ElectroMagnetic Fault Injection Characterization on ARM Cortex-A9

George Thessalonikefs

George.Thessalonikefs@os3.nl

University of Amsterdam

February 5, 2014

Introduction

Hardware Fault Injection

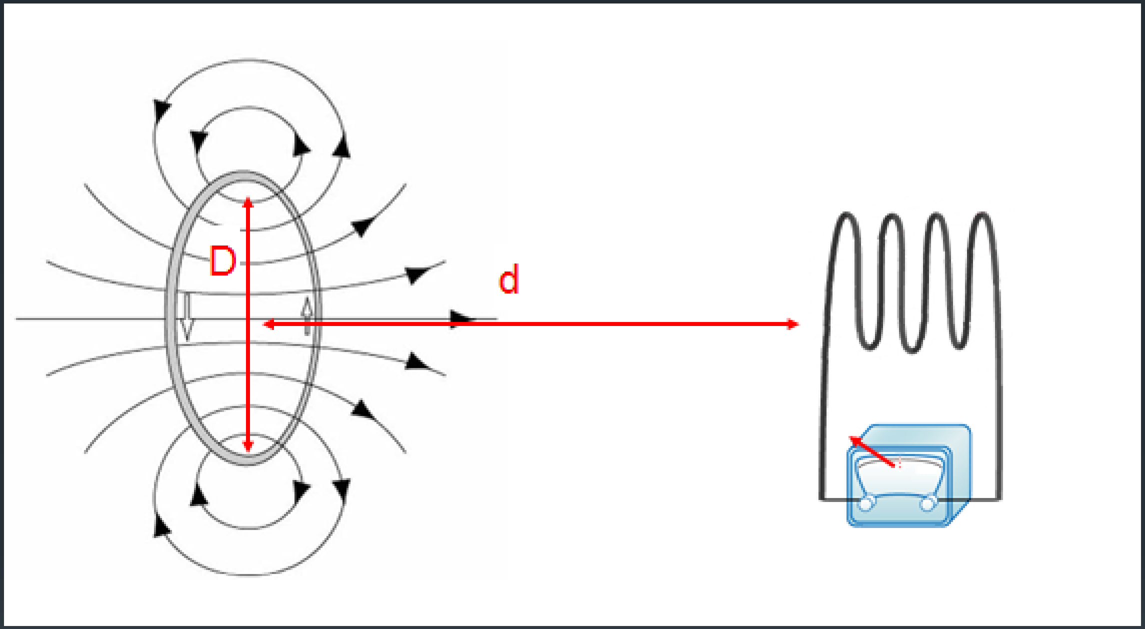
Induce faults to hardware through side channels:

- Clock
- Power supply
- Electromagnetic radiation
- Light
- Temperature

Goals

- Change behavior
- Change data

ElectroMagnetic Fault Injection



For inducing a significant voltage spike, distance d < D

Source: Riscure

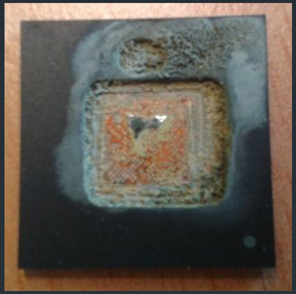
EMFI vs VCC & Optical FI

No preparation needed for the target

- VCC FI : Need to work with capacitors to glitch the core voltage line
- Optical FI : Decapsulation of the chip

Countermeasures for:

- VCC FI: Glitch sensors
- Optical FI: Light sensors


Picture:
Decapsulated chip

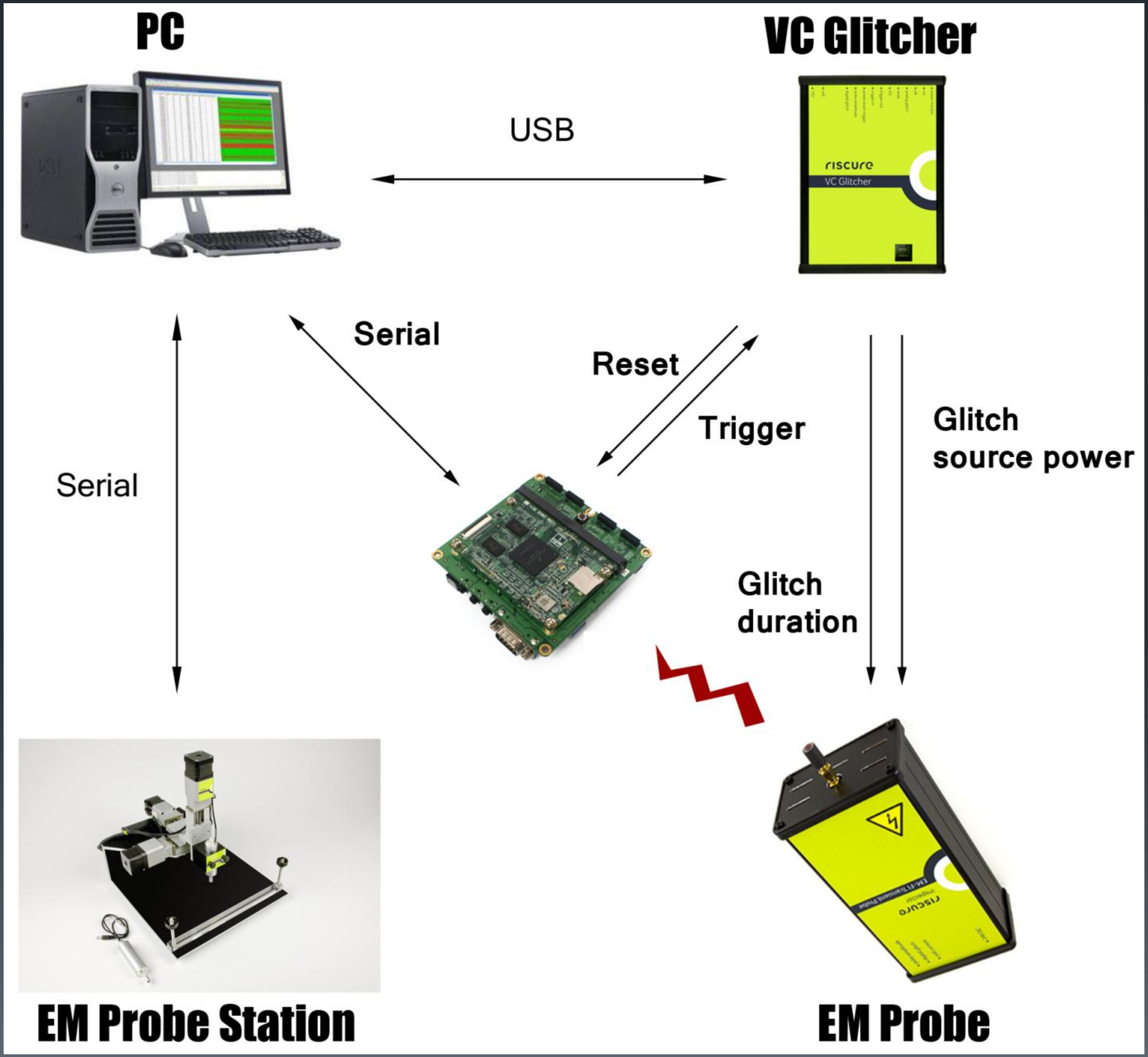
EMFI in action

http://www.youtube.com/watch?v=dew0KD_-ypw

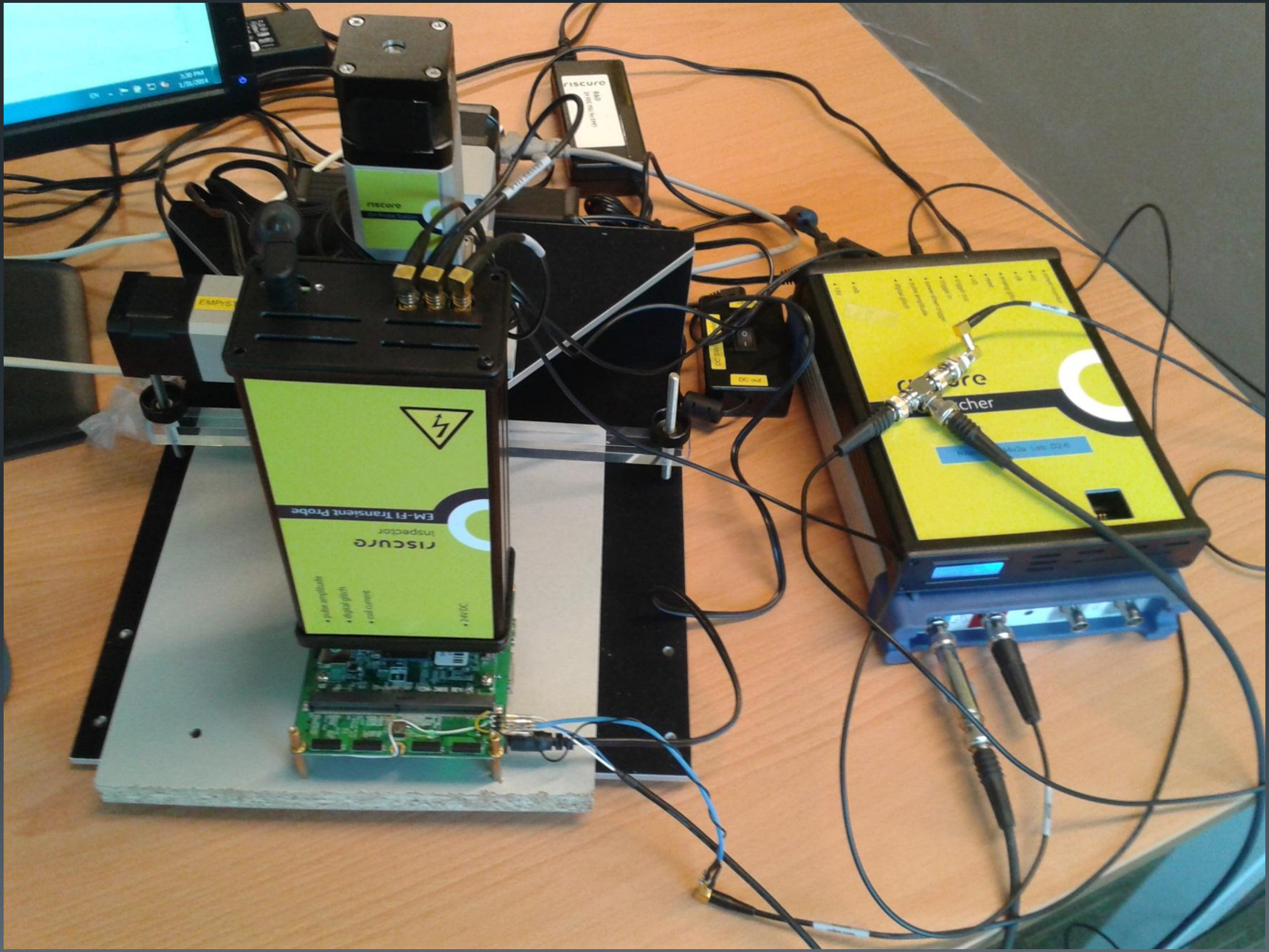
Research question

What are the effects of ElectroMagnetic Fault Injection (EMFI) on embedded chips?

Setup


PC
VC Glitcher
USB
Serial
Reset
Trigger
Glitch source power
Serial
Glitch duration
EM Probe Station
EM Probe

Setup


riscure
EM-FI Transient Probe
riscure inspector
pulse amplitude
digital glitch
coil current
24V DC
EMPrST
DC out

Target

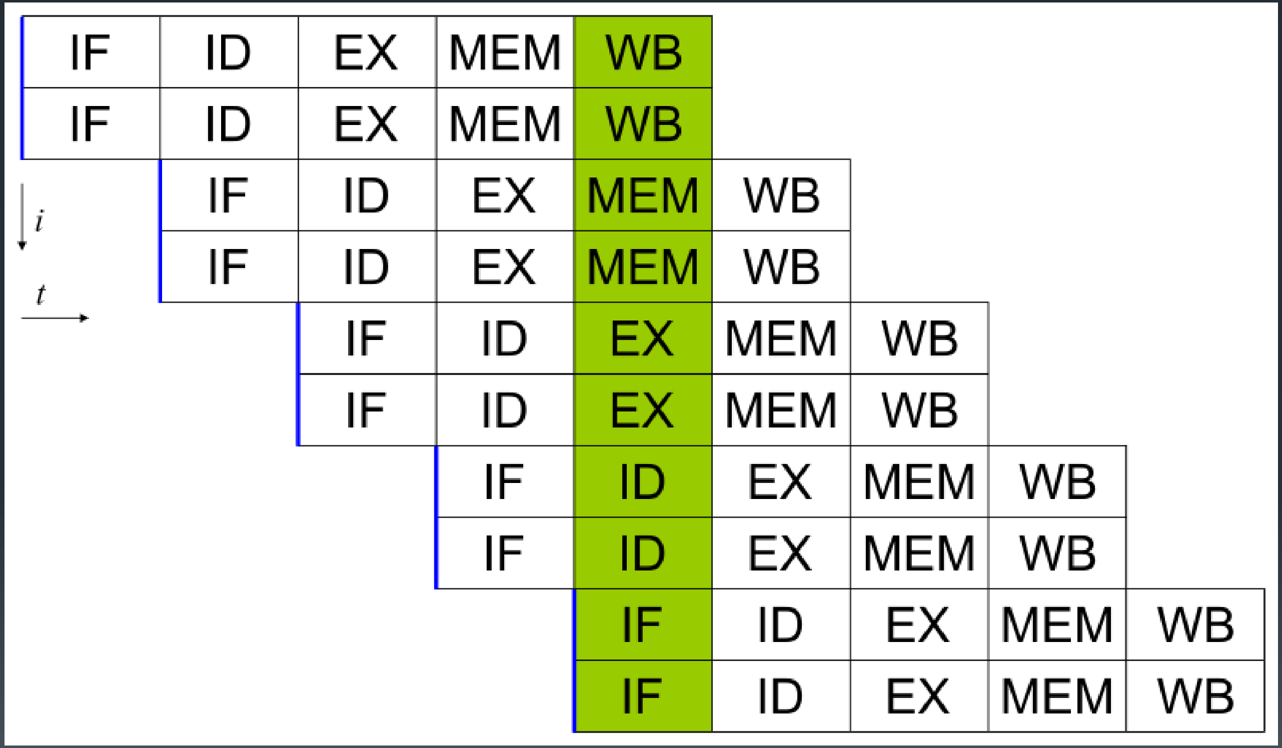
Freescale i.MX6 Solo Processor
Using an ARM Cortex-A9 Single Core

Specifications:
- 32-bit processor
- ARMv7 architecture based on RISC
- Clock speed of 792 MHz:
  1,26 ns/cycle
- Pipeline
  - Dual-issue superscalar
  - Out-of-order
  - Speculative execution
  - 8-stage

Wandboard
SOLO

Dual-issue superscalar Pipeline

Example:

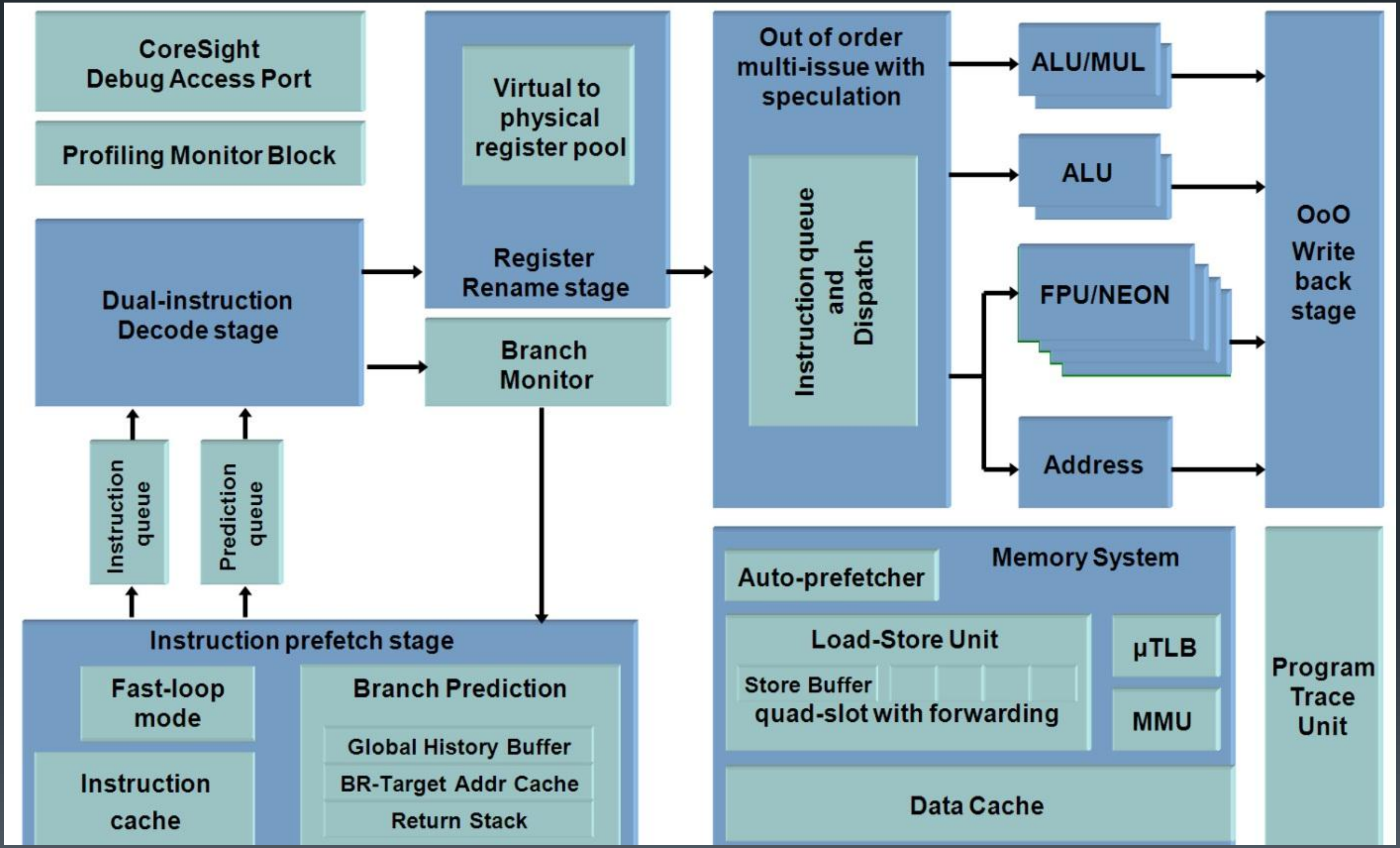


IF	ID	EX	MEM	WB				
IF	ID	EX	MEM	WB				
	IF	ID	EX	MEM	WB			
	IF	ID	EX	MEM	WB			
		IF	ID	EX	MEM	WB		
		IF	ID	EX	MEM	WB		
			IF	ID	EX	MEM	WB	
			IF	ID	EX	MEM	WB	
				IF	ID	EX	MEM	WB
				IF	ID	EX	MEM	WB

IF:	Instruction Fetch
ID:	Instruction Decode
EX:	Execute
MEM:	Memory access
WB:	Write Back

http://en.wikipedia.org/wiki/File:Superscalarpipeline.svg

ARM Cortex-A9 Pipeline



http://www.arm.com/images/A9-Pipeline-hres.jpg

Code instrumentation

- Initialize registers to known values
- Trigger ON
- Critical area code
- Trigger OFF
- Print results

Code was written in ARM assembly to avoid C compiler’s optimization

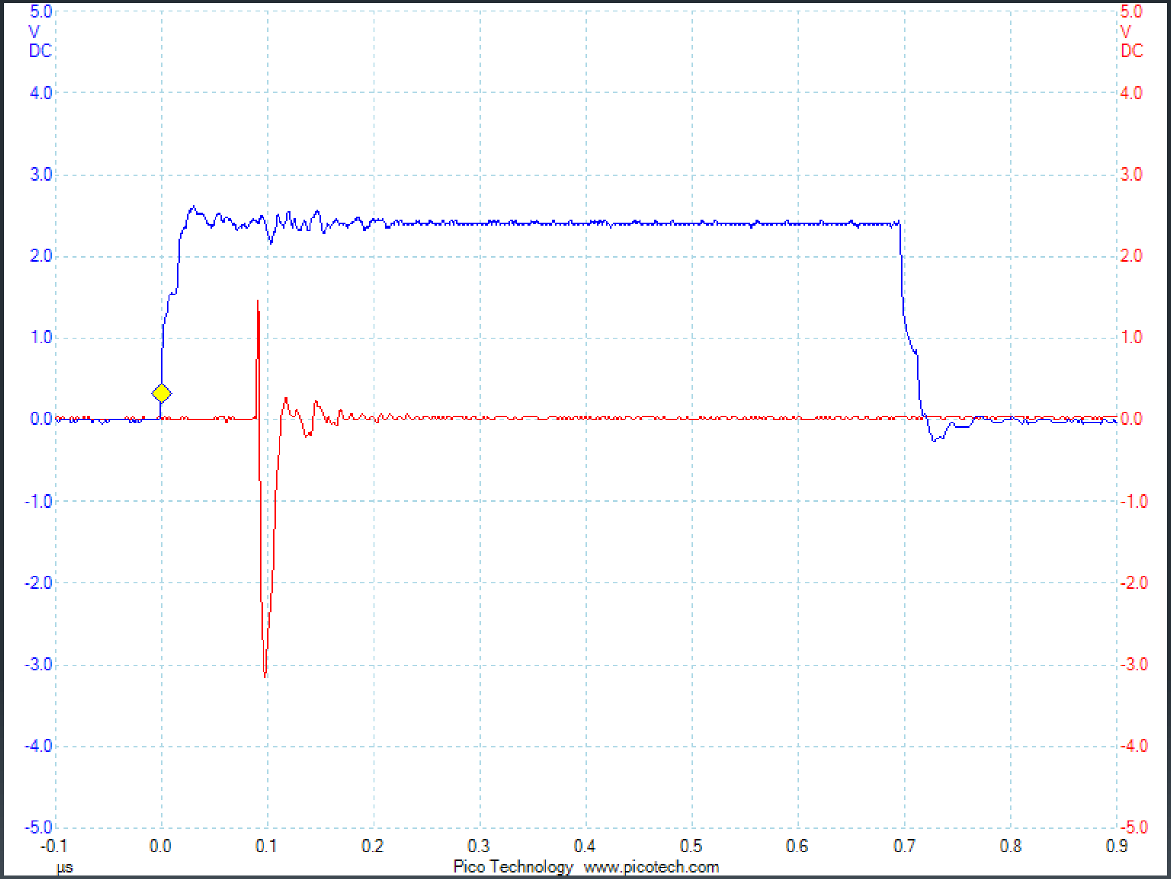
Critical area code

- R0 initialized to 0xFFFFFFFF
- R1 initialized to 0x00000001
- Unrolled loop of 32 pairs of instructions:
  - Logical operation
  - Shift R1 1-bit to the left

Logical operations:

- BIC (BIt Clear)
- EOR (Exclusive OR)

Visualization of fault injection



Blue line: Trigger signal
Red line: Coil current

Correct Output

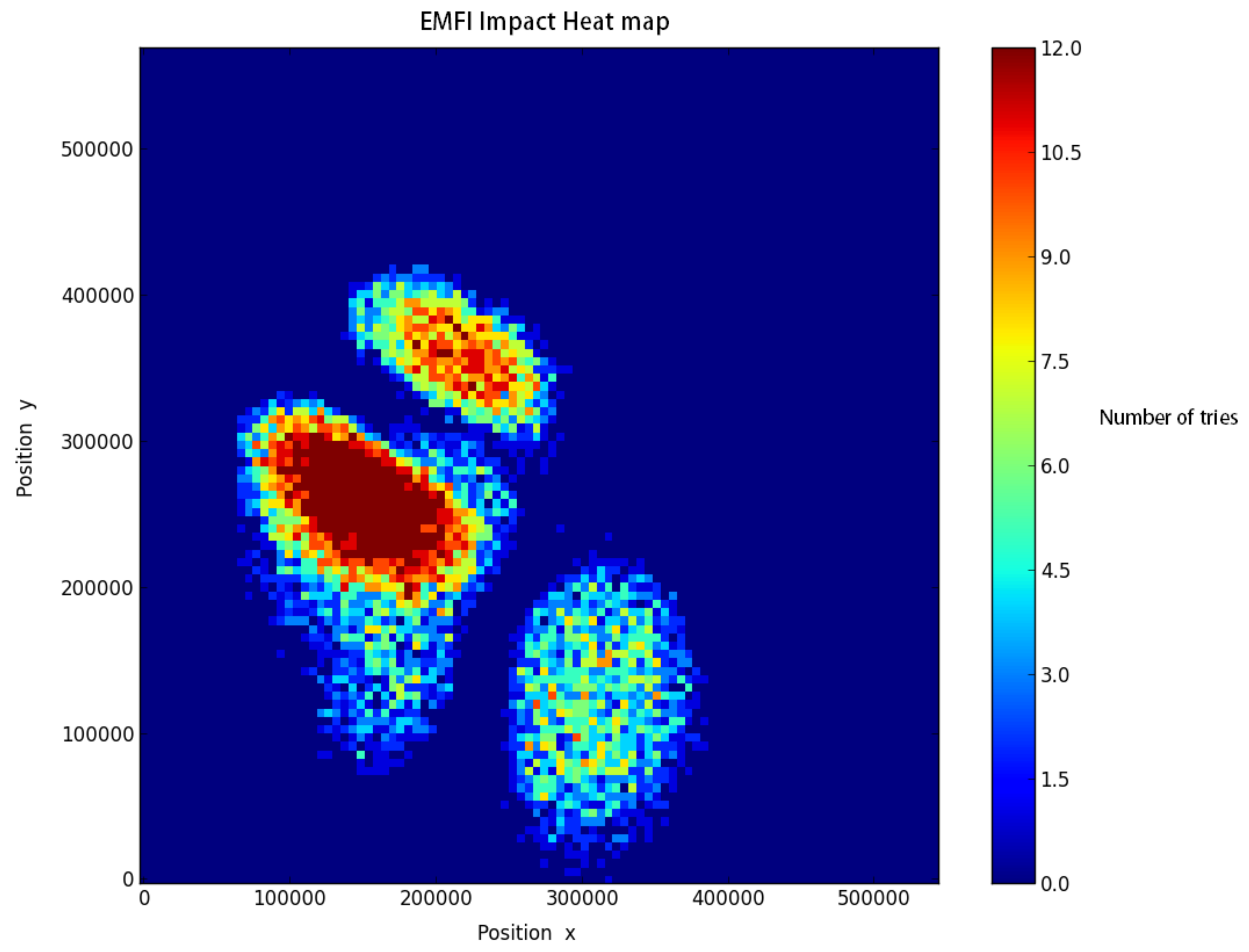
BIC version

```
R0: 00000000   R1: 80000000   R2: FFFFFFFF   R3: 020B4000
R4: A54444A5   R5: A55555A5   R6: A56666A5  …….
```

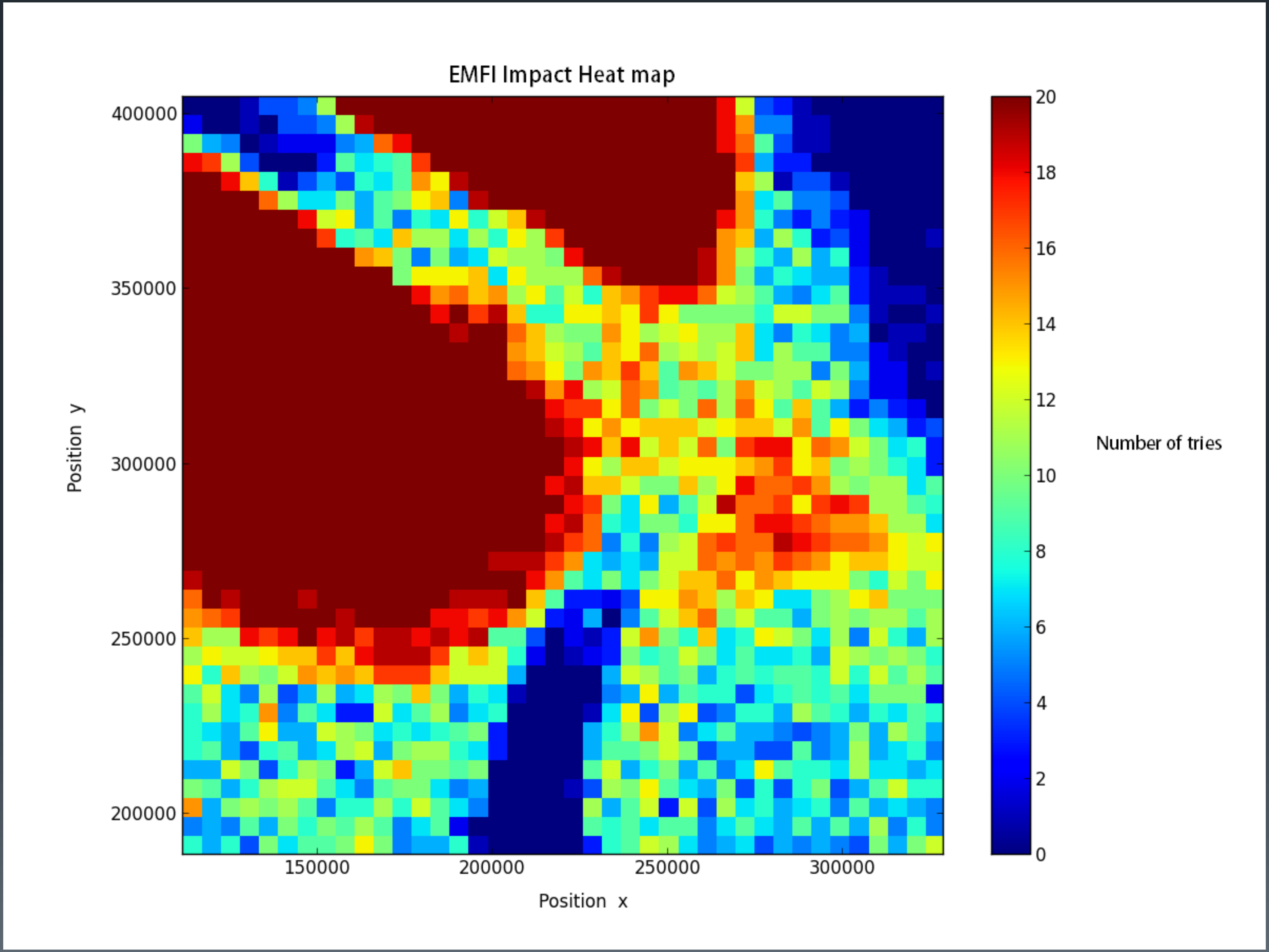
EOR version

```
R0: 00000000   R1: 80000000   R2: FFFFFFFF   R3: 020B4000
R4: A54444A5   R5: A55555A5   R6: A56666A5  …….
```

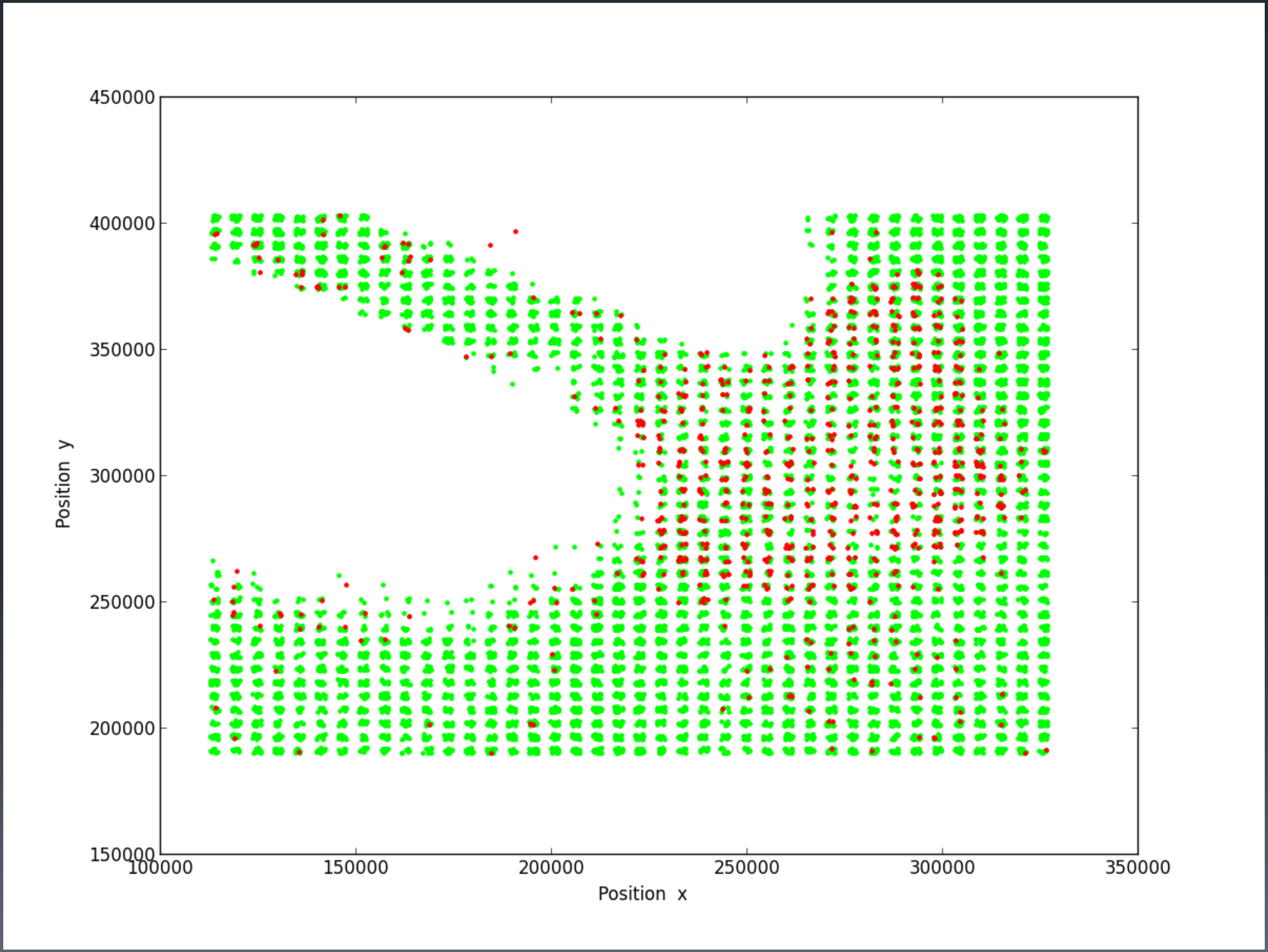
Full chip detailed scan



Die detailed scan



Glitches with desired results



Glitch results

Logical operation not executed

- Suspects:
  - Instruction Fetch
  - Instruction Execution
  - Write back

Expected result:

R0: 00000000 R1: 80000000

Glitched result:

R0: 00000001 R1: 80000000

Glitch results

Logical shift not executed

- Suspects:
  - Instruction Fetch
  - Instruction Execution
  - Write back

Expected result:

R0: 00000000   R1: 80000000

Glitched result:

R0: 80000000   R1: 40000000

Glitch results

Logical operation and Logical shift not executed

- Suspects:
  - Instruction Fetch
  - Instruction Execution
  - Write back

Expected result:

R0: 00000000 R1: 80000000

Glitched result:

R0: 80000001 R1: 40000000

Glitch results

Data abort exception due to unaligned access

- Suspects:
  - PC register glitched
  - Stack corrupted

Glitch results

Prefetch abort exception due to non-existing memory regions

- Suspects:
  - PC register glitched
  - Stack corrupted

Conclusion

- Edges of the chip more sensitive than the top of the die
- No unused register corruptions
- Difficult to constantly have the same results with EMFI

Future work

- Comparison of full area scans of the package between ALU and memory instructions
- Research the impact of EMFI on jump commands

Thank you

Questions?