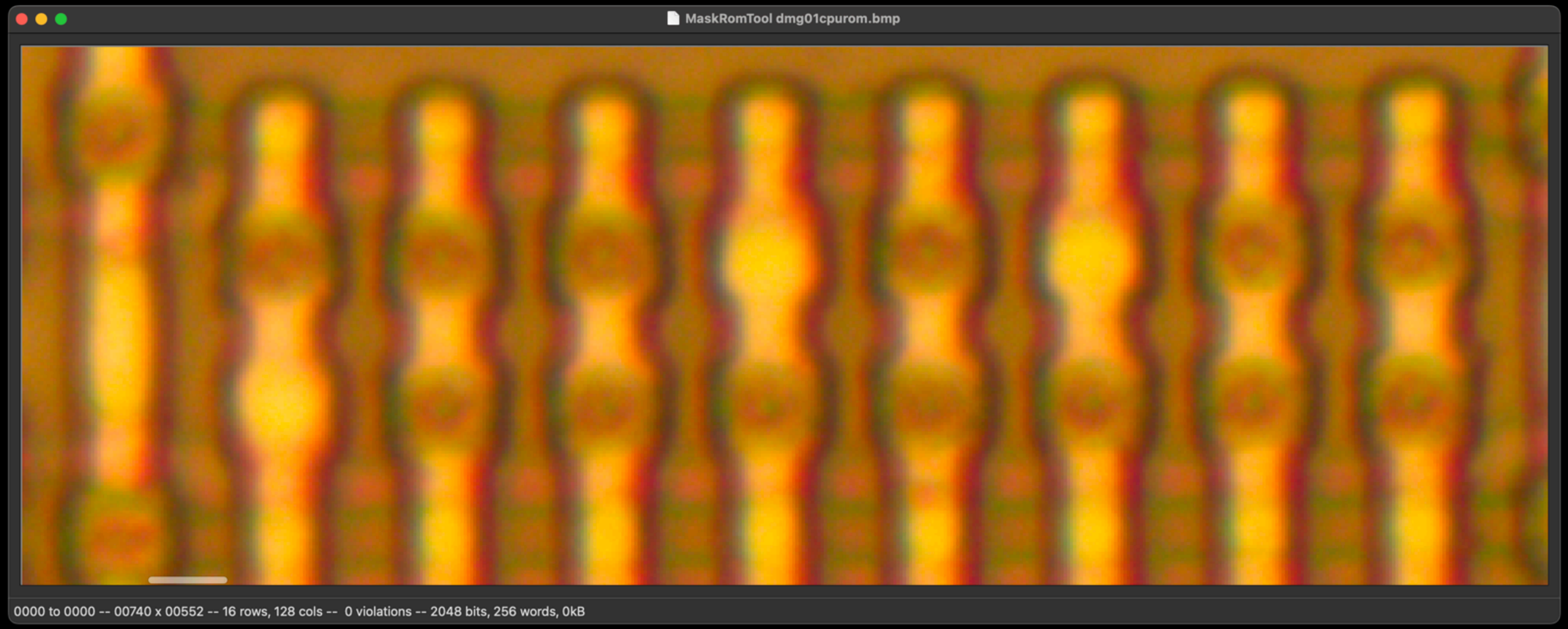
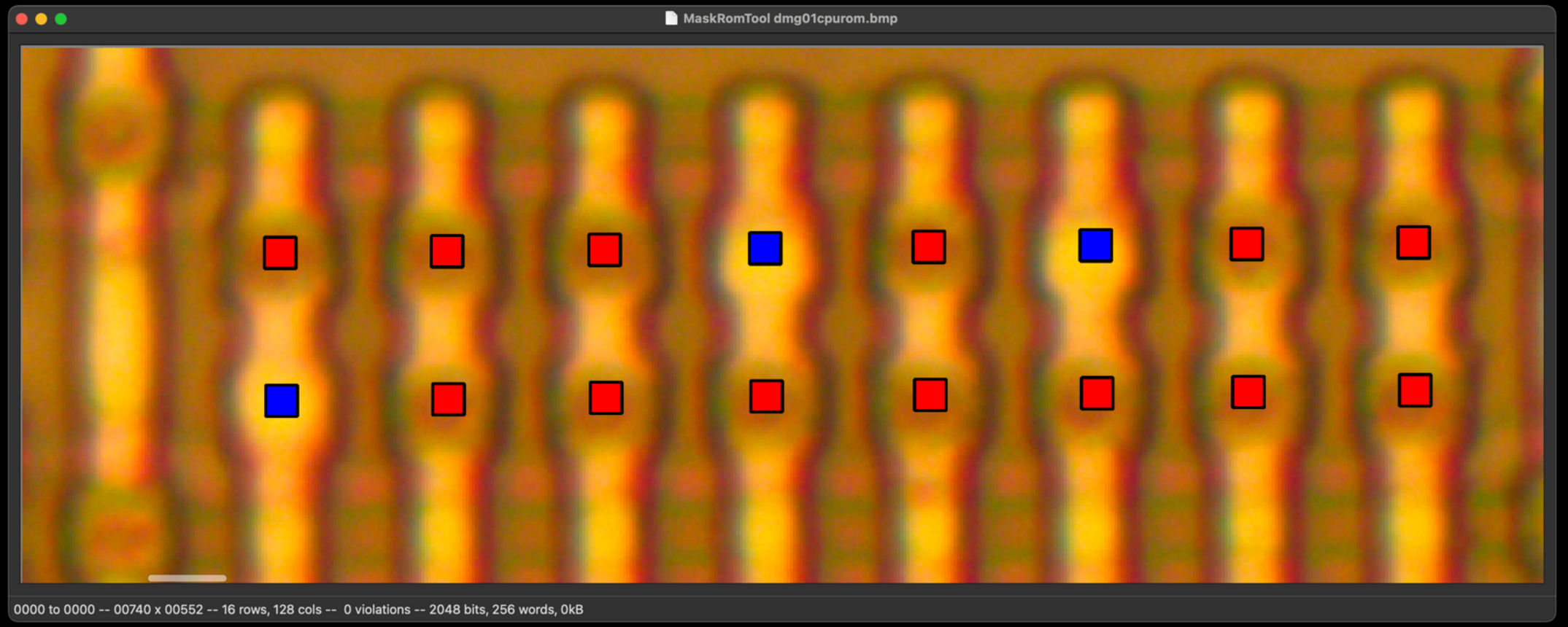
GameBoy ROM Extraction!



R2Con, Barcelona, 8 November 2024

Travis Goodspeed

GameBoy ROM Extraction!



R2Con, Barcelona, 8 November 2024

Travis Goodspeed

GameBoy ROM Extraction

- First, some theory:
  - What's a mask ROM?
  - How can we identify thousands of bits?
  - How can we convert those bits to bytes?
- Second, some practice:
  - Nintendo's Game Boy ROM: 2048 bits.

Software and target!

- https://github.com/travisgoodspeed/maskromtool/
  - Latest release for Windows or macOS.
  - Build from source with Qt6 for Linux.
- https://github.com/travisgoodspeed/gbrom-tutorial
  - Clone this locally, follow instructions in README.

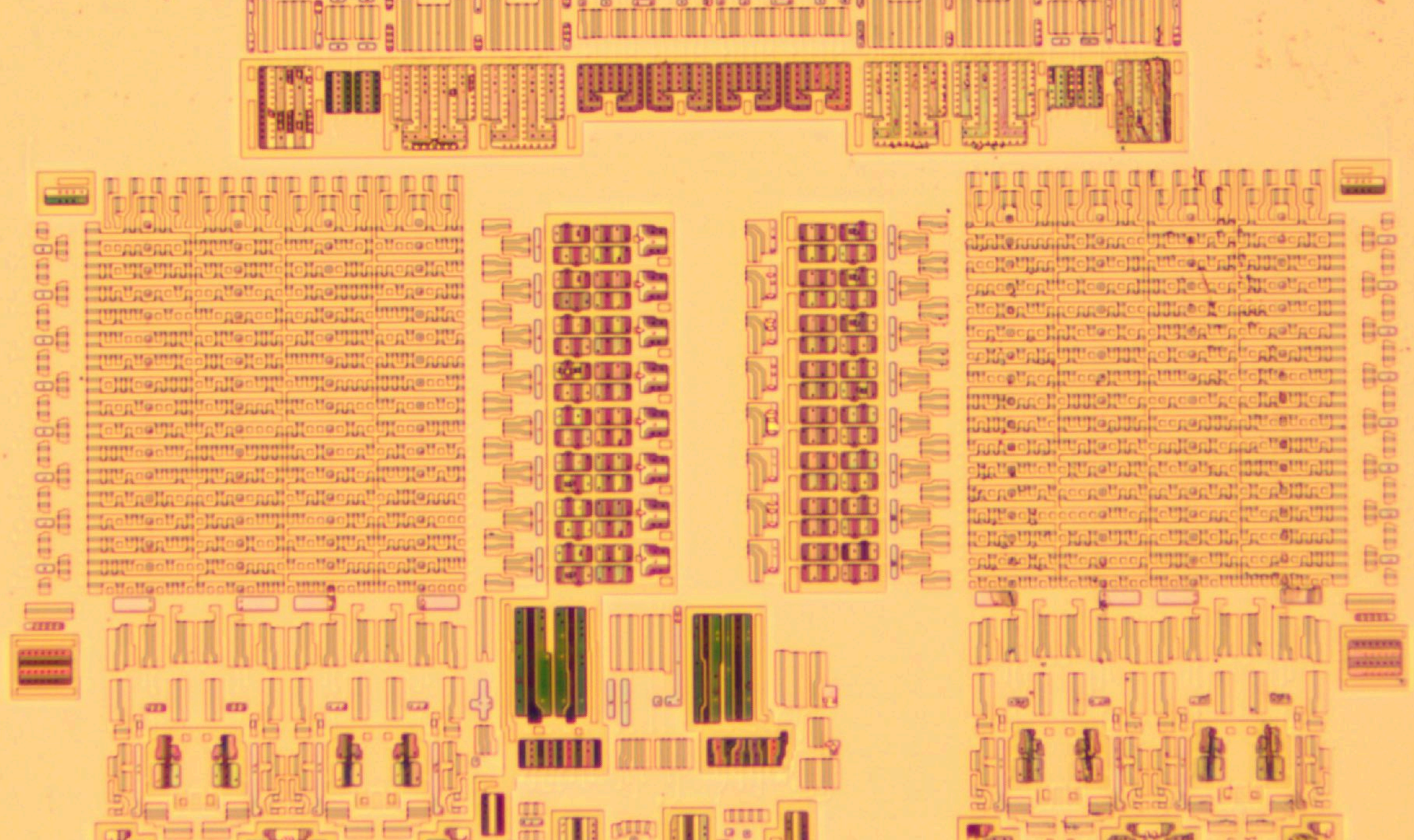
- WIFI: UPCguest

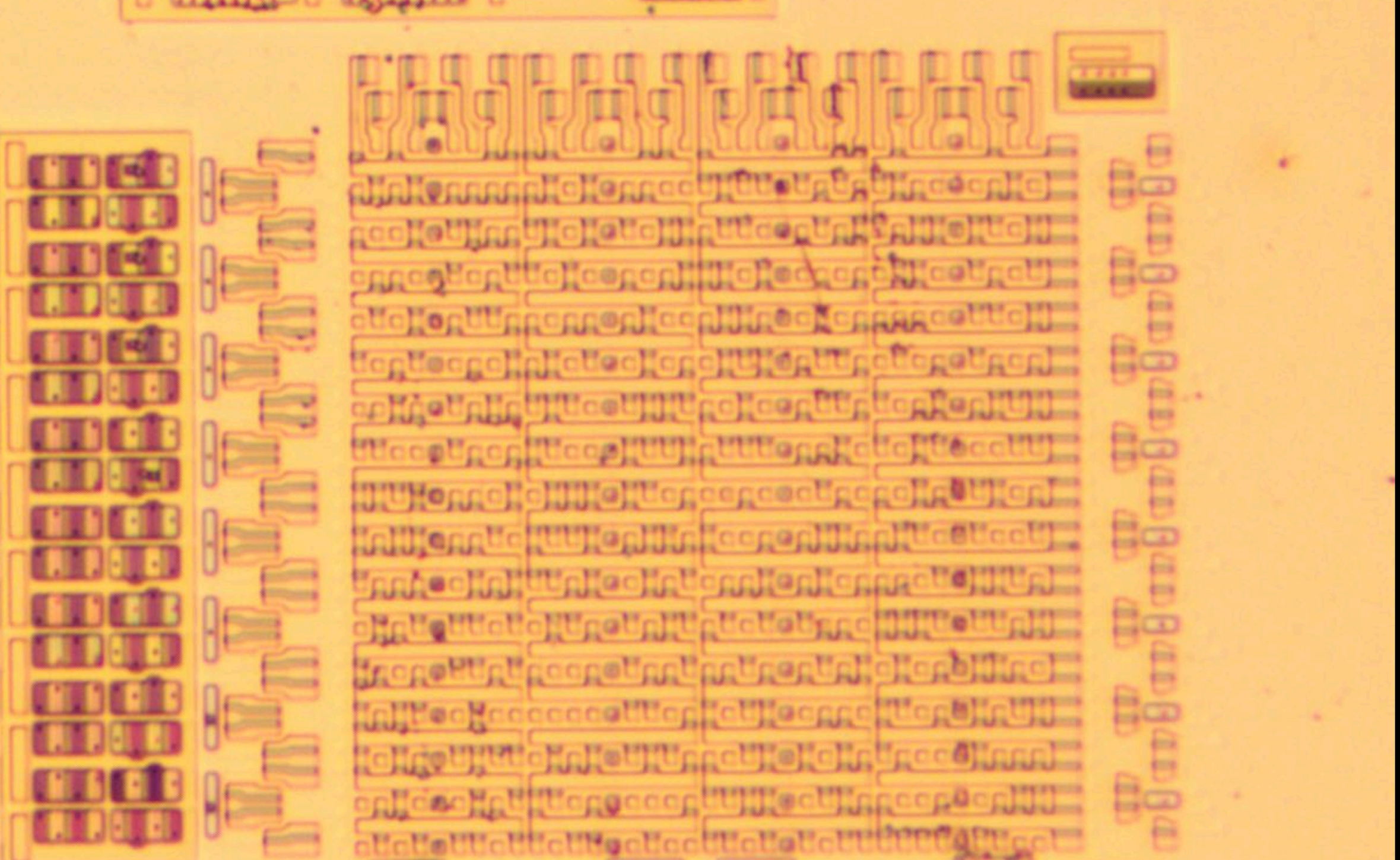
Flash ROM, EEPROM, and Mask ROM

- Flash ROM and EEPROM
  - Electrically, Individually Programmable
  - Great for small quantities of code.
- Mask ROM
  - Mask Programmed
  - Only for very large quantites.

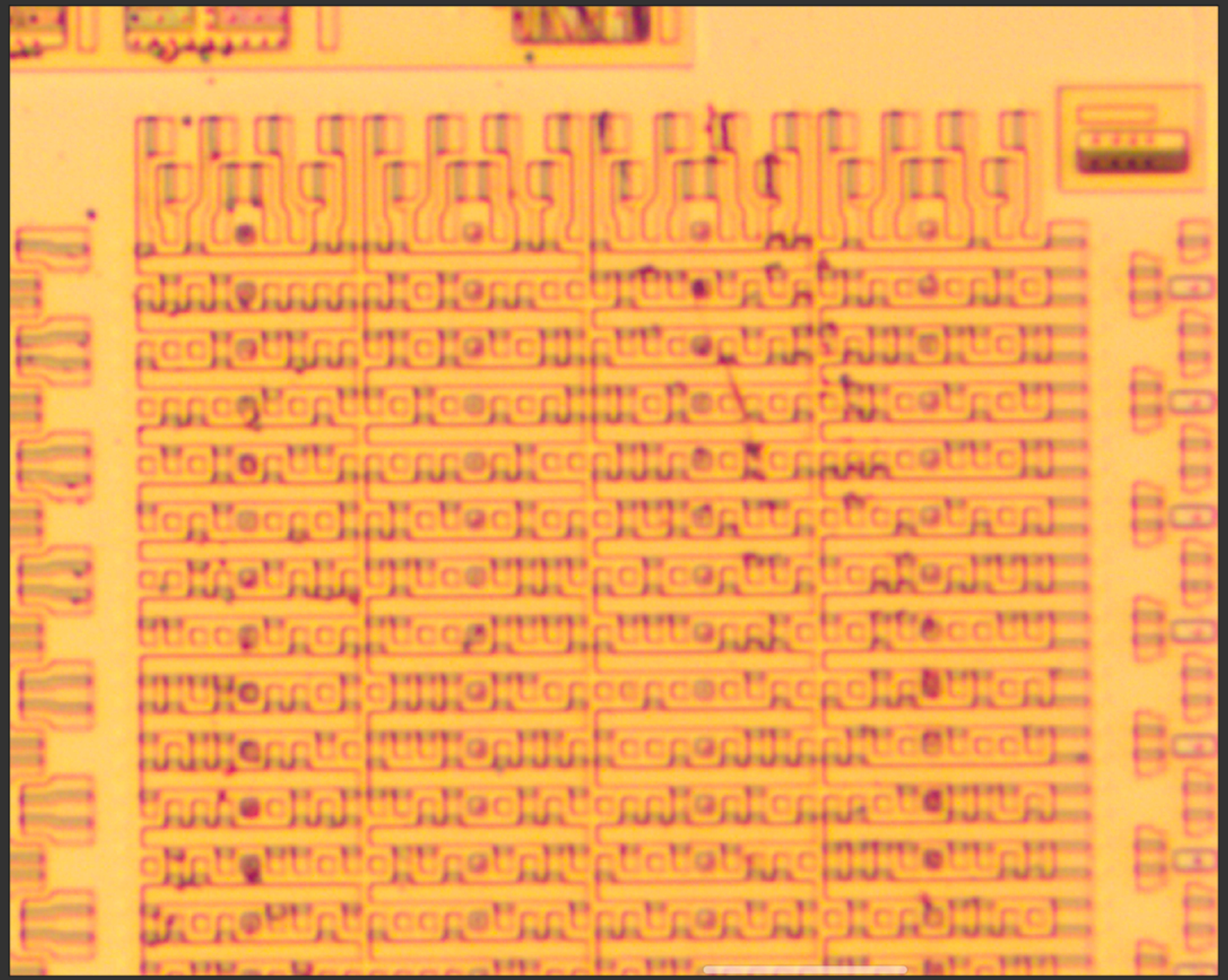
Mask ROMS

- Mask Programmed at the Factory
- Contain code, data, or microcode.
- Often require chemical processing.
- Good targets for reverse engineering:
  - Video Games, Copy Protection
  - Cryptography

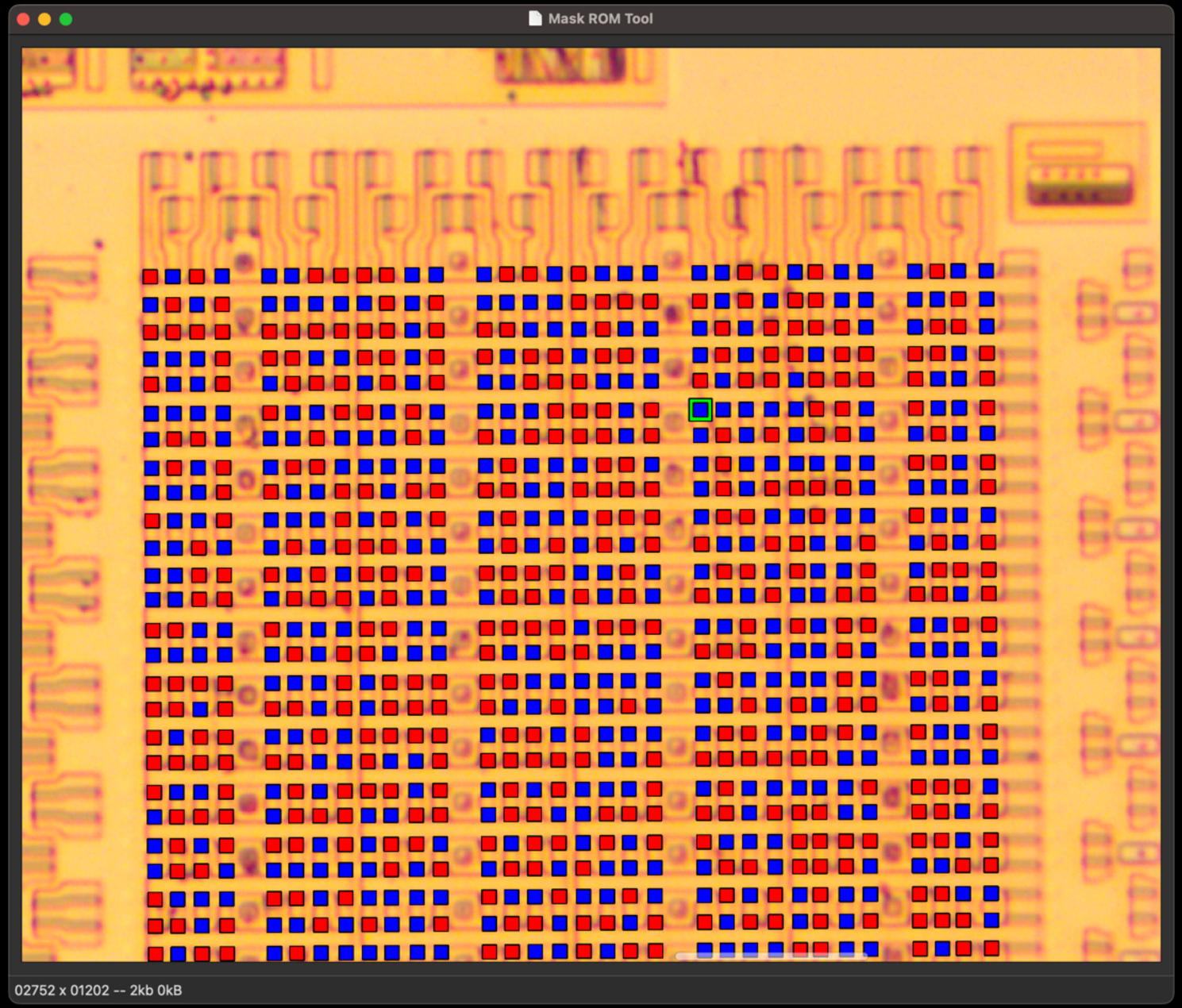




Mask ROM Tool

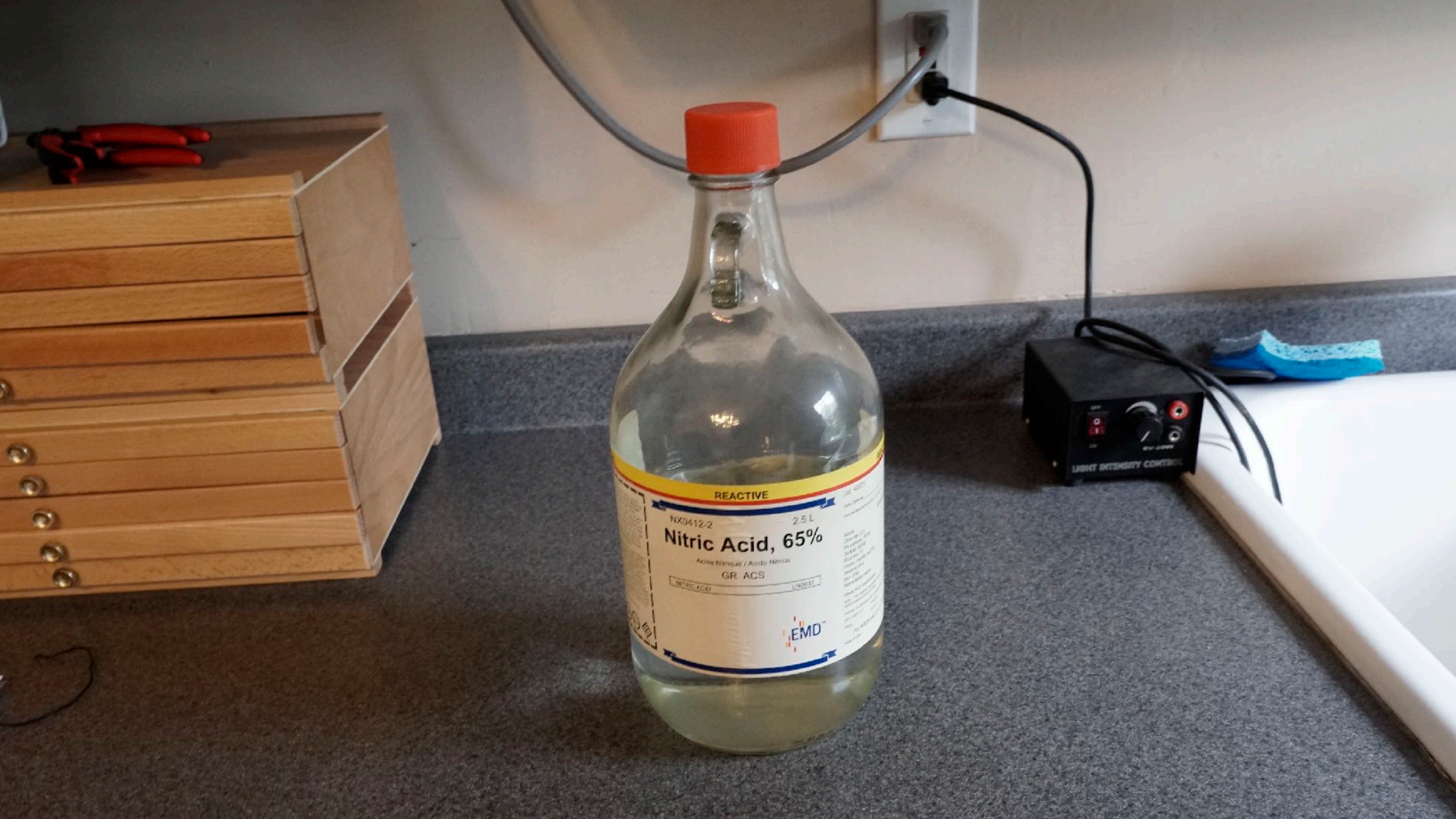


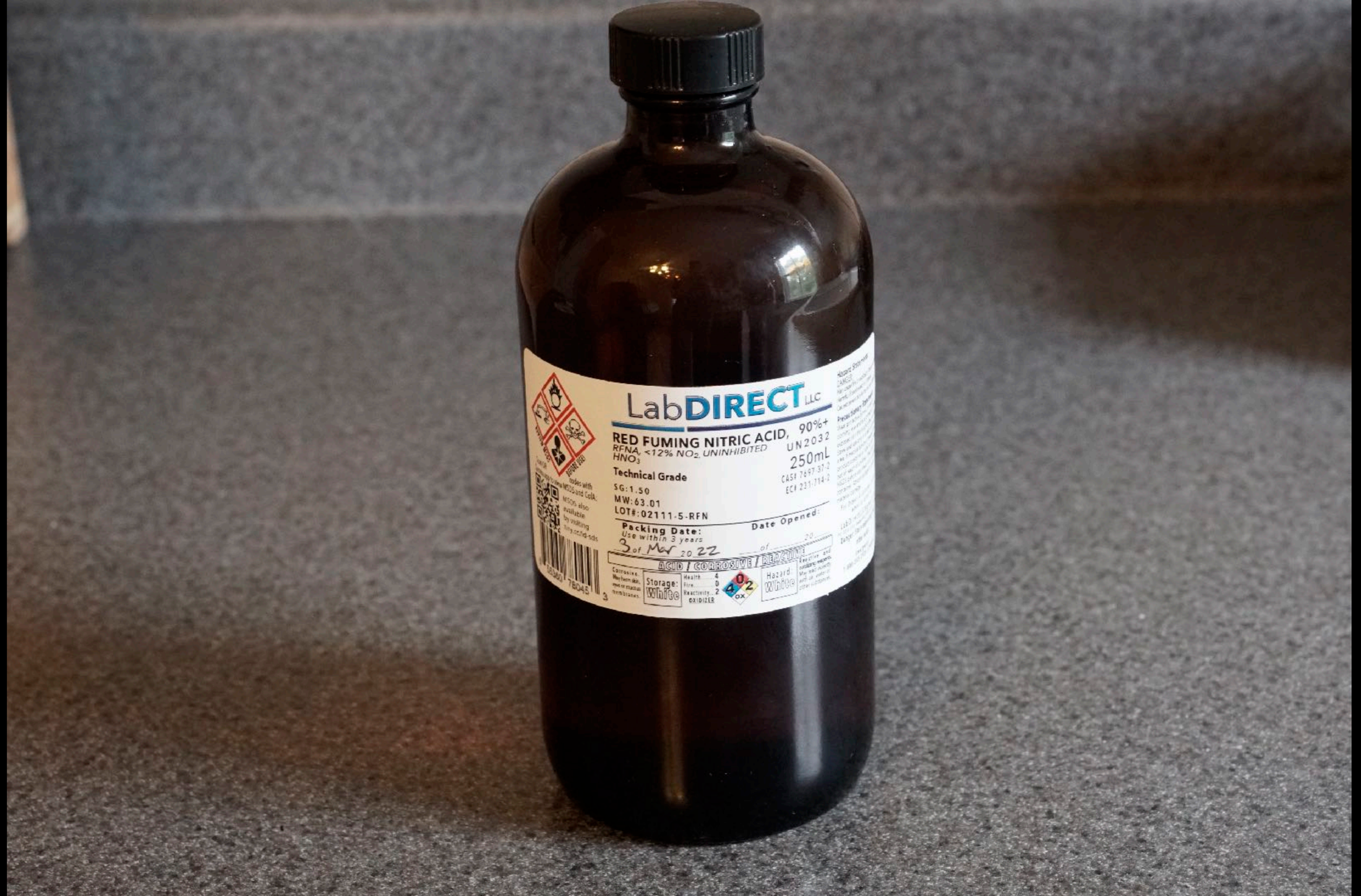
Deleted item.


Mask ROM Tool
02752 x 01202 -- 2kb 0kB

A Sidebar into Chemistry

- Depackage the chip with 65% HNO_3 and heat.
- Delayer the chip with HF.
- Stain the ROM with a Dash Etch.


REACTIVE
NX0412-2
2.5 L
Nitric Acid, 65%
Acide Nitrique / Ácido Nítrico
GR ACS
EMD


LabDIRECT LLC
RED FUMING NITRIC ACID, 90%+
RFNA, <12% NO_2, UNINHIBITED
HNO_3
UN2032
250mL
Technical Grade
CAS# 7697-37-2
EC# 231-714-2
SG:1.50
MW:63.01
LOT#:02111-5-RFN
Packing Date:
Use within 3 years
3 of Mar 20 22
Date Opened:
ACID / CORROSIVE / REACTIVE
Storage: White
Hazard White
OXIDIZER


CH582M
1091DC06
CH582M
1091DC06

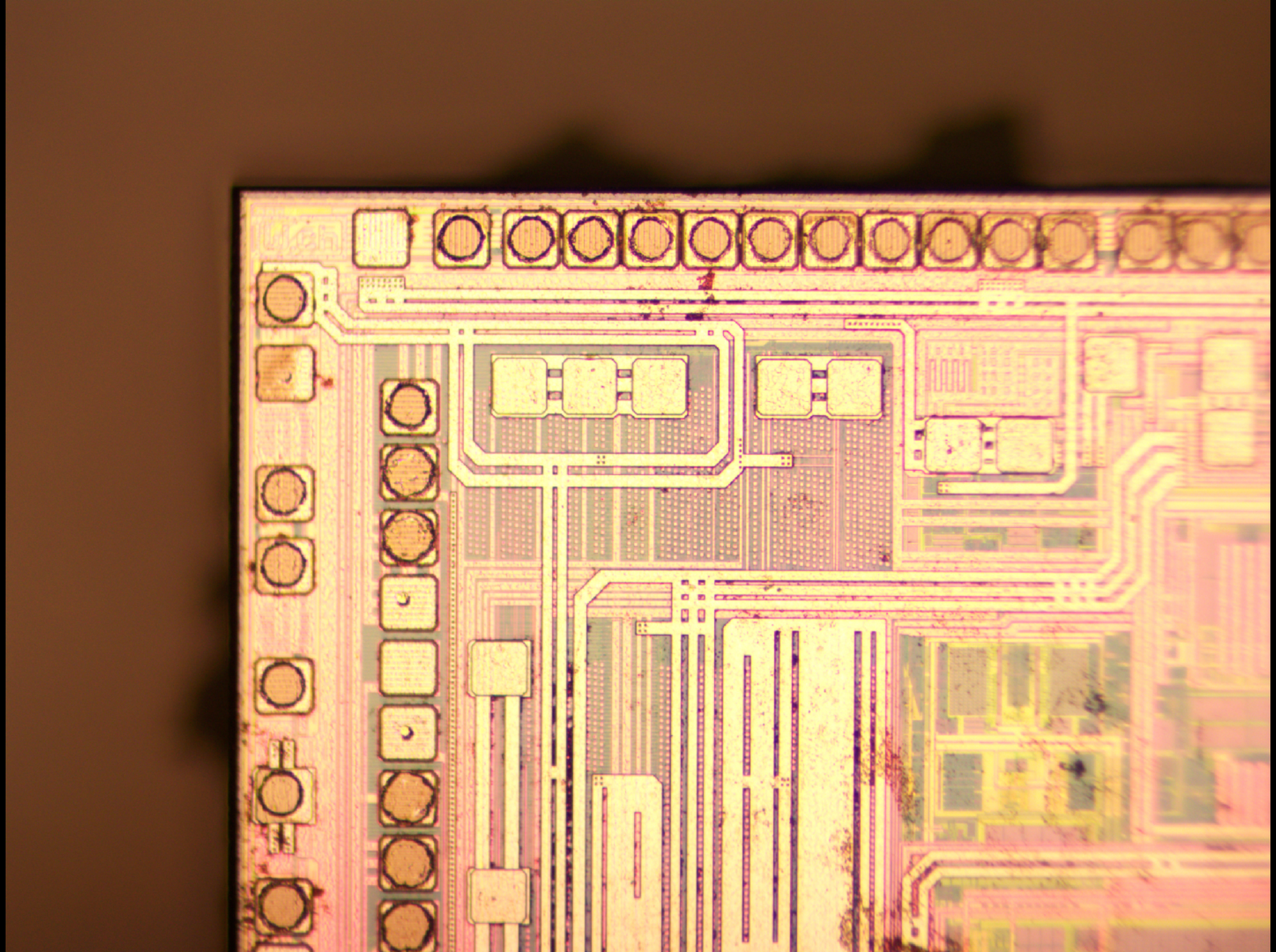

VWR
STIR
100 mL
PYREX
25 mL
± 5%

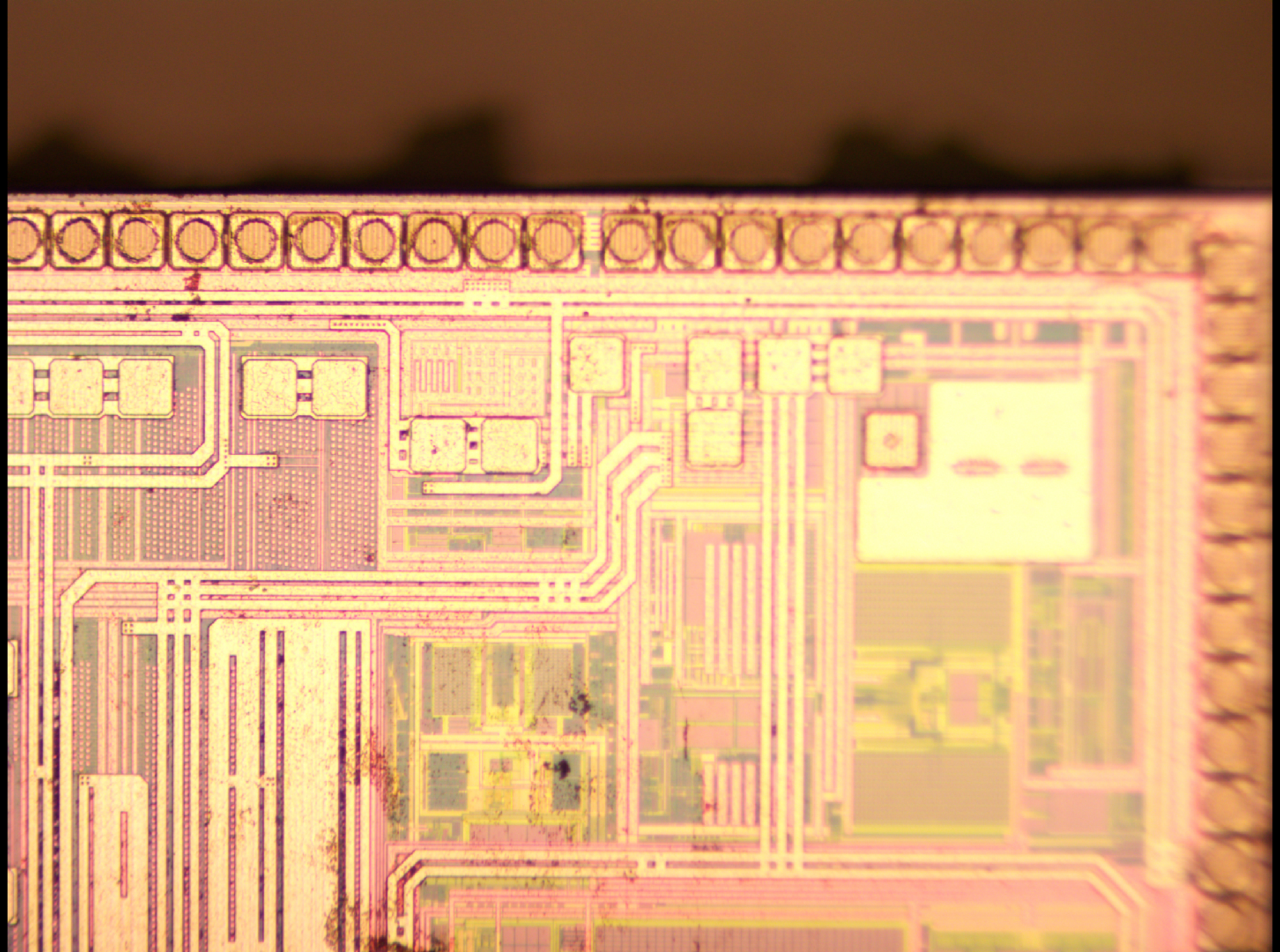

VWR
MODEL 320
STIR

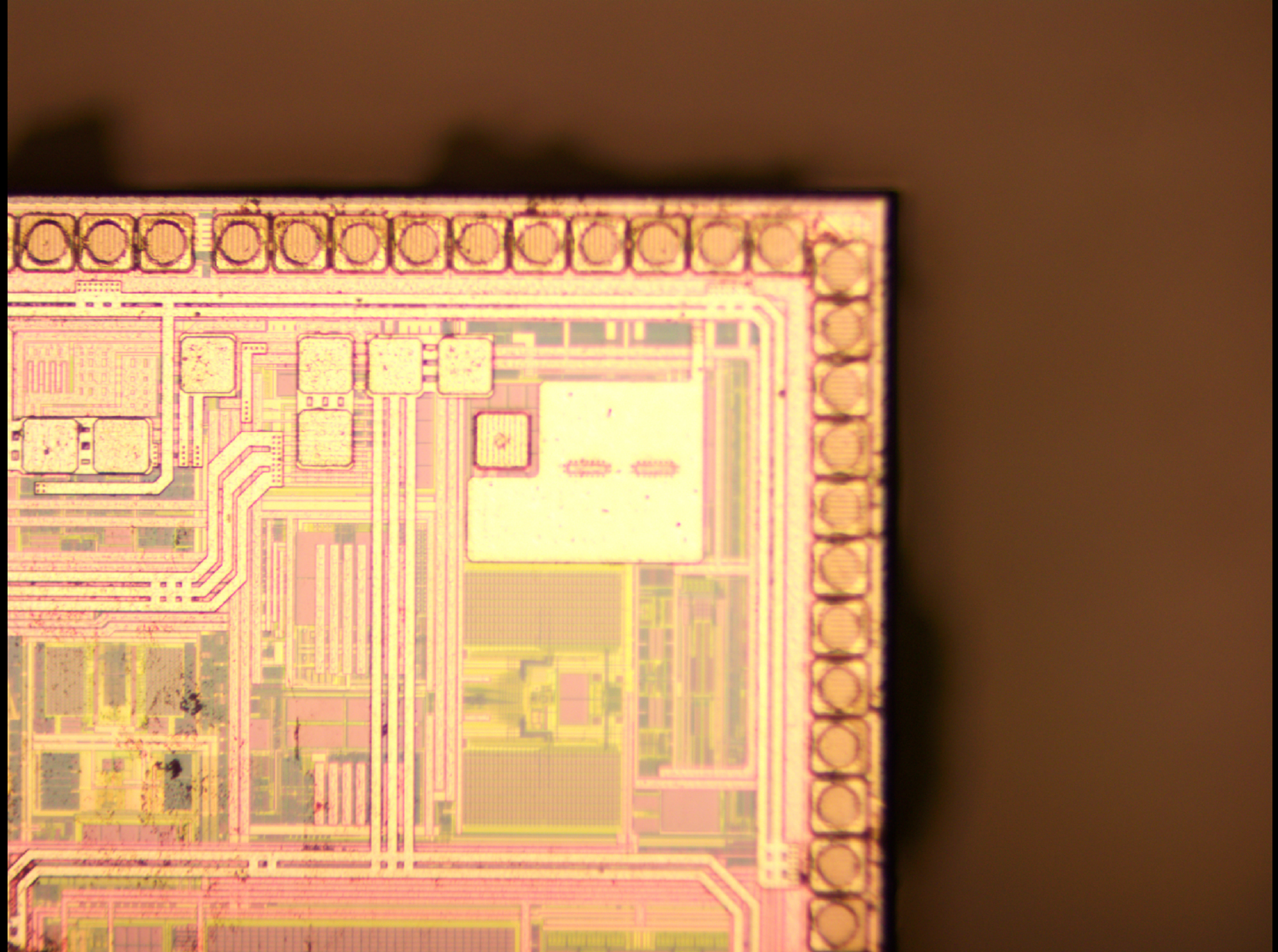

25 mL
± 5%
15
5

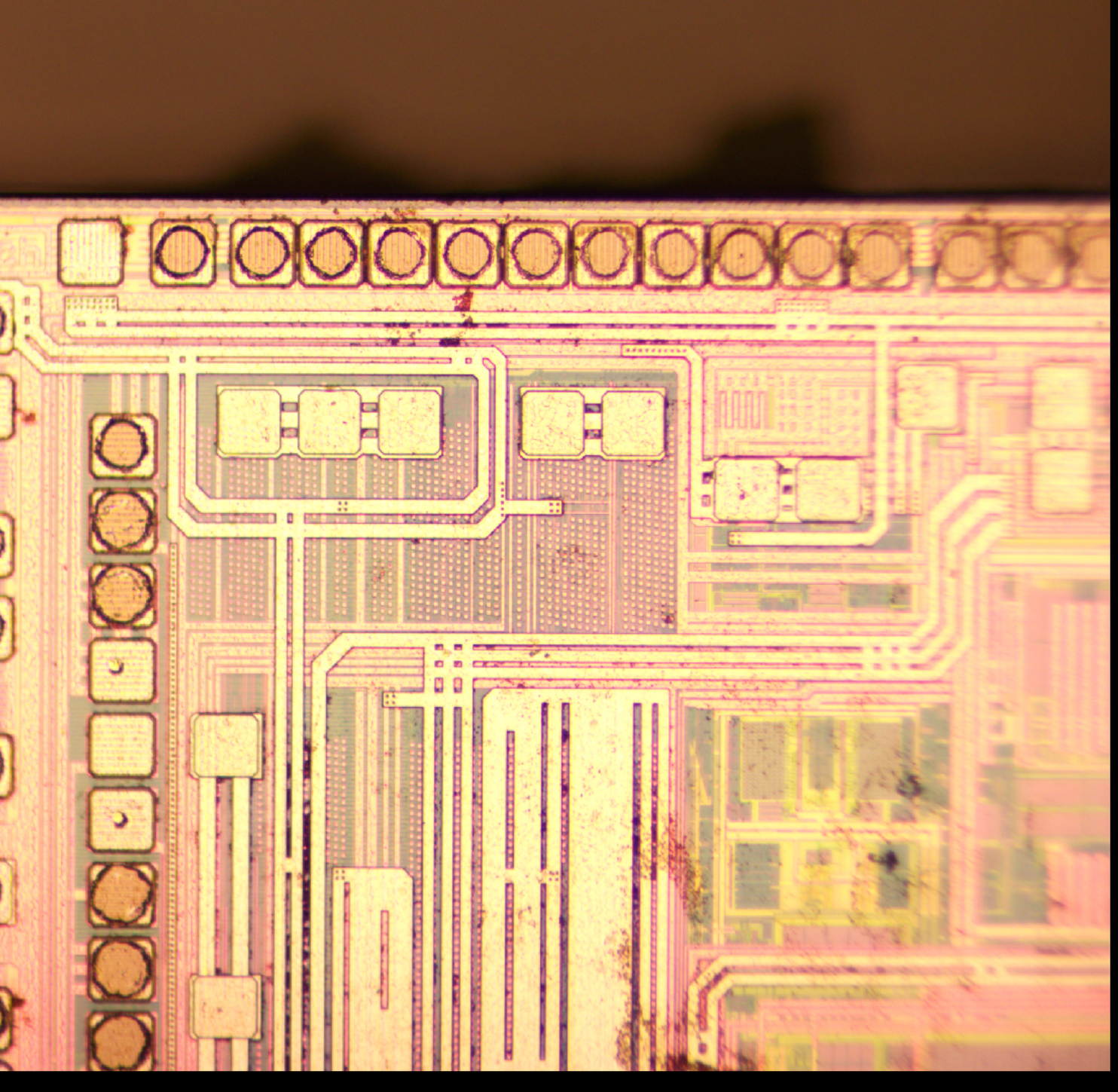

LMPlan
5/0.13
160/0 WD19
I O II

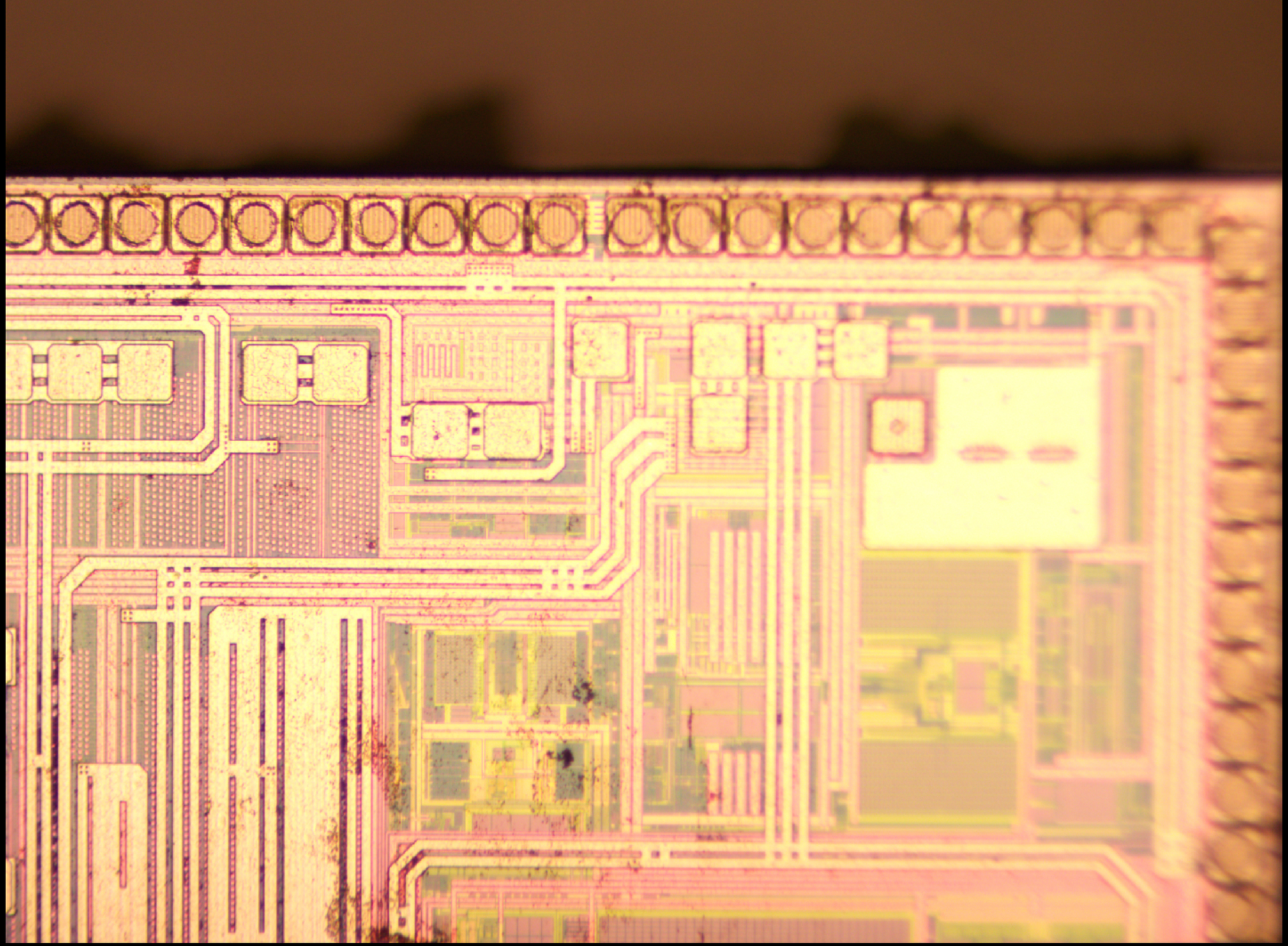

LG

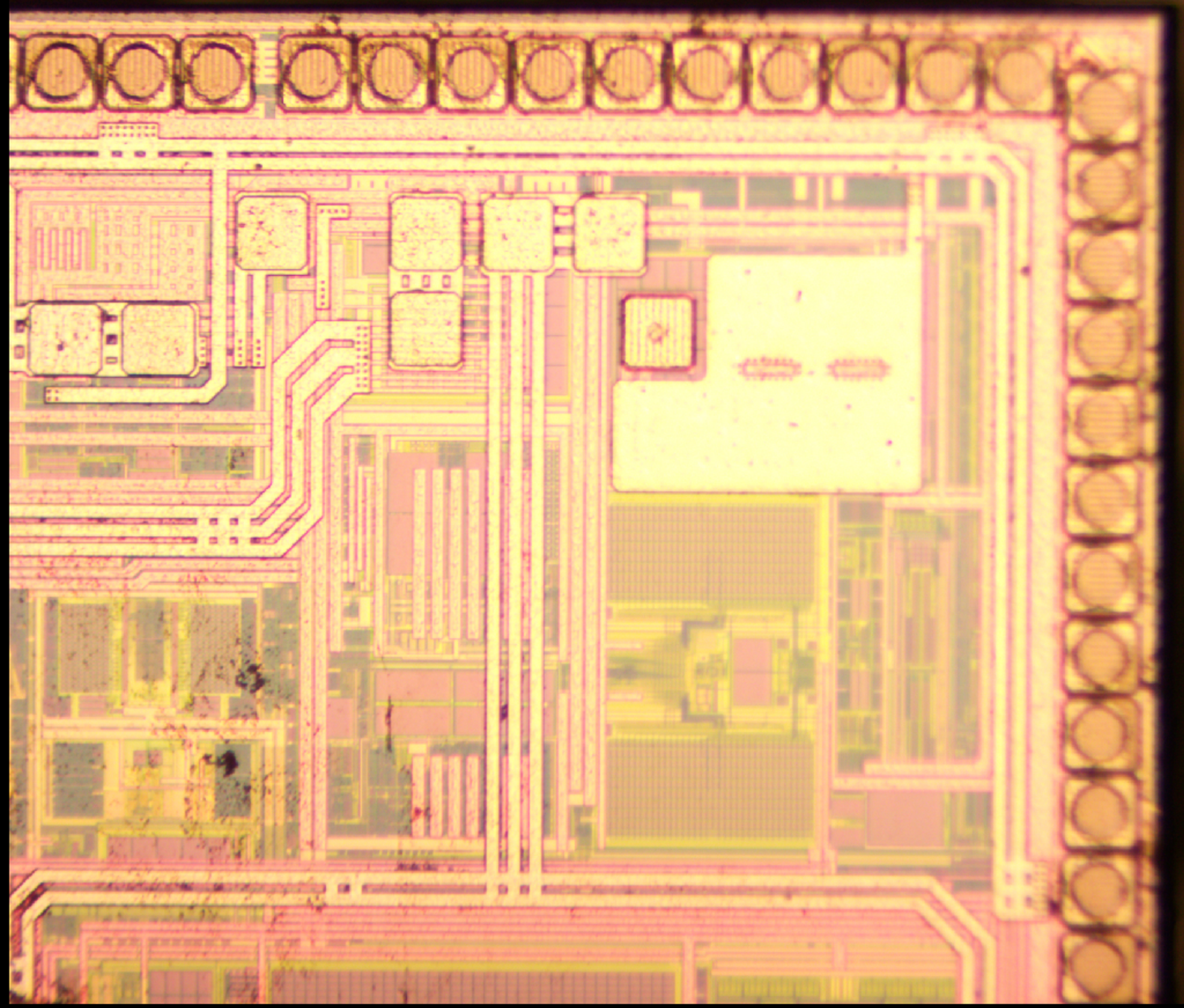


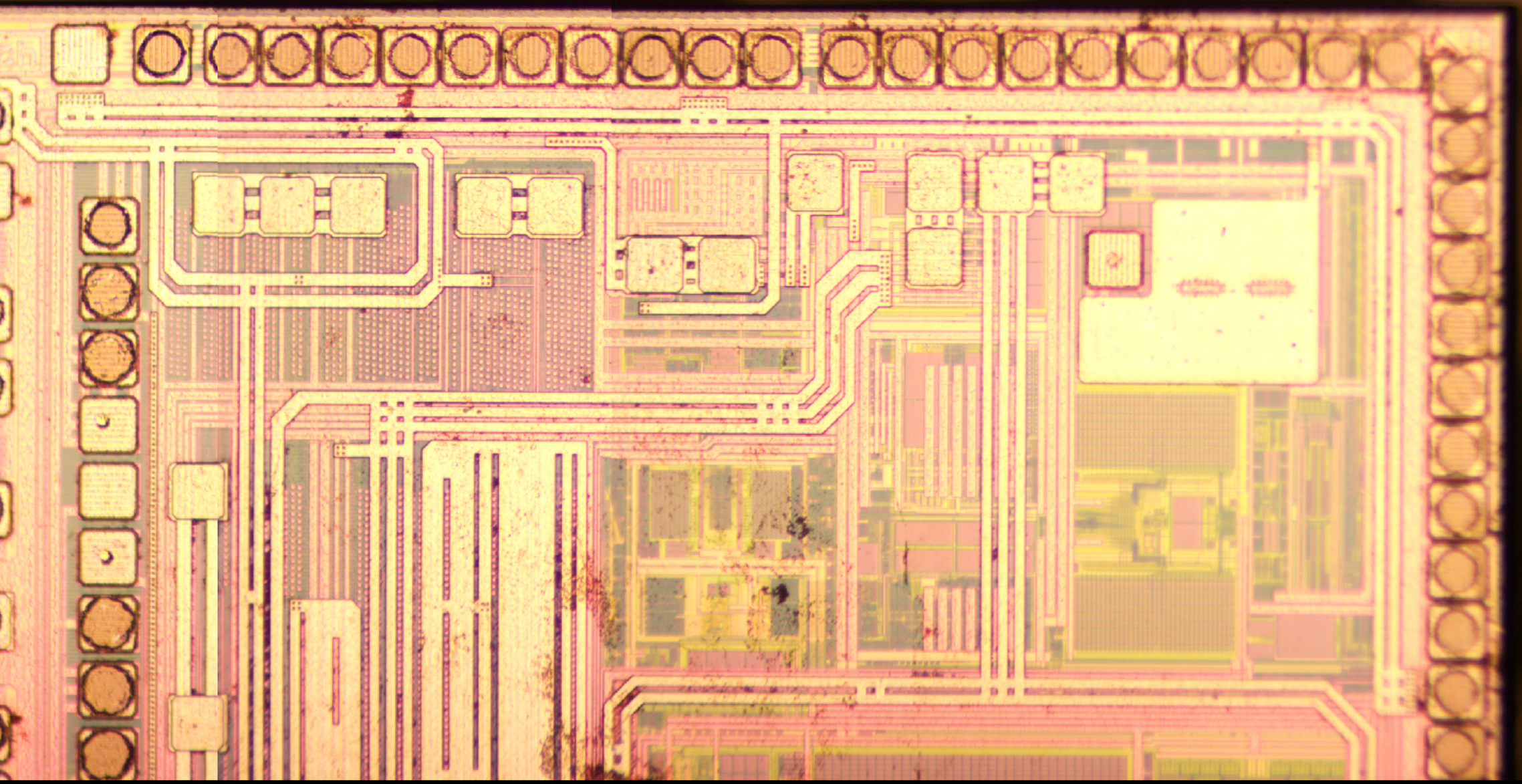


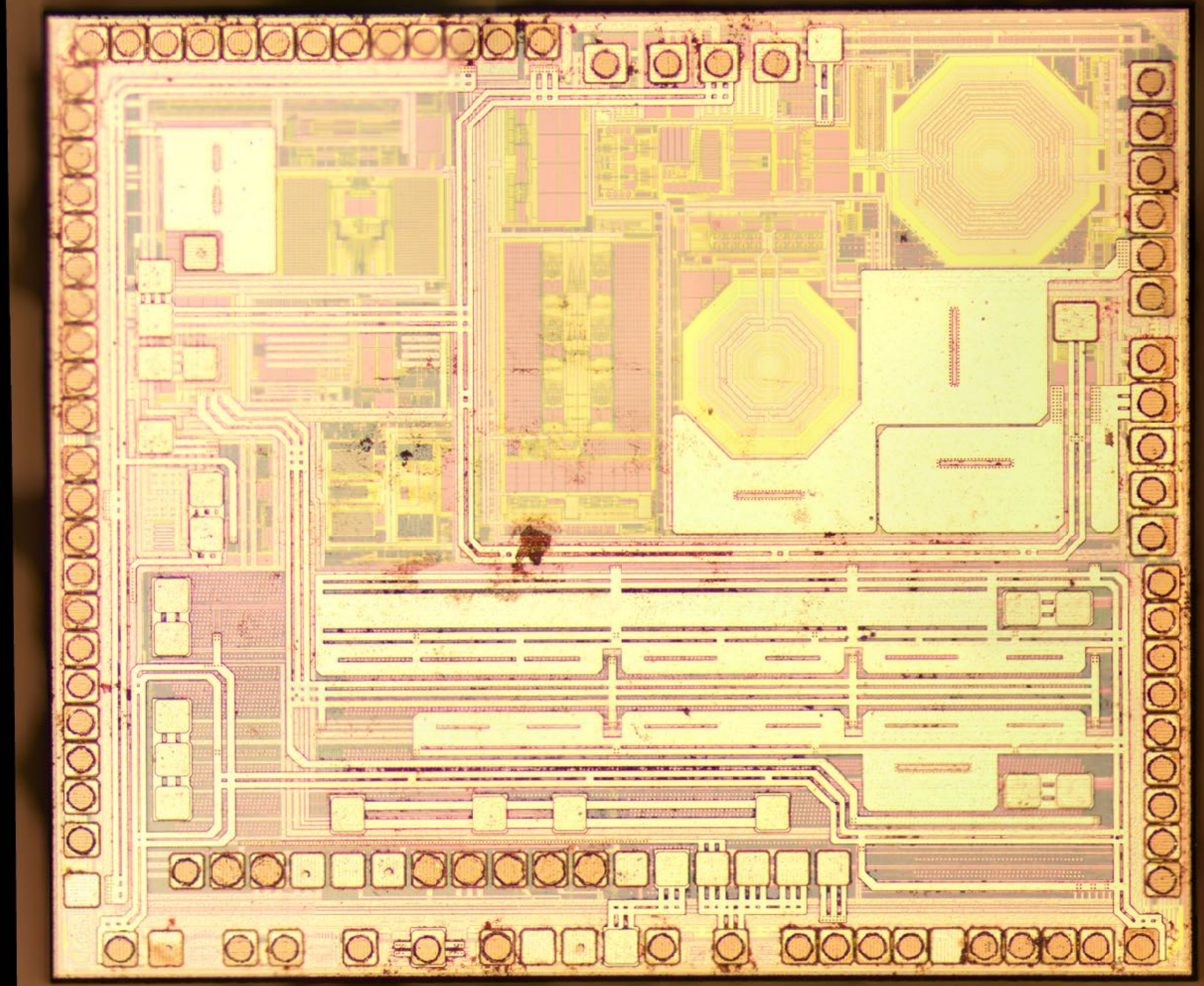










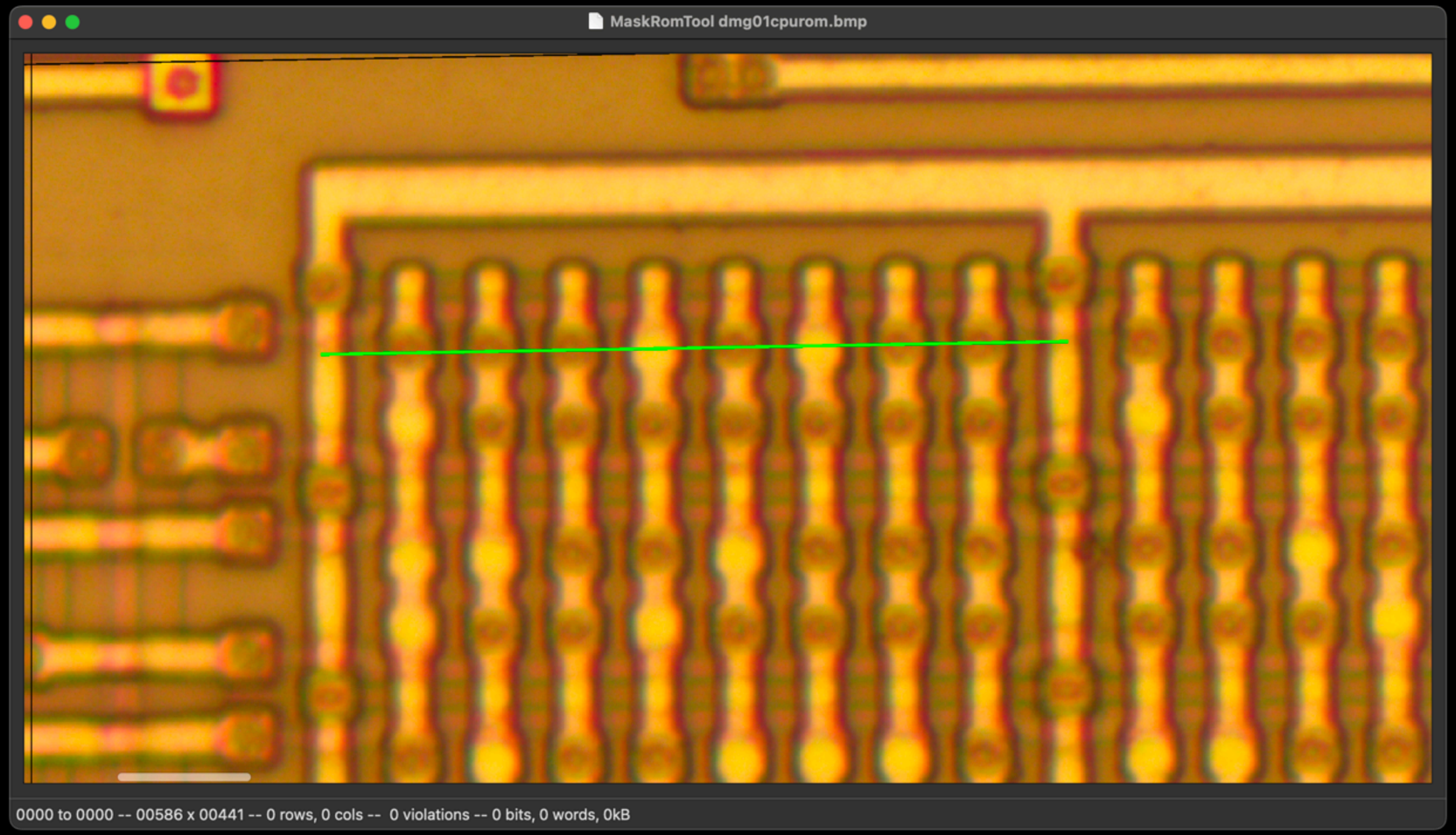


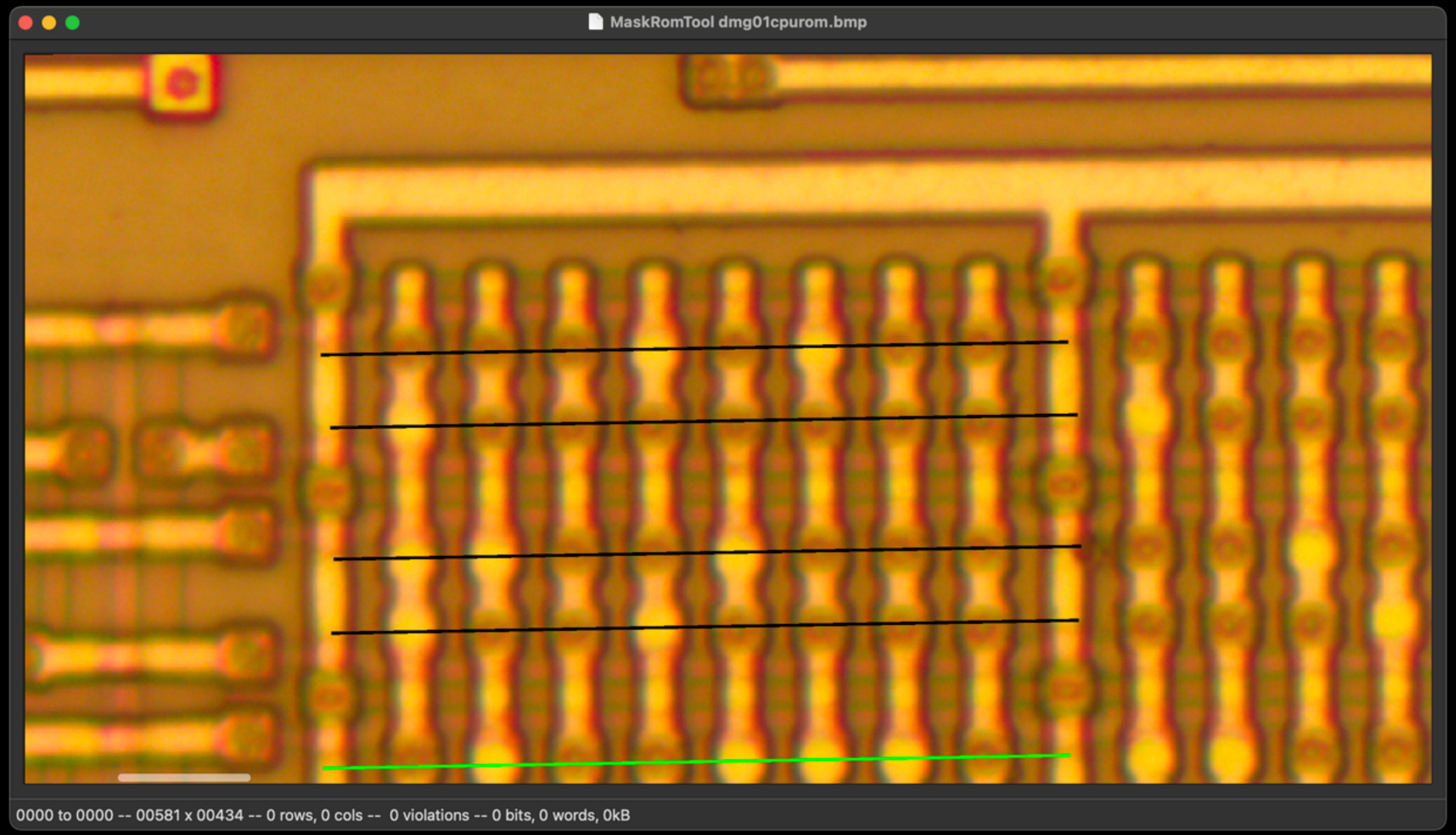
Delayering

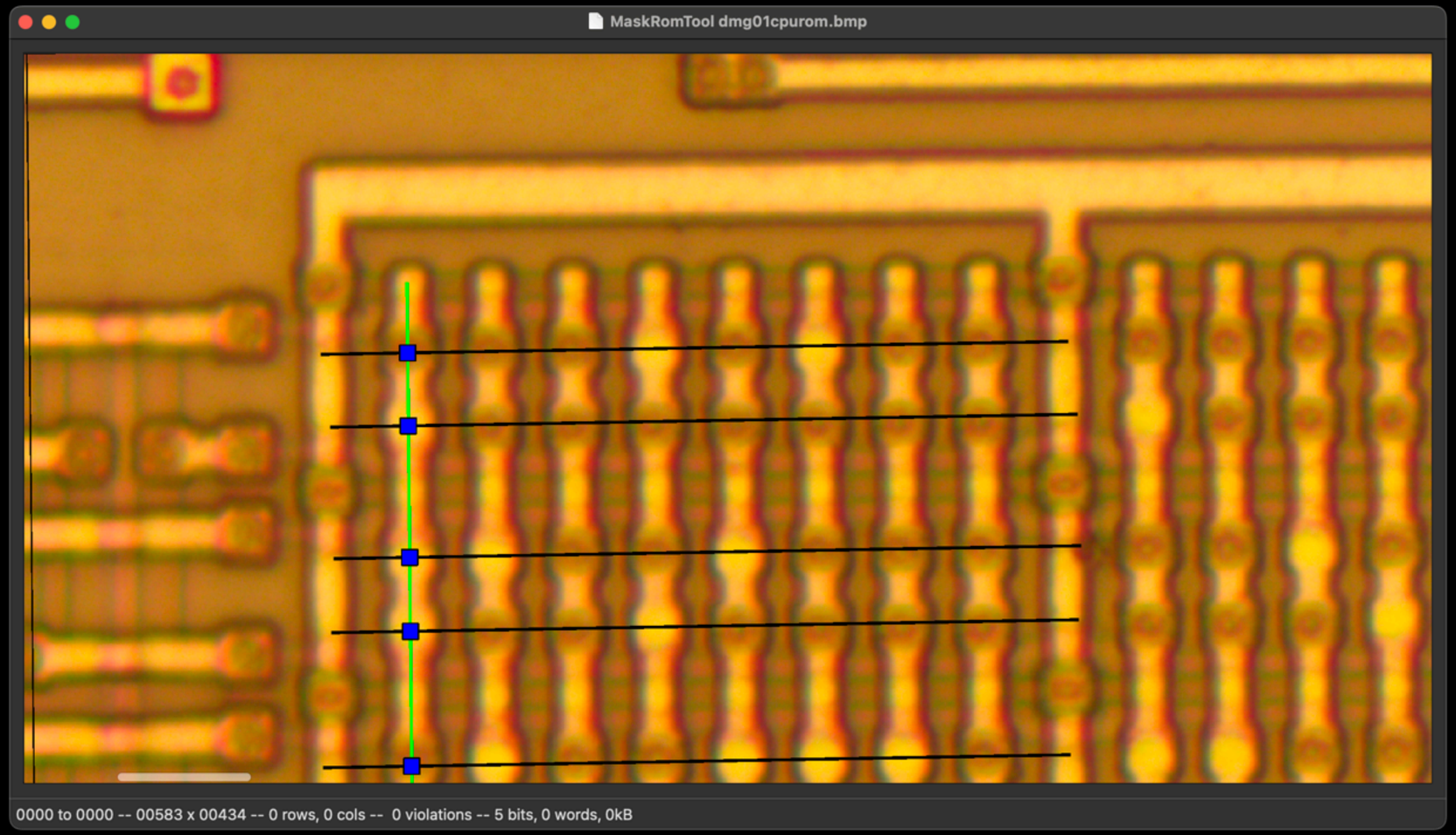
- HNO3 or H2SO4 gets us to the top metal layer.
- Many ROMs are lower in the chip:
  - Diffusion ROMs require HF delayering.
  - Implant ROMs require Dash Etch staining.

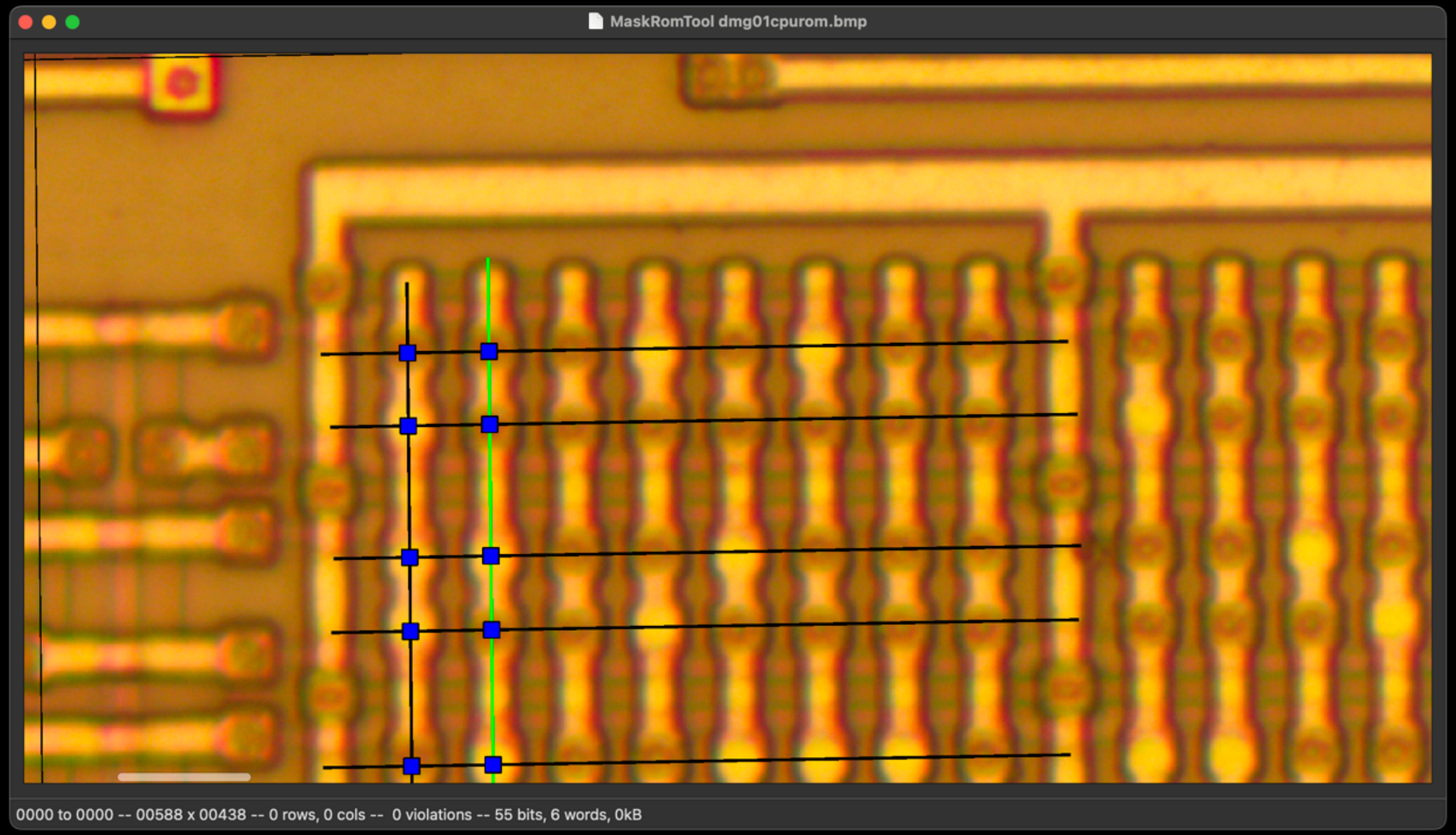
A Quick Demo of Extraction

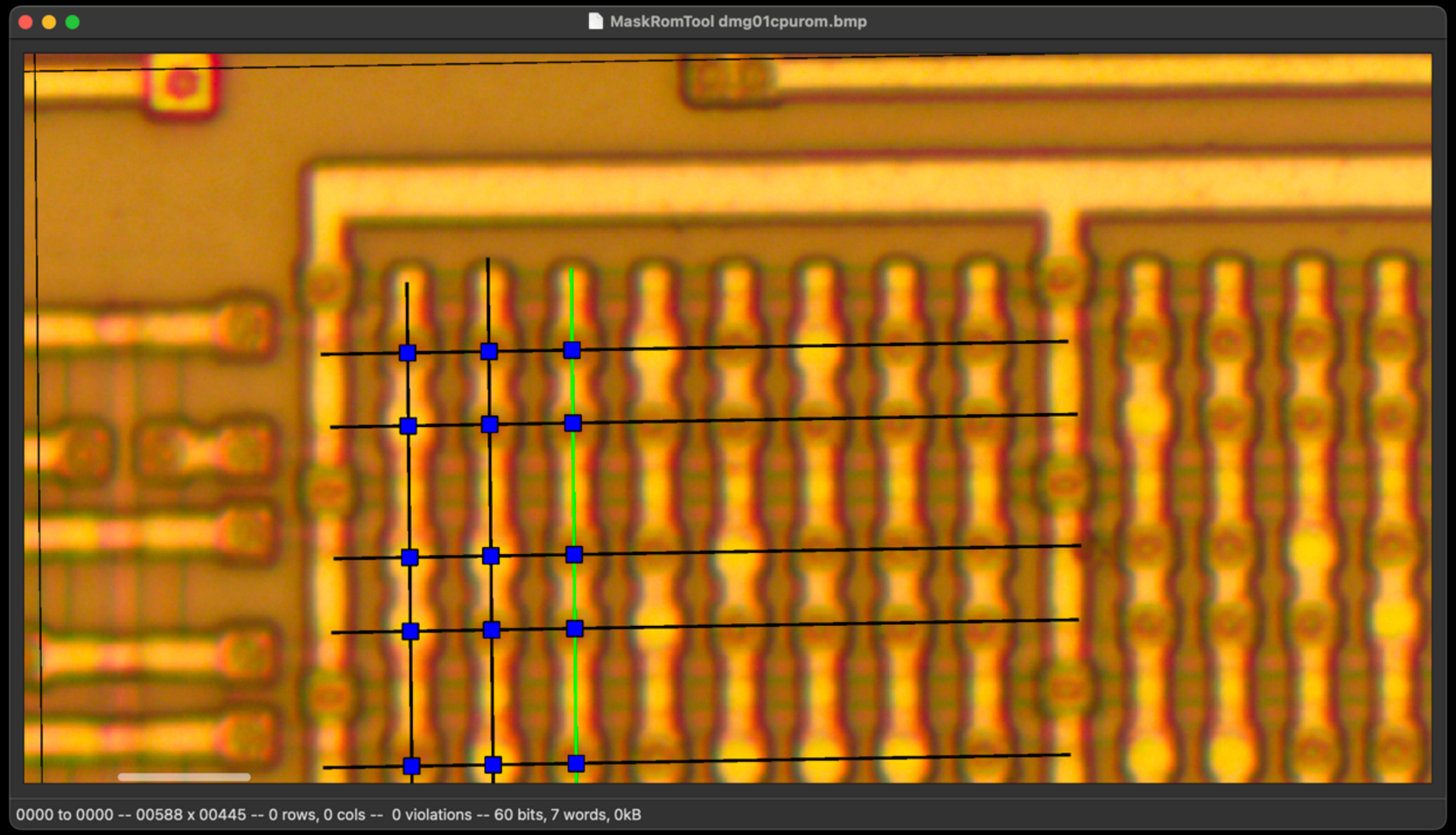
- We annotate some features:
  - Row and Column lines.
  - Threshold between One and Zero.
  - Forced bits.
- Tool helps out:
  - Bits marked at Row/Column intersections.
  - Design Rule Checks (DRC) reduce bit errors.
  - Importing diffs, Exporting to other formats.

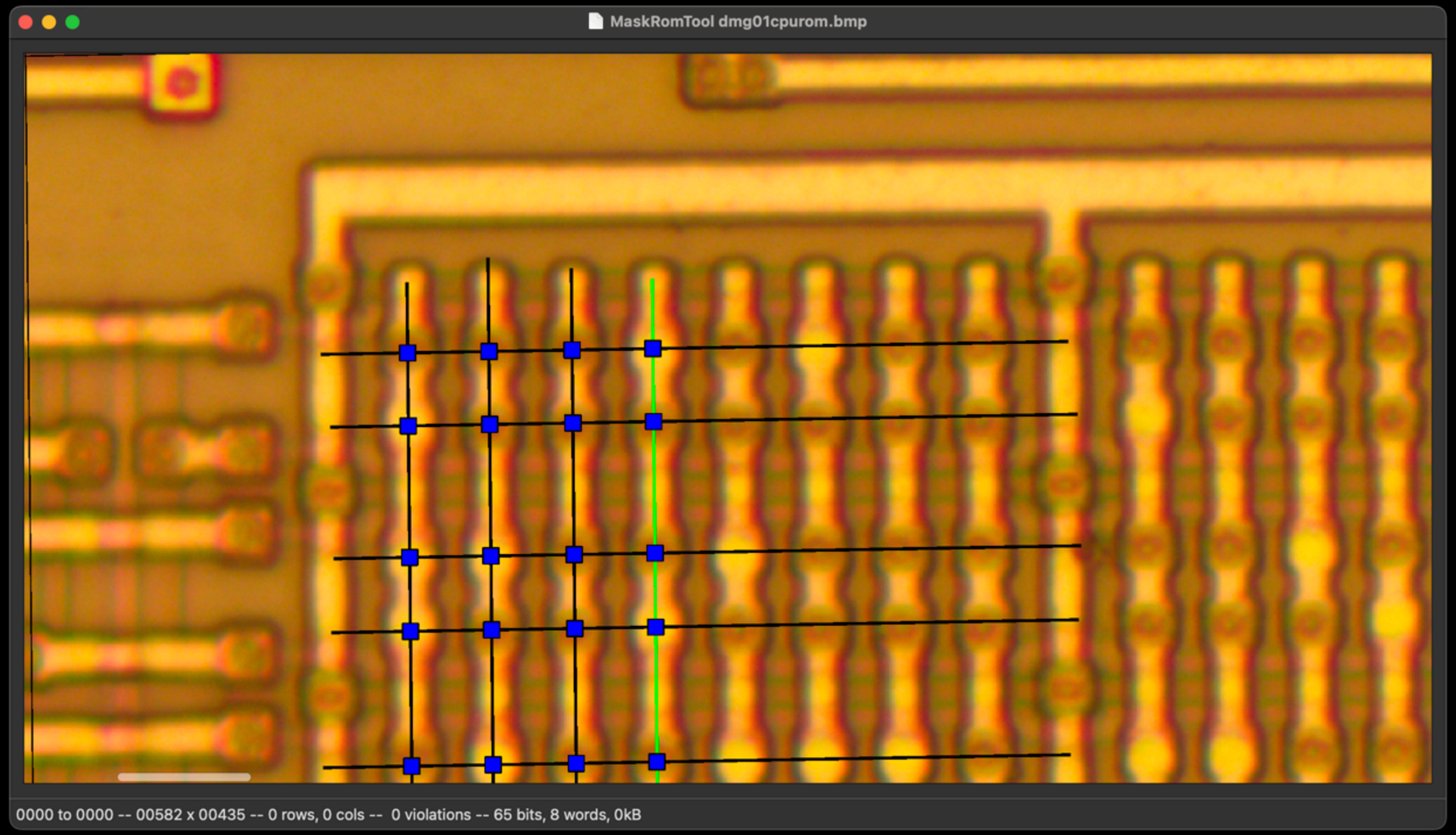

MaskRomTool dmg01cpurom.bmp
0000 to 0000 -- 00586 x 00441 -- 0 rows, 0 cols -- 0 violations -- 0 bits, 0 words, 0kB

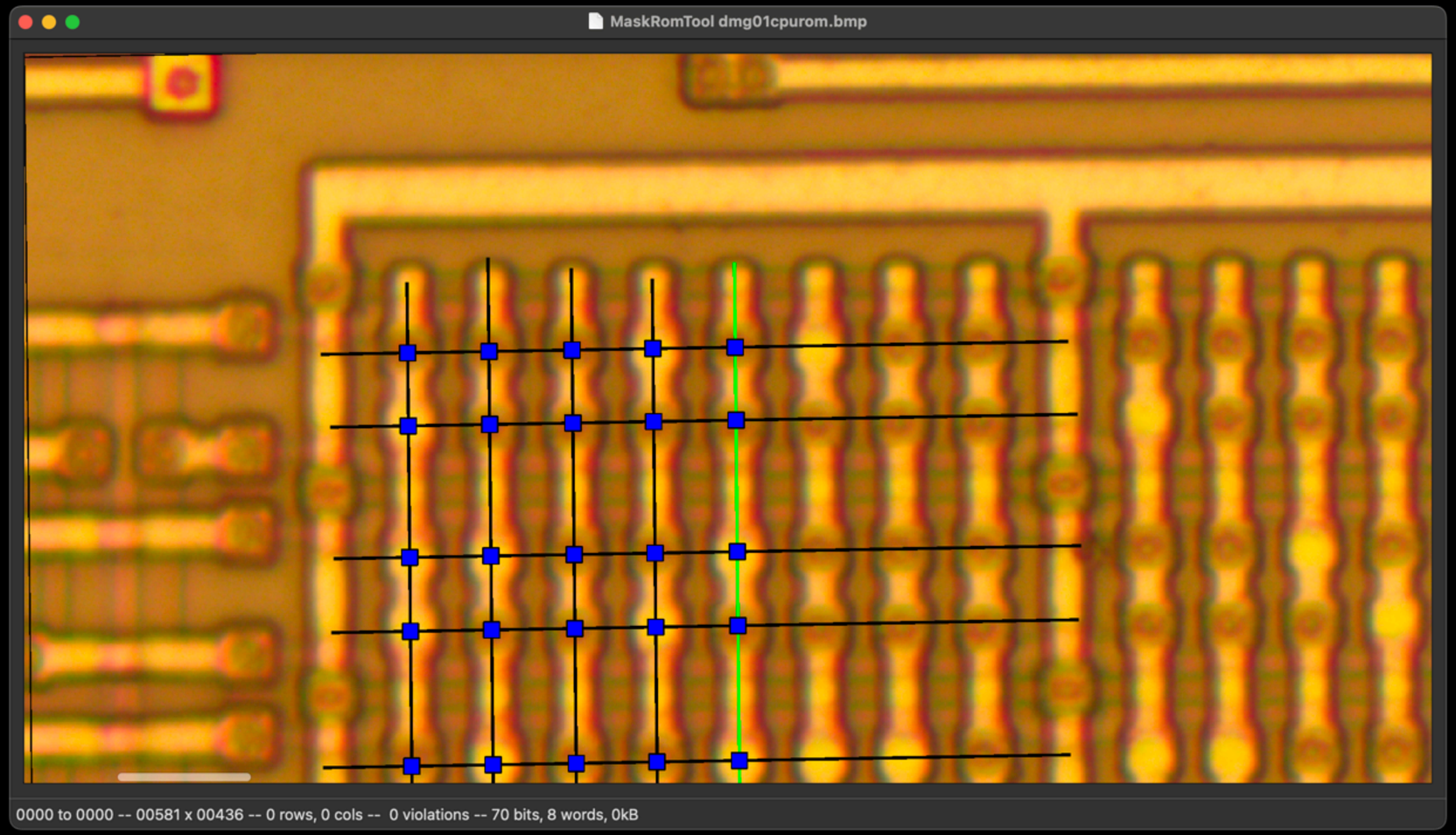

MaskRomTool dmg01cpurom.bmp
0000 to 0000 -- 00581 x 00434 -- 0 rows, 0 cols -- 0 violations -- 0 bits, 0 words, 0kB

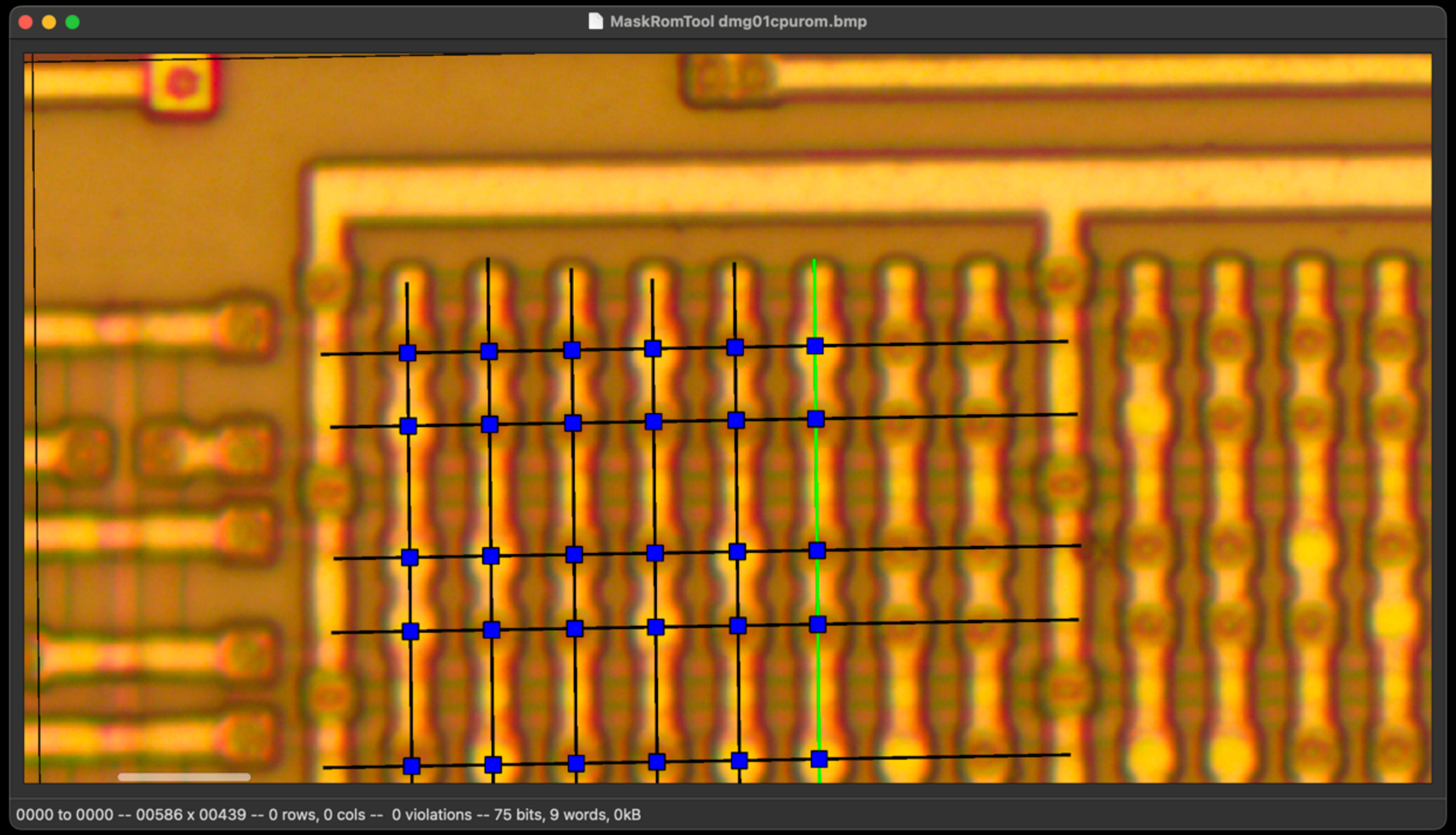

MaskRomTool dmg01cpurom.bmp
0000 to 0000 -- 00583 x 00434 -- 0 rows, 0 cols -- 0 violations -- 5 bits, 0 words, 0kB

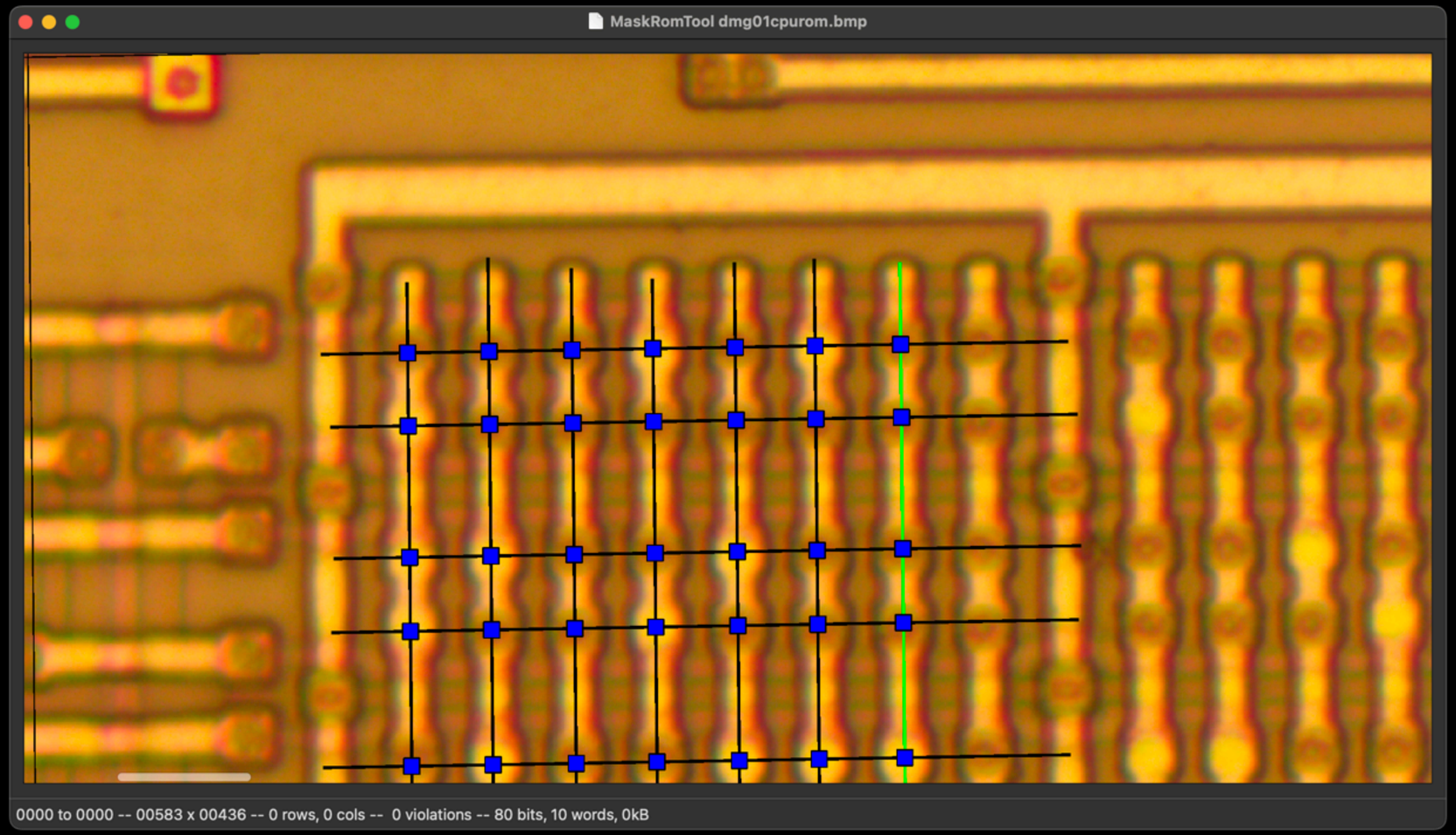

MaskRomTool dmg01cpurom.bmp
0000 to 0000 -- 00588 x 00438 -- 0 rows, 0 cols -- 0 violations -- 55 bits, 6 words, 0kB

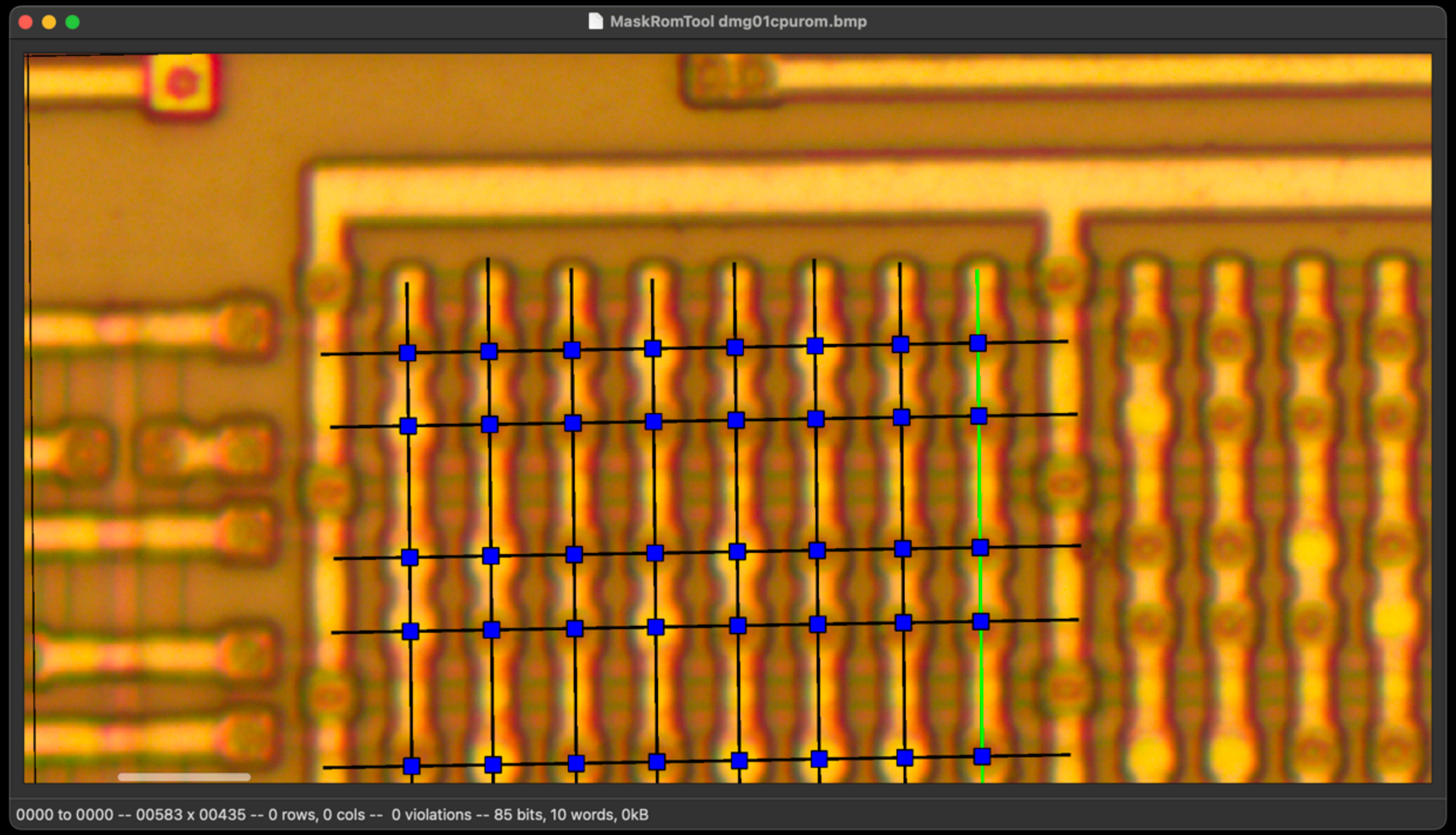

MaskRomTool dmg01cpurom.bmp
0000 to 0000 -- 00588 x 00445 -- 0 rows, 0 cols -- 0 violations -- 60 bits, 7 words, 0kB


MaskRomTool dmg01cpurom.bmp
0000 to 0000 -- 00582 x 00435 -- 0 rows, 0 cols -- 0 violations -- 65 bits, 8 words, 0kB


MaskRomTool dmg01cpurom.bmp
0000 to 0000 -- 00581 x 00436 -- 0 rows, 0 cols -- 0 violations -- 70 bits, 8 words, 0kB


MaskRomTool dmg01cpurom.bmp
0000 to 0000 -- 00586 x 00439 -- 0 rows, 0 cols -- 0 violations -- 75 bits, 9 words, 0kB


MaskRomTool dmg01cpurom.bmp
0000 to 0000 -- 00583 x 00436 -- 0 rows, 0 cols -- 0 violations -- 80 bits, 10 words, 0kB


MaskRomTool dmg01cpurom.bmp
0000 to 0000 -- 00583 x 00435 -- 0 rows, 0 cols -- 0 violations -- 85 bits, 10 words, 0kB

Settings

Sampling | Display

Red 0

Green 0

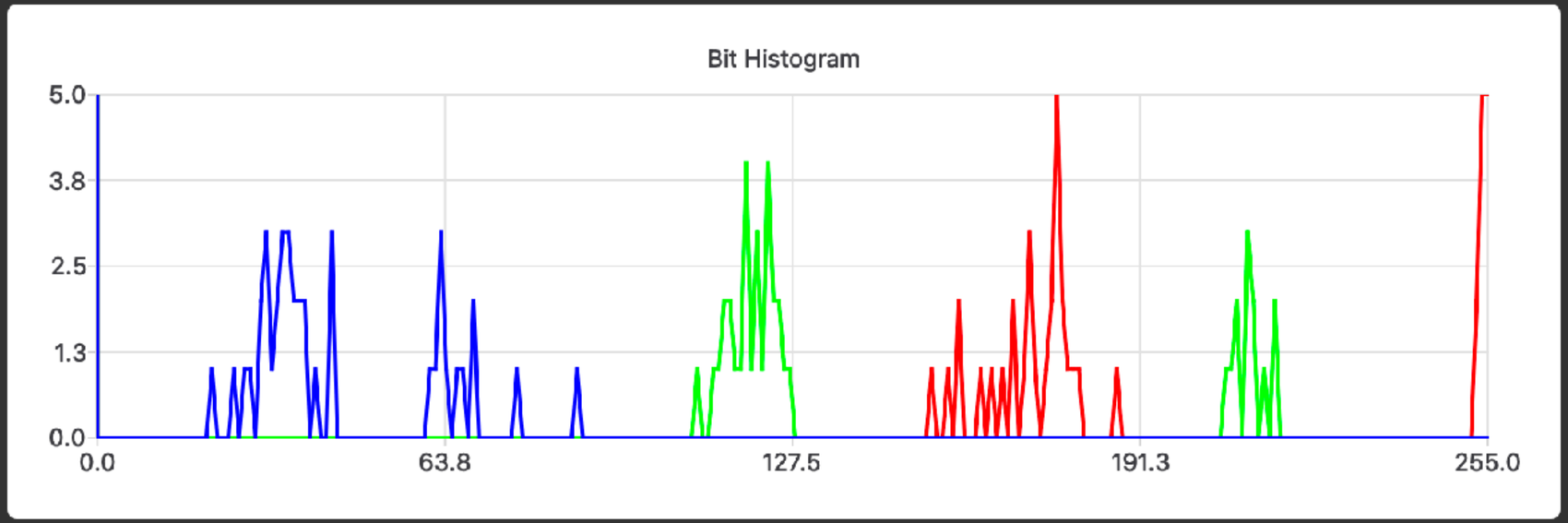
Blue 0

Average

Sampler Default

Size 0

Inverted Bits



Settings

Sampling | Display

Red 0

Green 166

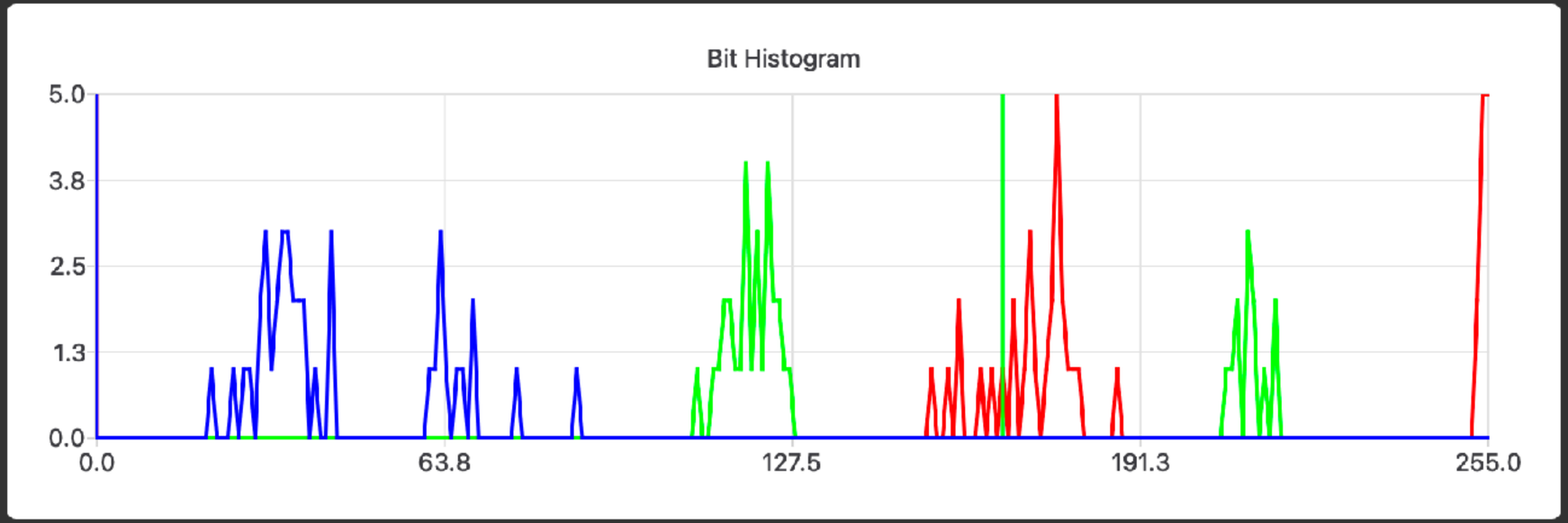
Blue 0

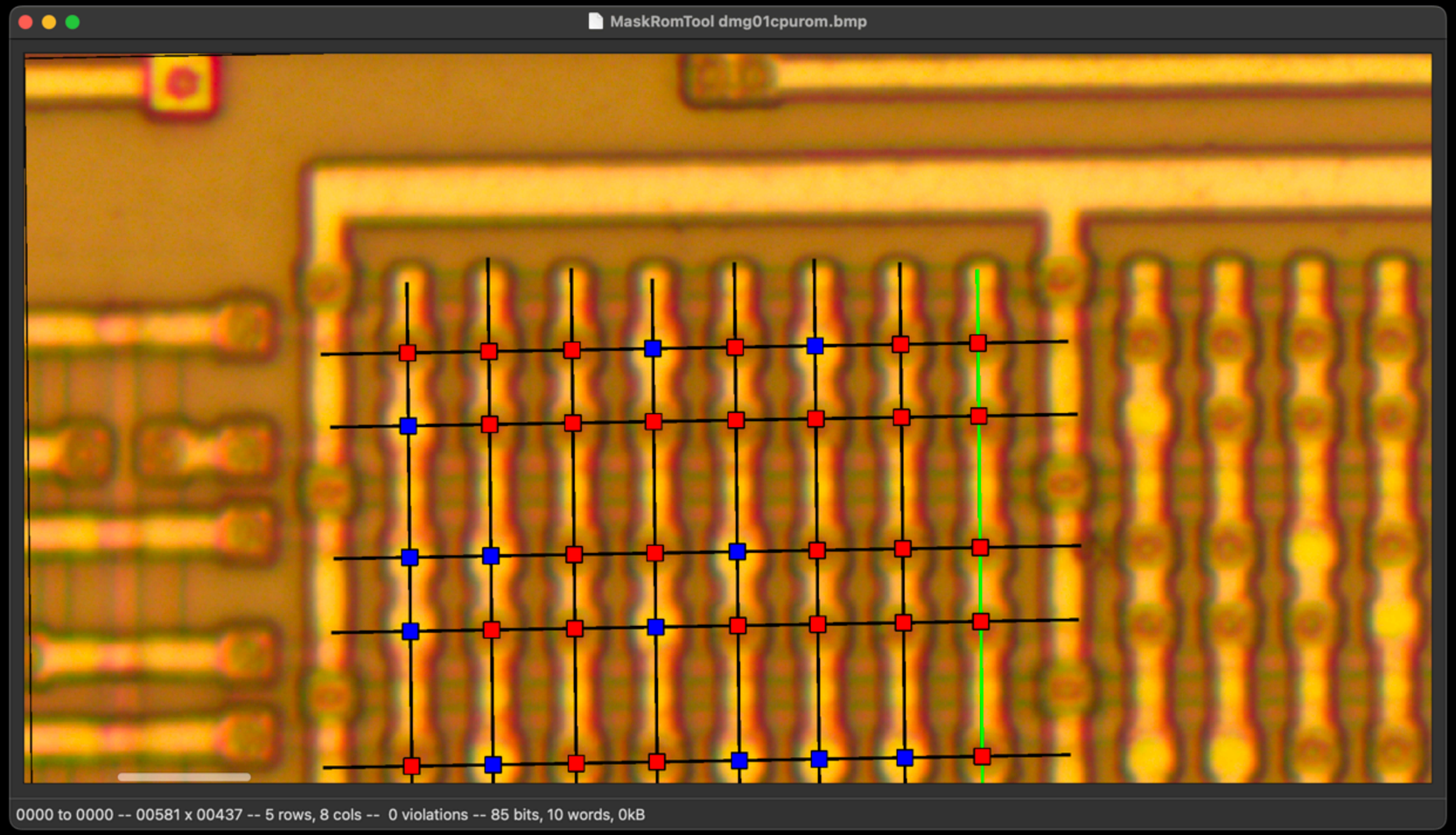
Average

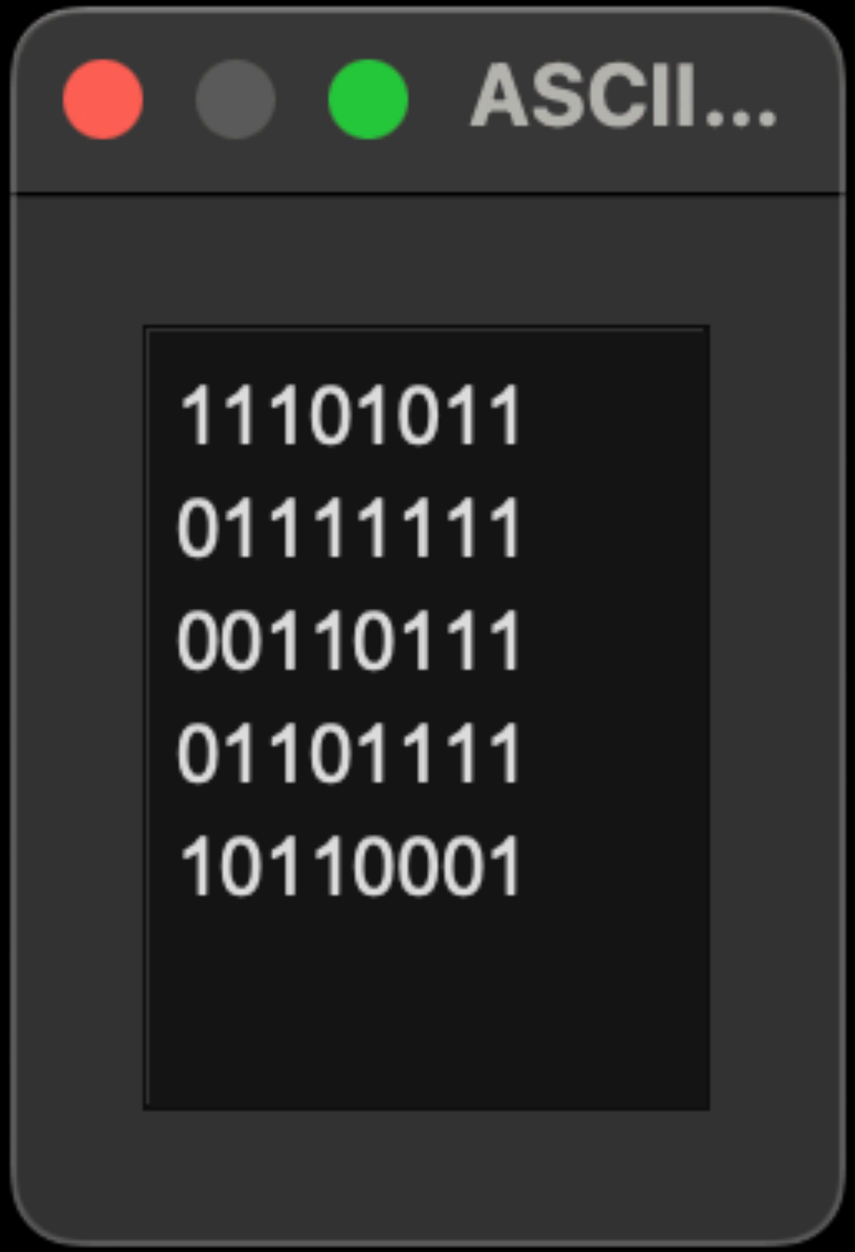
Sampler Default

Size 2

Inverted Bits




MaskRomTool dmg01cpurom.bmp
0000 to 0000 -- 00581 x 00437 -- 5 rows, 8 cols -- 0 violations -- 85 bits, 10 words, 0kB


ASCII...
11101011
01111111
00110111
01101111
10110001

Settings

Sampling | Display

Red 0

Green 155

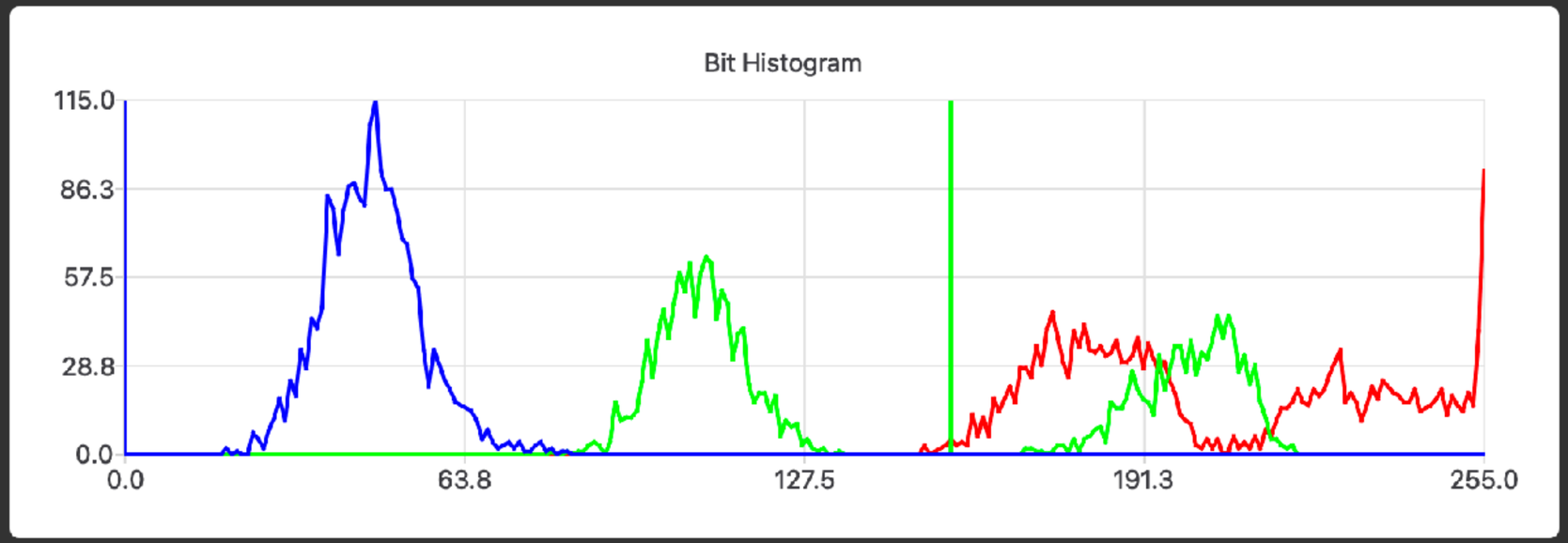
Blue 0

Average

Sampler Default

Size 2

Inverted Bits



ASCII Preview

11101011111100101100101100110010011000110111110000100001011100101110000000110011110001001001001011000100001011110001101100001000
01111111011100110111111101100011001010110100111100000110101010110011001111110011001010111011000000111000001011101101011011101111
00110111110110110111011111010111011001101001011101111111110110100111000101110010010110000101011101110001111101110111101111011000
01101111111000111110111011110110001011100101011010100010001110000111100000111010011111000111001100101111010000110100111111101001
10110001001100001011100110110111000110011101100111100000111001111110110101100011111111001110110101110100001101001000011000100110011
01101110011100110110100100110111011100110101101001001111111001100101111101000110011100110100111011110100001111001110010101100010
01011100011101111111110001110111101100000101101100111000011100010011000011110101001100001011111000110000100110010111011111010010
10110101110101110111010010111110011101011111010000101011111000111010111110000111110101110101011000011100111100110110111110001010
00011111000101000001101110011110101111001111000011111011011100000100011011010000110100111001100100110011110101000101101110110110
11011001000101100111100100011011001110010101010100001101111110111011001011000011101010110110111101010110010110010110111111001011
11111111111110011111101101010101001101111101000010100110101010011011010011110011011010011010101101010100101010111100101101000111
01011110111110100001111011010110001000011101000000111011001011100101001111101110100110110000101000101011100001000001111000100001
11011000100010111011110010010101001011000111000010011000111110001110110110000000100111001101010010001100111101100010100110111001
11111011001010011111101000111101001100101111100110110101011111011011110110000101001011001101000100110111001101011110110110001010
00111111111101100011101111110010001011001111010000011111011110100110111111101110010111011101010101111111110110101011100111111
10101101111101111000110111110110100001010111100111001101101110100100111111000011011110101010001101011100100011111100111110011111

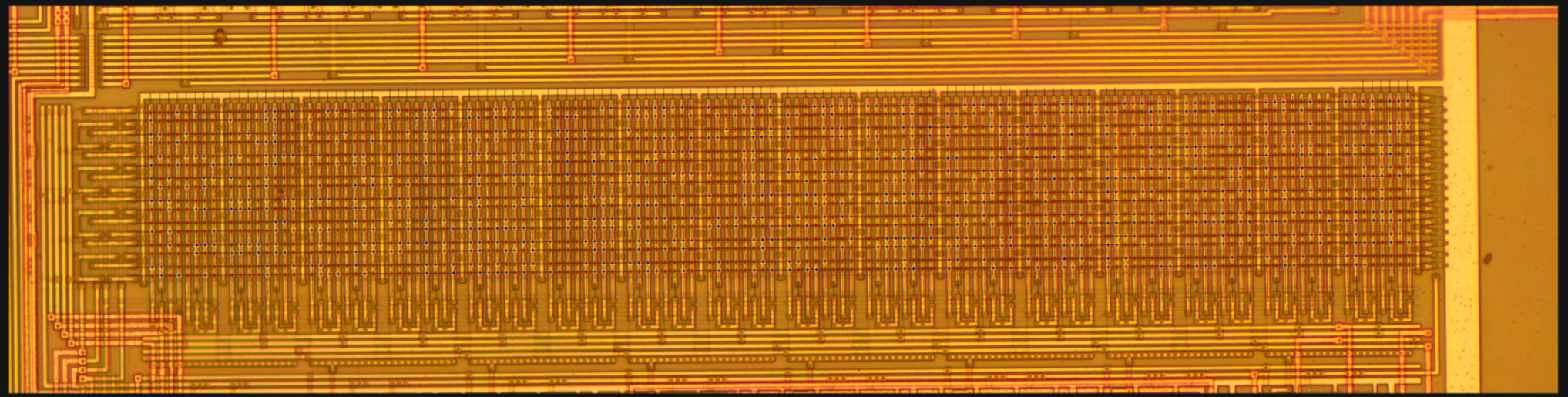
After Bit Extraction

- Bits must be ordered into bytes.
- Errors must be corrected.
- Disassembly, reverse engineering!

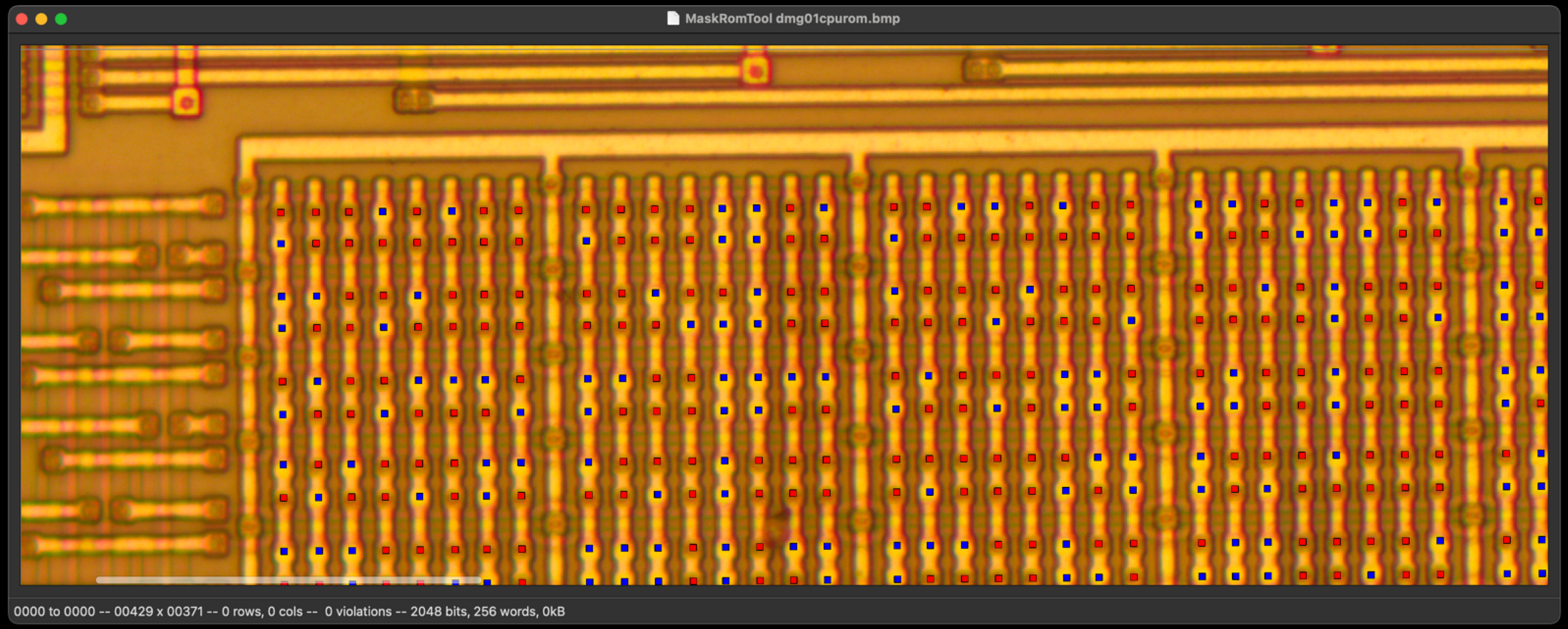
Bits into Bytes

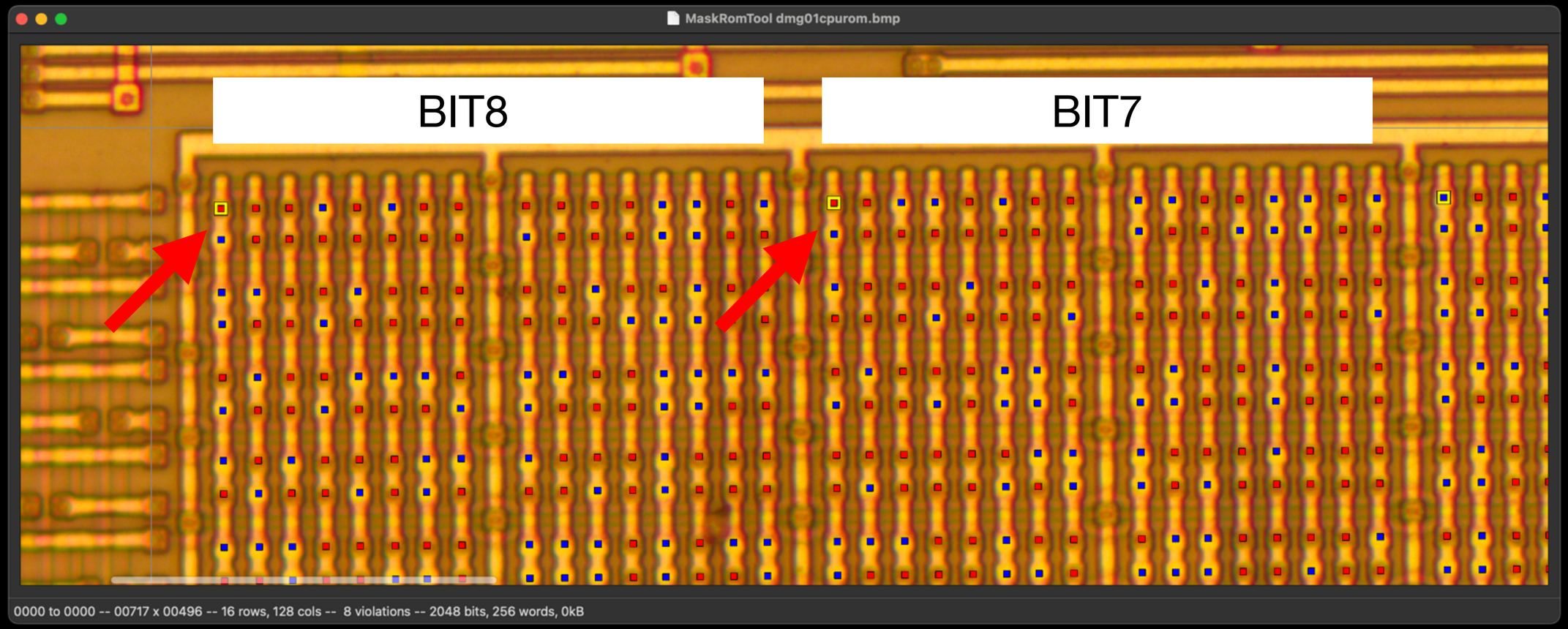
- Each "major column" holds one bit of significance.
  - 8-bit ROM, eight major columns.
  - 32-bit ROM, thirty-two major columns.
- Take one bit from the same position in each major column to form a word.
- Word ordering is less consistent:
  - Left to right? Top to bottom?
  - Rotations, flips.

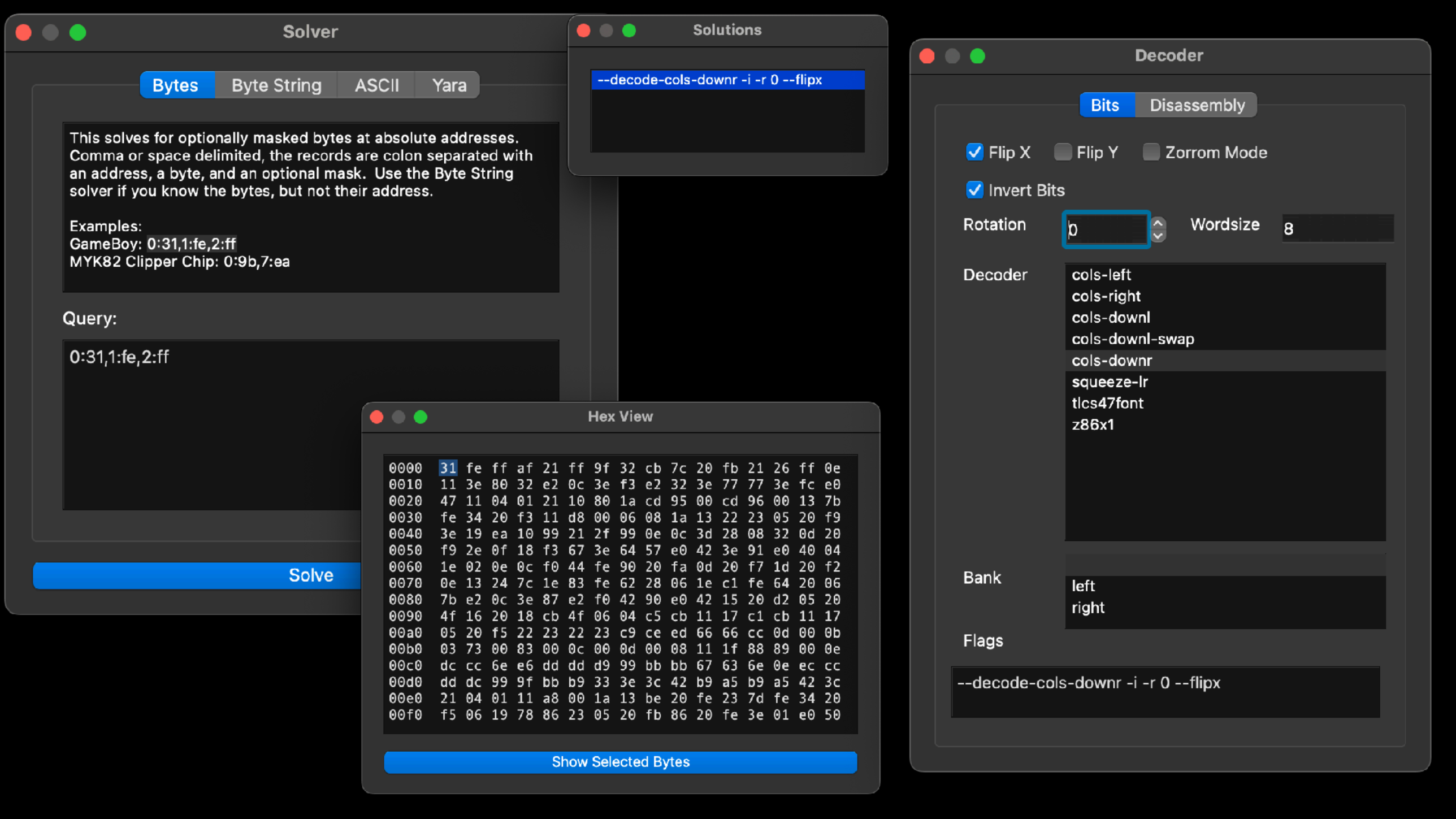
MaskRomTool dmg01cpurom.bmp

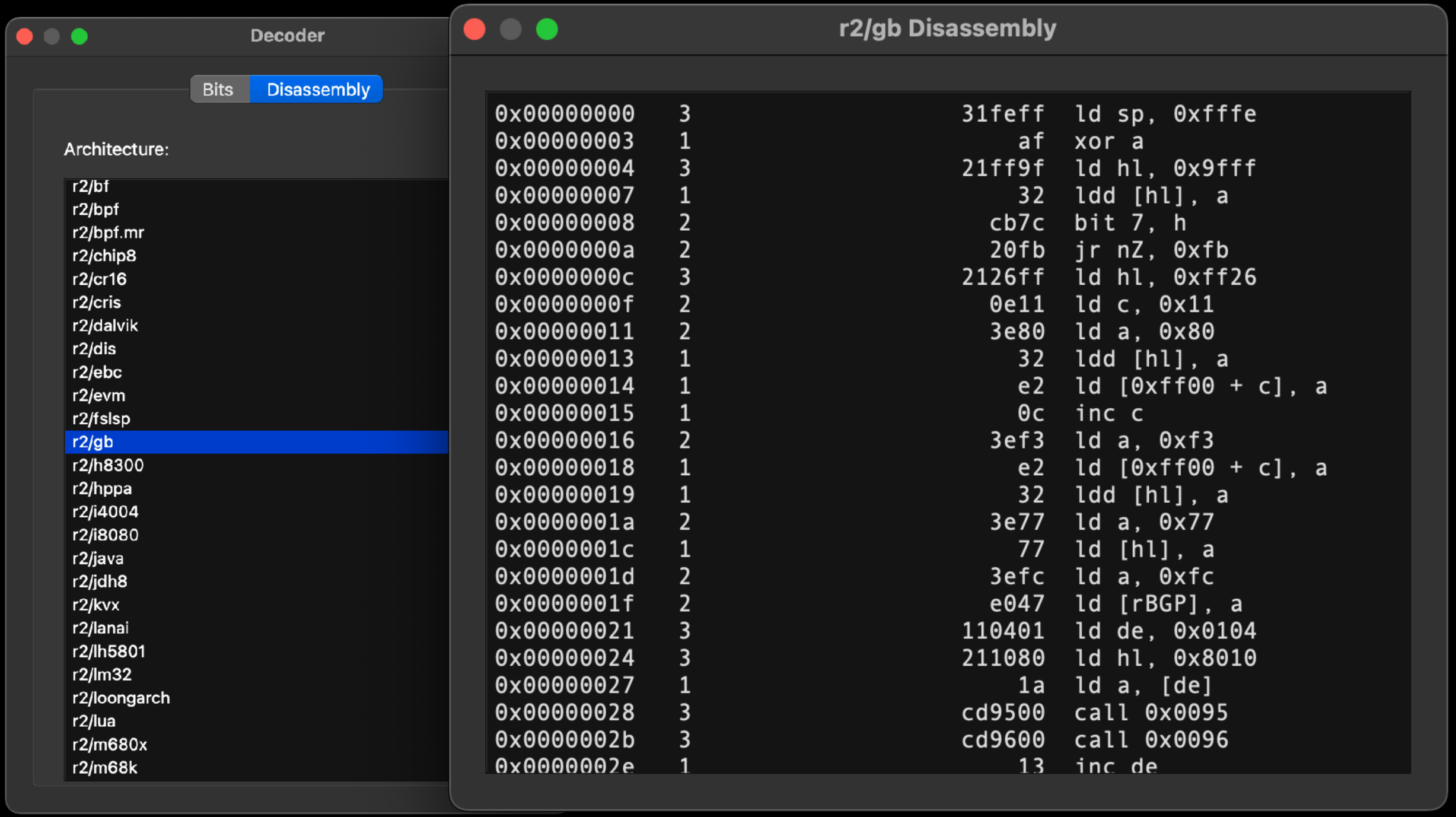


0000 to 0000 -- -0090 x -0414 -- 0 rows, 0 cols -- 0 violations -- 2048 bits, 256 words, 0kB


MaskRomTool dmg01cpurom.bmp
0000 to 0000 -- 00429 x 00371 -- 0 rows, 0 cols -- 0 violations -- 2048 bits, 256 words, 0kB


MaskRomTool dmg01cpurom.bmp
BIT8
BIT7
0000 to 0000 -- 00717 x 00496 -- 16 rows, 128 cols -- 8 violations -- 2048 bits, 256 words, 0kB


Solver
Bytes
Byte String
ASCII
Yara
This solves for optionally masked bytes at absolute addresses. Comma or space delimited, the records are colon separated with an address, a byte, and an optional mask. Use the Byte String solver if you know the bytes, but not their address.
Examples:
GameBoy: 0:31,1:fe,2:ff
MYK82 Clipper Chip: 0:9b,7:ea
Query:
0:31,1:fe,2:ff
Solve
Solutions
--decode-cols-downr -i -r 0 --flipx
Hex View
0000 31 fe ff af 21 ff 9f 32 cb 7c 20 fb 21 26 ff 0e
0010 11 3e 80 32 e2 0c 3e f3 e2 32 3e 77 77 3e fc e0
0020 47 11 04 01 21 10 80 1a cd 95 00 cd 96 00 13 7b
0030 fe 34 20 f3 11 d8 00 06 08 1a 13 22 23 05 20 f9
0040 3e 19 ea 10 99 21 2f 99 0e 0c 3d 28 08 32 0d 20
0050 f9 2e 0f 18 f3 67 3e 64 57 e0 42 3e 91 e0 40 04
0060 1e 02 0e 0c f0 44 fe 90 20 fa 0d 20 f7 1d 20 f2
0070 0e 13 24 7c 1e 83 fe 62 28 06 1e c1 fe 64 20 06
0080 7b e2 0c 3e 87 e2 f0 42 90 e0 42 15 20 d2 05 20
0090 4f 16 20 18 cb 4f 06 04 c5 cb 11 17 c1 cb 11 17
00a0 05 20 f5 22 23 22 23 c9 ce ed 66 66 cc 0d 00 0b
00b0 03 73 00 83 00 0c 00 0d 00 08 11 1f 88 89 00 0e
00c0 dc cc 6e e6 dd dd d9 99 bb bb 67 63 6e 0e ec cc
00d0 dd dc 99 9f bb b9 33 3e 3c 42 b9 a5 b9 a5 42 3c
00e0 21 04 01 11 a8 00 1a 13 be 20 fe 23 7d fe 34 20
00f0 f5 06 19 78 86 23 05 20 fb 86 20 fe 3e 01 e0 50
Show Selected Bytes
Decoder
Bits
Disassembly
Flip X
Flip Y
Zorrom Mode
Invert Bits
Rotation
0
Wordsize
8
Decoder
cols-left
cols-right
cols-downl
cols-downl-swap
cols-downr
squeeze-lr
tlcs47font
z86x1
Bank
left
right
Flags
--decode-cols-downr -i -r 0 --flipx


Decoder
Bits
Disassembly
Architecture:
r2/bf
r2/bpf
r2/bpf.mr
r2/chip8
r2/cr16
r2/cris
r2/dalvik
r2/dis
r2/ebc
r2/evm
r2/fslsp
r2/gb
r2/h8300
r2/hppa
r2/i4004
r2/i8080
r2/java
r2/jdh8
r2/kvx
r2/lanai
r2/lh5801
r2/lm32
r2/loongarch
r2/lua
r2/m680x
r2/m68k
r2/gb Disassembly
0x00000000 3 31feff ld sp, 0xfffe
0x00000003 1 af xor a
0x00000004 3 21ff9f ld hl, 0x9fff
0x00000007 1 32 ldd [hl], a
0x00000008 2 cb7c bit 7, h
0x0000000a 2 20fb jr nZ, 0xfb
0x0000000c 3 2126ff ld hl, 0xff26
0x0000000f 2 0e11 ld c, 0x11
0x00000011 2 3e80 ld a, 0x80
0x00000013 1 32 ldd [hl], a
0x00000014 1 e2 ld [0xff00 + c], a
0x00000015 1 0c inc c
0x00000016 2 3ef3 ld a, 0xf3
0x00000018 1 e2 ld [0xff00 + c], a
0x00000019 1 32 ldd [hl], a
0x0000001a 2 3e77 ld a, 0x77
0x0000001c 1 77 ld [hl], a
0x0000001d 2 3efc ld a, 0xfc
0x0000001f 2 e047 ld [rBGP], a
0x00000021 3 110401 ld de, 0x0104
0x00000024 3 211080 ld hl, 0x8010
0x00000027 1 1a ld a, [de]
0x00000028 3 cd9500 call 0x0095
0x0000002b 3 cd9600 call 0x0096
0x0000002e 1 13 inc de

Software and target!

- https://github.com/travisgoodspeed/maskromtool/
  - Latest release for Windows or macOS.
  - Build from source with Qt6 for Linux.
- https://github.com/travisgoodspeed/gbrom-tutorial
  - Clone this locally, follow instructions in README.

Cheat Sheet

- Left-click sets source position.
  - Left-drag selects lines.
- Middle-drag pans the view.
- Right-drag moves the selected lines.
- R places a Row, C places a Column, SPACE duplicates last placement.
- D deletes a selection, S sets a new end position.
- Shift+D duplicates a selected group, then right-drag to move the copy.
- V to check for errors, E for next error.