

Extracting Antifuse Secrets

from the DEF CON 32 badge (RP2350)

Andrew D. Zonenberg

 @azonenberg@ioc.exchange

IOActive[®]





About Me

- Ph.D RPI '15
- Principal @ IOA Seattle lab
- Broad range of research interests
 - General infosec
 - Open T&M
 - Silicon RE
 - High speed digital



IOActive Presentation Content

Legal Notices

- **Disclaimer Notification**

The views, opinions, findings, conclusions, positions, and/or recommendations expressed herein are those of the authors individually and do not necessarily reflect the views, opinions, or positions of IOActive, Inc.

- **No Warranties or Representations**

The information presented herein is provided "AS IS" and IOActive disclaims all warranties whatsoever, whether express or implied. Further, IOActive does not endorse, guarantee, or approve, and assumes no responsibility for nor makes any representations regarding the content, accuracy, reliability, timeliness, or completeness of the information presented. Users of the information contained herein assume all liability from such use.

- **Publicly Available Material**

All non-IOActive source material referenced in this presentation was obtained from the Internet without restriction on use.

- **Fair Use**

This primary purpose of this presentation is to educate and inform. It may contain copyrighted material, the use of which has not always been specifically authorized by the copyright owner. We are making such material available in our efforts to advance understanding of cyber safety and security. This material is distributed without profit for the purposes of criticism, comment, news reporting, teaching, scholarship, education, and research, and constitutes fair use as provided for in section 107 of the Copyright Act of 1976.

- **Trademarks**

IOActive, the IOActive logo and the hackBOT logo are trademarks and/or registered trademarks of IOActive, Inc. in the United States and other countries. All other trademarks, product names, logos, and brands are the property of their respective owners and are used for identification purposes only.

- **No Endorsement or Commercial Relationship**

The use or mention of a company, product or brand herein does not imply any endorsement by IOActive of that company, product, or brand, nor does it imply any endorsement by such company, product manufacturer, or brand owner of IOActive. Further, the use or mention of a company, product, or brand herein does not imply that any commercial relationship has existed, currently exists, or will exist between IOActive and such company, product manufacturer, or brand owner.

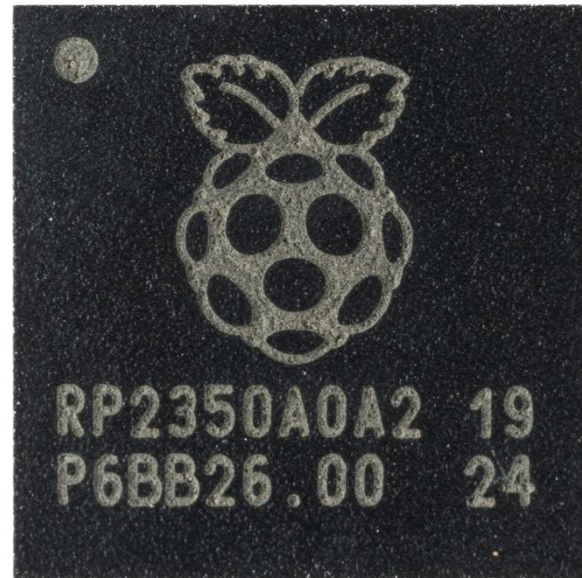
- **Copyright**

©2025 IOActive, Inc. All rights reserved. This work is protected by US and international copyright laws. Reproduction, distribution, or transmission of any part of this work in any form or by any means is strictly prohibited without the prior written permission of the publisher.



So... what's the RP2350?

- Dual core MCU
 - Switchable M33 or RV32!
- 32 kB boot ROM
- 520 kB SRAM
- 8 kB ECC OTP (or 12 kB non-ECC)
- No internal flash
 - Boots from external QSPI flash
 - Supports signed images or nonsecure XIP





But why pwn this micro in particular?

- Lots of chips out there with secure boot
- What makes this one interesting?

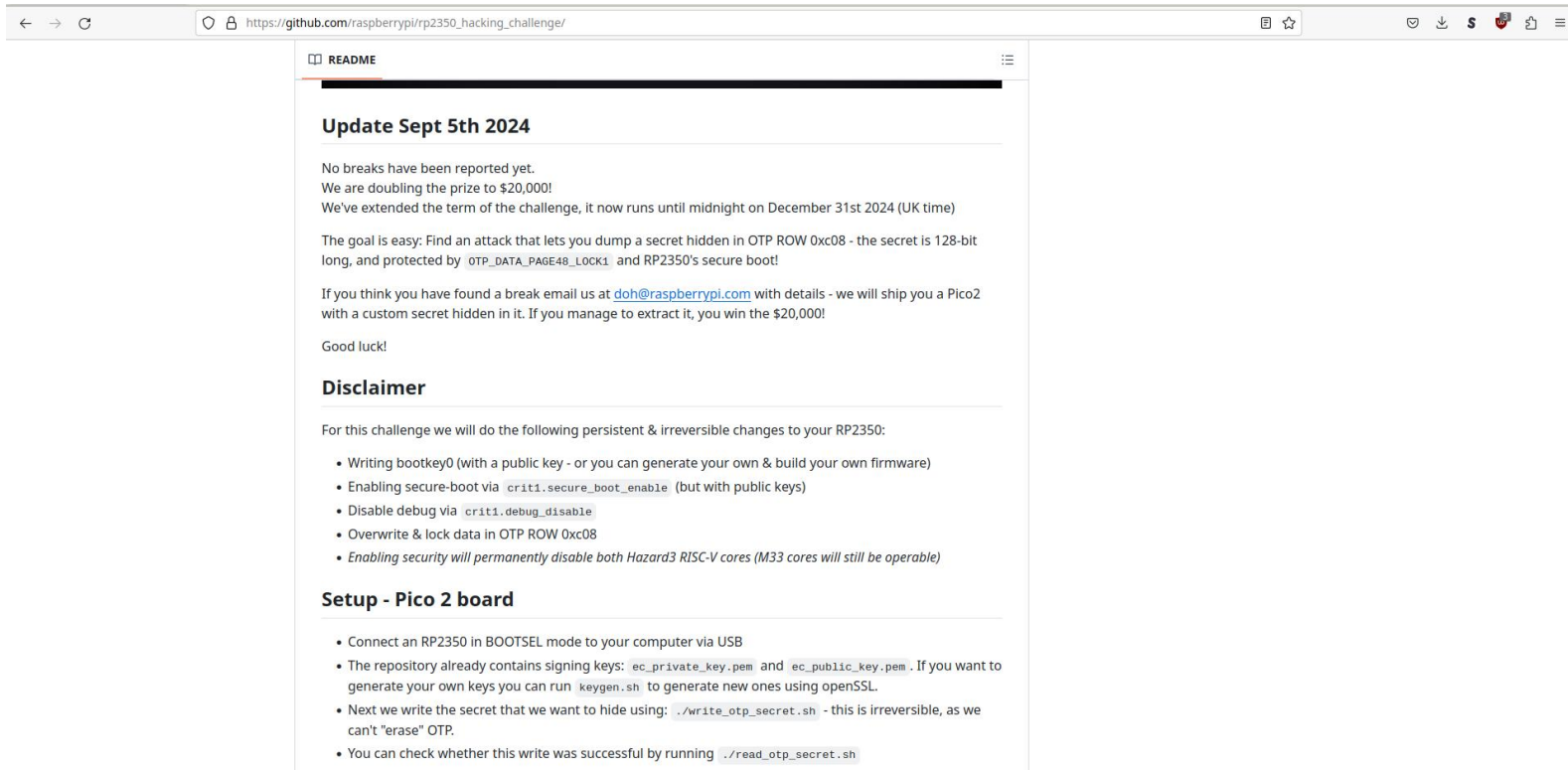


Introduced on the DEF CON 32 badge





Oh, and there was a bounty too...



← → ↺ https://github.com/raspberrypi/rp2350_hacking_challenge/

README

Update Sept 5th 2024

No breaks have been reported yet.
We are doubling the prize to \$20,000!
We've extended the term of the challenge, it now runs until midnight on December 31st 2024 (UK time)

The goal is easy: Find an attack that lets you dump a secret hidden in OTP ROW 0xc08 - the secret is 128-bit long, and protected by `OTP_DATA_PAGE48_LOCK1` and RP2350's secure boot!

If you think you have found a break email us at doh@raspberrypi.com with details - we will ship you a Pico2 with a custom secret hidden in it. If you manage to extract it, you win the \$20,000!

Good luck!

Disclaimer

For this challenge we will do the following persistent & irreversible changes to your RP2350:

- Writing bootkey0 (with a public key - or you can generate your own & build your own firmware)
- Enabling secure-boot via `crit1.secure_boot_enable` (but with public keys)
- Disable debug via `crit1.debug_disable`
- Overwrite & lock data in OTP ROW 0xc08
- *Enabling security will permanently disable both Hazard3 RISC-V cores (M33 cores will still be operable)*

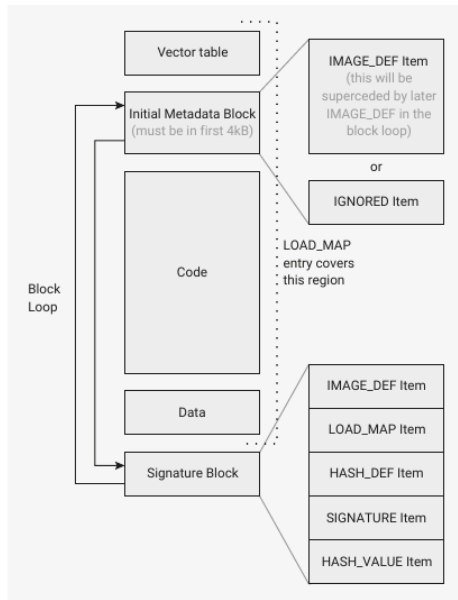
Setup - Pico 2 board

- Connect an RP2350 in BOOTSEL mode to your computer via USB
- The repository already contains signing keys: `ec_private_key.pem` and `ec_public_key.pem`. If you want to generate your own keys you can run `keygen.sh` to generate new ones using openssl.
- Next we write the secret that we want to hide using: `./write_otp_secret.sh` - this is irreversible, as we can't "erase" OTP.
- You can check whether this write was successful by running `./read_otp_secret.sh`



Secure boot flow

- TL;dr: copy image from QSPI to SRAM
- Check signature
 - Pubkey + sig in image header
 - Hash of key burned into fuses
- If sig valid, run it
- Image is cleartext
 - Encrypted boot not *directly* supported





Encrypted boot flow

- Secure-boot a trusted, open source “stub”
- Stub copies main firmware blob to SRAM
- Can then decrypt using a key stored in fuses
 - Fuses support read protection so can’t dump via JTAG etc
 - Attacker who can read fuses could thus decrypt FW



Challenge goal

- Given a locked RP2350, extract secret stored in fuses
- Goal is *not* a secure boot bypass per se
 - If executing unsigned code is the best attack, that's fine
 - But we want *the secret out of a single chip*
 - Don't care about *mass pwnage* of other hardware



So, what do we know about the fuses?

RP2350 provides 8 kB of one-time programmable storage (OTP), which holds:

- Preprogrammed per-device information, such as unique device identifier and oscillator trim values
- Security configuration such as debug disable and secure boot enable
- Public key fingerprints for secure boot
- Symmetric keys for decryption of flash contents into SRAM
- Configuration for the USB bootloader, such as customising VID/PID and descriptors
- Bootable software images, for low-cost flashless applications or custom bootloaders
- Any other user-defined data, such as per-device personalisation values

Logical array size

For the full listing of predefined OTP contents, see [Section 13.9](#).

OTP is physically an array of 4096 rows of 24 bits each. You can directly access these 24-bit values, but there is also hardware support for storing 16 bits of data in each row, with 6 bits of Hamming ECC protection and 2 bits of bit polarity reversal protection, yielding an ECC data capacity of 8192 bytes.

On a blank device, the OTP contents is all zeroes, except for some basic device information pre-programmed during manufacturing test. Each bit can be *irreversibly* programmed from zero to one. To program the OTP contents:

0 -> 1 = antifuse



Let's do some OSINT

Now we know who makes it

13.2. Background: OTP IP Details

The RP2350 OTP subsystem uses the Synopsys NVM OTP IP, which comes in 3 parts:

- Integrated Power Supply (IPS), including:
 - Charge Pump (for programming)
 - Regulator (for reading)
- OTP Macro (SHF, Fuse)
 - 4096×24 (8 kB with ECC, 16-bit ECC write granularity)
- Access port (AP), providing:
 - Basic read access
 - Programming access
 - ECC and bit redundancy
 - BOOT function, which polls for stable OTP power supply at start-of-day



Let's do some OSINT

SYNOPSYS®

Solutions

Products

Support

News & Views

Company

Home

Synopsys IP

myDesignWare

dwc_nvm_ts40np5sxxxh0nopxxxi

dwc_nvm_ts40np5sxxxh0nopxxxi

IP Directory Component Detail

Description: NVM OTP TSMC 40nm LP 2.5V
Name: dwc_nvm_ts40np5sxxxh0nopxxxi
Version: 2.02a
ECCN: 3E991/NLR
STARs: Open and/or Closed STARs
myDesignWare: Subscribe for Notifications
Product Type: DesignWare Embedded Memory IP
Documentation: Contact Us for More Information
Download: v-nvm_ts40np5sxxxh0nopxxxi
Product Code: C976-0

Don't know if the 2350 is
40LP, ULP, or another flavor
But it's this IP or a close cousin



Let's do some OSINT

synopsys®

Solutions Products Support News & Views Company

Home / Synopsys IP / Foundation IP / NVM / Antifuse-Based Split-Channel 1T-Fuse Bit Cell for OTP NVM IP

Antifuse-Based Split-Channel 1T-Fuse Bit Cell for OTP NVM

At the heart of all Synopsys One-Time Programmable (OTP) Non-Volatile Memory (NVM) IP is the memory array, using a patented, area-efficient, antifuse bit cell – 1T-Fuse – that employs gate-oxide breakdown as a robust, non-reversible programming mechanism. This is a proven, reliable and secure technology that has been widely adopted and used in a broad range of applications and markets.

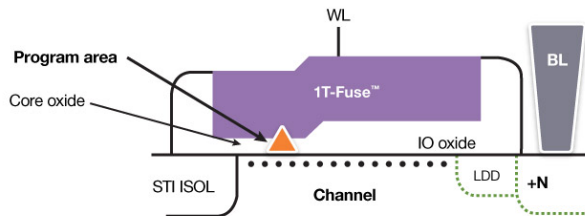


Figure 1: 1T-Fuse Bit Cell in Synopsys OTP NVM IP

The single-transistor OTP bit-cell (1T-Fuse) uses a unique split-channel architecture, where a single transistor gate covers both I/O (thick) and core (thin) oxides. The cell is programmed by a controlled, irreversible breakdown from the gate through the core (gate) oxide to the channel. The bit cell is programmed either with an embedded charge pump or through an external pin (at the tester, for example). The bit-cell is implemented using standard CMOS processes and requires no additional mask steps.

Gate oxide breakdown
= antifuse

Cross section of
memory bit cell

https://www.synopsys.com/dw/ipdir.php?ds=nvm_1t-bit-cell



Let's do some OSINT

Arrays are available from a few 100 bits to around 1M bit and may be cascaded for even larger macros. Synopsys memory arrays are very flexible and can replace multi-time programmable memory in many applications. Conversion to mask ROM is also simple.

Available in standard CMOS processes, Synopsys OTP memory arrays don't require additional mask layers or process steps and provide a viable alternative to mask ROM, eFuses and Flash memory. The reduced size of the single-transistor bit cell results in better yield, higher security, improved reliability and lower overall cost.

Security Features

The inherent security features of the 1T-Fuse bit-cell include:

- Programming is by permanent structural change in few atomic layers (located far from diffusion)
- No physical attack can reveal programmed state in FinFET or HKMG technologies
- No leakage in non-programmed state
- State cannot be changed through exposure to high temperature, voltage or radiation
- No charge or voltage involved in state retention (unlike floating gate NVM, e.g. flash)
- State of memory (even for a few bits) is **virtually impossible** to detect using physical attack or reverse engineering techniques

That sounds like a challenge...

https://www.synopsys.com/dw/ipdir.php?ds=nvm_1t-bit-cell



Attack path 1

- RCE the challenge firmware
- Read out fuses



~~Attack path 1~~

- Almost zero attack surface ☹️
 - They even turned off the *printf*'s in case they were glitchable
- There goes that idea



Attack path 2

- Find a way to bypass secure boot
 - Bootrom bug
 - Glitch
- Run our own unsigned firmware
 - Trivial to read fuses at this point
 - ... if we can get there



~~Attack path 2~~

- No obvious bootrom vulns
- Lots of glitch detectors, randomized timing, etc
- And our glitching experts in Madrid were busy 😞
- But the Seattle silicon lab had time...
 - What if we could just read the secret right off the fuses?



Attack path 3

- Decap the chip
- Find the fuses
- Figure out a way to dump the bits
 - But it's “virtually impossible”...

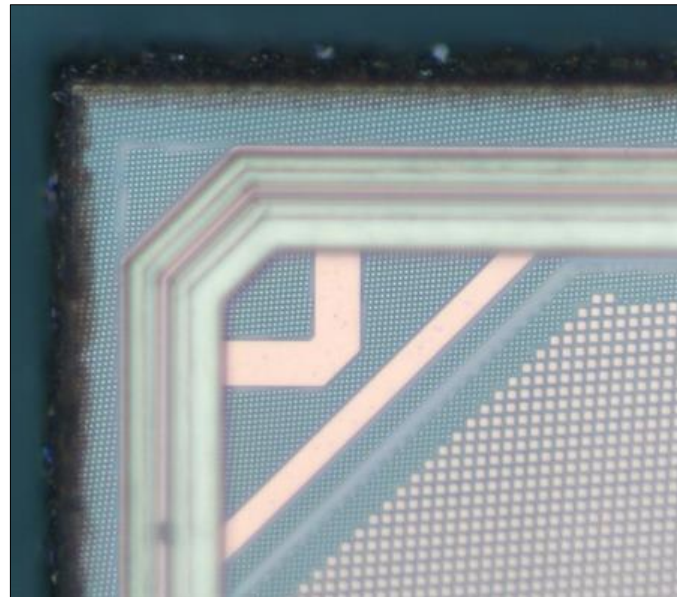
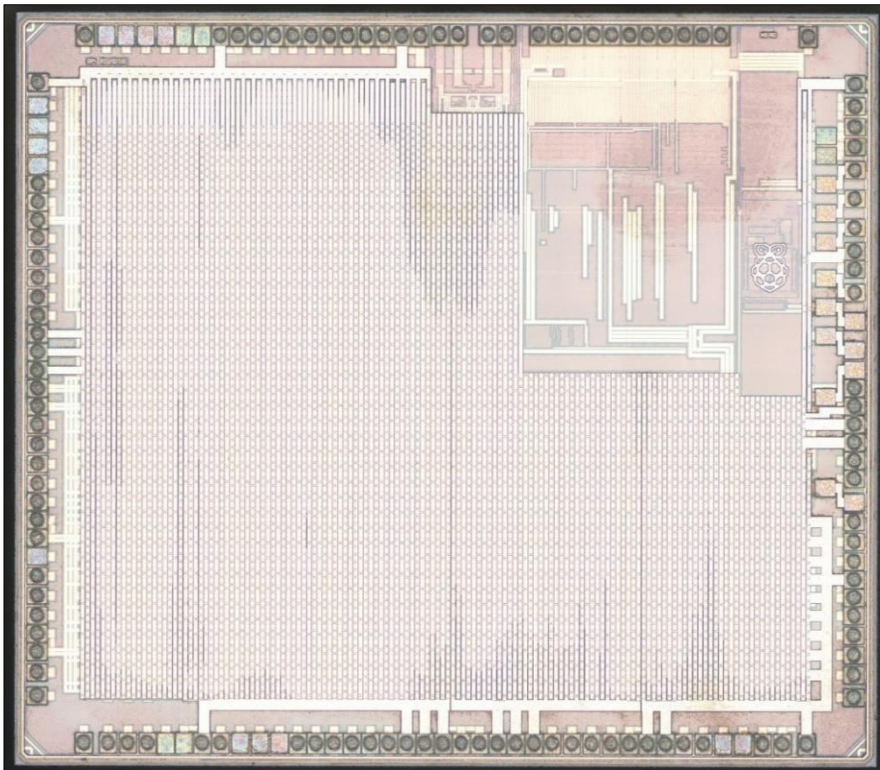


Decap it

- Boil ‘em, heat ‘em, stick ‘em in acid
- You know the drill, we won’t bore you with details



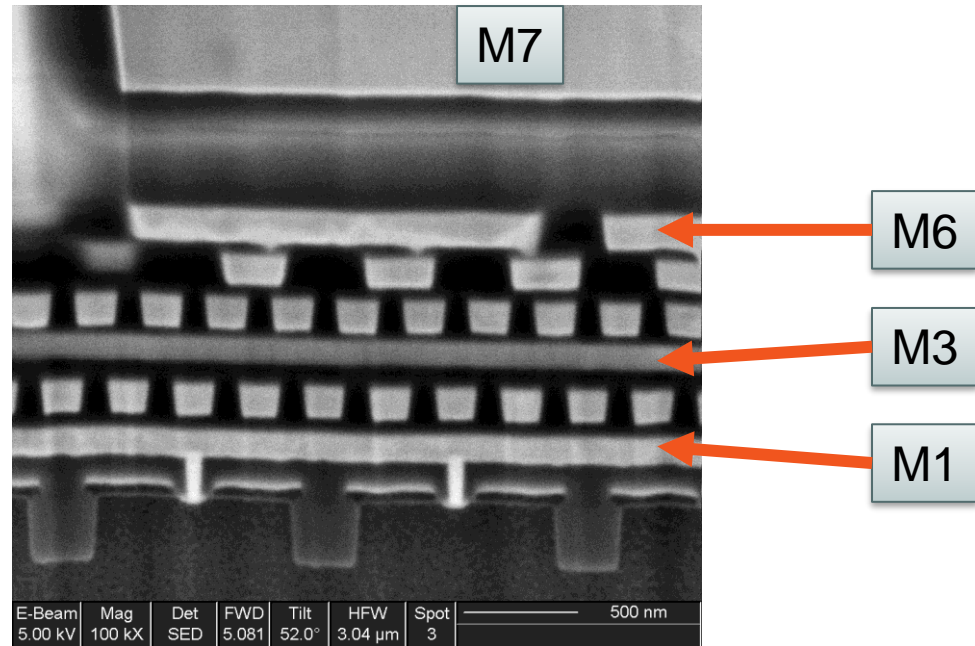
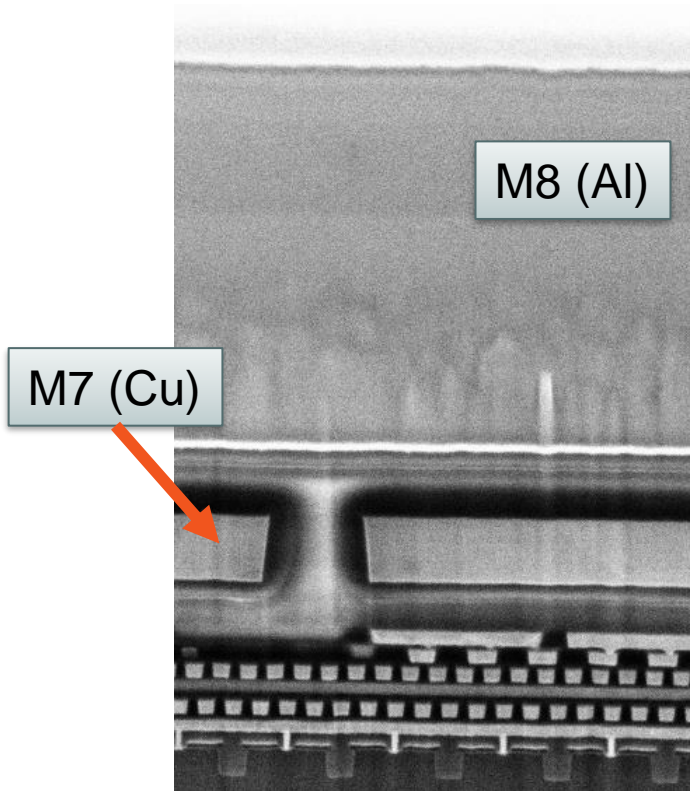
Top metal (M8) overview



2.218 x 2.477 mm (5.493 mm²)

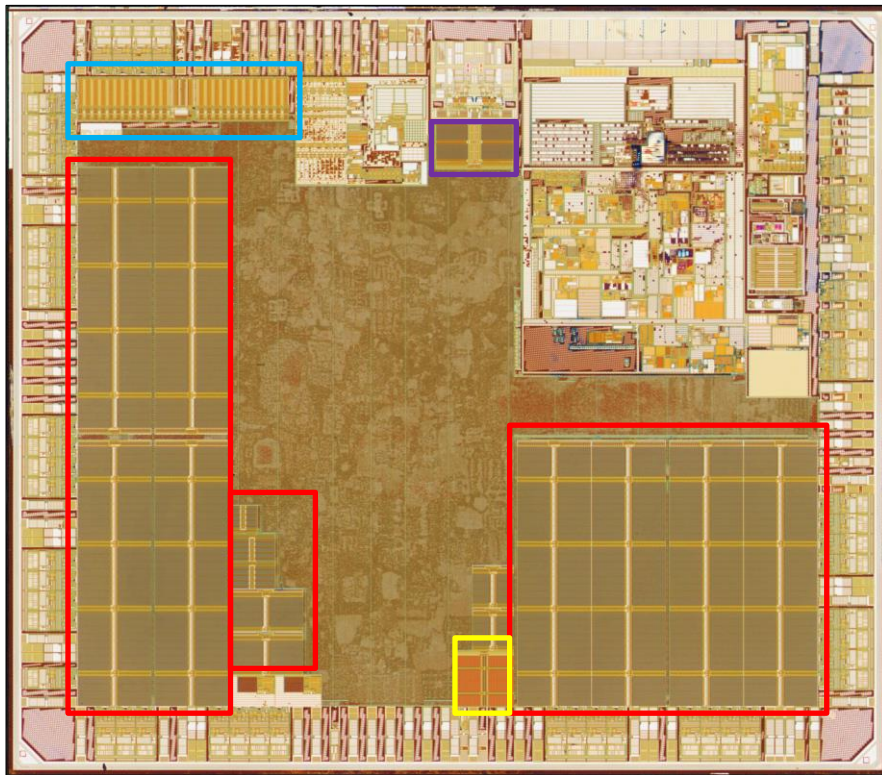


40nm TSMC (7 Cu + 1 Al)





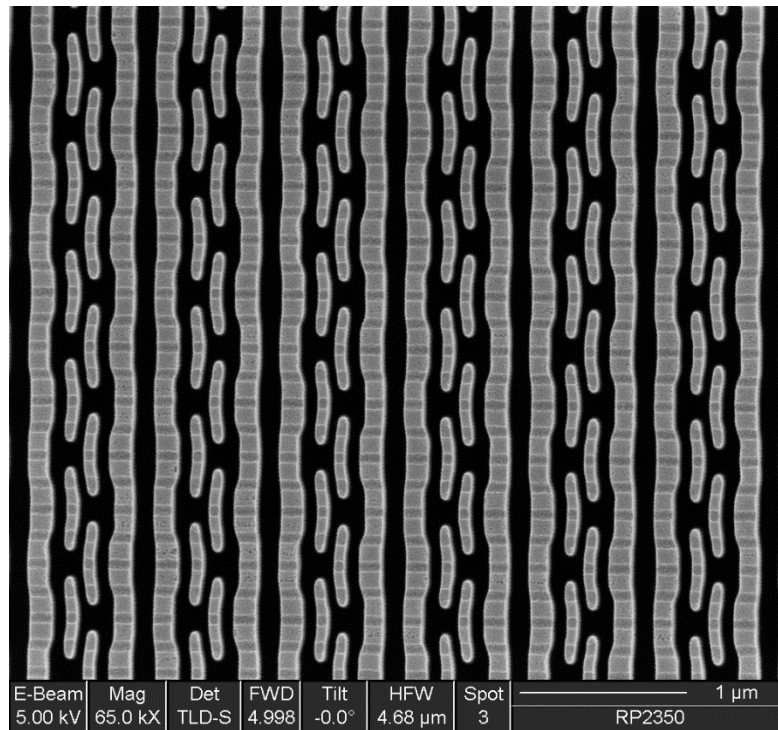
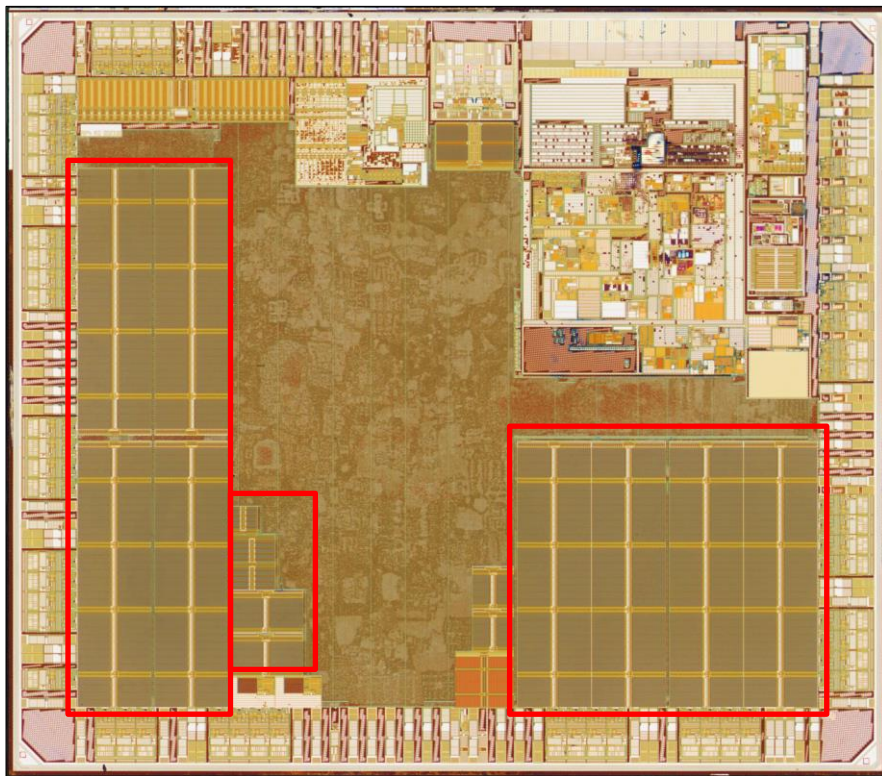
Substrate overview



Lots of memories of several types
Which one is the fuses?



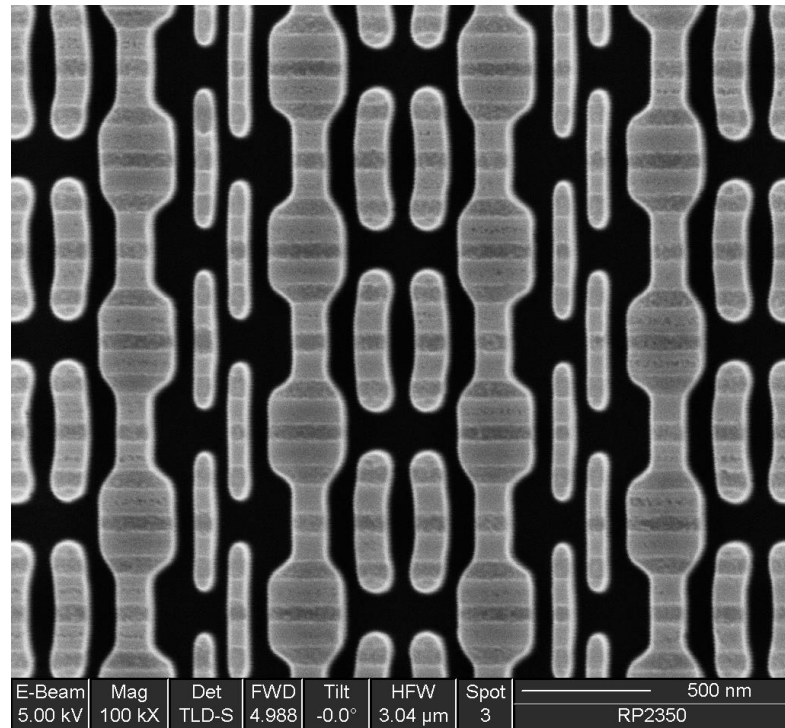
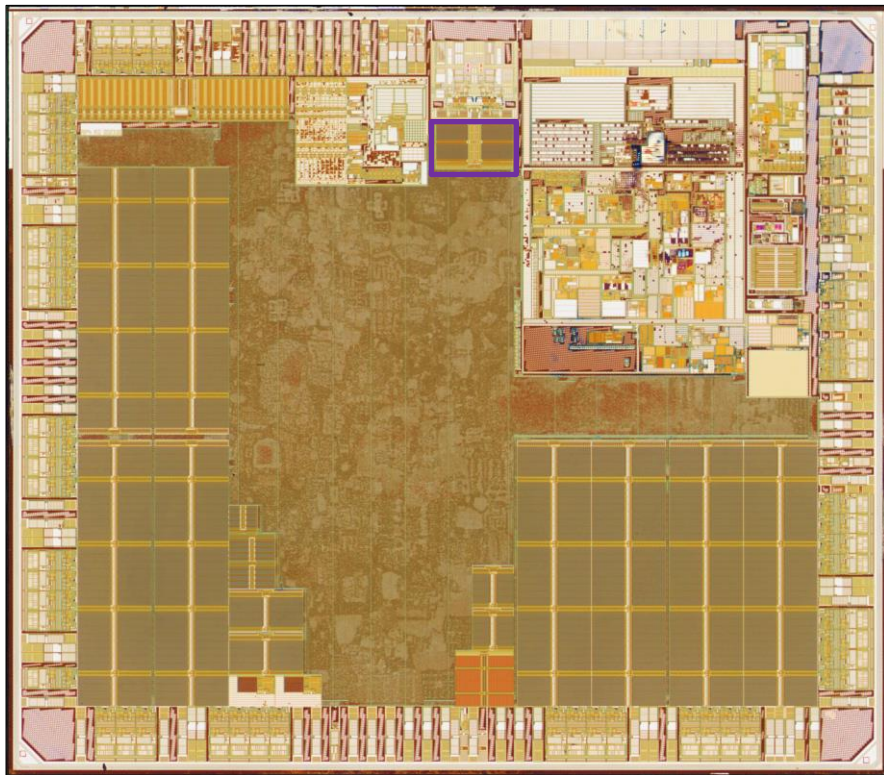
6T SRAM (single port)



E-Beam	Mag	Det	FWD	Tilt	HFW	Spot	1 μm
5.00 kV	65.0 kX	TLD-S	4.998	-0.0°	4.68 μm	3	RP2350

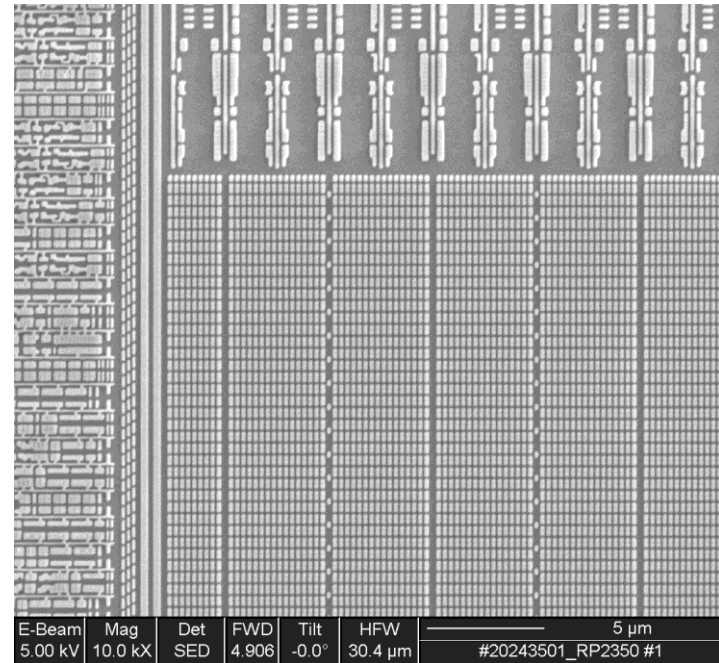
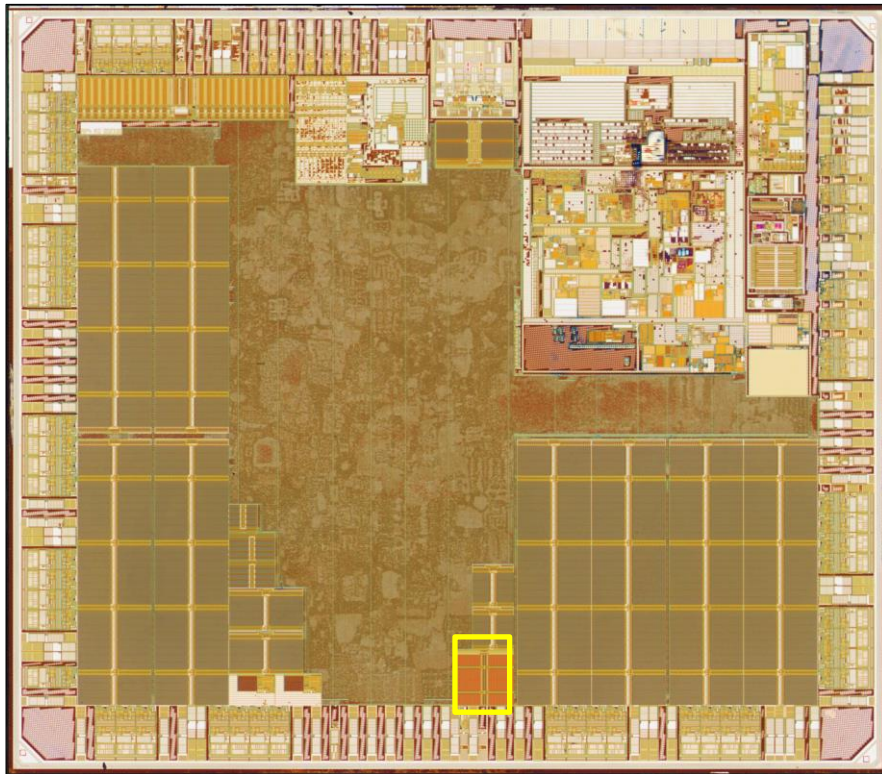


8T SRAM (dual port)





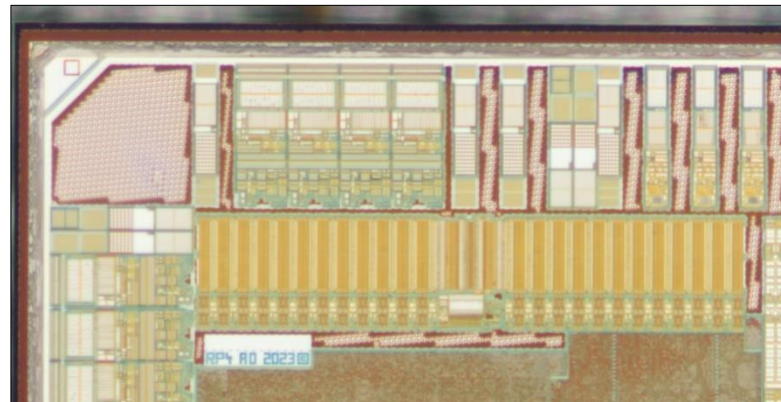
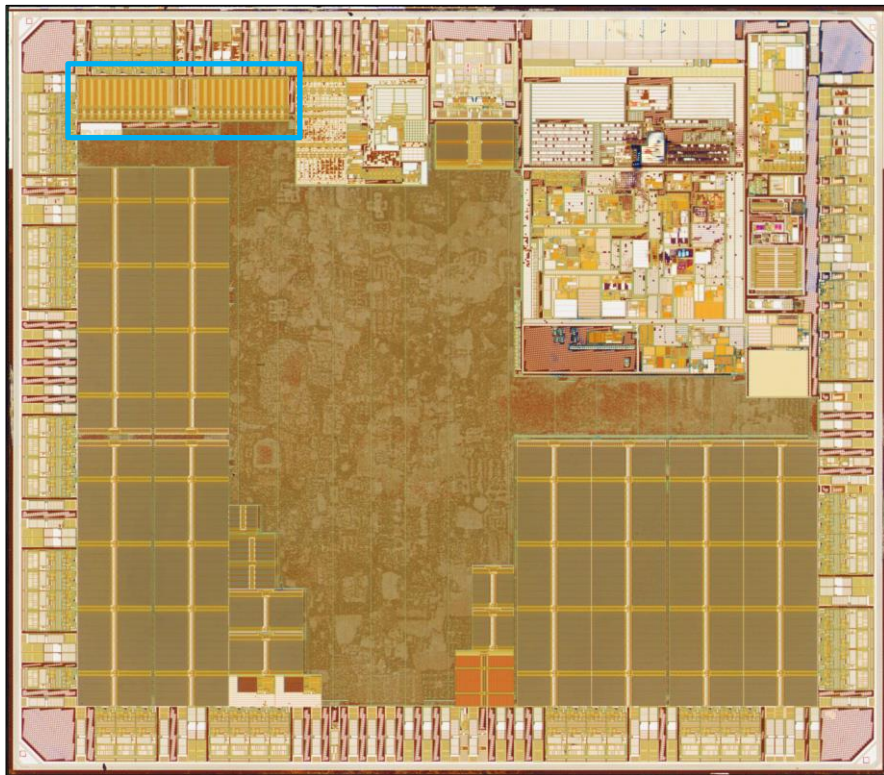
Mask ROM (no bits visible, likely on M1)



Boot ROM is open source
No point spending time dumping it

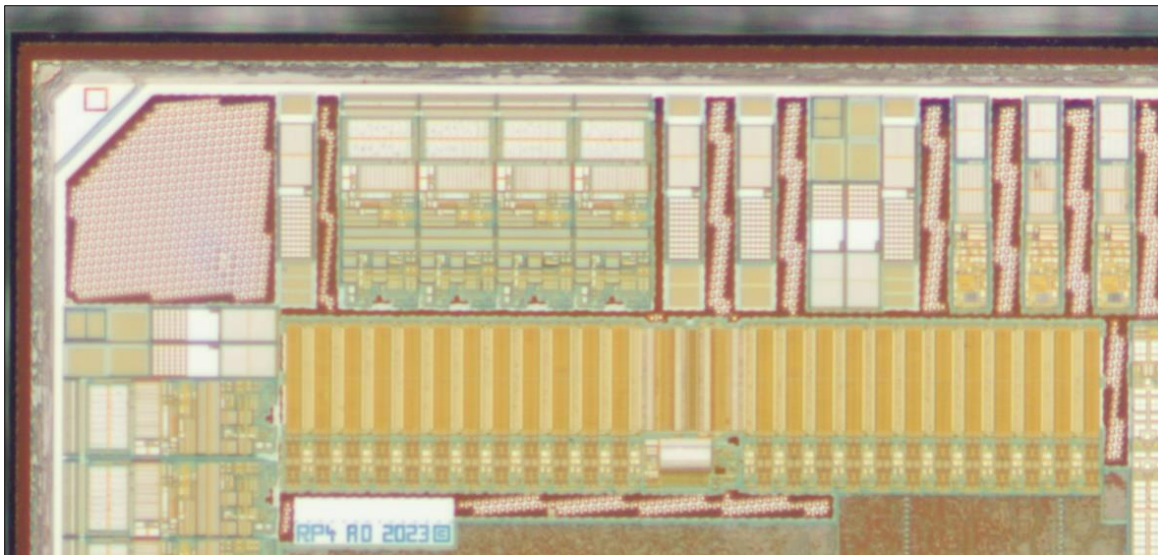


0w0 what's this?

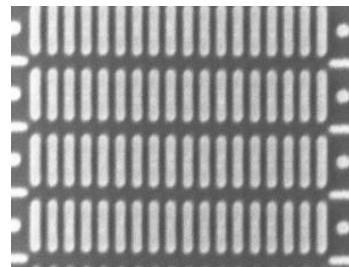




0w0 what's this?

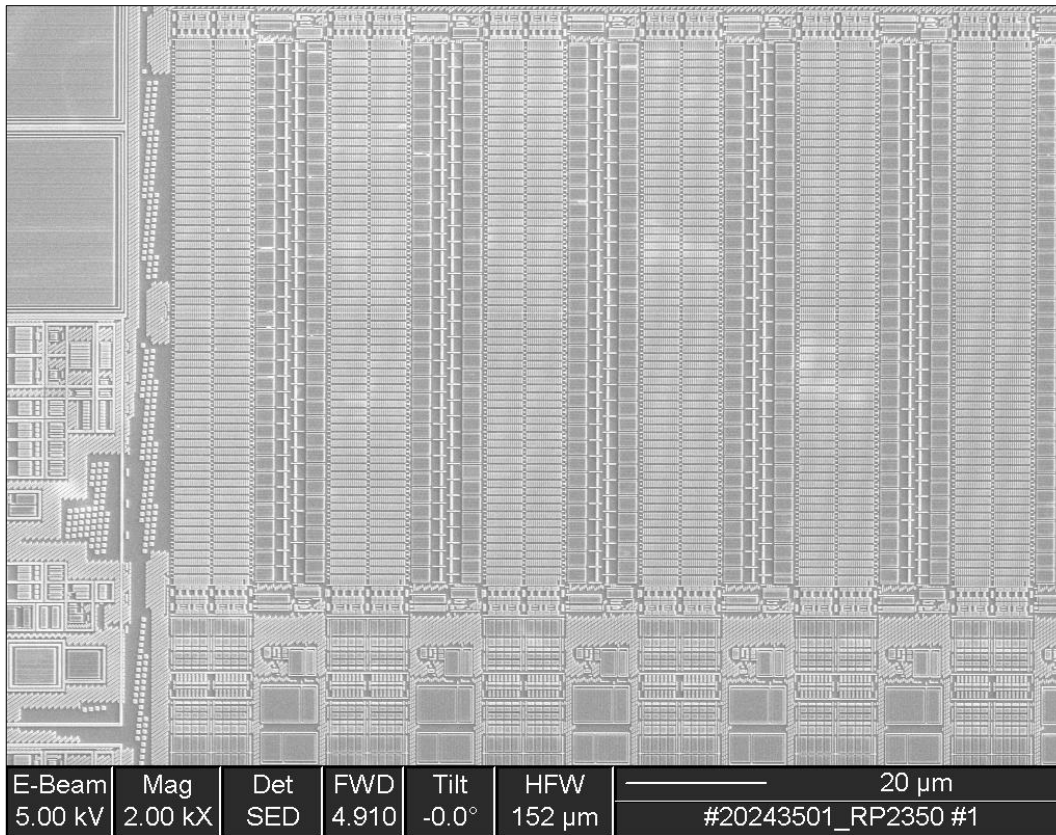


- Not SRAM
- Not ROM
- Not flash
- Looks to be 24 cols
- Fuses are 24 bits
- Hmmm...



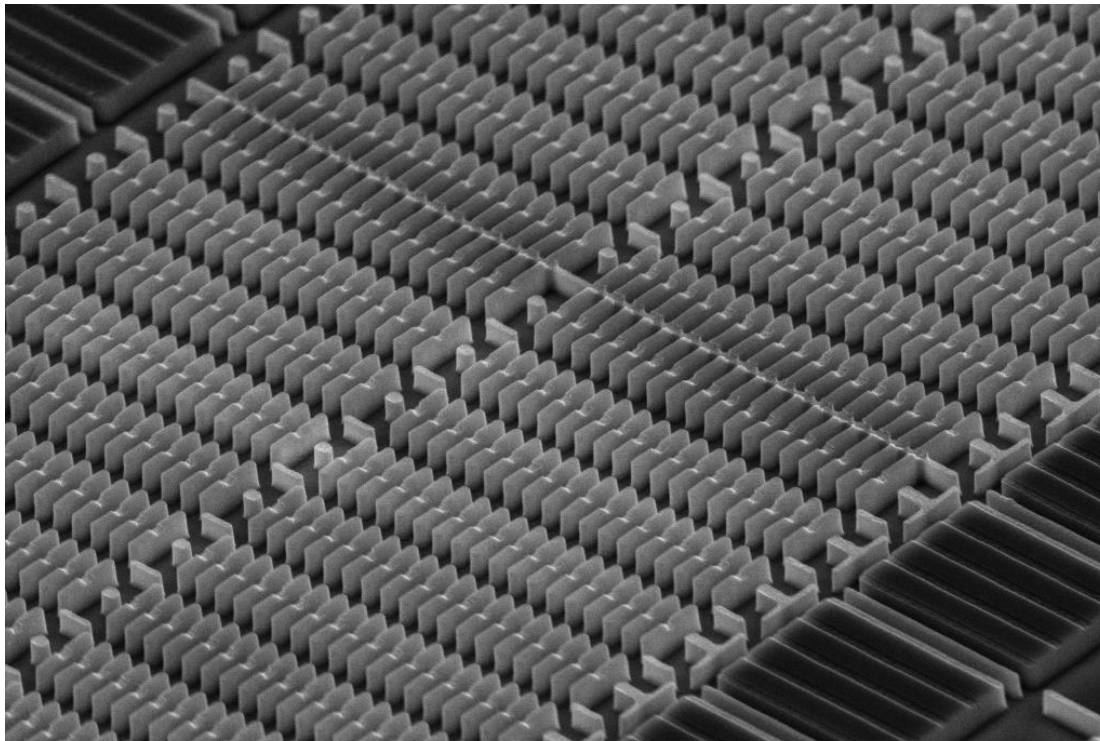


Let's take a closer look





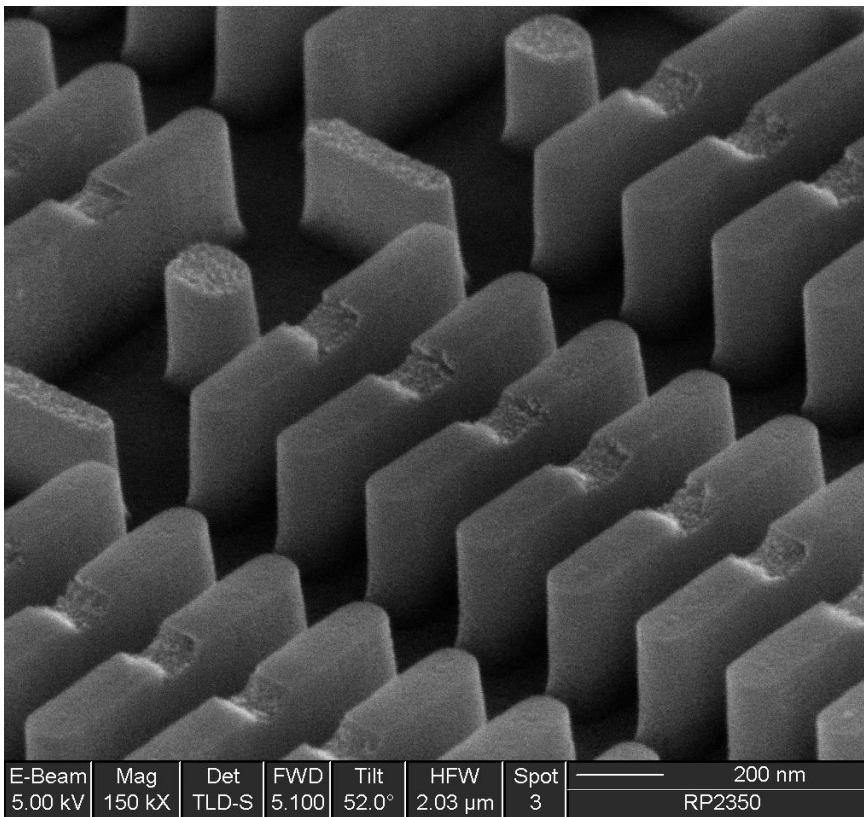
Closer...



E-Beam	Mag	Det	FWD	Tilt	HFW	Spot	2 μ m
5.00 kV	25.0 kX	TLD-S	5.097	52.0°	12.2 μ m	3	RP2350

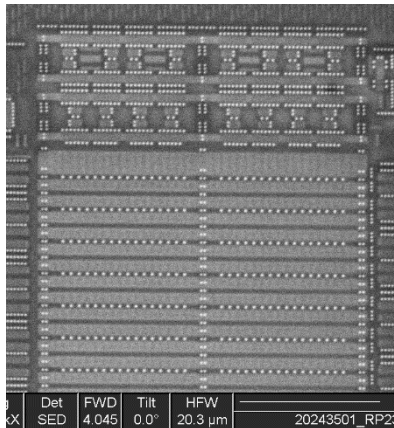


Closer!

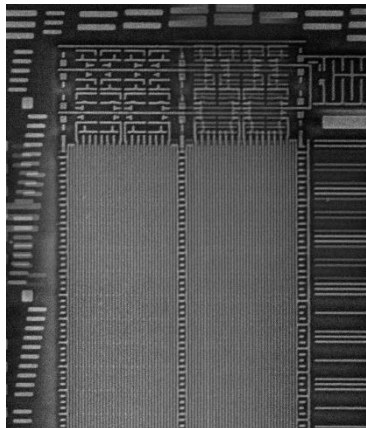




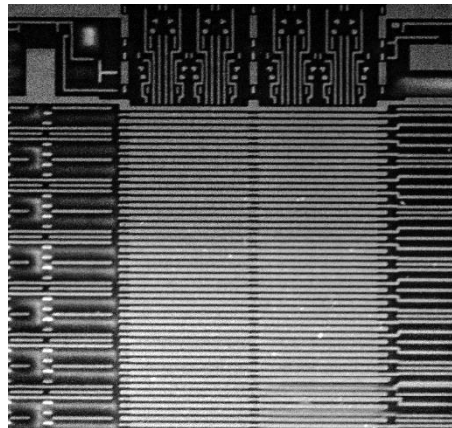
Multi layer overview



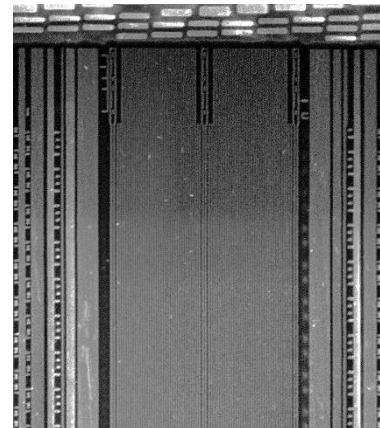
Poly
Contact



M1



M2

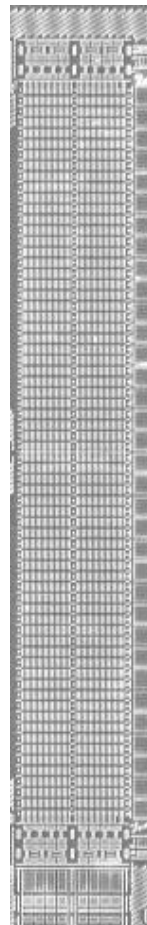
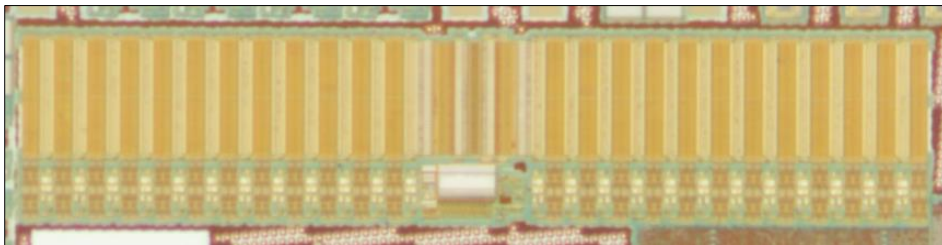


M3



High level address map

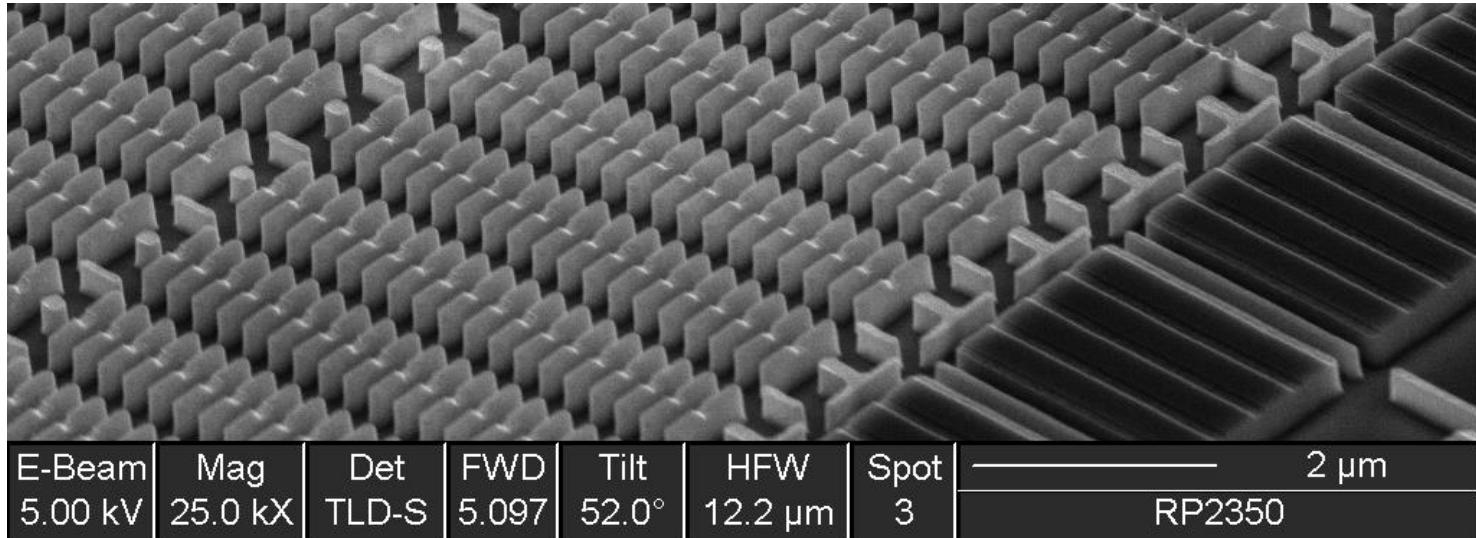
- 24 columns: matches the 24-bit fuse width
 - Each column is probably all rows of one bit
 - Don't know which column is which bit plane yet
- 2 x 2 sub-arrays in each bit plane
 - 4096 words in the whole memory
 - Each sub-array must be 1024 words





But wait, something doesn't add up

- Each sub array is 18 columns x 34 rows
- That's 612 tiles, not 1024...





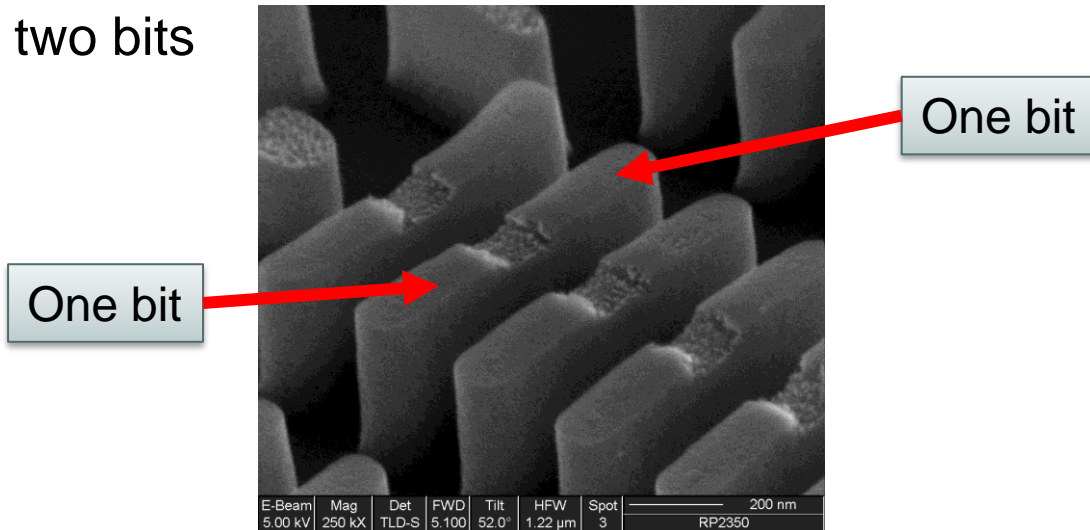
Lithography dummy features

- Memories tend to push fab limits for density
- Periodic arrays are easy to make
 - Every tile has identical diffraction patterns, etch loading, etc
- But what about the edges?
 - Features on the perimeter may have fab issues!
 - So just add an extra “dummy” row / col that’s not used
 - Improves yield of the core area



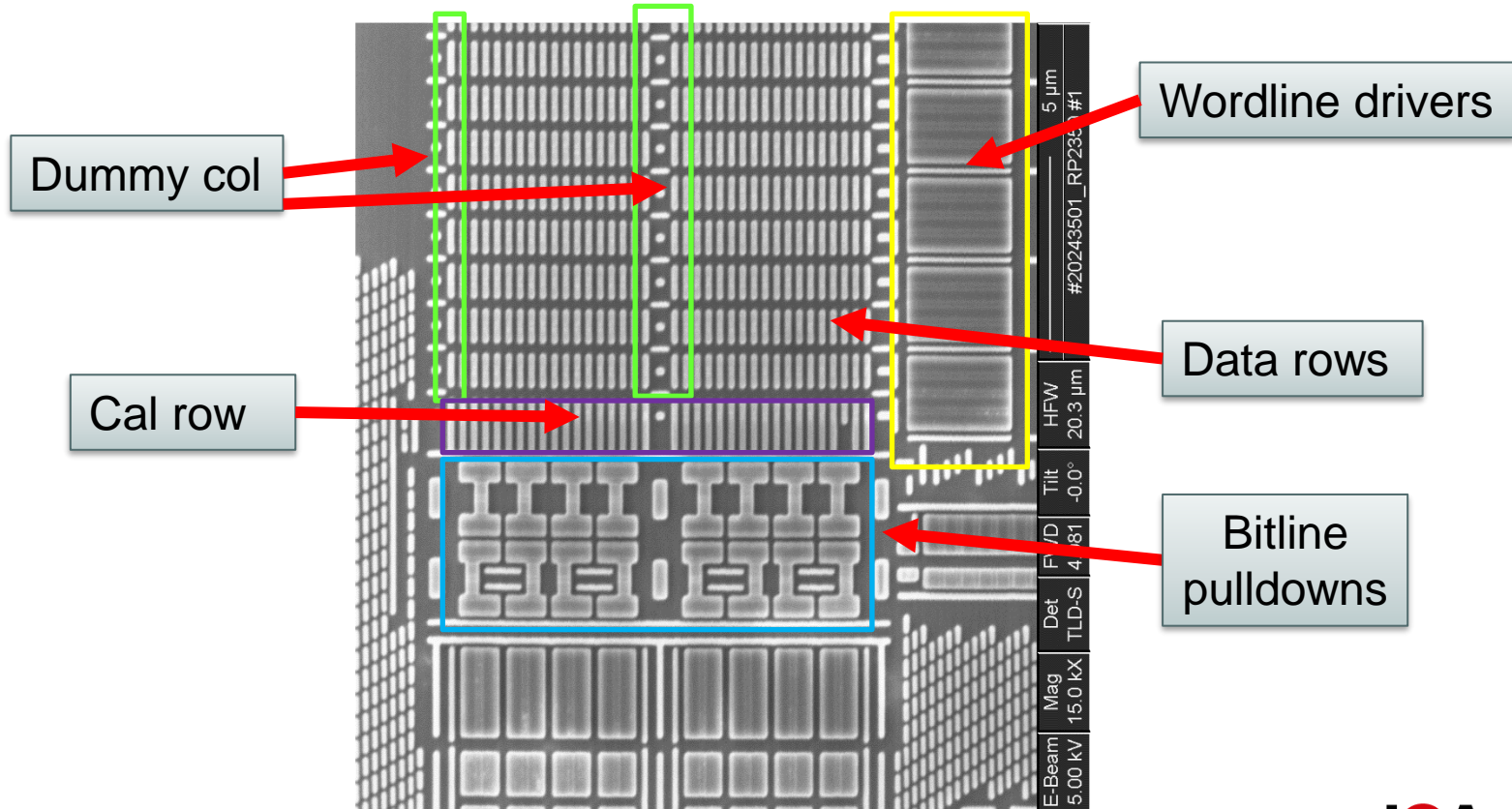
Subtract the dummy features and...

- 16 columns x 32 rows per sub array
- That's 512 tiles for 1024 bits. Much more plausible
 - Each tile must store two bits



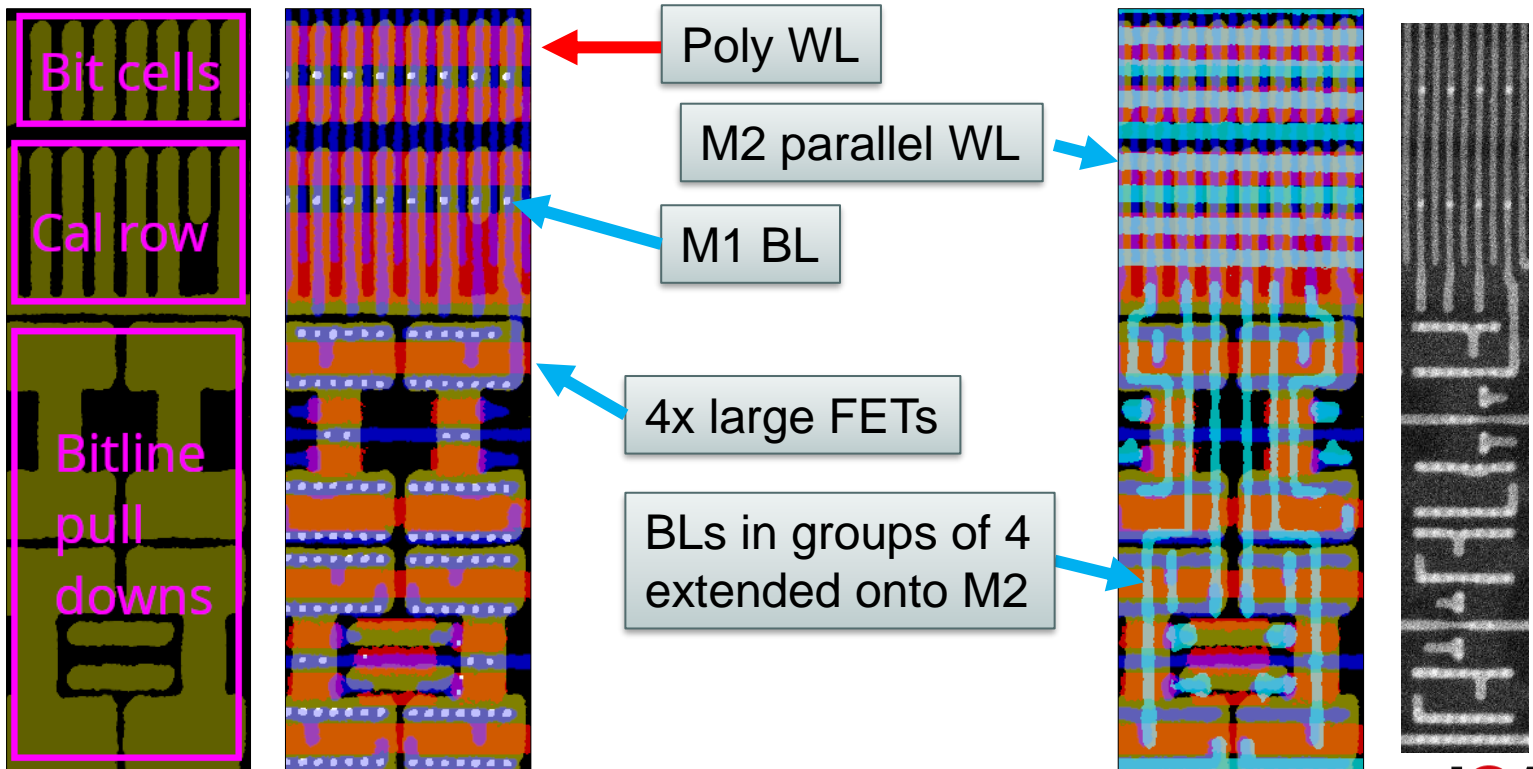


Bottom of a single bit plane (substrate)



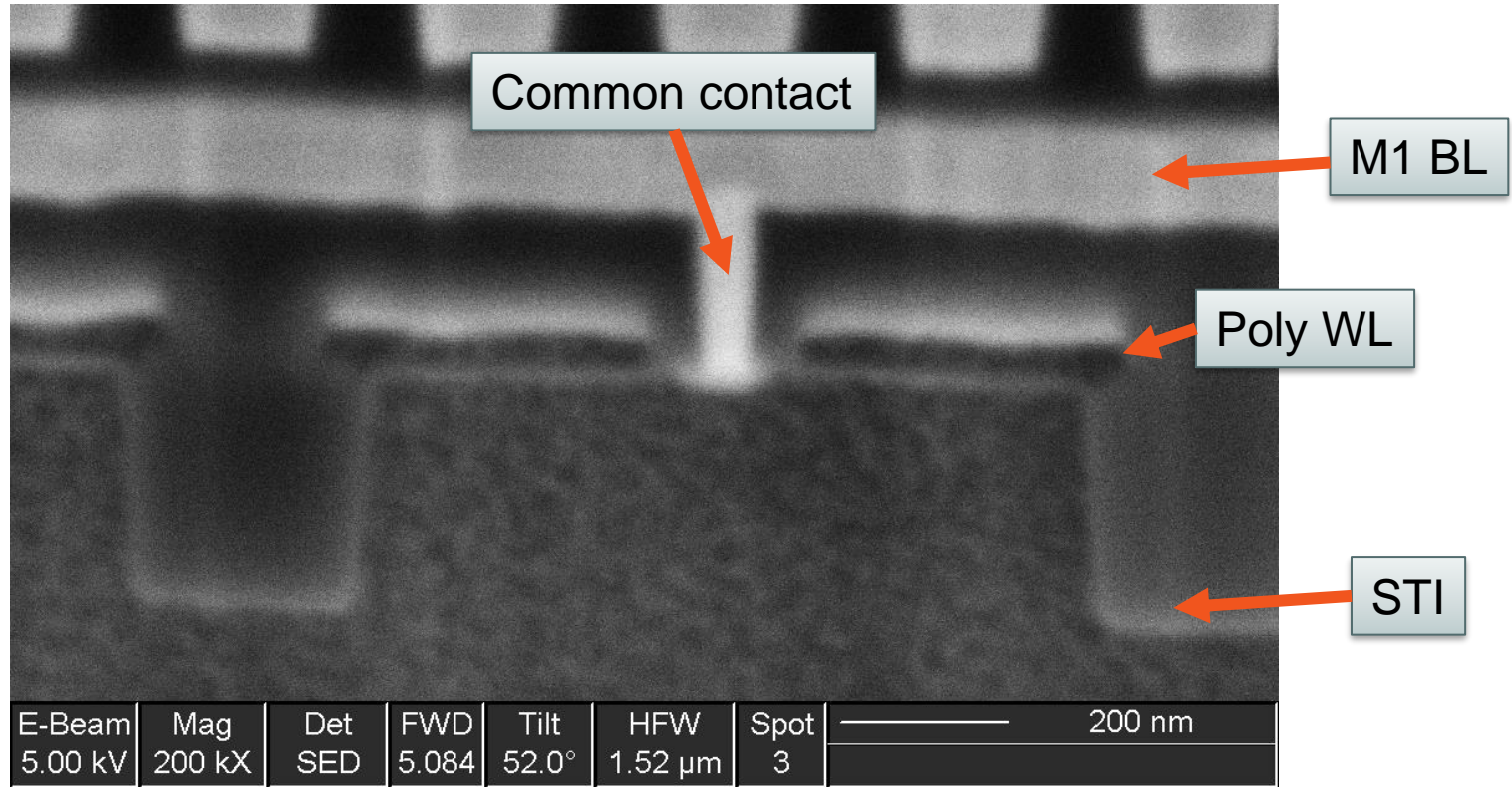


Contacts, poly, M1, M2



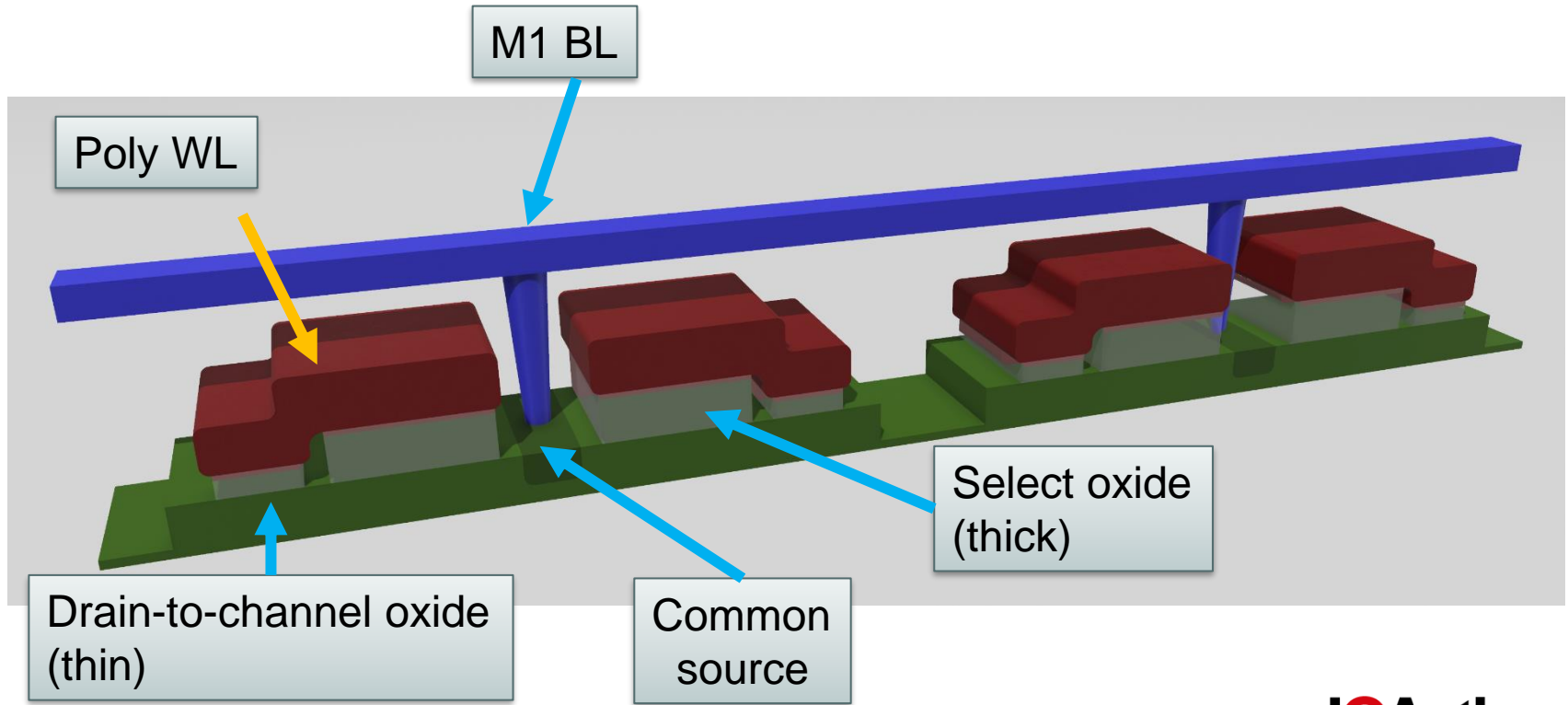


Bitcell pair in cross section



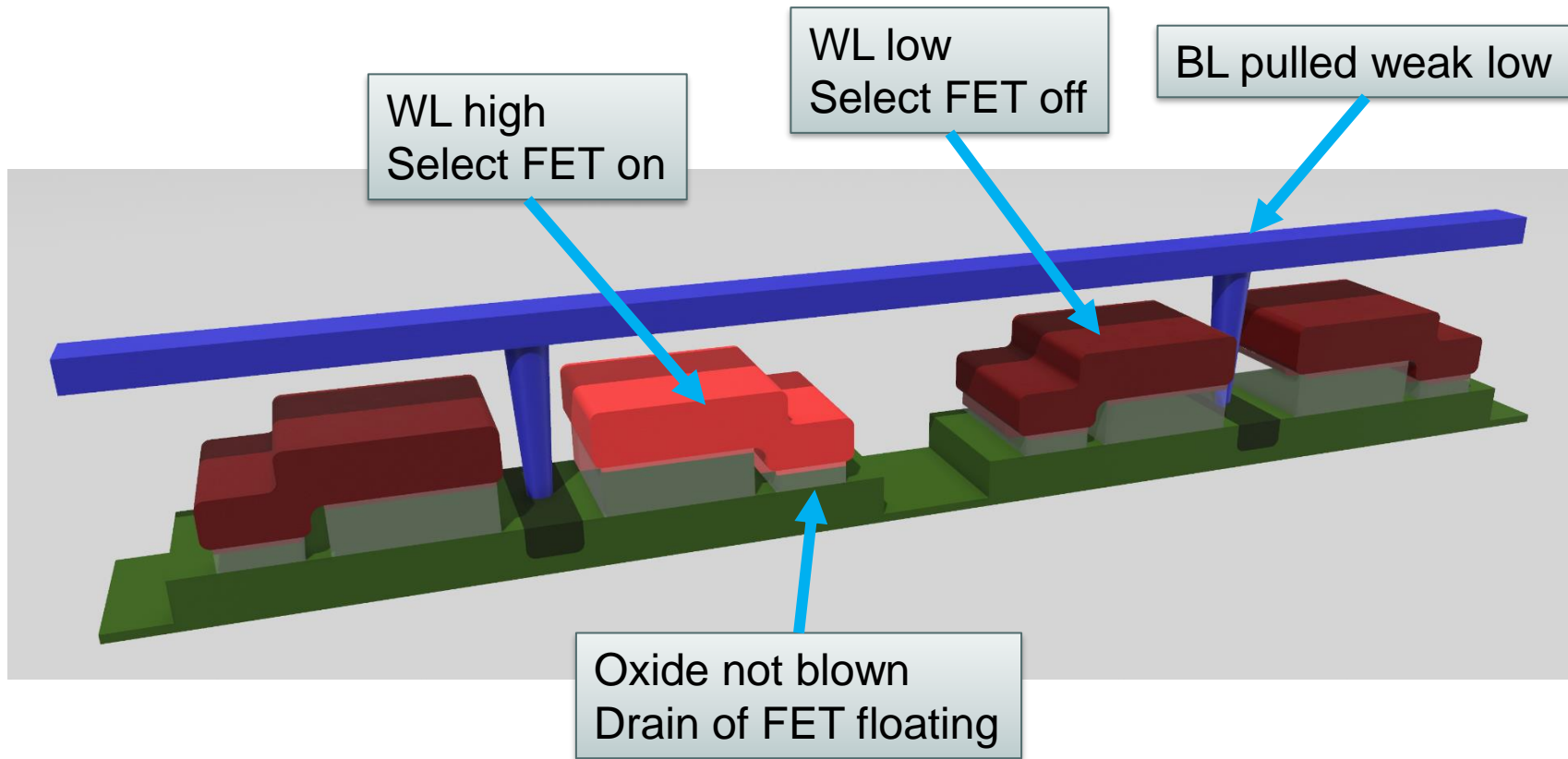


Two rows of bit cells (simplified)





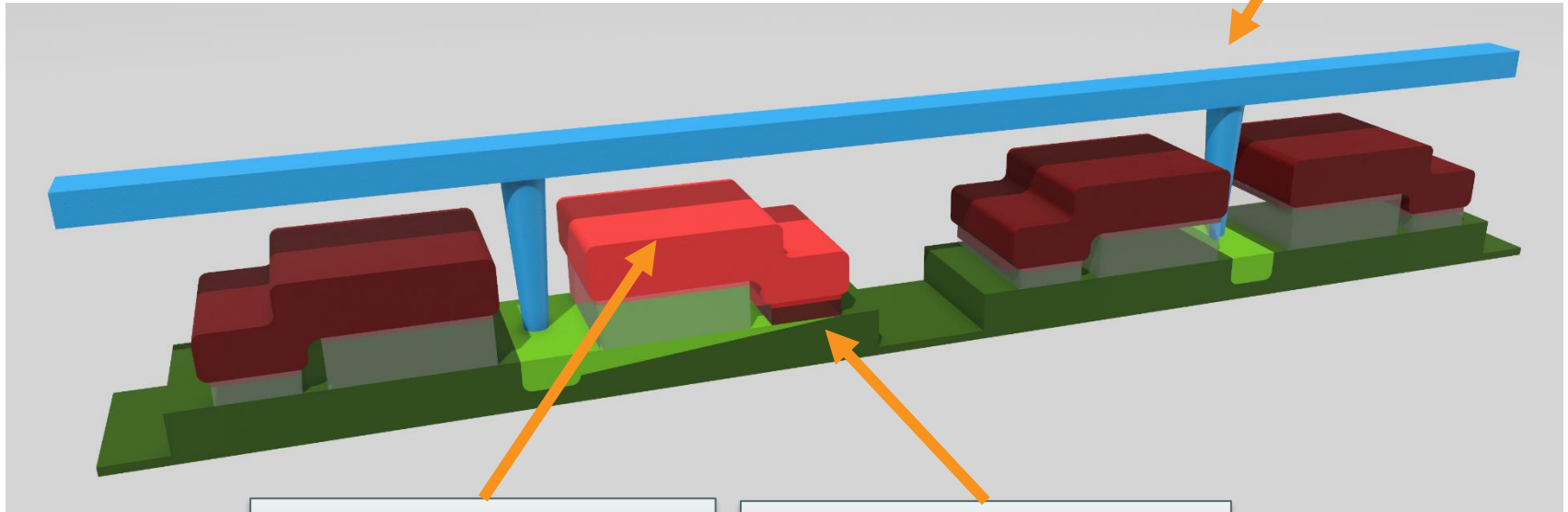
Reading blank (0) bit





Reading programmed (1) bit

BL driven high
Overrides pulldown

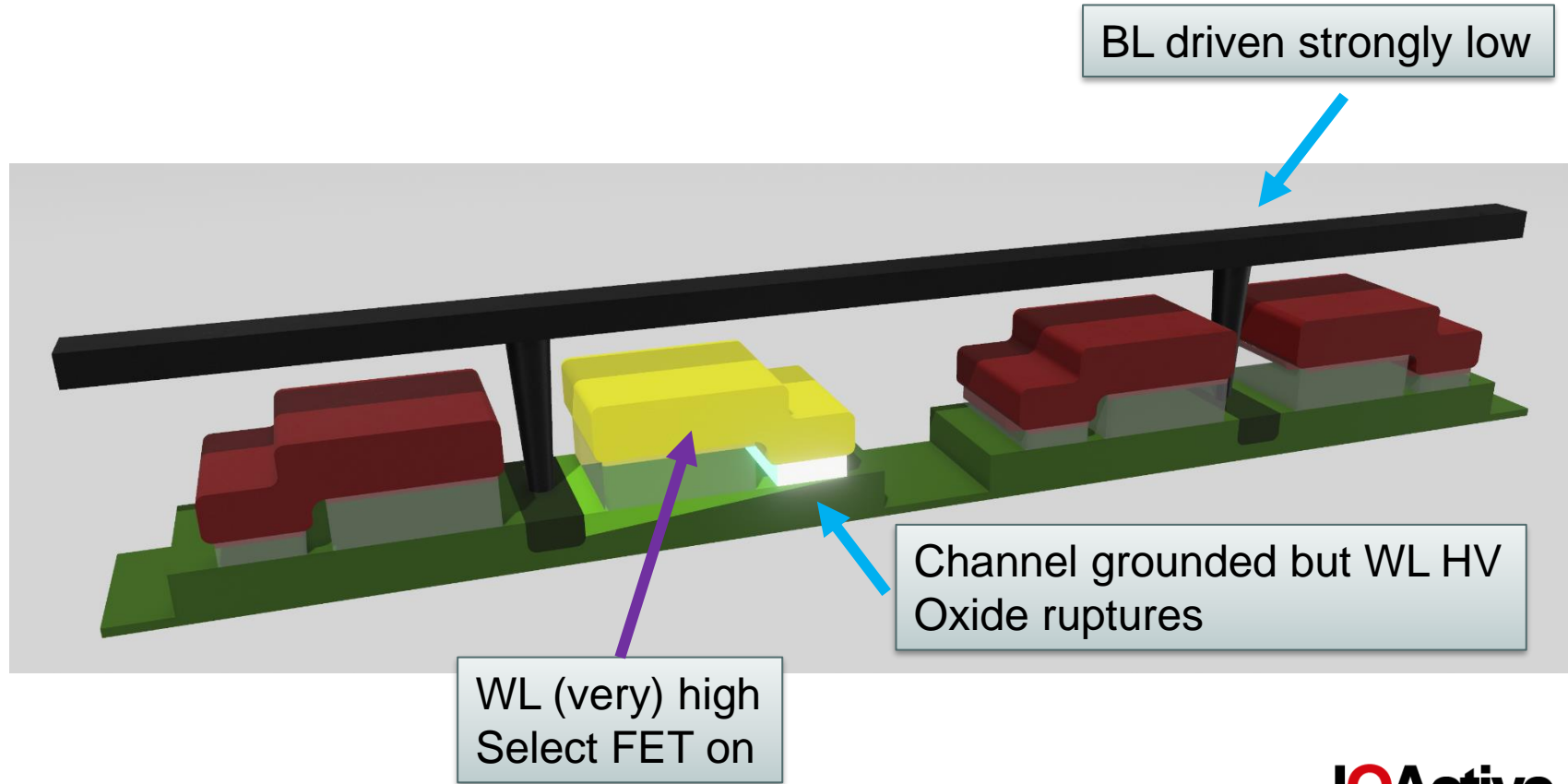


WL over channel high
Select FET on

Oxide blown
N+ poly *is* drain of FET



Programming 0 bit to 1





So now how do we dump it?

- We need to figure out some way to get data out
- Two fundamental approaches
 - Use the normal readout logic
 - Go after the bit cells directly



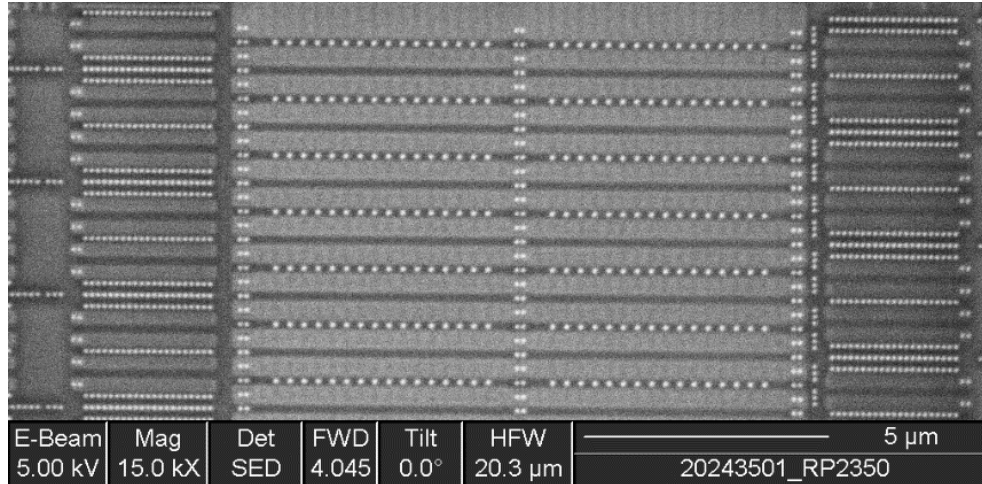
Direct readout via bus interfacing

- Data bus is 32-bit APB
 - We *could* trace out the bus signals
 - Lots of additional netlist RE even to find them
 - Probing >32 nets on a 40nm target would be a nightmare
 - Physically possible buuuuut...



What if we could just see the bits?

- But...
 - All bits look the same in SEM
- Maybe a different technique?





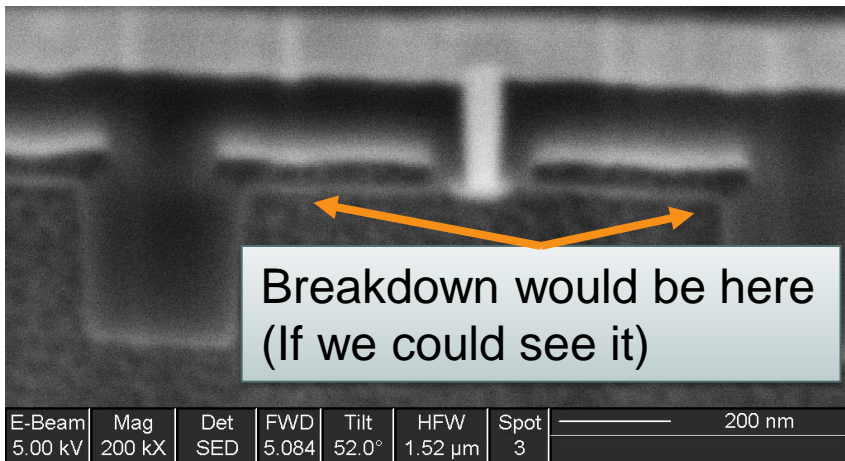
Plan view scanning probe imaging

- Deprocess down to poly, etch the poly
 - Needs etch that's very selective for poly vs oxide
- STM or AFM on field oxide
 - Single atom resolution
 - Even tiny oxide defects should be visible
- Would probably work, but take *forever*



FIB section bits

- Slow (hour or so per bit)
- Extremely thin gate dielectric (few atoms thick)
 - Not visible in SEM





TEM section bits

- FIB section, but then lift out and image in TEM
- Almost guaranteed to work, but...
 - 1-2 bits per day * \$\$\$/hr of lab time
 - Rough guesstimate: low 4 digits USD *per bit*
 - TLA or megacorp could afford this if only option
 - But not most attackers (i.e. us)



What if we could **see voltages**?

- Voltage contrast imaging!
 - Passive VC: beam used for imaging + biasing
 - Active VC: beam used for imaging, external bias



Particle beam imaging recap

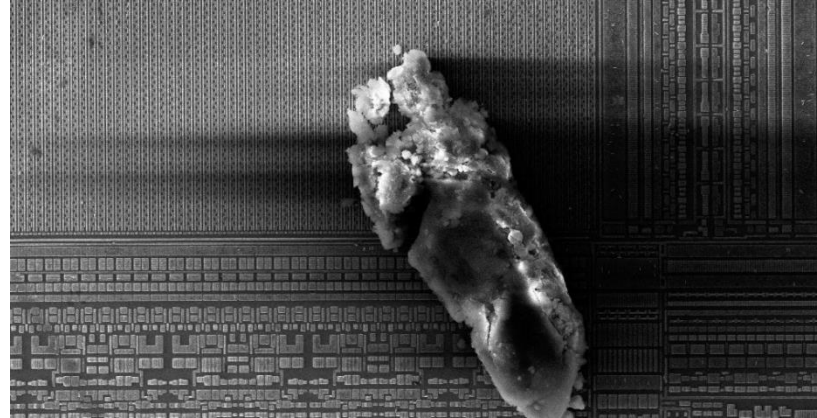
- Raster a beam of charged particles over the specimen
 - e^- (SEM), Ga^+ (FIB) most common
- Beam interacts with sample atoms
- Detect these interactions somehow
 - X-rays
 - Secondary electrons
 - Backscattered electrons
 - Cathodoluminescence



Particle beam imaging w/ charged sample

- SEs are low energy / low mass – easily deflected
- Charged samples are normally bad for SE imaging
 - Causes image shifts, dark spots, difficulty imaging
 - This is why we coat
- But we can put this to use!

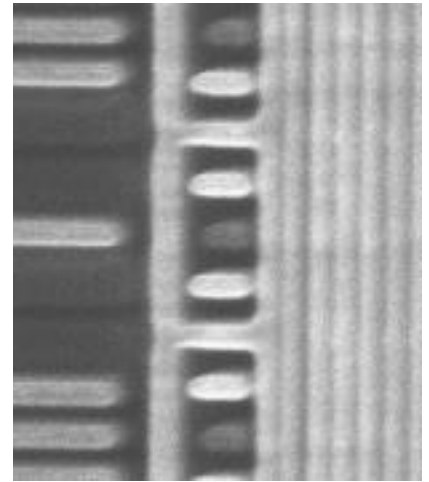
Positively charged dust speck
Attracts SEs from adjacent areas





Voltage contrast imaging

- **Deliberately** charge sample surface
- Charged polygons attract / repel SE's
 - Causes visible brightness shifts in the image
 - Positive charge: attract SE's, less detected, dark image
 - Negative charge: repel SE's, more detected, light image
- Infer memory state, connectivity, etc





Active voltage contrast

- Apply DC bias to the DUT (probe or IO pad)
- Nontrivial, especially for work on lower layers
 - May require a lot of circuit edits or probe needles
- Most flexible technique
 - Full control over exact injection point
 - Can control magnitude and polarity of bias



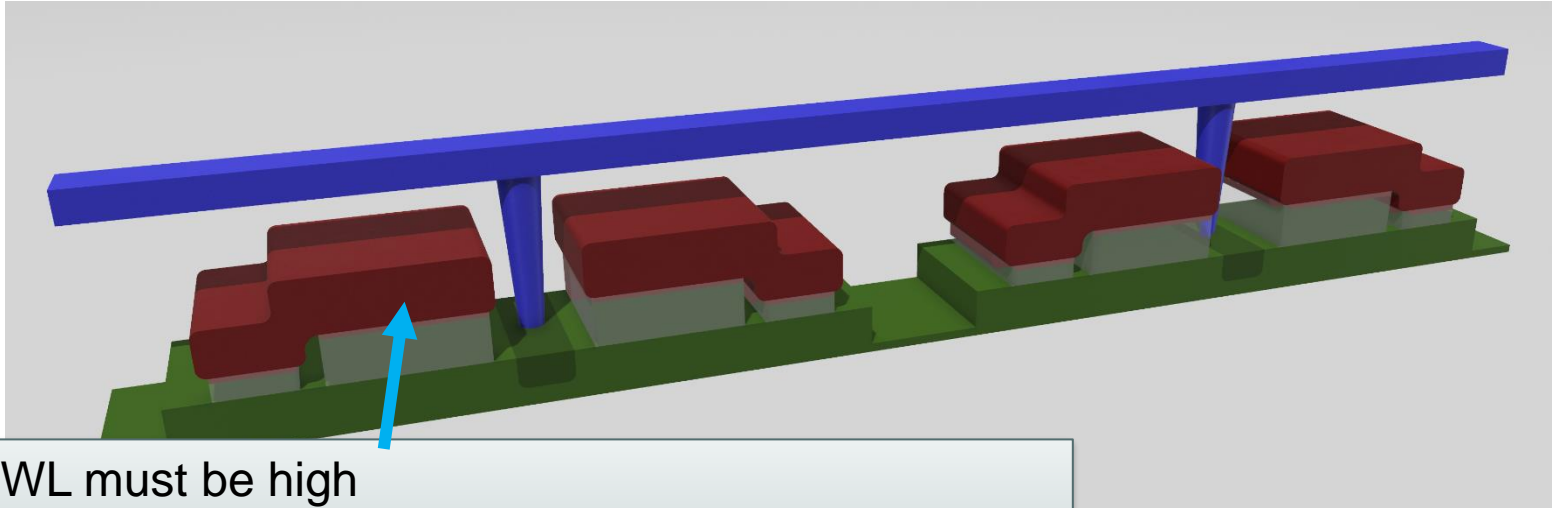
Passive voltage contrast (PVC)

- Imaging beam *also injects charge* into DUT
 - No probe or external power source needed
- Less precise control
 - Everything you can see is also having charge injected
 - Scan rate, coatings, accelerating voltage, etc. affect results
 - Balance charge deposition and bleed-off



Can't do SEM PVC on this bitcell structure

- Electron beam is negatively charged



WL must be high

If select FET is off, we see nothing

So injecting negative bias with e-beam is useless



FIB PVC

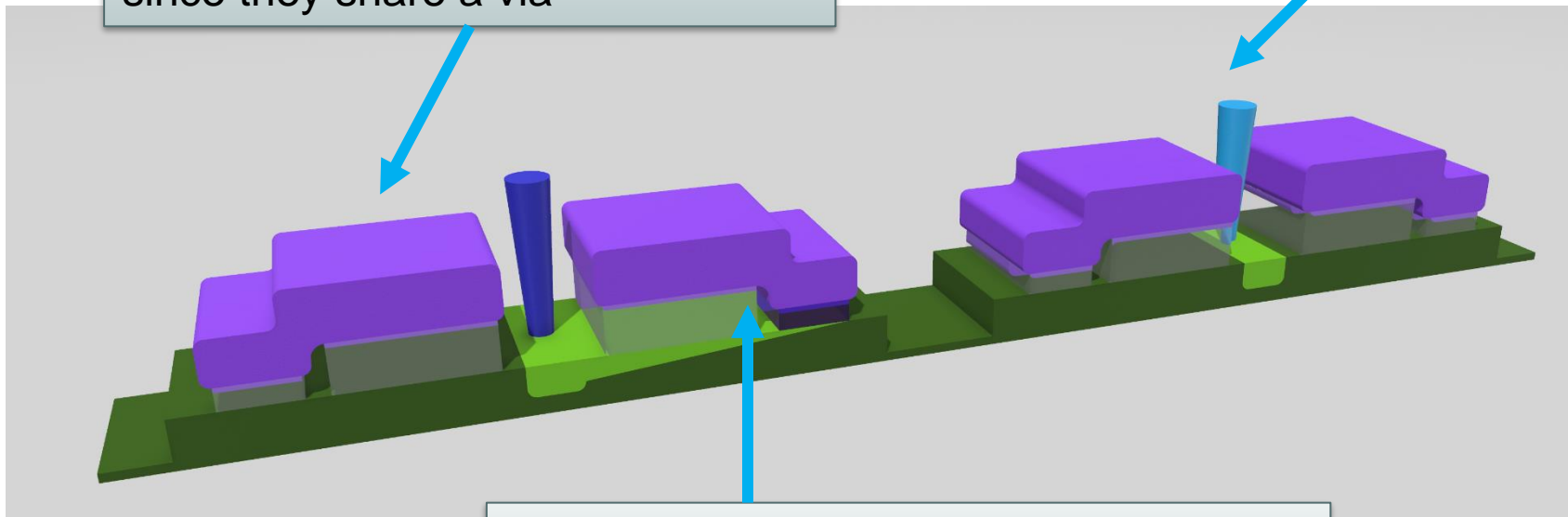
- Use positively charged Ga^+ beam
 - Injects positive charges into sample
- Destructive (sputters DUT surface)
 - Use low beam current, work fast to minimize damage



Mass fuse readout via FIB PVC

Both halves on simultaneously
Cannot distinguish between halves
since they share a via

Deprocess to contacts
No BL remaining



Ion beam injects positive charge on WLs
Positive V_{gs} = bitcell transistor on



First light

Logic 1
Programmed

Logic 0
Unprogrammed

Page 1

Page 0
Factory trim

Cal page
Single 0 bit

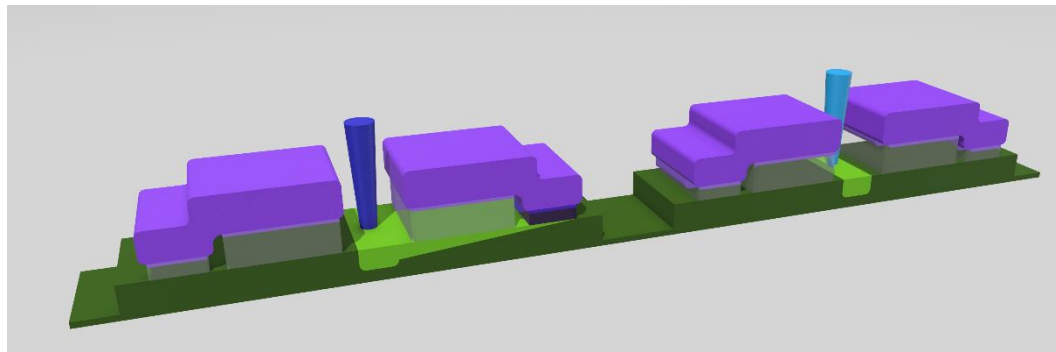
Terrible contrast
But it worked! Time to optimize

I-Beam	Mag	Det	FWD	Tilt	HFW	2 μ m	
30.0 kV	25.0 kX	CDM-E	18.0	52.0°	12.2 μ m	RP2350	



But wait, it's backwards!

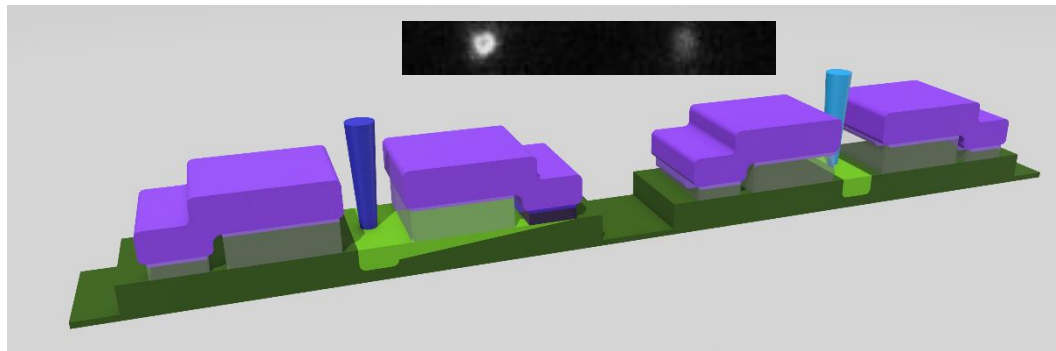
- Original hypothesis: beam charges WL high
 - Therefore, select FET turns on
 - This would make the BL +ve if bit is programmed
 - Which would make it *dark* in the PVC image





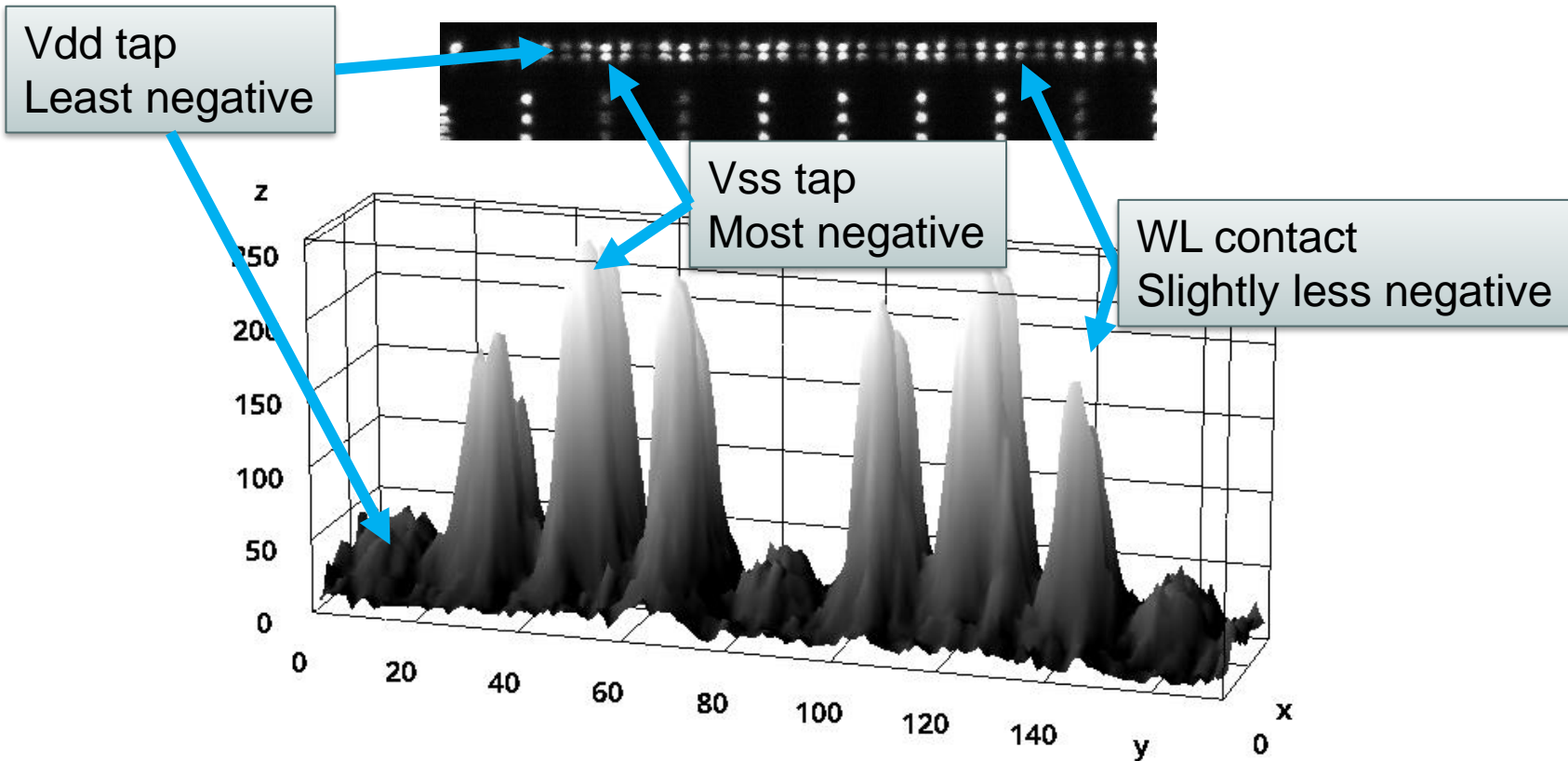
But wait, it's backwards!

- We see the opposite!
 - Programmed bits are *light*
 - But select FET must be *on*, or we won't see anything
 - There's more to the story...





Qualitative voltage measurements via PVC

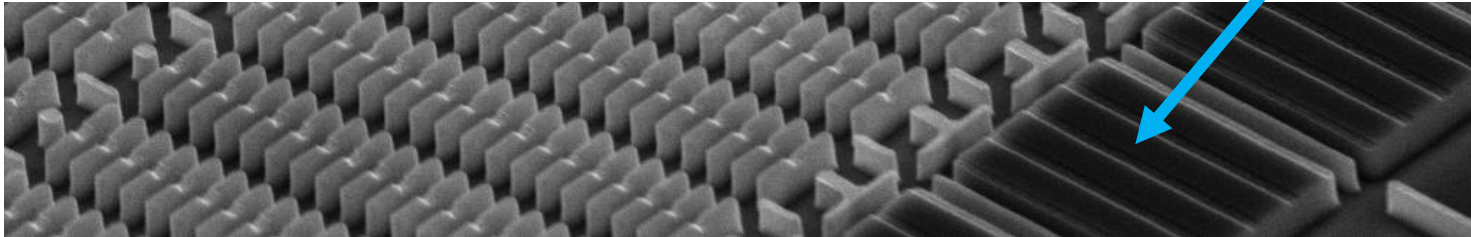




Proposed Contrast Mechanism

- WL builds up *some* positive charge
- But most of it leaks to ground
- Likely sits at a few hundred mV
 - Around V_t , bitcell transistors starting to conduct
 - Can pass ~ 10 pA with reasonably low V_{ds}

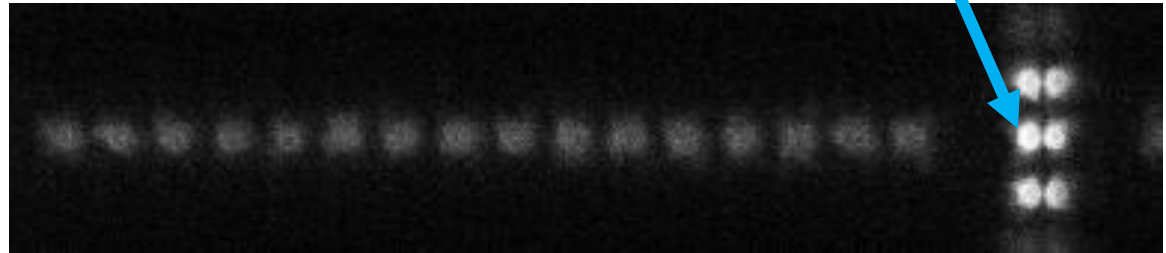
Big WL driver
Lots of channel to leak





Proposed Contrast Mechanism

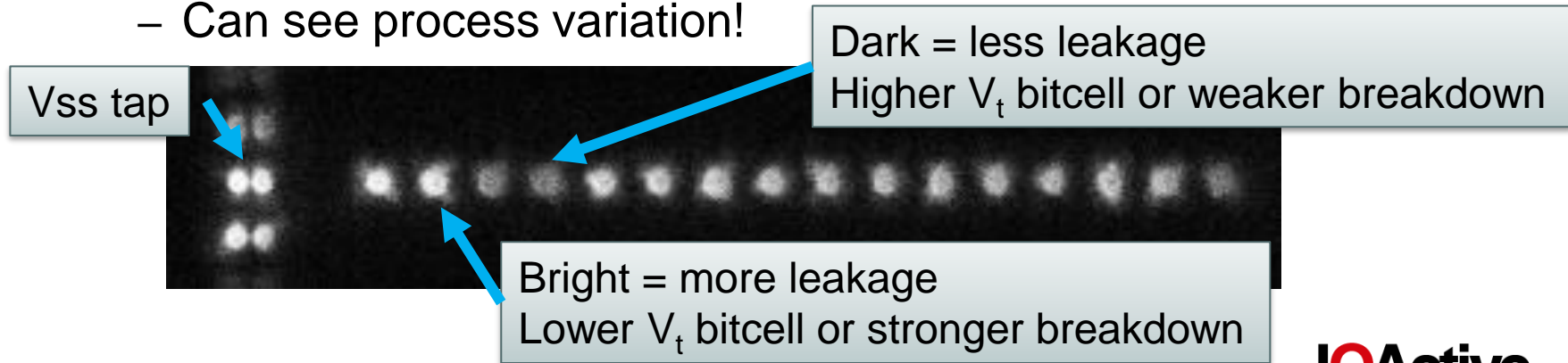
- Unprogrammed bit
 - BL is floating even though select FET is on
 - Beam-induced charge has nowhere to go
 - BL strongly positive, attracts SEs
 - Dark image





Proposed Contrast Mechanism

- Programmed bit
 - BL is connected to WL by $R_{ds(on)}$ of select FET
 - WL (thus BL) is at very low positive voltage
 - Light image, but a bit darker than grounded
 - Can see process variation!





Reversing the address map

- Array has several levels of symmetry
 - Hypothesis: lots of mirroring



Reversing the address map

- Dump factory trim values before decap
 - Known and fairly high entropy, can help ID columns

```
ROW 0x0000: OTP_DATA_CHIPID0 (Part 1/4)
"Bits 15:0 of public device ID. (ECC) The CHIPID0..3 rows contain a
64-bit random identifier for this chip, which can be read from the USB
bootloader PICOBOOT interface or from the get_sys_info ROM API. The
number of random bits makes the occurrence of twins exceedingly
unlikely: for example, a fleet of a hundred million devices has a 99.97%
probability of no twinned IDs. This is estimated to be lower than the
occurrence of process errors in the assignment of sequential random IDs,
and for practical purposes CHIPID may be treated as unique."

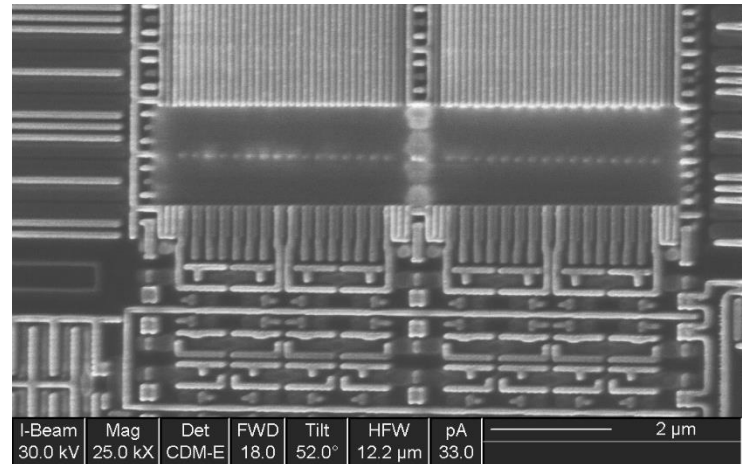
VALUE 0x1af81d

ROW 0x0001: OTP_DATA_CHIPID1 (Part 2/4)
"Bits 31:16 of public device ID (ECC)"

VALUE 0x262f1b

ROW 0x0002: OTP_DATA_CHIPID2 (Part 3/4)
"Bits 47:32 of public device ID (ECC)"

VALUE 0x0adde8
```





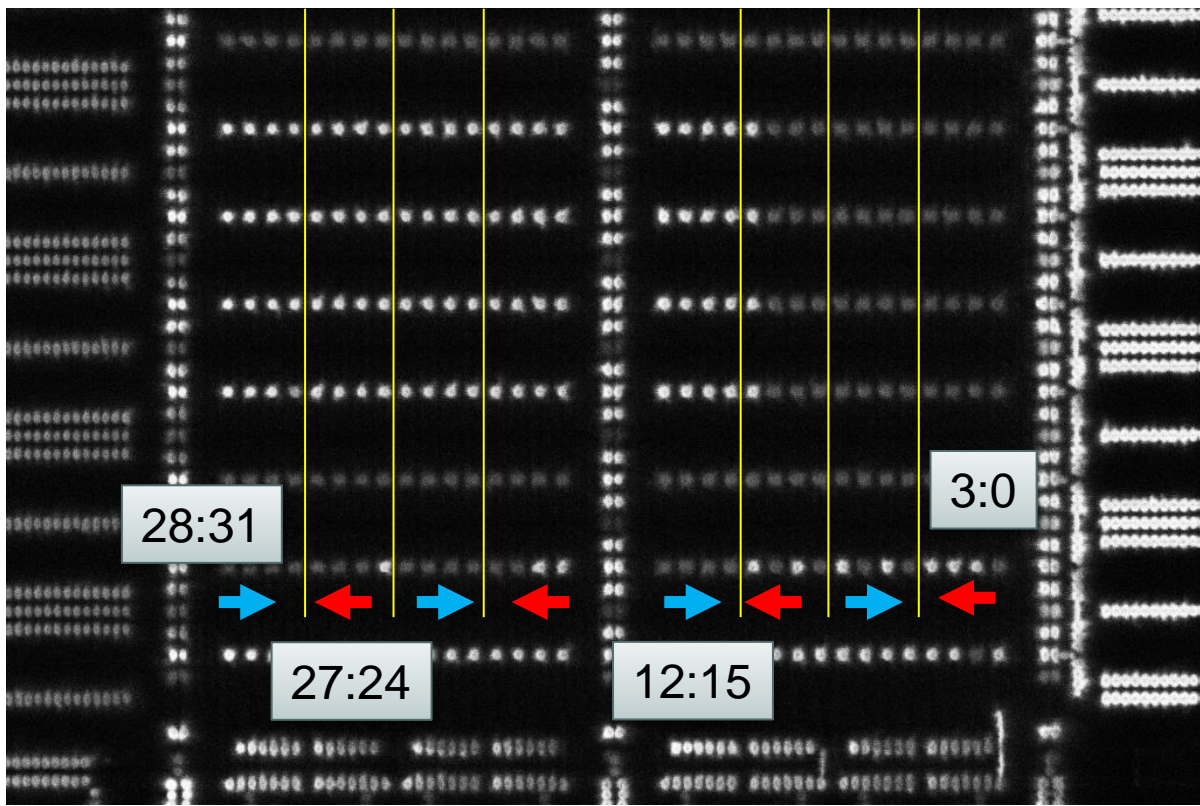
Reversing the address map

- Program our own test patterns
 - Different data in each bit plane to ID columns
 - Horizontally + vertically asymmetric to ID mirroring
 - Find the challenge key
 - And, of course, have some fun in the process

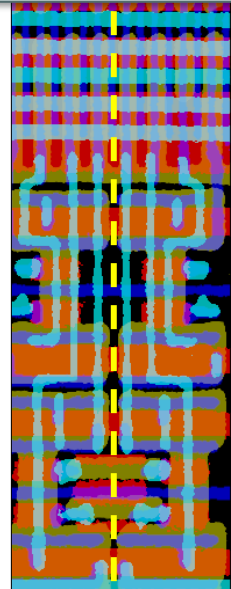


Column ordering

West side of array shown
East side mirrored L-R



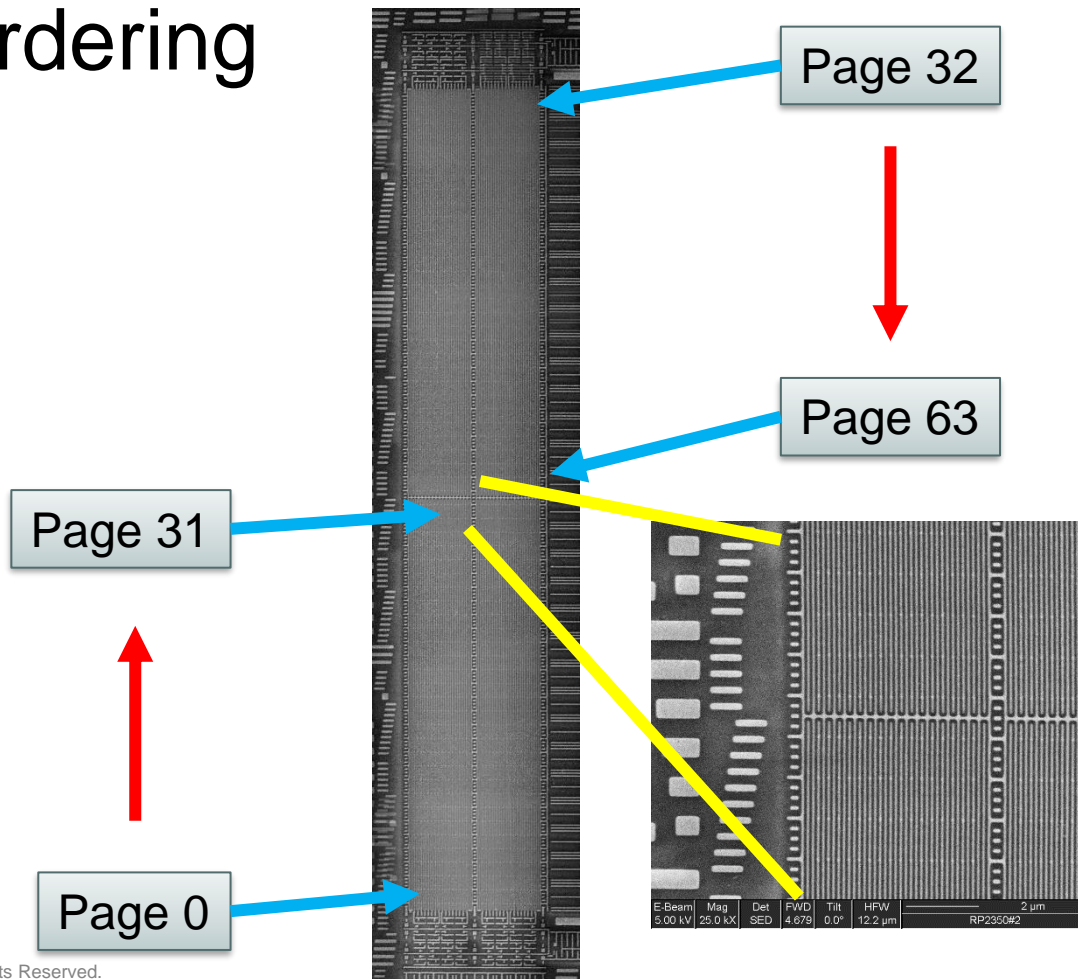
Even/odd nibbles
mirrored





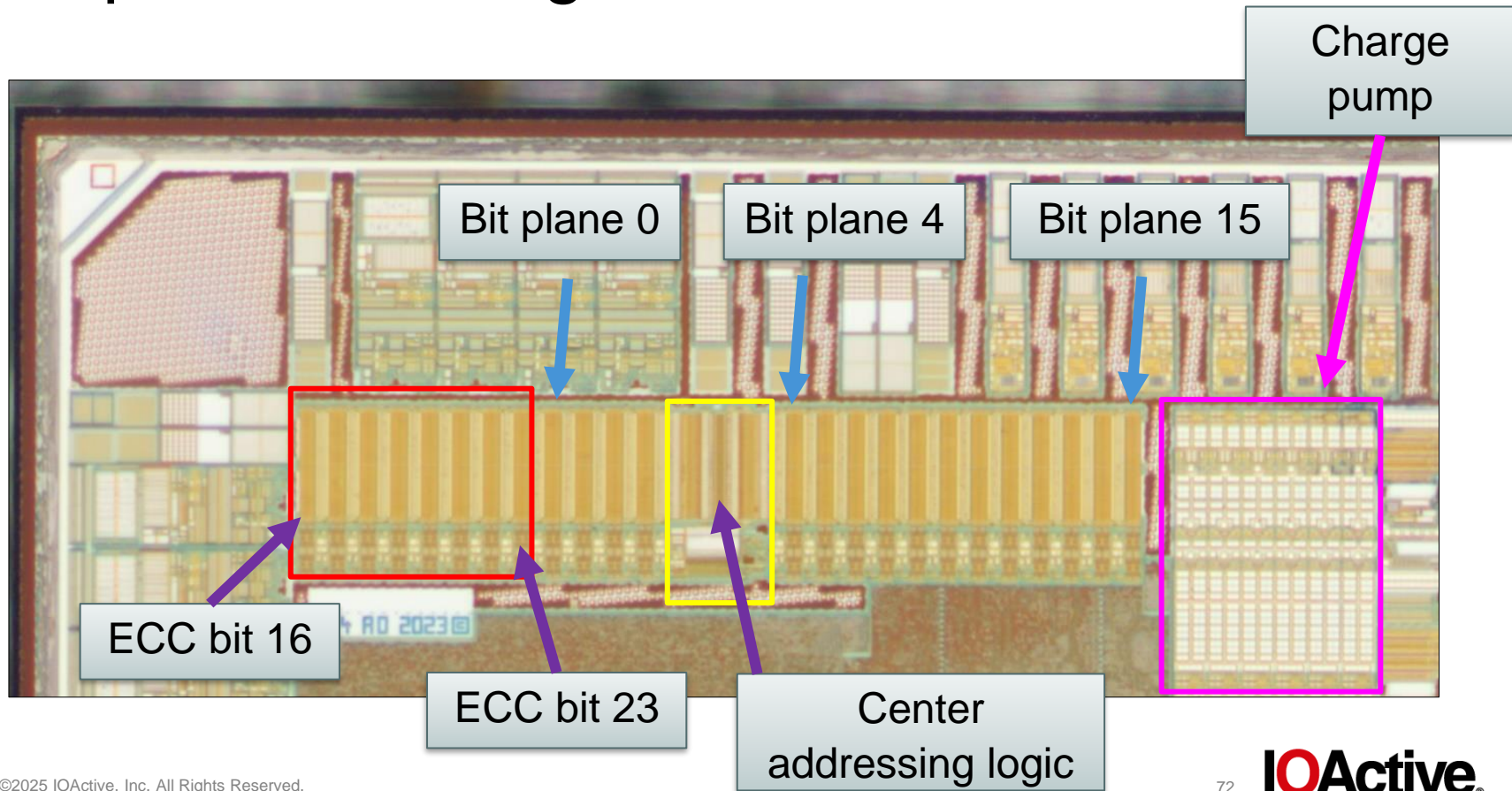
Row ordering

M1 view



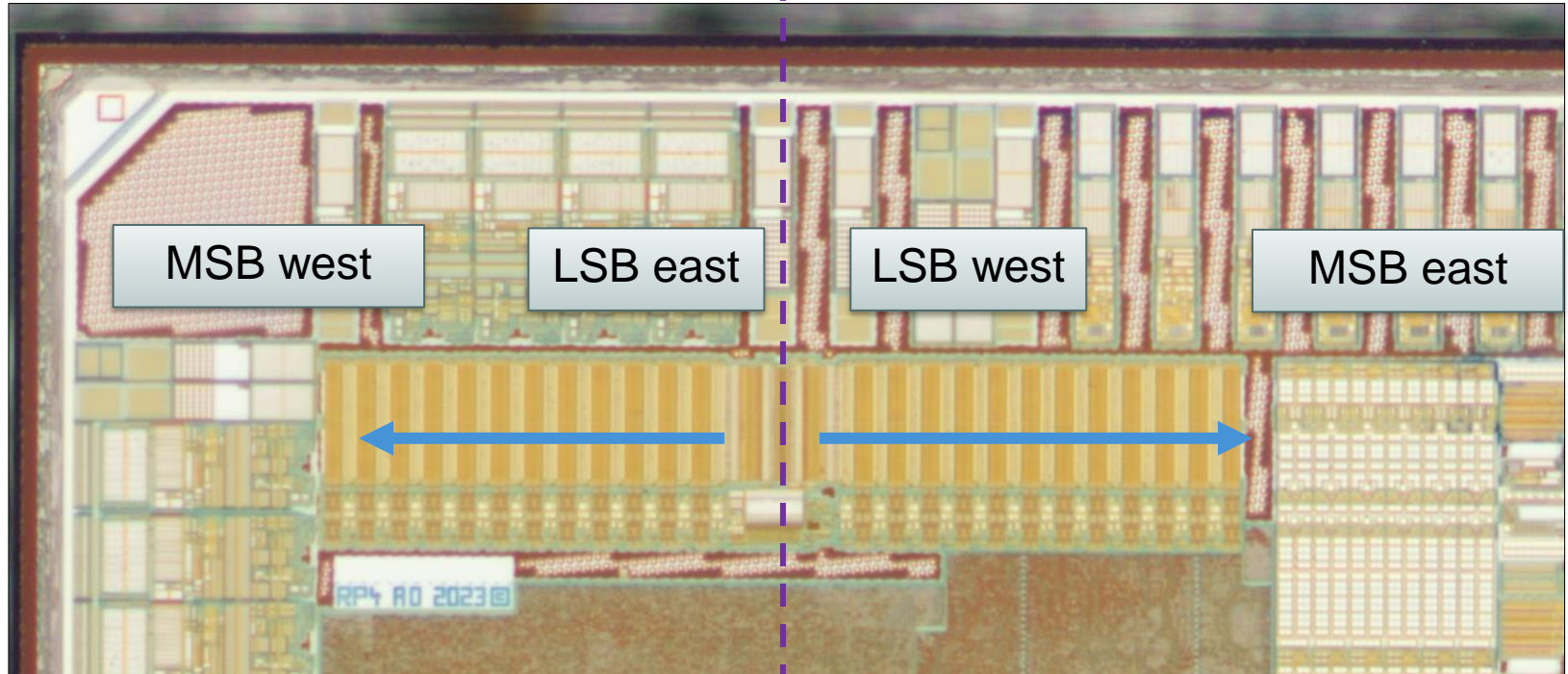


Bit plane ordering





Bit plane mirroring





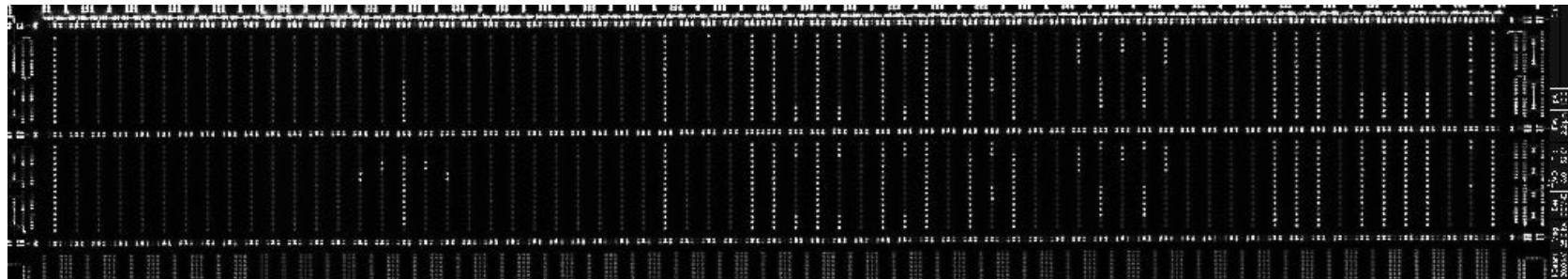
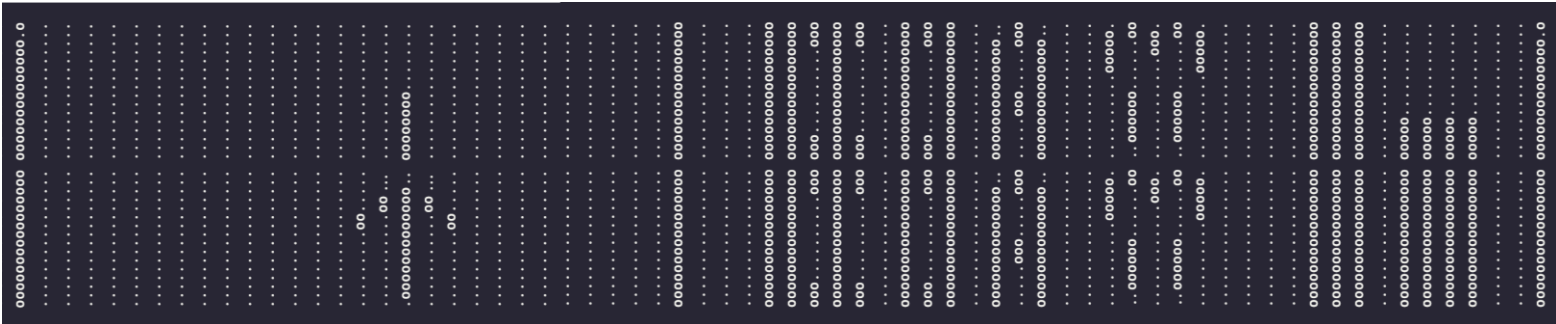
Fuse rendering script

- Python script: fuse values in, ASCII art map out



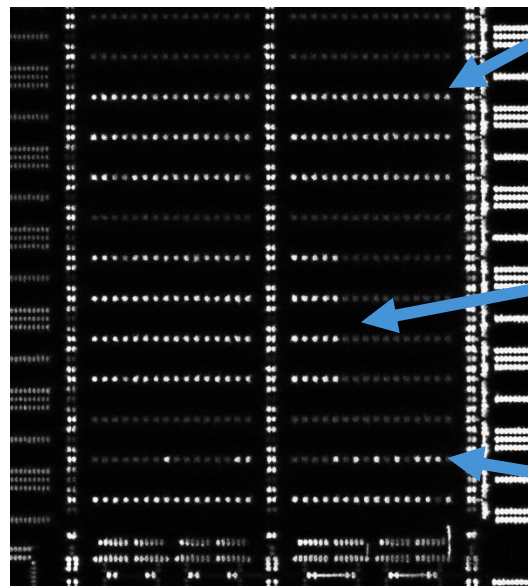
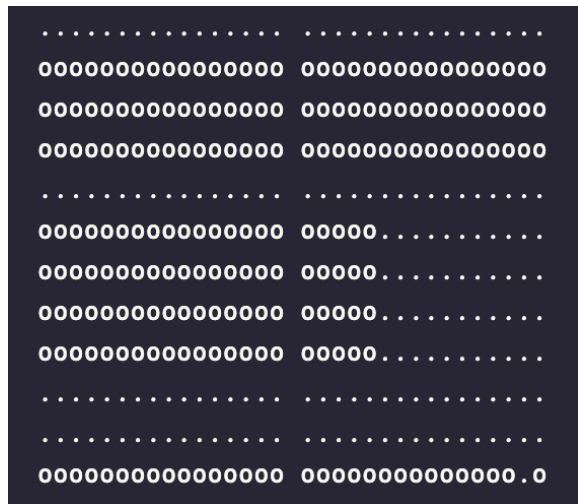


Full bitplane extraction test





Test pattern closeup



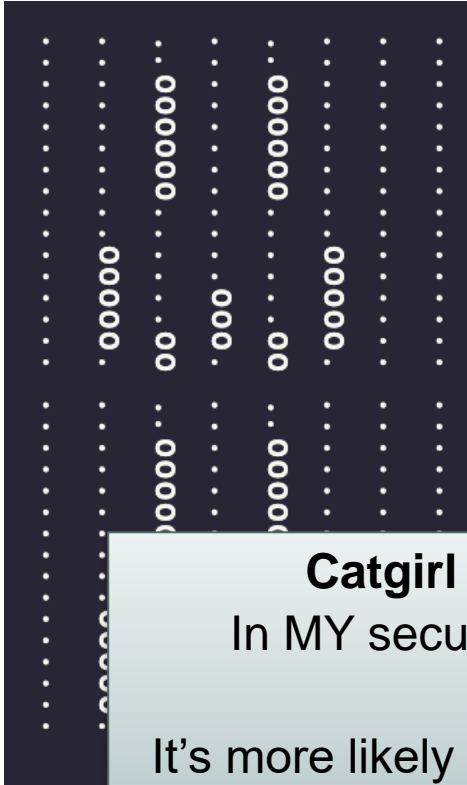
Even-odd test
(all lit up)

Bit plane
index (21)

Factory
trim data

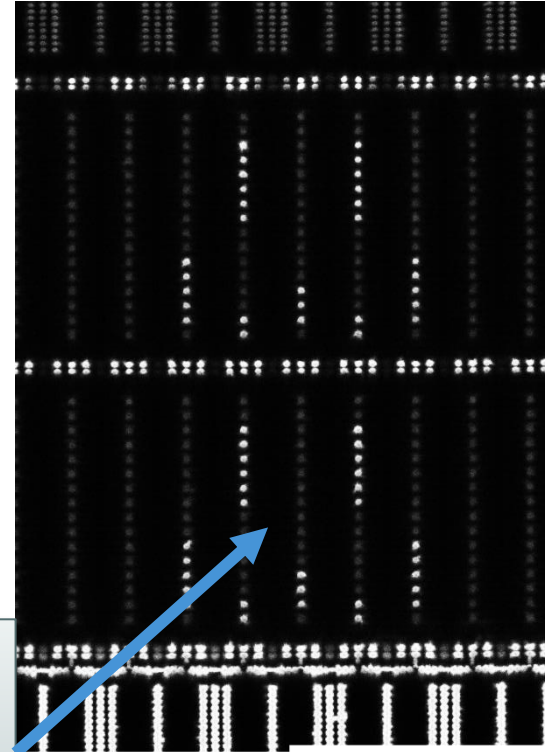


Test pattern closeup



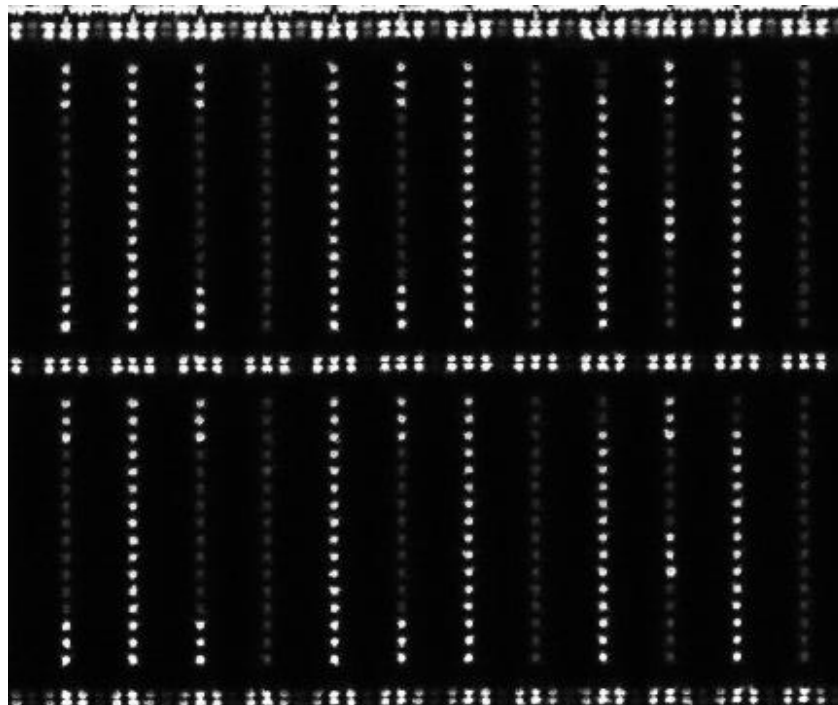
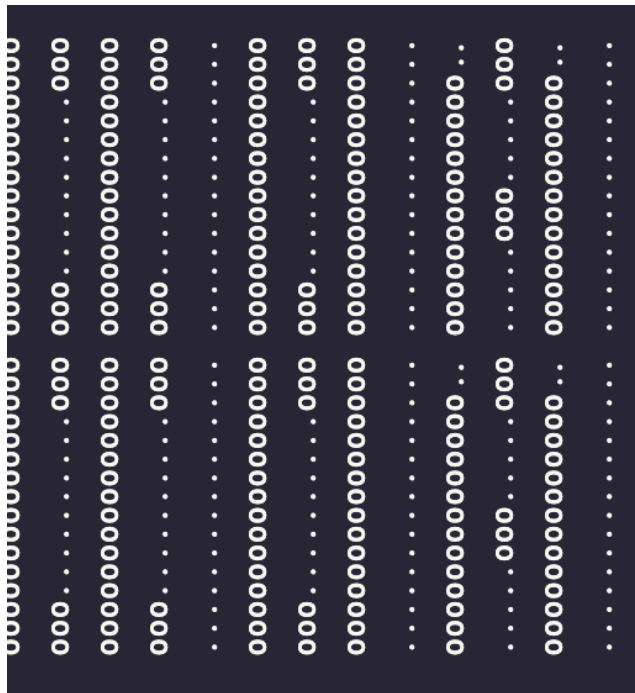
Catgirl hackers?
In MY secure boot keys?

It's more likely than you think =3



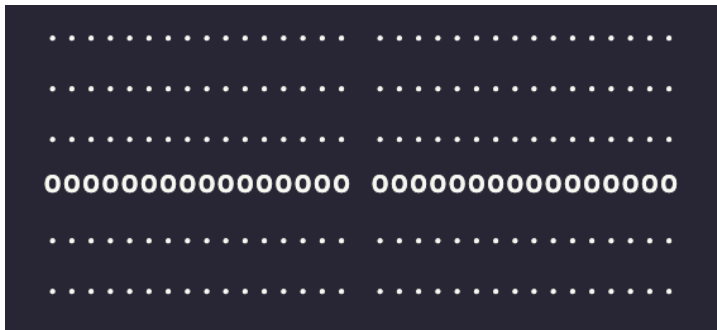


Test pattern closeup

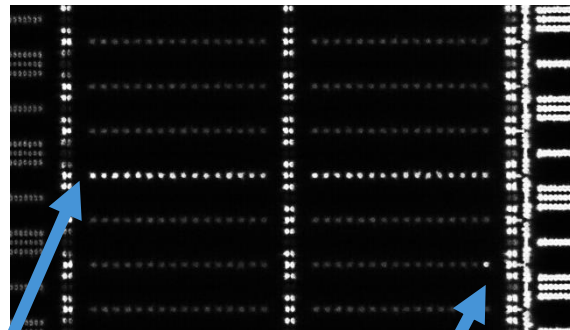




Test pattern closeup



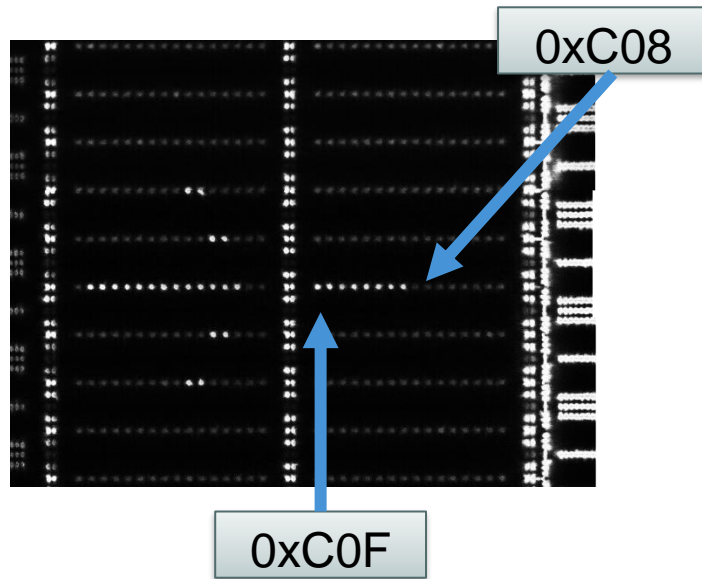
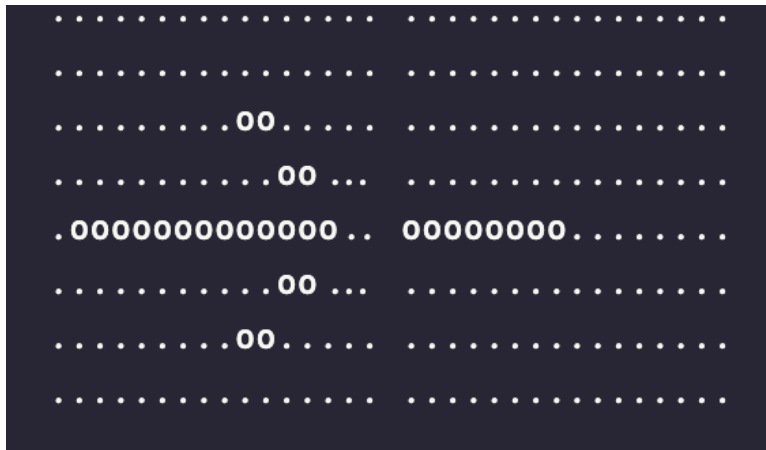
Page 60 0xFFFF
Just above center line
Confirms vertical mirroring



Page 62
OTP_DATA_PAGE0_LOCK0
at 0xf80 is set to lock trim/cal



Test pattern closeup



The goal is easy: Find an attack that lets you dump a secret hidden in OTP ROW 0xc08 - the secret is 128-bit long, and protected by `OTP_DATA_PAGE48_LOCK1` and RP2350's secure boot!



Possible improvements

- Use probe or FIB platinum to ground half the WLs
 - Should allow readout of even / odd halves separately
 - First experiment failed, haven't had time to try again
- Play w/ scan speed and reduced area scans
 - Bias everything –ve with e-beam
 - Then scan I-beam over WL (to charge) and BL (to image)
 - Make sure not to get *any* beam energy on the opposite WL
 - If perfectly focused, *might* work



Mitigations

- PVC is physics, impossible to prevent
- Proper fix (requires silicon spin)
 - Encrypt data at rest so fuse dump is worthless to attacker
 - Force adversary to RE many Mgates to find hardwired key
- Near term: increase adversary workload
 - Store 0xFF or ~key in opposite half of paired fuses
 - Blocks the trivial mass-readout attack
 - Requires more work to dump one WL at a time



Mitigations

- Spread key across many physical words
 - 1 word in N pages is $N \times$ the work to dump vs data in same page
 - Store data all over the place and hash or XOR down
- Goal: force a FIB edit of ~all 2048 WLs in the fuse array
 - Can't quite use 100% of rows because factory trim etc
 - But you can make the adversary hate you!



Conclusions

- Antifuses aren't as invisible as claimed
- A few k\$ of FIB time goes a long way
- Any secret in on chip memory can and will be extracted
- Encrypting / obfuscating fuse data raises the bar



Acknowledgements

- Daniel Slone, Mario Cop, Tony Moor (IOA)
 - Decap, deprocessing, some imaging
- Lain Agan (IOA)
 - Memory map RE, 3D renders, analysis script
- Entropic Engineering
 - RP2350s were hard to find right after launch, they hooked us up
- Raspberry Pi
 - For inviting the hacker community to pwn their product



Questions?