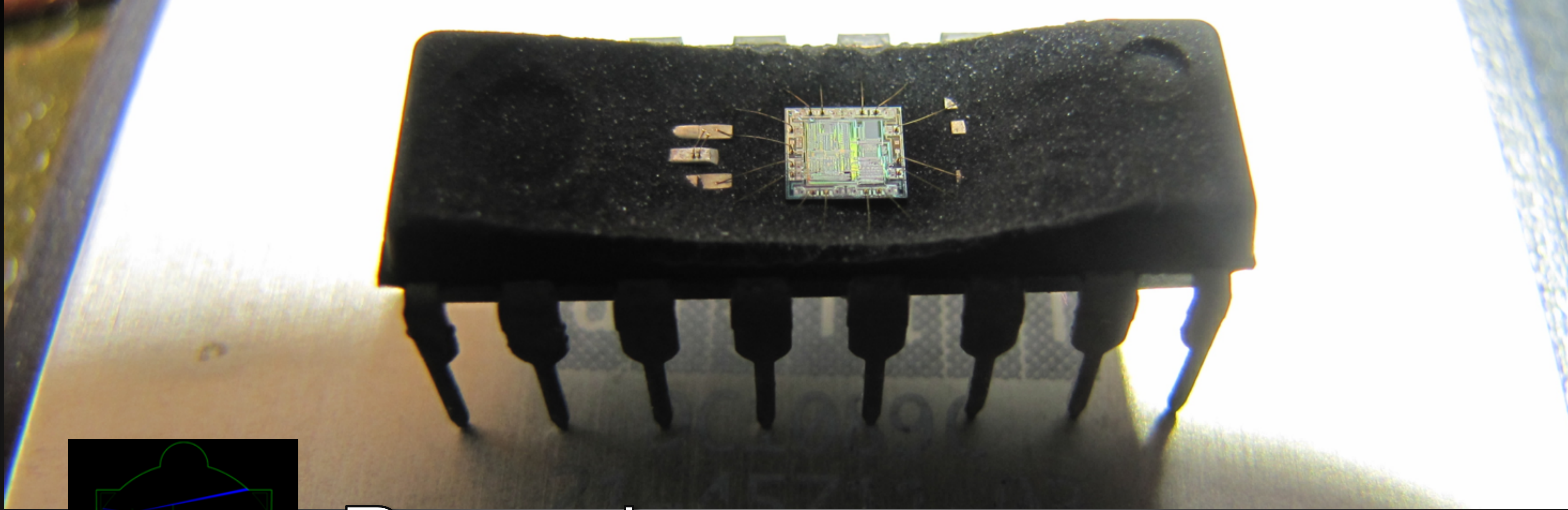
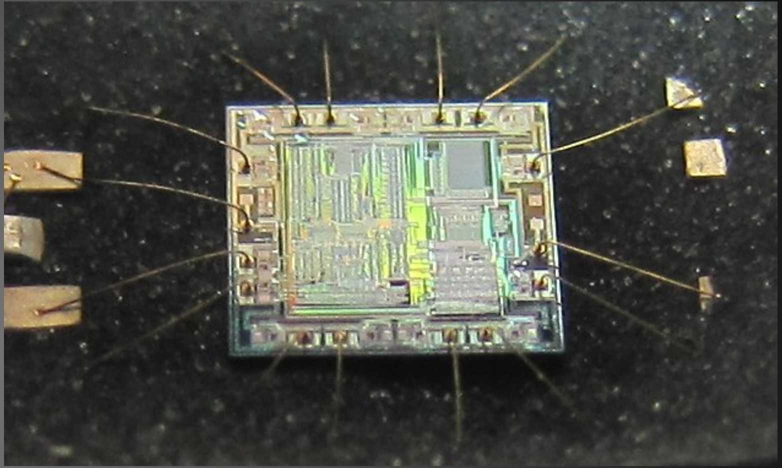


Reversing the Nintendo 64 CIC

Mike Ryan, John McMaster, marshallh

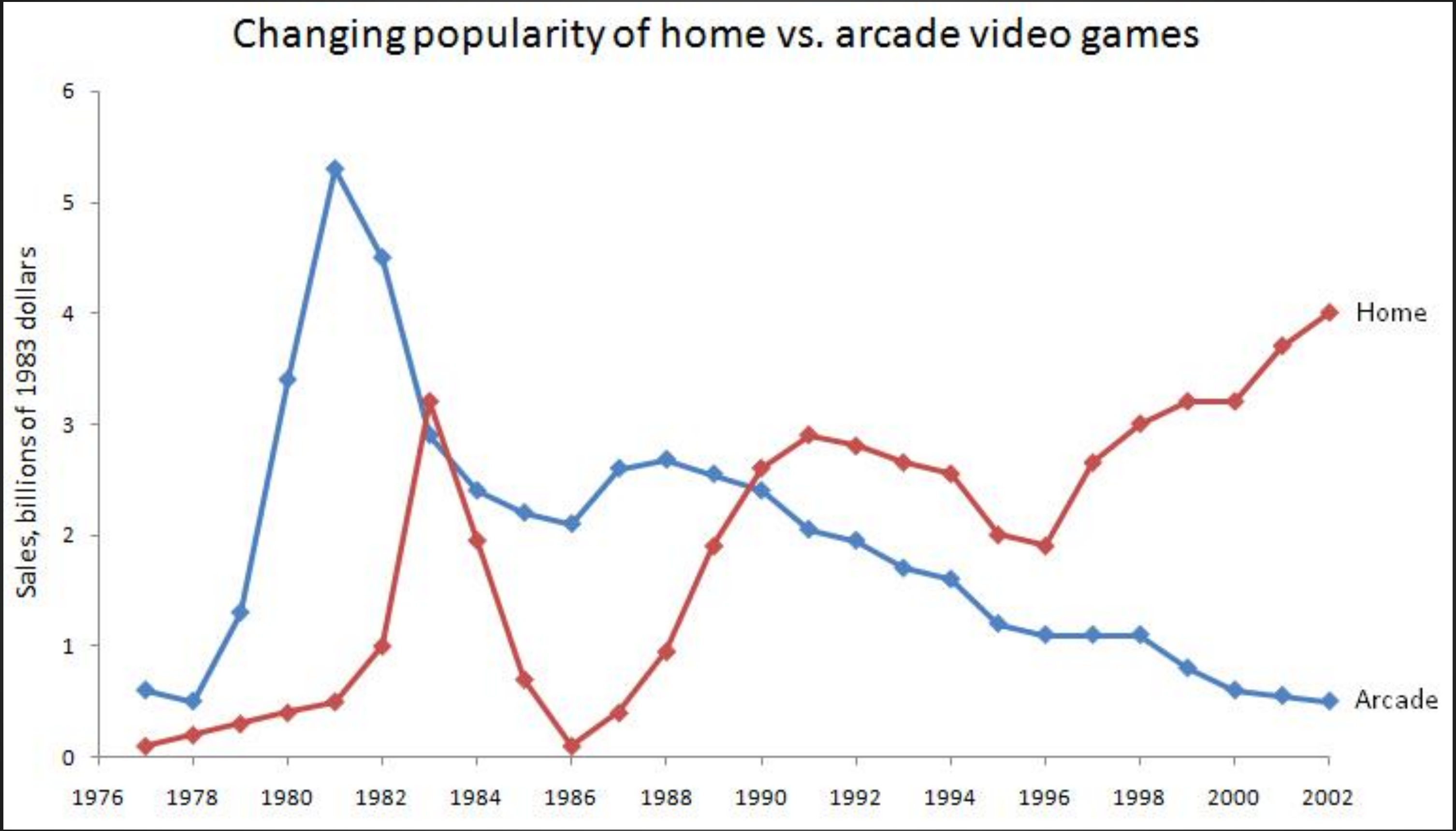
REcon 2015-06-21

Introduction



History

ATARI
GAMES



Source: http://vincenzoferme.github.io/informatics_history_HCI_atelier_2015/html/games/videogame_crash.html

“Atari collapsed because they gave too much freedom to third-party developers and the market was swamped with rubbish games.”

Hiroshi Yamauchi

President of Nintendo, 1986

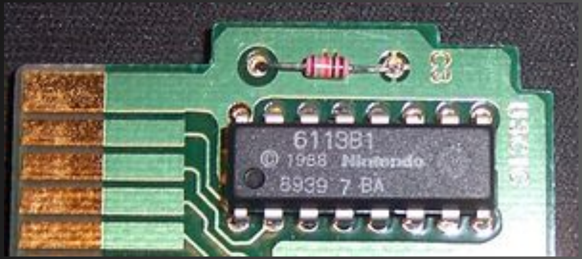
Official

Nintendo®

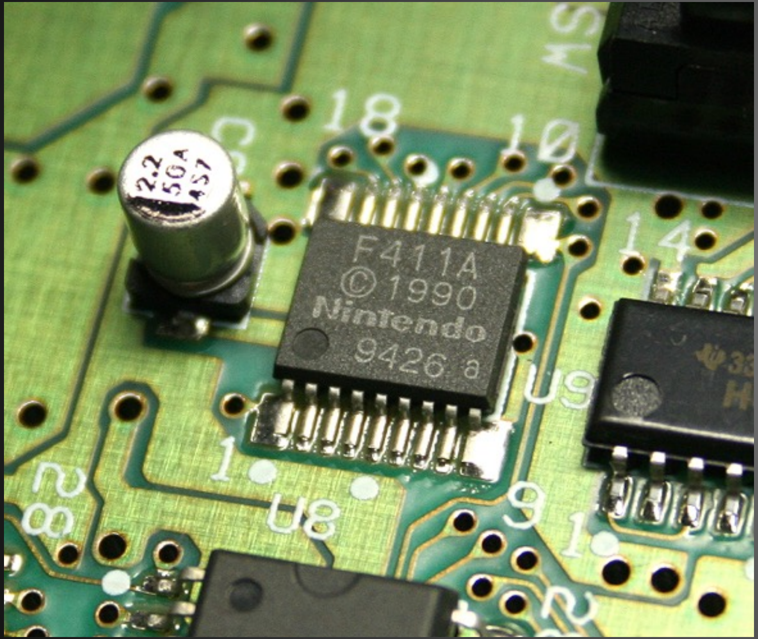
Seal

Birth of the CIC

AKA the 10NES

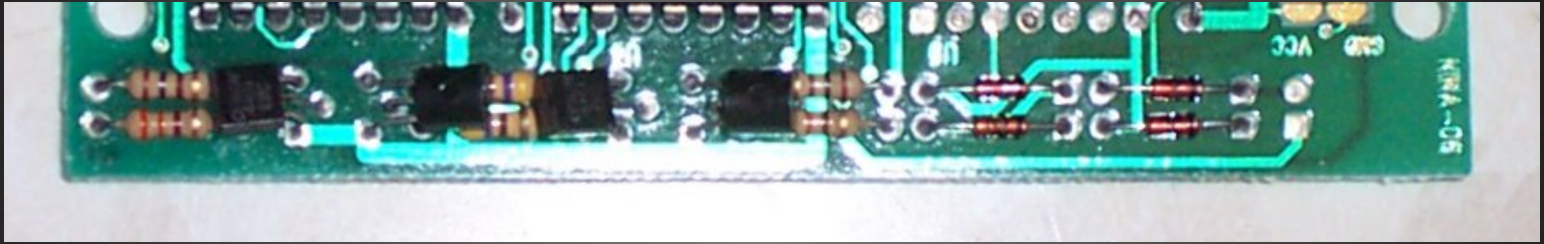


SNES


F411A
©1990
Nintendo
9426 a

SUPER NINTENDO
POWER
RESET
EJECT

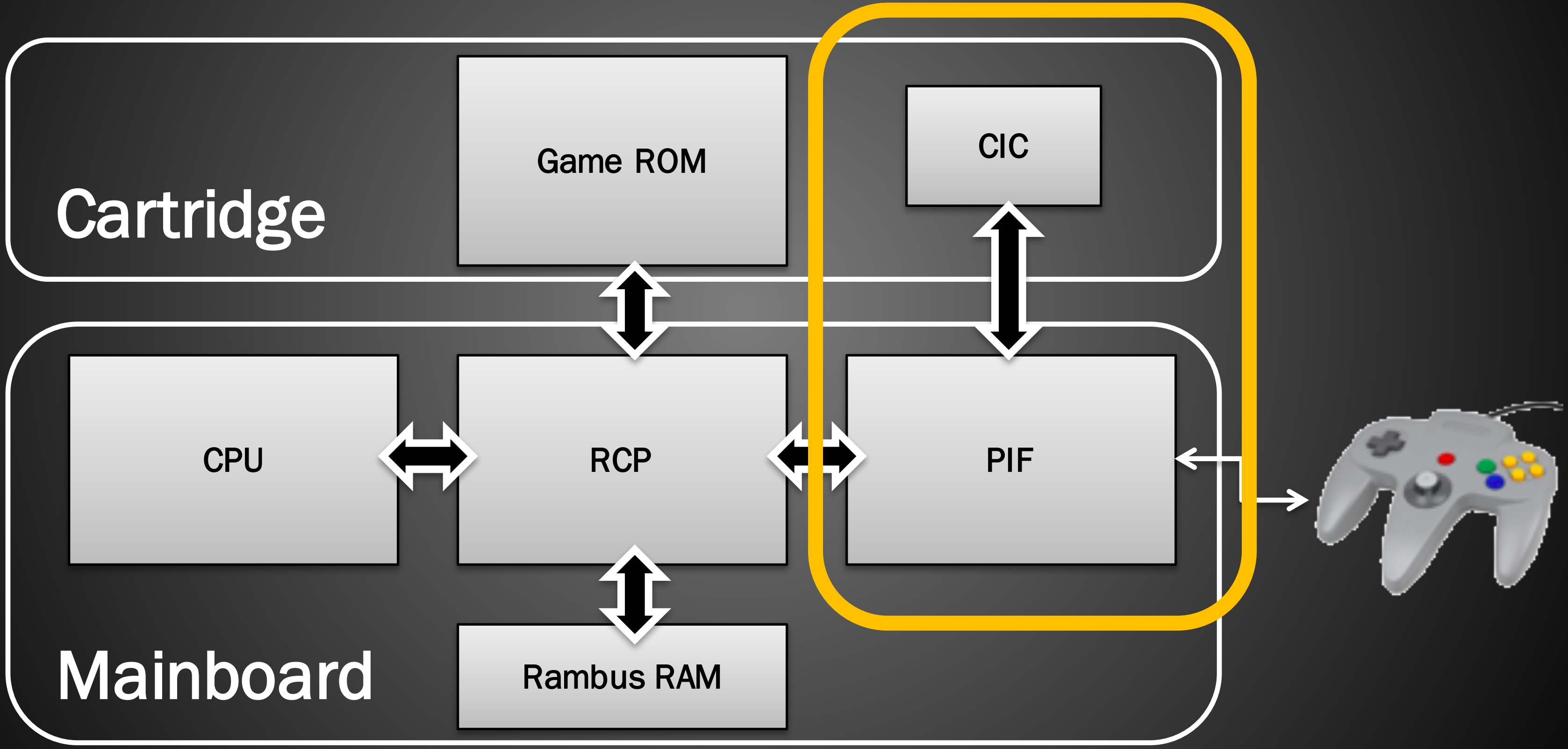
Bypassing the NES CIC

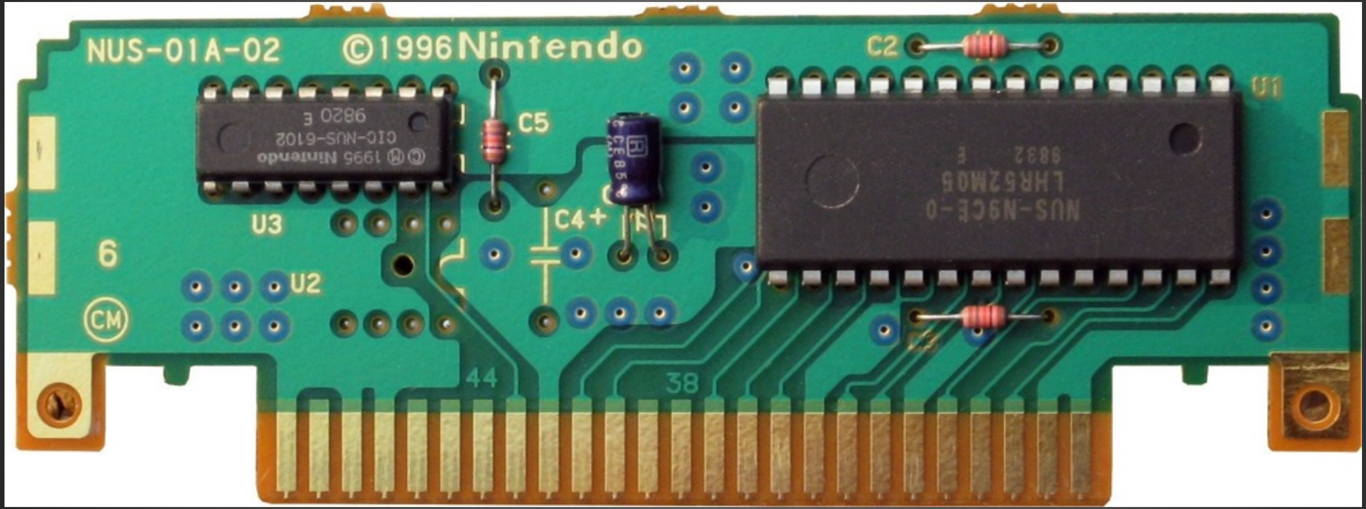


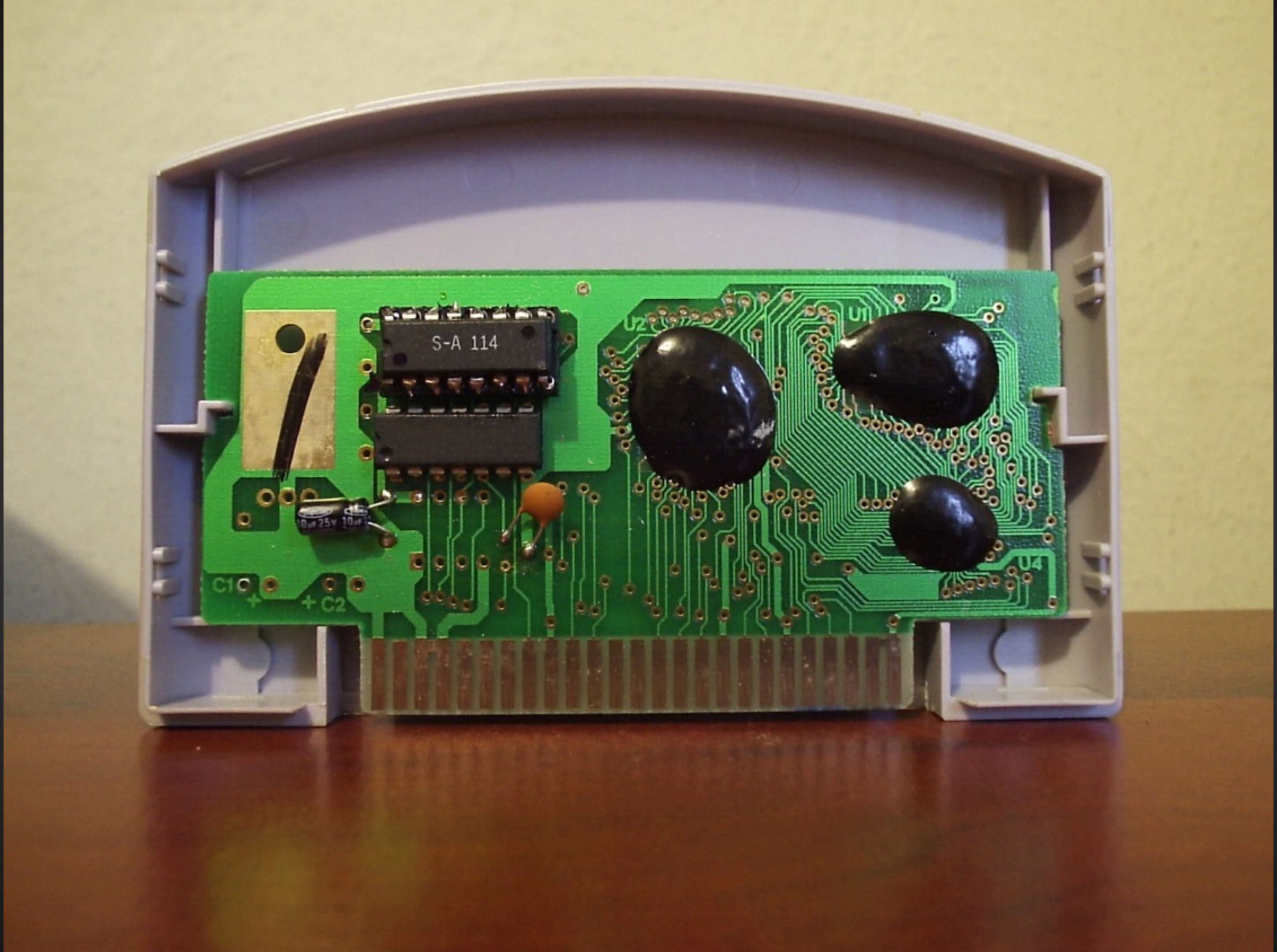
Sneaky Sneaky Atari

- Atari Games Corp. v. Nintendo of America Inc.
- “This ‘reverse engineering’ process, to the extent untainted by the 10NES copy purloined from the Copyright Office, qualified as a fair use.”

N64 Block Diagram

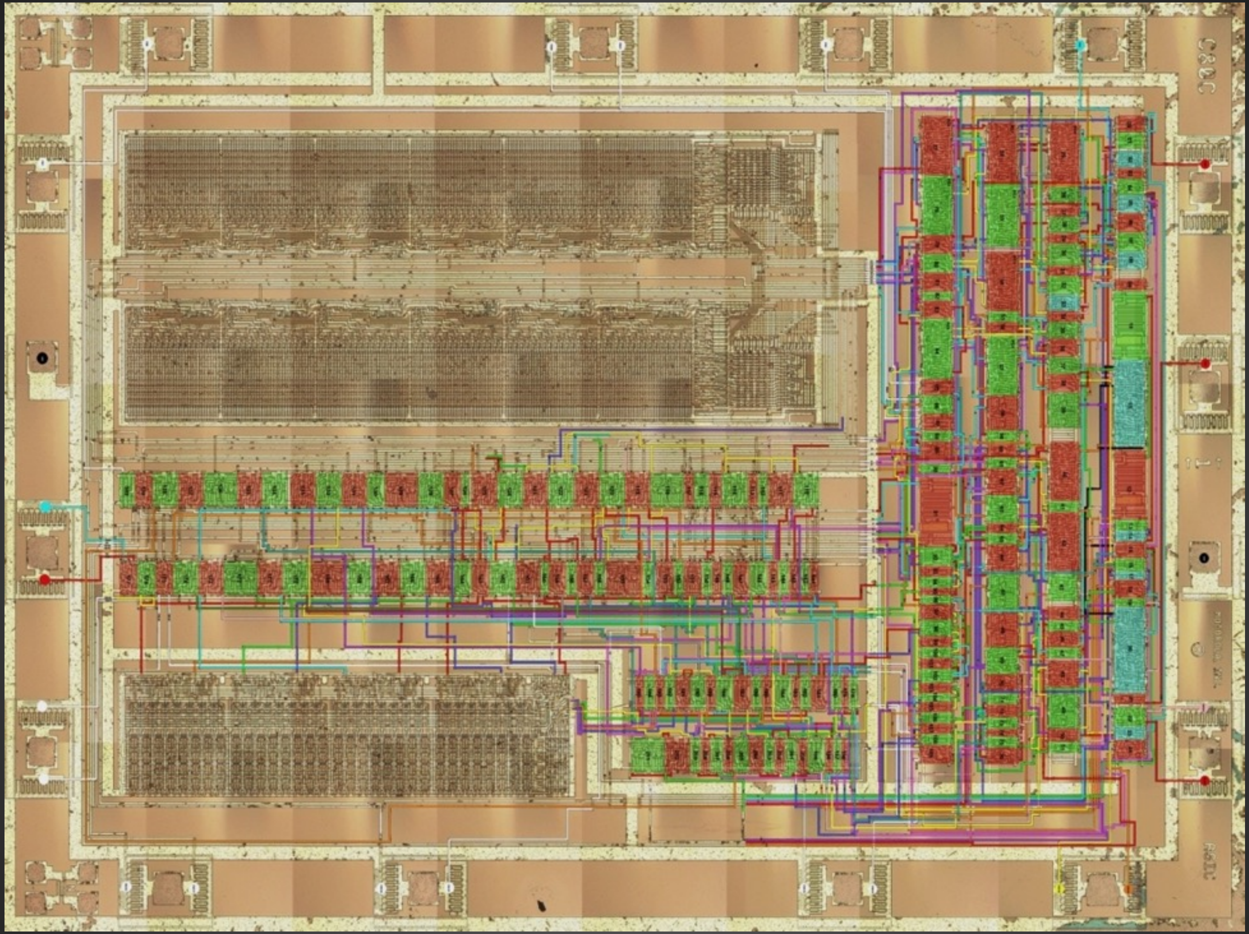



NUS-01A-02
©1996 Nintendo
CIC-NUS-6102
9820 E
© Ⓜ 1995 Nintendo
NUS-N9CE-0
LHR52M05
9832 E
C2
C4+
C5
U1
U2
U3
6
CM
44
38


S-A 114
U2
U1
U4
C1
C2
10µF 25V 10µF

NINTENDO64

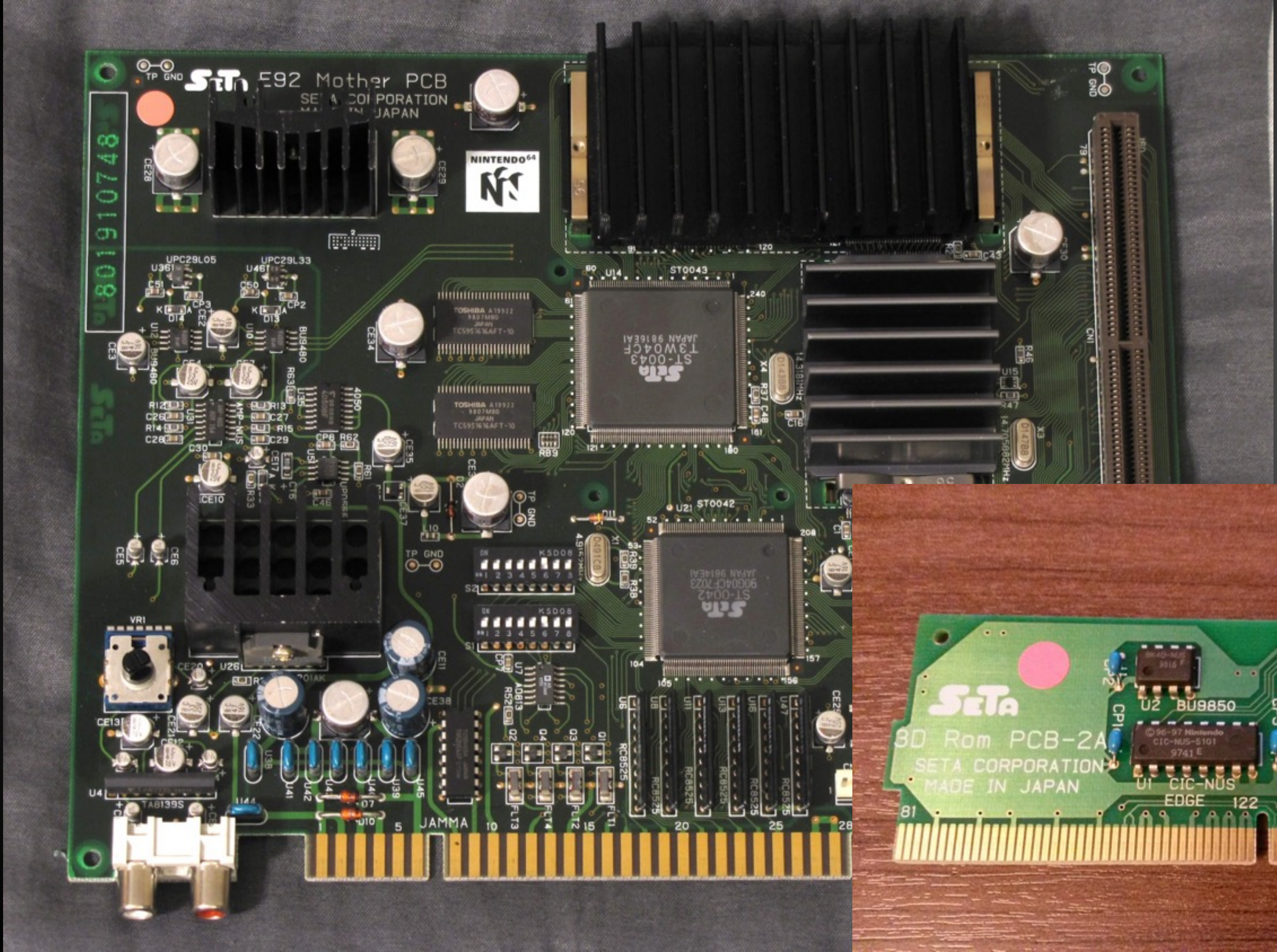
Modern CIC Clones

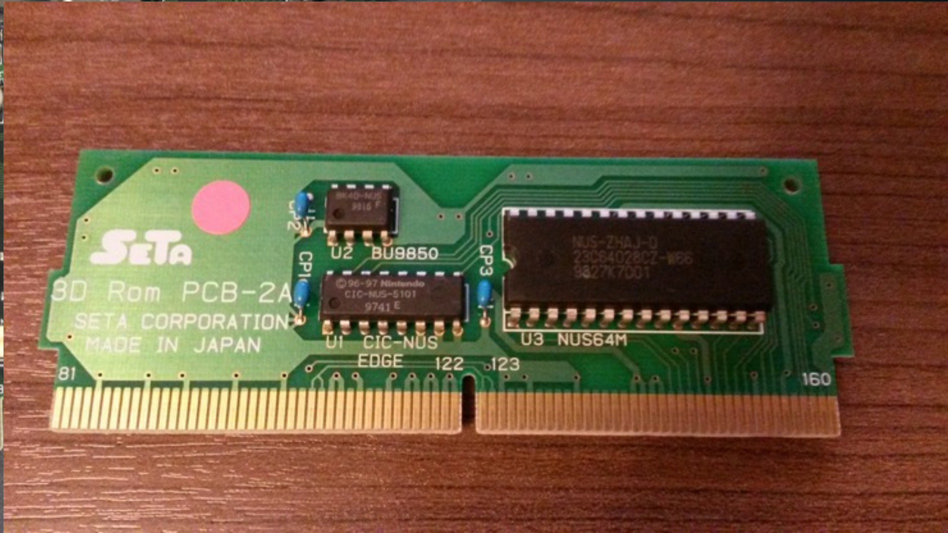

Source: neviktsi

Insert CompactFlash
Insert MicroSD (rear)
64drive
DEVELOPMENT TOOL
64drive.retroactive.be
RETRO active

M-m-m-m-multi CIC

- 6 variants / region
- 2 regions

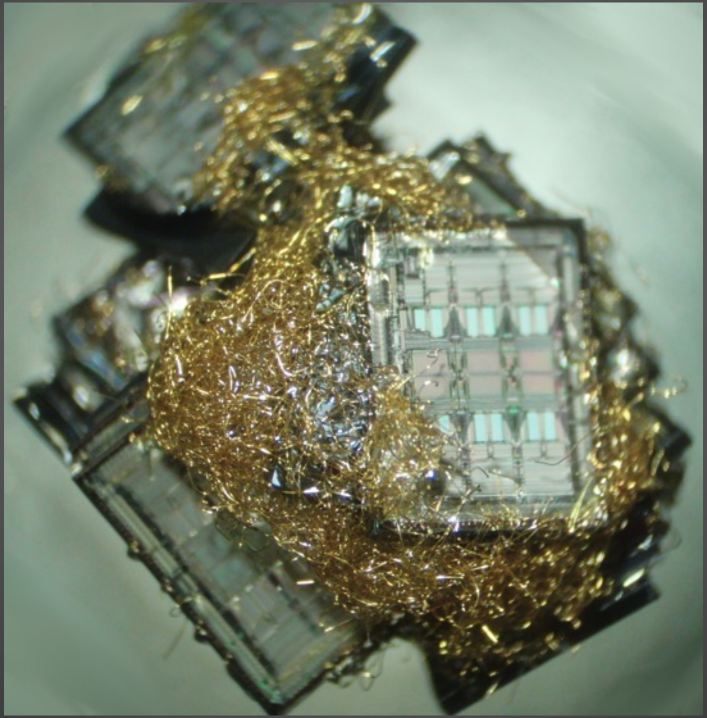




Credit: brizzo

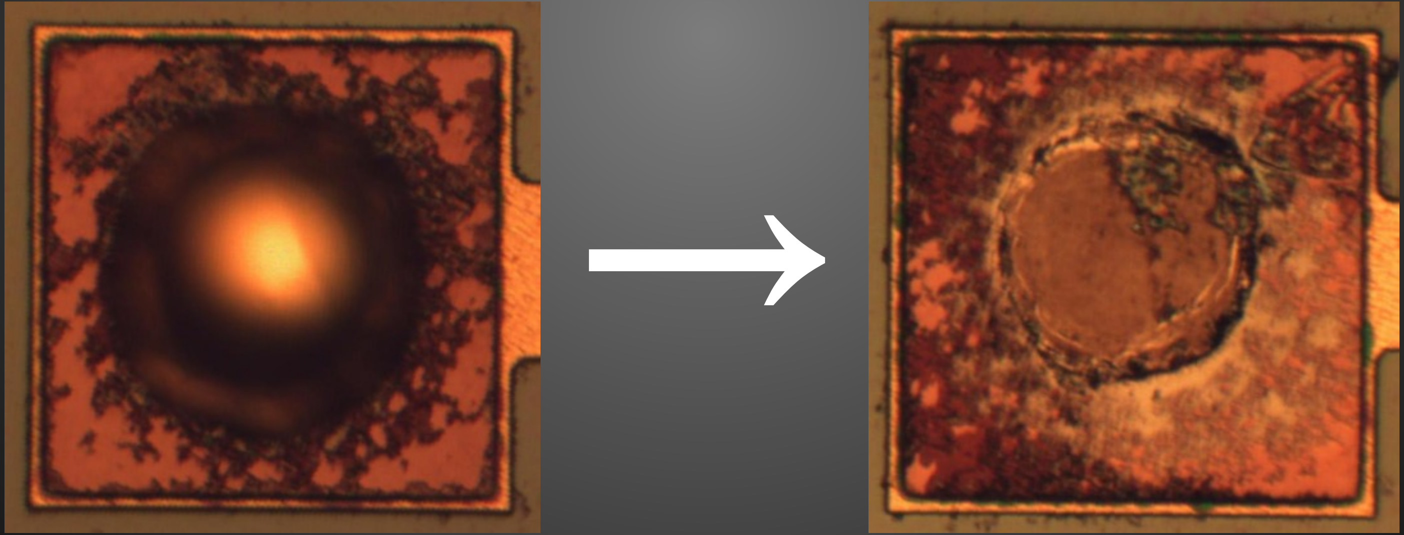
Getting @ silicon

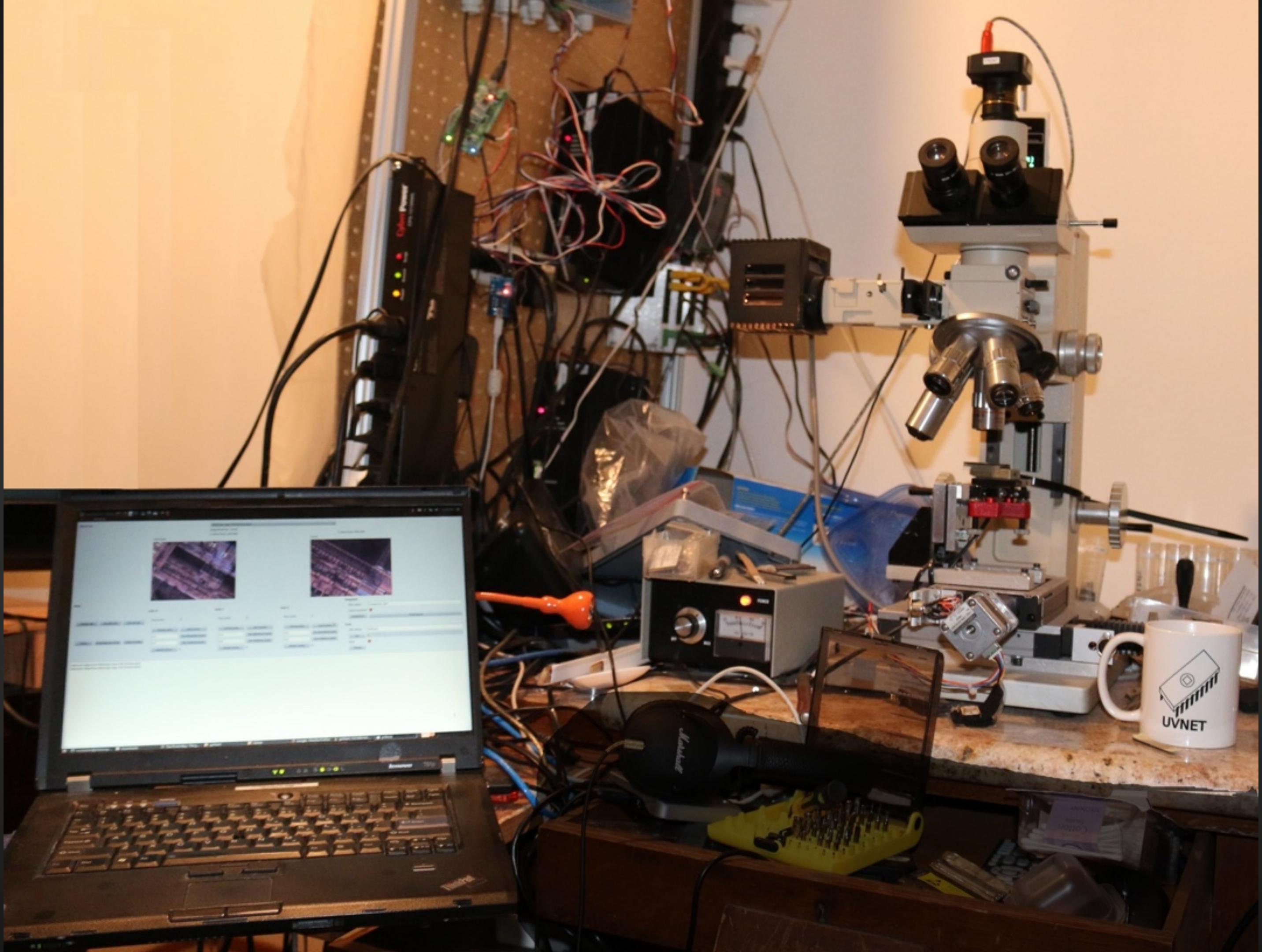
Resulting tangle



Removing bond wires

- Pluck with tweezers
- Solder amalgam



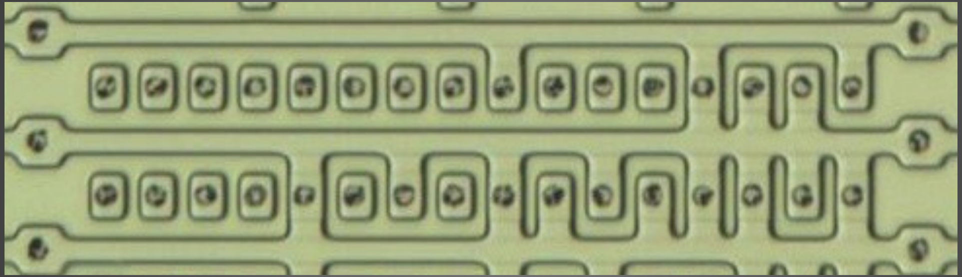

UVNET

Delayer

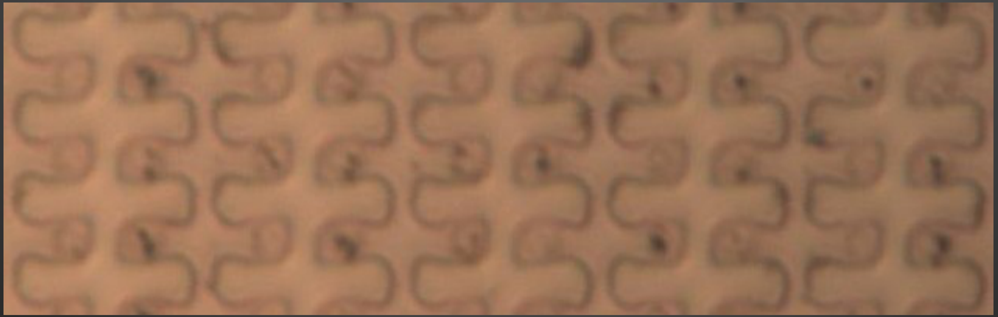
- Remove SiO_2 + metal
- HF uneven, AlF_3 residue
- Solution: vortex, buffer w/ NH_4F (BOE)

ROM silicon

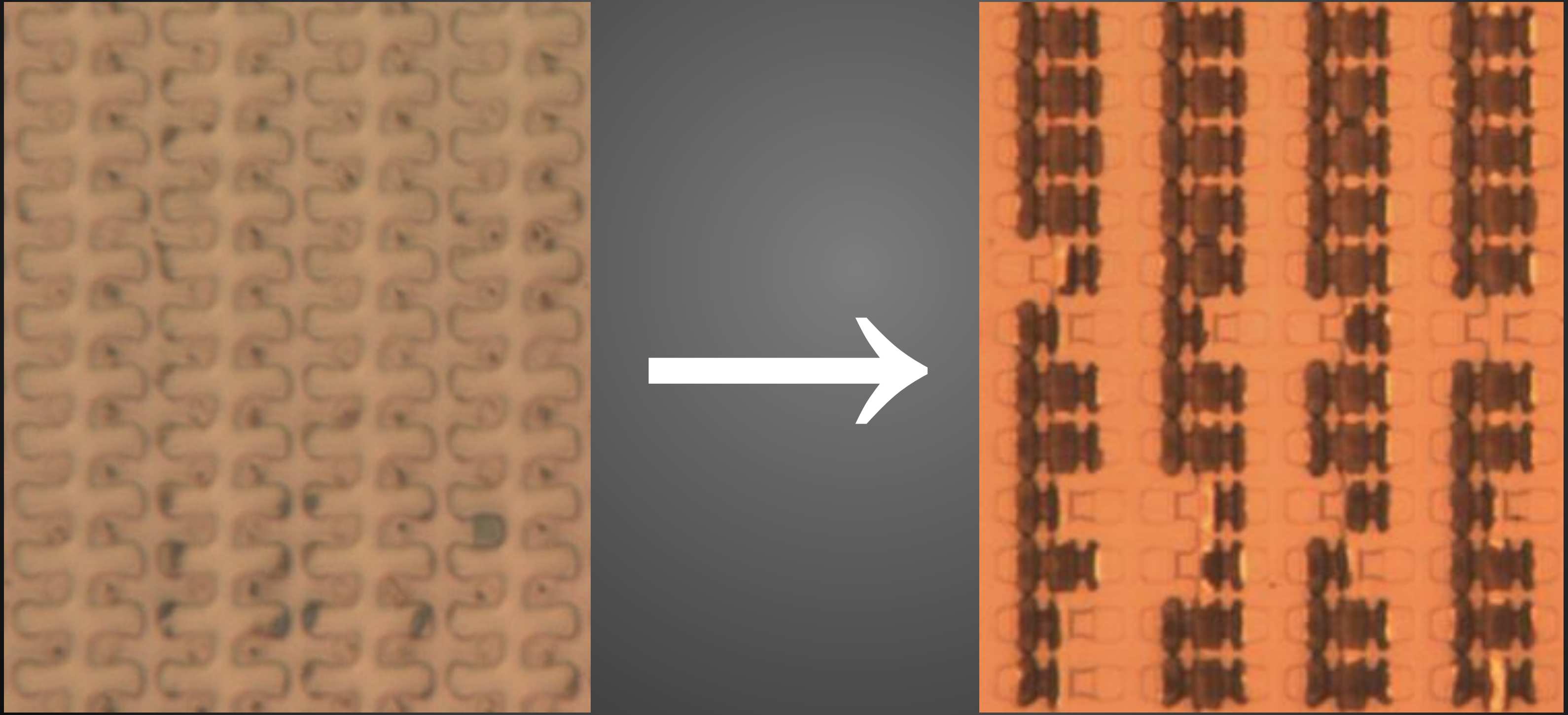
- An active programmed ROM is easy to see:



- CIC ROM: where's the data?

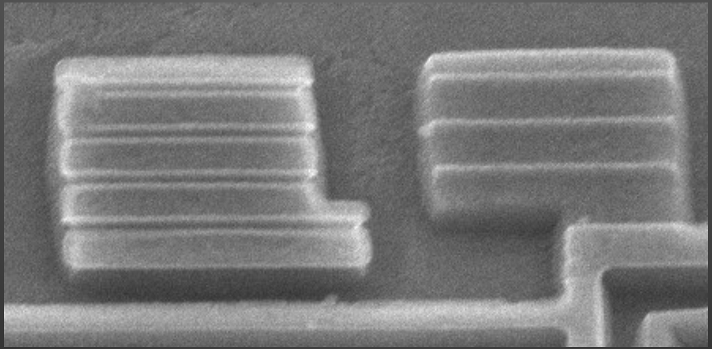


Dash etch



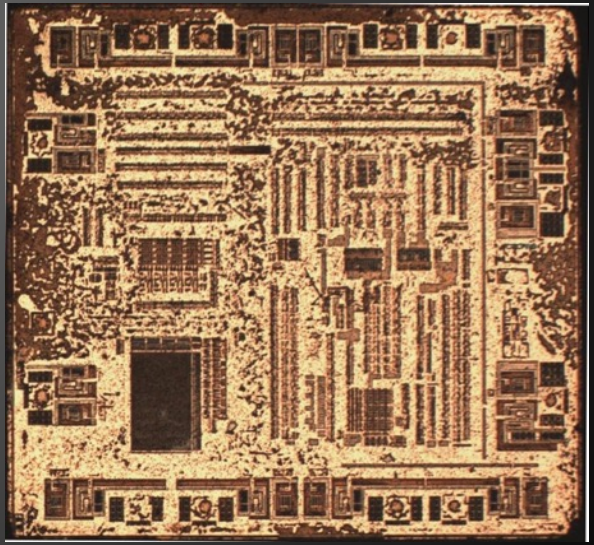
Dash: theory

- Doping influences competing reactions
  - HNO3: convert Si to SiO_2
  - HF: dissolve SiO_2
- Form thin SiO_2 => optical interference



Dash: practice

- Etch 10-15 seconds w/ light recommended
- Many fine points for good etch
- Over etching destroys data



Dash: process variation

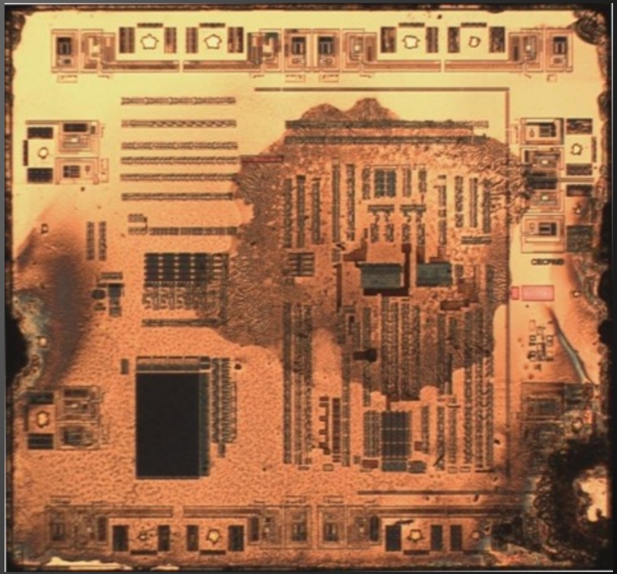
- Different oxide thickness, etc
- Need to practice a few times
- Old sports games: $0.50/cart

FIFA Soccer 64
International Superstar Soccer 64
FIFA 98
International Superstar Soccer '98
FIFA 99
World Cup 98
Mia Hamm Soccer

→

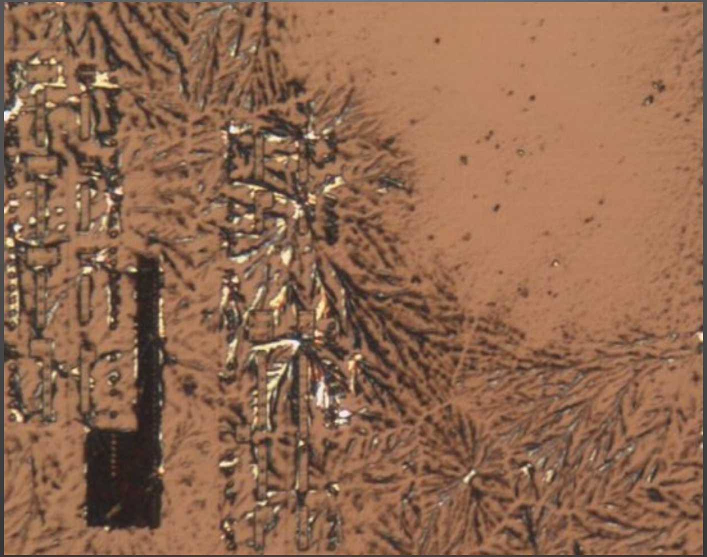
Dash: cleanliness, chemical purity

- Small residues can block etchant
- Very sensitive to other metals, ex Cu
  - Beware decap residue
- Recommend dedicated glassware



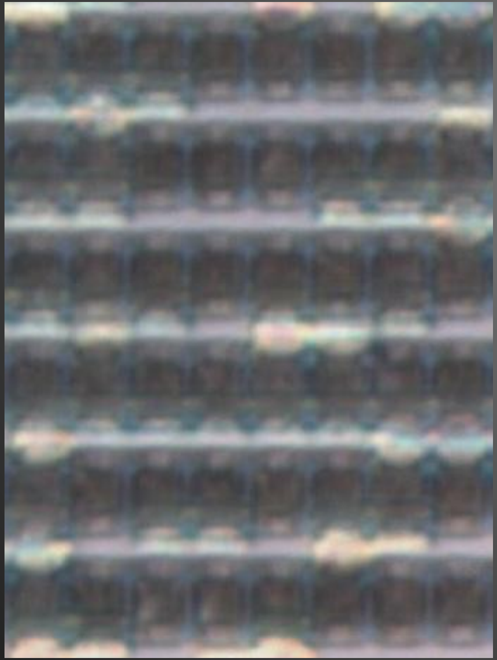
Dash: temperature

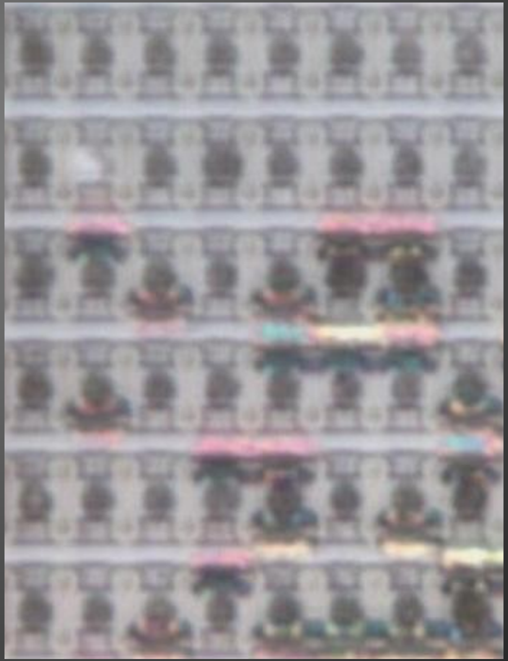
- Strongly influences rate
- HAc crystallization (below)



Dash: lapping

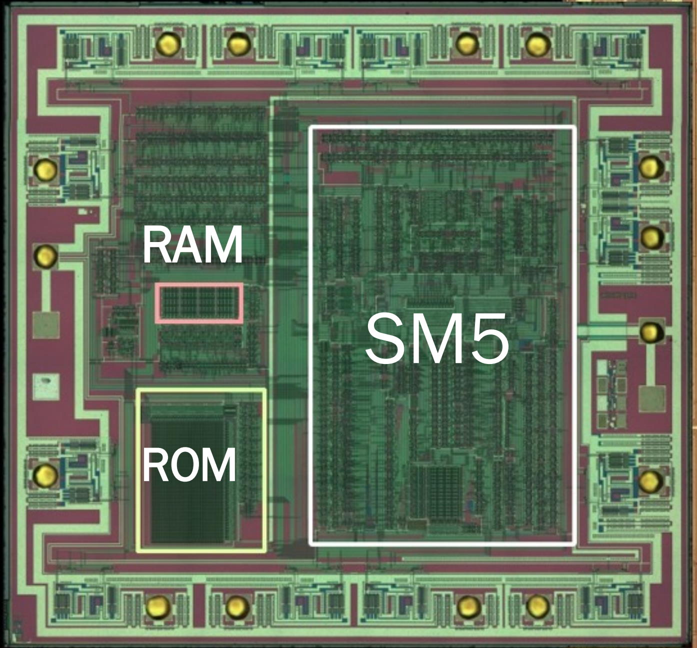
- Chip lapped at angle and stained



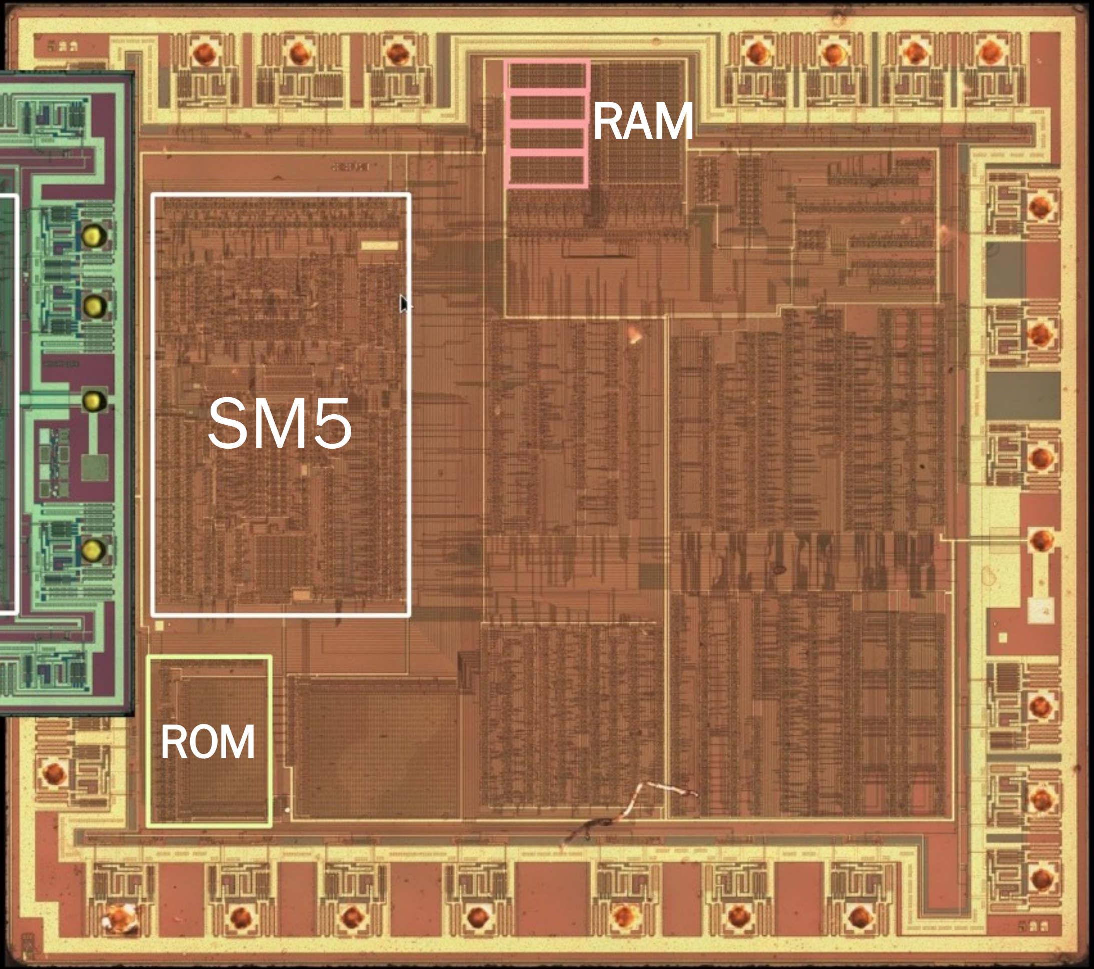


Safety

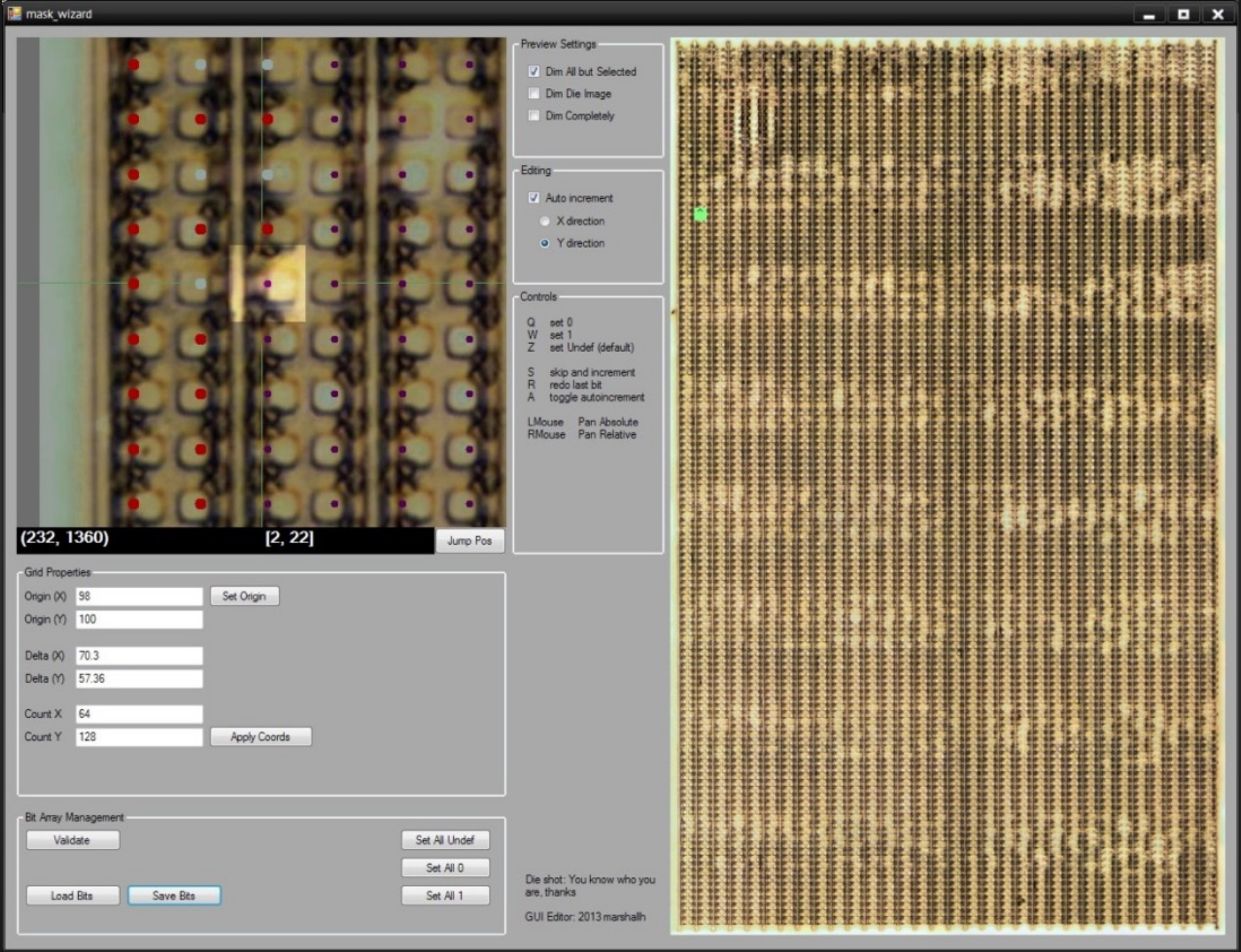
- Aggressive chemicals: HazMat suit / respirator
- Risk life/limb for science!



CIC



PIF



Preview Settings

- [✓] Dim All but Selected
- [ ] Dim Die Image
- [ ] Dim Completely

Editing

- [✓] Auto increment
  - X direction
  - [✓ Y direction]

Controls

```
Q       set 0
W       set 1
Z       set Undef (default)

S       skip and increment
R       redo last bit
A       toggle autoincrement

LMouse  Pan Absolute
RMouse  Pan Relative
```

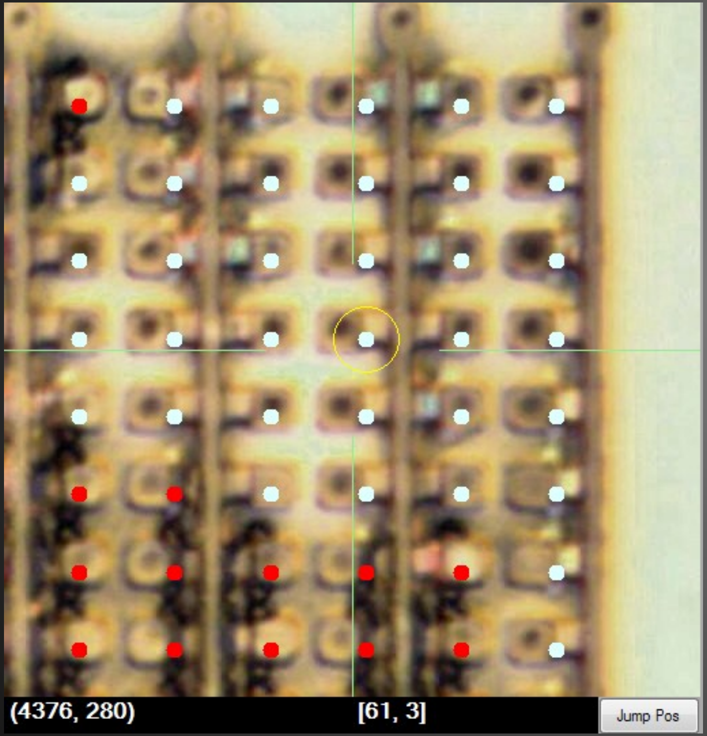
Grid Properties

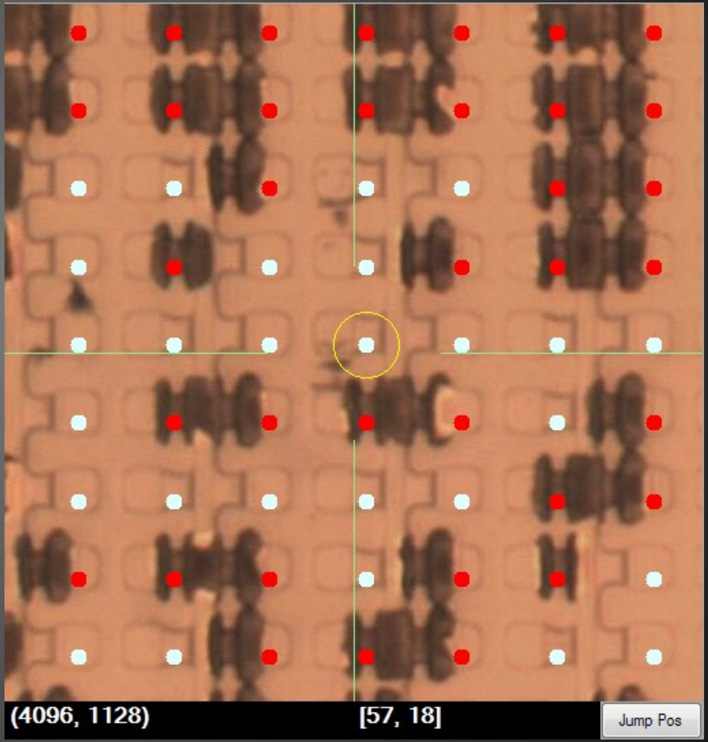
Origin (X)	98
Origin (Y)	100
Delta (X)	70.3
Delta (Y)	57.36
Count X	64
Count Y	128

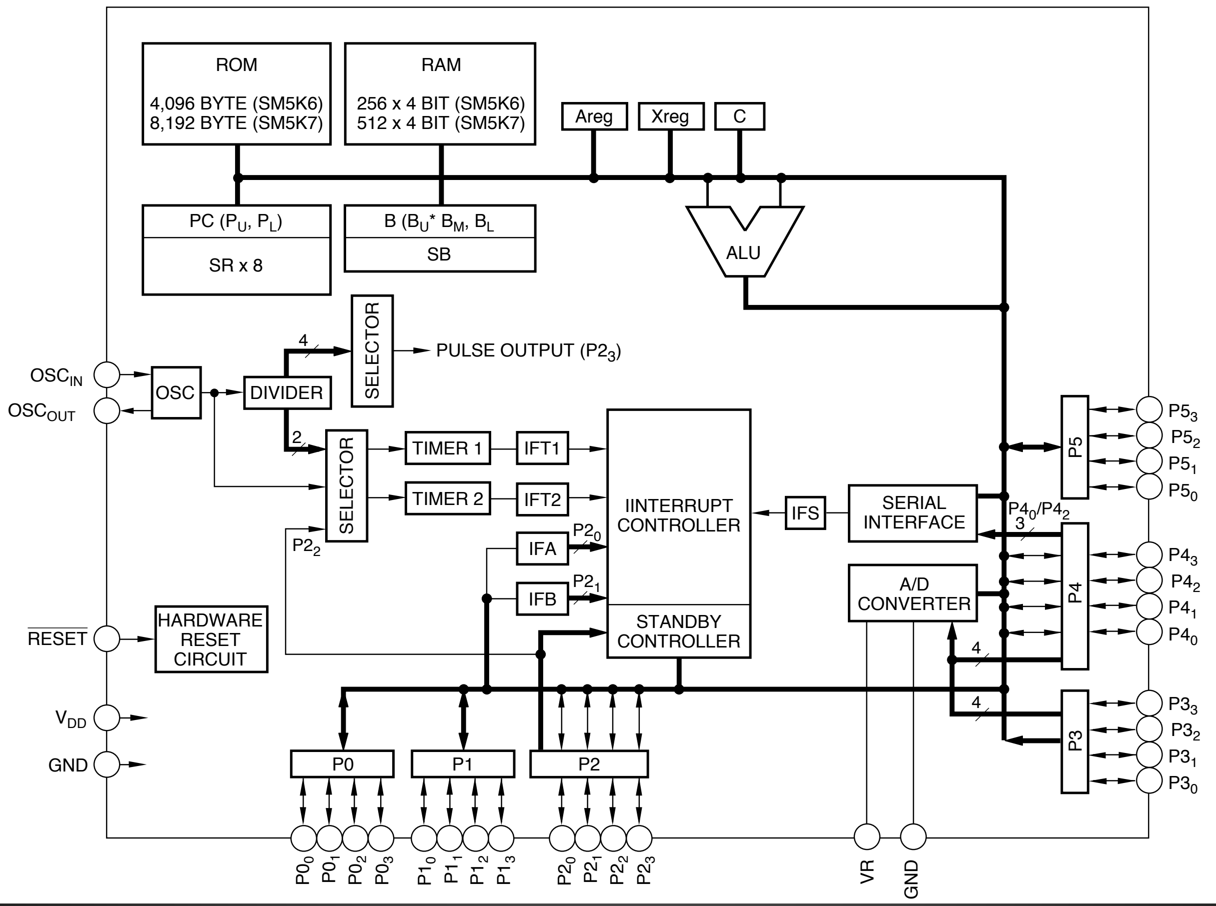
Die shot: You know who you are, thanks

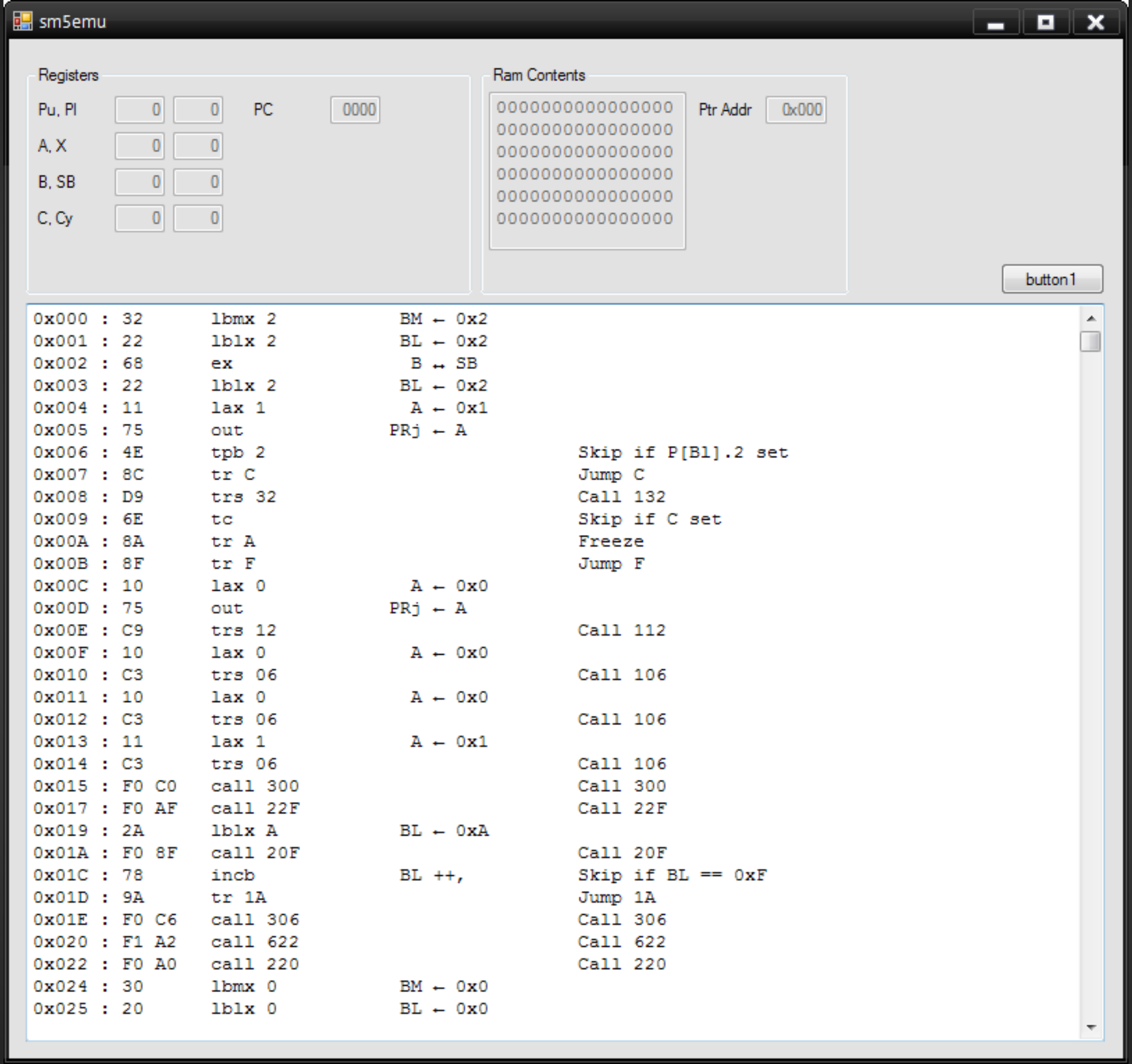
GUI Editor: 2013 marshallh

Improvements






ROM
4,096 BYTE (SM5K6)
8,192 BYTE (SM5K7)
RAM
256 x 4 BIT (SM5K6)
512 x 4 BIT (SM5K7)
Areg
Xreg
C
PC (PU, PL)
SR x 8
B (BU* BM, BL)
SB
ALU
SELECTOR
PULSE OUTPUT (P23)
OSCIN
OSCOUT
OSC
DIVIDER
SELECTOR
TIMER 1
TIMER 2
IFT1
IFT2
IINTERRUPT CONTROLLER
STANDBY CONTROLLER
IFS
SERIAL INTERFACE
A/D CONVERTER
IFA
IFB
P20
P21
P22
P40/P42
RESET
HARDWARE RESET CIRCUIT
VDD
GND
P0
P1
P2
P3
P4
P5
P00 P01 P02 P03
P10 P11 P12 P13
P20 P21 P22 P23
VR
GND
P53 P52 P51 P50
P43 P42 P41 P40
P33 P32 P31 P30



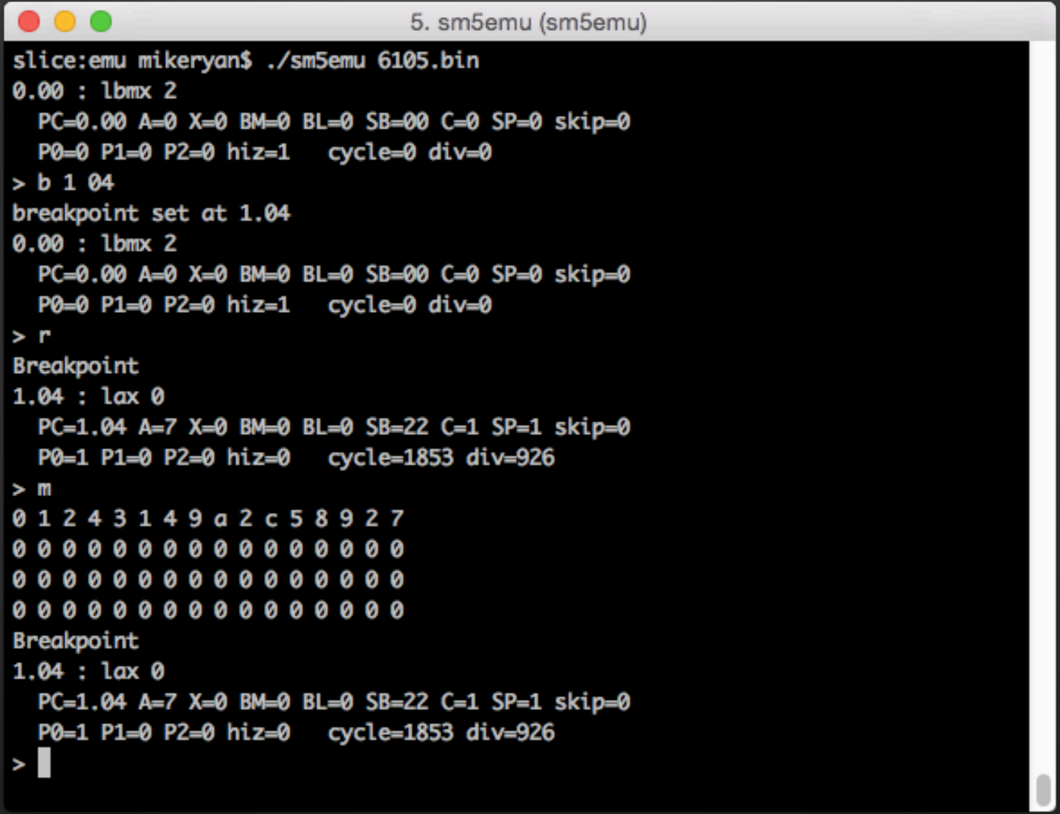
```
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
```

```
0x000 : 32      lbmx 2          BM ← 0x2
0x001 : 22      lblx 2          BL ← 0x2
0x002 : 68      ex               B ↔ SB
0x003 : 22      lblx 2          BL ← 0x2
0x004 : 11      lax 1            A ← 0x1
0x005 : 75      out            PRj ← A
0x006 : 4E      tpb 2                            Skip if P[Bl].2 set
0x007 : 8C      tr C                             Jump C
0x008 : D9      trs 32                           Call 132
0x009 : 6E      tc                               Skip if C set
0x00A : 8A      tr A                             Freeze
0x00B : 8F      tr F                             Jump F
0x00C : 10      lax 0            A ← 0x0
0x00D : 75      out            PRj ← A
0x00E : C9      trs 12                           Call 112
0x00F : 10      lax 0            A ← 0x0
0x010 : C3      trs 06                           Call 106
0x011 : 10      lax 0            A ← 0x0
0x012 : C3      trs 06                           Call 106
0x013 : 11      lax 1            A ← 0x1
0x014 : C3      trs 06                           Call 106
0x015 : F0 C0   call 300                         Call 300
0x017 : F0 AF   call 22F                         Call 22F
0x019 : 2A      lblx A          BL ← 0xA
0x01A : F0 8F   call 20F                         Call 20F
0x01C : 78      incb            BL ++,           Skip if BL == 0xF
0x01D : 9A      tr 1A                            Jump 1A
0x01E : F0 C6   call 306                         Call 306
0x020 : F1 A2   call 622                         Call 622
0x022 : F0 A0   call 220                         Call 220
0x024 : 30      lbmx 0          BM ← 0x0
0x025 : 20      lblx 0          BL ← 0x0
```

Data Transfer Instructions

LAX x	10 to 1F	$A_{CC} \leftarrow x\ (I_3\text{-}I_0)$
LBMX x	30 to 2F	$B_M \leftarrow x\ (I_3\text{-}I_0)$
LBLX x	20 to 2F	$B_L \leftarrow x\ (I_3\text{-}I_0)$
LDA x	50 to 53	$A_{CC} \leftarrow M$ $B_{Mi} \leftarrow B_{Mi} \oplus x\ (I_1, I_0)\ (i = 1, 0)$
EXC x	54 to 57	$M \leftrightarrow A_{CC}$ $B_{Mi} \leftarrow B_{Mi} \oplus x\ (I_1, I_0)\ (i = 1, 0)$
EXCI x	58 to 5B	$M \leftrightarrow A_{CC},\ B_L \leftarrow B_L + 1$ $B_{Mi} \leftarrow B_{Mi} \oplus x\ (I_1, I_0)\ (i = 1, 0)$ Skip if $Cy = 1\ (B_L = 0F_H \rightarrow 0)$
EXCD x	5C to 5F	$M \leftrightarrow A_{CC},\ B_L \leftarrow B_L + 0F_H$ $B_{Mi} \leftarrow B_{Mi} \oplus x\ (I_1, I_0)\ (i = 1, 0)$ Skip if $Cy = 1\ (B_L = 0 \rightarrow 0F_H)$
EXAX	64	$A_{CC} \leftrightarrow X$
ATX	65	$x \leftarrow A_{CC}$

SM5 Emulator



5. sm5emu (sm5emu)

```
slice:emu mikeryan$ ./sm5emu 6105.bin
0.00 : lbmx 2
  PC=0.00 A=0 X=0 BM=0 BL=0 SB=00 C=0 SP=0 skip=0
  P0=0 P1=0 P2=0 hiz=1   cycle=0 div=0
> b 1 04
breakpoint set at 1.04
0.00 : lbmx 2
  PC=0.00 A=0 X=0 BM=0 BL=0 SB=00 C=0 SP=0 skip=0
  P0=0 P1=0 P2=0 hiz=1   cycle=0 div=0
> r
Breakpoint
1.04 : lax 0
  PC=1.04 A=7 X=0 BM=0 BL=0 SB=22 C=1 SP=1 skip=0
  P0=1 P1=0 P2=0 hiz=0   cycle=1853 div=926
> m
0 1 2 4 3 1 4 9 a 2 c 5 8 9 2 7
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Breakpoint
1.04 : lax 0
  PC=1.04 A=7 X=0 BM=0 BL=0 SB=22 C=1 SP=1 skip=0
  P0=1 P1=0 P2=0 hiz=0   cycle=1853 div=926
> 
```

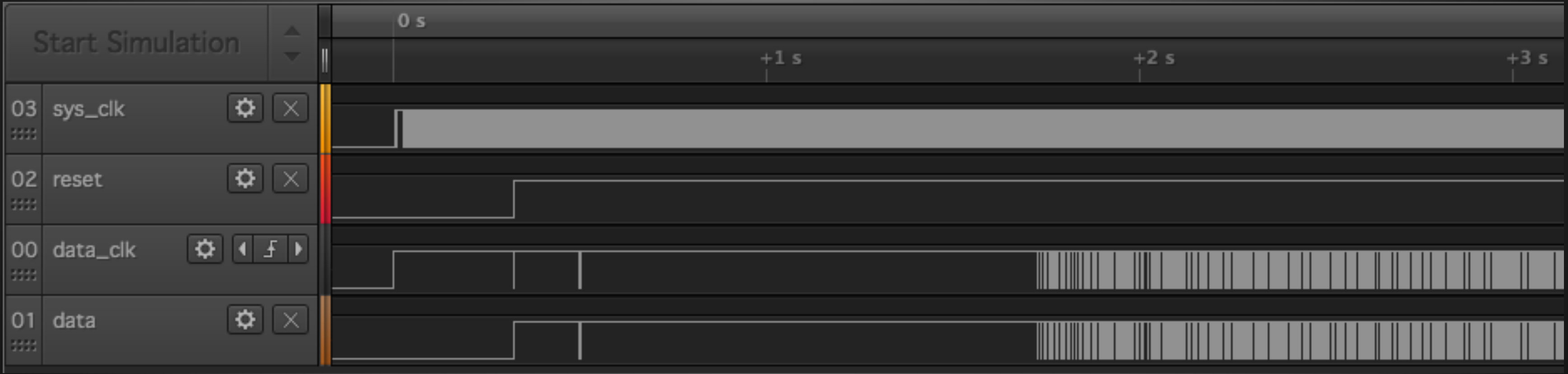
The CIC sends...

1. Hello + region ID
2. Seed
3. Checksum

And if the PIF is happy

- PIF → CIC: 2 nibbles
- Main runtime

Overview

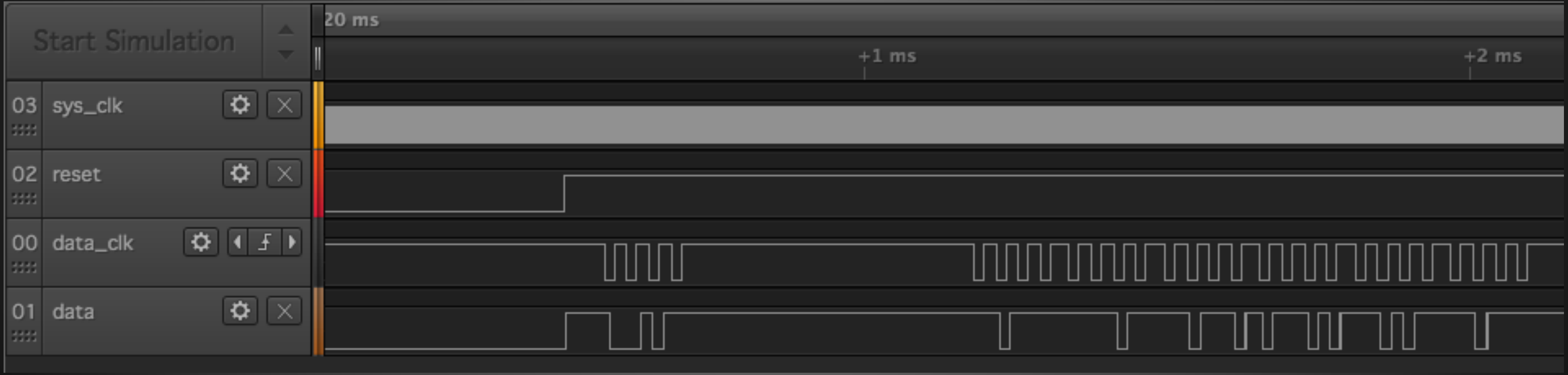


Boot

Checksum and RAM load

Runtime

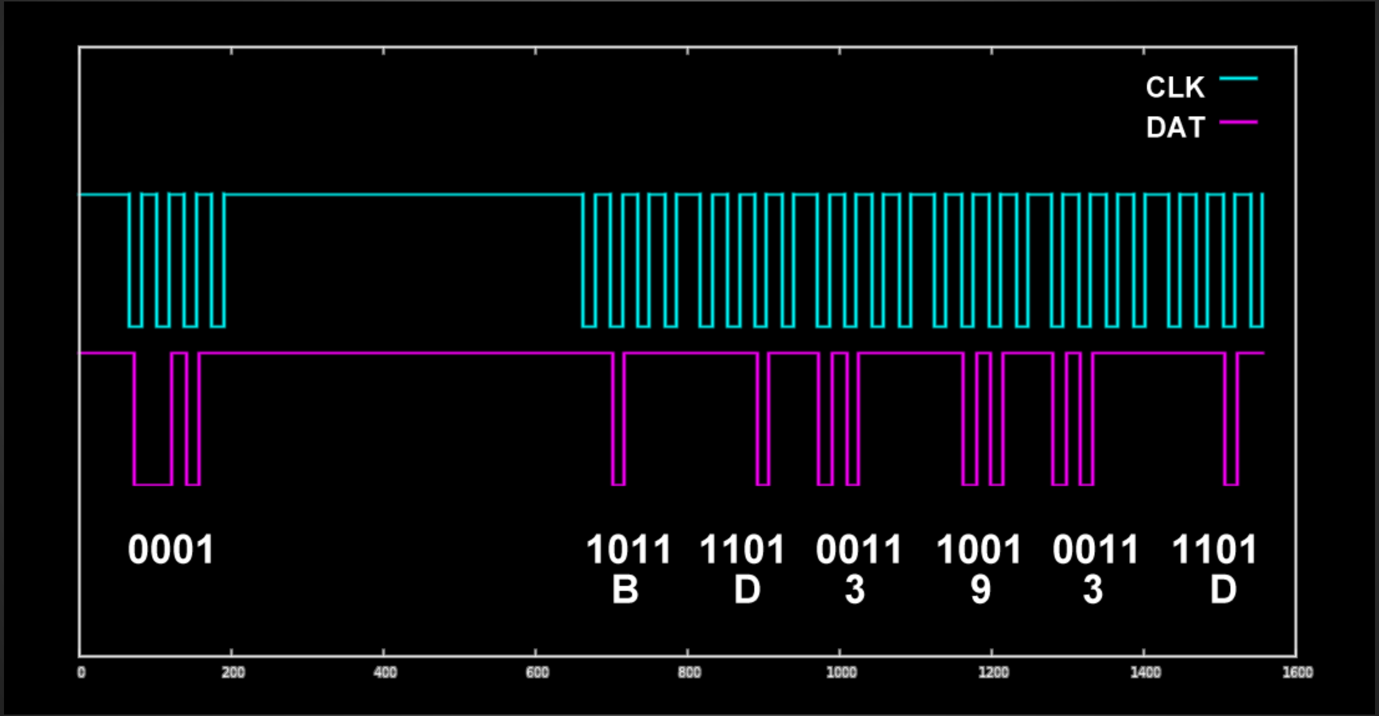
Boot



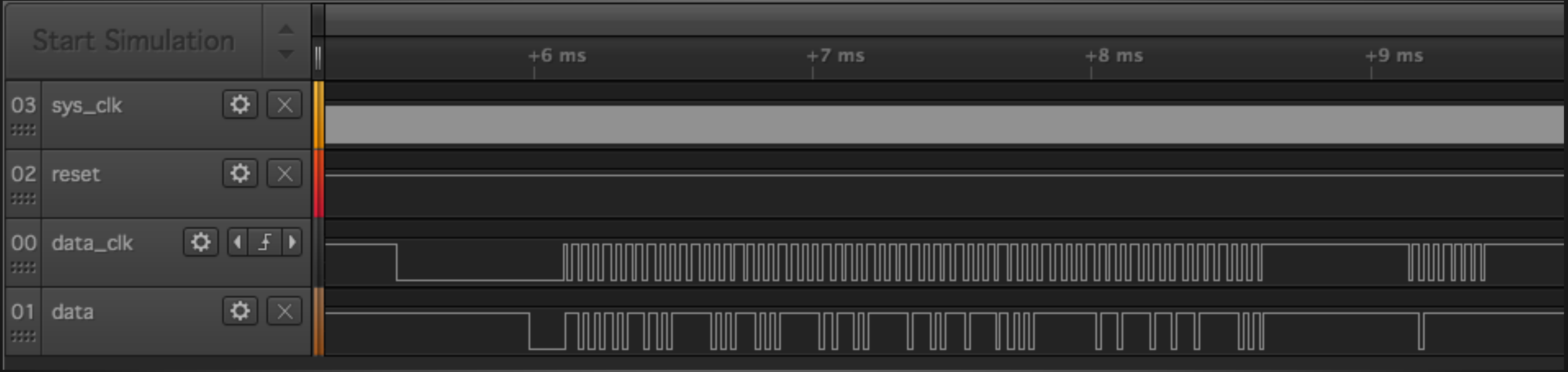
Hello / Region ID

Encoded seed

Boot Detail


CLK
DAT
0001
1011
B
1101
D
0011
3
1001
9
0011
3
1101
D
0
200
400
600
800
1000
1200
1400
1600

Checksum and RAM Load

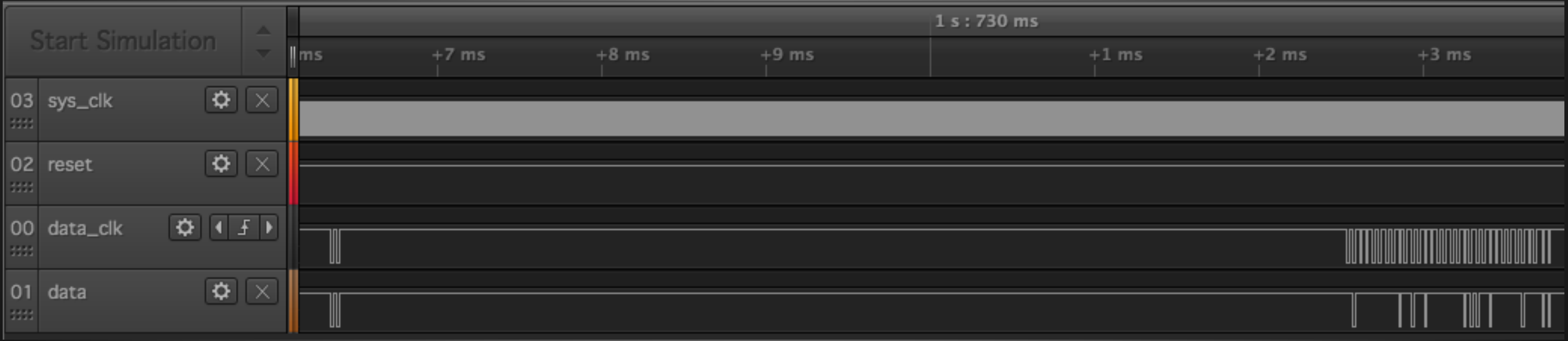


Encoding delay

Encoded checksum

Two nibbles PIF → CIC

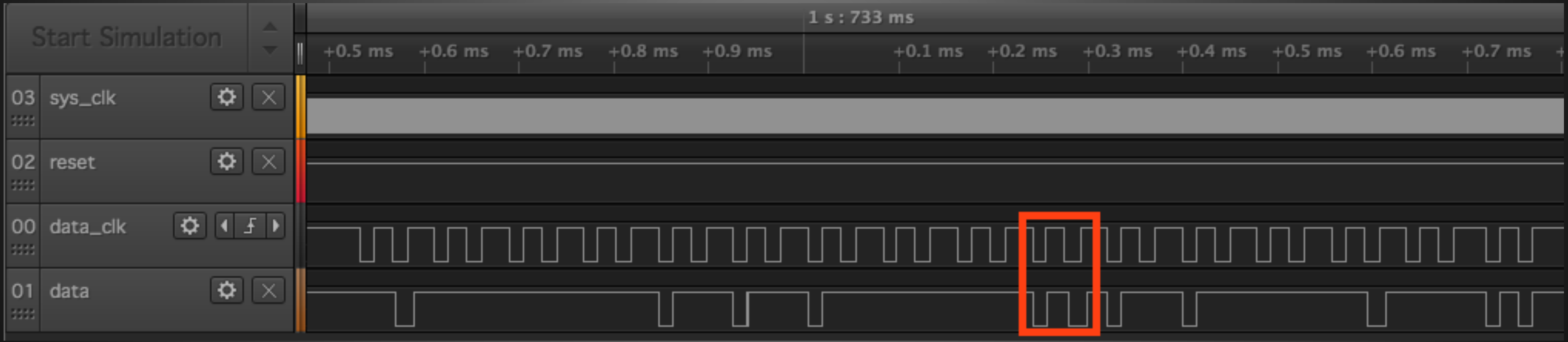
Runtime



00: memory compare mode

memory compare

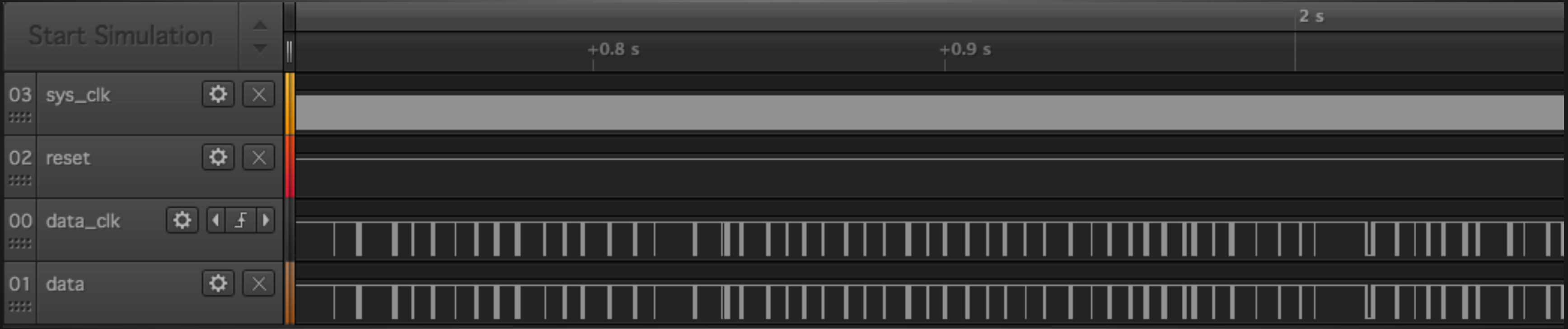
Memory Compare Detail



delay when CIC sends 0

00: memory compare mode

The Party Never Ends



Main Algorithm

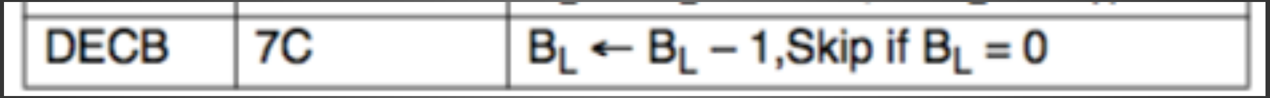
```
void cic_round(uint4_t m[16]) {
    uint4_t a, b, x;
    x = m[15];
    do {
        m[1] += x + 1;
        b = 6;
        if (15 - m[3] > m[2])
            b += 1;

        m[b - 3] += m[3];
        m[3] += m[2] + 1;
        m[2] = ~(m[2] + m[1] + 1);
        a = m[b - 1];
        if (m[b - 2] > 7)
            m[b - 1] = 0;
        m[b - 1] += m[b + 2] + 8;
        m[b - 2] += m[b - 3];
        do {
            a += m[b] + 1;
            m[b] = a;
            b += 1;
        } while (b != 0);
        x -= 1;
    } while (x != 15);
}
```

Credit: Kammerstetter, M. et al

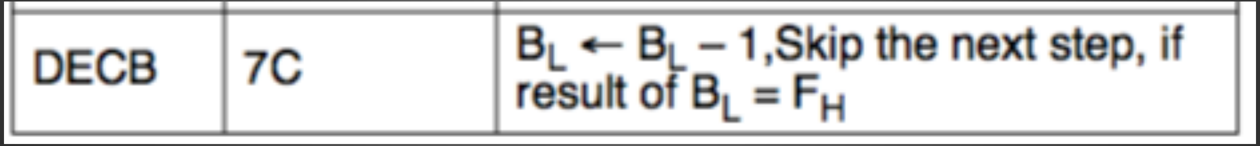
6105 Algorithm

```
uint8_t A = 5; int carry = 1;
for (int i = 0; i < 14; ++i) {
    if (!(mem[i] & 1)) A += 8;
    if (!(A & 2)) A += 4;
    A = (A + mem[i]) & 0xf;
    mem[i] = A;
    if (!carry)
        A += 7;
    A = (A + mem[i]) & 0xF;
    A = A + mem[i] + carry;
    if (A >= 0x10) {
        carry = 1;
        A -= 0x10;
    } else {
        carry = 0;
    }
    A = (~A) & 0xf;
    mem[i] = A;
}
```

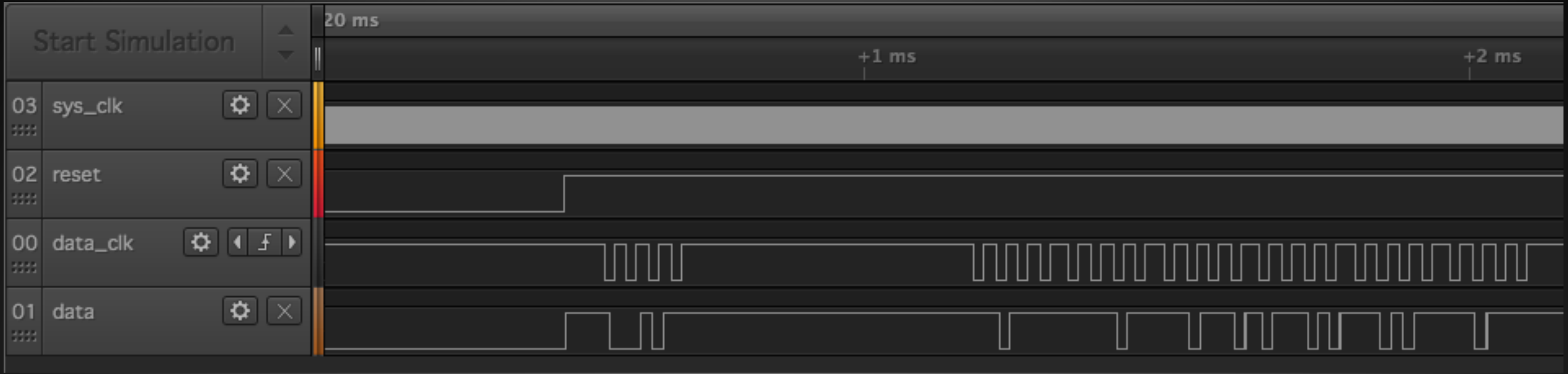
A Few More Hangups



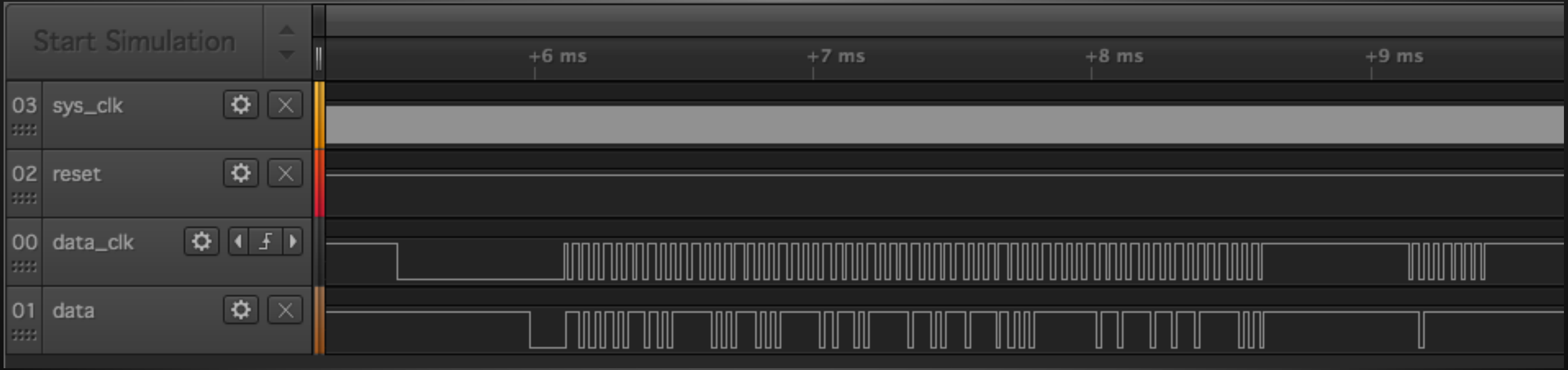
DECB	7C	$B_L \leftarrow B_L - 1$,Skip if $B_L = 0$



DECB	7C	$B_L \leftarrow B_L - 1$,Skip the next step, if result of $B_L = F_H$



Encoded seed



Encoded checksum

Encoding Algorithm

```
void fn_22b(uint8_t *mem, int start) {
    int i; uint8_t A;
    A = mem[start];
    for (i = start+1; i < 16; ++i) {
        A = (A + 1) % 16;
        A = (A + mem[i]) % 16;
        mem[i] = A;
    }
}
```

Inverse Algorithm

```
void inverse_22b(u8 *mem, int start) {
    int i; uint8_t A, nextA;
    A = mem[start];
    nextA = A;
    for (i = start+1; i < 16; ++i) {
        nextA = mem[i];
        mem[i] -= (A + 1);
        if (mem[i] > 16) mem[i] += 16;
        A = nextA;
    }
}
```

Decoding Seed

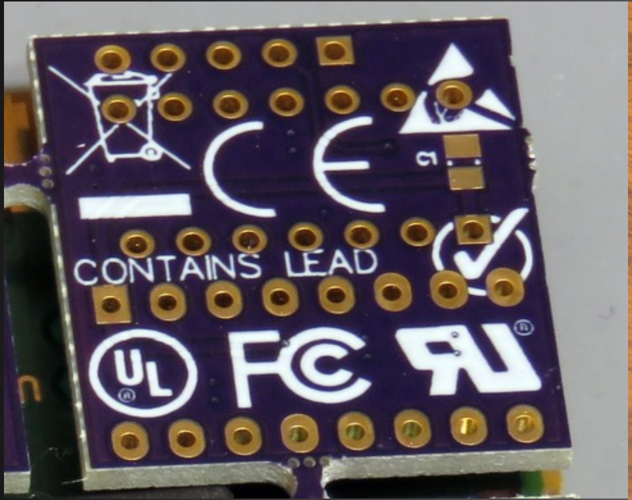
- B D 3 9 3 D → B 5 $\underline{\text{3 F 3 F}}$
- B D 7 A 5 9 → B 5 $\underline{\text{7 8 7 8}}$

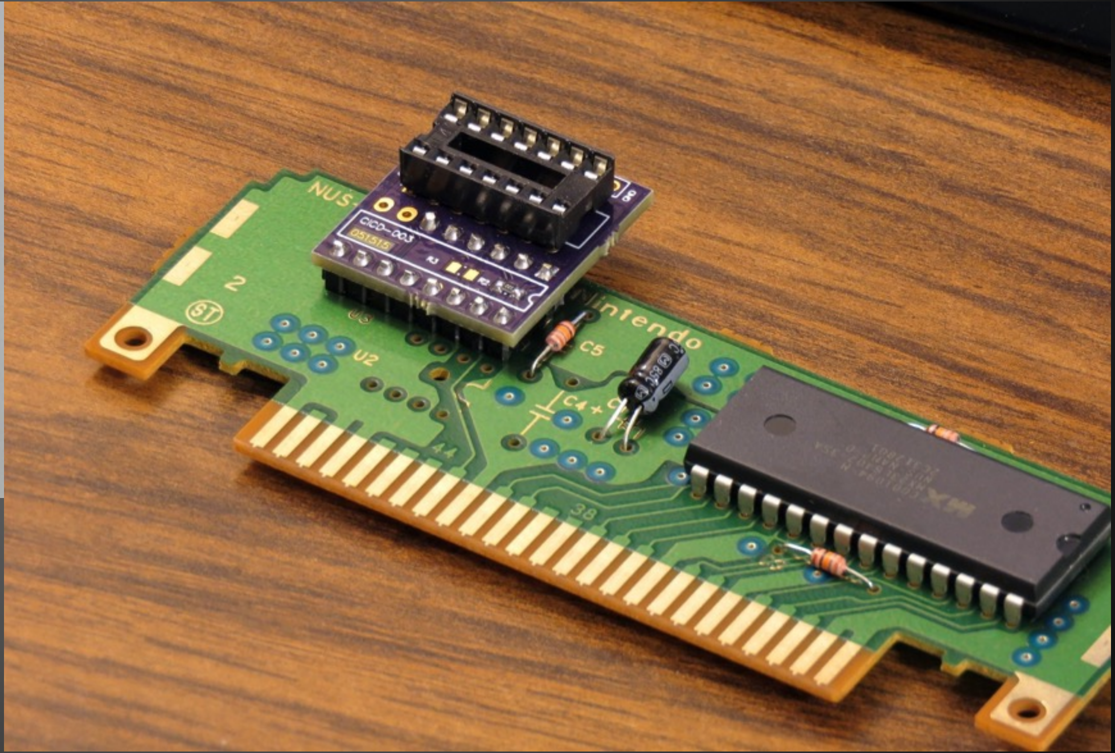
- Encryption “key” is always B 5

Decoding Checksum

- 3 F 5 7 2 9 3 E 5 4 7 C F 5 9 0 → 3 F D F $\underline{\text{A 5 3 6 C 0 F 1 D 8 5 9}}$
- “Key” varies based on delay from PIF
- RC pseudo-random delay

Working implementation





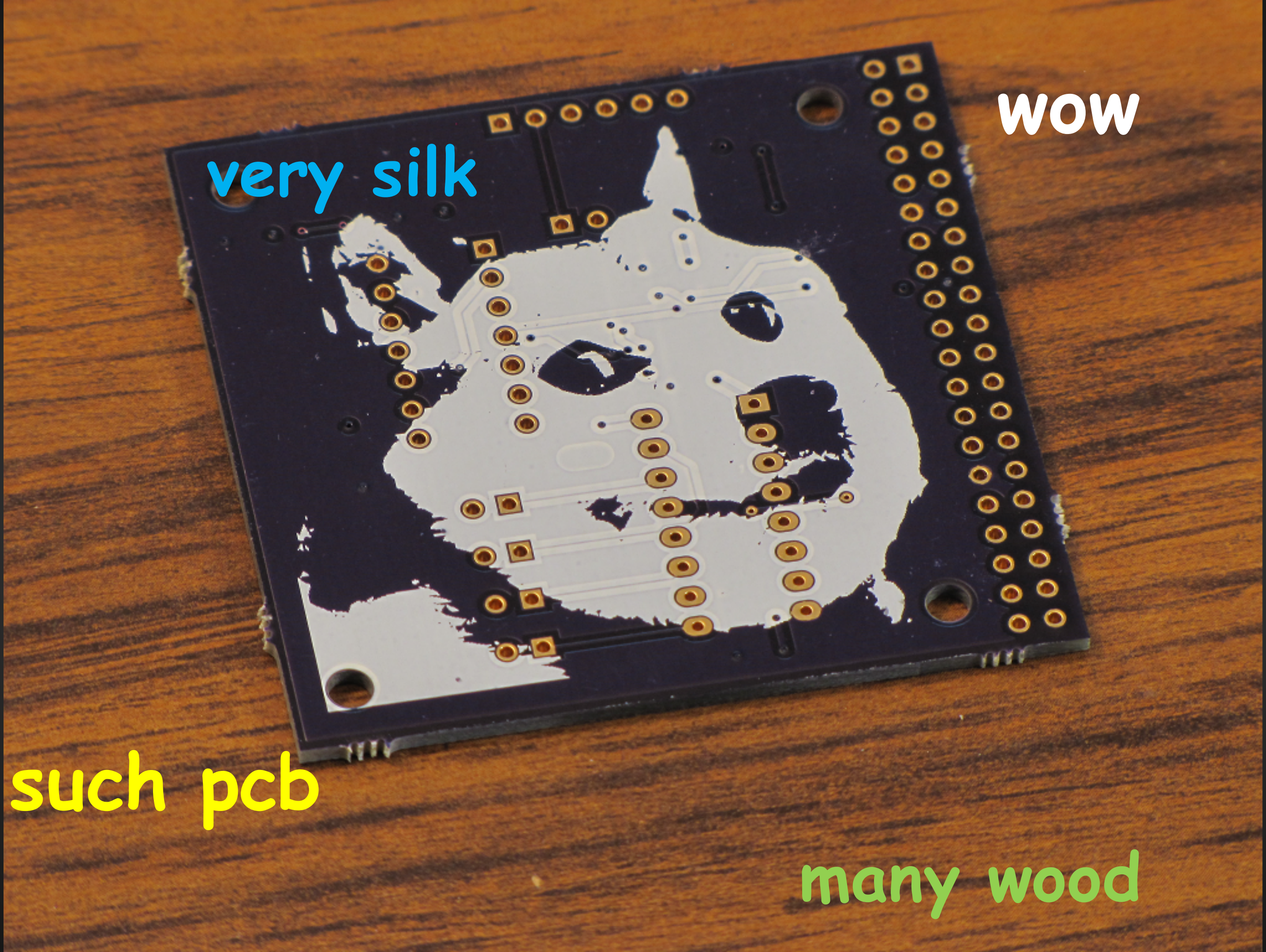
Video

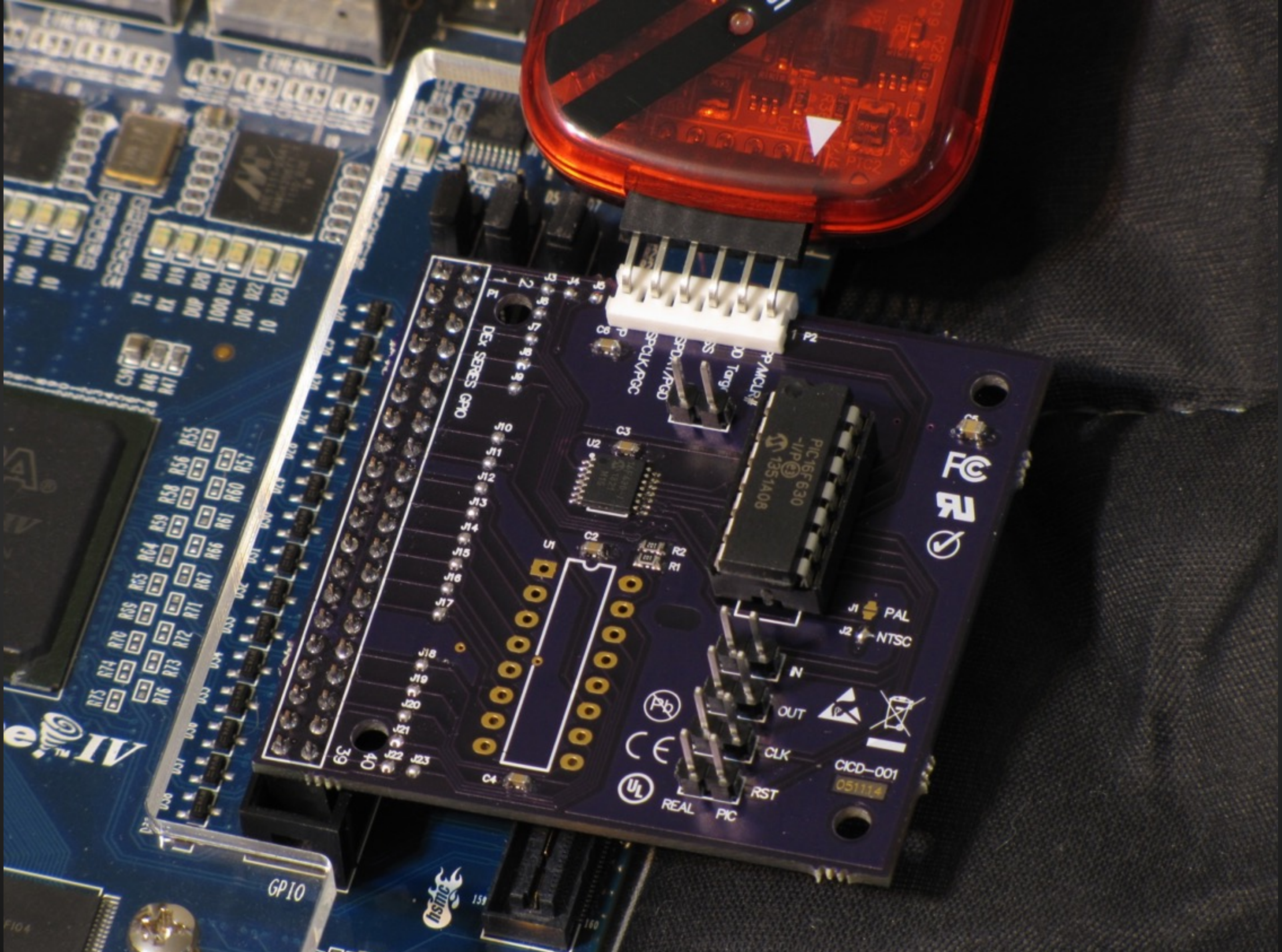
Features

- Region free
- Barebones version is open source
- PIC PROBLEMS

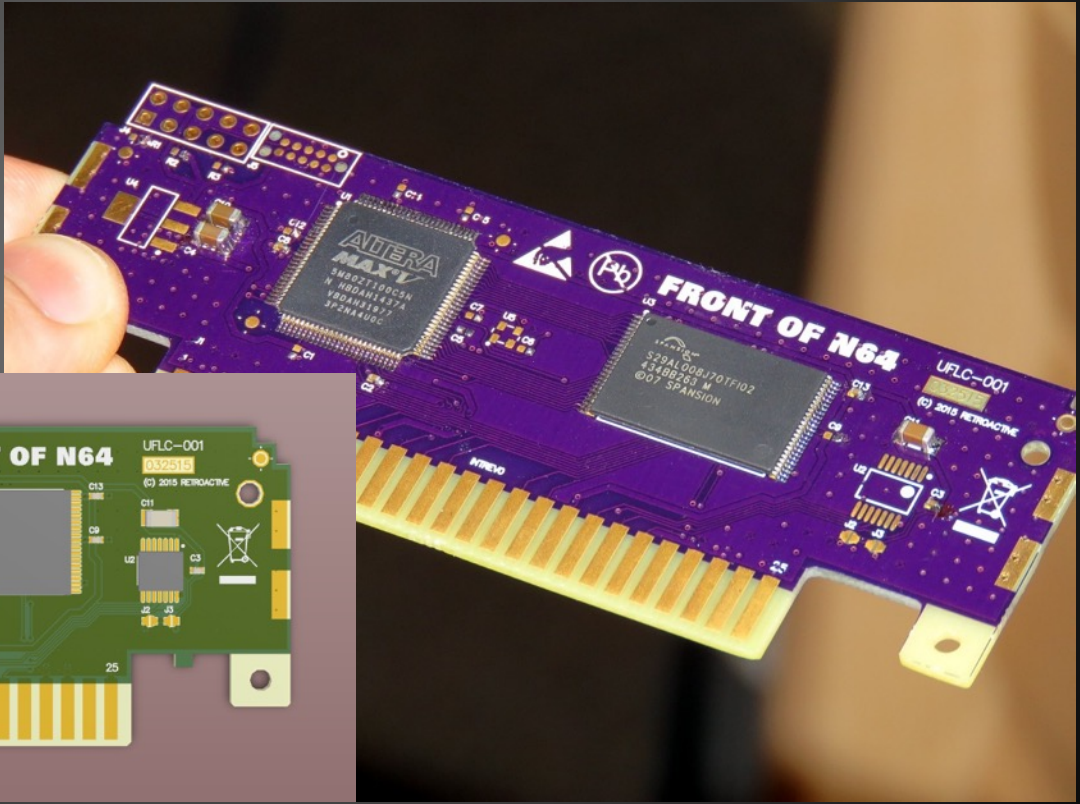
Related Work

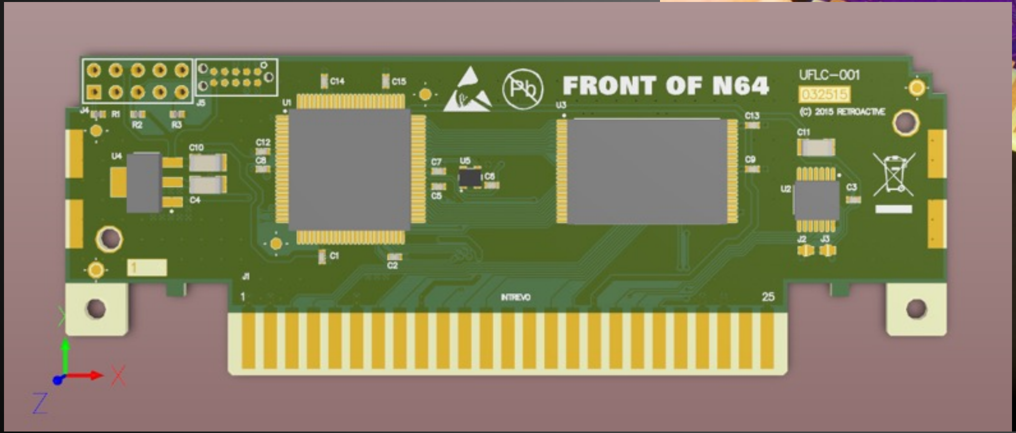
- Breaking Integrated Circuit Device Security through Test Mode Silicon Reverse Engineering. Kammerstetter, Markus et al.


wow
very silk
such pcb
many wood

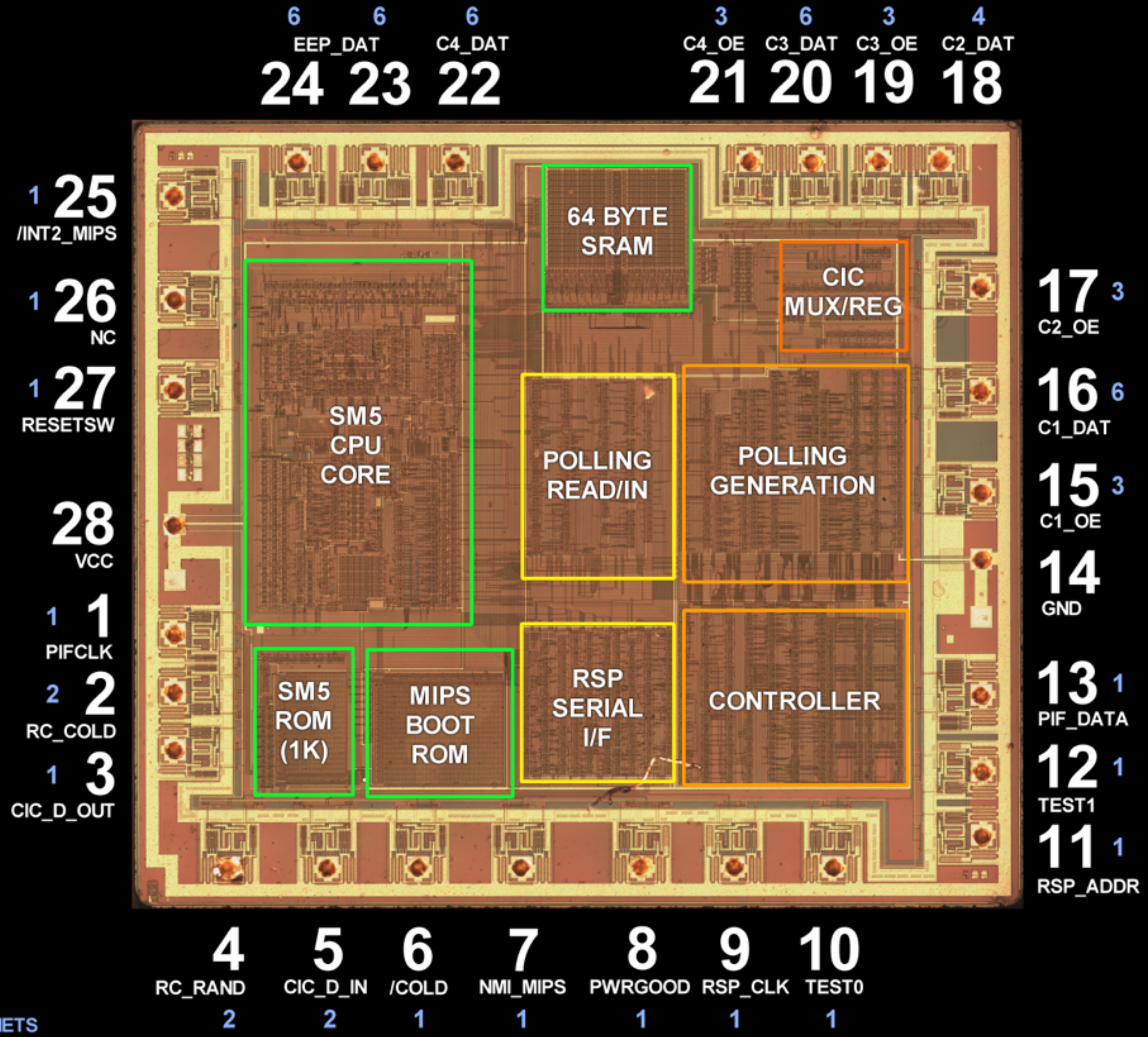

DEx SERIES GPIO
PIC16F630
CICD-001
PAL
NTSC
IN
OUT
CLK
RST
REAL
PIC
GPIO

Working implementation

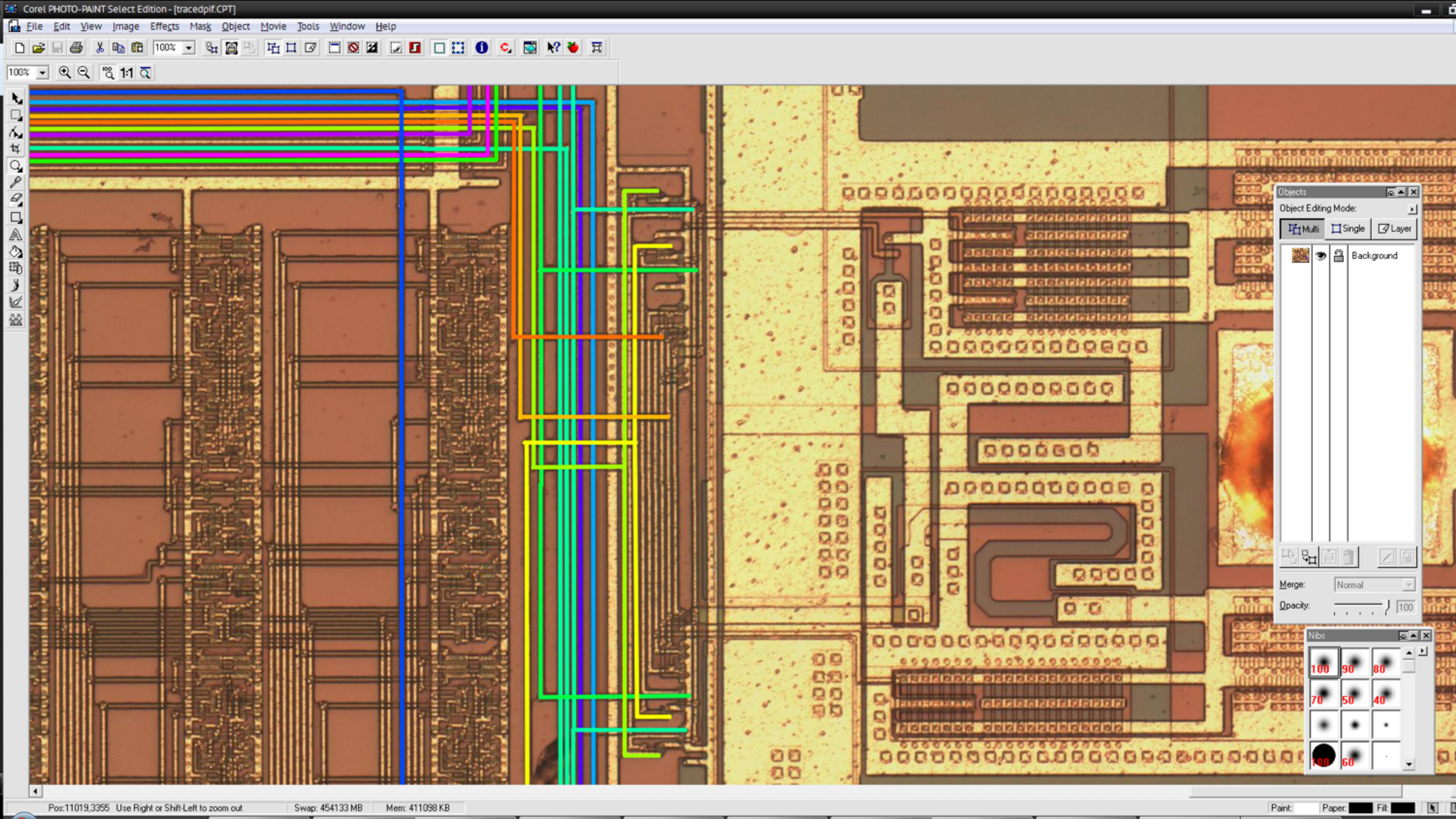




Future work



6 6 6 3 6 3 4
EEP_DAT C4_DAT C4_OE C3_DAT C3_OE C2_DAT
24 23 22 21 20 19 18
1 25 /INT2_MIPS
1 26 NC
1 27 RESETSW
28 VCC
1 1 PIFCLK
2 2 RC_COLD
1 3 CIC_D_OUT
17 3 C2_OE
16 6 C1_DAT
15 3 C1_OE
14 GND
13 1 PIF_DATA
12 1 TEST1
11 1 RSP_ADDR
4 RC_RAND 2
5 CIC_D_IN 2
6 /COLD 1
7 NMI_MIPS 1
8 PWRGOOD 1
9 RSP_CLK 1
10 TEST0 1
NETS
64 BYTE SRAM
CIC MUX/REG
SM5 CPU CORE
POLLING READ/IN
POLLING GENERATION
SM5 ROM (1K)
MIPS BOOT ROM
RSP SERIAL I/F
CONTROLLER


Corel PHOTO-PAINT Select Edition - [tracedpif.CPT]
File Edit View Image Effects Mask Object Movie Tools Window Help
100%
Objects
Object Editing Mode:
Multi
Single
Layer
Background
Merge:
Normal
Opacity:
100
Nibs
Pos:11019,3355 Use Right or Shift-Left to zoom out
Swap: 454133 MB
Mem: 411098 KB
Paint:
Paper:
Fill:

Thanks

- Markus + Markus
- Segher
- Zoinkity
- emu_kidid
- arbin
- jrra

Greetz

- #n64dev
- azonenberg

?

THANKS !!!!!!!!!!!!!!!!!!!!!!!!

- Mike – @mpeg4codec
- Marsh – @fpga_nugga
- John – siliconpr0n.org

- https://github.com/mikeryan/UltraCIC
- https://github.com/mikeryan/sm5emu
- https://github.com/JohnDMcMaster/pr0ntools