- Execution of a **computation on the target device**
- Sending of a **voltage pulse**
- Stop** of the microcontroller
- Harvesting** of the microcontroller's internal data
- Analysis of the obtained results

```

C:\Documents and Settings\Wicolas\MyCloud\These\EMSE\STM32\Mesures\STM32\bin\Debug\Mesures STM32.exe
[OM] Connexion avec succès
[OM] Chargement du programme
[OM] Démarrage du debug
Nombre d'exécutions : 4

Exécution normale pour calibration
[OM] Début de l'exécution du programme
[OM] 1 seconde de pause
[OM] Arrêt de la carte
[OM] Récupération des registres
[OM] Récupération du registre xPSR
[OM] Récupération du Fault Status Register
[OM] Reset de la carte pour l'exécution suivante

Registres:
R0=0x20000010
R1=0x20000010
R2=0x00000000
R3=0x00000000
R4=0x00000000
R5=0x00000000
R6=0x00000000
R7=0x00000000
R8=0x00000000
R9=0x00000000
R10=0x00000000
R11=0x00000000
R12=0x00000000

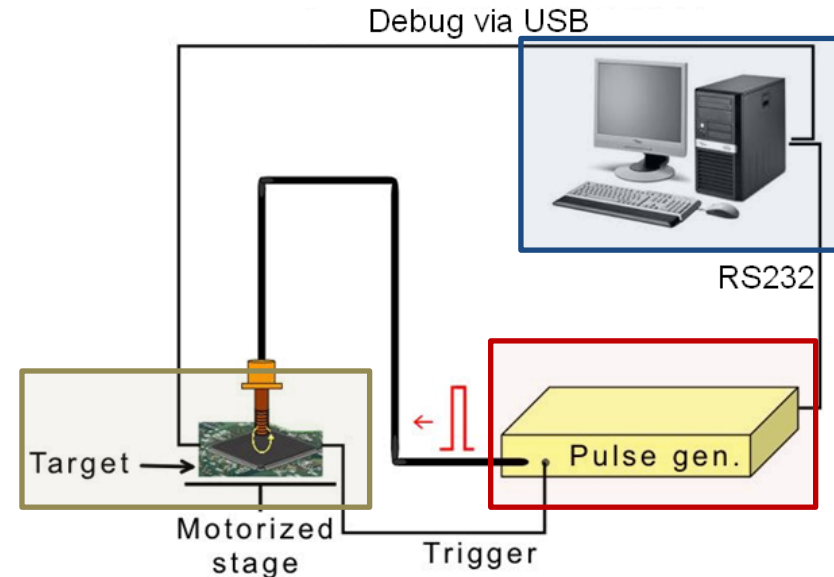
Registres particuliers:
R13 Stack Pointer = 0x20000100
R14 Link Register = 0xFFFFFFFF
R15 Program Counter = 0x00000000
xPSR Program Status Register = 0x21000000

Flags:
N - Negation = 0
Z - Zero = 0
C - Carry = 1
O - Overflow = 0
S - Saturation = 0

Interruption:
Execution = UsageFault
00007107H - Undefined instruction UsageFault
  
```

Main experimental parameters

- Position** of the injection antenna
- Electric parameters** of the pulse
- Injection time** of the pulse
- Executed code** on the microcontroller



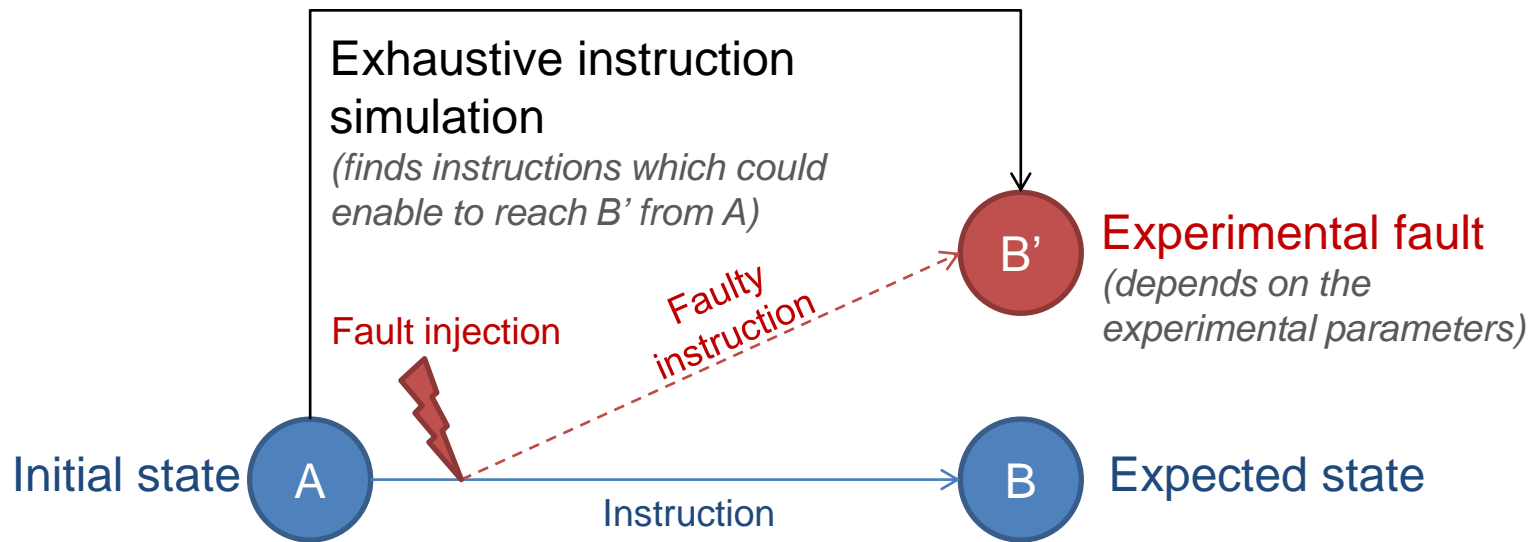
I. Experimental setup

 **II. General approach**

III. Study of the injection parameters

IV. Register-transfer level fault model

V. Conclusion



Output pieces of data	Detail
R0 to R12	General-purpose registers
R13 (SP)	Stack pointer
R14 (LR)	Link register
R15 (PC)	Program counter
XPSR	Program Status Register <ul style="list-style-type: none"> - Flags - Details about the triggered interruptions - Details about the execution mode
Result	Memory address that contains the calculation's output

Instruction skip simulation

IT	Ligne assembleur	Detail IT	Adresse	R0	R1	R2	R3	R4	R5
Thread_mode		None	None	0x2000040c	0x2	0x20000421	0x3	0x2	0x400
Thread_mode	0x08000224 6803 LDR r3,[r0,#0x00]	None	None	0x2000040c	0x2	0x20000421	0x7[FAUTE]	0x2	0x400
Thread_mode	0x08000226 4423 ADD r3,r3,r4	None	None	0x2000040c	0x2	0x20000421	0x2[FAUTE]	0x2	0x400
Thread_mode	0x08000228 6003 STR r3,[r0,#0x00]	None	None	0x2000040c	0x2	0x20000421	0x2[FAUTE]	0x2	0x400
Thread_mode	0x0800022A F1010101 ADD r1,r1,#0x01	None	None	0x2000040c	0x2	0x20000421	0x4[FAUTE]	0x2	0x400
Thread_mode	0x0800022E 2902 CMP r1,#0x02	None	None	0x2000040c	0x1[FAUTE]	0x20000421	0x1[FAUTE]	0x1[FAUTE]	0x400
Thread_mode	0x08000230 DBF6 BLT 0x08000220	None	None	0x2000040c	0x2	0x20000421	0x3	0x2	0x400
Thread_mode	0x08000220 F8524021 LDR r4,[r2,r1,LSL #2]	None	None	0x2000040c	0x2	0x20000421	0x2[FAUTE]	0x1[FAUTE]	0x400
Thread_mode	0x08000224 6803 LDR r3,[r0,#0x00]	None	None	0x2000040c	0x2	0x20000421	0x3	0x2	0x400
Thread mode	0x08000226 4423 ADD r3,r3,r4	None	None	0x2000040c	0x2	0x20000421	0x1[FAUTE]	0x2	0x400

Experimental measurements

Instant	Exec	IT	Ligne assembleur	Detail IT	Adres	R0	R1	R2	R3	R4	R5	R6	R7
37600	1	Thread_mode	None	None	None	0x2000040c	0x2	0x20000421	0x3	0x2	0x40010c10	0x40010c14	0x100
37600	2	Thread_mode	None	None	None	0x2000040c	0x2	0x20000421	0x3	0x2	0x40010c10	0x40010c14	0x100
37600	3	Thread_mode	None	None	None	0x2000040c	0x2	0x20000421	0x3	0x2	0x40010c10	0x40010c14	0x100
37600	4	Thread_mode	None	None	None	0x2000040c	0x2	0x20000421	0x3	0x2	0x40010c10	0x40010c14	0x100
37600	5	Thread mode	None	None	None	0x2000040c	0x2	0x20000421	0x2[FAUTE]	0x1[FAUTE]	0x40010c10	0x40010c14	0x100

- Two lines are equal ⇔ R0 to R12 + XPSR + result + SP + PC are equal

Example of simulation of a 16-bit instruction replacement

Execution	IT	Ligne assembleur	Detail IT	Adresse	R0	R1	R2	R3	R4	R5	R6
0	Thread_mode		None	None	0x200004f0	0x2	0x20000421	0x3	0x2	0x40010c10	0x40010c14
0	Thread_mod	0x20000B0F 0000 MOVs r0,r0	None	None	0x200004f0	0x1[FAUTE]	0x20000421	0x1[FAUTE]	0x1[FAUTE]	0x40010c10	0x40010c14
1	Thread_mod	0x20000B0F 0001 MOVs r1,r0	None	None	0x200004f0	0x200004f0	0x20000421	0x1[FAUTE]	0x1[FAUTE]	0x40010c10	0x40010c14
2	Thread_mod	0x20000B0F 0002 MOVs r2,r0	None	None	0x200004f0	0x1[FAUTE]	0x200004f0	0x1[FAUTE]	0x1[FAUTE]	0x40010c10	0x40010c14
3	Thread_mod	0x20000B0F 0003 MOVs r3,r0	None	None	0x200004f0	0x1[FAUTE]	0x20000421	0x200004f0	0x1[FAUTE]	0x40010c10	0x40010c14
4	Thread_mod	0x20000B0F 0004 MOVs r4,r0	None	None	0x200004f0	0x1[FAUTE]	0x20000421	0x1[FAUTE]	0x200004f0	0x40010c10	0x40010c14
5	Thread_mod	0x20000B0F 0005 MOVs r5,r0	None	None	0x200004f0	0x1[FAUTE]	0x20000421	0x1[FAUTE]	0x1[FAUTE]	0x200004f0	0x40010c14
6	Thread_mod	0x20000B0F 0006 MOVs r6,r0	None	None	0x200004f0	0x1[FAUTE]	0x20000421	0x1[FAUTE]	0x1[FAUTE]	0x40010c10	0x200004f0
7	Thread_mod	0x20000B0F 0007 MOVs r7,r0	None	None	0x200004f0	0x1[FAUTE]	0x20000421	0x1[FAUTE]	0x1[FAUTE]	0x40010c10	0x40010c14
8	Thread_mod	0x20000B0F 0008 MOVs r0,r1	None	None	0x1[FAUTE]	0x1[FAUTE]	0x20000421	0x1[FAUTE]	0x1[FAUTE]	0x40010c10	0x40010c14
9	Thread_mod	0x20000B0F 0009 MOVs r1,r1	None	None	0x200004f0	0x1[FAUTE]	0x20000421	0x1[FAUTE]	0x1[FAUTE]	0x40010c10	0x40010c14
10	Thread_mod	0x20000B0F 000A MOVs r2,r1	None	None	0x200004f0	0x1[FAUTE]	0x1[FAUTE]	0x1[FAUTE]	0x1[FAUTE]	0x40010c10	0x40010c14
11	Thread_mod	0x20000B0F 000B MOVs r3,r1	None	None	0x200004f0	0x1[FAUTE]	0x20000421	0x1[FAUTE]	0x1[FAUTE]	0x40010c10	0x40010c14
12	Thread_mod	0x20000B0F 000C MOVs r4,r1	None	None	0x200004f0	0x1[FAUTE]	0x20000421	0x1[FAUTE]	0x1[FAUTE]	0x40010c10	0x40010c14
13	Thread_mod	0x20000B0F 000D MOVs r5,r1	None	None	0x200004f0	0x1[FAUTE]	0x20000421	0x1[FAUTE]	0x1[FAUTE]	0x1[FAUTE]	0x40010c14
14	Thread_mod	0x20000B0F 000E MOVs r6,r1	None	None	0x200004f0	0x1[FAUTE]	0x20000421	0x1[FAUTE]	0x1[FAUTE]	0x40010c10	0x1[FAUTE]

- Very long for an exhaustive simulation over the whole instruction set
- Two lines are **equal** \Leftrightarrow **R0 to R12 + XPSR + result** are equal

I. Experimental setup

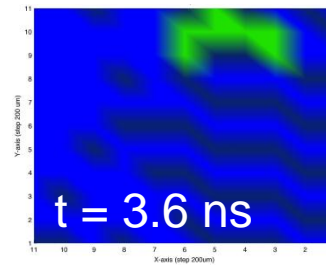
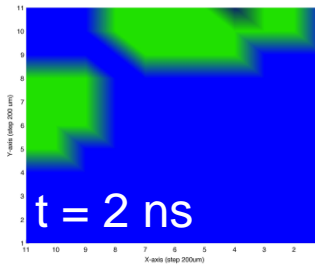
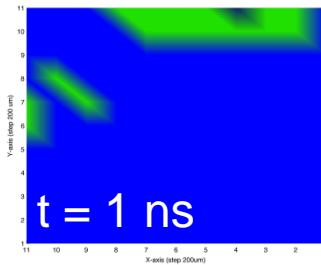
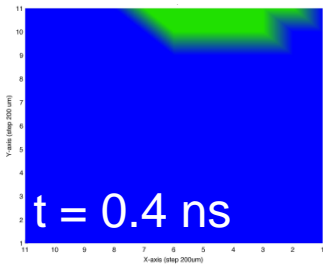
II. General approach

 **III. Study of the injection parameters**

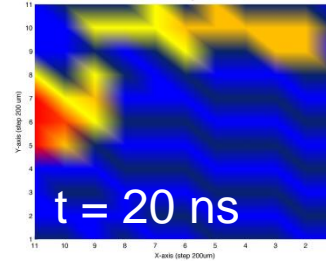
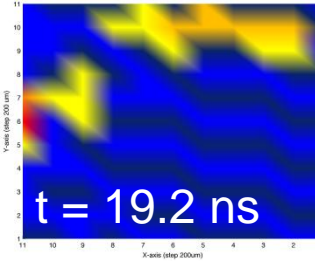
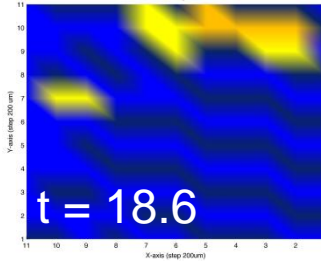
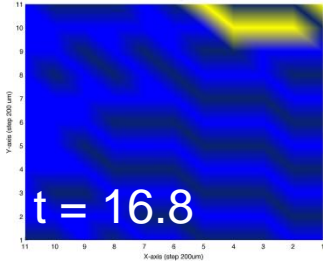
IV. Register-transfer level fault model

V. Conclusion

- **Green** : hardware interrupts have been triggered
- **Red** : faults on the output value have been obtained

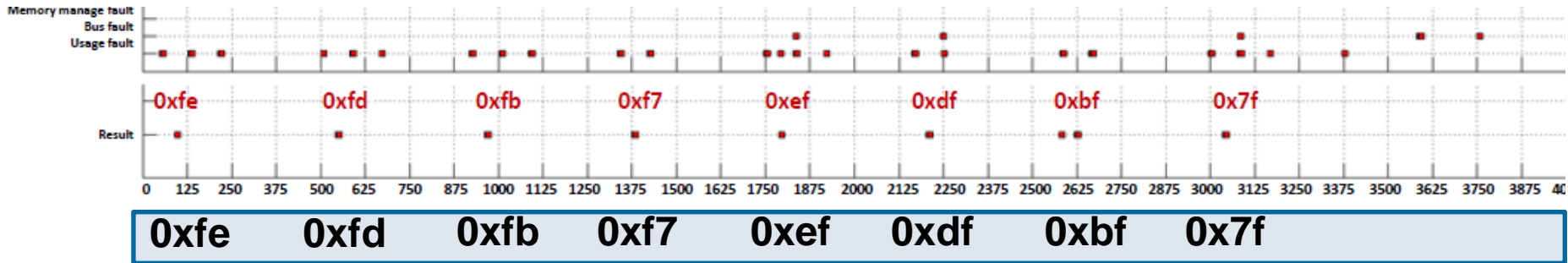


Frequency **56 MHz** – Pulse width **10 ns** – Pulse voltage **190V** – Period **17ns**



- **Target instruction** : single LOAD instruction that loads 0x12345678 into R8
- **20 ns** time interval, by steps of **200 ps** - **3 mm** square, by steps of **200 μm**
- **Variable increase** of the Hamming weight of the loaded piece of data
- **No fault** on other registers than **R8** (except for very few faults on R0)

Example of temporal cartography on an addition loop



Observations:

- One power of two has not been added
- BusFault or UsageFault interrupts

➔ Does our fault injection have an effect on the data flow or the control flow ?

Test program:

loop to sum the elements of an array that contains eight powers of two
3.5 μ s, by steps of 200 ps

Expected result: 0xFF

```
addition_loop:
ldr r4, [r2,r1, lsl #2] ; r4 = array[i]
ldr r3, [r0,#0] ; r3 = result
add r3, r4 ; r3 = r3 + r4
str r3, [r0,#0] ; result = r3
add r1, r1, #1 ; r1 = r1 + 1
cmp r1, #8 ; r1 == 8 ?
blt addition_loop
```

LDR R4, PC#44 with 0x12345678 at the address PC#44

Pulse voltage	Output value	Occurrence rate
172V	1234 5678	100 %
174V	9 234 5678	73 %
176V	FE 34 5678	30 %
178V	FFF 4 5678	53 %
180V	FFFD 5678	50 %
182V	FFFF 7F 78	46 %
184V	FFFF FFFB	40 %
186V	FFFF FFFF	100 %

- Simulation : corresponds to **no instruction replacement**
- Looks like a **set at 1** fault model on the **Flash memory data transfers**
- Possible **precharge of the data bus** on this architecture

I. Experimental setup

II. General approach

III. Study of the injection parameters

 **IV. Register-transfer level fault model**

V. Conclusion

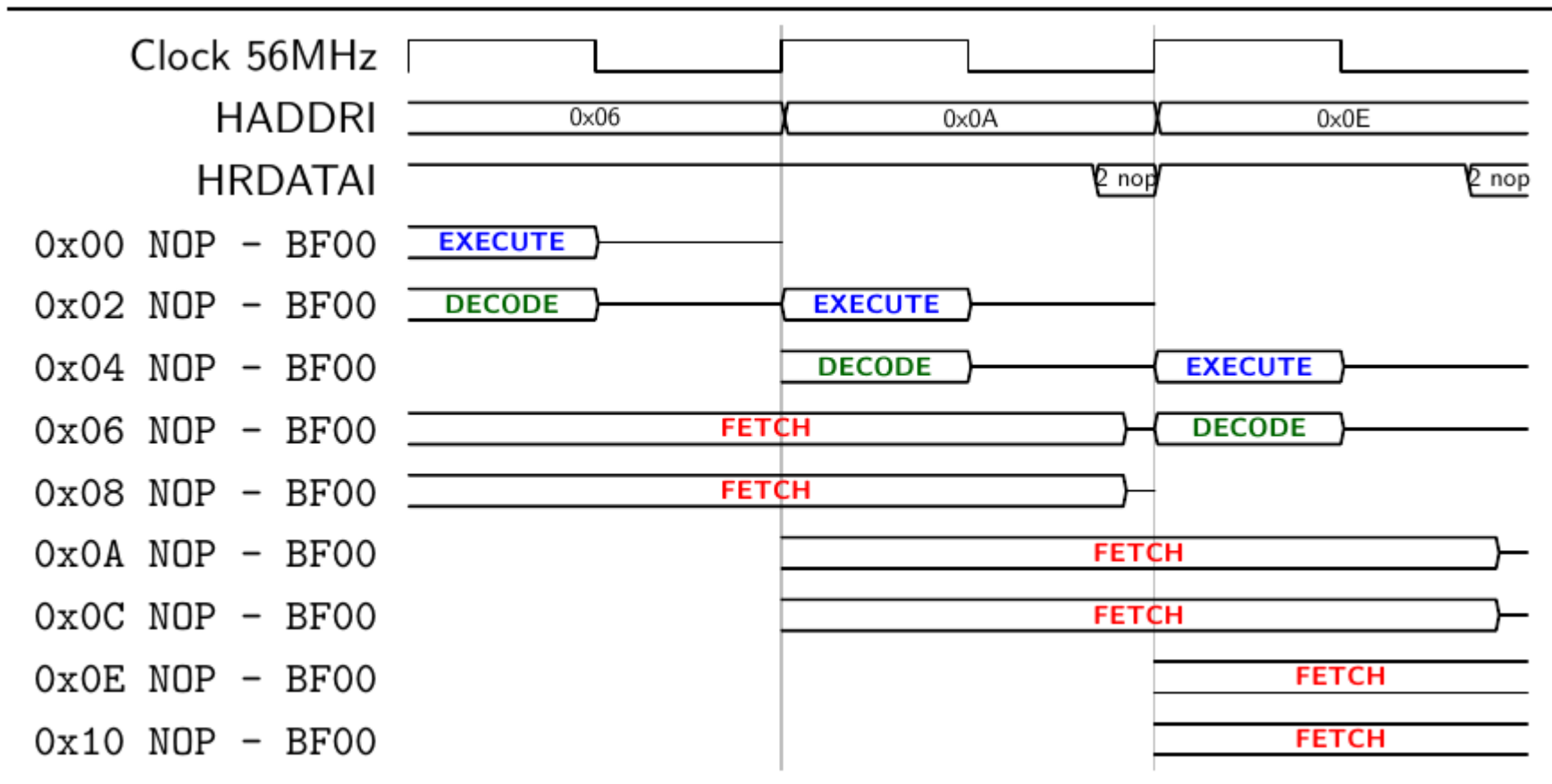
- Experiments with a sequence of **NOP** (BF 00)
- **Four kinds of faults**
 - Fault on **R7**
 - The program does not stop
 - **UsageFault** exceptions (Invalid Instruction / No Coprocessor)
 - Fault on **R0**
- Sometimes a modification of the **number of executed cycles**
- Simulation on the ISA: some instructions can explain the results
- **Some faults only equivalent to a STR R0, [R0, #0] instruction**

NOP - BF00 1011 1111 0000 0000
NOP - BF00 1011 1111 0000 0000

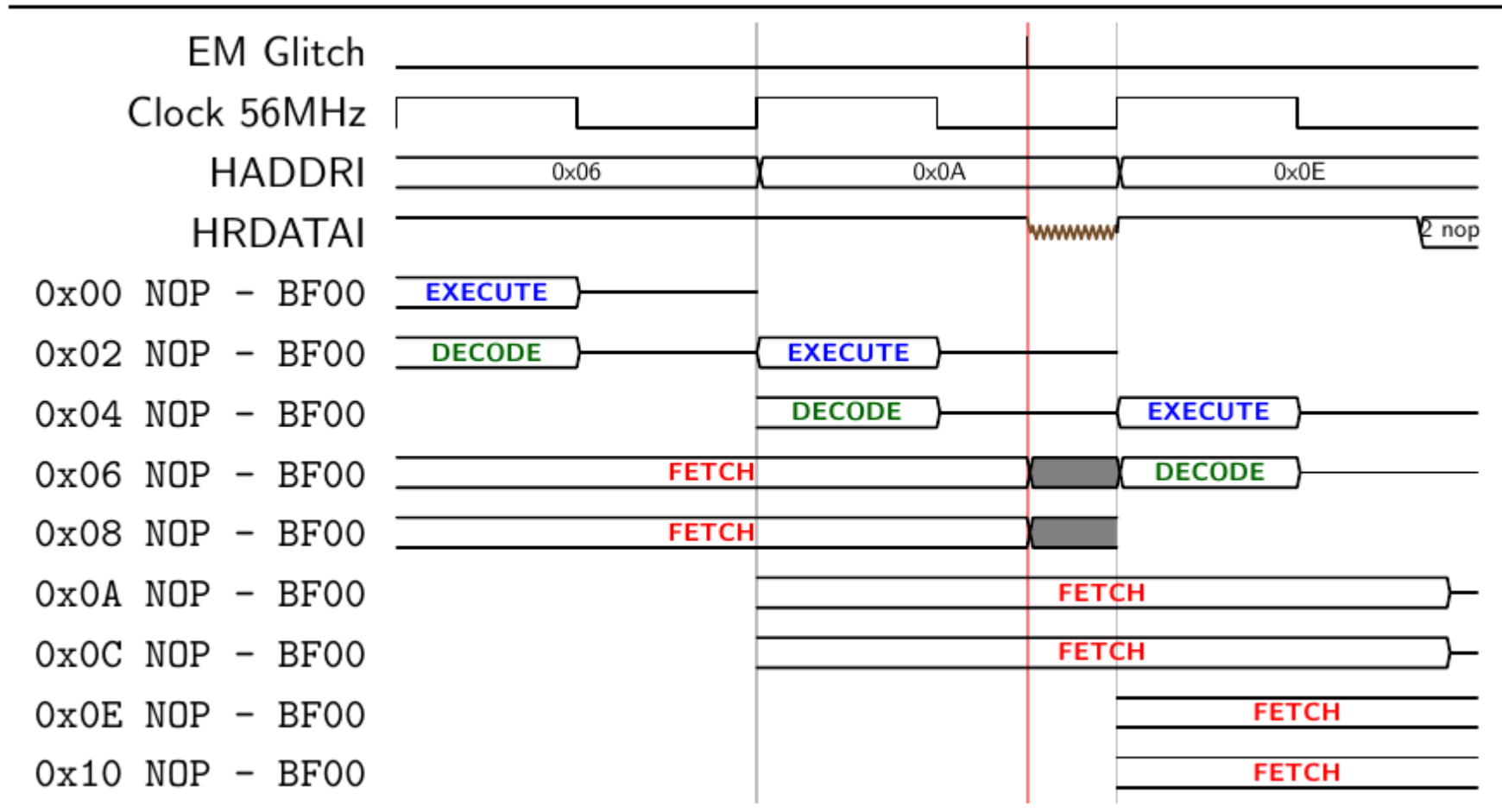


STR R0, [R0, #0] - 6000 0110 0000 0000 0000
NOP - BF00 1011 1111 0000 0000

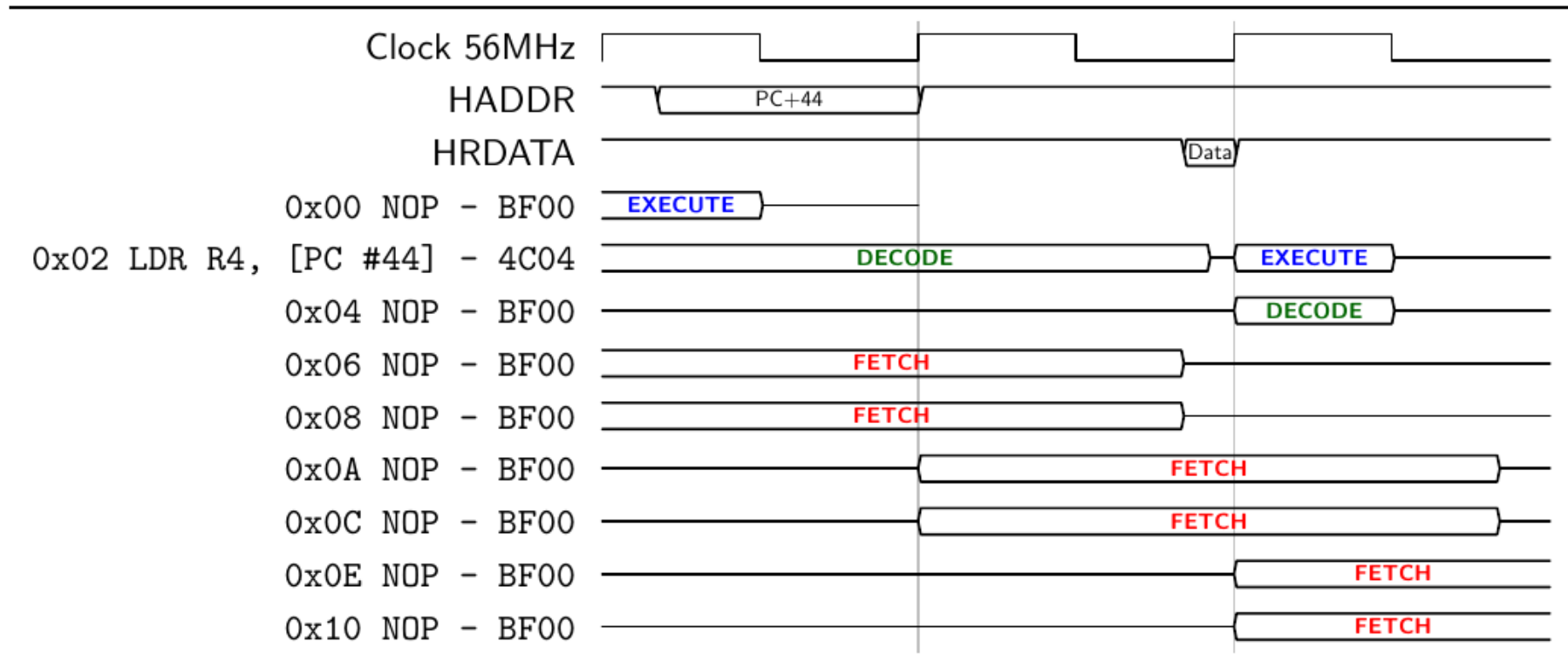
Normal behaviour



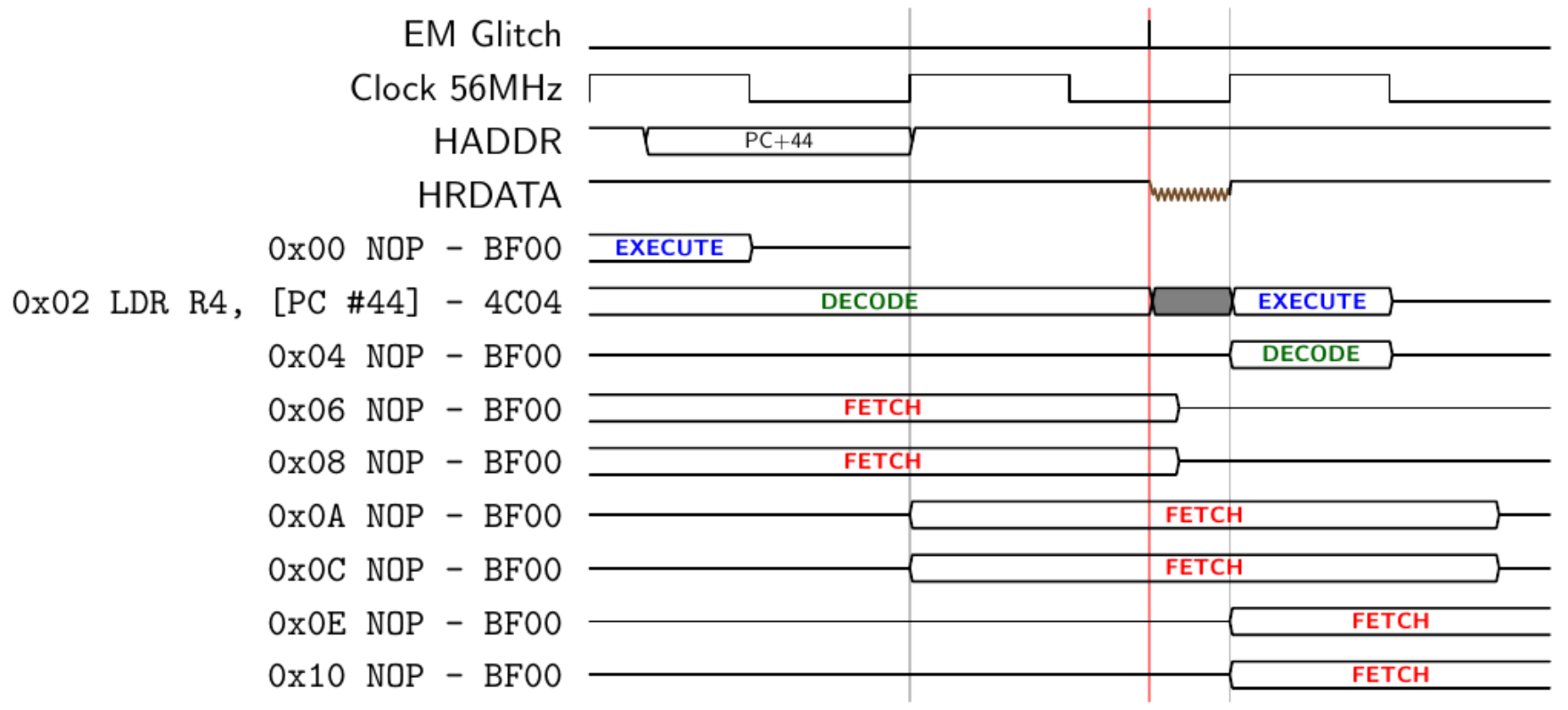
With an electromagnetic fault injection



Normal behaviour



With an electromagnetic fault injection



➔ Possible to fault the **transfers from the Flash memory**

Consequences regarding the **instruction flow**

- Instructions **replacements**
- Instruction **skips under certain conditions** (~ 20-30% of time)
- Some instructions may be **more sensitive than others**
- Some registers seem to be **more sensitive than others**

Consequences regarding the **data flow**

- Corruption of the `LOAD` instructions from the Flash memory (encryption keys,...)
- Some metastability phenomena, but deterministic under some conditions
- Faulty values with higher Hamming weight (on this architecture)

I. Experimental setup

II. General approach

III. Study of the injection parameters

IV. Register-transfer level fault model

 **V. Conclusion**

- A first attempt of **fault model** for EM fault injection on a 32-bit μC
- Corruption of the **transfers from the Flash memory** on the buses
- The obtained effects seem **very similar** to the ones obtained with **clock glitches or other fault injection means**
- **Similar effects** obtained previously on a very different architecture
(Atmel AVR ATmega128 8-bit microcontroller)
- Possibility to perform **instruction skips** under some specific conditions

Perspectives

- Use more advanced debug techniques to understand better instruction replacements
- Define a higher-level fault model that can be used for theoretical attacks

Any questions ?