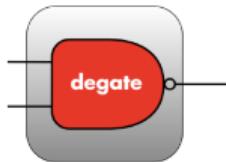


Degate

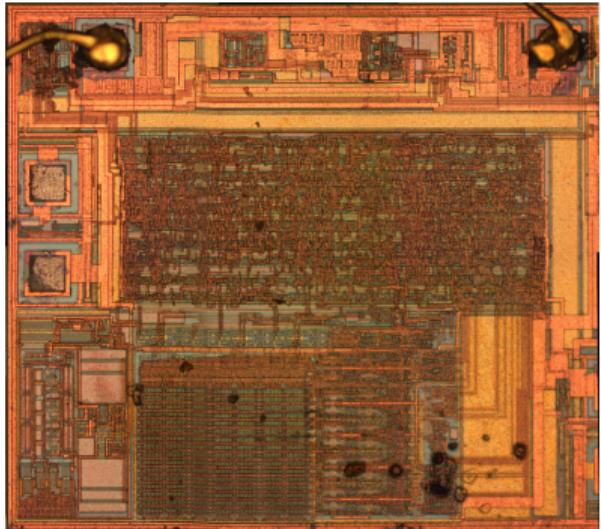
The stakes and challenges of silicon reverse engineering
<https://www.degate.org>

D. Bachelot

SecSea2k24,
October 11th and 12th, 2024



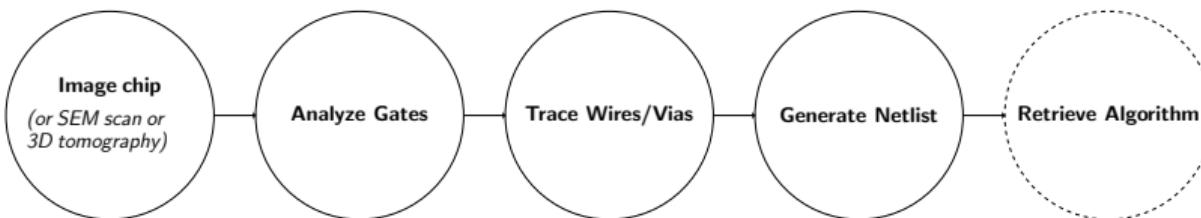
What is Silicon Chips RE?



Same idea than with software RE (from binary, to assembly and to code), silicon chip RE go **from silicon, to images, to transistors, to gates, to netlist and to algorithm**.

With proper preparation and knowledge, we can go into silicon, **analyze transistors, retrieve gates/wires/vias** and **reconstruct implemented algorithms**. This can be used to **analyze old hardware, build software emulators, search for vulnerabilities and backdoors, break/test a protection, secret extraction or check intellectual property**.

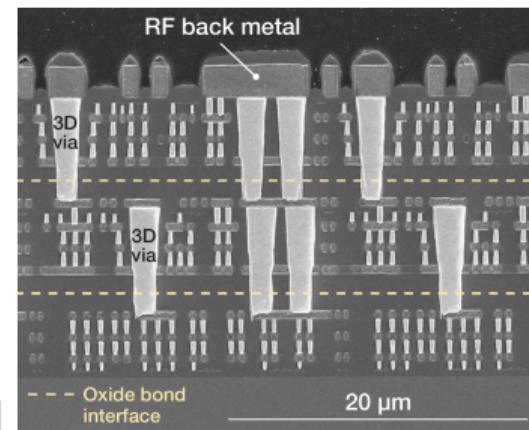
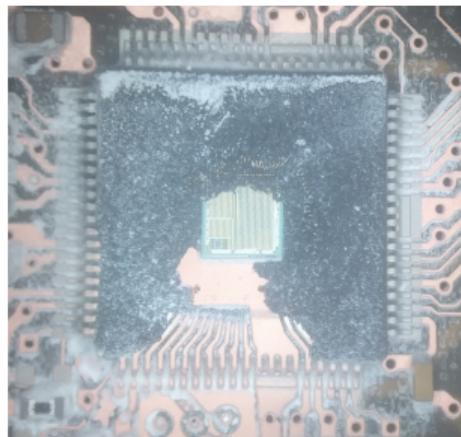
Used in IC industry for fault/failure detection & analysis, but not at the same scale.



How to Access Silicon?

Can be very costly (plasma & laser) and destructive... But also accessible with simpler methods (like chemical/mechanical). More on [4].

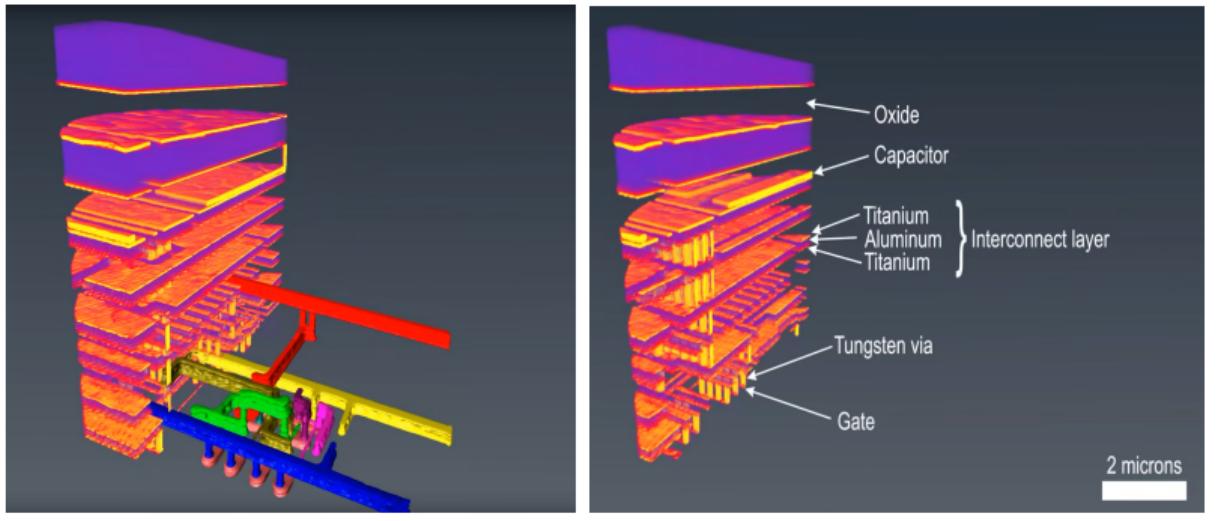
- ① **Decapsulation** (heat, acid, mechanical, plasma, laser...)
- ② **Delayering** (chemical, abrasive, laser, plasma...)
- ③ **Cleaning** (ultrasound, acid...)



How to Retrieve Images?

Using each layer (invasive) or directly using the chip (non-invasive):

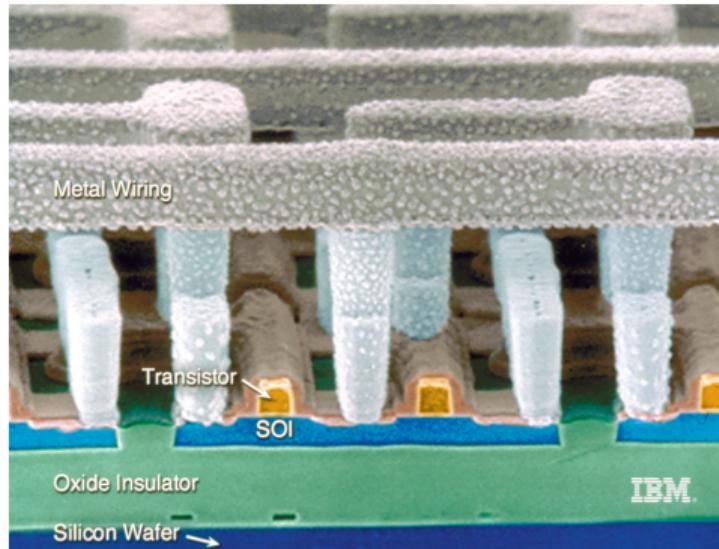
- Take very-high resolution images from **optical microscope** (basic, confocal) ;
- Scan from an **electron microscope** (SEM, TEM...) ;
- Generate a 3D model using **electron tomography** ;



How to Perform the Analysis?

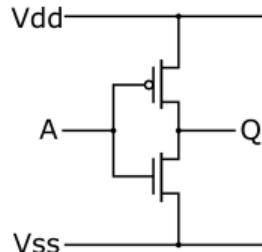
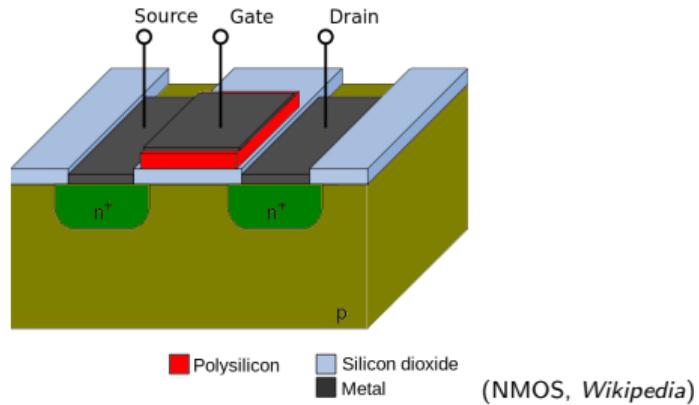
Overview:

- ① Choose a **zone of interest**,
- ② Identify each **gate type**, annotate, and place in a "**gate library**",
- ③ Find other **gates instance** from gate library,
- ④ Link gates by tracing **wires and vias**,
- ⑤ Export to **netlist** (e.g. by translating each gate to VHDL/Verilog code).

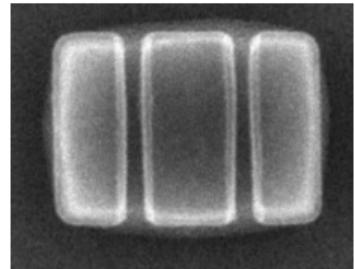


How to identify a transistor?

- ① Search, at transistor layer, for **doped zones**.
- ② Spot the **zebras**.
- ③ Use logic to identify the **type of each transistor** (e.g. PMOS are bigger to compensate with lower hole mobility).
- ④ Search for **wires** (to identify inputs and outputs).



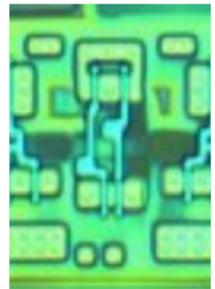
(Inverter, Wikipedia)



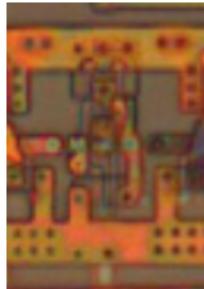
(PMOS [10])



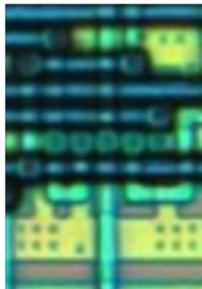
How to Identify a Gate?



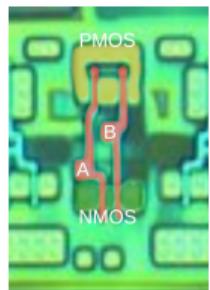
Transistor layer



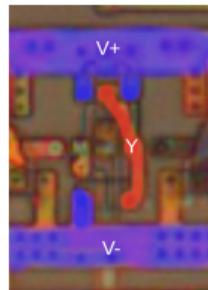
Logic layer



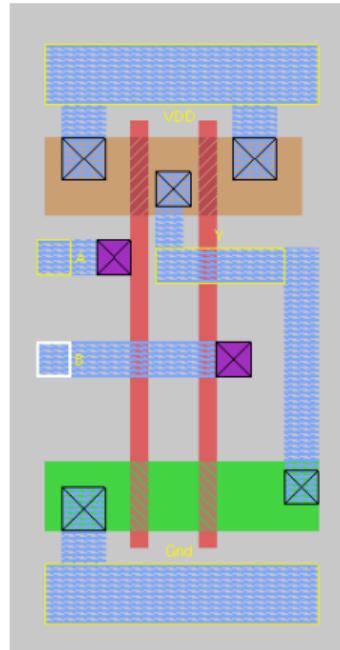
Metal layer



P & N zones and 2 inputs



V+ & V-, and output



[7] ⇒

NAND gate!

A	B	Y
0	0	1
1	0	1
0	1	1
1	1	0



How to Retrieve the Netlist from Analyzed Gates?

```

module jsrflipflop(q,qbar,clk,rst,sr);
    output reg q;
    output qbar;
    input clk, rst;
    input [1:0] sr;

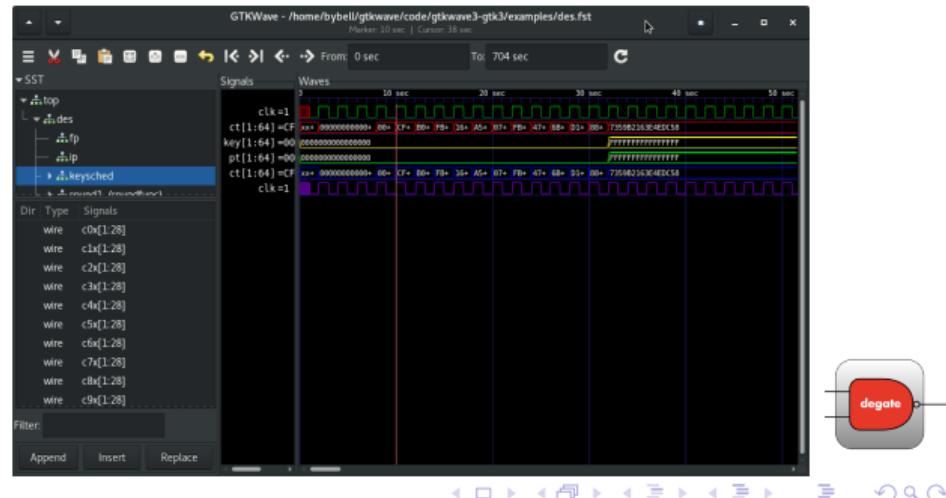
    assign qbar = ~q;

    always @(posedge clk)
    begin
        if (rst)
            q <= 0;
        else
            case(sr)
                2'b00: q <= q;
                2'b01: q <= 0;
                2'b10: q <= 1;
                2'b11: q <= 1'bx;
            endcase
    end
endmodule

```

- Each gate can be described with **hardware description language (HDL)**, like **Verilog** or **VHDL**.
- Wires & vias** can also be described.
- That's all we need to **obtain the netlist**!

We can, from HDL, **simulate the extracted netlist** and find **incoherence** (*example with gtkwave below*):



How to Get the Algorithm/Specification from Netlist? [3]

After retrieving the **netlist**, we are left with a **huge and "unorganized" number of gates**. The **specification discovery** phase aims to **retrieve IC's algorithm/functionality** from the extracted netlist.

Using specific algorithms you can **automate some phase**:

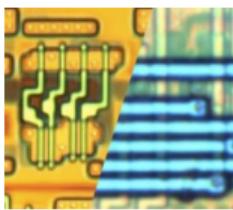
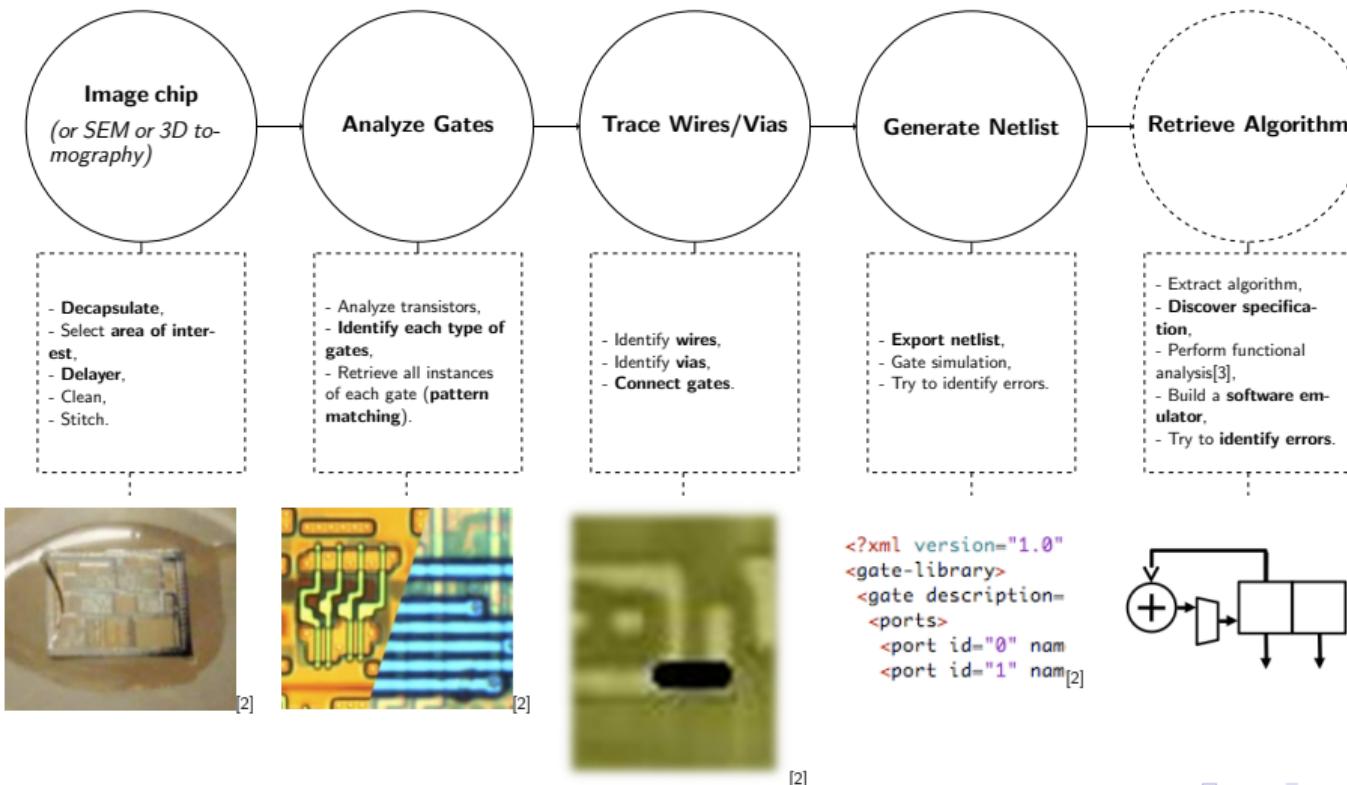
- **Partitioning** of the netlist (*to retrieve a semblance of "code" structure*).
- **Recovery** of the registers (*if applicable*).
- **Identification** of the extracted "groups" (*partitions*) of the netlist.
- **Construction** of a library of netlist components from the identified "groups".

These algorithms **need to allow some degrees of error** from the netlist extraction. This phase is ~analogous with **duplicated, standard & library functions identification** for **software engineering**. A nice open source tool for this is **HAL**³ (compatible with Degate's outputs!).

³<https://github.com/emsec/hal>

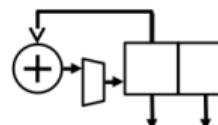


To Summarize



```

<?xml version="1.0" 
<gate-library>
  <gate description=
    <ports>
      <port id="0" nam
      <port id="1" nam[2]
    
```



Importance for Cybersecurity

How can we **trust software** if we **can't trust hardware** (e.g. "specialized" ASIC)?

- Is there any **vulnerability in the hardware implementation** of an algorithm (e.g. crypto standard with predictable initialization, bad randomness...)?
- Is there any **hardware Trojan** (e.g. placed by the foundry)?
- If there is a vulnerability/backdoor, **patching is impossible**, far **more impactful** than software vulnerabilities.

Some examples of vulnerabilities found thanks to silicon RE:

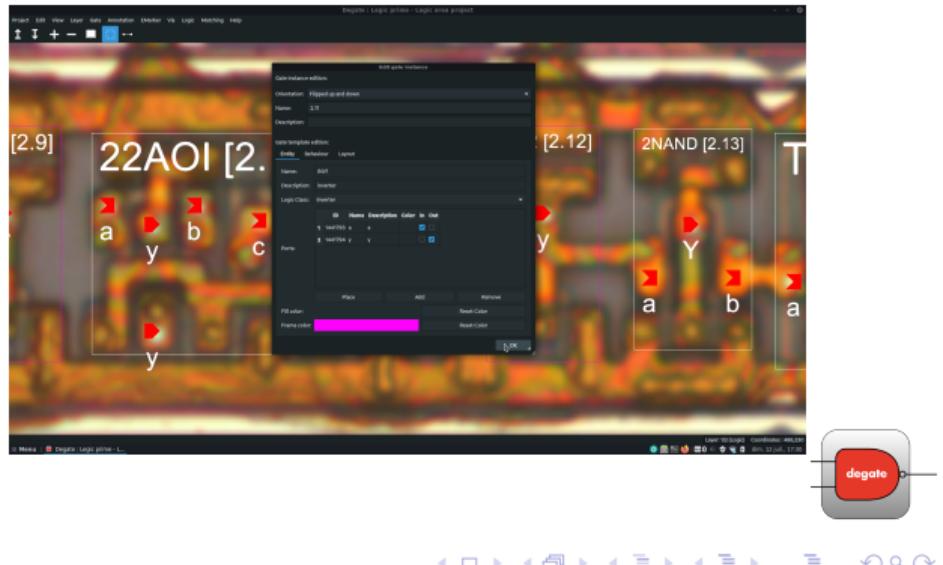
- *Legic Prime, NXP Hitag2, DECT DSC, CryptoRF, Atmel CryptoMemory & NXP Mifare Crypto-1 (~2008, Nohl et al)*: **weak (or potentially weak) cryptographic ciphers**.
- Undisclosed ones?



Introduction

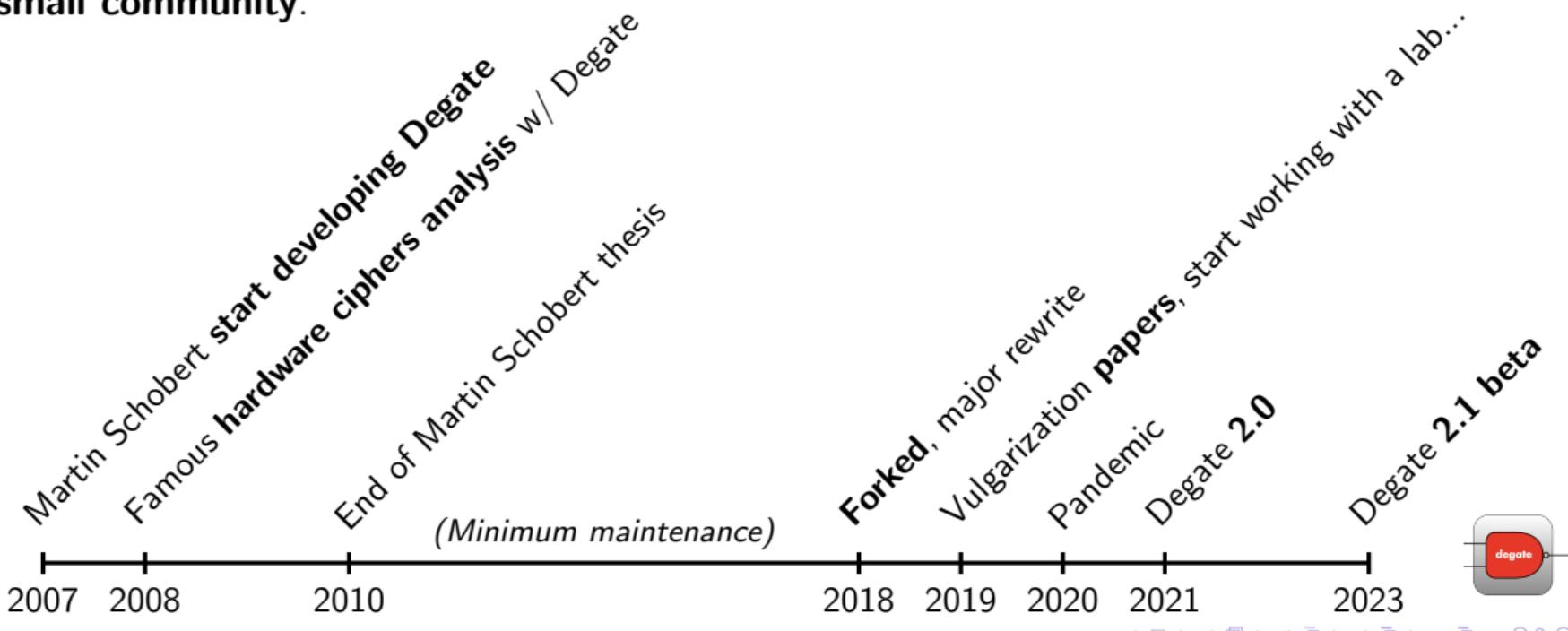
Dlegate is a multi-platform software for semi-automatic **Very-Large-Scale Integration (VLSI) chips reverse engineering** of digital logic in chips.

- ~70k LoC
- Supports Mac, Linux & Windows,
- Qt based,
- Multi-language support,
- Gate definition,
- Gate template, via & wire matching,
- Rule checks,
- ...



History

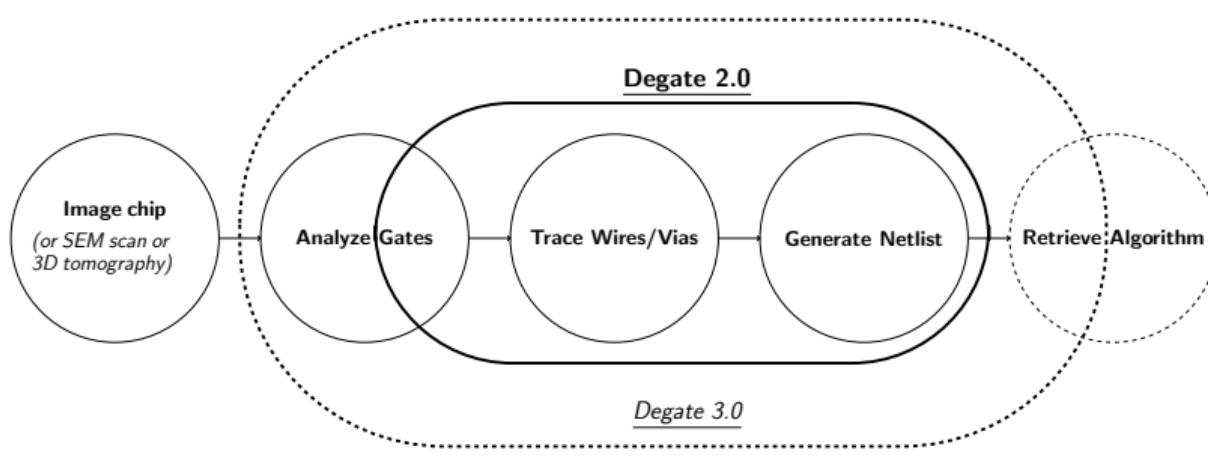
A long story, with **technical debt** and **major IC evolution** (in transistor count), along with a **small community**.



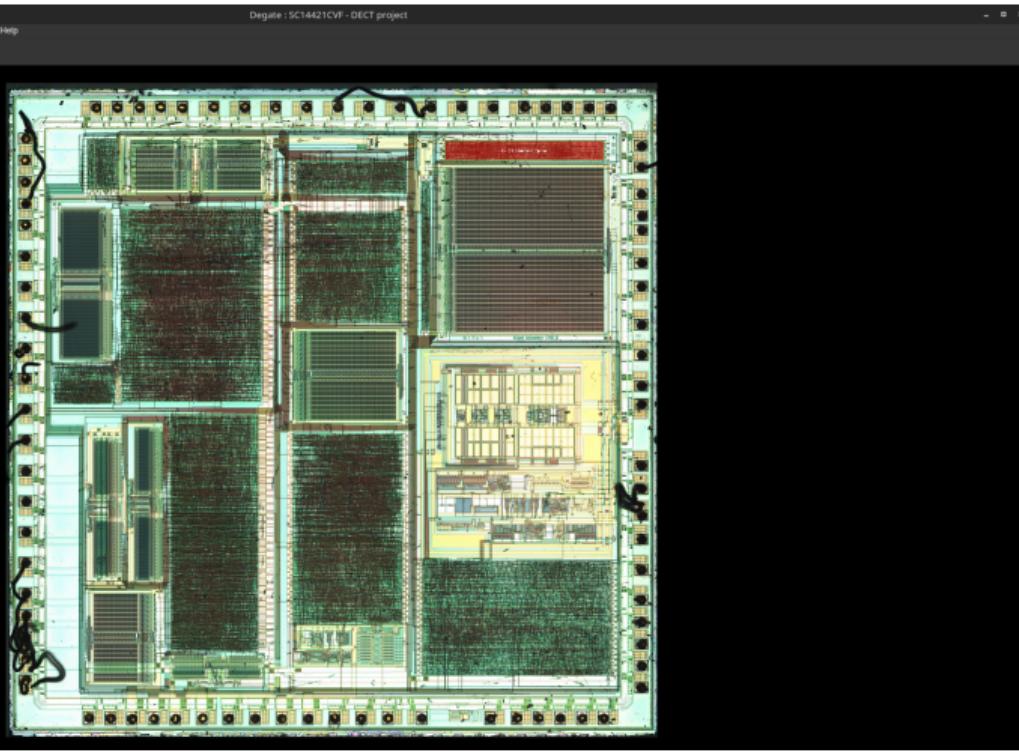
Usage

Degate help to reverse **VLSI chips** by creating an analyzed **gate library**, doing **template matching** to find gates instances from this library, **matching wires & vias**, **exporting netlist** and **navigating really huge images**.

Focus on **modern ICs** with **standard cells**, and supports **any 2D capture/imaging method** (SEM, optical...).



Small Demonstration

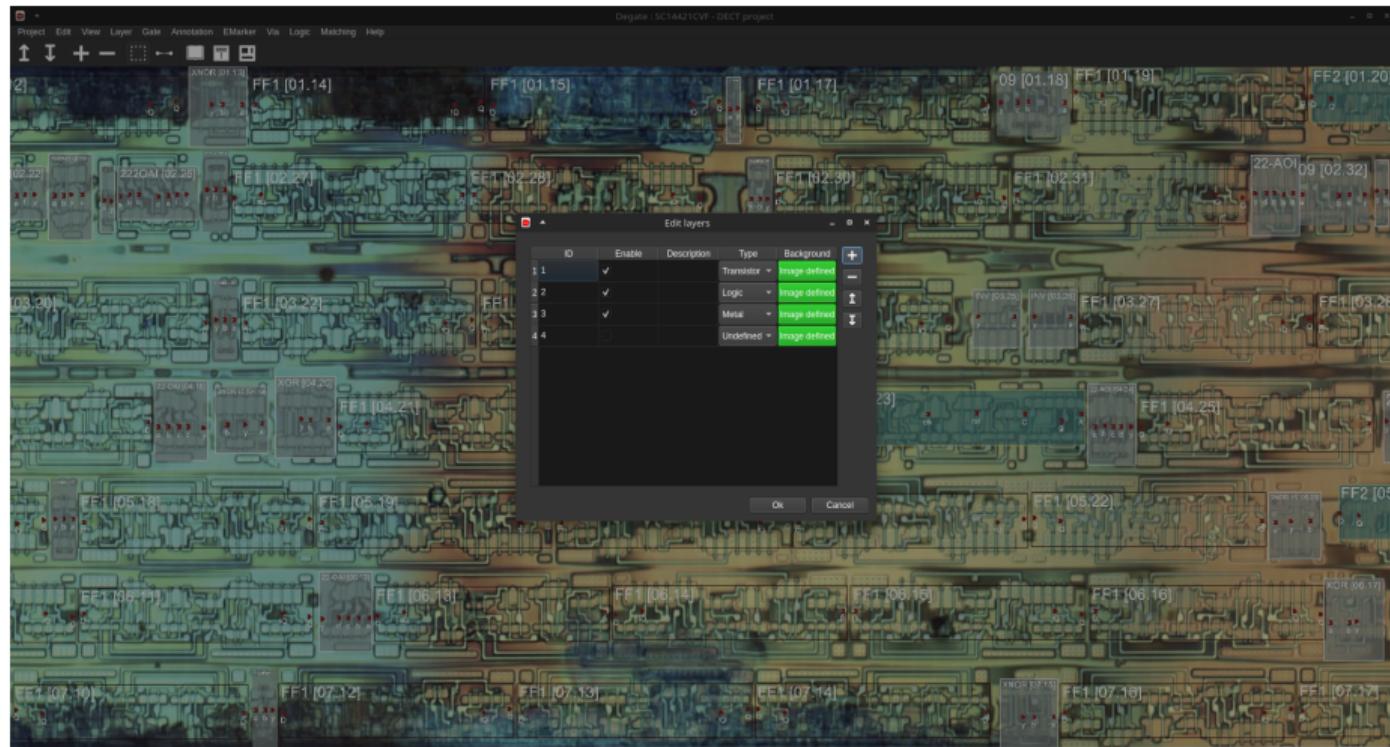


Overview of the chip, for zone of interest selection.

A sub-project can then be created on the zone of interest, and specific layers can be added (independent from the rest).



Small Demonstration

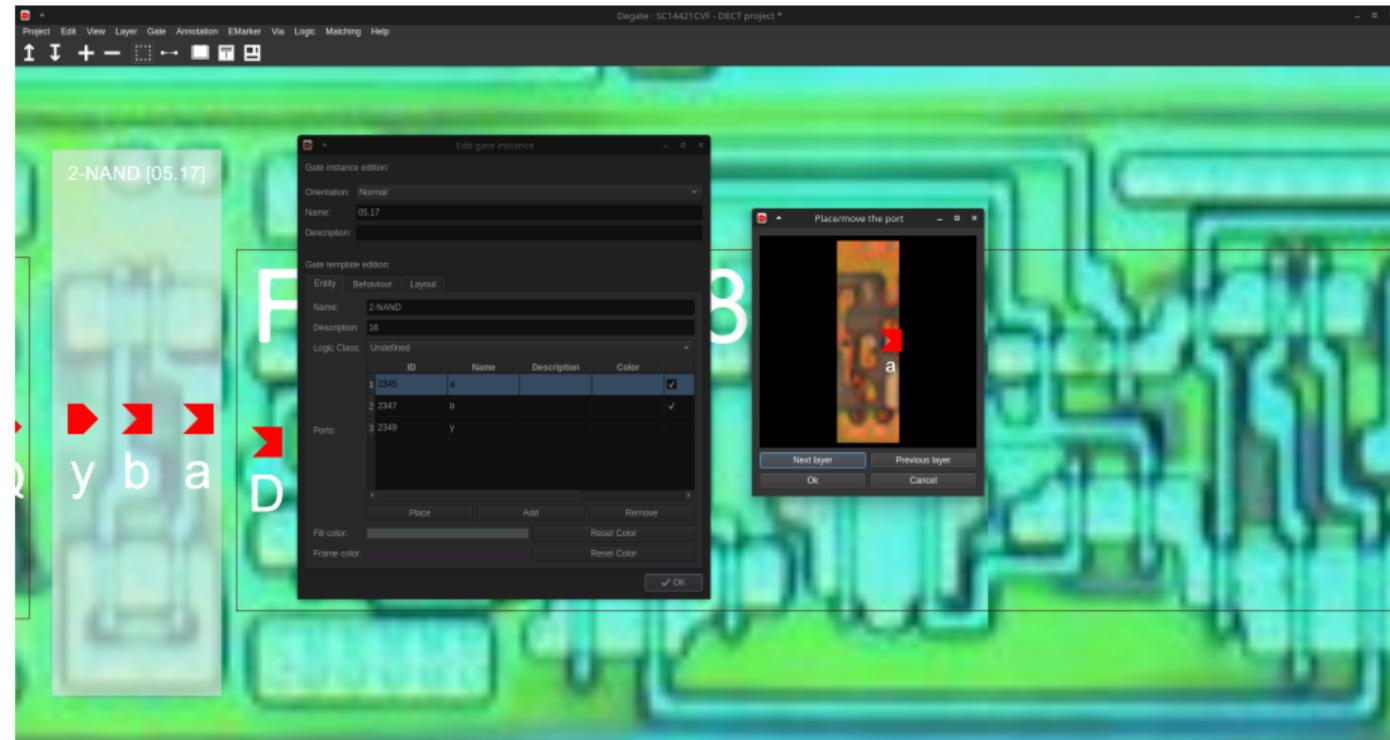


Each sub-project can contains multiple layers (pre-aligned images).

Two project mode: 1. For smaller images, will convert each images in Degate's format (for fast access) and 2. New (WIP, beta) mode for huge images (load only partial tiles in RAM, and doesn't change/import initial file).



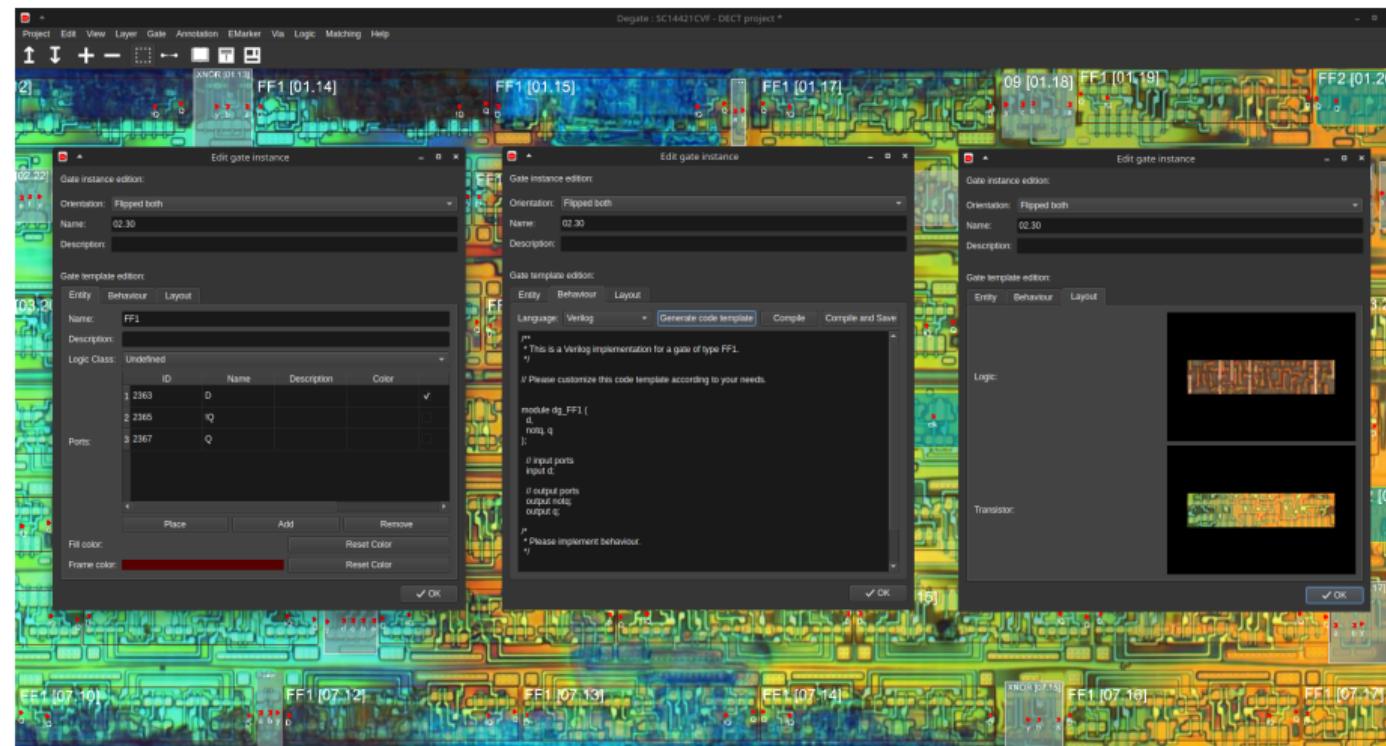
Small Demonstration



Each gate can be described with VHDL/Verilog, have a list of port (placed on image), a type associated etc.



Small Demonstration



Each identified gate (from the gate library) can be matched manually or using template matching algorithms.



Small Demonstration

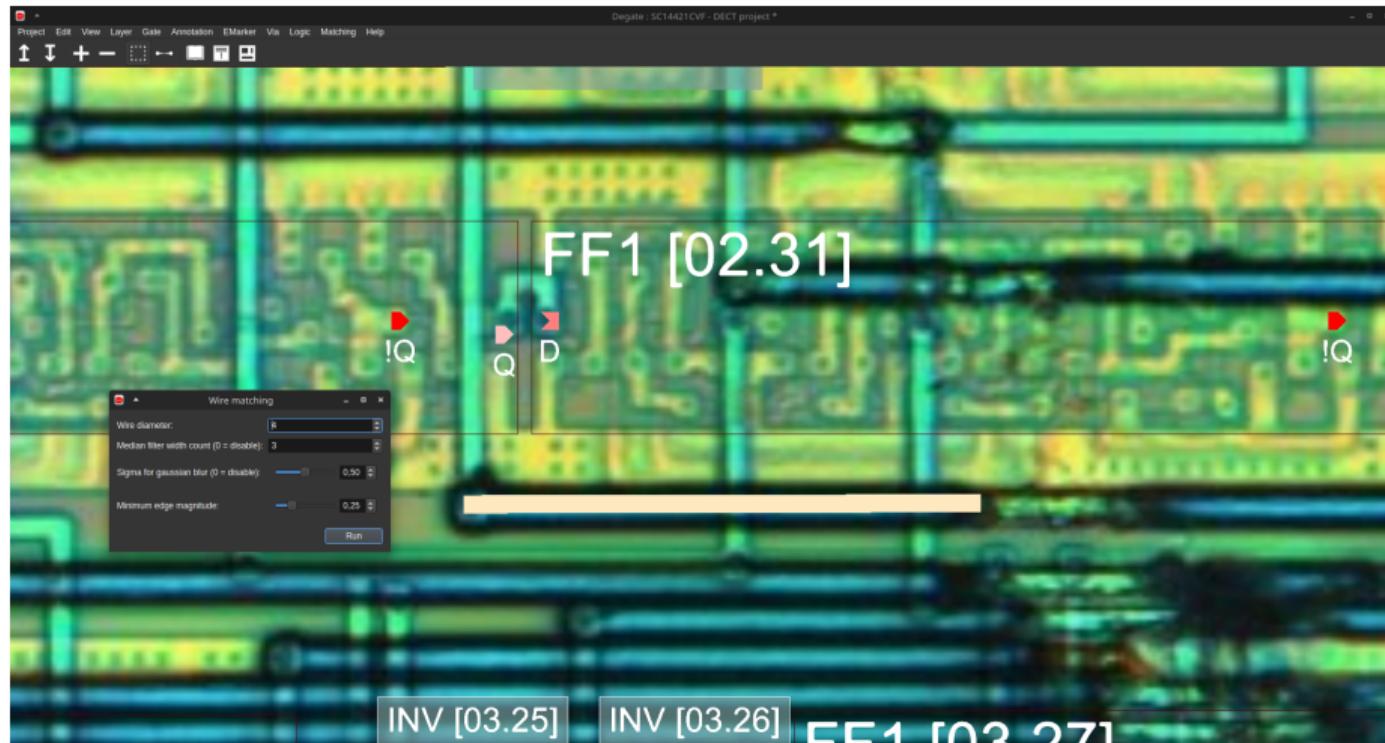


Template matching (will soon be ported to OpenCV) will use gate library to automate gate identification.

Currently it uses normalized cross-correlation (with some more steps).



Small Demonstration

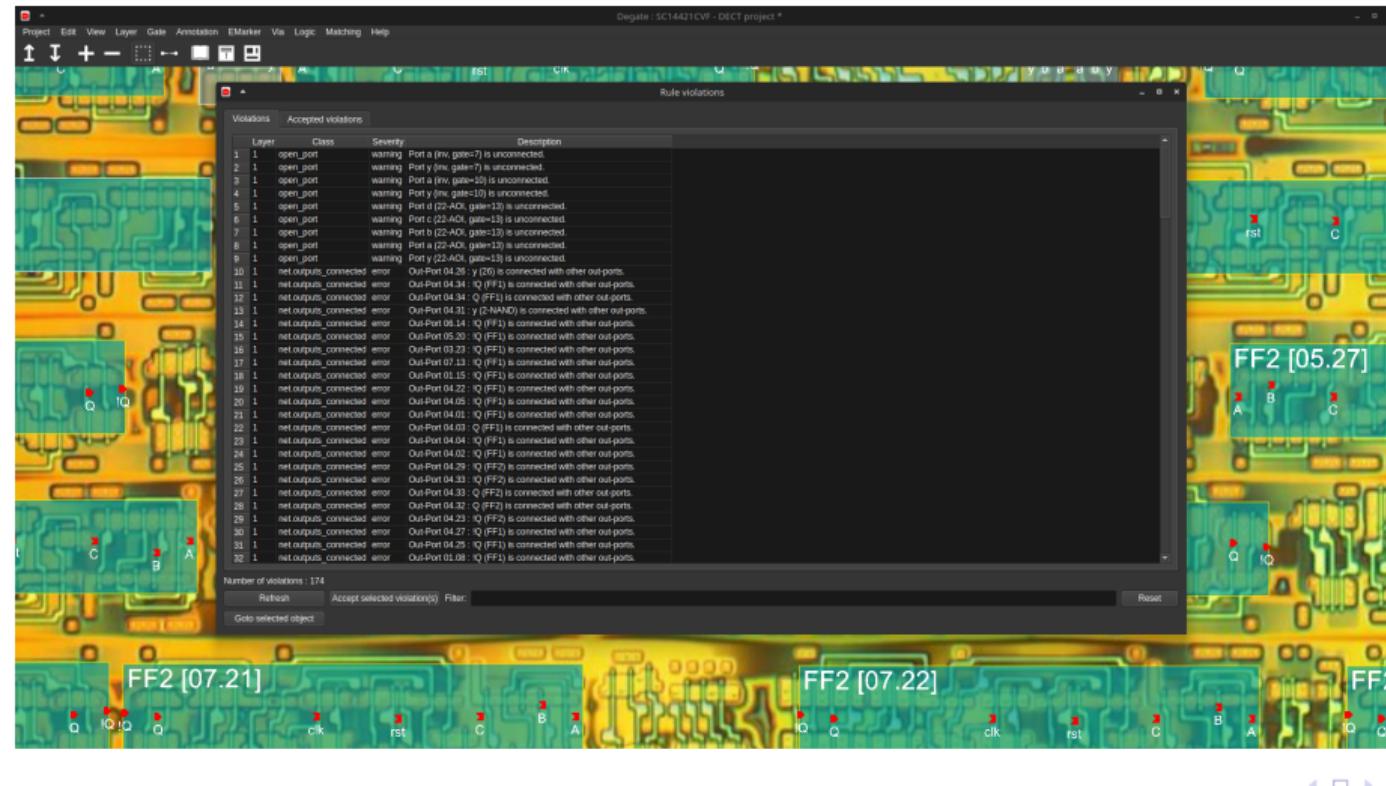


Wire matching, and specifically port interconnection, is the real challenge (and very error prone).

Currently it uses zero crossing edge detection.



Small Demonstration



Helpers are available, like rudimentary (but to be improved) rule checking (e.g. for coherency).

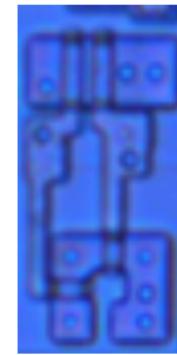
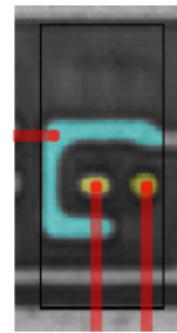
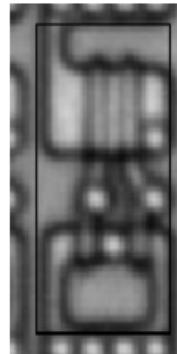


Small Demonstration



Engineering Challenges

- Gate template, wires & vias **matching**.
- Very **huge images** handling.
- **Error** recovery/acceptance/identification.
- Multiple possible **image format** (e.g. .tiff, .png...) & **image source** (e.g. SEM, confocal...).
- 10+ years **old software** (mix of old & new C++).
- **Collaborative** analysis.
- Integrated **gate analyzer**.
- Explicit full **netlist exporter**.



MIFARE NAND[2]

LEGIC NAND[2]

Research Challenges

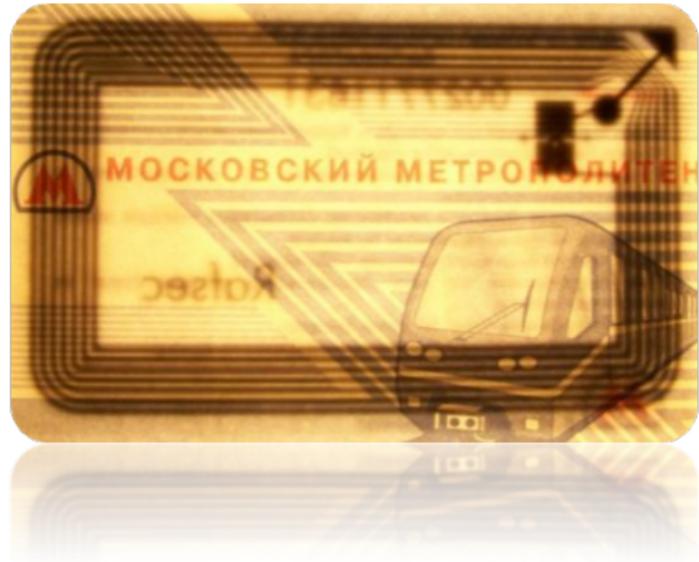
- **3D capture**, imply rethinking Degate (New 3D mode? New software? Really accessible?), and **new algorithms** (e.g. for matching, tracing and gate identification).
- **Machine learning/better algorithms for:**
 - Auto-**vectorization** ;
 - Gate auto **identification** (from vectorized analysis) ;
 - Gate auto **wiring** ;
 - Auto vias & wires **identification**.
- Take advantage of certain capture methods such as **SEM** which makes **automation easier**.
- Making the **field more accessible** (more automation, new abstractions for analysis, communication...).
- Use Degate for **advanced analysis** and **published results**.



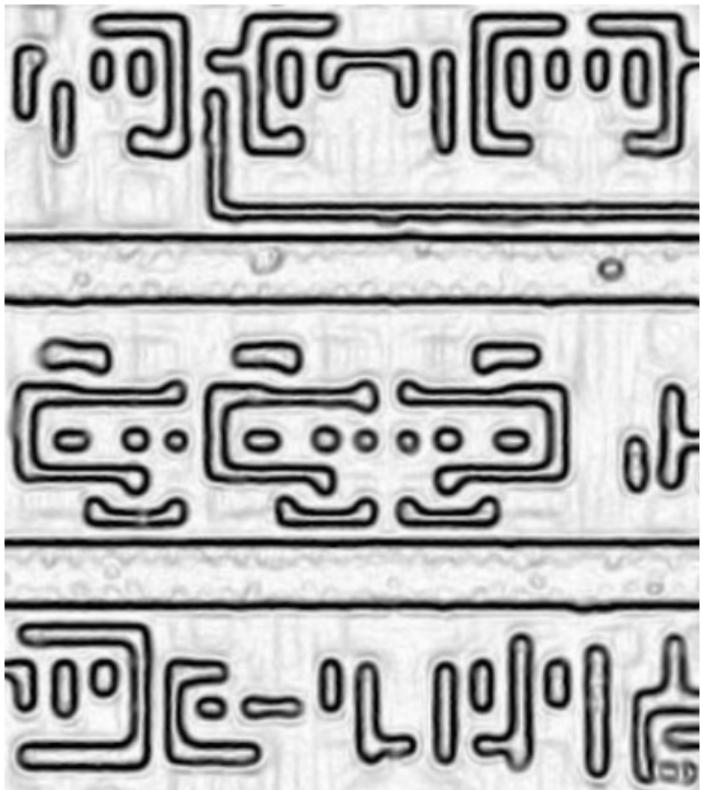
MIFARE Classic Chip [2]

- **RFID card** from NXP launched in 1994.
- Used the **Crypto1 cypher** (until MIFARE Classic EV1, that are using **Hitag2 cipher**).
- **Proprietary encryption** algorithm (stream cipher), security by obscurity.
- Crypto1 cipher is only **implemented in hardware**.
- Used (back in 2008) in more than **3.5 billions cards** (including many building access control systems).

A huge target with a suspicious cypher and security standards?



Degate's origins [5]

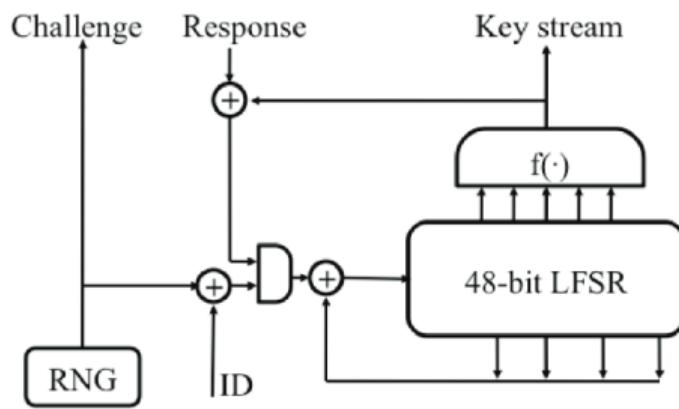


- K. Nohl & Starbug **reverse-engineered the Crypto1 cypher** from MIFARE Classic chip in 2007.
- Used **acetone** to dissolve the RFID cards.
- Used **manual polishing** for delayering.
- Image a total of **6 layers**.
- Identify zone of interest, **searching for 48-bit register &** group of XOR gates.
- Used **standard optical microscope (500x)** & hugin tool for stitching.
- Identified **around 70 types of gates**.
- Used **home-made scripts** (which became the base of Degate) for **template matching** to identify all gates.
- **Manually** reconstructed **connections** between gates.
- Made a **script** to help detecting **wires & vias**.



Consequences [5]

- Using the reverse-engineering results and protocol analysis, authors found **multiple weakness** in the cipher:
 - The cipher is vulnerable to **brute force** attack, key is too small.
 - RNG is predictable, it uses a 16-bit LFSR (linear feedback shift register) **initialized with constant value** and reset at each power-up.
 - There is **only one secret key** for each ID that can result to a specific session key, and all shifts are linear.
- Meaning that just by **sniffing interactions** with the card and the reader, we can compute the key and **retrieve all the data** of the card.
- NXP release a retro-compatible & "hardened" version of the Cipher (Hitag2), which was also weak, MIFARE Classic were "**discontinued**" in 2015.



- Authors **analyzed other RFID devices** after.
- Degate was created** from this analysis, used for other RFID devices reverse-engineering and open-sourced in 2008.



References I

- [1] Mirko Holler, Manuel Guizar-Sicairos, Esther H. R. Tsai, Roberto Dinapoli, Elisabeth Müller, Oliver Bunk, Jörg Raabe, and Gabriel Aeppli.
High-resolution non-destructive three-dimensional imaging of integrated circuits.
Nature, 543(7645):402–406, March 2017.
- [2] Starbug Karsten Nohl.
Pacsec silicon conference.
2009.
- [3] Nils Albartus Ran Ginosara Avi Mendelson Leonid Azriel, Julian Speith and Christof Paar.
Azriel and julian speith and nils albartus and ran ginosara and avi mendelson and christof paar.
Cryptology ePrint Archive, Paper 2021/1278, 2021.



References II

- [4] John McMaster.
Siliconpr0n, <https://siliconpr0n.org/>.
- [5] Karsten Nohl, David Evans, and Henryk Plotz.
Reverse-Engineering a Cryptographic RFID Tag.
page 9.
- [6] Martin Schobert.
Gnu software degate.
Webpage: <http://www.degater.org>.
- [7] Berlin Security Research Labs.
Siliconzoo, <http://siliconzoo.org>.



References III

[8] Ken Shirriff.

Ken shirriff's blog, <https://www.righto.com/>.

[9] Mikhail Svarichevsky.

Zeptobars, <https://zeptobars.com/en/>.

[10] Zonenberg Andrew Yener Bulent.

Csci 4974/6974 hardware reverse engineering, 2014.

