Nano-Enigma:
Uncovering the Secrets Within eFuse Memories

Finding and Reading Secrets from eFuse-Based Memories



Introduction

Embedded Systems Security Testing Team

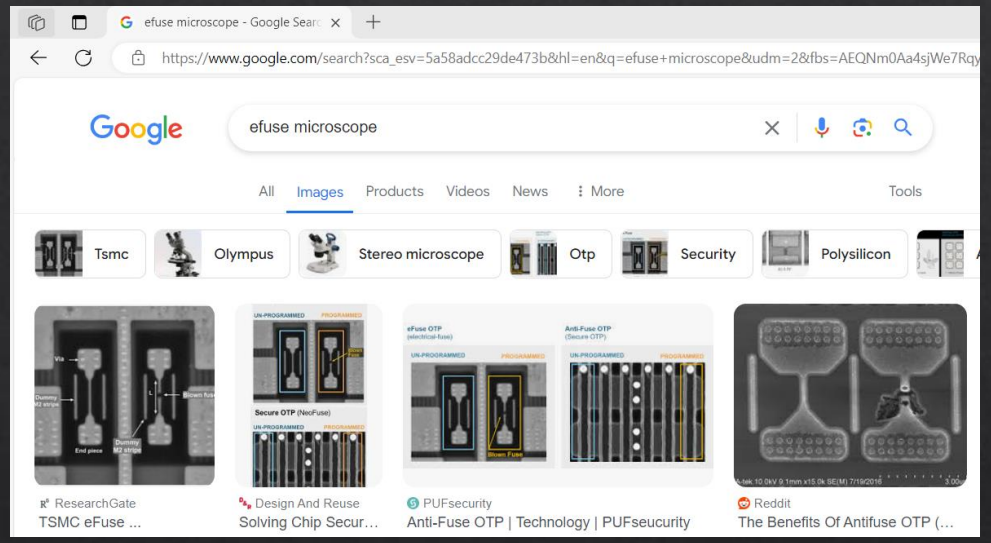
Prague – Czech Republic

Martin Petran
Michal Grygarek
Hayyan Ali

Overview

Answering basic questions:

- What is an eFuse and what are the benefits of using it for security?
- What exactly are these used for in security context?
- How are these protected from reading?



Execution of the attack



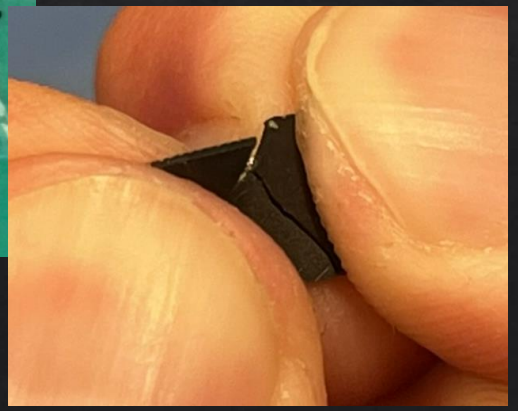
Sample preparation - Decapsulation

- Thermal – (100 USD)
  - Brute force, High risk of damage
- Chemical decapsulation - (~100 – 10.000 USD)
  - Gentle technique, Good results, Adequate equipment is necessary
- Laser etching - (~1.000 – 10.000 USD)
  - High risk of burning DIE, Good combination with the chemical method
- Plasma / ION etching - (~100.000 - 1.000.000 USD)
  - Gentle and precise technique













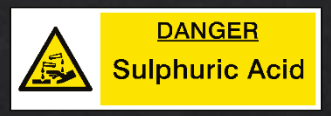
Chemical Decapsulation

- We chose Chemical Decapsulation due to its availability
  - In combination with laser etching, it delivers excellent results.
  - It completely removes all housing.
  - The quality of the result is repeatable.

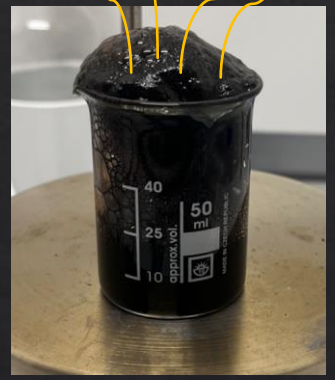
Protective equipment is necessary







Evolving gas SO_X

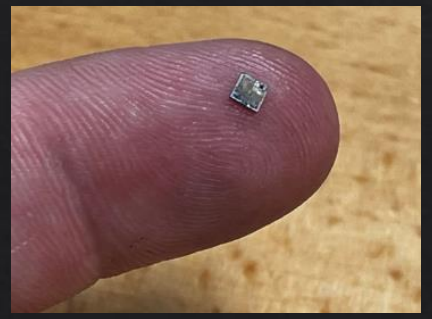


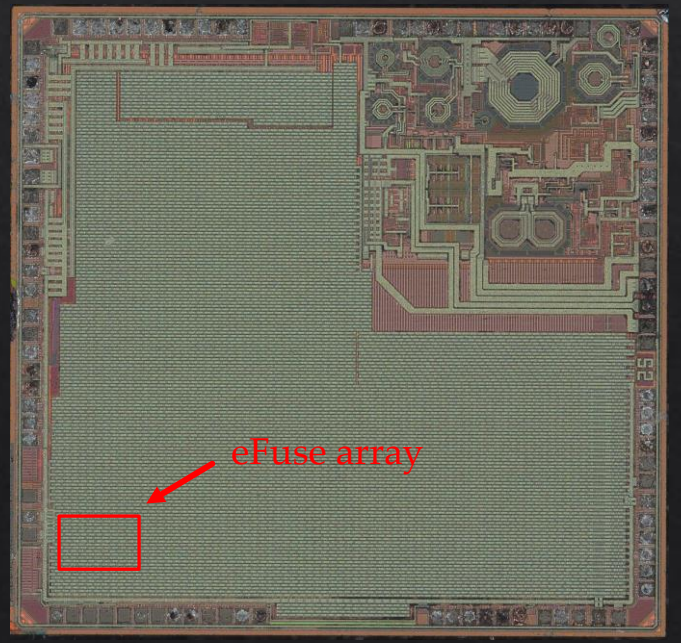
Heat up to 250°F

H_2SO_4 - **96%**



Rinsing residues





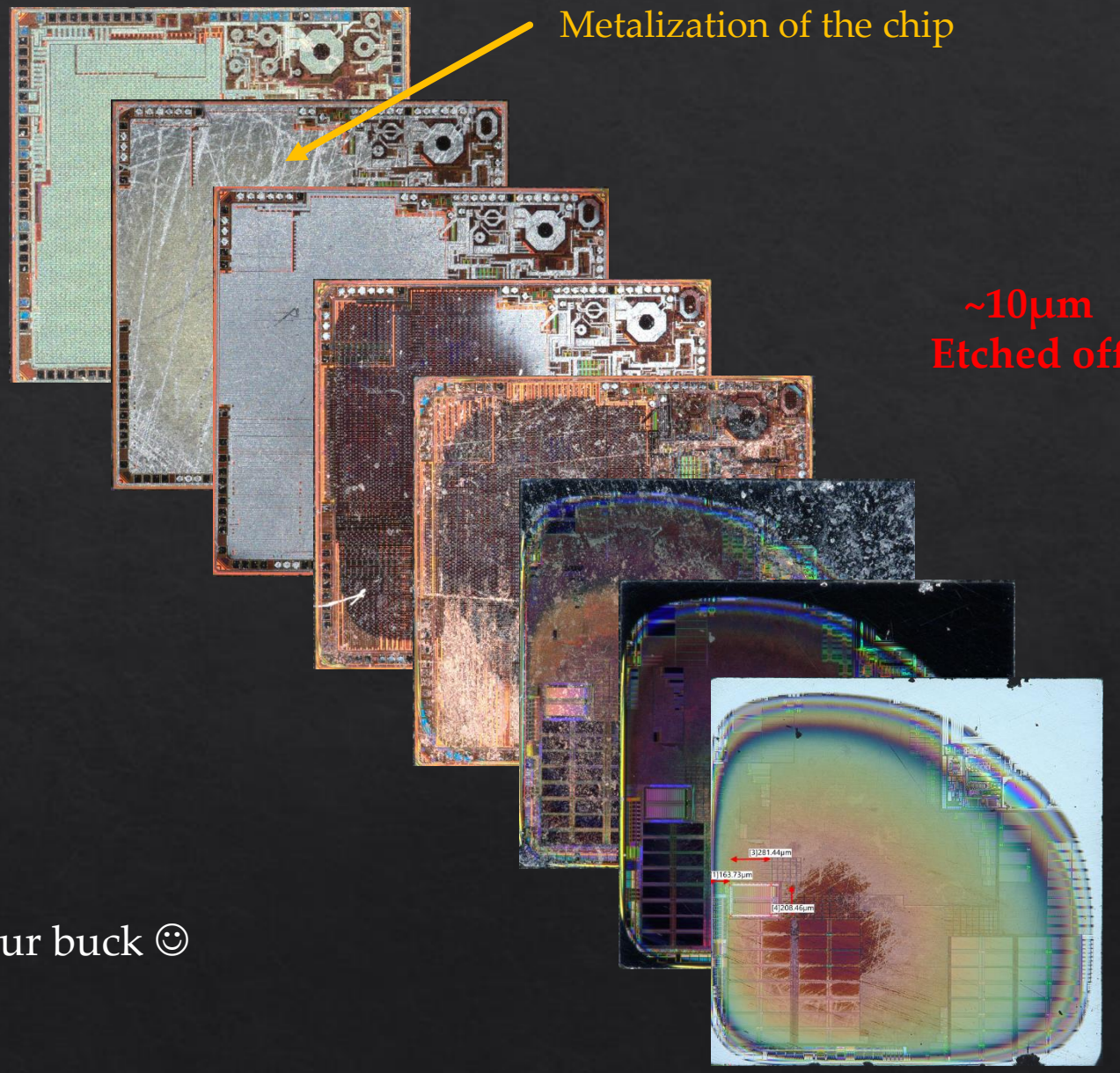
Perfectly clean DIE

Delayering

- Mechanical (100 USD)
- Chemical (250 USD)
- Laser ablatition (~50.000 USD)
- FIB / Plasma (~2.000.000 USD)

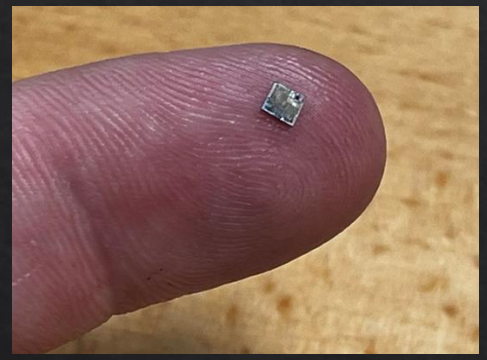
- We use mechanical delayering because
  - Easy accessibility
  - Cheapest
  - Minimum equipment required

Mechanical way - a lot of bang for your buck ☺

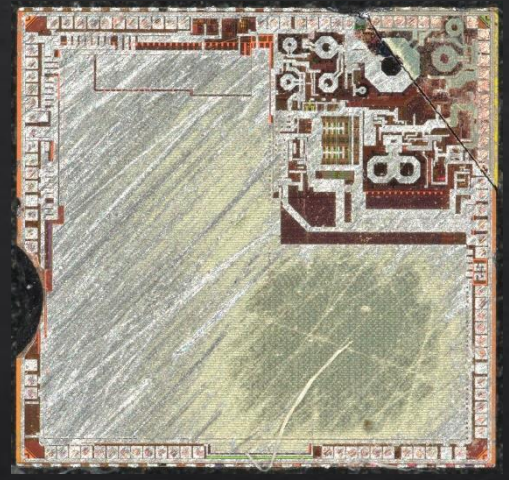


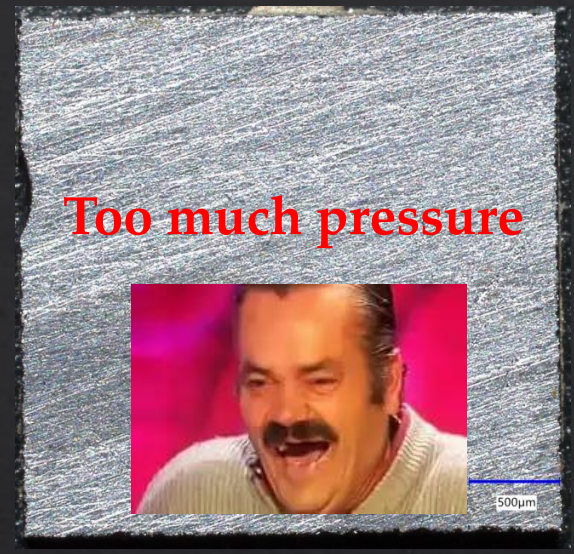
Delayering – wetstone

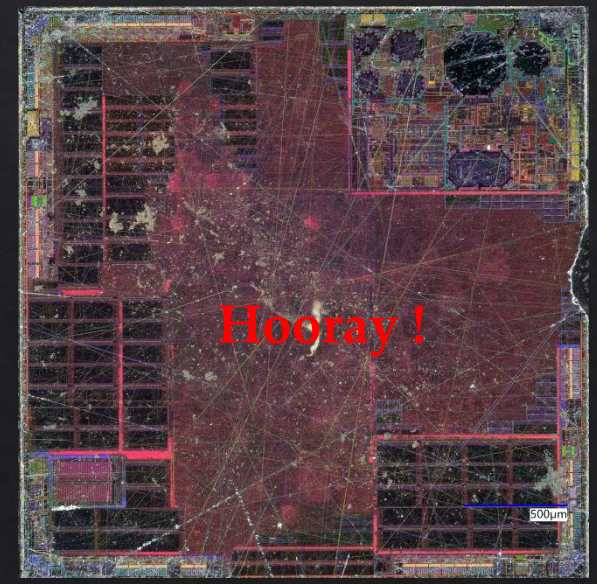
- It is necessary to have an abrasive surface with the highest possible grit size
- A capable worker prepares one sample within 60 min
- The skin on the fingers of one hand is enough for delayering 3 ESP32 chips.

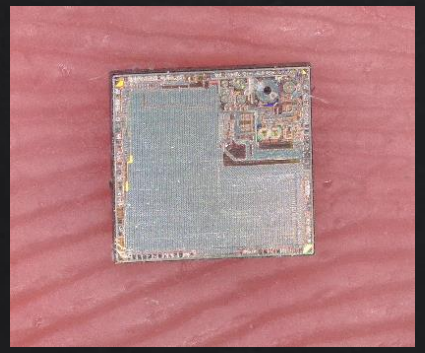














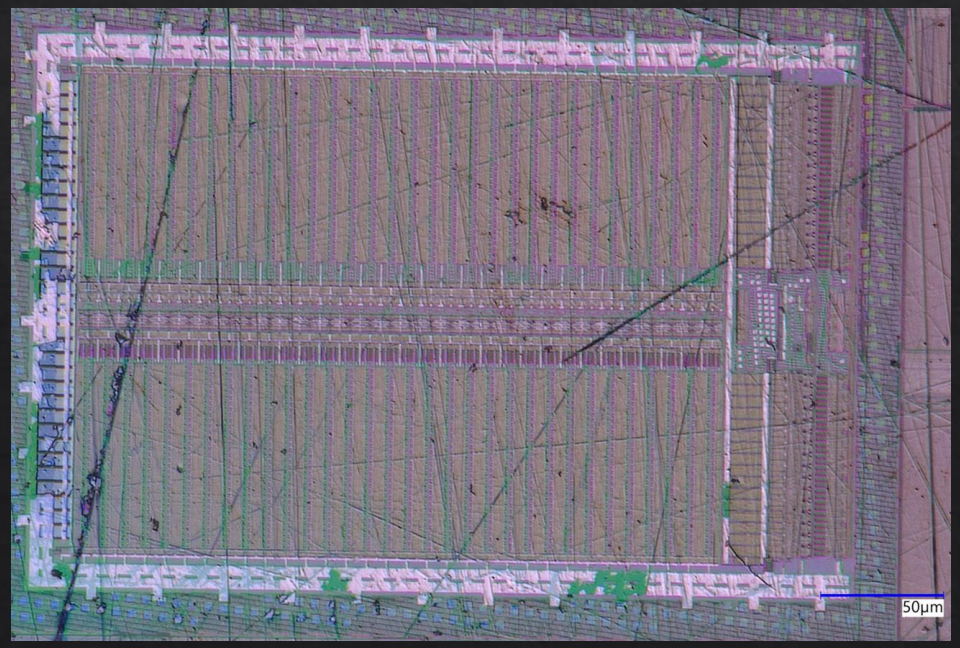


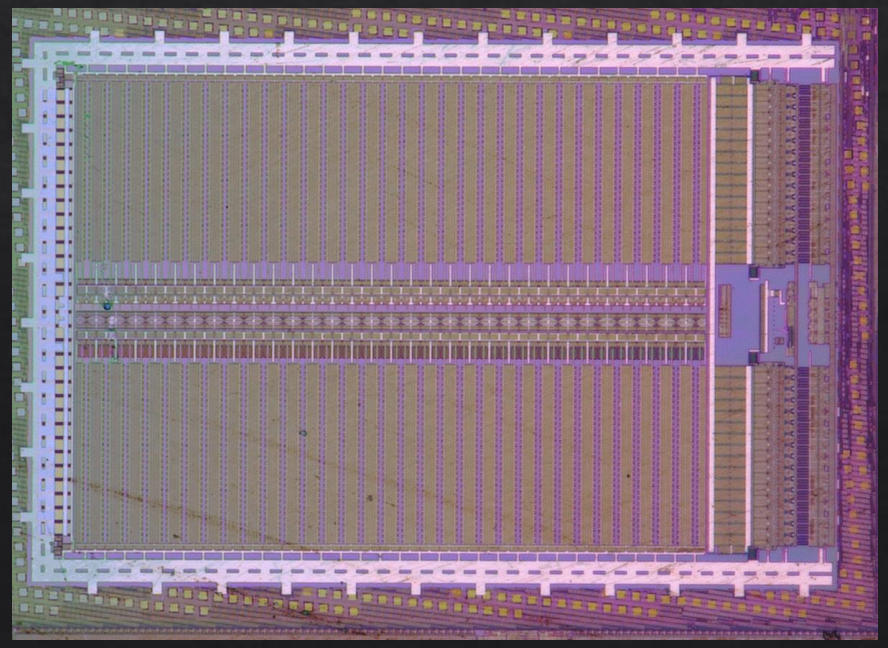
Delayering – polishing

- Initial etching on a wet stone with a grit of 8000 JIS (2µm)
- The last phase needs to be finished on the diamond lapping foil (<0,5µm)







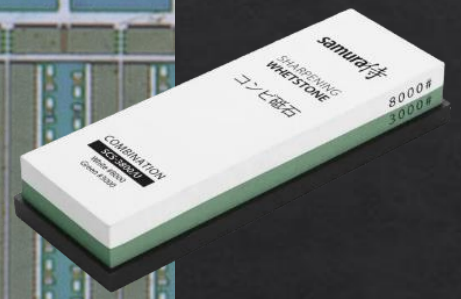


Delayering – methode comparing

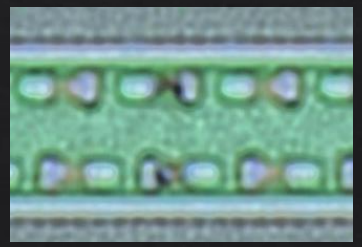


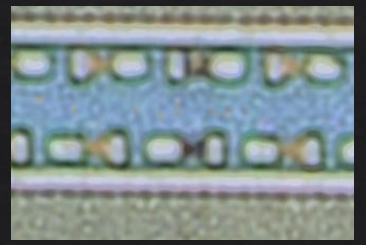
Mechanical etching machine ~ 65.000 USD

Wet stone + lapping foil ~ 100 USD





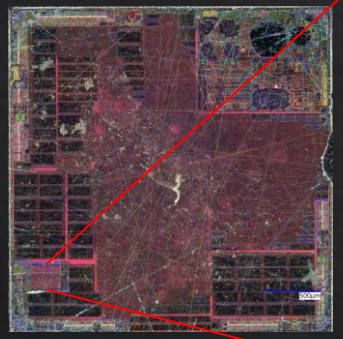


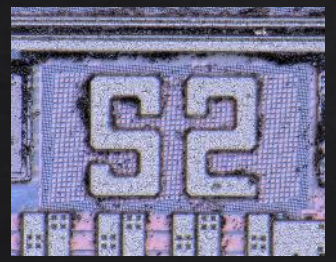


Localization of the On-chip eFuse field

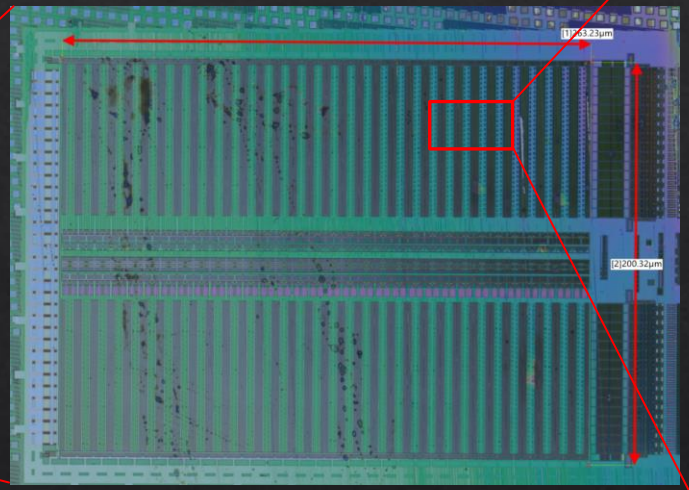
- We evaluated the eFuse bank identity with the ESP32 S2 / S3 / C6 model.
- eFuse is essentially a fuse

Magnification 80x





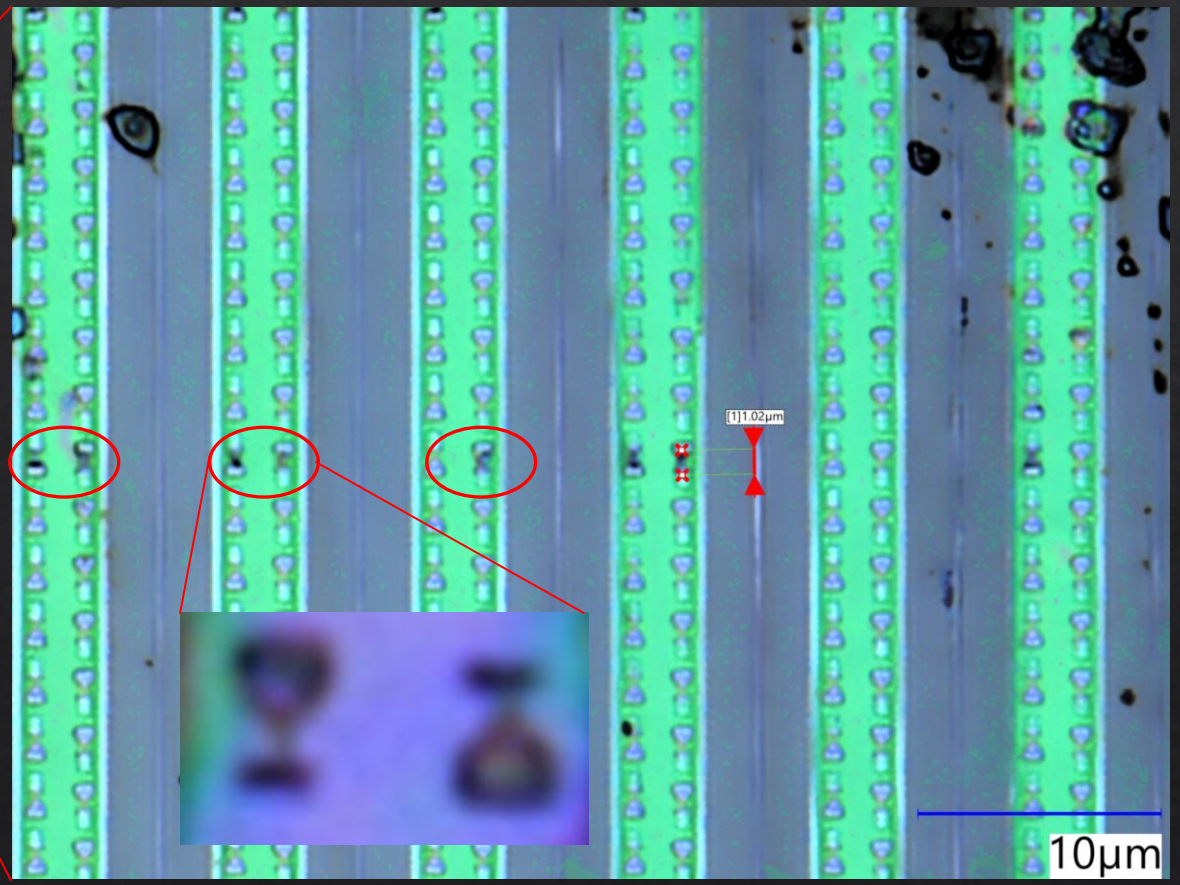
Magnification 700x



View of the complete 4096-bit eFuses bank

PROM bits

Magnification 2000x

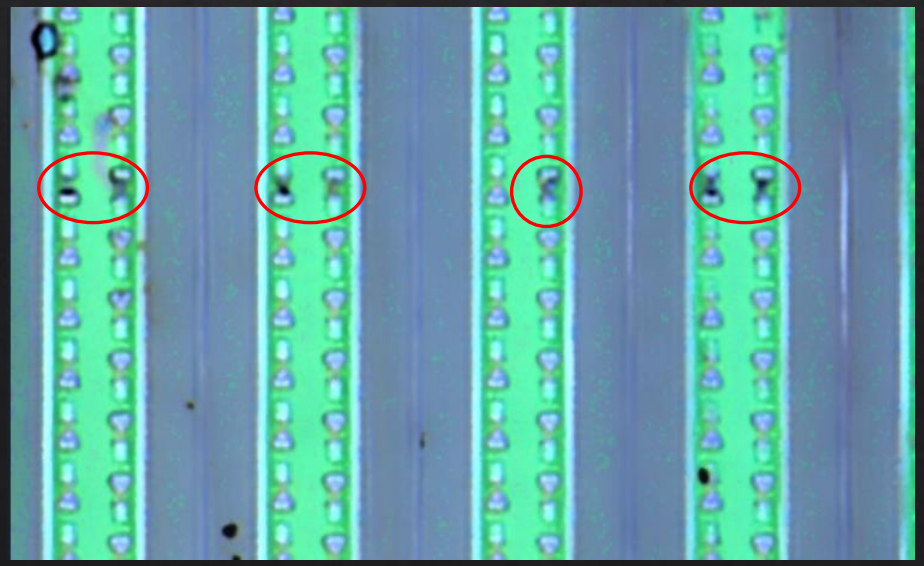


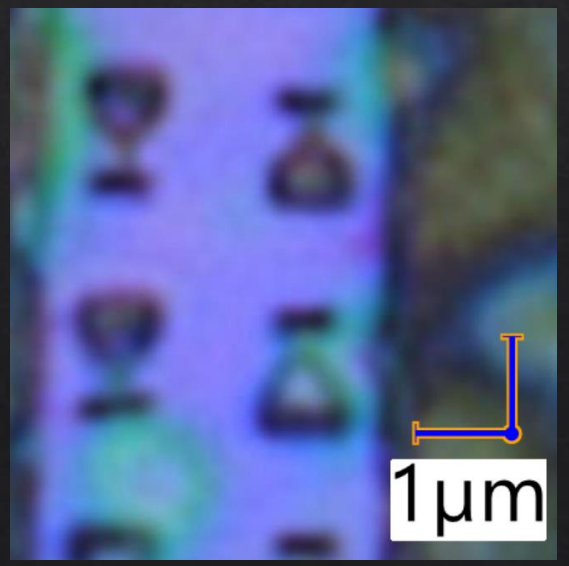
Detail of individual eFuses, here we can see that optical microscopy no longer provides the required level of detail

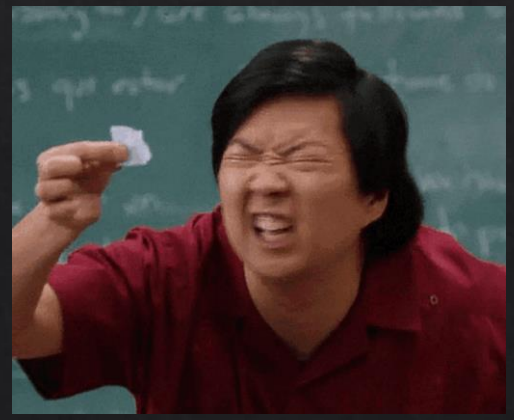
Optical microscopy observing - limits

- Used equipment – Keyence VXH-7000
  - Maximum resolution ~ 300nm (~2000x magnification) – Physical limits
- Sufficient for locating the eFuse bank
- On a well-prepared sample, even burned eFuses can be seen
  - The resolution is insufficient for reliable state reading of eFuses
    - A human hair is 100 times bigger than an eFuse



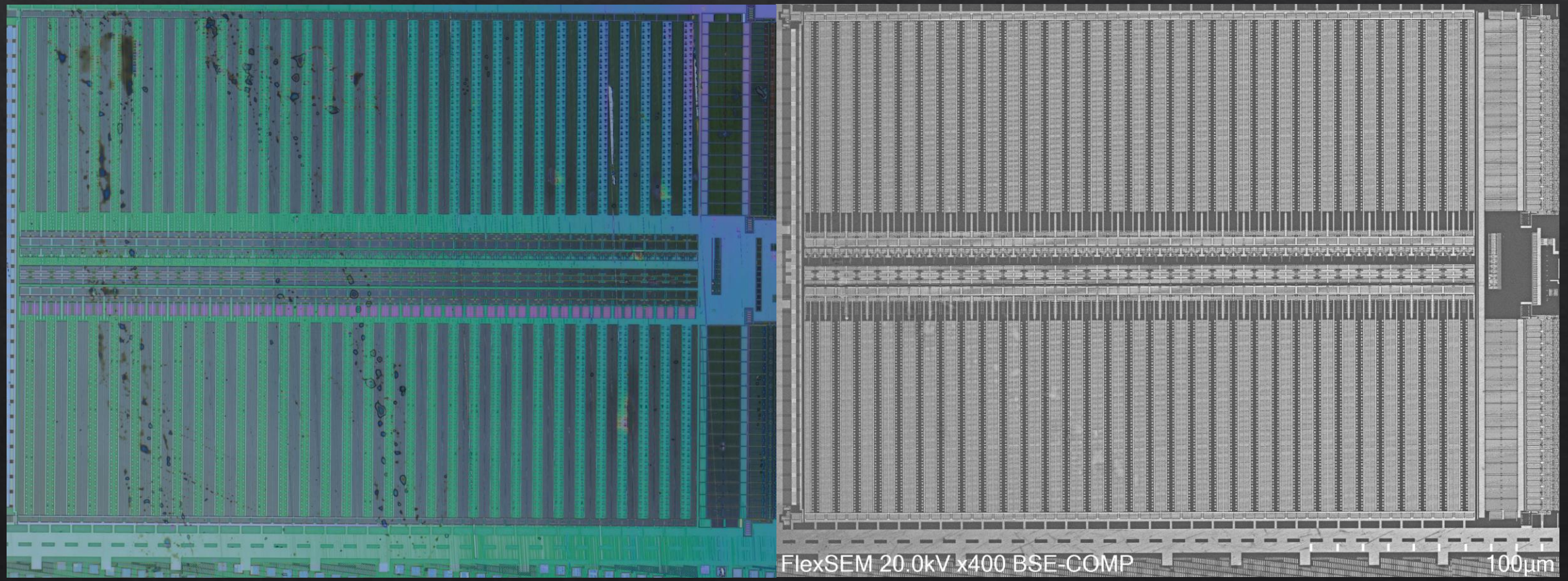




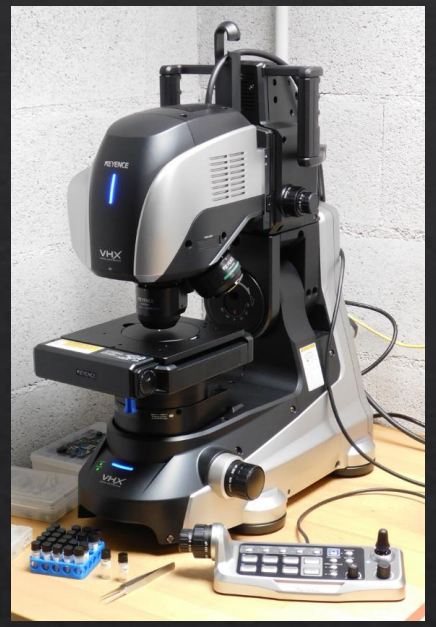


We need Scanning Electron Microscopy

- We need more advanced imaging technique
- SEM provides higher resolution



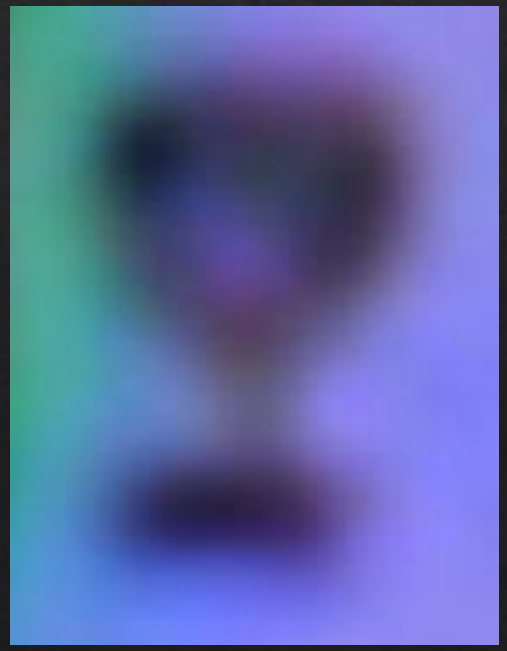
Optical vs. Scanning Electron Microscopy

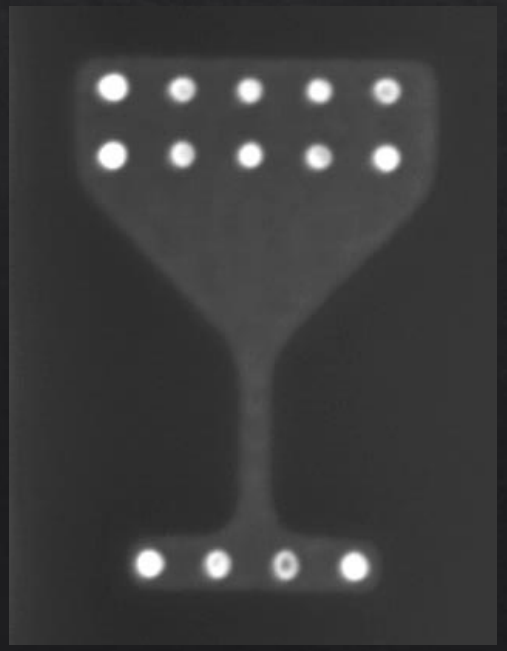


- SEM provides ~100 times higher resolution
- Structural changes are more observable
- eFuse is 100 x smaller than a human hair
- The level of detail is significant



1µm

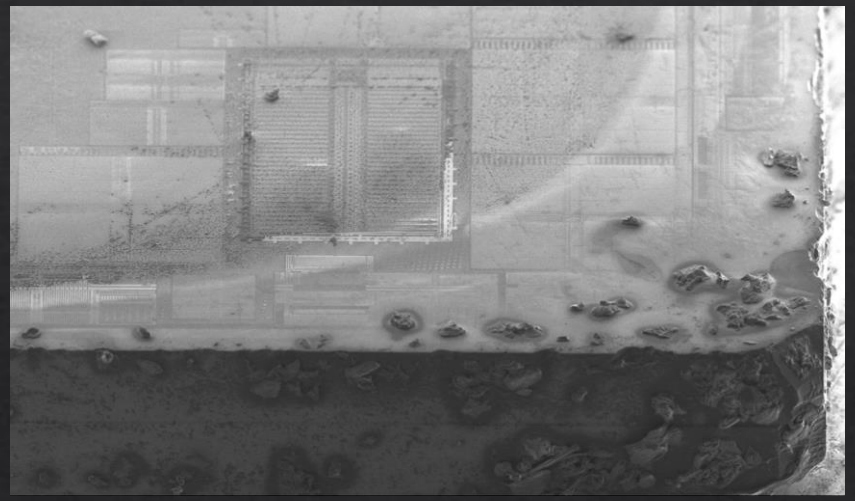


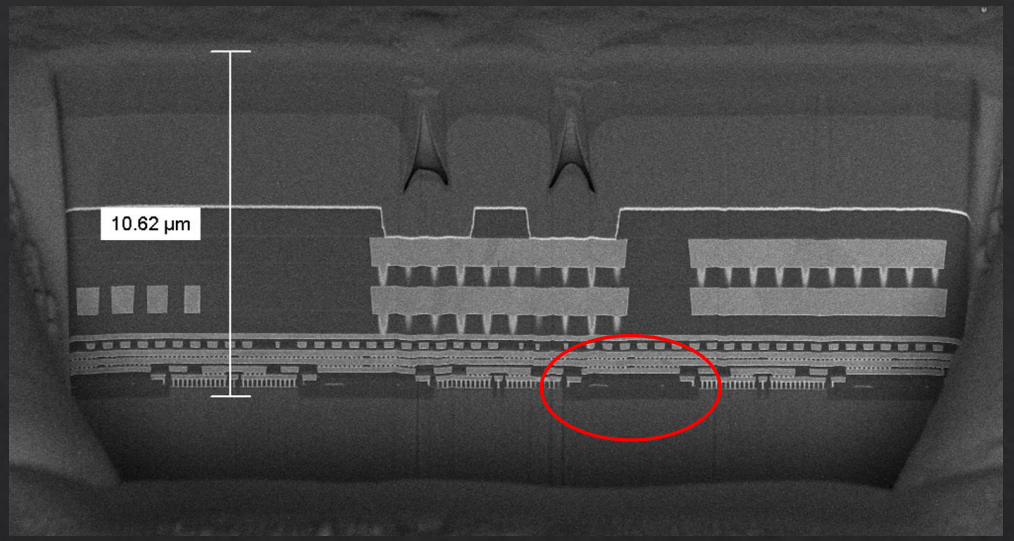


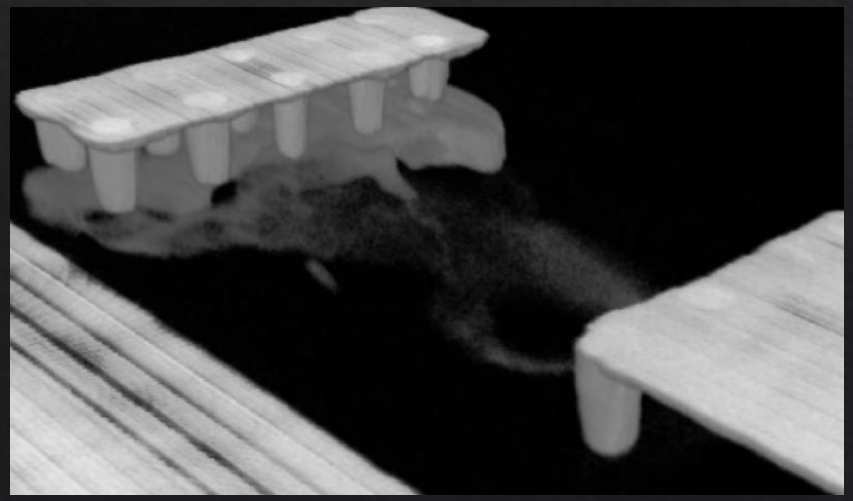
Optical
Resolution ~300nm
Magnification 2000x

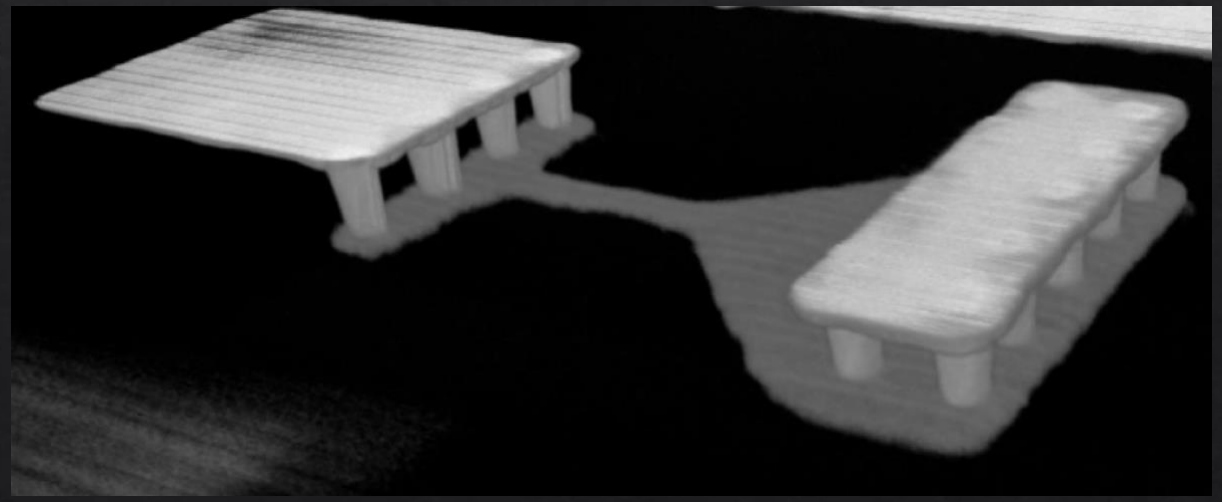
SEM
Resolution ~5nm
Magnification 100.000x

Electron microscopy observe details



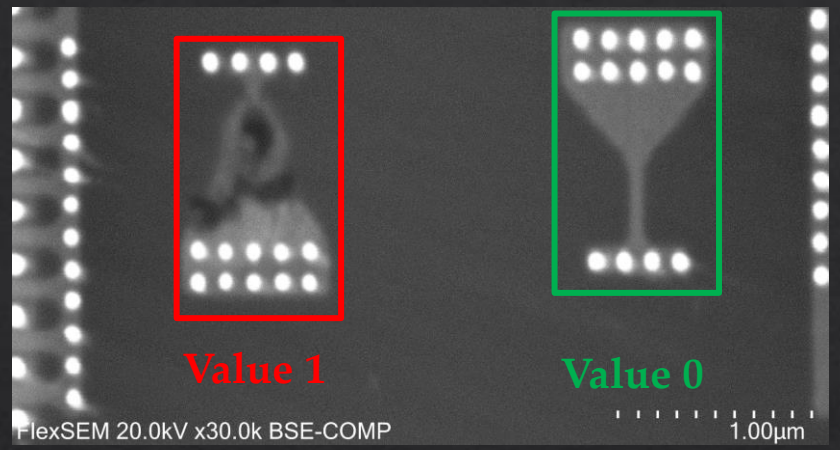


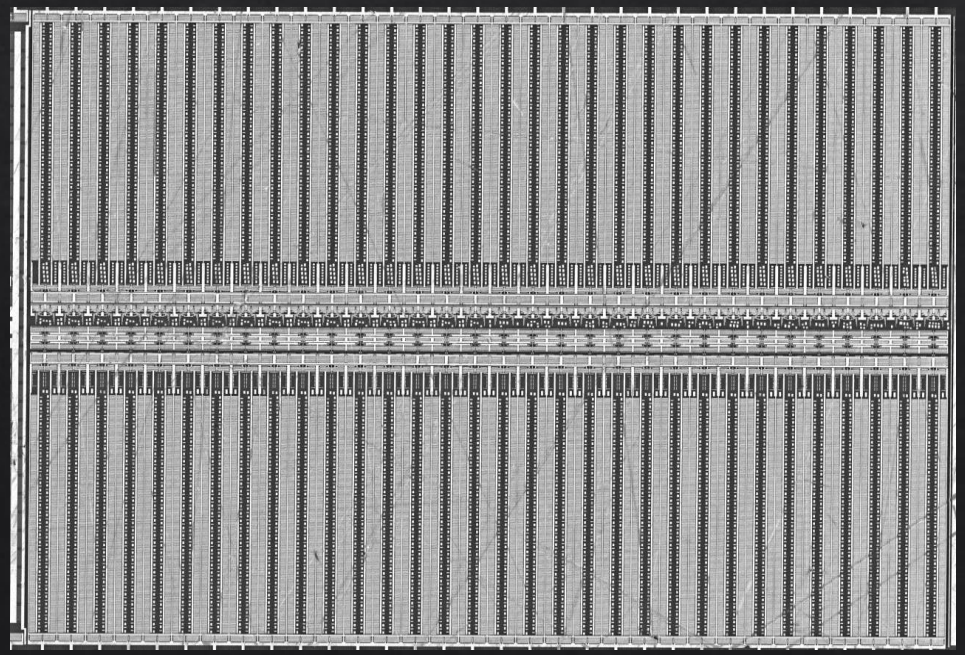


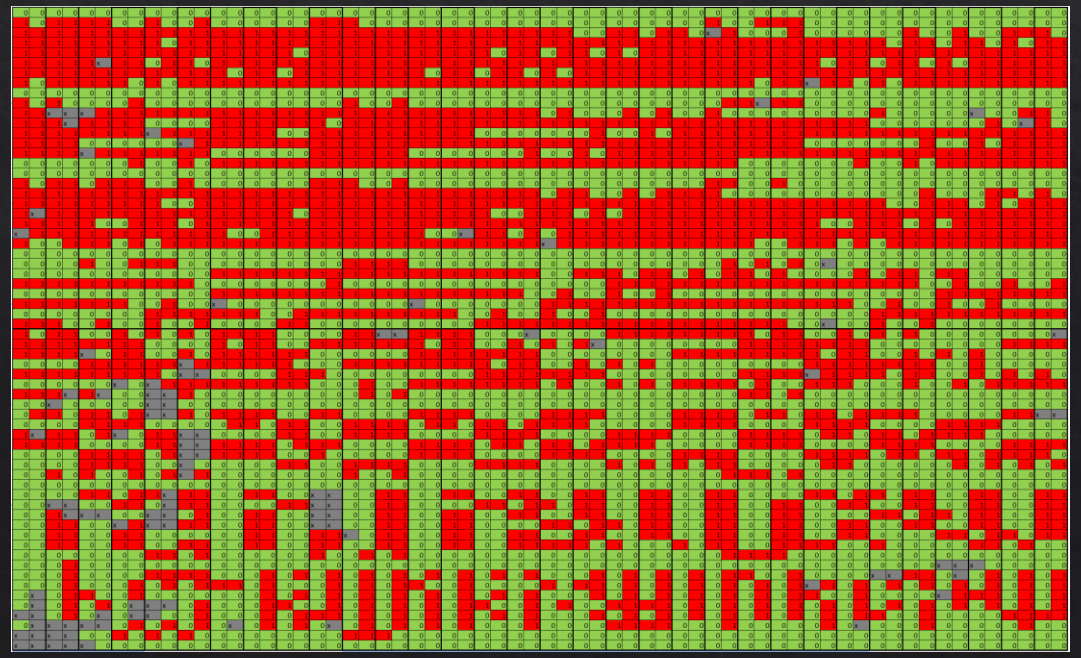


eFuses data extraction – „visually reading“

- Extremely time consuming, intense focus
  - Reading a 4096-bit bank takes ~10 hours of focused work
- In the initial stages, about ~20kb of eFuses were read visually ☺







Manually transcribed into Excel

Memory layout evaluation

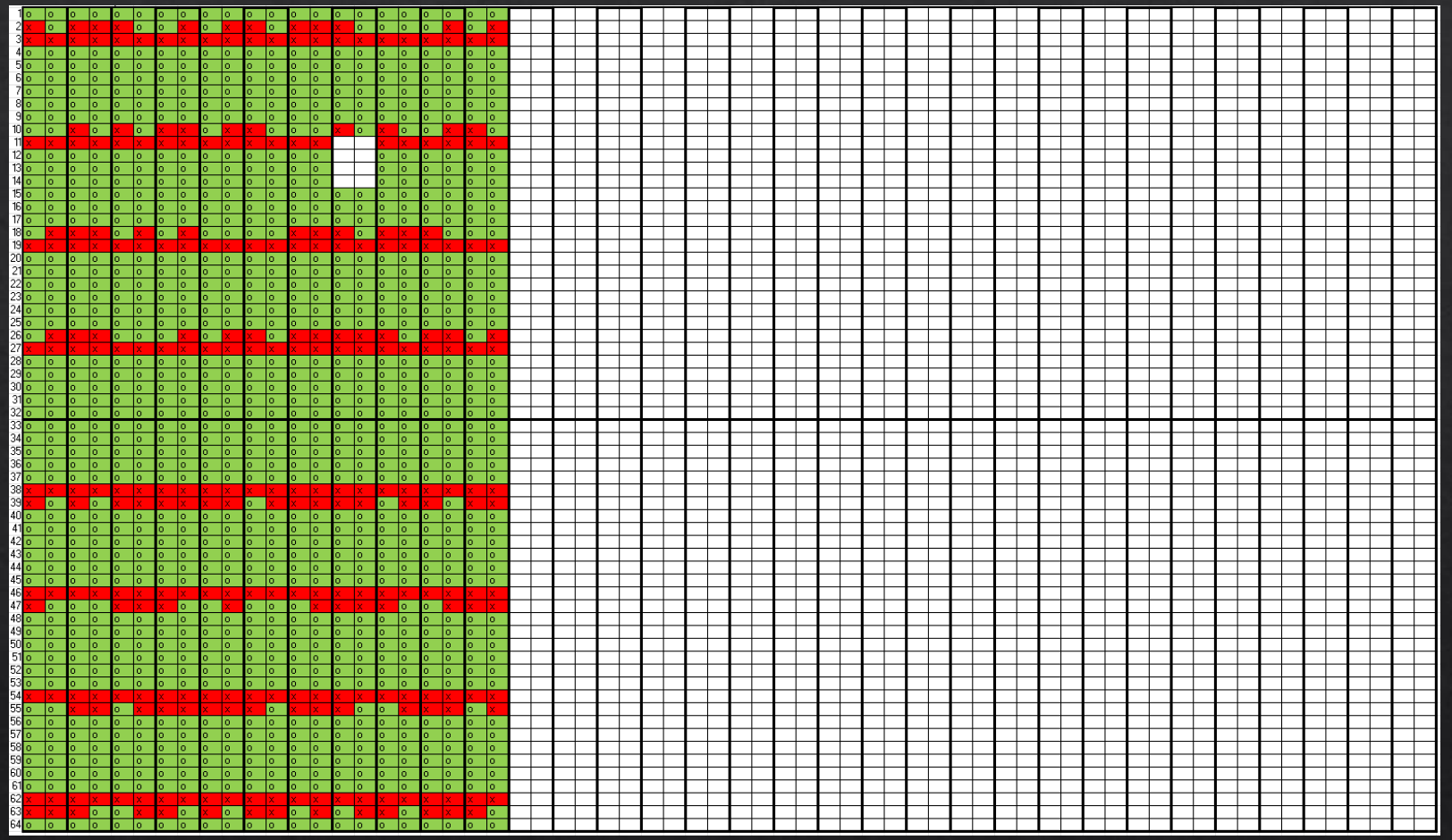
- **What we know?**
- The documentation mentions:
  - Memory is organized in blocks.
    - Total of 11 eFuse blocks
    - Blocks 0-2 are used for system purposes
    - The rest of the blocks are user programmable
  - There are 4 backups of BLOCK0 and BLOCK1-10 are using RS coding scheme for automatic error correction (adds 12 bytes to each block).
- **What we want to know?**
  - How is the eFuse array physically mapped to logical blocks?

Memory layout evaluation

Manual Experiment #1:

Burn only to user writeable first block values 0xFF.

Was not the best idea to begin with …



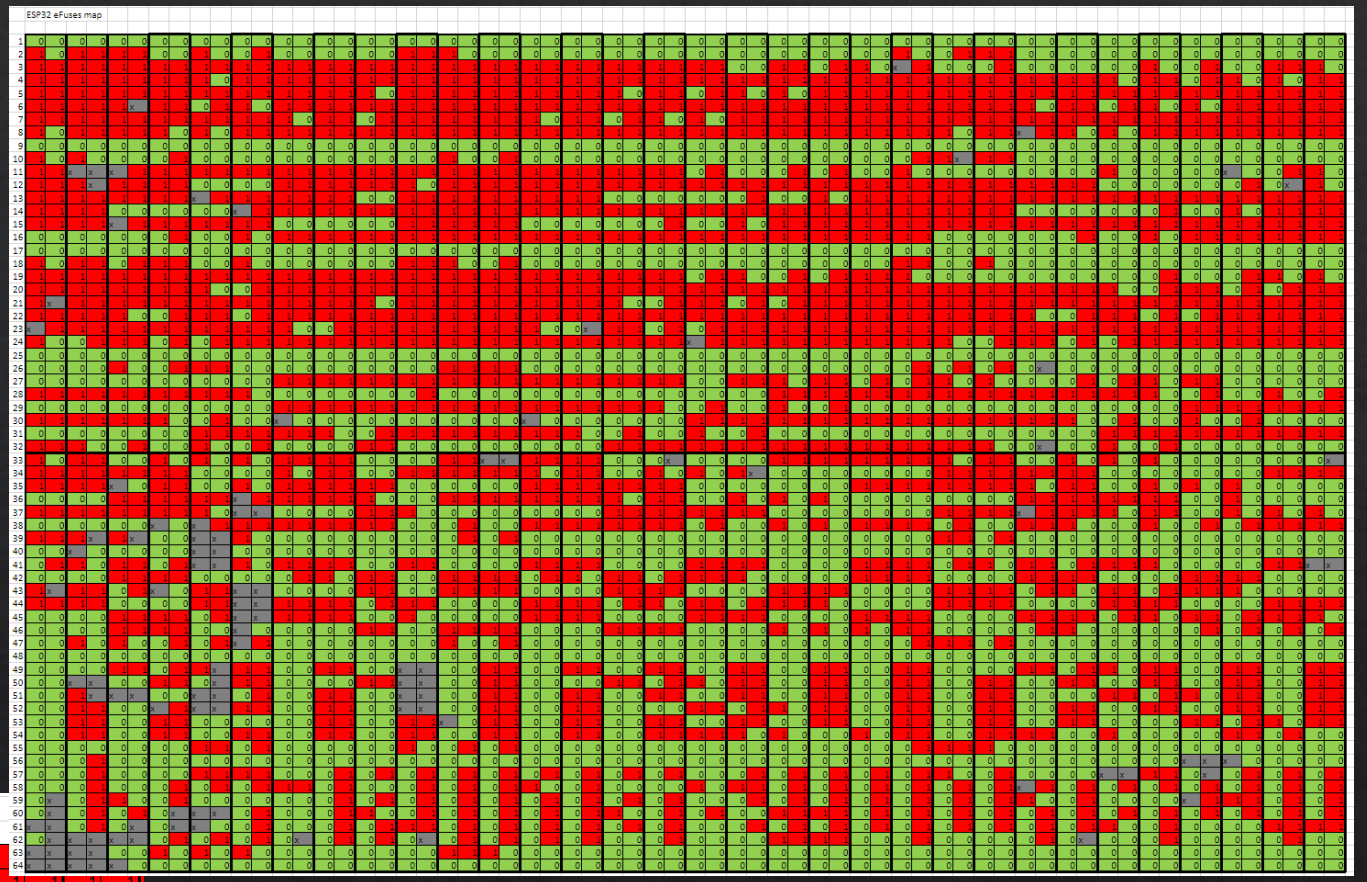
Memory layout evaluation

Manual Experiment #2:

Burn all user writeable eFuse bytes with identical pattern.

Bingo …

```
espefuse.py v4.7-dev
Connecting.....
Detecting chip type... Unsupported detection protocol, switching and trying again...
Detecting chip type... ESP32-S2
BLOCK0          (                ) [0 ] read_regs: 00000000 00000000 00000000 00000000 00000000 00000000
MAC_SPI_8M_0    (BLOCK1          ) [1 ] read_regs: 4e9edff2 0000a076 00000000 00000000 00000000 00000000
BLOCK_SYS_DATA  (BLOCK2          ) [2 ] read_regs: a5cd9406 cb53e8ac 0f379edc 2c4970a6 145c882d 401028a2 0e232970 0604491d
BLOCK_USR_DATA  (BLOCK3          ) [3 ] read_regs: fcfdfeff f8f9fafb f4f5f6f7 f0f1f2f3 ecedeeef e8e9eaeb e4e5e6e7 e0e1e2e3
BLOCK_KEY0      (BLOCK4          ) [4 ] read_regs: fcfdfeff f8f9fafb f4f5f6f7 f0f1f2f3 ecedeeef e8e9eaeb e4e5e6e7 e0e1e2e3
```



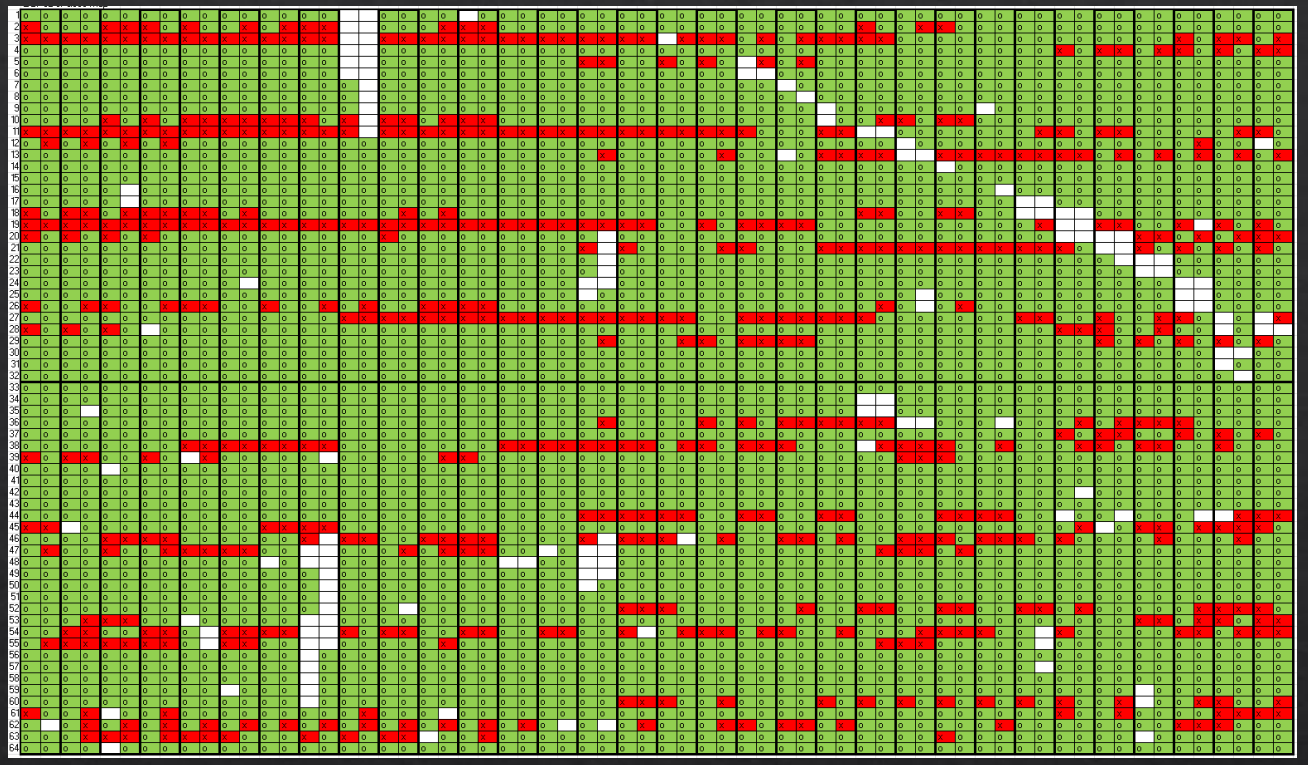
ESP32 eFuses map

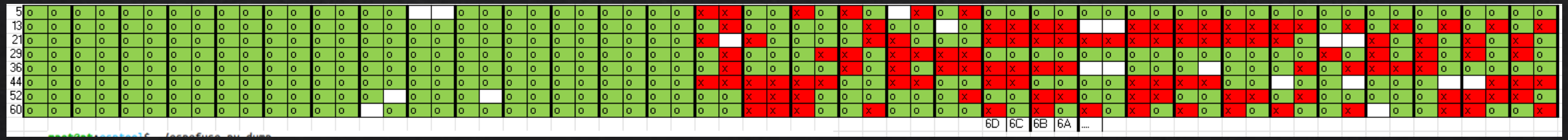
	e0	e1	e2	e3	e4	e5	e6	e7	e8	e9	ea	eb	ec	ed	ee	ef	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	fa	fb	fc	fd
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
38	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1
46	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1
54	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
62	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Memory layout evaluation

Manual Experiment #3:

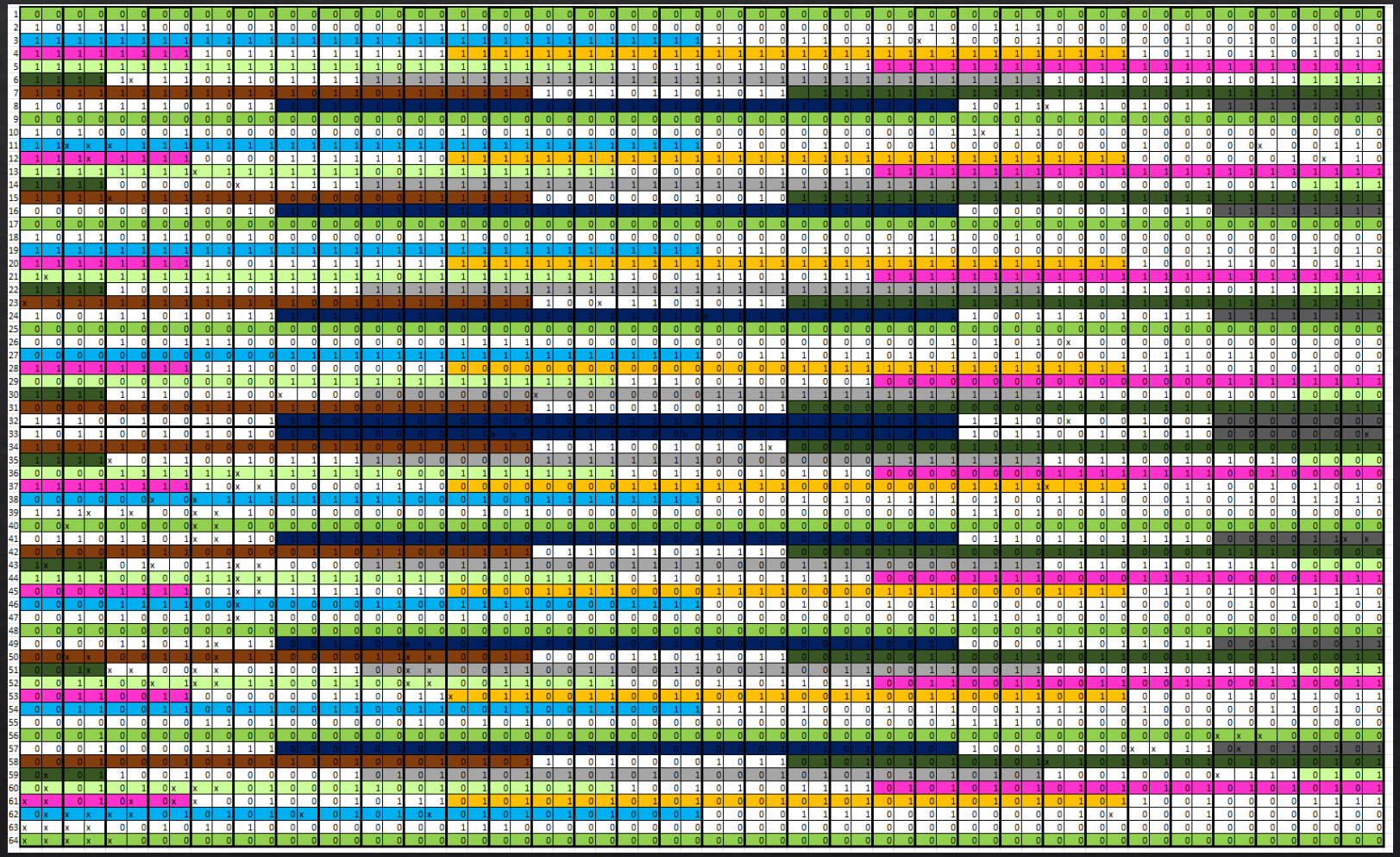
Burn unique values to BLOCK3 and BLOCK5 (0x6a 0x6b 0x6c 0x6d …).



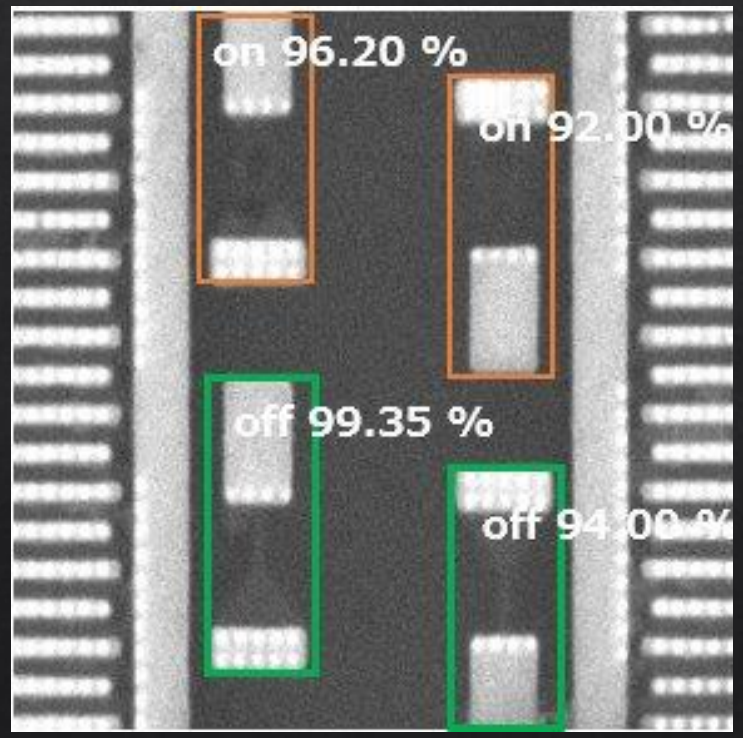


Memory layout evaluation

At this point we know what is what …



Automating the Process



Conclusion

- eFuse-based memories do not provide any confidentiality assurance
- Readability protection methods such as layering increase effort but do not prevent reading
- If you can rent a microscope, it is not that expensive to perform such attack
- Espressif has great team handling submissions for potential security issues
- Integrity assurance of the eFuse-based memories … to be continued ☺

Q & A