

Using Photon Emission Microscopy as a Hardware Attack Tool

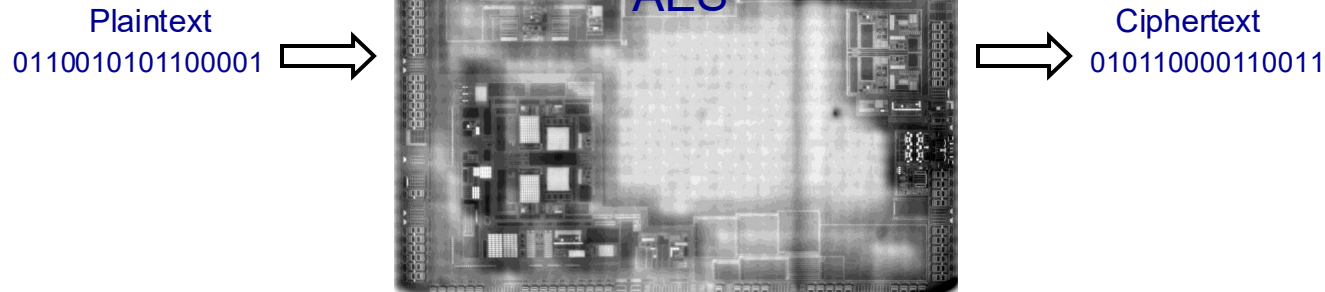
R. Silva Lima^{1, 2}, H.Perrin¹, R. Viera¹, J.B. Rigaud¹, W. Magrini²,
M. Pommies², A. Bertrand², J.-M. Dutertre¹

(1) Equipe Commune Systèmes et Architectures Sécurisées
Mines Saint-Etienne, CEA, Leti, Centre CMP
13541 Gardanne FRANCE

(2) Centre Technologique ALPhANOV – Systèmes Optiques

Context – Hardware security

- Hardware security – hardware attacks
- Secure HW: **integrated circuits implementing security features**
 - ✓ MCU with hardware cryptographic accelerator
 - ✓ Memory readback protection (IP & user data protection)



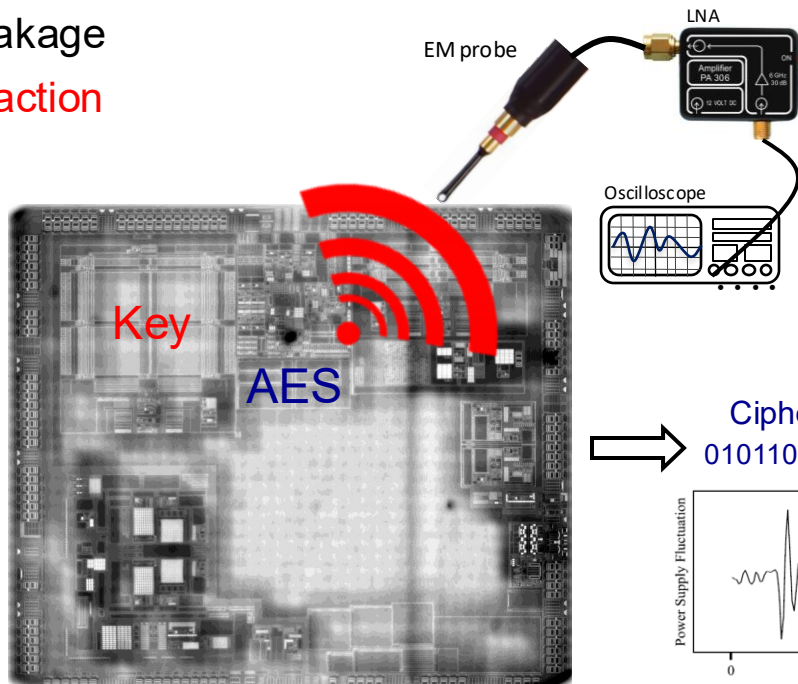
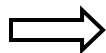
Context – Hardware security

- Hardware attacks
- Observation attacks (Side Channel Analysis)

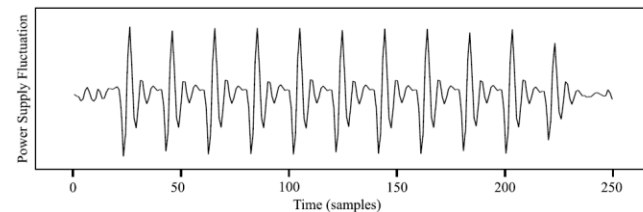
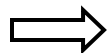
✓ Information leakage

→ **secret key extraction**

Plaintext
0110010101100001

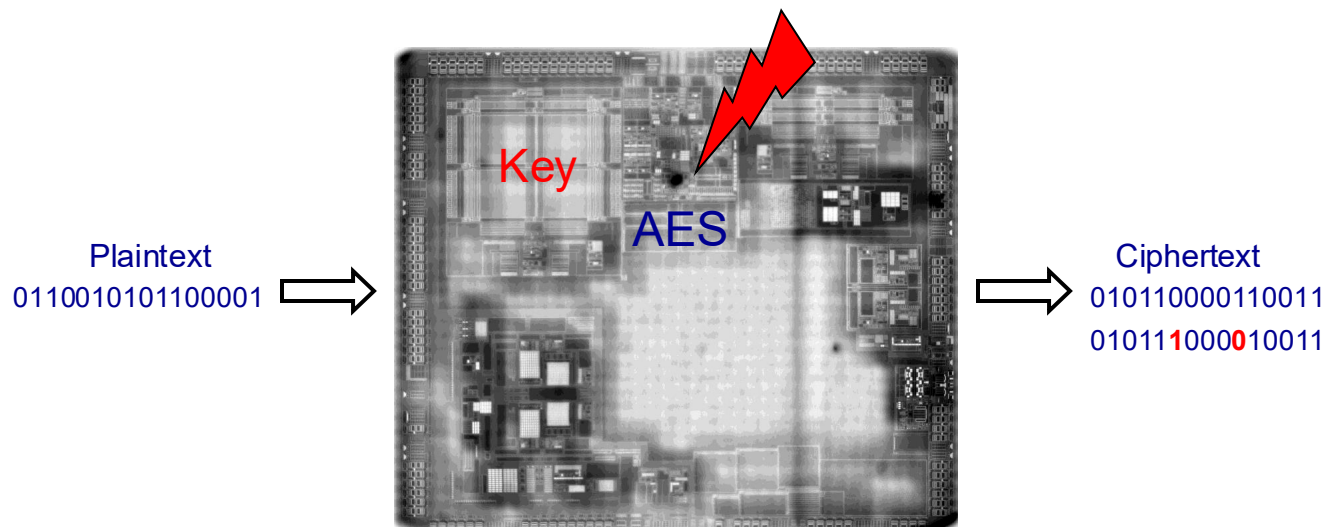


Ciphertext
010110000110011



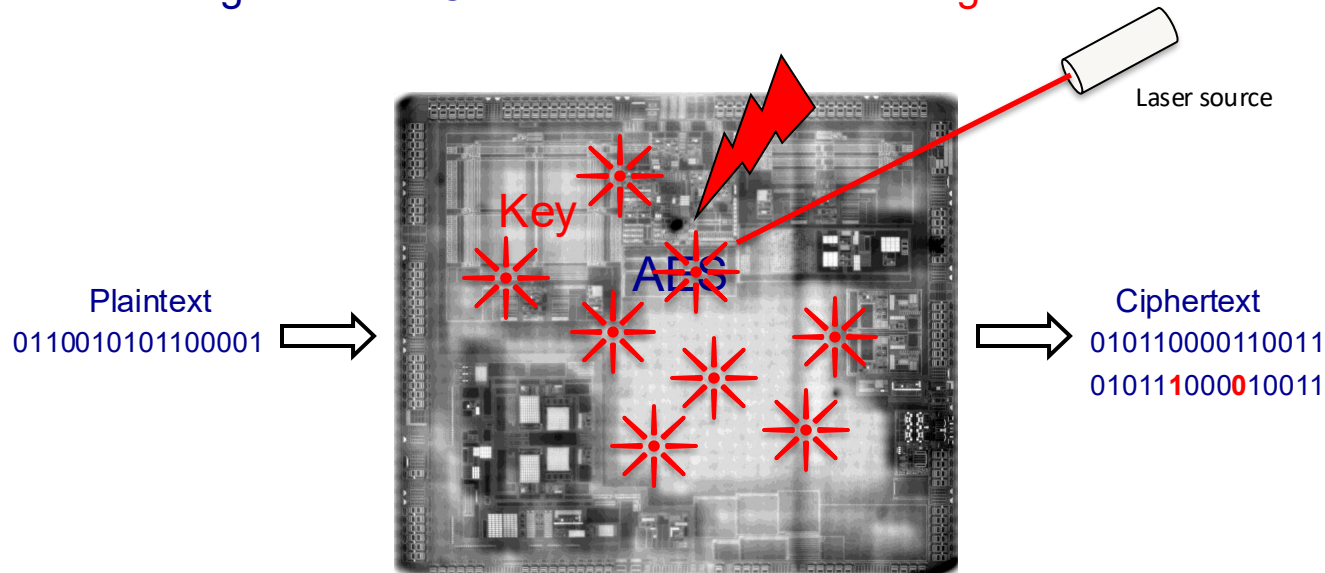
Context – Hardware security

- Hardware attacks
- Fault injections attacks
 - ✓ Information leakage (DFA) → secret key extraction
 - ✓ Control flow attacks (e.g., test inversion → memory extraction)



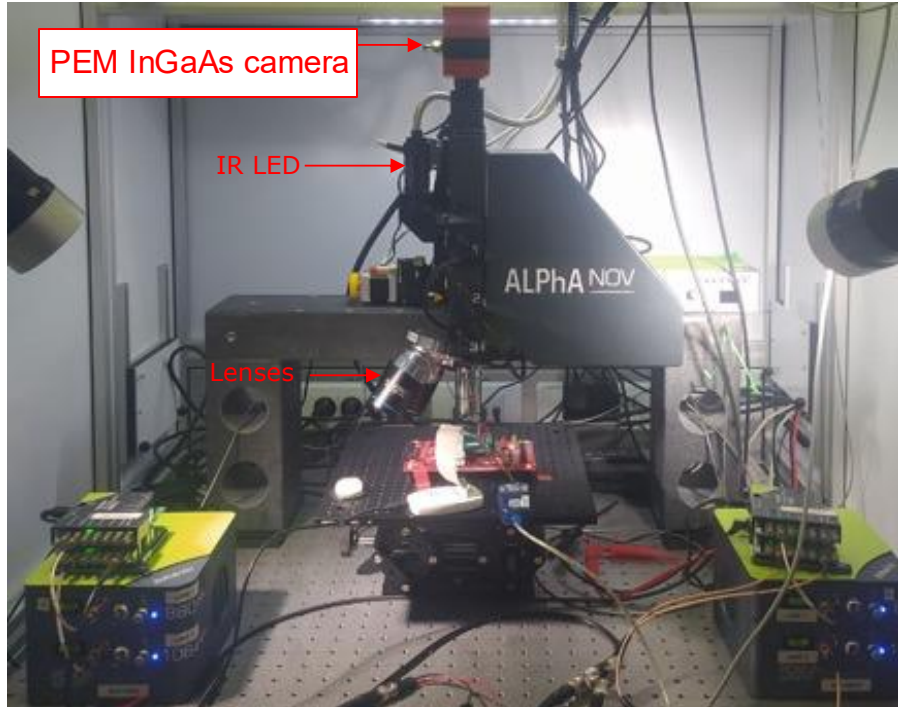
Context – Hardware security

- Hardware attacks
- Laser Fault Injection (LFI)
 - ✓ Accurate (μm accuracy) & efficient (bit-set/reset/flip)
 - Finding the Point Of Interest = **time consuming**



This talk

- Failure analysis as a hardware attack facilitation tools?



- FA tool: **photon emission analysis**
 - ✓ Reverse engineering to accelerate fault injection attacks
 - ✓ LFI: **where?** and when?

→ **Photon Emission Microscopy**

- Q? Use of PEM to accelerate LFI?
- Q? Data leakage/extraction?

Using PEM as a Hardware Attack Tool

- Photon Emission (PE) basics
- PEM for LFI facilitation
- Data extraction from SRAM memories
- Data extraction from Flash memories
- From limitations to a practical attack scenario

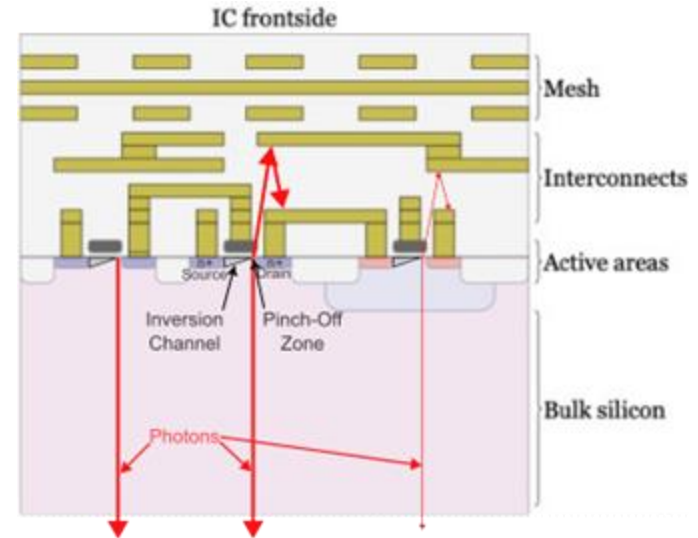
PE mechanism

- Photon emission from transistors activity
 - ✓ Source-drain electric field: charge carrier acceleration
 - ✓ Kinetic energy released as photons
 - ✓ MOS transistors in saturation mode (pinch-off channel, drain)
 - ✓ $\text{NMOS}_{\text{emission}} > \text{PMOS}_{\text{emission}}$

FA tool: default localization (90s)

Also efficient to observe transistors in nominal mode

- ✓ Switching transistors (digital logic)
- ✓ Bias current of analog parts
- + tunneling effects (Fowler-Nordheim)



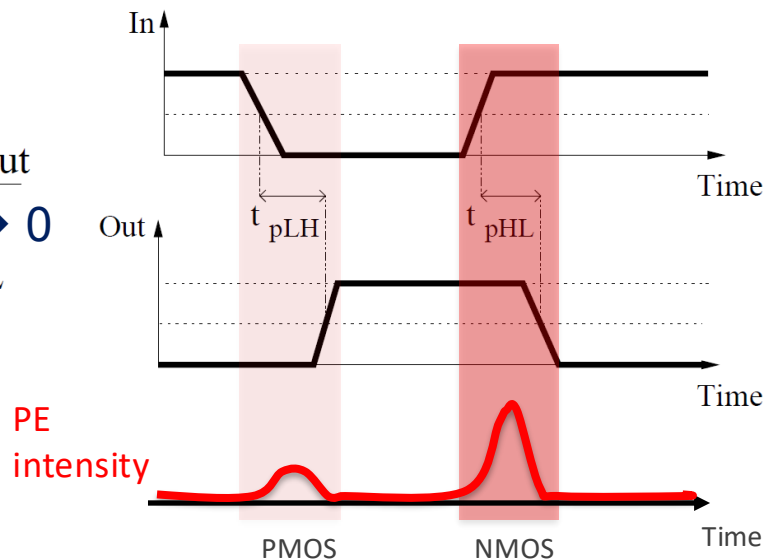
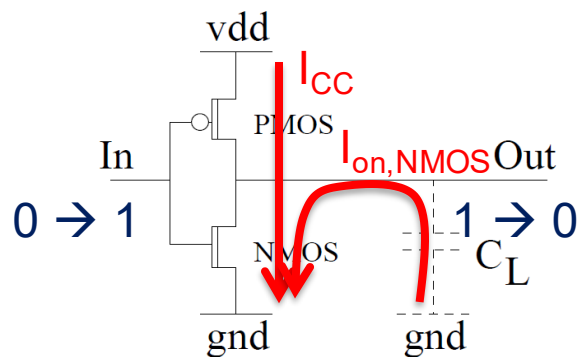
[Security of the IC Backside,
D. Nedospasov, 2015]

PE mechanism

- Photon emission from logic gates

The inverter case

- ✓ Switching transistors (digital logic)



PE mechanism

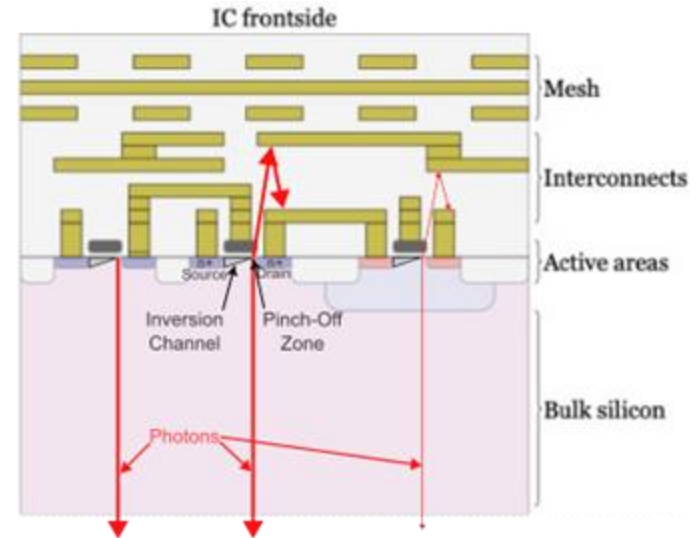
Backside PEA ($\lambda = 1-2 \mu\text{m}$)

(to avoid reflection on metal lines and dummies)

- ✓ Si substrate transparent to NIR
- ✓ Substrate thinning improves SNR

Factors favorizing PE

- ✓ Current density
- ✓ V_{DS} voltage



[Security of the IC Backside,
D. Nedospasov, 2015]

Photon Emission Microscopy (PEM) setup

Photon Emission maps → **transistors activity maps**

Camera:

- ✓ 640x512 **InGaAs** sensor
- ✓ On a LFI bench
- ✓ Typical readout noise (rms) : 18 e^-
- ✓ Typical dark current (@-15 °C) : $< 750 \text{ e}^-$
- ✓ High sensitivity from $\lambda = 0.6$ to $1.7 \text{ }\mu\text{m}$
- ✓ $15 \times 15 \mu\text{m}$ pixel pitch
- ✓ Peak Quantum Efficiency : $> 90\%$ @ $1.3 \mu\text{m}$
- ✓ Air-cooled to -15 °C



PEM constraints

Signal to Noise Ratio

✓ Information shall emerge from noise

- PEM: long integration time

On a running device

→ execution of code loops



Strong constraints for attack purposes

→ white box model:

- ✓ Ability to execute arbitrary code loops
- ✓ Synchronization



sensor

Using PEM as a Hardware Attack Tool

- Photon Emission (PE) basics
- PEM for LFI facilitation
- Data extraction from SRAM memories
- Data extraction from Flash memories
- From limitations to a practical attack scenario

PEM for LFI facilitation

Target 1

Microcontroller:

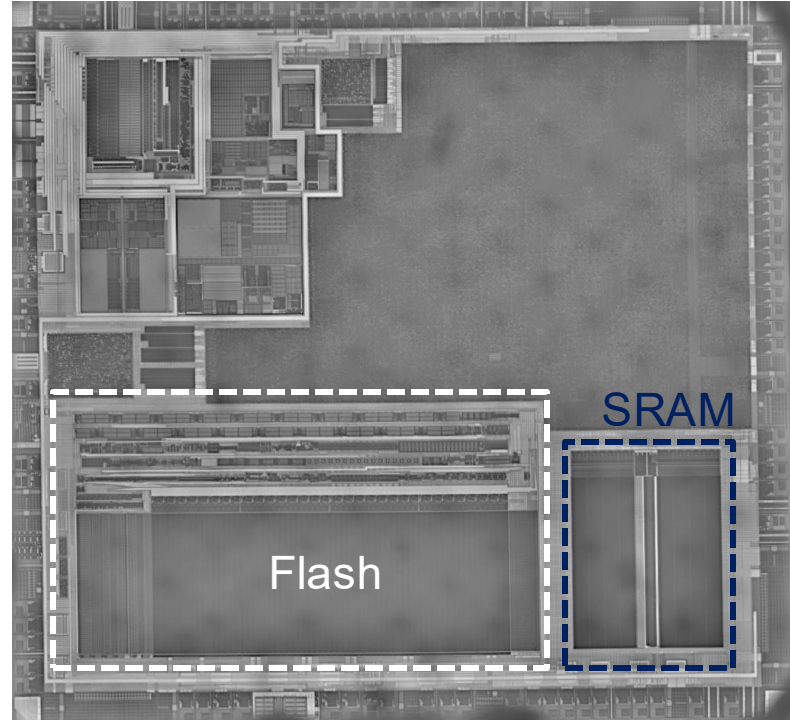
- ✓ ARM Cortex M3
- ✓ CMOS 90 nm
- ✓ 32-bit CPU, 24 MHz
- ✓ 128 kBytes Flash
 - page size = 1 kB
- ✓ 8 kBytes SRAM
- ✓ Si thickness: $\sim 350 \mu\text{m}$

Si die: $3,000 \times 2,500 \mu\text{m}$

Flash: $1,400 \times 550 \mu\text{m}$ – 1.3 bits/ μm^2

SRAM: $245 \times 660 \mu\text{m}$ (x2) – 0,1 bits/ μm^2

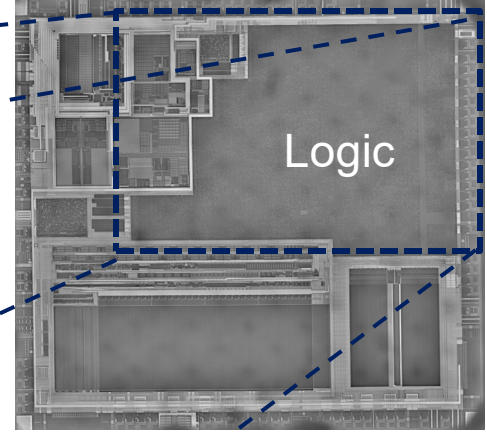
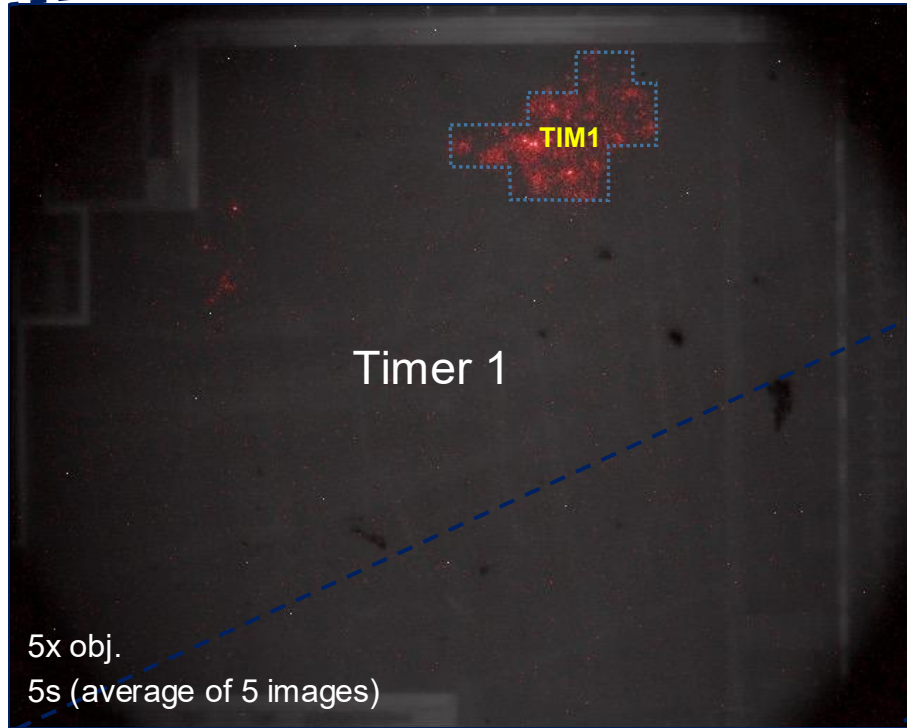
Backside IR view



PEM for LFI facilitation – Reverse engineering

PEM on target 1

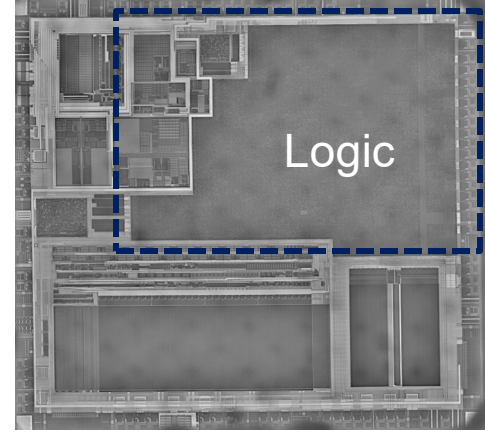
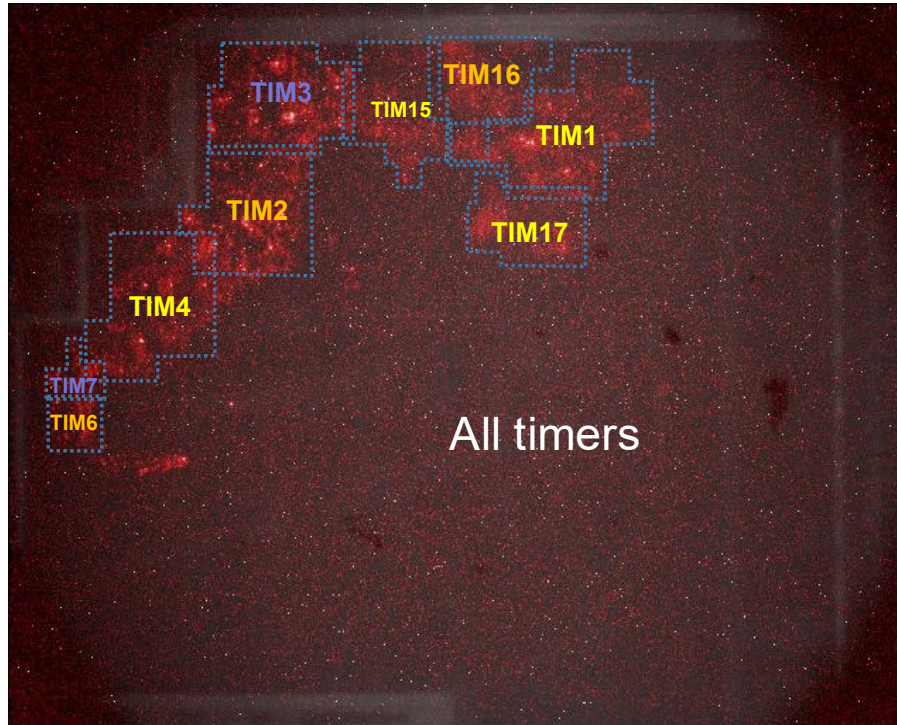
- ✓ MCU floorplan reverse engineering



PEM for LFI facilitation – Reverse engineering

PEM on target 1

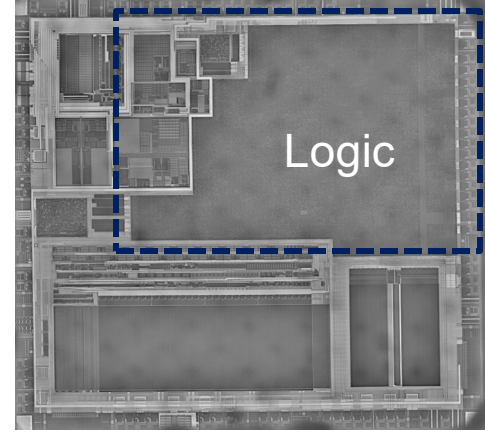
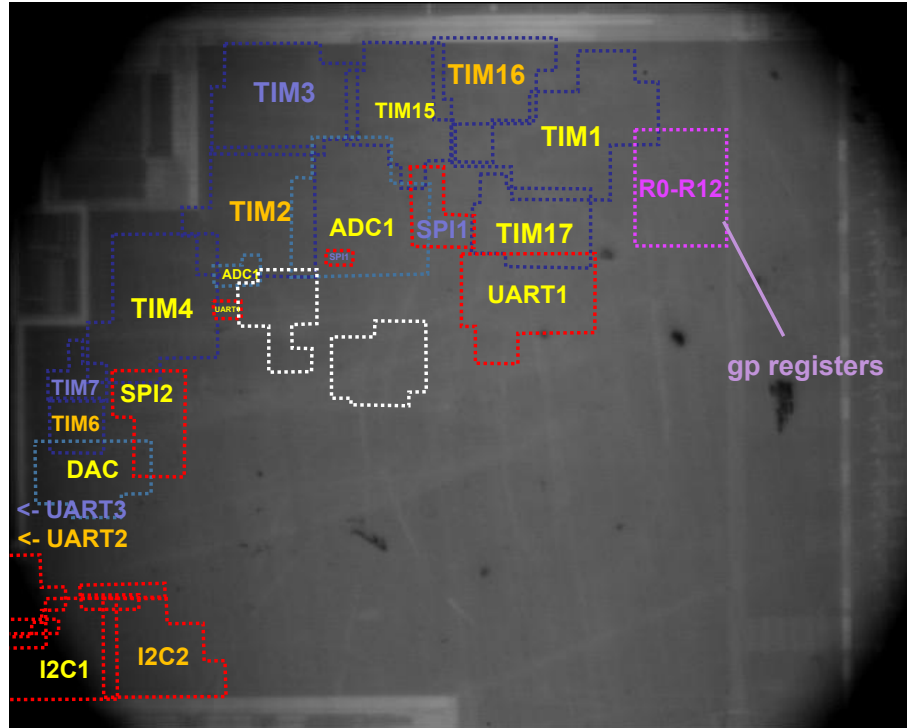
- ✓ MCU floorplan reverse engineering



PEM for LFI facilitation – Reverse engineering

PEM on target 1

- ✓ MCU floorplan reverse engineering

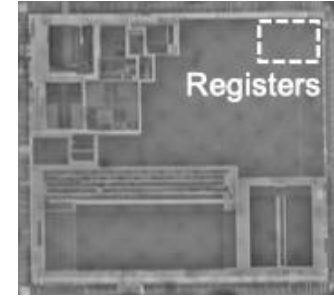


- ✓ MCU gp registers localization

PEM for LFI facilitation – Reverse engineering

PEM on target 1

- ✓ MCU floorplan reverse engineering

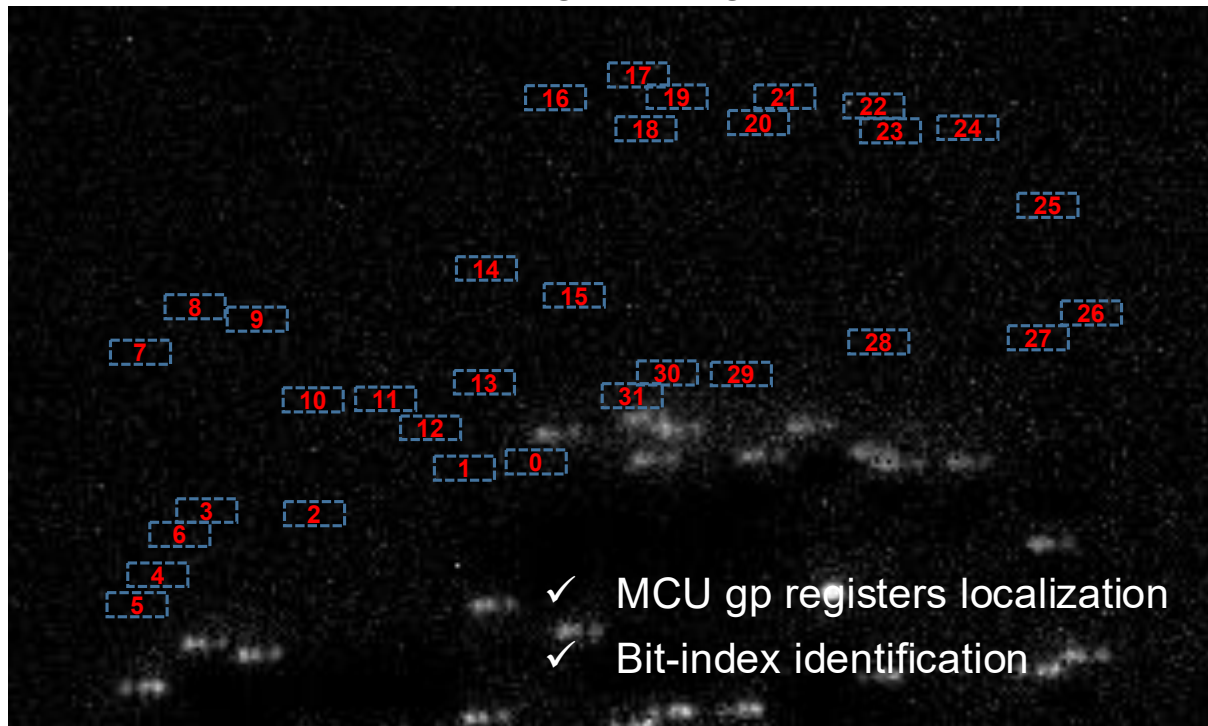


Lens x20, 5s (avg. 5)
600 Mio. EORS
register R3

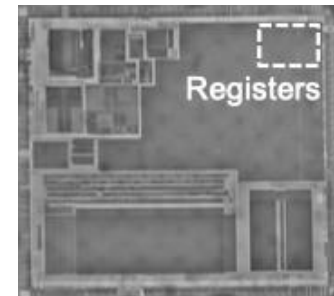
PEM for LFI facilitation – Reverse engineering

PEM on target 1

- ✓ MCU floorplan reverse engineering



- ✓ MCU gp registers localization
- ✓ Bit-index identification

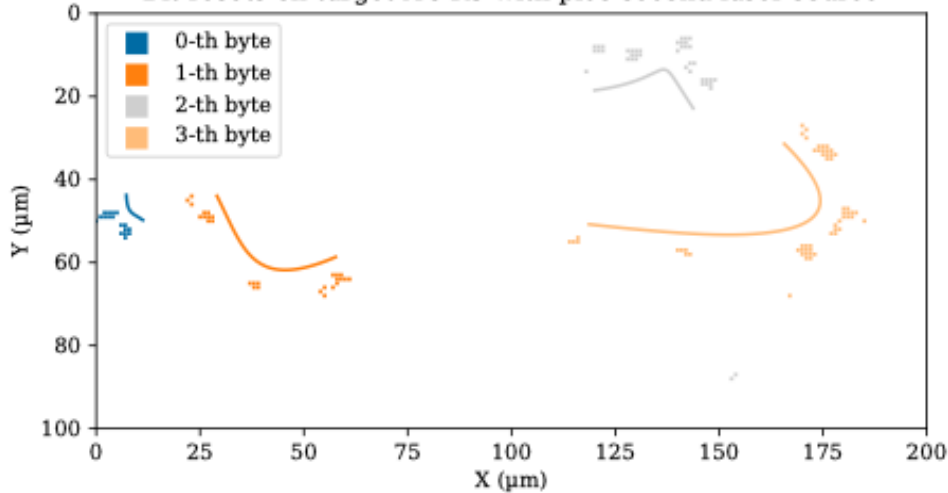


Lens x20, 5s (avg. 5)
600 Mio. EORS
register R3

PEM for LFI facilitation – LFI

LFI from PEM-based register DFFs localization

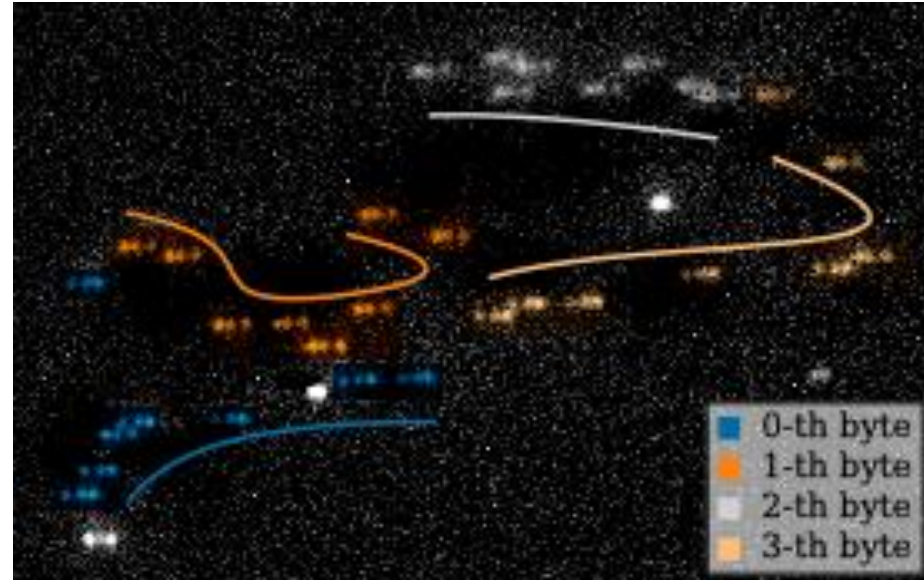
Bit-resets on target A's R3 with pico-second laser source



Pico-second laser source: 60 ps pulses

- ✓ 1,064 nm
- ✓ 0.12 nJ

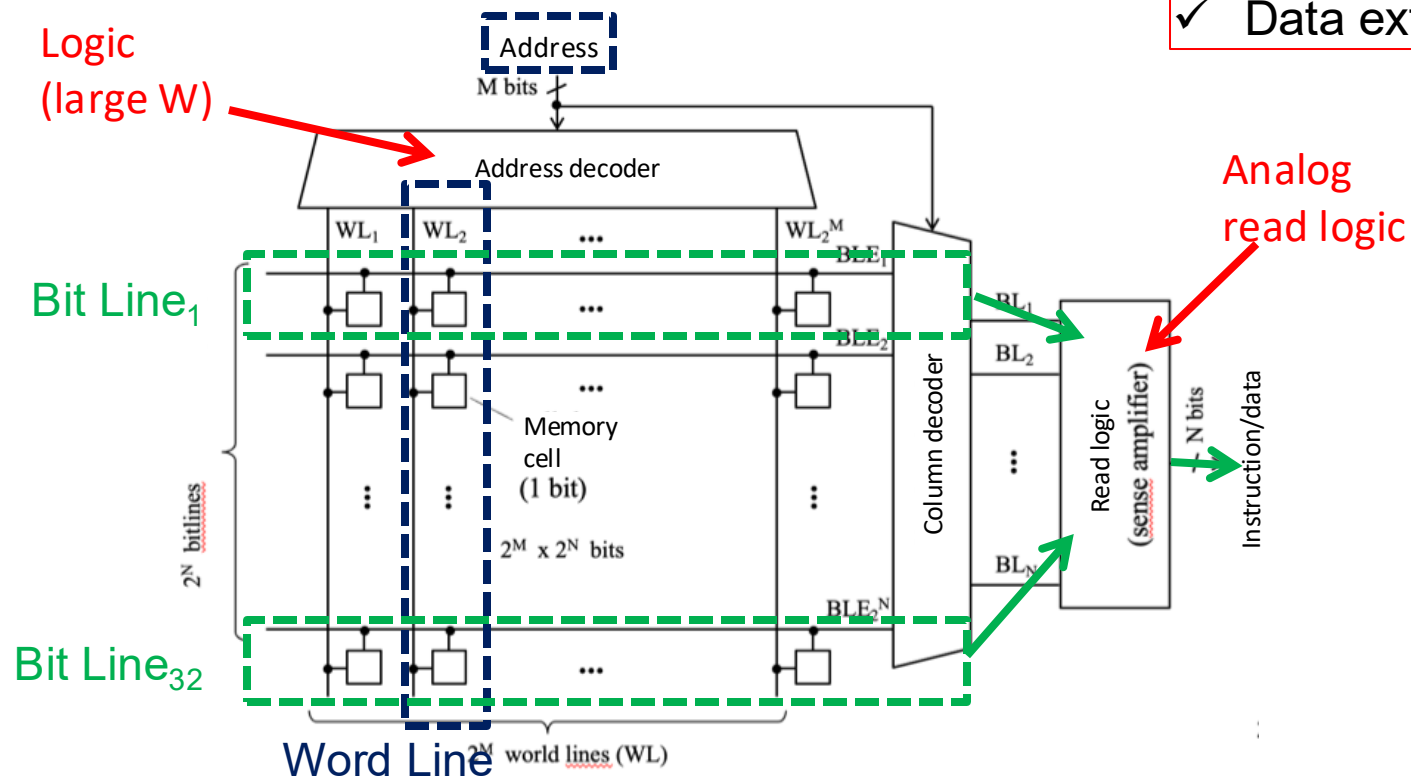
R3 bytes identification



Using PEM as a Hardware Attack Tool

- Photon Emission (PE) basics
- PEM for LFI facilitation
- Data extraction from SRAM memories
 - ✓ Memory organization
 - ✓ PEM at read and write time
 - ✓ PE from read/write logic
- Data extraction from Flash memories
- From limitations to a practical attack scenario

Memory organization

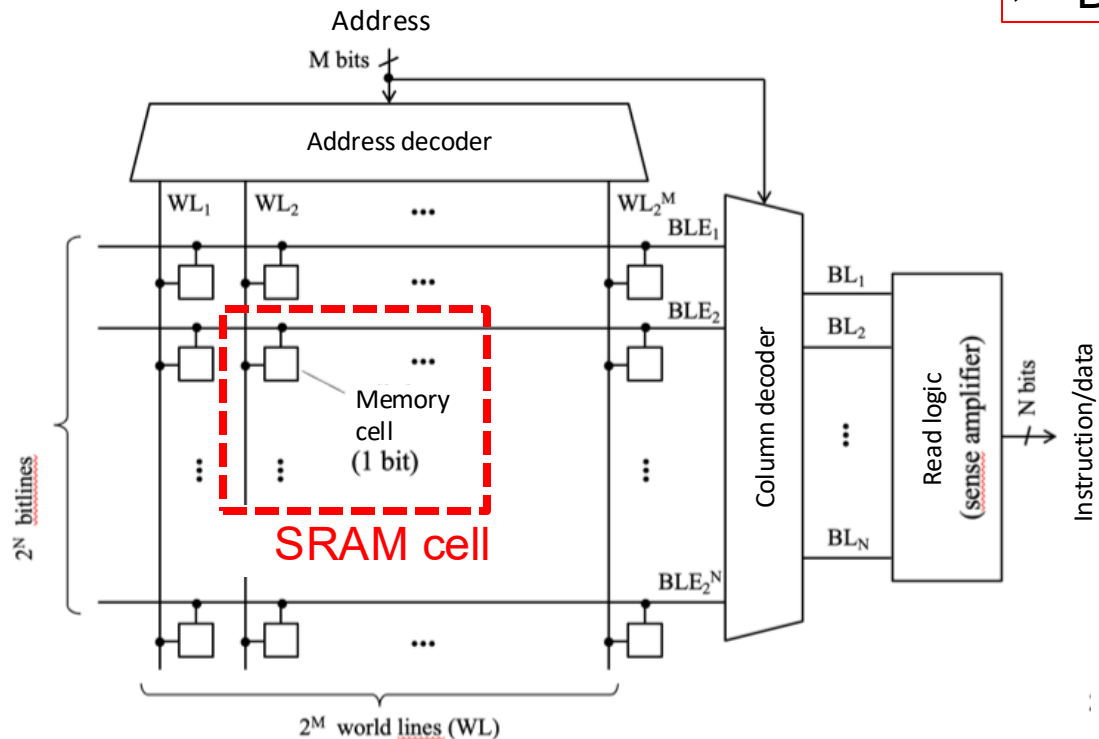


Attacker:

- ✓ Memory organization (RE)
- ✓ Data extraction

Main photon emitters

Memory organization



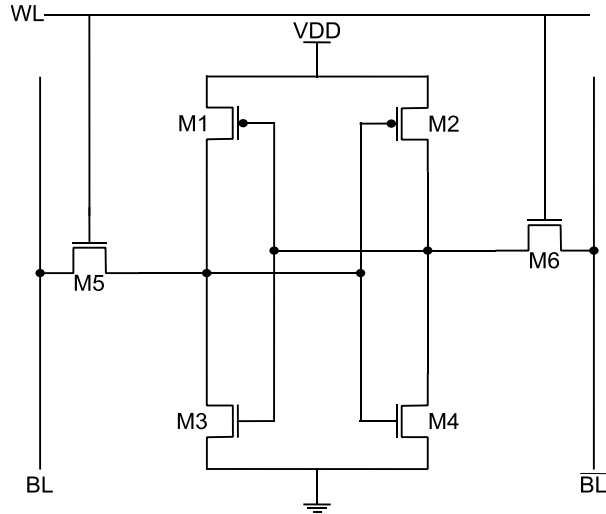
Attacker:

- ✓ Memory organization (RE)
- ✓ Data extraction

Main photon emitters

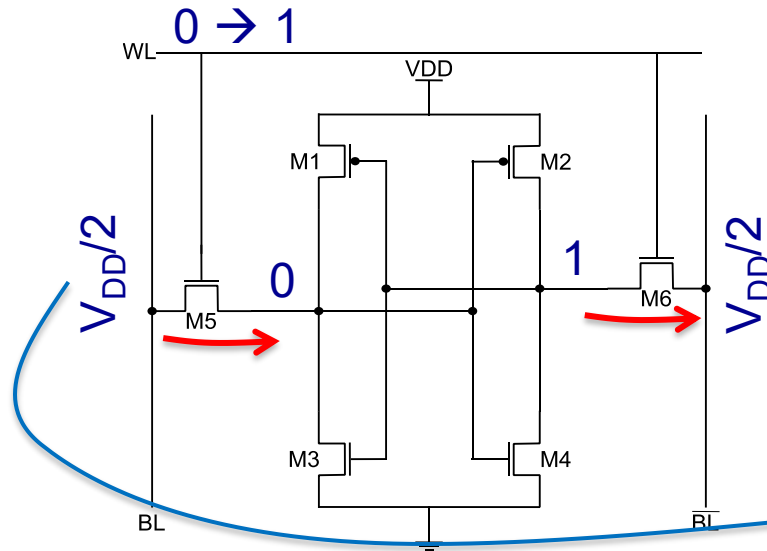
SRAM photon emission – Theory

6T SRAM cell

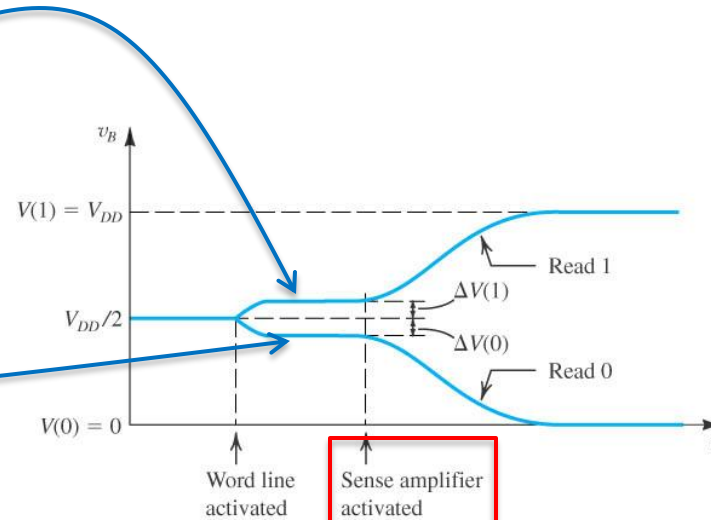


SRAM photon emission – Theory

6T SRAM cell – At read time



Weak access transistor current
→ Weak PE



SRAM photon emission – Exp. results

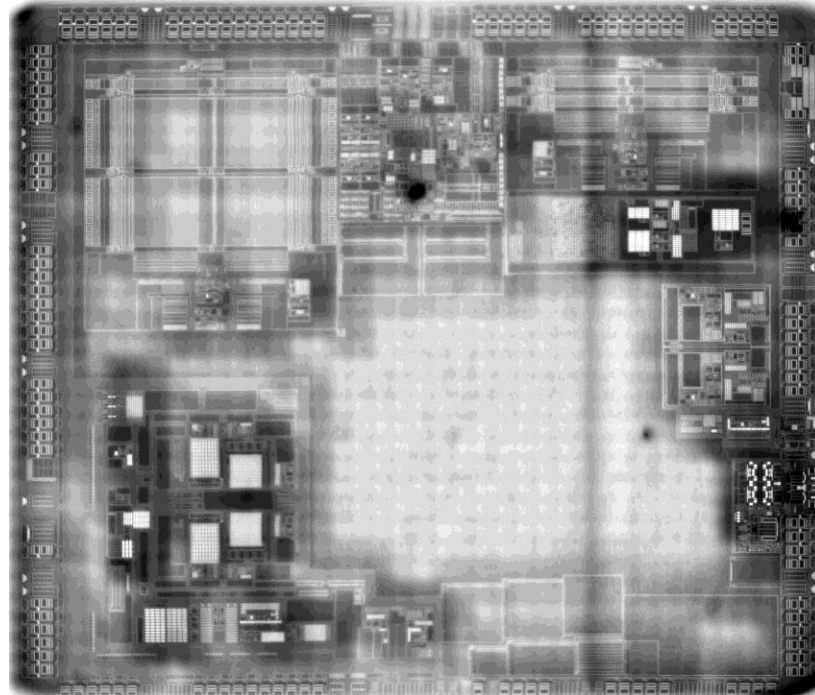
Target 2

- ✓ ARM Cortex M4
- ✓ 32-bit CPU, 80 MHz
- ✓ 256 kBytes Flash
page size = 1 kB
- ✓ 32 kBytes SRAM
- ✓ Si thickness: $\sim 250 \mu\text{m}$

Si die: $3,600 \times 3,300 \mu\text{m}$

Flash: $330 \times 310 \mu\text{m}$ (x4) – 5 bits/ μm^2

SRAM: $250 \times 265 \mu\text{m}$ (x2) – 2 bits/ μm^2

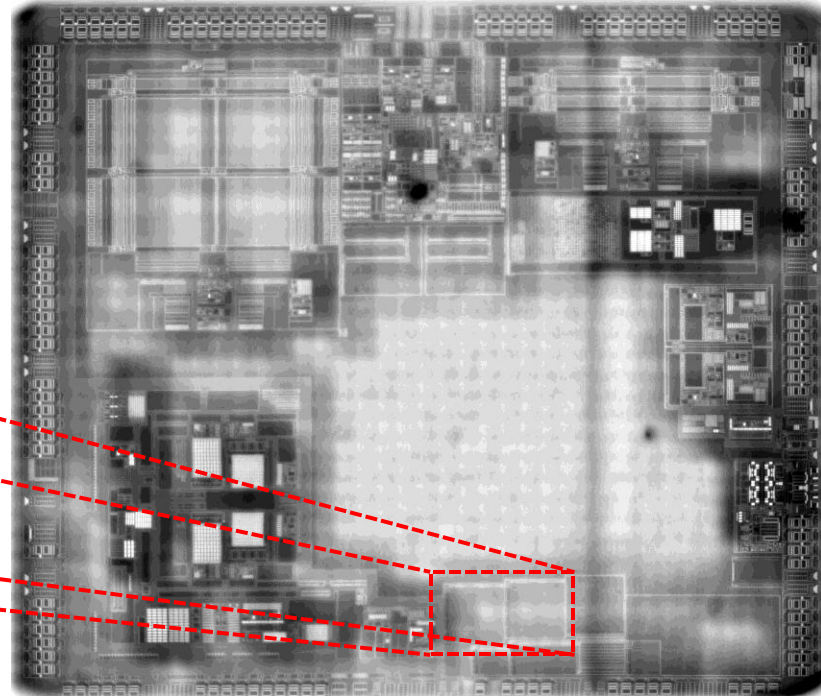
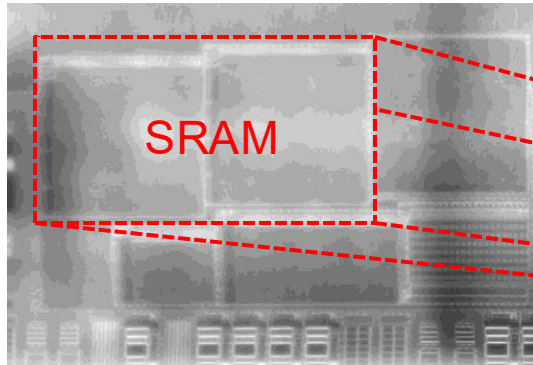


SRAM photon emission – Exp. results

Target 2 – At read time

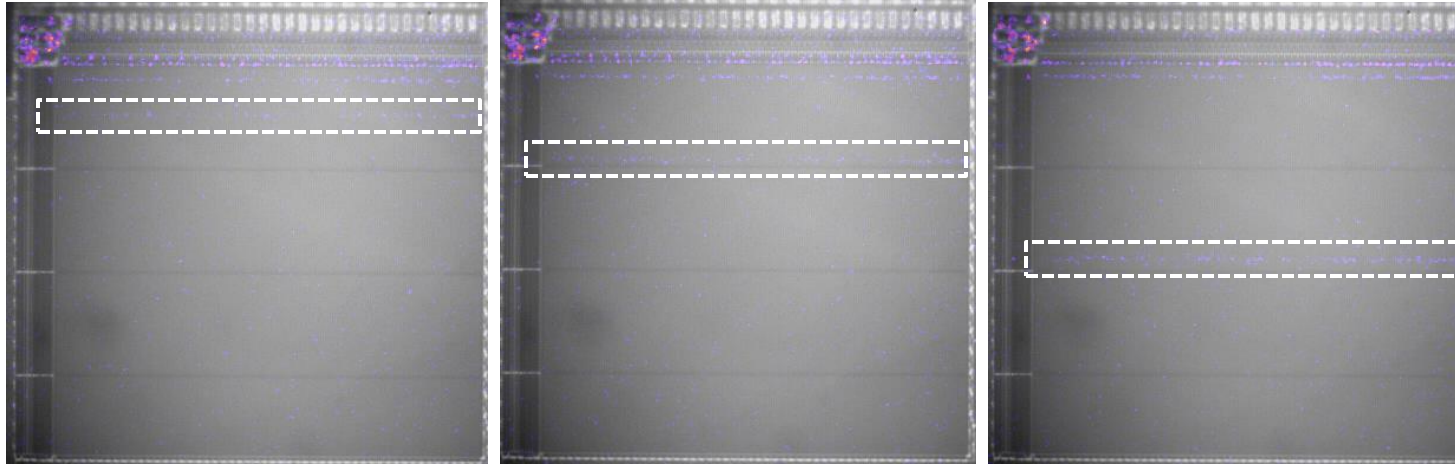
2x 16 kBytes SRAM

- ✓ Left even @
- ✓ Right odd @
- ✓ 0x20000000 – 0x20007FFF



SRAM photon emission – Exp. results (read)

Target 2 – PE at read time



Photon emission map at read time: 20x lens, exposure 5s

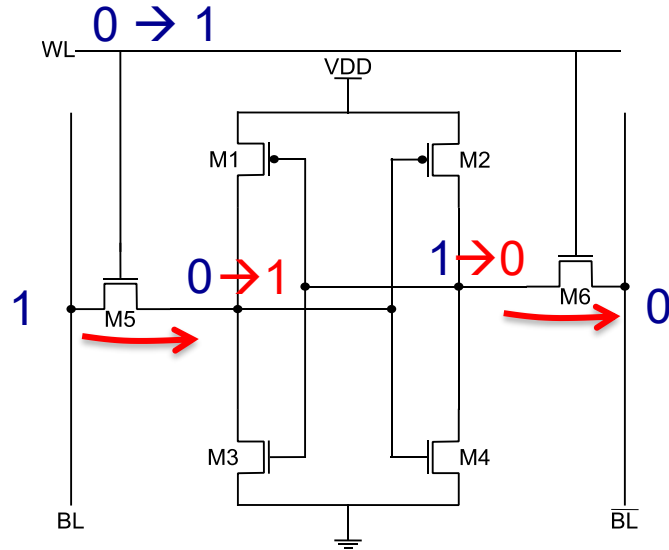
@: 0x20001000 - 0x20003000 - 0x20004000 (left to right)



Weak emission at read time

SRAM photon emission – Theory

6T SRAM cell – At write time ($0 \rightarrow 1$)

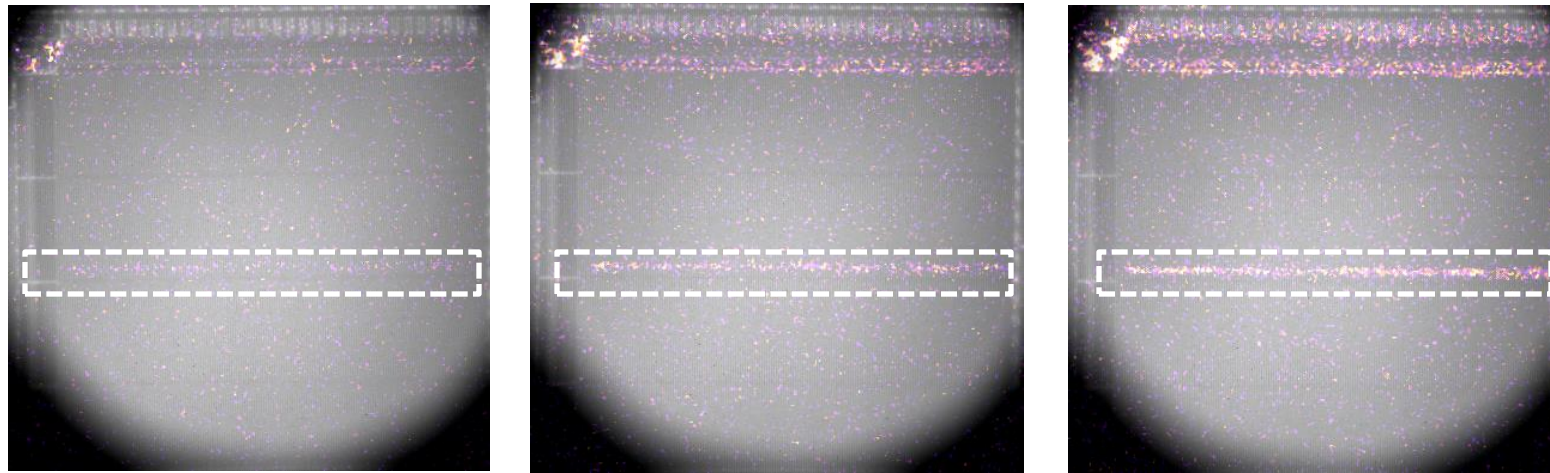


Switching inverters
+ Strong access T current
→ Strong PE

SRAM photon emission – Exp. results (write)

Target 2 – PE at write time

Test code: write 0x00000000, then 0xFFFFFFFF



Photon emission map at write time: 20x lens, exposure 5s, @: 0x20004000

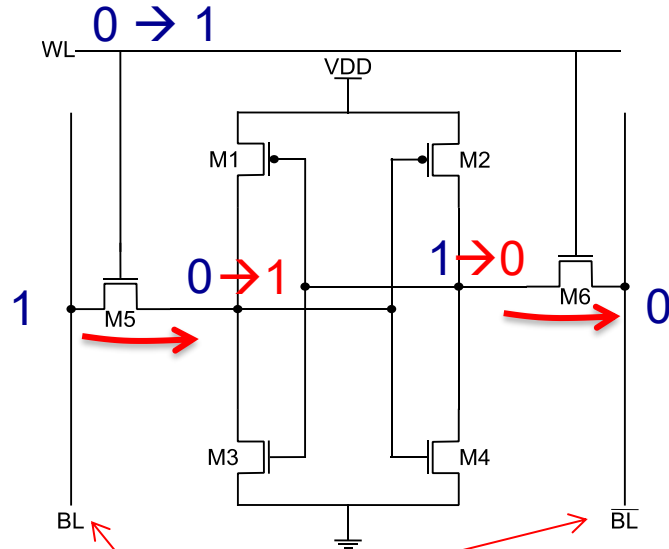
1 word, 8 words, 64 words (left to right)

1 write cycle ~ 550ns → : 9.1 Mio. cycles in 5s

SRAM photon emission – Theory

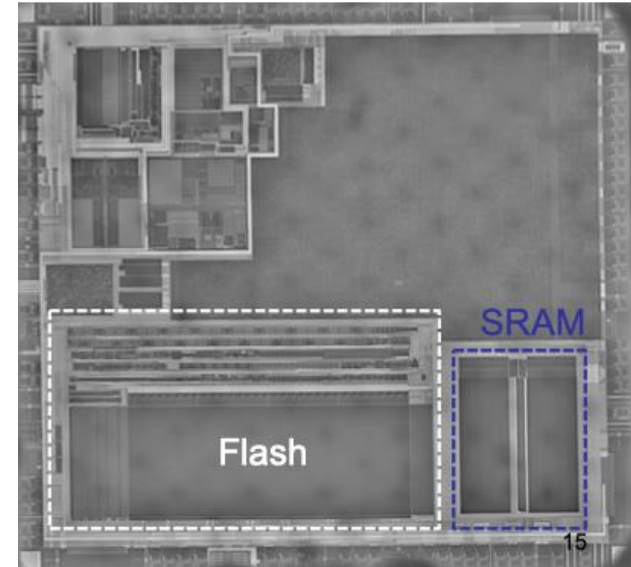
6T SRAM cell – At write (read) time

Target 1



BL drivers (analog)
→ Strong PE

Backside IR view



128 kBytes Flash

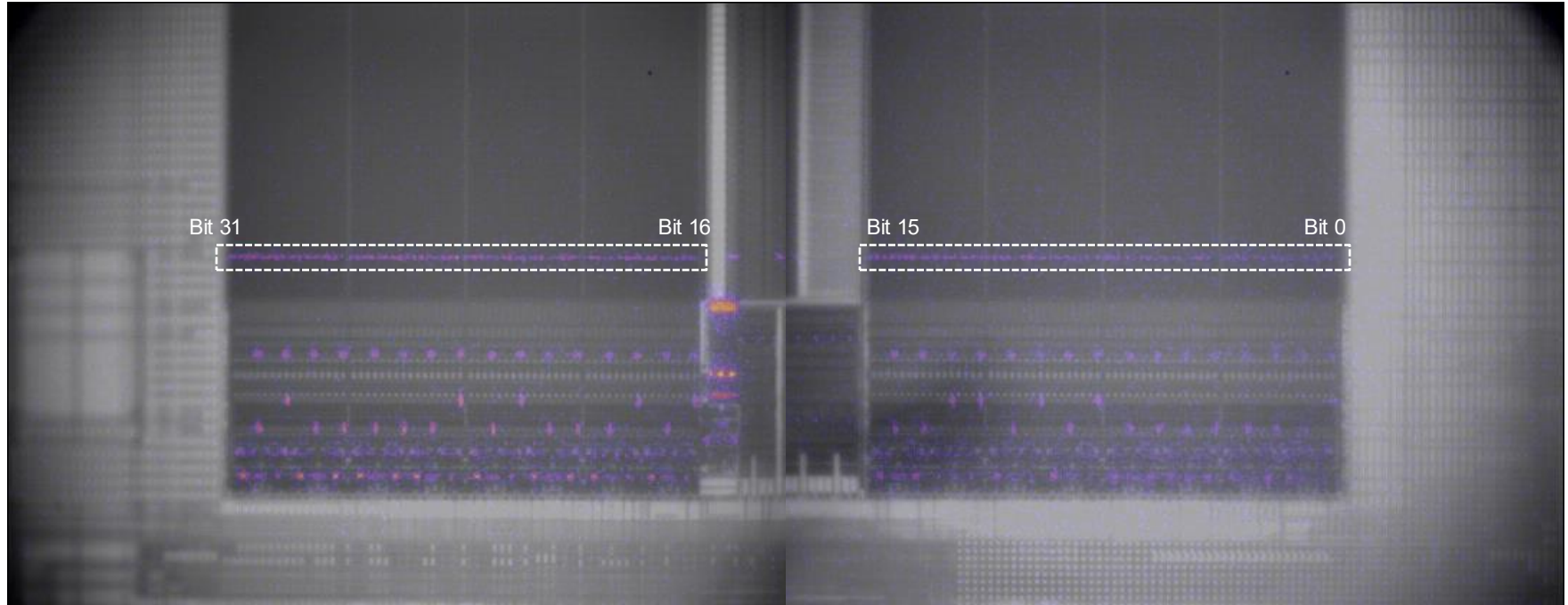
8 kBytes SRAM

✓ page size = 1 kB

SRAM data leakage – Exp. results

Target 1 – PE from read/write analog logic

Test code: write 0xBEBACAFE (loops)

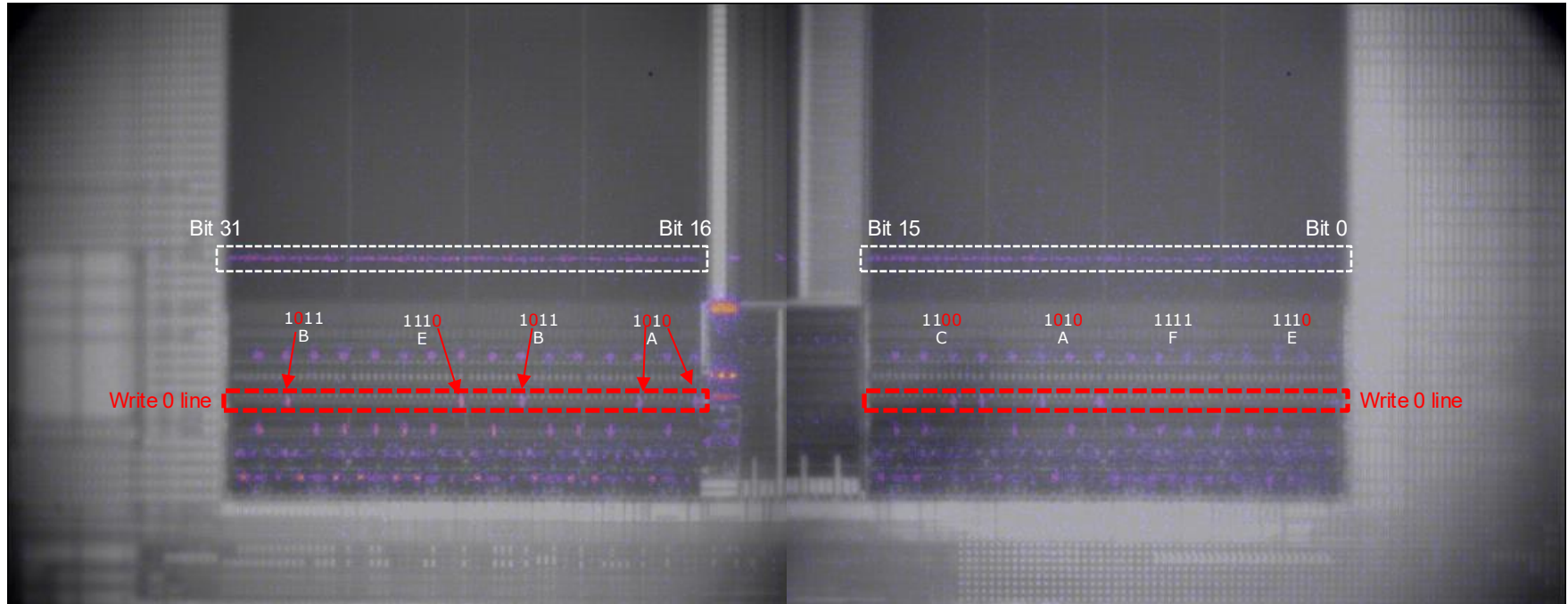


Photon emission map at write time: 20x lens, exposure 2.5s, 1 word

SRAM data leakage – Exp. results

Target 1 – PE from read/write analog logic

Test code: write 0xBEBACAFE (loops)

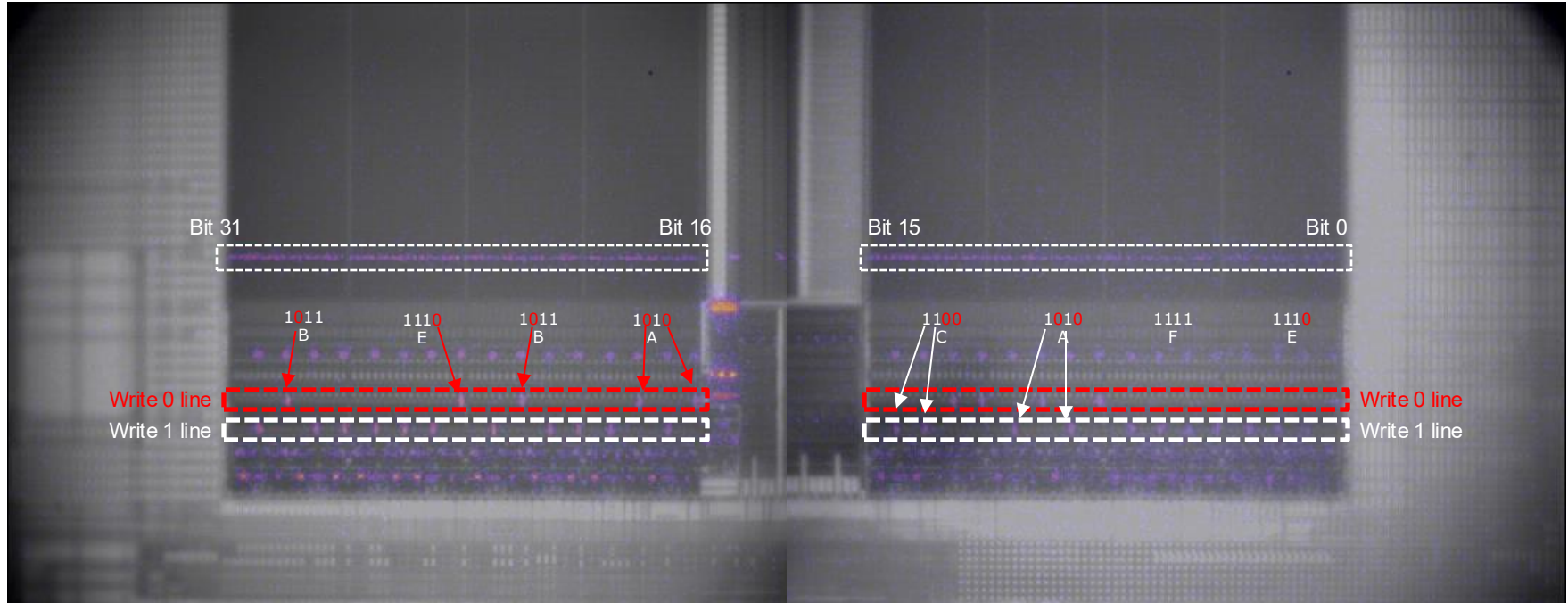


Photon emission map at write time: 20x lens, exposure 2.5s, 1 word

SRAM data leakage – Exp. results

Target 1 – PE from read/write analog logic

Test code: write 0xBEBACAFE (loops)



Photon emission map at write time: 20x lens, exposure 2.5s, 1 word

SRAM data leakage – Exp. results

Target 1 – PE from read/write analog logic

Test code: write 0xBEBACAFE (loops)



Written/read data are exposed in plain sight
(card player piano like)



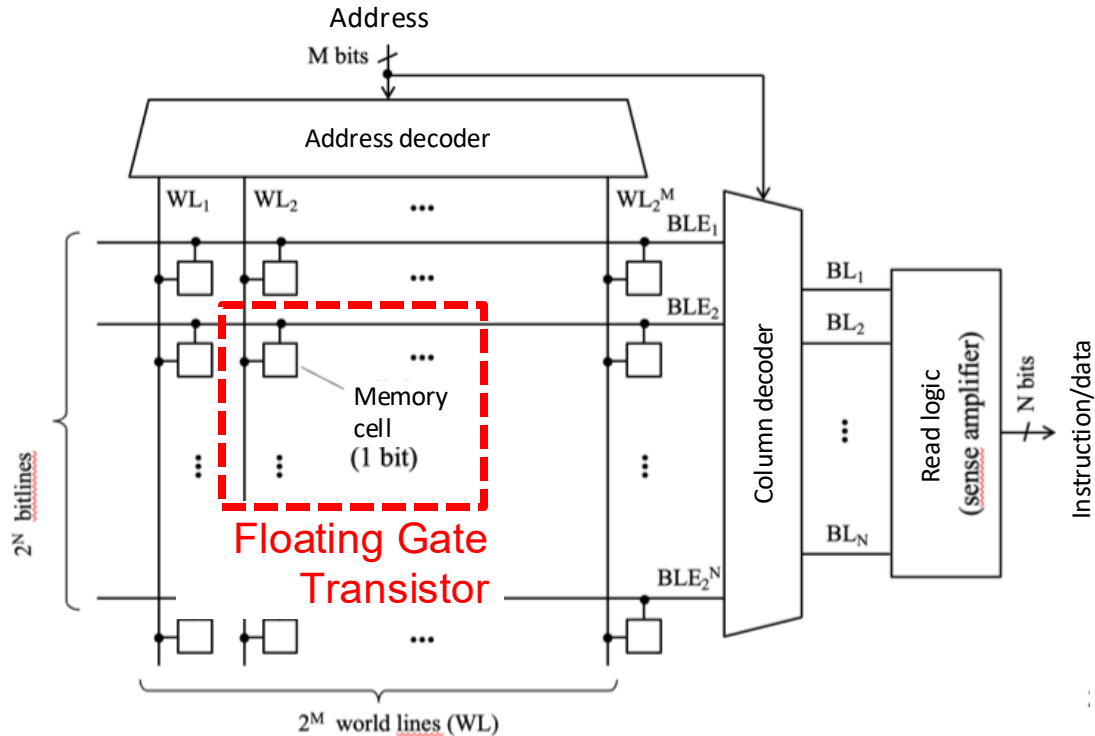
Photon emission map at write time: 20x lens, exposure 2.5s, 1 word

Using PEM as a Hardware Attack Tool

- Photon Emission (PE) basics
- PEM for LFI facilitation
- Data extraction from SRAM memories
- Data extraction from Flash memories
 - ✓ Flash modes of operation
 - ✓ PEM at erase and program time
 - ✓ Data dependency of PE
- From limitations to a practical attack scenario

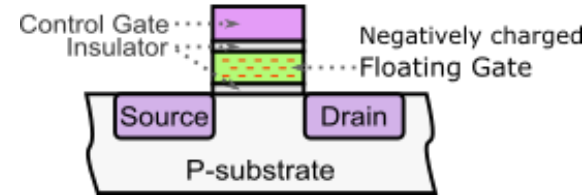
Flash memory organization

Unit cell: FG transistor



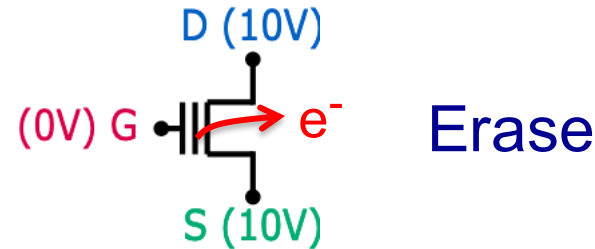
Flash memory modes of operation

Writing in an embedded Flash is a complex 2-step process: **erase & program**

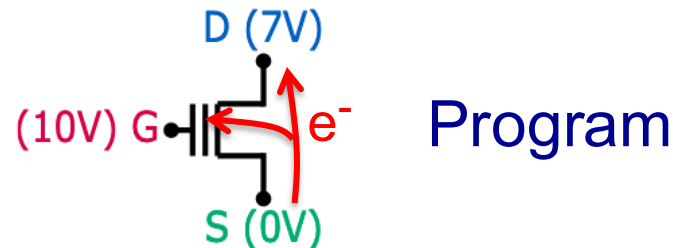


Flash memories are ...

- ✓ ... **erased** at **page level** (e.g. 1 kB)
- ✓ Fowler-Nordheim tunneling effect
- **Set to 1** (or 0xFFFFFFFF at word level)



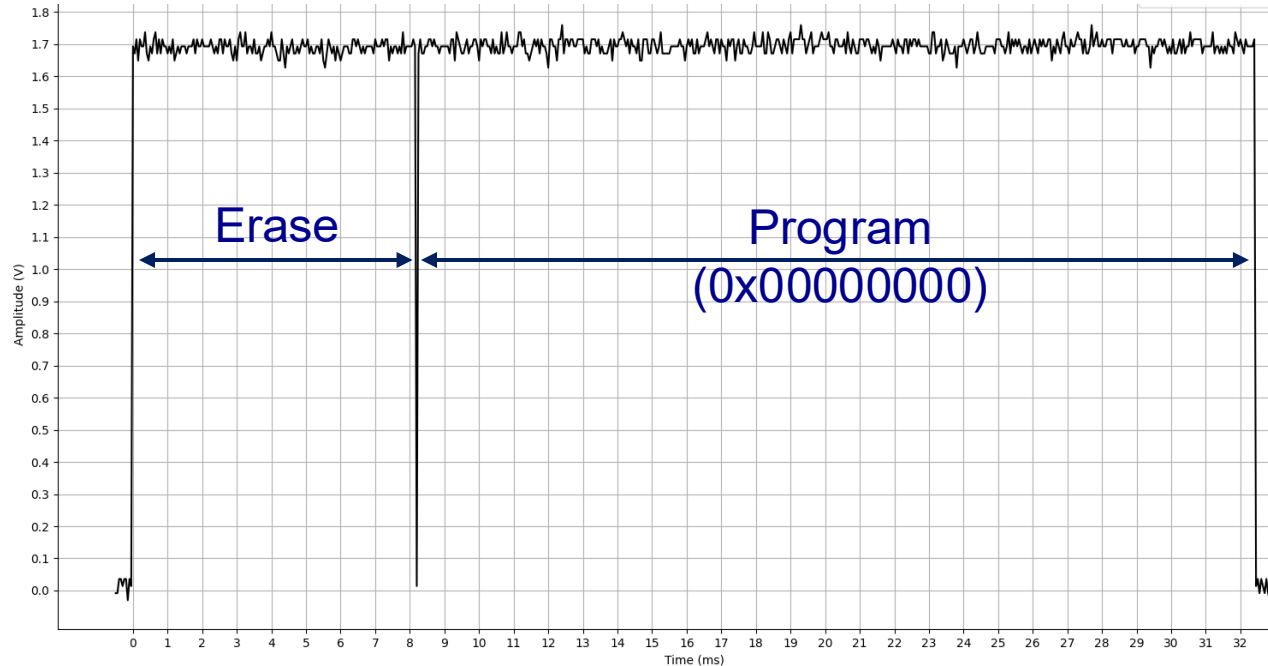
- ✓ ... **programmed** (i.e. written) **at word level**
- ✓ Using channel-hot-electron injection
- **Set to 0** (or 0x00000000 at word level)



Flash memory modes of operation

Erase + Program cycle time = **32 ms** (32 cycles in 1 second)

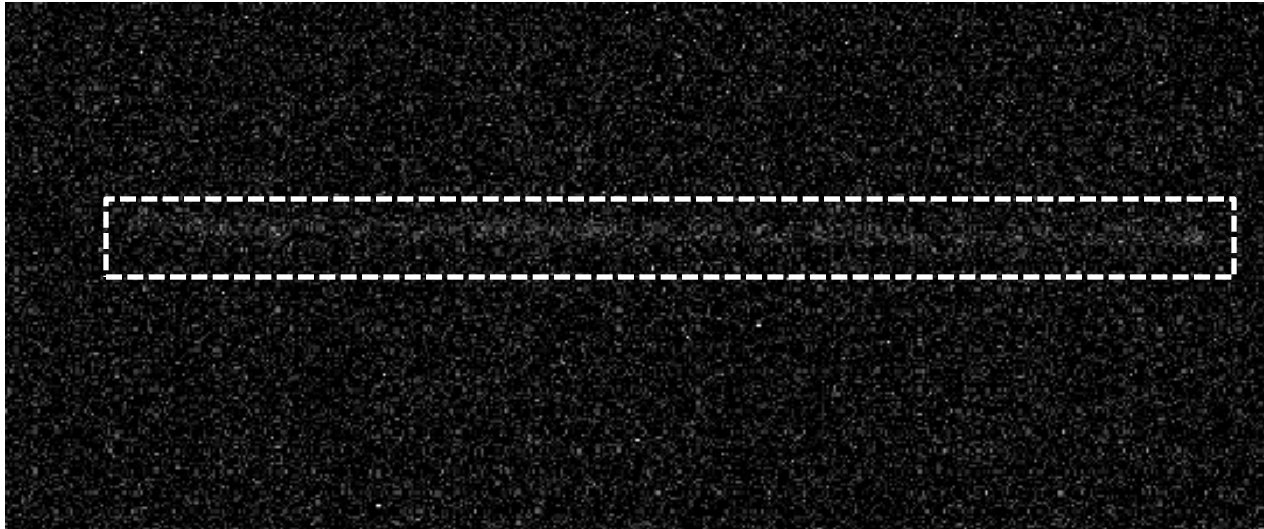
1 page



Flash photon emission – Exp. Results

Target 2 – erase + program

Number of erase + program cycles needed for the information to emerge from noise?

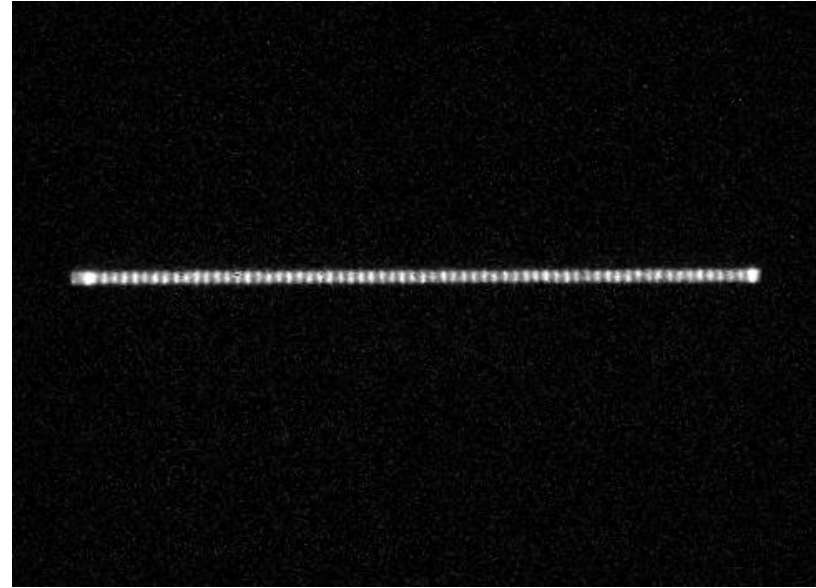
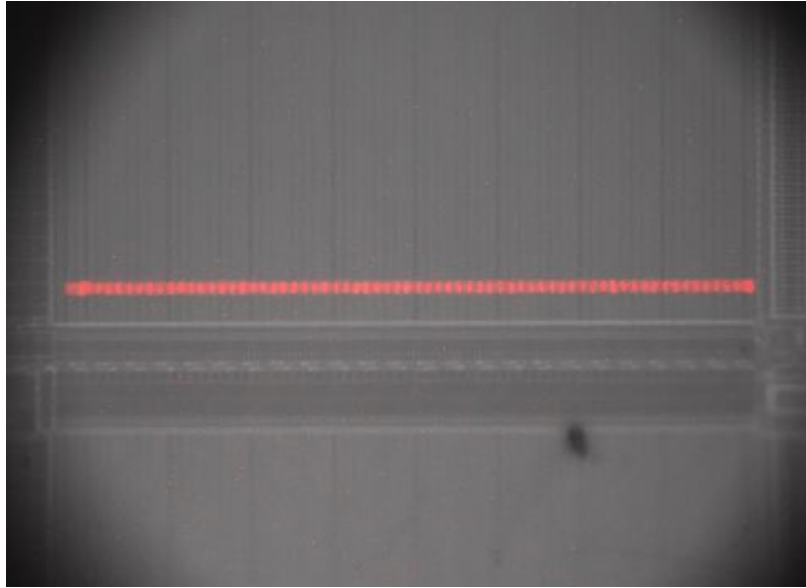


Photoemission map: 10 cycles, Page #255

Flash photon emission – Exp. Results

Target 2 – Data dependency

Target thinned down to 50 μm

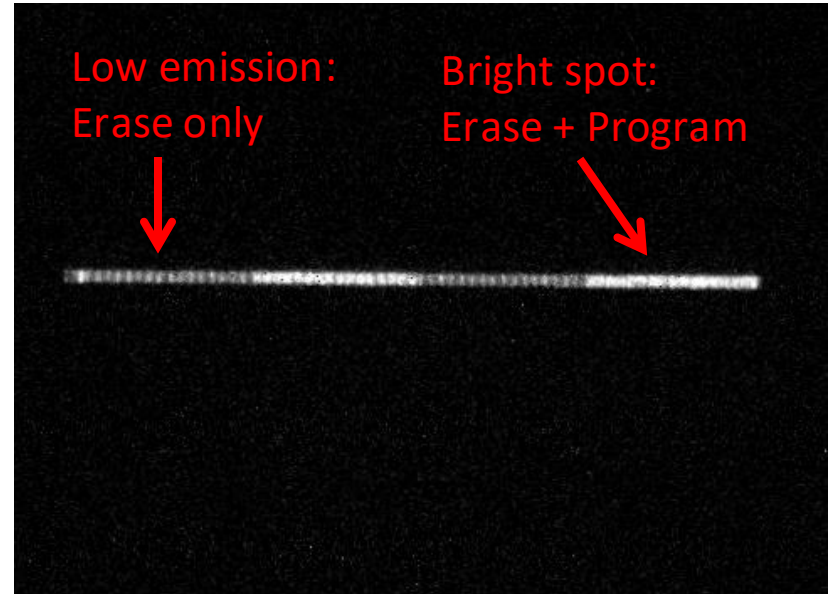
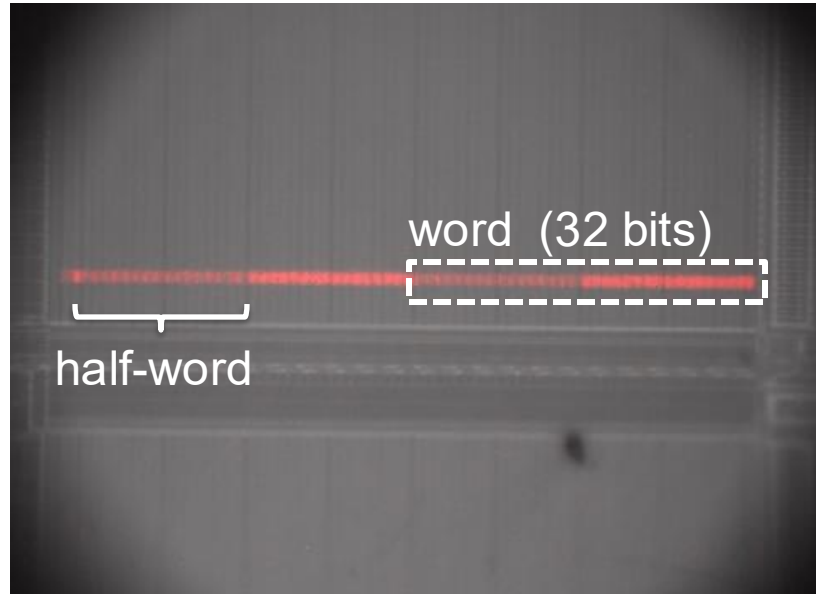


Page #120, 20x lens, exposure 2.5 s, program `0X00000000` (erase + program)
Overlay (left) & camera output (right)

Flash photon emission – Exp. Results

Target 2 – Data dependency

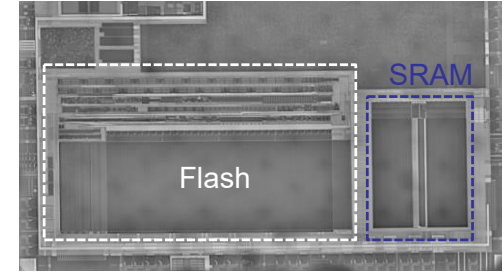
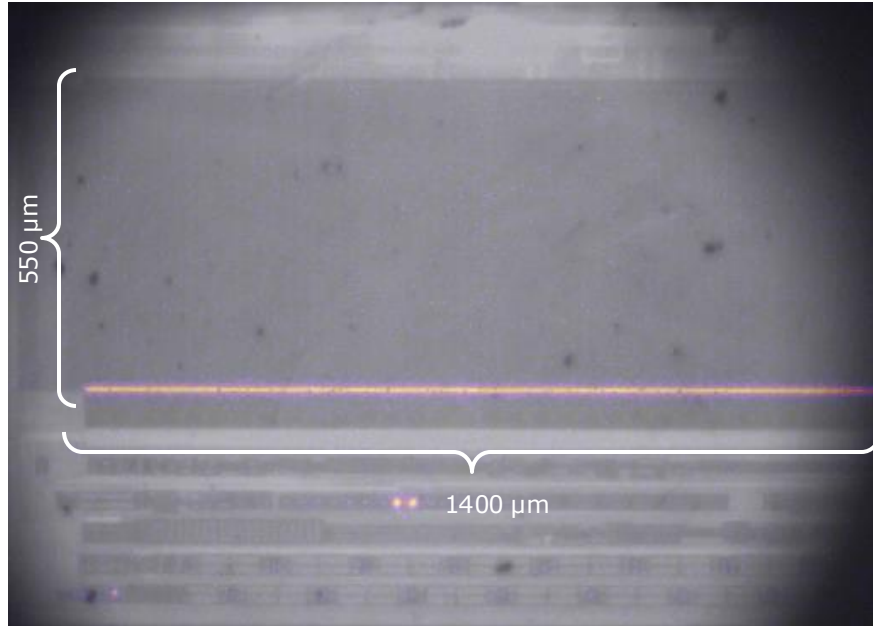
Target thinned down to 50 μm



Page #120, 20x lens, exposure 2.5 s, program `0X0000FFFF`

Flash photon emission – Exp. Results

Target 1 – Data dependency

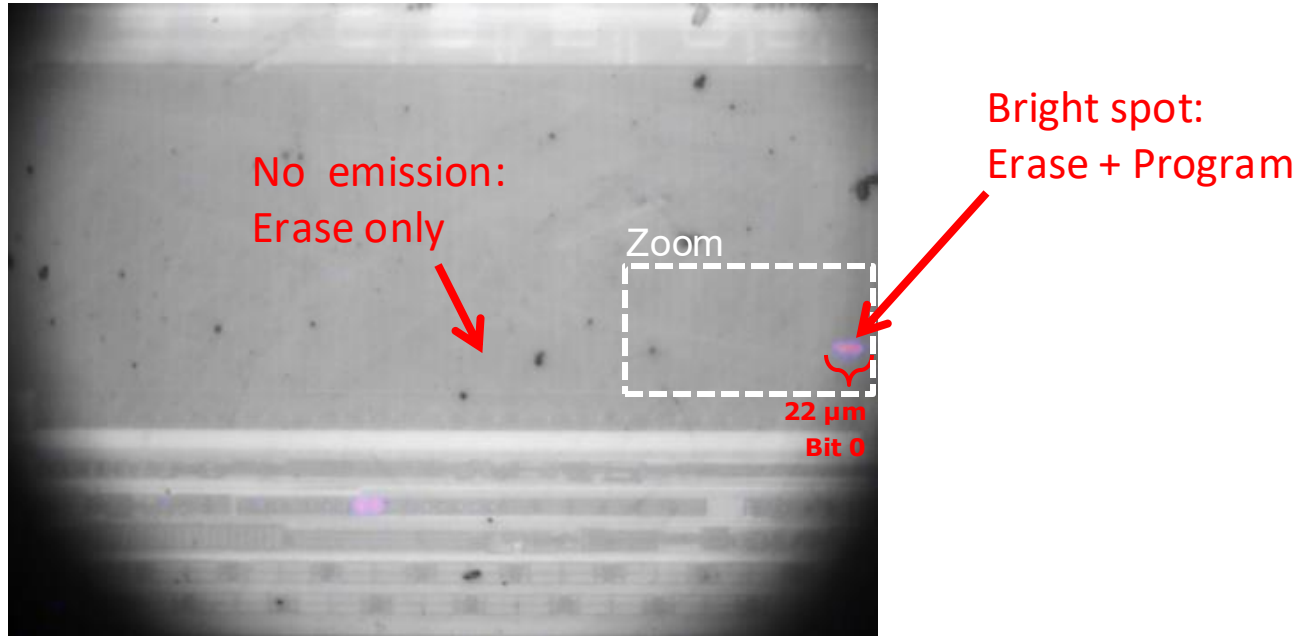


Page #127, 5x lens, exposure 2.5 s, program 0x00000000, 80 erase+prog. cycles

@: 0801FC00 – 0801FFFF

Flash photon emission – Exp. Results

Target 1 – Data dependency, Flash bits location

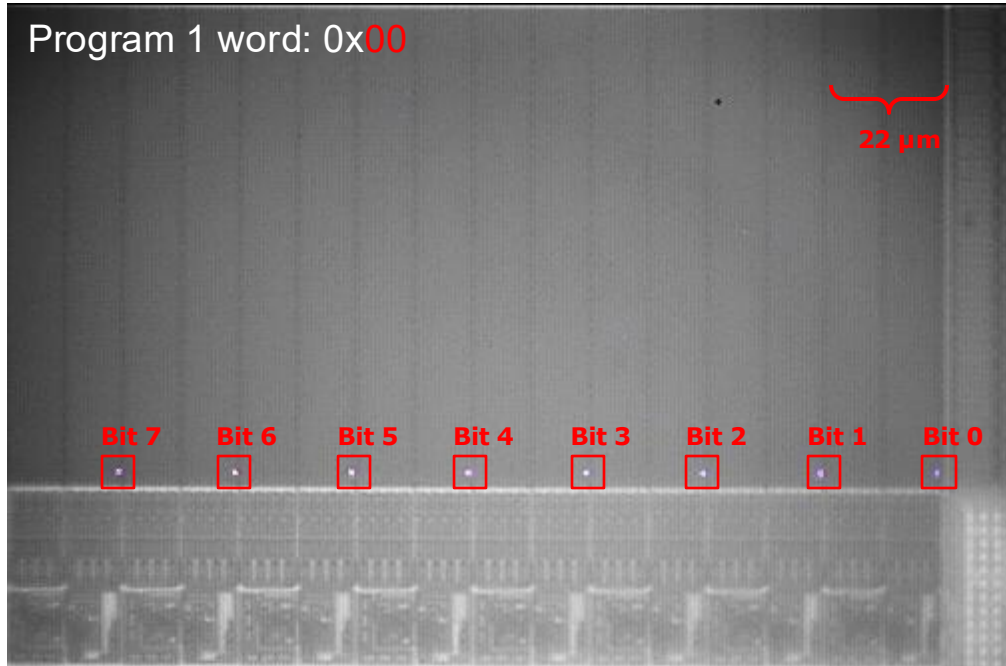


Page #127, 5x lens, exposure 2.5 s, program 0xFFFFFFFFE, 80 erase+prog. cycles

@: 0801FC00 – 0801FFFF

Flash photon emission – Exp. Results

Target 1 – Data dependency, Flash bits location → lsb (programming a single word)

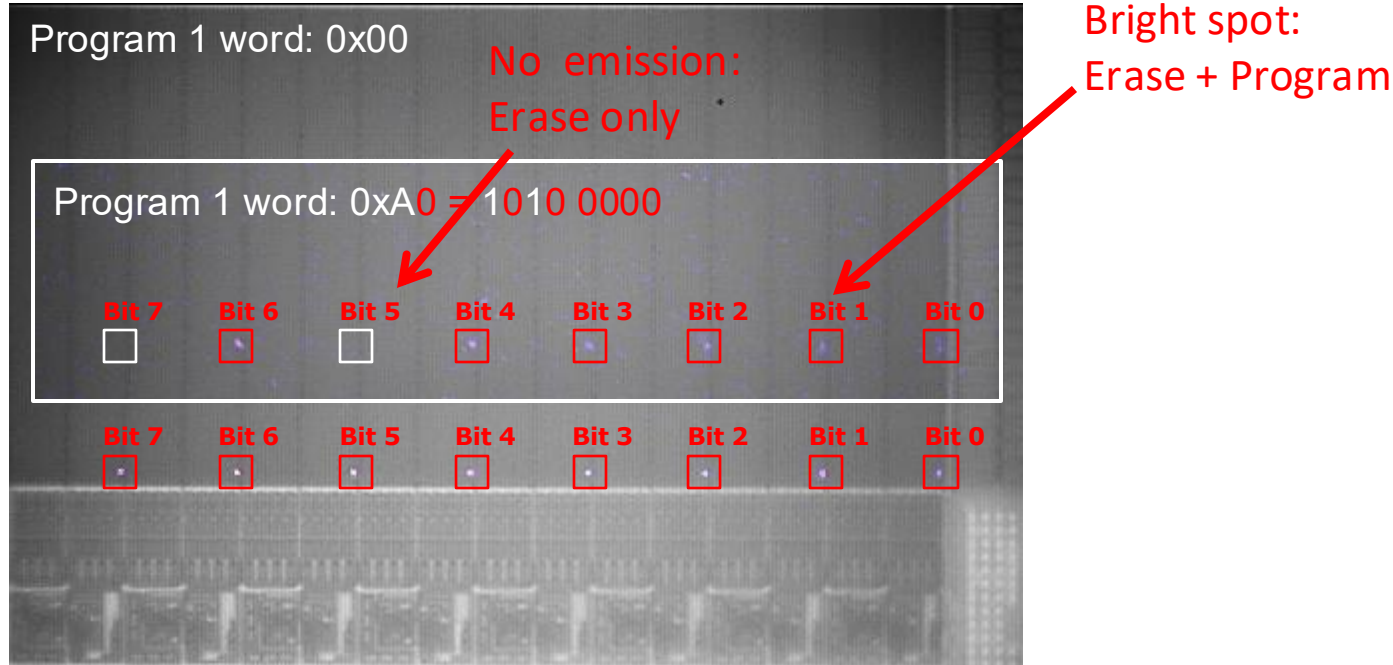


Page #127, 20x lens, exposure 2.5 s, program 0x00, 250 erase+prog. cycles

@: 0801FC00

Flash photon emission – Exp. Results

Target 1 – Data extraction at byte level



Page #127, 20x lens, exposure 2.5 s, program 0x00, 250 erase+prog. cycles

@: 0801FC00

Using PEM as a Hardware Attack Tool

- Photon Emission (PE) basics
- PEM for LFI facilitation
- Data extraction from SRAM memories
- Data extraction from Flash memories
- From limitations to a practical attack scenario

Practical attack scenario

Taking advantage of Flash modes of operation

✓ Page-level erase

Password identification routine

- ✓ Admin password (secret) stored in Flash memory
- ✓ User password stored in the same Flash page
- ✓ The user can update his own password → initiate a erase + program cycle

Data extraction attack

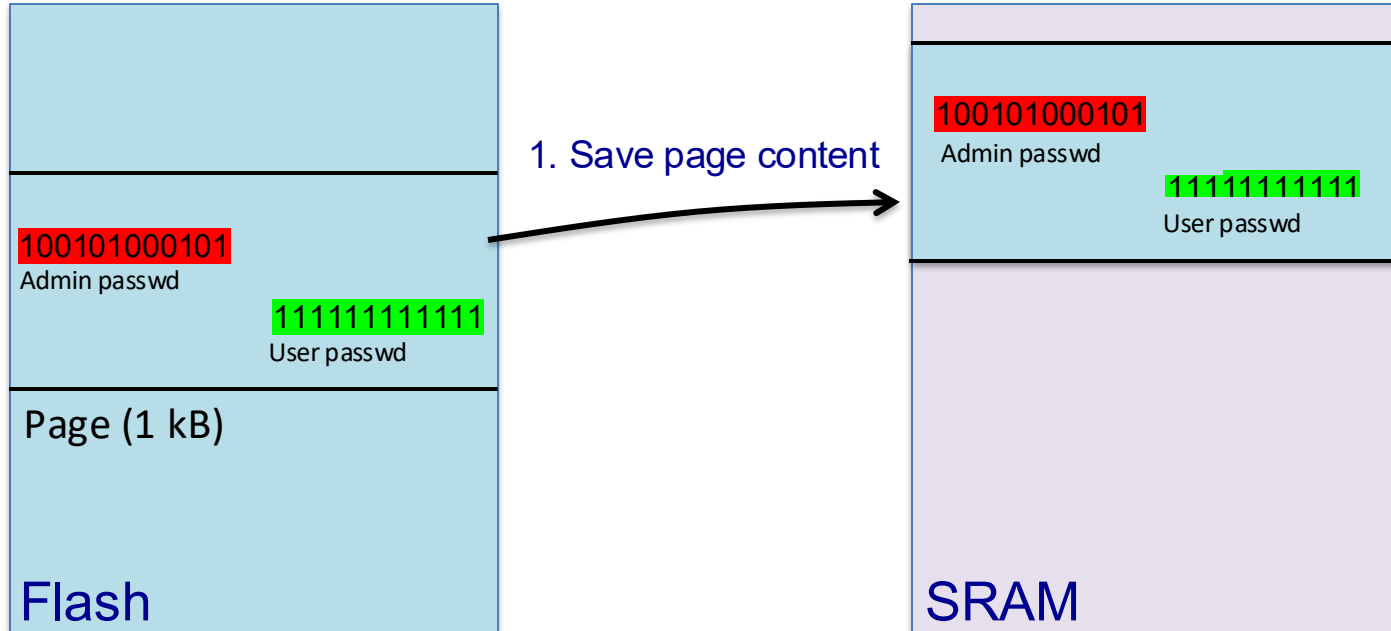
The user updates (continuously) his password → loops

- ✓ Admin passwd stored in RAM
- ✓ Admin passwd written back in Flash → a erase + program cycle

Practical attack scenario

User password update

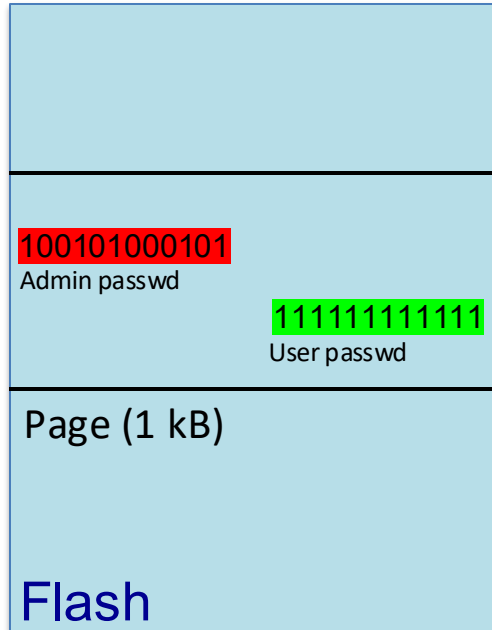
Flash Write → Erase (page-level) + Program



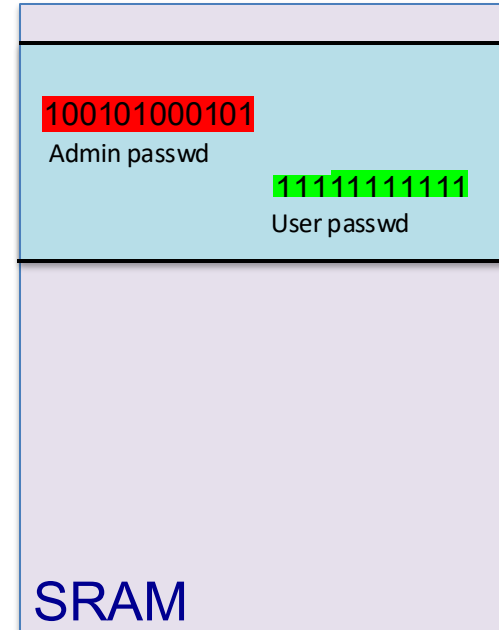
Practical attack scenario

User password update

Flash Write → **Erase (page-level)** + Program



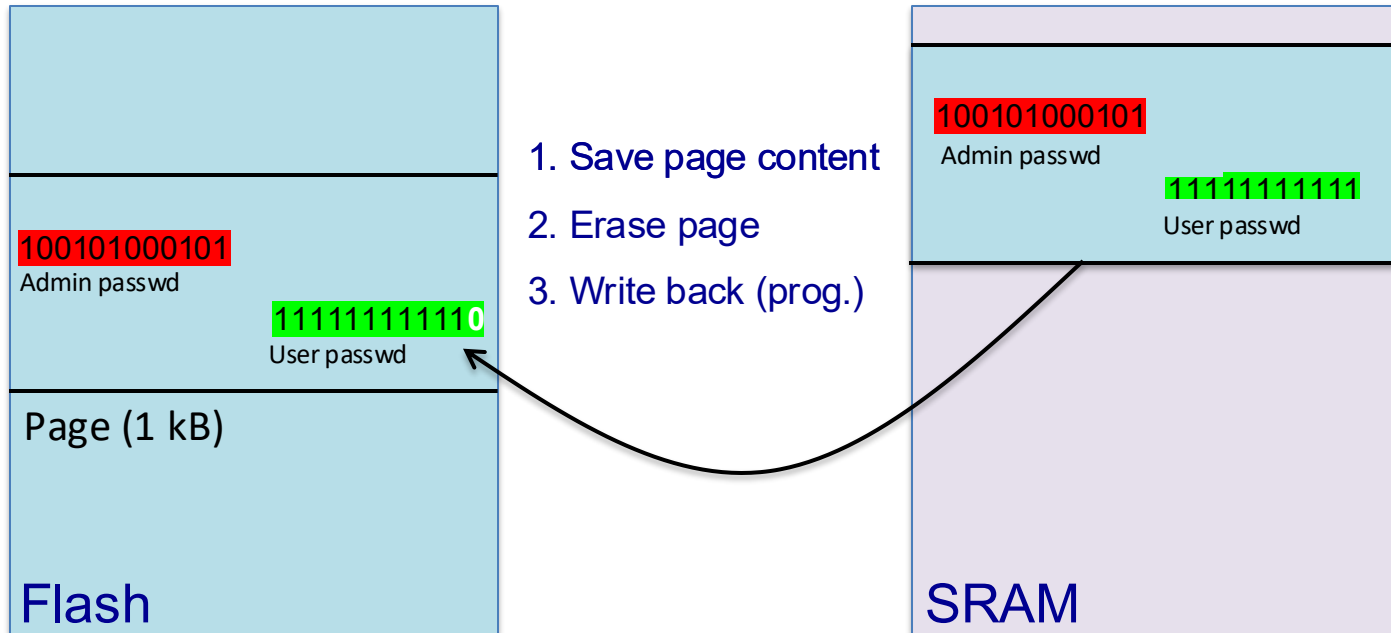
1. Save page content
2. Erase page



Practical attack scenario

User password update

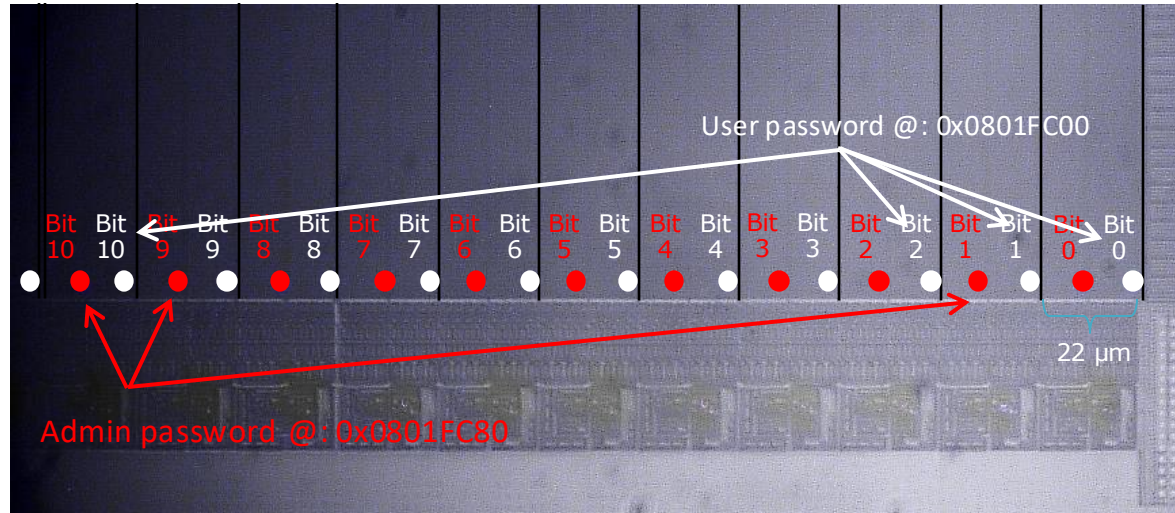
Flash Write → **Erase (page-level)** + Program



Flash att. scenario – Password extraction

Experimental results

- ✓ Admin password: @0x0801FC80
- ✓ User password: @0x0801FC00, 0xFFFFFFFFE

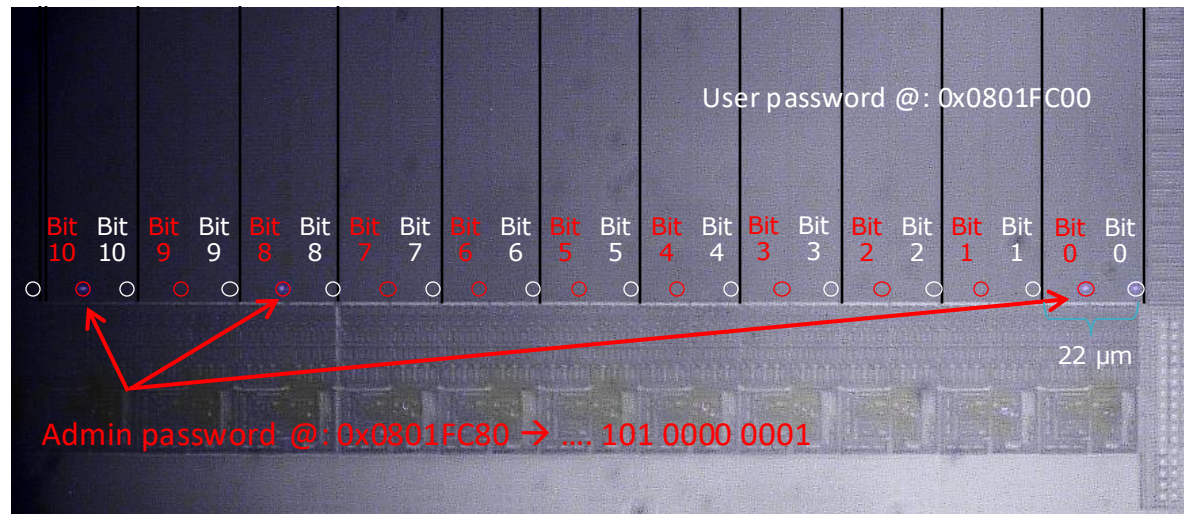


Photon emission map: 20x lens, exposure 100 s

Flash att. scenario – Password extraction

Experimental results

- ✓ Admin password: @0x0801FC80
- ✓ User password: @0x0801FC00, 0xFFFFFFFFE



Photon emission map: 20x lens, exposure 100 s

Conclusion

PEM reverse engineering capabilities

- ✓ POI identification → accelerate LFI attacks
- ✓ Flash/SRAM architecture: WL, BL, bit cells (**Write** or read)
- ✓ Analog blocks & FG transistors → data extraction

PEM constraints

- ✓ Execution of code loops → ok in white box model (consistent with POI identification)



Data extraction from memory at read/write time

Scaling → analog blocks are strong emitters of photons (adv. tech.)

Practical attack scenarios do exist

Contact:
dutertre@emse.fr

To go further:

- ✓ H. Perrin et al., [Betrayed by Light: How Photon Emission Microscopy Empowers Register Bit-level Laser Attacks on Microcontrollers](#) , HOST 2025
- ✓ R.S. Lima et al., [When data shines - leaking data from microcontrollers through photon emission analysis](#), ASHES 2024
- ✓ R. S. Lima et al., [Reverse-engineering and data extraction from SRAM using photon emission analysis](#), IEEE PAINE 2024