Fault attacks on System On Chip

Thomas TROUCHKINE [1 3] Guillaume BOUFFARD [1] David EL-BAZE [1]
Jessy CLÉDIÈRE [2 3]
[1]ANSSI - Hardware Security Labs

[2]CEA LETI

[3]Doctoral School EEATS

September 20, 2018

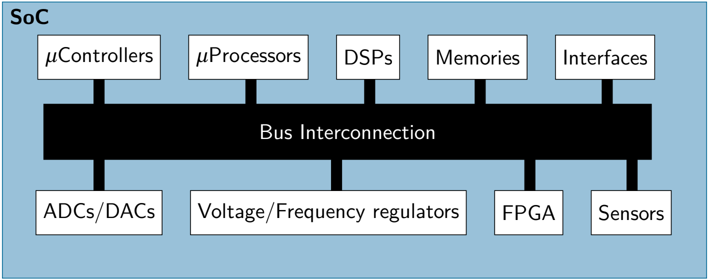
Introduction - Context

Based on a full featured SoC

- Complex SoC
- Designed for performance
- Adding TEE[1] for software security
- Used for sensitive services (payment, healthcare...)

[1] **Trusted Environment Execution**

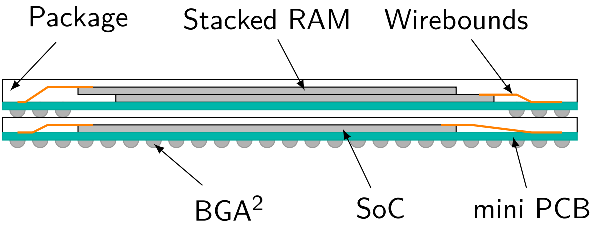
Introduction - What is a System on Chip ?



- Integrate all components on the same chip
- Reduce power consumption
- Reduce chip size

Introduction - The packaging

Package on package



[2] **Ball Grid Array**

Introduction - The goal

Evaluate the sensibility of complex SoCs against physical attacks

(Get my PhD.)

- **Software to hardware approach**
  - Observe physical perturbation on a program
  - Realize low level debugging to find the underlying cause
  - Conclude about the physical effect induced by the perturbation

But first... state of the art !

Introduction - Known attacks

Injection medium	Physical target	Software target	Software security
Software	RAM	Virtual to physical translation table	Memory partitioning
Glitch voltage	Clock	Key	Cryptography
Laser	Register	Instruction	Secure boot
EM	Bus	Return value	Execution flow integry
	Cache	Program counter	
	MMU	User rights	
	Pipeline		

Introduction - Known attacks

Injection medium	Physical target	Software target	Software security
Software →	RAM →	Virtual to physical translation table →	Memory partitioning
Glitch voltage	Clock	Key	Cryptography
Laser	Register	Instruction	Secure boot
EM	Bus	Return value	Execution flow integry
	Cache	Program counter	
	MMU	User rights	
	Pipeline		

Project Zero attack/Drammer (2015 - 2016) [Vee+16]

Introduction - Known attacks

Injection medium	Physical target	Software target	Software security
Software	RAM	Virtual to physical translation table	Memory partitioning
Glitch voltage	Clock	Key	Cryptography
Laser	Register	Instruction	Secure boot
EM	Bus	Return value	Execution flow integry
	Cache	Program counter	
	MMU	User rights	
	Pipeline		

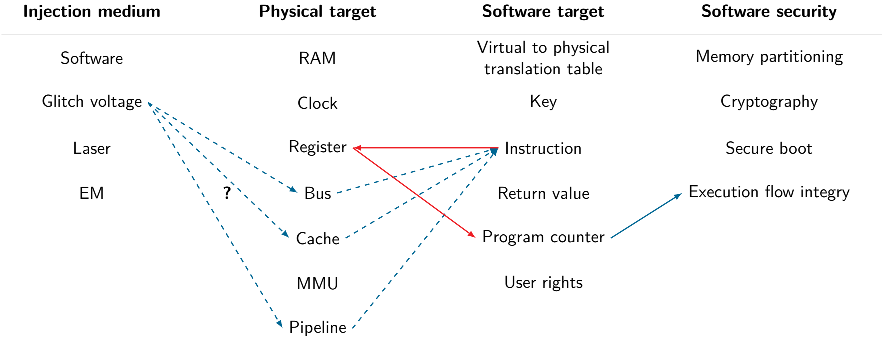
Project Zero NaCl/Rowhammer on TrustZone (2015) [Car17]

Introduction - Known attacks

Injection medium	Physical target	Software target	Software security
Software	RAM	Virtual to physical translation table	Memory partitioning
Glitch voltage	Clock	Key	Cryptography
Laser	Register	Instruction	Secure boot
EM	Bus	Return value	Execution flow integry
	Cache	Program counter	
	MMU	User rights	
	Pipeline		

ClkScrew (2017) [AS17]

Introduction - Known attacks



Controlling PC on ARM (2016) [TSW16]

Introduction - Known attacks

Injection medium	Physical target	Software target	Software security
Software	RAM	Virtual to physical translation table	Memory partitioning
Glitch voltage	Clock	Key	Cryptography
Laser	Register	Instruction	Secure boot
EM	Bus	Return value	Execution flow integry
	Cache	Program counter	
	MMU	User rights	
	Pipeline		

Attack on PS3

Introduction - Known attacks

Injection medium	Physical target	Software target	Software security
Software	RAM	Virtual to physical translation table	Memory partitioning
Glitch voltage	Clock	Key	Cryptography
Laser	Register	Instruction	Secure boot
EM	Bus	Return value	Execution flow integry
	Cache	Program counter	
	MMU	User rights	
	Pipeline		

Attack on Xbox 360 (2015) [Bla15]

Introduction - Known attacks

Injection medium	**Physical target**	**Software target**	**Software security**
Software	RAM	Virtual to physical translation table	Memory partitioning
Glitch voltage	Clock	Key	Cryptography
Laser →	Register	Instruction	Secure boot
EM	Bus	Return value	Execution flow integry
	Cache	Program counter	
	MMU	User rights	
	Pipeline		

Laser induced fault on smartphone (2017) [Vas+17]

Introduction - Attack paths we investigate

EM Fault Injection

- ✓ Non invasive
- ✓ Good resolution
- ✓ Good reproductibility
- ⚠ Never tested on SoC before
- ✗ Uncertain behaviour

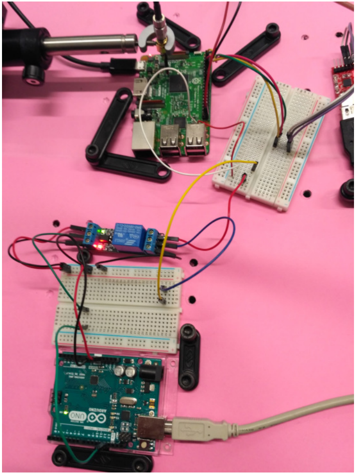
ClkScrew

- ✓ Non invasive
- ✓ Target the TEE
- ⚠ Specific to complex SoCs
- ✗ Need root access
- ✗ Lot of parameters

The experiments - Target

Raspberry Pi 3

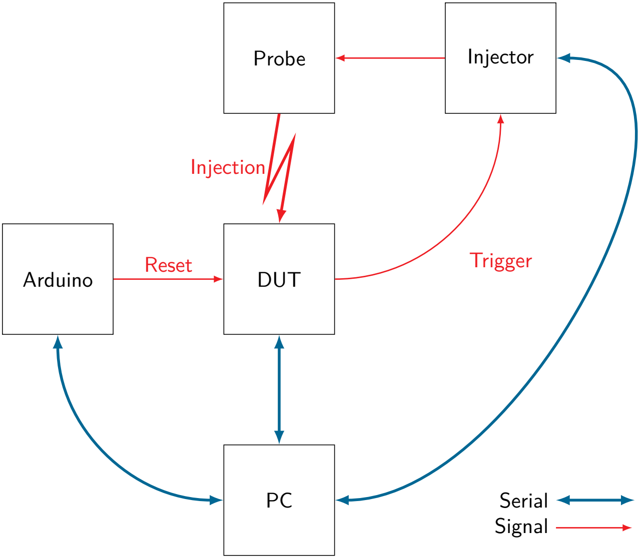
- Broadcom BCM2837
- 4 Cortex A53
- 1.2 GHz



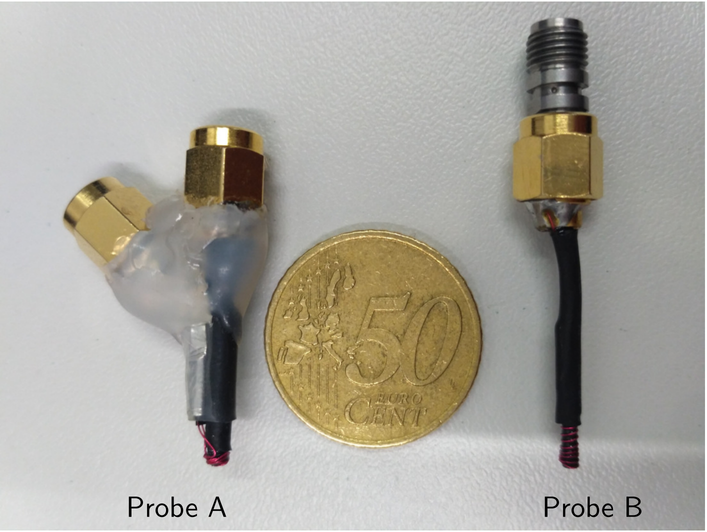
The experiments - Code for test

```c
void loop(void){
  int i = 0;
  int j = 0;
  int cnt = 0;
  trigger_up();
  for(i=0; i<50; i++){
    for(j=0; j<50; j++){
      cnt++;
    }
  }
  trigger_down();
  print("i=%d j=%d cnt=%d\n", i, j, cnt);
}
```

The experiments - The setup



The experiments - The probes

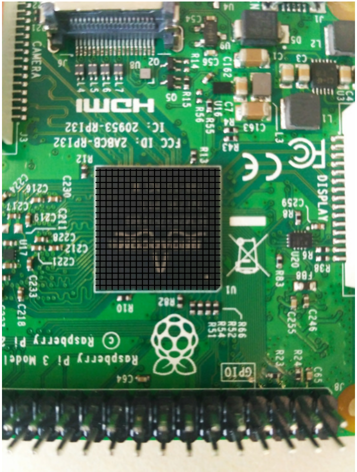


Experiments on Raspberry Pi 3 - BCM2837 cartography

BCM2837 on the Raspberry Pi 3

- 20x20 grid
- 3 different delays
- 6 different powers
- 3 repetitions
- 54 operations/position

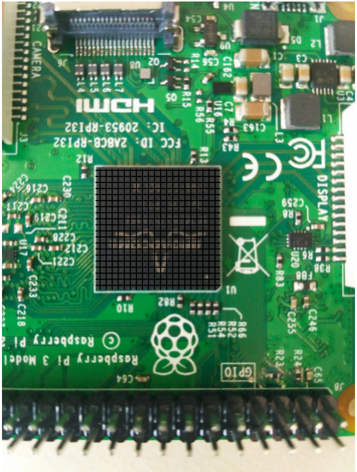
Experiments on Raspberry Pi 3 - BCM2837 cartography

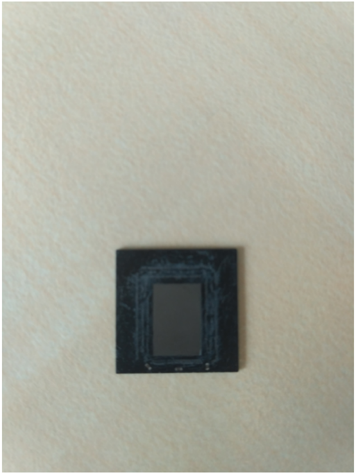


BCM2837 on the Raspberry Pi 3

- 20x20 grid
- 3 different delays
- 6 different powers
- 3 repetitions
- 54 operations/position

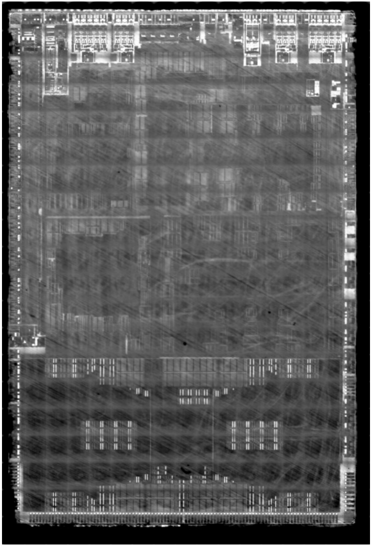
Experiments on Raspberry Pi 3 - BCM2837 cartography



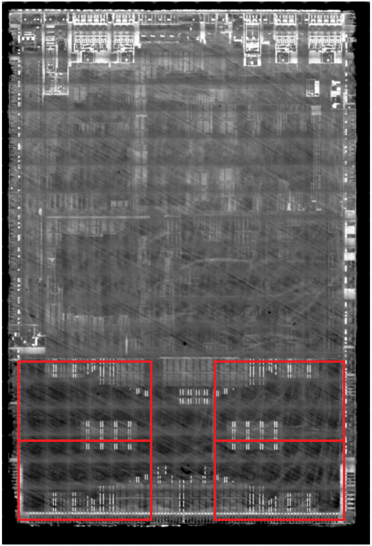


BCM2837 on the Raspberry Pi 3

Experiments on Raspberry Pi 3 - Opened BCM2837

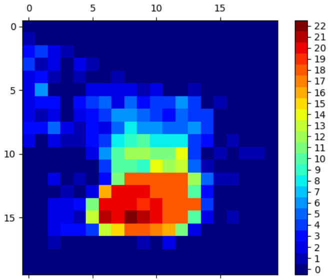


Experiments on Raspberry Pi 3 - Opened BCM2837

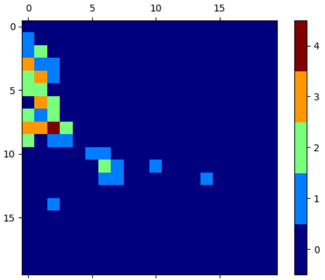


Experiments on Raspberry Pi 3 - EM sensibility of the BCM2837

All effects

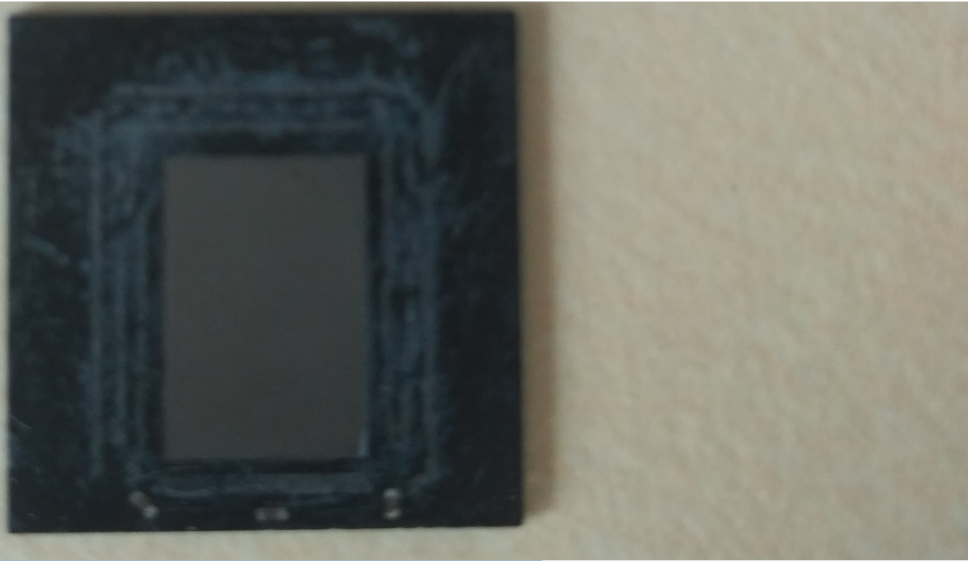


No reboot effects

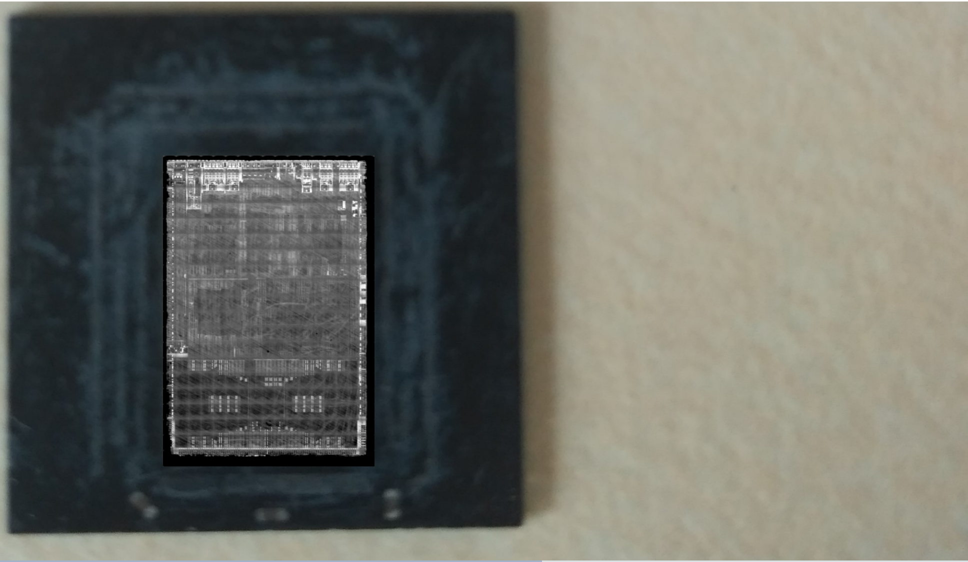


1192 effects for 21600 operations (5.51%)
55 effects without reboot for 1192 operations (4.61%)

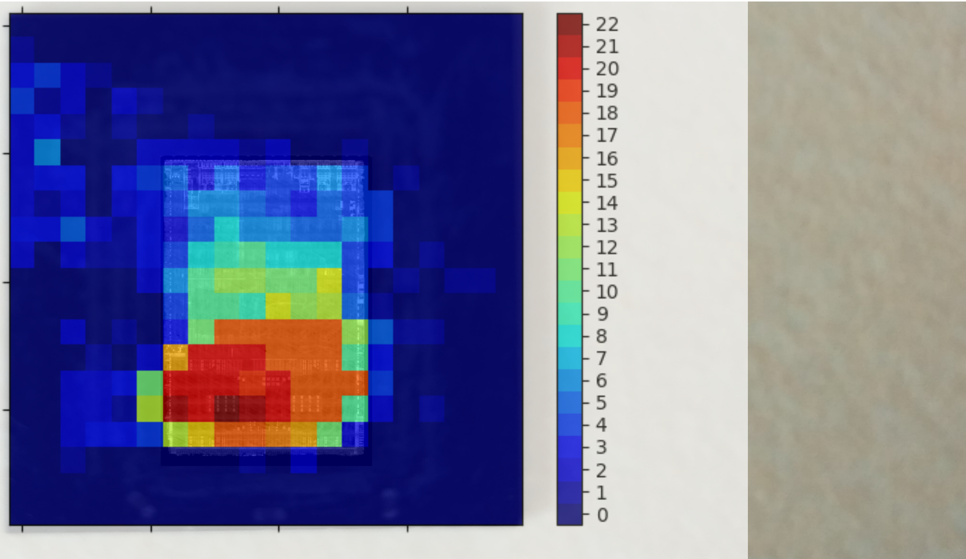
Experiments on Raspberry Pi 3 - Hardware correspondance with EM sensibility of the BCM2837



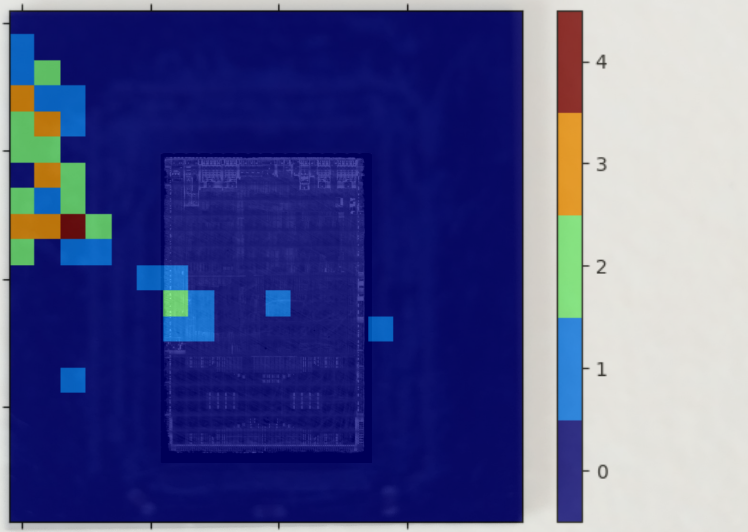
Experiments on Raspberry Pi 3 - Hardware correspondance with EM sensibility of the BCM2837



Experiments on Raspberry Pi 3 - Hardware correspondance with EM sensibility of the BCM2837



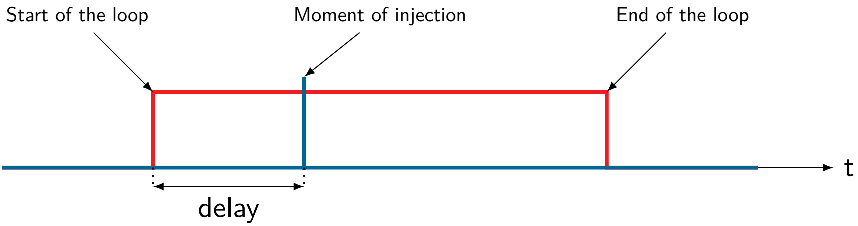
Experiments on Raspberry Pi 3 - Hardware correspondance with EM sensibility of the BCM2837



Experiments on Raspberry Pi 3 - Cartography on delay

Protocol

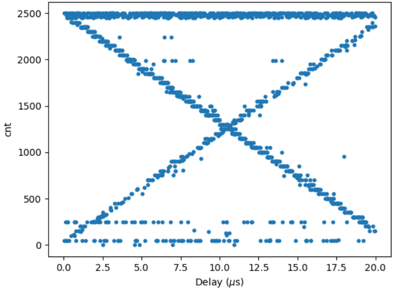
- Fixed position
- Fixed EM intensity
- Variation of the delay from the start to the end of the loop



Experiments on Raspberry Pi 3 - Cartography on delay

```
void loop(void){
  int i = 0;
  int j = 0;
  int cnt = 0;
  trigger_up();
  for(i=0; i<50; i++){
    for(j=0; j<50; j++){
      cnt++;
    }
  }
  trigger_down();
  print("i=%d j=%d cnt=%
     d\n", i, j, cnt);
}
```

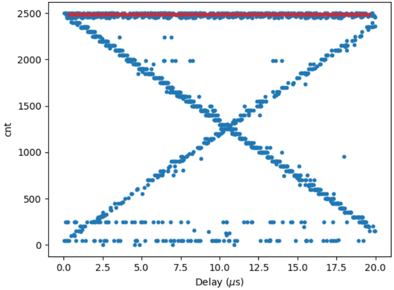
cnt vs delay



Experiments on Raspberry Pi 3 - Cartography on delay

```
void loop(void){
  int i = 0;
  int j = 0;
  int cnt = 0;
  trigger_up();
  for(i=0; i<50; i++){
    for(j=0; j<50; j++){
      cnt++;
    }
  }
  trigger_down();
  print("i=%d j=%d cnt=%
    d\n", i, j, cnt);
}
```

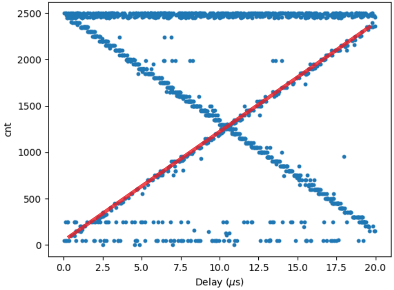
cnt vs delay



Experiments on Raspberry Pi 3 - Cartography on delay

```c
void loop(void){
  int i = 0;
  int j = 0;
  int cnt = 0;
  trigger_up();
  for(i=0; i<50; i++){
    for(j=0; j<50; j++){
      cnt++;
    }
  }
  trigger_down();
  print("i=%d j=%d cnt=%
    d\n", i, j, cnt);
}
```

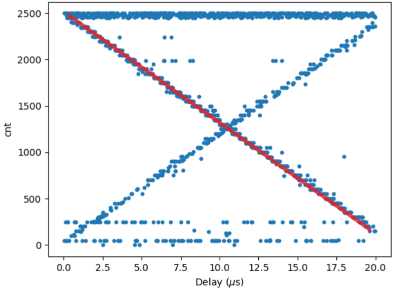
cnt vs delay



Experiments on Raspberry Pi 3 - Cartography on delay

```
void loop(void){
  int i = 0;
  int j = 0;
  int cnt = 0;
  trigger_up();
  for(i=0; i<50; i++){
    for(j=0; j<50; j++){
      cnt++;
    }
  }
  trigger_down();
  print("i=%d j=%d cnt=%
     d\n", i, j, cnt);
}
```

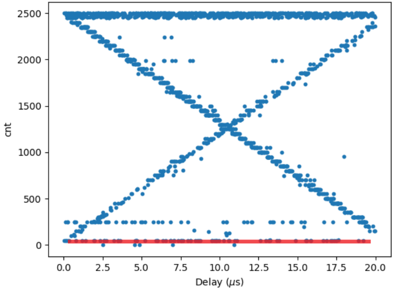
cnt vs delay



Experiments on Raspberry Pi 3 - Cartography on delay

```c
void loop(void){
  int i = 0;
  int j = 0;
  int cnt = 0;
  trigger_up();
  for(i=0; i<50; i++){
    for(j=0; j<50; j++){
      cnt++;
    }
  }
  trigger_down();
  print("i=%d j=%d cnt=%
     d\n", i, j, cnt);
}
```

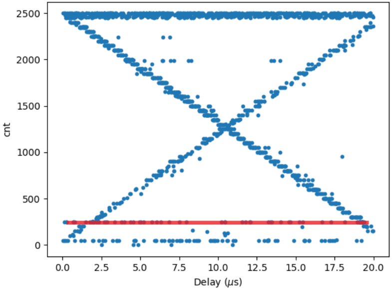
cnt vs delay



Experiments on Raspberry Pi 3 - Cartography on delay

```c
void loop(void){
  int i = 0;
  int j = 0;
  int cnt = 0;
  trigger_up();
  for(i=0; i<50; i++){
    for(j=0; j<50; j++){
      cnt++;
    }
  }
  trigger_down();
  print("i=%d j=%d cnt=%
     d\n", i, j, cnt);
}
```

cnt vs delay



Experiments on Raspberry Pi 3 - What happened ?

```
10e2c: e3a03000 mov    r3, #0
10e30: e50b3010 str    r3, [fp, #-16]
10e34: e3a03000 mov    r3, #0
10e38: e50b3008 str    r3, [fp, #-8]
10e3c: ea00000e b      10e7c <main+0x148>
10e40: e3a03000 mov    r3, #0
10e44: e50b300c str    r3, [fp, #-12]
10e48: ea000005 b      10e64 <main+0x130>
10e4c: e51b3010 ldr    r3, [fp, #-16]
10e50: e2833001 add    r3, r3, #1
10e54: e50b3010 str    r3, [fp, #-16]
10e58: e51b300c ldr    r3, [fp, #-12]
10e5c: e2833001 add    r3, r3, #1
10e60: e50b300c str    r3, [fp, #-12]
10e64: e51b300c ldr    r3, [fp, #-12]
10e68: e3530031 cmp    r3, #49 ; 0x31
10e6c: dafffff6 ble    10e4c <main+0x118>
10e70: e51b3008 ldr    r3, [fp, #-8]
10e74: e2833001 add    r3, r3, #1
10e78: e50b3008 str    r3, [fp, #-8]
10e7c: e51b3008 ldr    r3, [fp, #-8]
10e80: e3530031 cmp    r3, #49 ; 0x31
10e84: daffffed ble    10e40 <main+0x10c>
```

Experiments on Raspberry Pi 3 - What happened ?

```
10e2c:  e3a03000  mov   r3, #0                  cnt=0
10e30:  e50b3010  str   r3, [fp, #-16]
10e34:  e3a03000  mov   r3, #0                  i=0
10e38:  e50b3008  str   r3, [fp, #-8]
10e3c:  ea00000e  b     10e7c <main+0x148>
10e40:  e3a03000  mov   r3, #0                  j=0
10e44:  e50b300c  str   r3, [fp, #-12]
10e48:  ea000005  b     10e64 <main+0x130>
10e4c:  e51b3010  ldr   r3, [fp, #-16]
10e50:  e2833001  add   r3, r3, #1              cnt++
10e54:  e50b3010  str   r3, [fp, #-16]
10e58:  e51b300c  ldr   r3, [fp, #-12]
10e5c:  e2833001  add   r3, r3, #1              j++
10e60:  e50b300c  str   r3, [fp, #-12]          if j<50
10e64:  e51b300c  ldr   r3, [fp, #-12]
10e68:  e3530031  cmp   r3, #49 ; 0x31
10e6c:  daffff f6 ble   10e4c <main+0x118>
10e70:  e51b3008  ldr   r3, [fp, #-8]
10e74:  e2833001  add   r3, r3, #1
10e78:  e50b3008  str   r3, [fp, #-8]           i++
10e7c:  e51b3008  ldr   r3, [fp, #-8]           if i<50
10e80:  e3530031  cmp   r3, #49 ; 0x31
10e84:  daffffed  ble   10e40 <main+0x10c>
```

Experiments on Raspberry Pi 3 - Our ideas

Injection medium	**Physical target**	**Software target**	**Software security**
Software	RAM	Virtual to physical translation table	Memory partitioning
Glitch voltage	Clock	Key	Cryptography
Laser	Register	Instruction	Secure boot
EM	Bus	Return value	Execution flow integry
	Cache	Program counter	
	MMU	User rights	
	Pipeline	Variable	

EM Fault Injection

Experiments on Raspberry Pi 3 - Our ideas

Injection medium	Physical target	Software target	Software security
Software	RAM	Virtual to physical translation table	Memory partitioning
Glitch voltage	Clock	Key	Cryptography
Laser	Register	Instruction	Secure boot
EM	Bus	Return value	Execution flow integry
	Cache	Program counter	
	MMU	User rights	
	Pipeline	Variable	

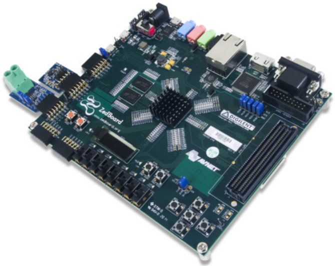
EM Fault Injection

Experiments on Raspberry Pi 3 - Conclusion

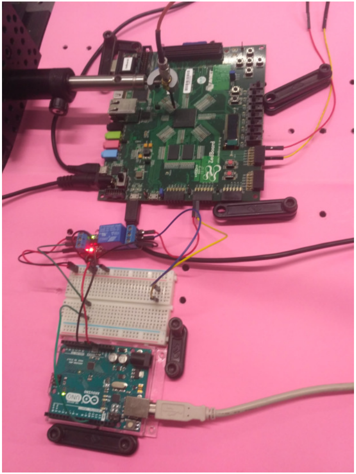
- ✓ EM Fault Injection is a promising attack path on complex SoCs
- ✓ Good repeatability
- ⚠ Few knowledge about the chip needed
- ✗ Very few knowledges about the behaviour of the chip
  - ✗ Not tested with a "real" program
  - ✗ Not tested on other SoCs and packages

The experiments - New target

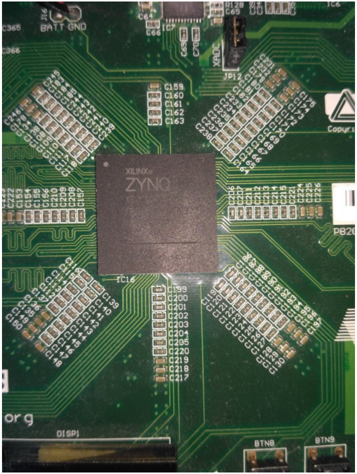
ZedBoard



- Xilinx Zynq 7000
- 2 Cortex A9
- 1 GHz



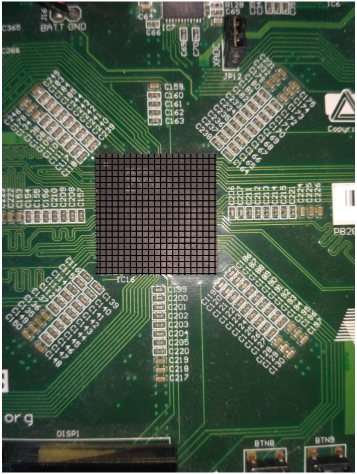
Experiments on ZedBoard - Zynq 7000 cartography



Zynq 7000 on the Zedboard

- 20x20 grid
- 3 different delays
- 3 different powers (positive and negative)
- 3 repetitions
- 54 operations/position

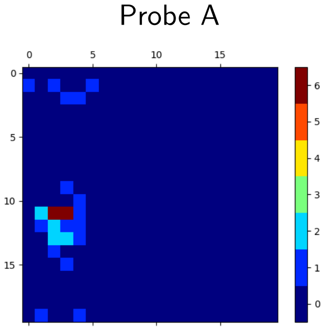
Experiments on ZedBoard - Zynq 7000 cartography



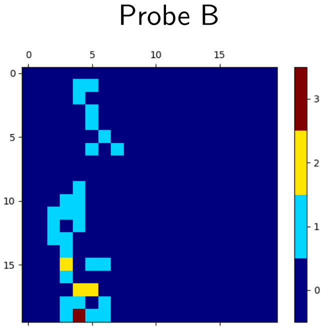
Zynq 7000 on the Zedboard

- 20x20 grid
- 3 different delays
- 3 different powers (positive and negative)
- 3 repetitions
- 54 operations/position

Experiments on ZedBoard - EM Sensibility of the Zynq 7000



36 effects for 21600 operations (0.17%)
12 effects without reboot for 36 effects (33.33%)



37 effects for 21600 operations (0.17%)
8 effects without reboot for 37 effects (21.6%)

Experiments on ZedBoard - Conclusion

✗ Only kernel exceptions

→ Paging request error

→ NULL pointer error

✗ Lot of OS crash

Future work

- Kernel debug via JTAG
- "Cold" cartography

Conclusion

I still don't have my PhD.

- SoCs are in every devices and use for sensitive services
- Lack of hardware understanding
- EM Fault Injection not investigated on SoCs yet
- My research fields for ANSSI
  - EMFI and software induced faults
  - Perturbation effects at high level with good repeatability
  - Deep investigation for EMFI
  - Investigation ongoing for ClkScrew

Questions?

References

[AS17] Simha Sethumadhavan Adrian Tang and Salvatore Stolfo. *CLKSCREW: Exposing the perils of security-oblivious energy management*. Tech. rep. Columbia University, 2017.

[Bla15] BlackHat. “XBOX 360 Glitching on fault attack”. Nov. 2015.

[Car17] Pierre Carru. “Attack TrustZone with Rowhammer”. In: eshard. 2017.

[TSW16] Niek Timmers, Albert Spruyt, and Marc Witteman. "Controlling PC on ARM Using Fault Injection". In: *2016 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2016, Santa Barbara, CA, USA, August 16, 2016*. IEEE Computer Society, 2016, pp. 25–35. DOI: 10.1109/FDTC.2016.18.

[Vas+17] Aurélien Vasselle et al. "Laser-induced fault injection on smartphone bypassing the secure boot". In: (Sept. 2017).

[Vee+16] Victor van der Veen et al. "Drammer: Deterministic Rowhammer Attacks on Mobile Platforms". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. Ed. by Edgar R. Weippl et al. ACM, 2016, pp. 1675–1689. DOI: 10.1145/2976749.2978406.