CSCI 4974 / 6974
Hardware Reverse Engineering

Lecture 9: Memory addressing / mask ROM

Quiz

Microscopy lecture

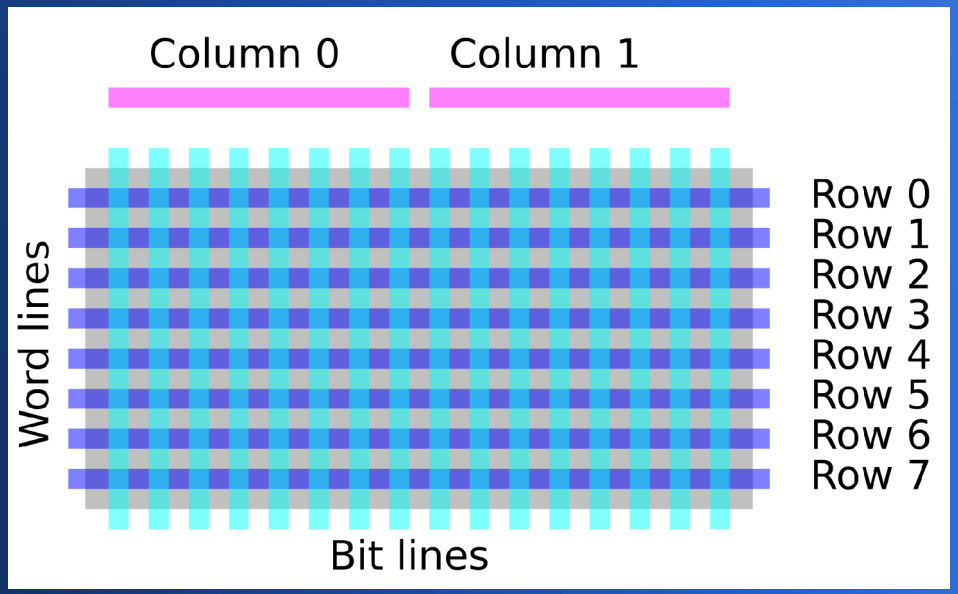
- Last ~10 slides from lecture 8 were skipped
- Cover them today

Generic memory components

- Address bus
- Row addressing logic
- Column addressing logic
- Data bus
- Memory array

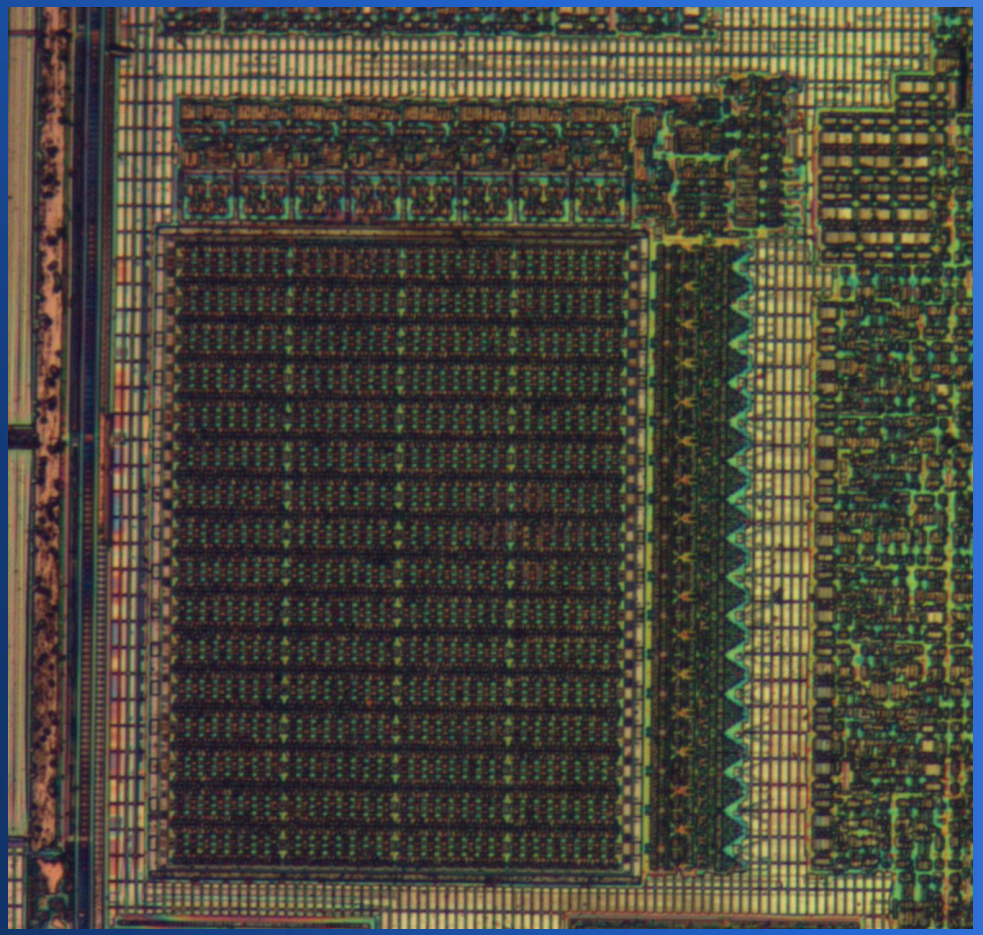
2D memory addressing

- Logical structure of memory is 1D
- 8 x 1M bit array is physically impractical!
  - Use 2D structure instead
  - Need col muxing
- Mapping of 2D linear addresses may vary



PIC12F683 SRAM

- 128 bytes
  - 32 rows
  - 4 cols of 8 bits

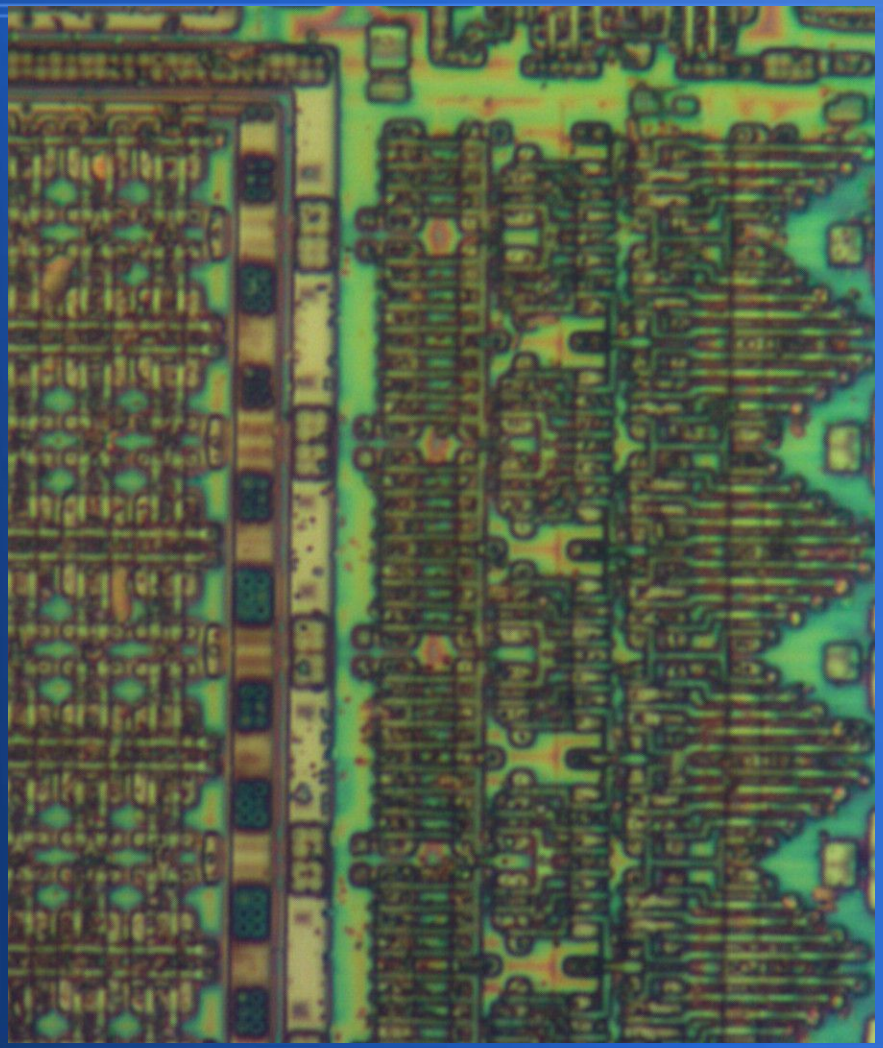


Row decode logic

- Input: N-bit row address bus
- Output: 2^n word lines
- WL[0] = ... & !A3 & !A2 & !A1 & !A0
- WL[1] = ... & !A3 & !A2 & !A1 & A0

Example row decode logic

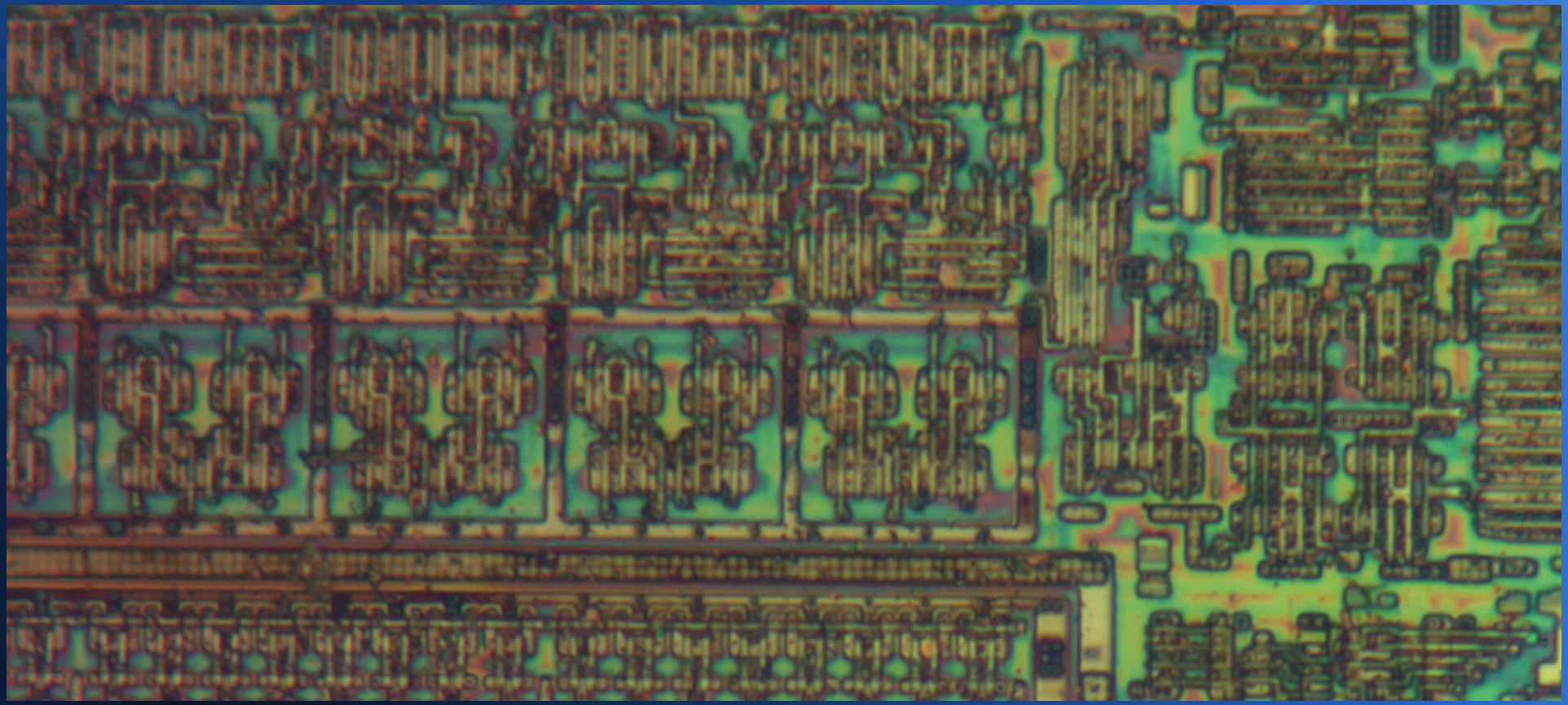
- PIC12F683 SRAM
- Data bus on M1 at right
- AND gates at center
- WL buffers at left
- Local interconnect on M1
- Word lines on poly



Column address logic

- For writable memories
  - Float BL except during writes
  - During writes, drive BL for appropriate column
- For all memories, during reads
  - Precharge before reads if necessary
  - All columns have data on them
  - Mux the one of interest to the data bus

Example column address logic



Why mask ROM?

- Lowest cost per bit of any memory tech
  - Single-transistor cells
  - No additional masks required (unlike flash etc)
- Highest density of any memory
- Immune to magnetic fields etc
- Can't be corrupted or tampered by software
  - But wait for lecture 14 ;)

Disadvantages

- Can't (practically) be patched after manufacture
  - Requires massively costly mask respin
  - Need fully debugged code beforehand!

Mask ROM vs OTP

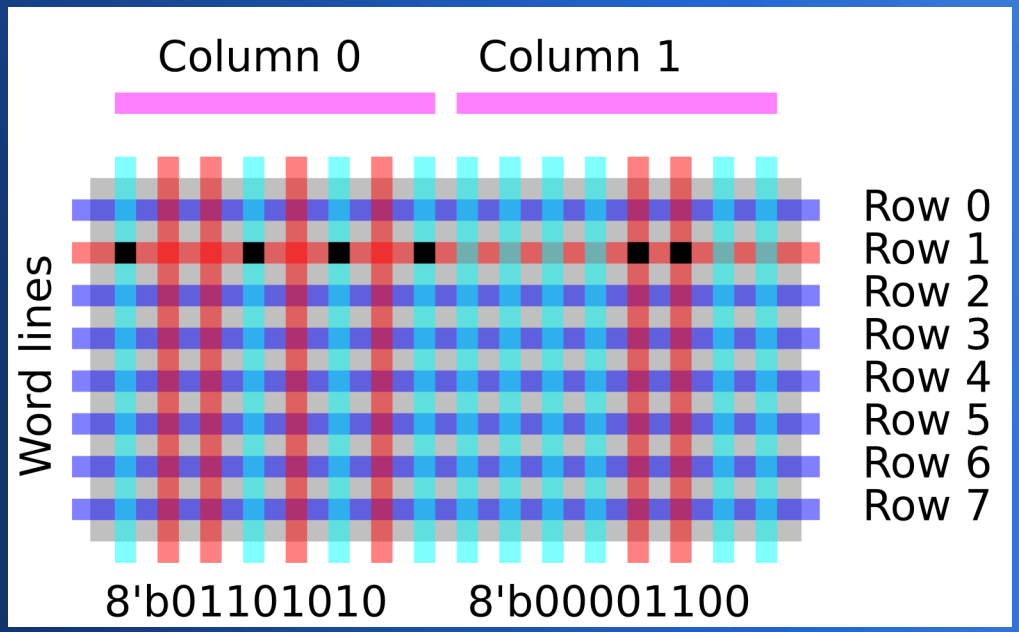
- Mask ROM
  - Physically hard-wired
  - Dedicated mask for each ROM image
- OTP (one-time-programmable) ROM (PROM)
  - Programmed (once) after manufacture
  - Same mask for all ROM images
  - Less dense, requires write circuitry
  - Covered in the next lecture

The hacker's viewpoint

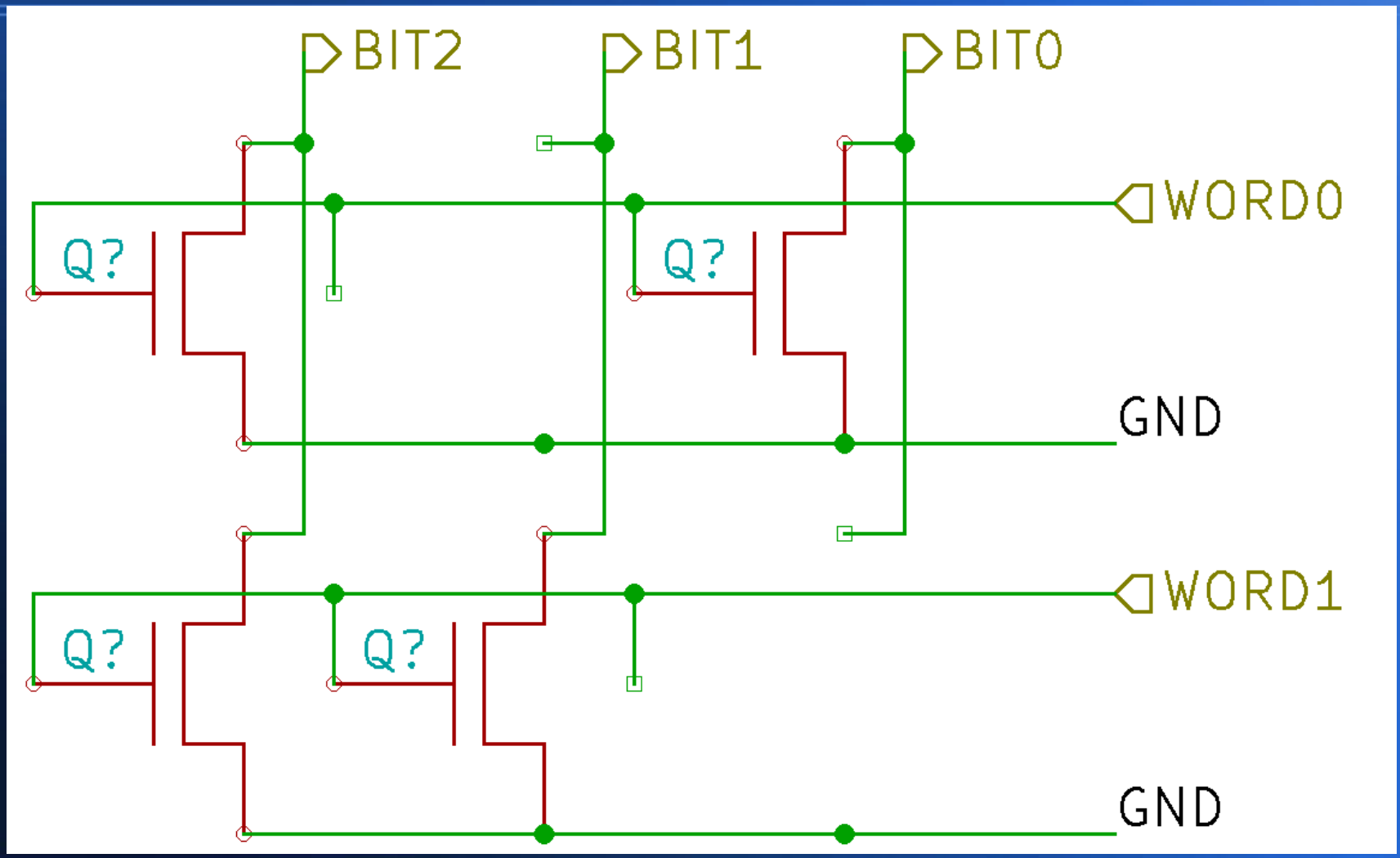
- Target has data in mask ROM
- How do we read it?
- Full ROM circuit analysis usually not required
  - We just want the data
  - Figure out enough to know what goes where

NOR mask ROM

- Pull all BL weakly to Vdd, assert one WL
- Switch to Vss may be at each WL/BL junction
  - WL off? Do nothing
  - WL on? Pull BL low



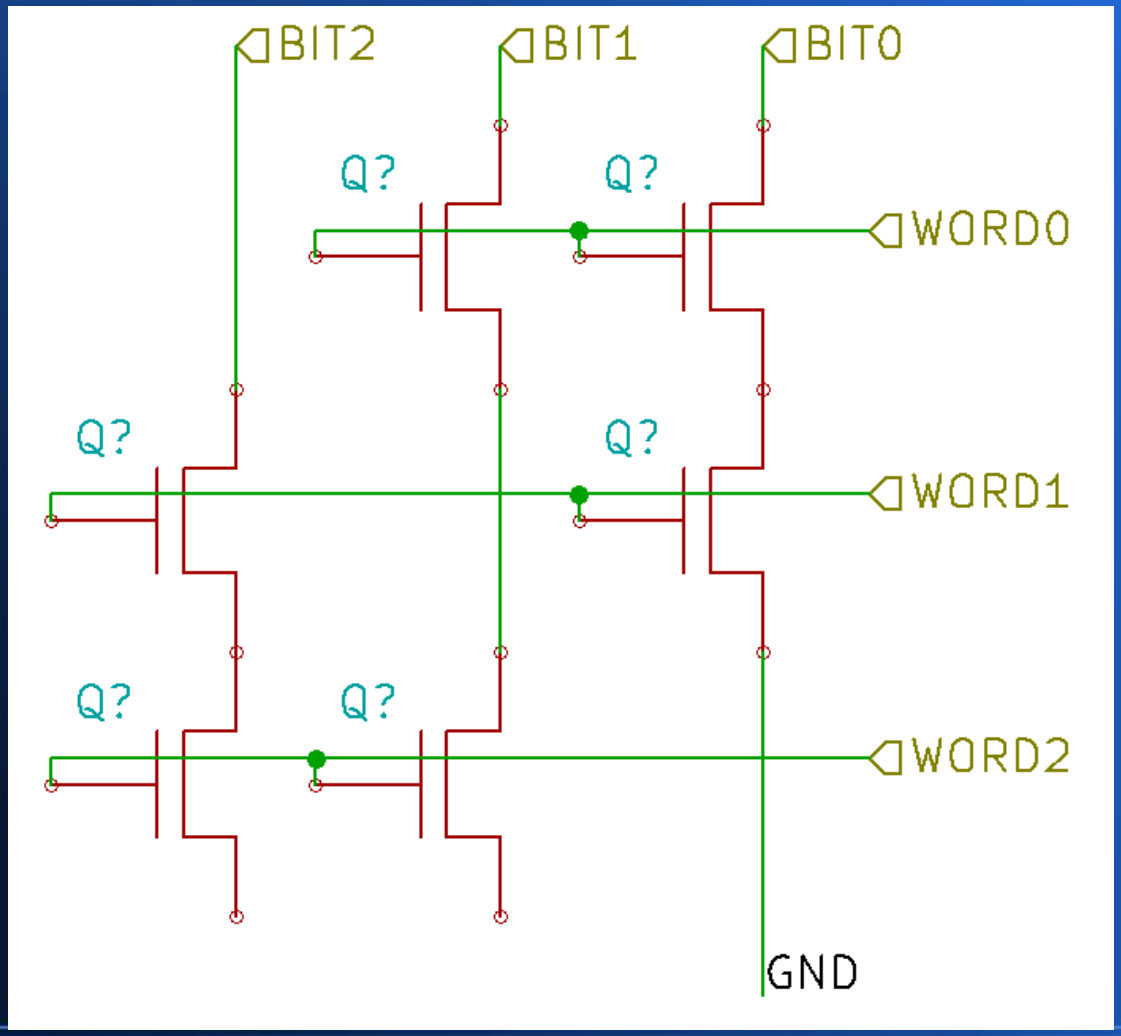
NOR mask ROM



NAND mask ROM

- Pull all BL weakly to Vdd
- Assert all but one WL
- Switches in series from BL to ground
- If no switch, output goes low
- If (open) switch, output stays high
- Denser than NOR, but slower

NAND mask ROM



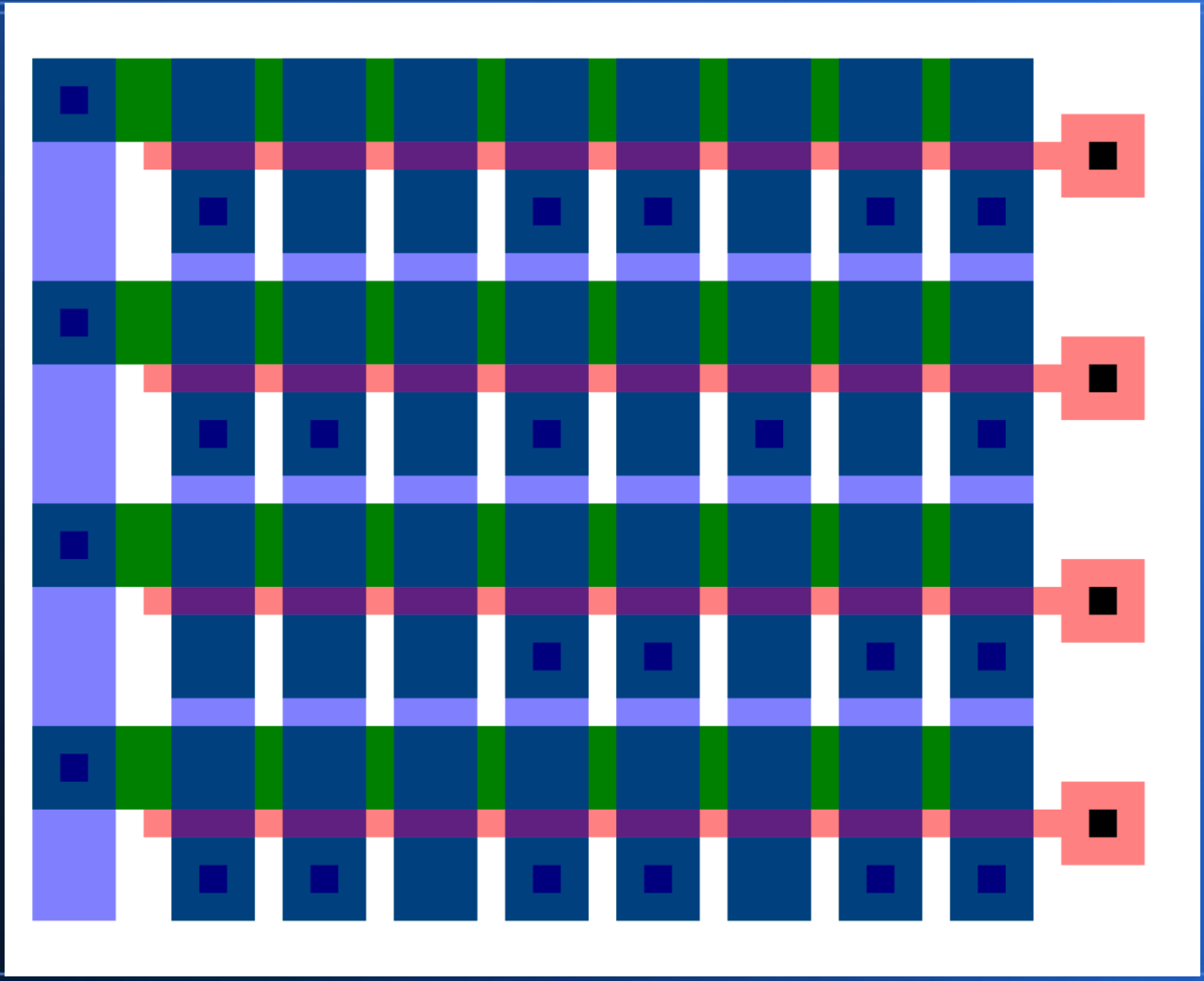
Mask ROM technologies

- Via based
- Metal based
- Implant based

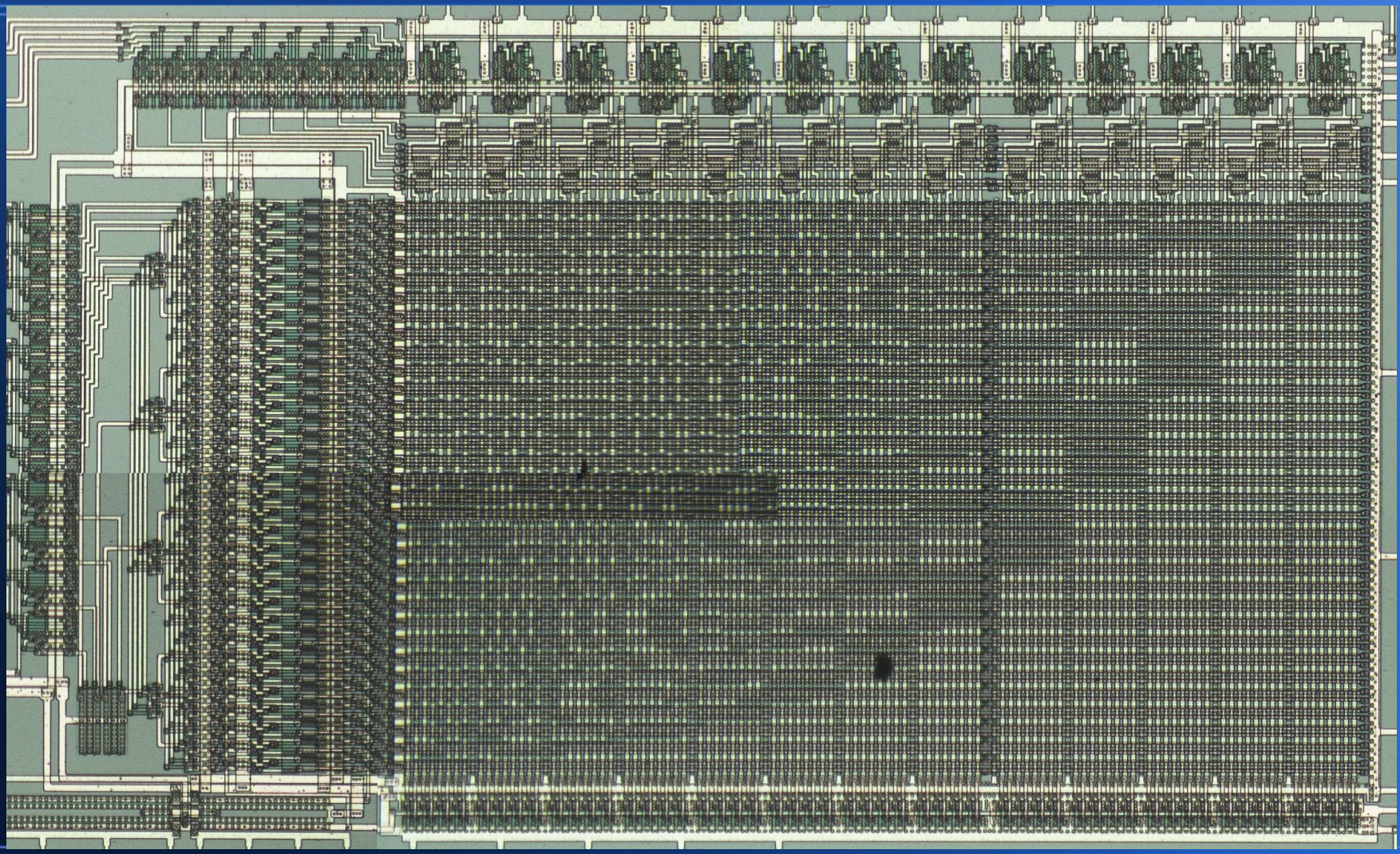
Via ROM

- Usually NOR type
- Transistor is always present
  - FEOL processing can be done for all chips
  - Then separate wafers for each product
- Remove M1-active via(s) to disable transistor
  - Single mask change
- Then standard M1 + interconnect routing on top

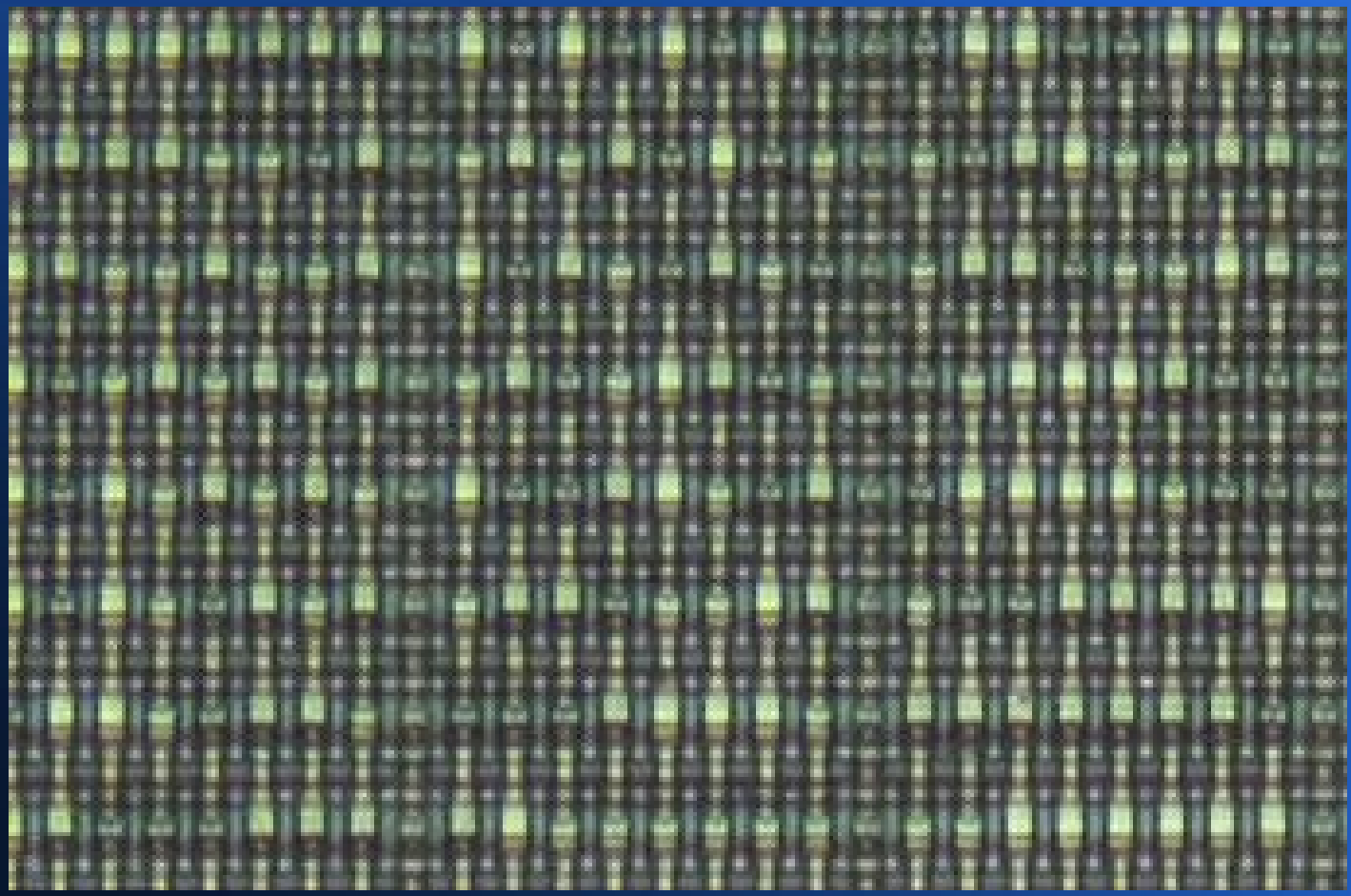
Via ROM layout



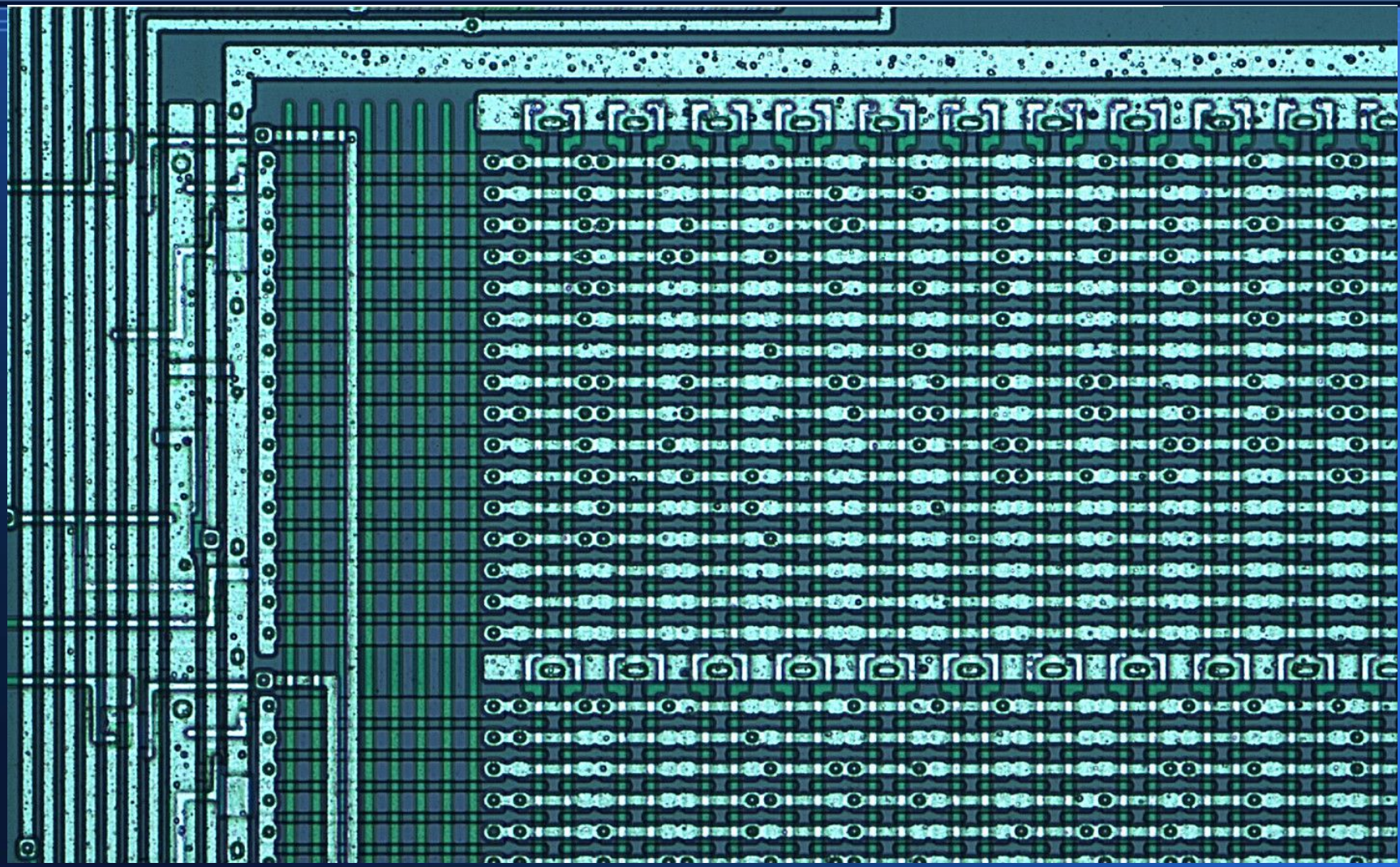
Via ROM (Roland LA32)



Via ROM (Roland LA32)



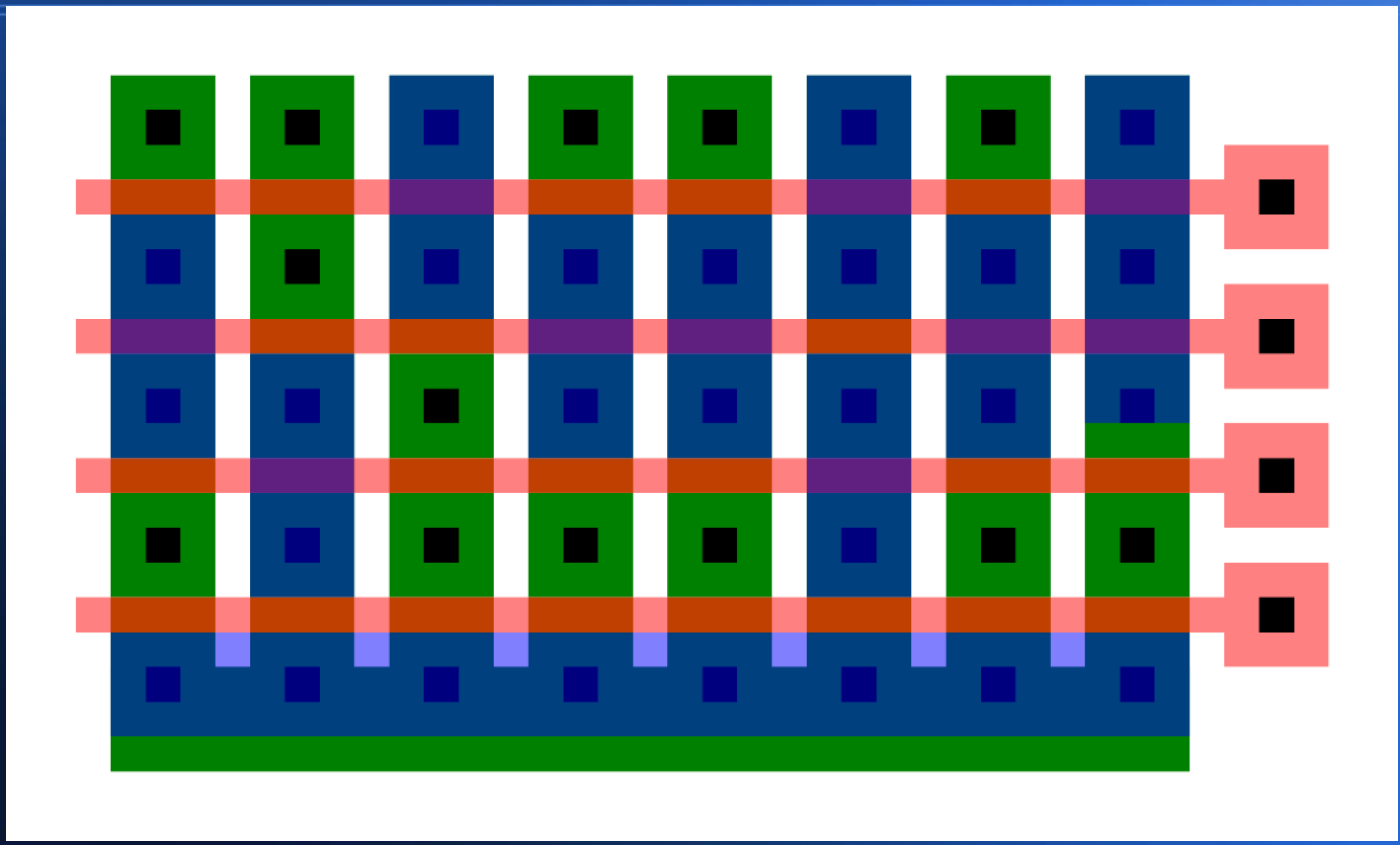
Via ROM (unknown source)



Metal ROM

- Usually NAND type
- Transistor is always present
- Short out transistors with M1

Metal ROM layout



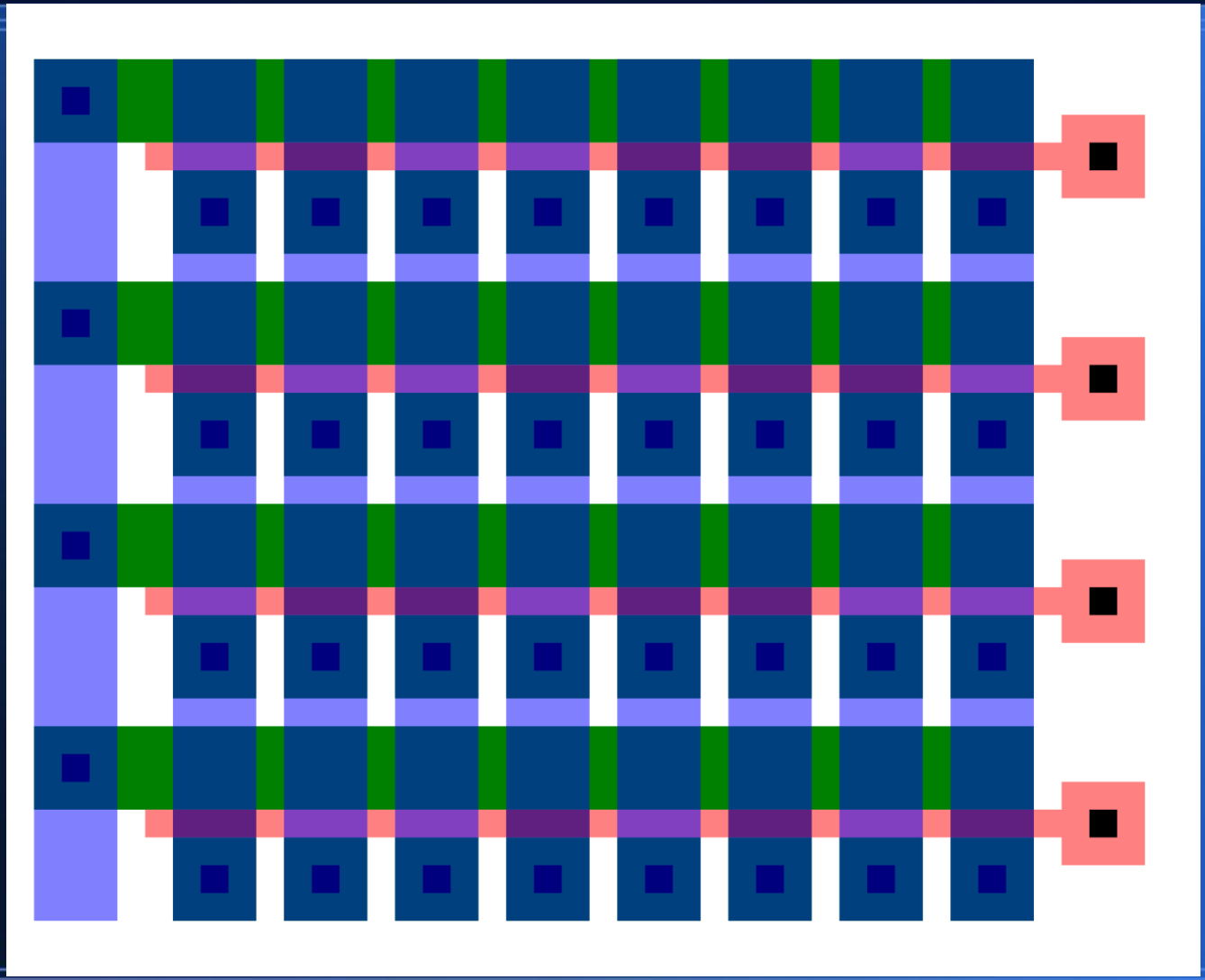
Metal ROM example

- TODO: Example die photo if we can find one

Active area ROM

- Usually NOR type
- Can be read optically after deprocessing
- Cut or don't cut channel for transistor

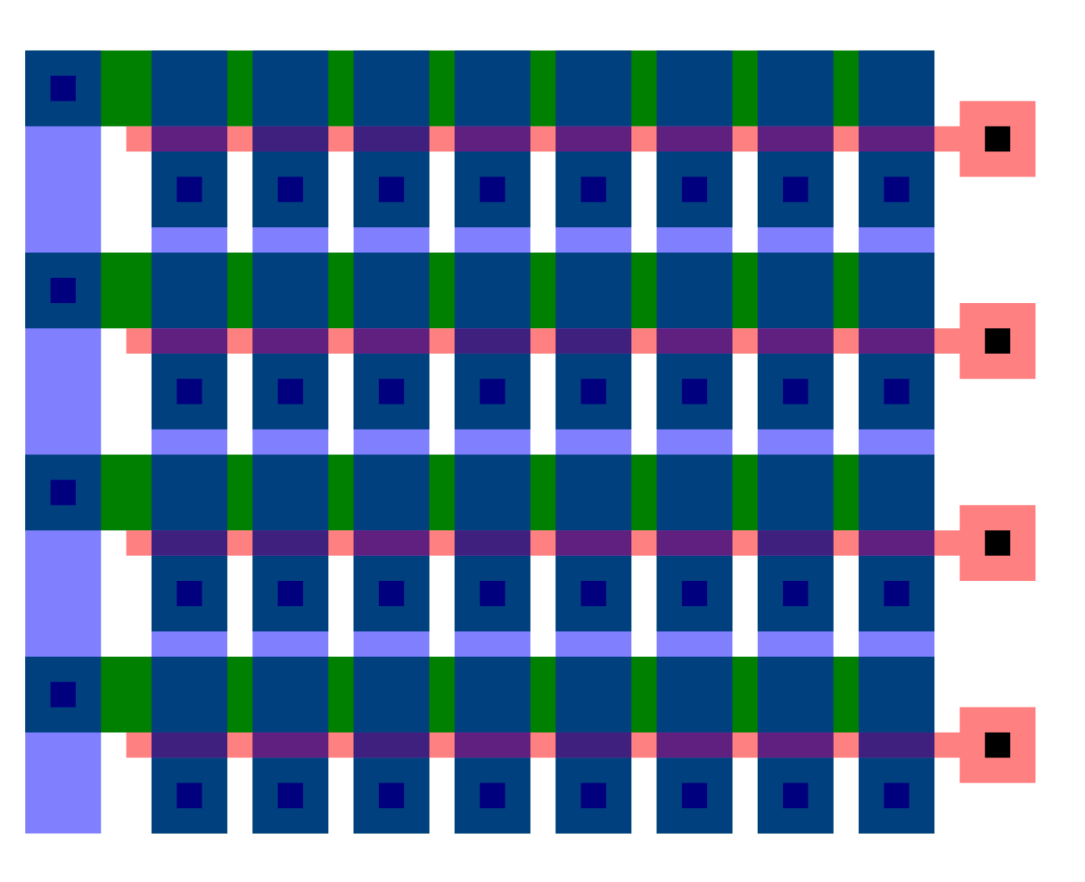
Active area ROM layout

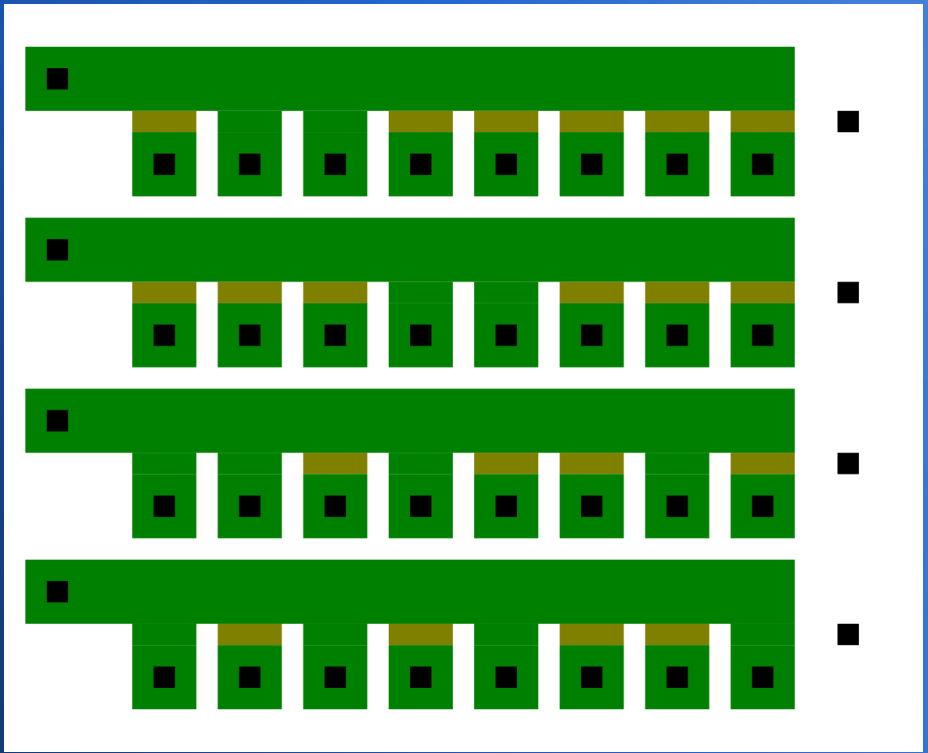


Implant ROM

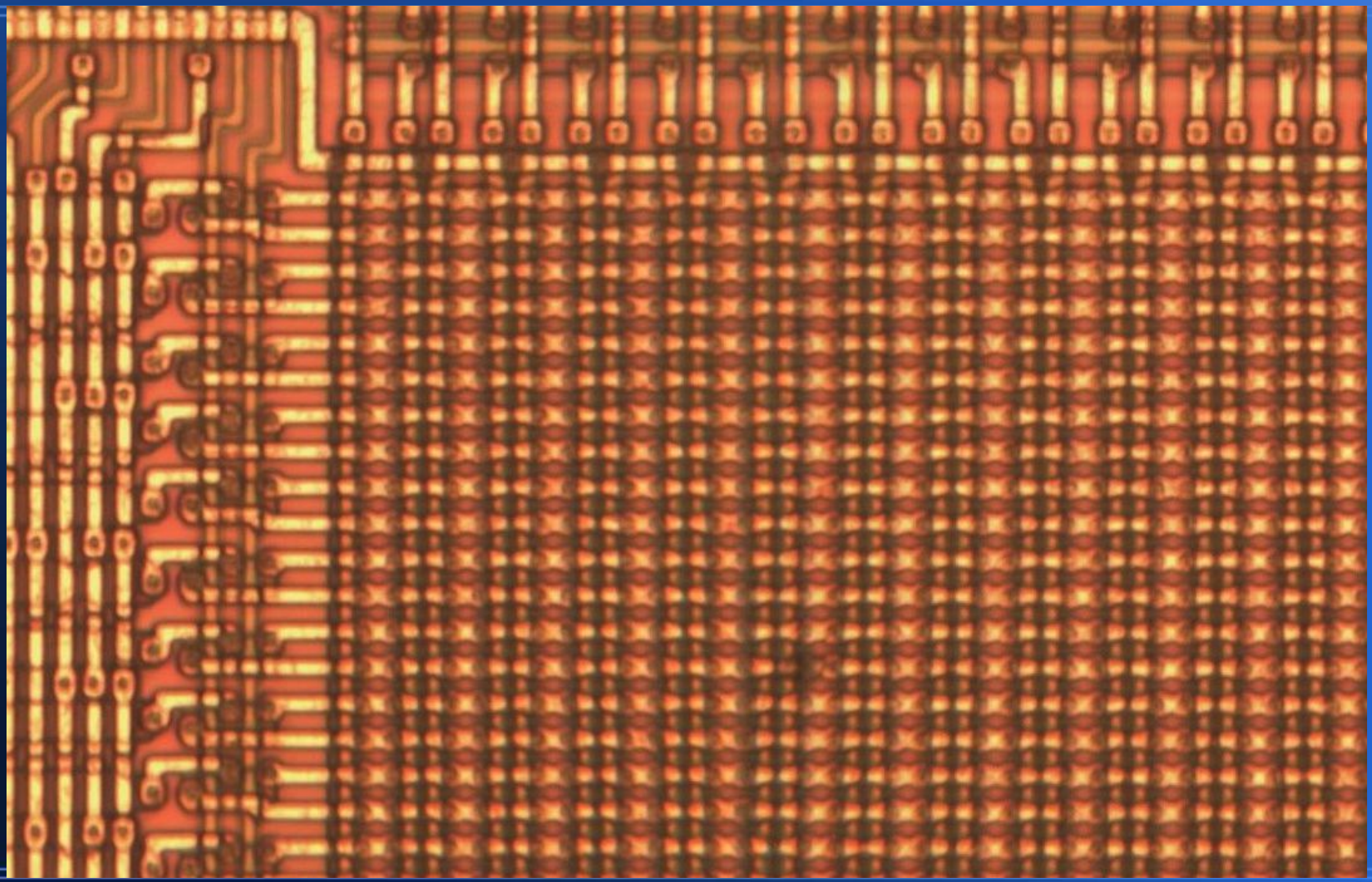
- Can be NAND or NOR
- Cannot be read optically
  - But can be revealed with SCM or Dash etch
- To short out transistor (for NAND ROMs)
  - use weak implant to shift Vt = 0
- To open transistor (for NOR ROMs)
  - use weak implant to shift Vt > Vdd

Implant ROM layout

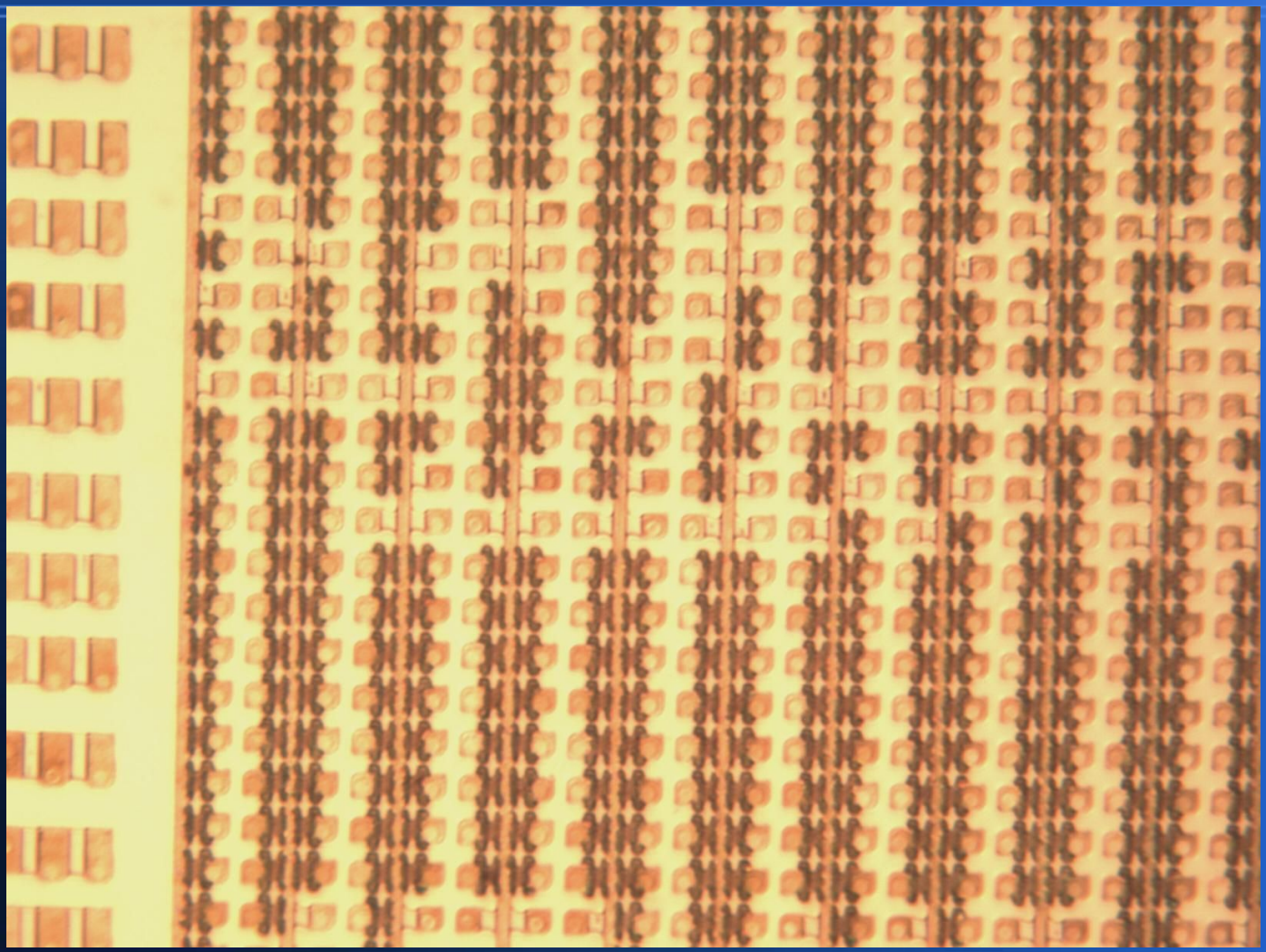




Implant rom top metal



Implant ROM after Dash etch



Reading data from mask ROM

- Need to make bits visible first
  - Via: Delayer to via layer (if nonplanar)
  - Metal: Delayer to metal layer
  - Implant: Strip to active, then Dash etch
- Acquire imagery

Reading data from mask ROM

- Extract the 2D bit pattern
  - Can be done manually or with machine vision
  - Polarity and structure aren't yet known
- Figure out structure
  - Which bit layout is 1 and which is 0?
  - How do 2D addresses map to linear?

Structure analysis

- Many possible layouts
  - Interleaving of several bytes/words per row
  - Top-down or bottom-up address increment
  - Interleaved or concatenated banks
- Several ways to make sense of it all
  - Reverse the decode circuitry
  - Trial and error until dump makes sense

Questions?

- TA: Andrew Zonenberg <azonenberg@drawersteak.com>
- Image credit: Some images CC-BY from:
  - John McMaster <JohnDMcMaster@gmail.com>