Trojan hardware – some strategies and defenses

Markus Kuhn

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

`http://www.cl.cam.ac.uk/~mgk25/`

Schloß Dagstuhl – 2008-06-19

From a discussion I had with Bob Morris (chief scientist of the NSA's National Computer Security Center 1986–1994) in Spring 1995:

Q: How important is hardware tamper resistance to the military?

A: For anything we built, I made sure the designers assumed that the first 100 units that come off the production line will go directly to Moscow.

Q: Did you also assume that some of the units used will be *produced* in Moscow?

A: No, we didn't do that. That would be rather difficult.

Exercise 1 You are a technician working for the intelligence agency of Amoria. Your employer is extremely curious about what goes on in a particular ministry of Bumaria. This ministry has ordered networked computers from an Amorian supplier and you will be given access to the shipment before it reaches the customer. What modifications could you perform on the hardware to help with later break-in attempts, knowing that the Bumarian government only uses software from sources over which you have no control?

Exercise 2 The Bumarian government is forced to buy Amorian computers as its national hardware industry is far from competitive. However, there are strong suspicions that the Amorian intelligence agencies regularly modify hardware shipments to help in their espionage efforts. Bumaria has no lack of software skills and the government uses its own operating system. Suggest to the Bumarians some operating system techniques that can reduce the information security risks of potential malicious hardware modifications.

M. Kuhn: Introduction to Security, 2nd year CS undergraduate course, Cambridge, Lent 2003.

FBI says military had bogus computer gear

John Markoff, The New York Times, 8 May 2008

Counterfeit products are a routine threat for the electronics industry. However, the more sinister specter of an electronic Trojan horse, lurking in the circuitry of a computer or a network router and allowing attackers clandestine access or control, was raised again recently by the FBI and the Pentagon.

The new law enforcement and national security concerns were prompted by Operation Cisco Raider, which has led to 15 criminal cases involving counterfeit products bought in part by military agencies, military contractors and electric power companies in the United States. Over the two-year operation, 36 search warrants have been executed, resulting in the discovery of 3,500 counterfeit Cisco network components with an estimated retail value of more than $3.5 million, the FBI said in a statement.

Trojan horse software and hardware

Motivations

- $\longrightarrow$ ability to bypass security mechanisms (computer fraud, blackmail, espionage, sabotage, botnets, etc.)
- $\longrightarrow$ copyright marking
- $\longrightarrow$ kill switch

Long and widely recognized as a risk with software and firmware.

A *highly* capable attacker could try to insert one into hardware netlists or even masks (in HDL, at fab, mask production, in ECAD software, standard cell library, etc.):

- $\longrightarrow$ can be particularly difficult to detect
- $\longrightarrow$ can potentially be very difficult to remove or mitigate
- $\longrightarrow$ can be highly potent

Trusted Computing Base

The Trusted Computing Base (TCB) are the parts of a system (hardware, firmware, software) that enforce a security policy.

A good security design should attempt to make the TCB as small as possible, to minimise the chance for errors in its implementation and to simplify careful verification. Faults outside the TCB will not help an attacker to violate the security policy enforced by it.

Example: in a typical PC, the TCB includes at least:

a) the operating system kernel including all its device drivers
b) all processes that run with root privileges
c) all program files owned by root with the set-user-ID–bit set
d) all libraries and development tools that were used to build the above
e) the CPU
f) the mass storage devices and their firmware
g) the file servers and the integrity of their network links

A security vulnerability in any of these could be used to bypass the access control mechanism.

Trojan CPU

Possibilities:

- $\longrightarrow$ undocumented instruction to switch into supervisor mode

  Exploit requirement:
  attacher can send machine code that will be executed (rare)

- $\longrightarrow$ magic sequences that cause data to be executed

  Exploit requirement:
  attacker can send data that will be handled (common)

- $\longrightarrow$ modifications that cause side-channel emanations

  Exploit requirement:
  physical proximity (portable modules)

Samuel T. King, et al.: Designing and implementing malicious hardware. USENIX Workshop on Large-Scale Exploits and Emergent Threats, 2008.
`http://www.usenix.org/event/leet08/tech/full_papers/king/`

Trojan CPU

Example:

- $\longrightarrow$ Take existing CPU design
- $\longrightarrow$ look for instruction sequence commonly used to implement `memcpy()`
- $\longrightarrow$ add a finite state machine that recognizes a fixed 12-byte password in a string that is being copied
- $\longrightarrow$ on detection of the password, the Trojan circuit switches to supervisor mode (ring 0) and executes the data following the password

Alternatives to monitoring `memcpy()`, `strlen()`, etc.:

- $\longrightarrow$ monitor sequence of accumulator values
- $\longrightarrow$ monitor data being written into an L1 cache line
- $\longrightarrow$ . . .

Trojan CPU

Some practicalities:

- $\longrightarrow$ most applications copy received data many times while handling it (e.g., `fprintf()`)
- $\longrightarrow$ different compilers can use very different code for `memcpy()` (e.g. some use trickery like extra-long floating-point registers or multimedia/DSP SIMD units to move data)
- $\longrightarrow$ if space were available, attacker could implement password-detector as a one-way/trap-door function (hash value, digital signature) to prevent others exploiting the same backdoor

Ongoing project: (with Andrew Lewis, Cambridge)

Implement different Trojan-circuit functionality of CPU emulators and evaluate practicality of attacks against common OS kernels, device drivers, applications (e-mail, routing, logging, etc.).

Trojan periphery

- → Harddisk firmware detects password followed by instructions in a written block, e.g. to copy disk blocks
- → Peripheral device with DMA access (graphics card, network adapter, etc.) detects password followed by direct instructions, e.g. to copy RAM regions
- → Human input devices add jitter, leak ssh plaintext (JitterBugs)
  Gaurav Shah, Andres Molina and Matt Blaze: Keyboards and covert channels. 15th USENIX Security Symposium. `http://www.usenix.org/events/sec06/tech/`
- → . . .

Modifications of finished products

- → Replace firmware
- → Insert miniature data-logger or wireless interface circuit
- → Subtle modifications (loose ground return, remove ferrite choke, pull-apart twisted pairs) to increase compromising emanations

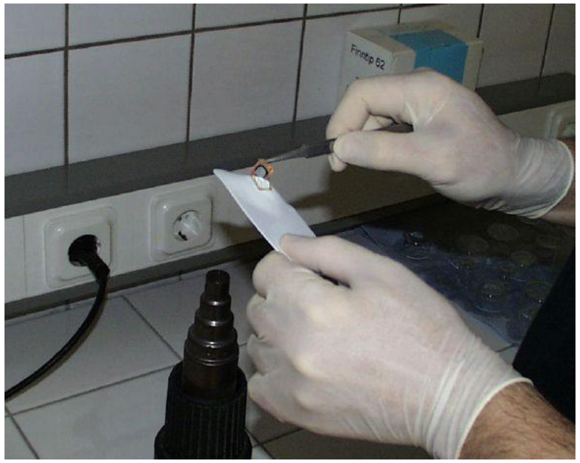
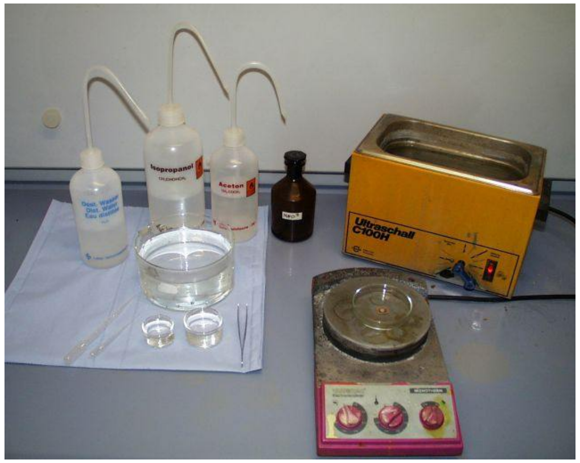
Detection technique I: Reverse engineering

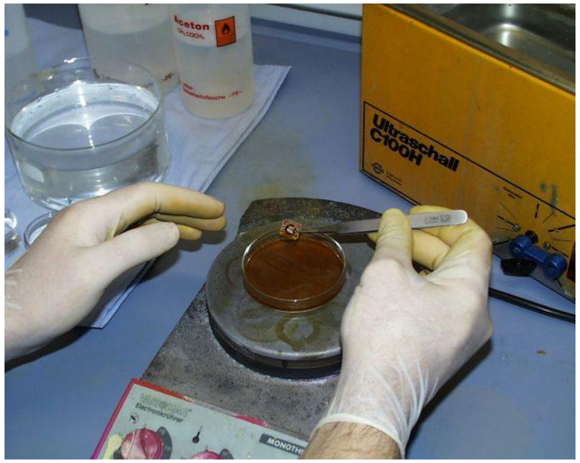
- $\longrightarrow$ depackage semiconductor
- $\longrightarrow$ remove metal-interconnect layers one at a time
- $\longrightarrow$ electron microscopy (in 20th century also optical and UV)
- $\longrightarrow$ reconstruct mask and netlist and compare with what was ordered
- $\longrightarrow$ commercially available services (Semiconductor Insights, Chip-Works, etc.) are contracted routinely to look for patent infringements this way

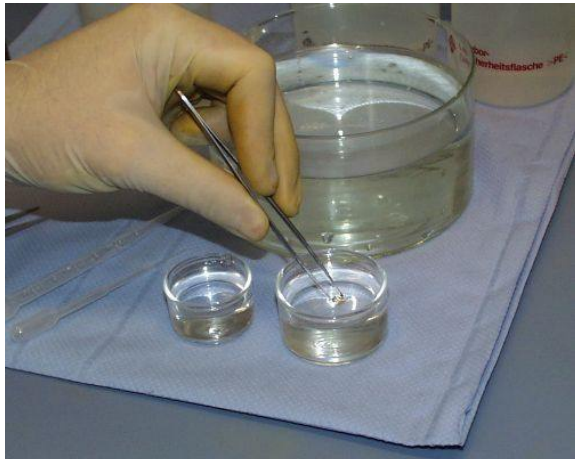
Labour-intensive and expensive $\Rightarrow$ only applicable to small samples.

Following slides from Kömmerling/Kuhn: Design principles for tamper-resistant smartcard processors. USENIX Smartcard Security, 1999.

Preparation I: Depackaging the Processor

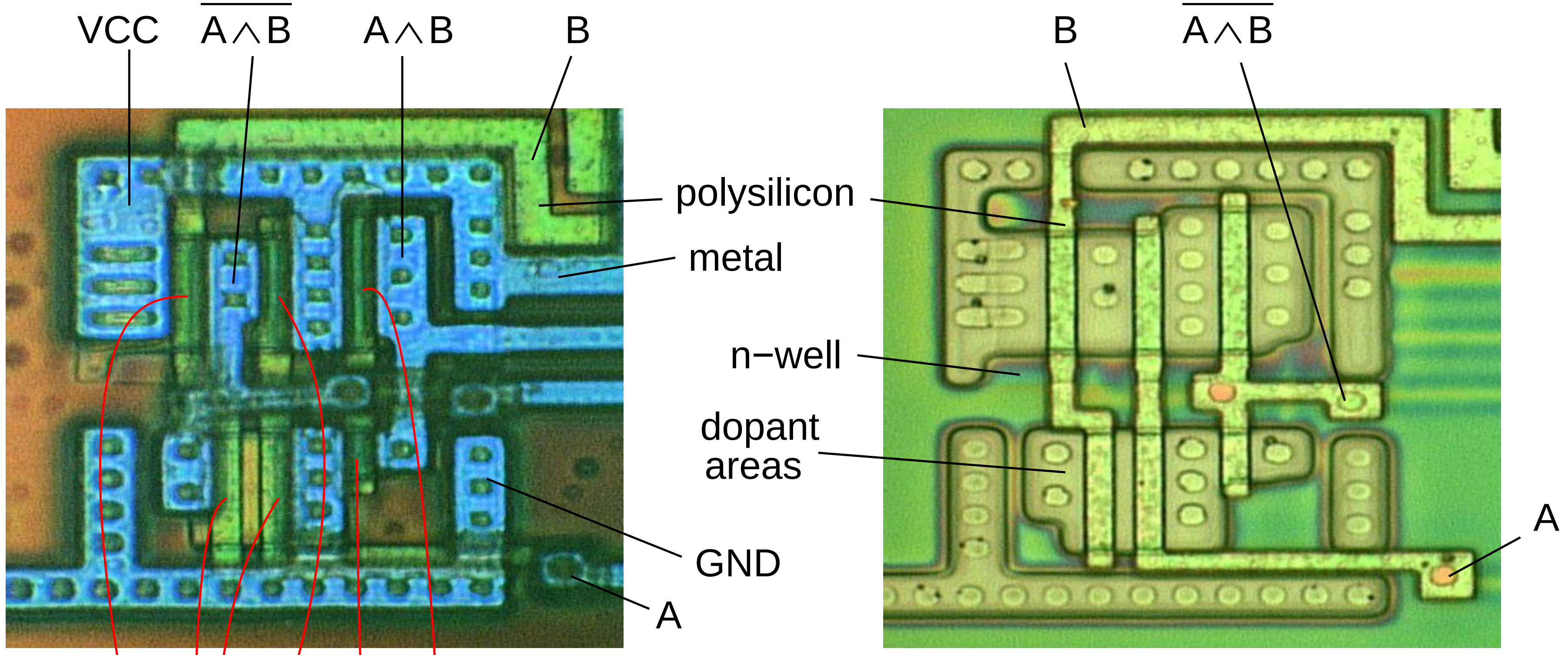



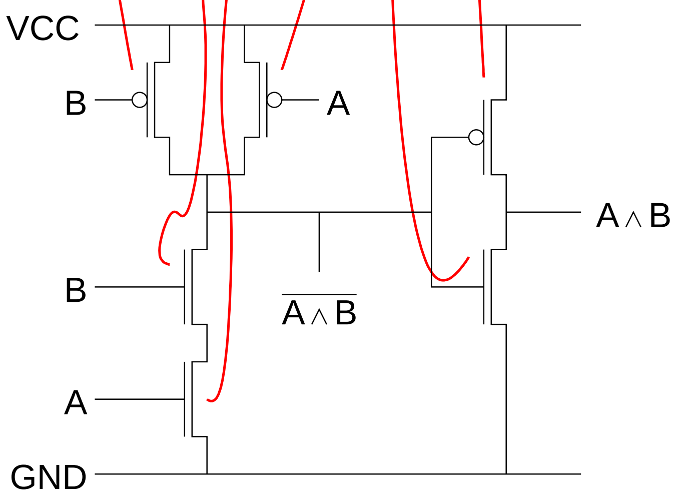




1) Heat up card plastic, bend it, and remove chip module
2) Dissolve package in 60 °C fuming nitric acid, then wash in acetone, deionized water, and finally isopropanol. The etching should be carried out under very dry conditions.

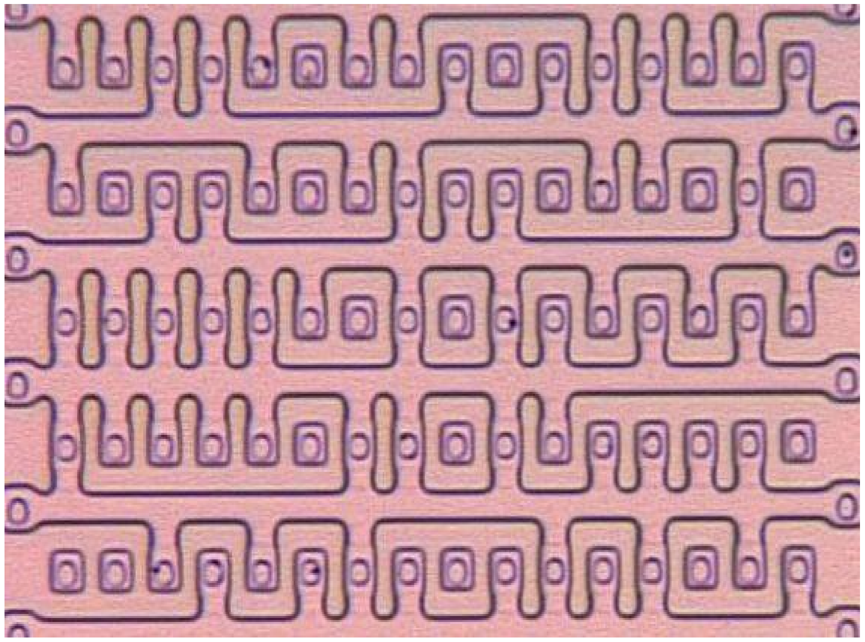
Optical Reverse−Engineering of VLSI Circuits

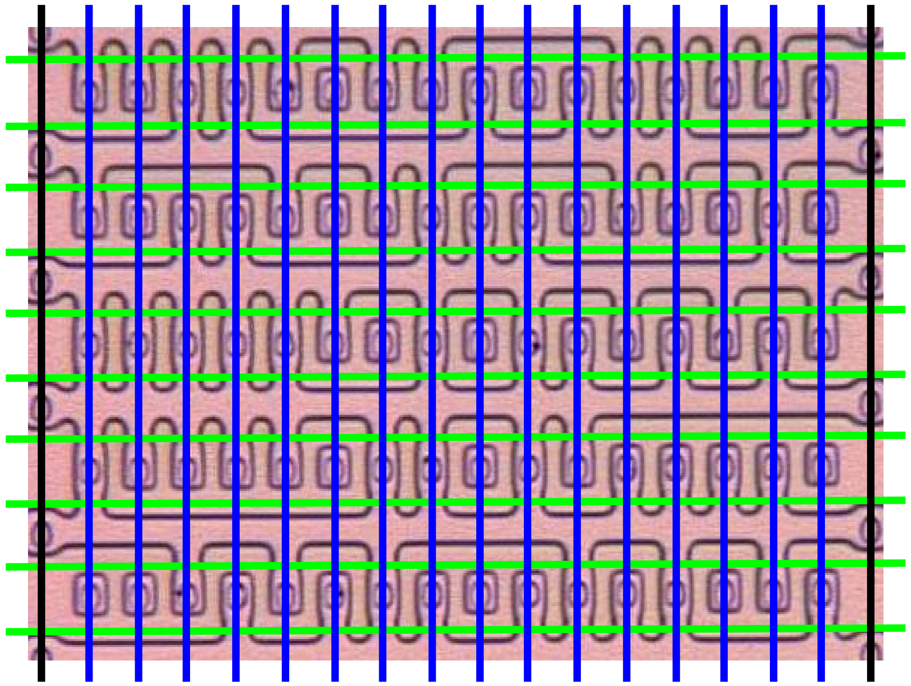




Confocal microscopes represent the different chip layers in different colors. In the right image, the metal interconnects have been removed with hydrofluoric acid. Both images together can be read almost as easily as a circuit diagram.

Optical Access to Diffusion Layer ROM Content

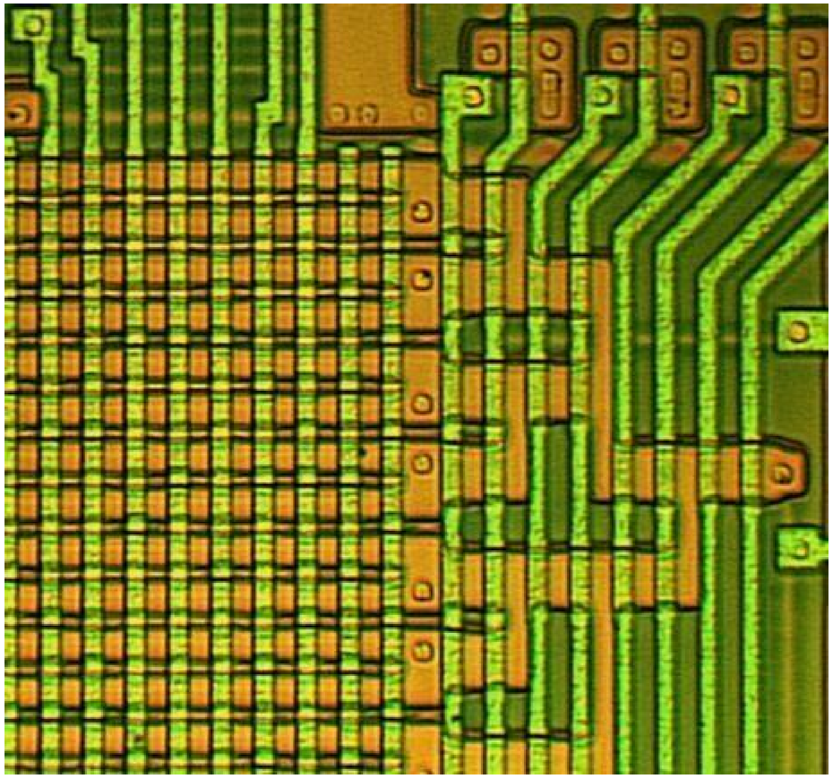




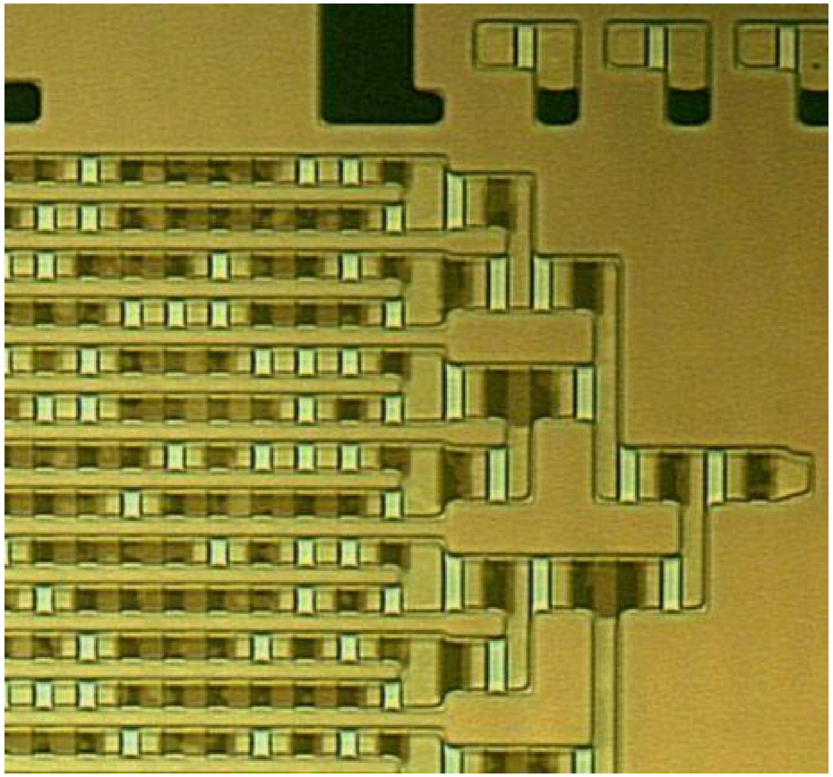
After all covering layers including the surrounding field oxide have been removed with hydrofluoric acid, the shape of the now visible diffusion areas will reveal the ROM content (here 16x10 bits).

🟩	polysilicon row access line
🟦	metal column access line
⬛	ground connection

Optical Reconstruction of Ion Implantation ROM Content



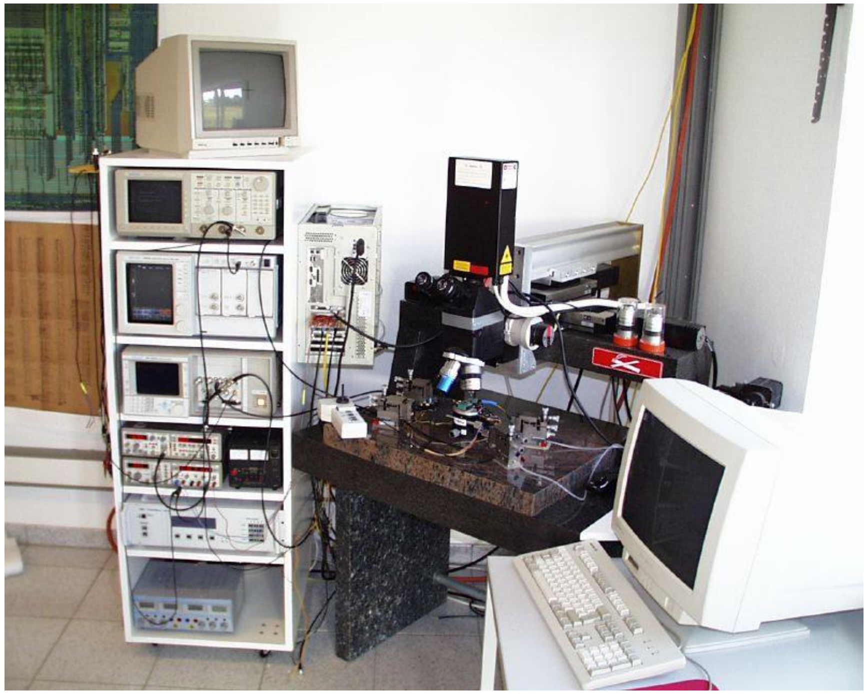
View of ROM with polysilicon intact

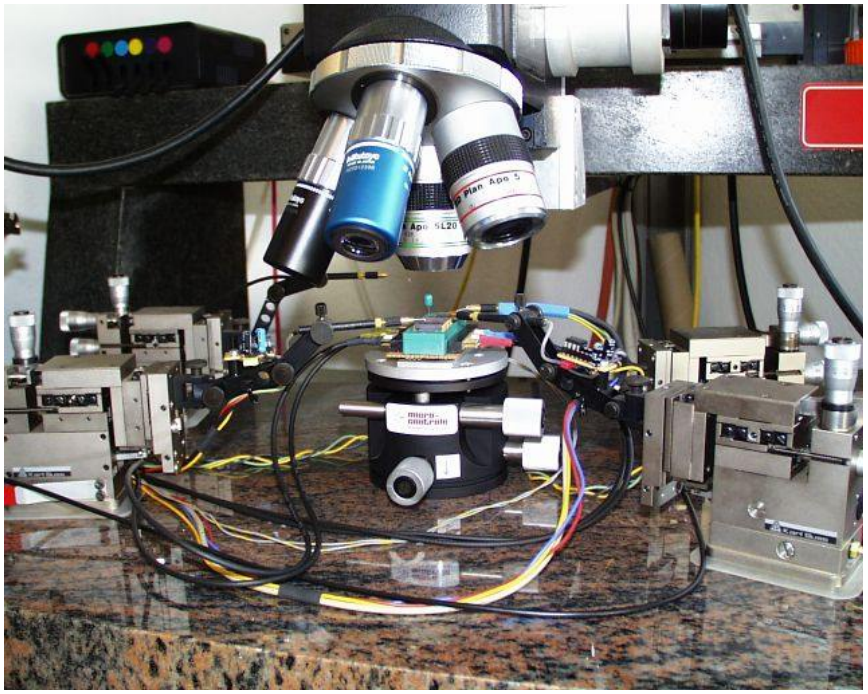


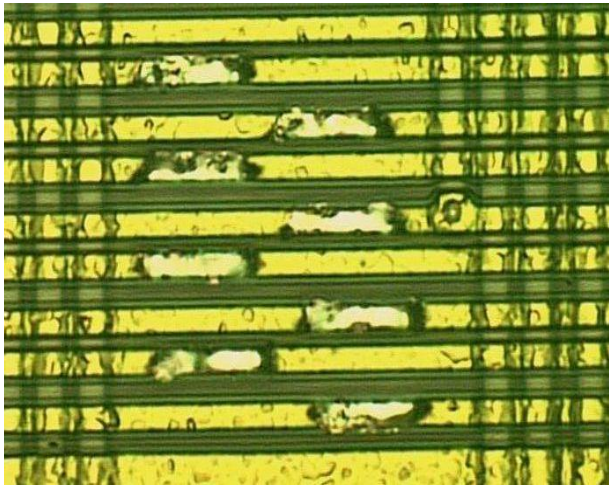
Diffusion layer after crystallographic etch

This type of ROM does not reveal the bit pattern in the shape of the diffusion areas, but a crystallographic staining technique (Dash etchand) that etches doped regions faster than undoped regions will still show the ROM bits.

Access to CPU Bus via Laser Depassivation and Microprobing







Top: A complete microprobing station consisting of a micro−scope (Mitutoyo FS−60), laser cutter (New Wave QuikLaze), four micropositioners (Karl Suss), CCD camera, PC with DSP card for card protocol interface handling and data acquisition, oscilloscope, pattern generator, power supply, logic analyzer, etc. Right: Eight depassivated data bus lines.

Photos: ADSR

Detection technique II: Non-invasive device characterization

Compare modeled global device characteristics with what comes back from the fab:

- $\longrightarrow$ HF current signature (power analysis)
- $\longrightarrow$ EM emissions
- $\longrightarrow$ Thermal signature
- $\longrightarrow$ Timing behaviour

Some techniques already offered commercially for copyright marking of embedded IP-core circuits (e.g., Algotronix DesignTag).

Several recent related papers at: IEEE International Workshop on Hardware-Oriented Security and Trust (HOST), 9 June 2008, Anaheim, CA, US. `http://www.engr.uconn.edu/HOST/`

Detection technique III: Space constraints

Basic idea:

$f(x) = h(x||m)$ cannot be implemented using less memory than the length of string m, if h is secure hash function.

For firmware:

$\longrightarrow$ Available total memory recognizeable by inspecting chip types

$\longrightarrow$ Optimize firmware for space efficiency (no loop unrolling, etc.)

$\longrightarrow$ Pad all remaining firmware memory with random bits

$\longrightarrow$ $m =$ firmware plus random bits

$\longrightarrow$ query $f(x)$ repeatedly in challenge-response protocol

$\longrightarrow$ no space left for attacker to insert Trojan without failing challenge-response protocol

Used in mid-1990s in BSkyB pay-TV smartcards to counter attackers rewriting their firmware.

Arvind Seshadri, Adrian Perrig, Leendert van Doorn, Pradeep Khosla: SWAtt: software-based attestation for embedded devices. IEEE Symposium on Security and Privacy, 2004.

Detection technique IV: Watchdogs

Basic idea:

- ⟶ Organizational compartmentalization can help to ensure that no single attacker has access to the full design of a chip.
- ⟶ Trojan could be implemented at a number of places, but its behaviour can also be detected at a number of places.
- ⟶ E.g., a number of preconditions need to be satisfied before virtual-memory is deactivated and supervisor mode entered (PC in defined address range, characteristic instructions executed).
- ⟶ If simple string copy triggers the non-VM memory access, watch-dog circuit prevents action (halts CPU?)