

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.

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.

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.

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 horse software and hardware

Motivations

- ability to bypass security mechanisms (computer fraud, blackmail, espionage, sabotage, botnets, etc.)
- copyright marking
- 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.):

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

Trojan CPU

Possibilities:

- modifications that cause side-channel emanations
Exploitable if: physical proximity (portable modules)
- tampered hardware RNG (low entropy)
Exploitable if: naive RBS post-processing
- undocumented instruction to switch into supervisor mode
Exploitable if:
attacker can send machine code that will be executed (rare)
- magic sequences that cause data to be executed
Exploitable if:
attacker can send data that will be handled (**common!**)

Trojan CPU

Example:

- Take existing CPU design
- look for instruction sequence commonly used to implement `memcpy()`
- add a FSM circuit that recognizes a fixed 12-byte password in a string that is being copied
- 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.:

- monitor sequence of accumulator values
- monitor data being read from L1 cache line
- ...

Trojan CPU

Some practicalities:

- most applications copy received data many times while handling it (e.g., `fprintf()`)
- `memcpy()` problem: different compilers use very different code (e.g. some use trickery like extra-long floating-point registers or multimedia/DSP SIMD units to move data)
- 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 work: (with Andrew Lewis)

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.).

Recent demonstration of similar idea:

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/

- detect magic sequence on data bus \Rightarrow disable memory protection and perform write access
- “Shadow mode:” switch to duplicated data and instruction cache HW. Helps malicious firmware to hide itself from scanners (i.e., not stored in OS-visible address space). Similar to Intel’s System Management Mode, but all on-chip.
- Minor extensions to existing debugging hardware (trap not only on address bus but also on data bus values)

Less than 0.1% increase in number of gates of Leon3 processor.

Example attack demonstrated:

- Attacker sends malicious UDP packet to Trojan CPU
- UDP checksum calculation triggers magic-sequence detection
- use that to upload malicious firmware into shadow cache
- on activation of malicious firmware, it uses debugging hardware to trap and modify behaviour of Linux login
- backdoor login deactivates and removes malicious firmware

Malicious firmware only active very briefly

⇒ hardly affects timing of OS and applications.

Not many details of the exact design of the Trojan circuits given in the paper.

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

Classes of Trojan IC attacker capabilities

- I full control over HDL source and mask-generation toolchain (design house)
- II partial access to HDL (compartmentalized design house)
- III access to mask geometry data only (mask producer)
- IV access to physical masks (mask producer → fab)
- V access to production process (lithography fab)
- VI access to finished die before/after packaging (laser individualization, firmware upload, etc.)

Detection technique I: Reverse engineering

- depackage semiconductor
- remove metal-interconnect layers one at a time
- electron microscopy (in 20th century also optical and UV)
- stain dopant areas
- reconstruct mask, recognize standard cells, reconstruct netlist and compare with what was ordered
- commercially offered service, used to spot patent infringements (Semiconductor Insights, ChipWorks, etc.), $> 10^6$ US\$/product

Labour-intensive and expensive \Rightarrow only applicable to small samples.
Might still fail to spot minor tampering *within* standard cells.

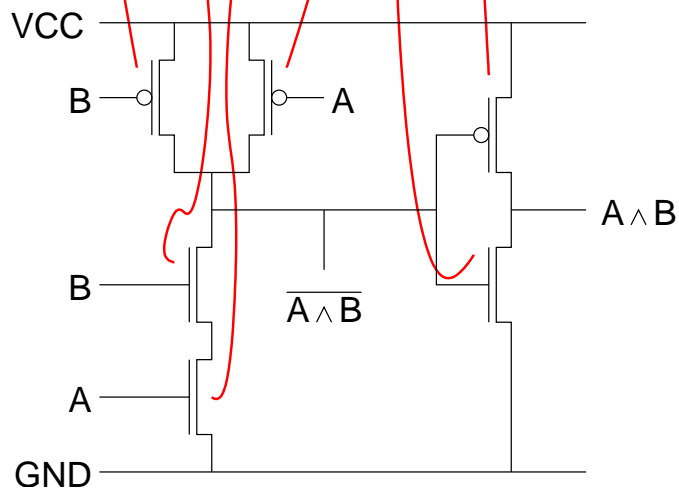
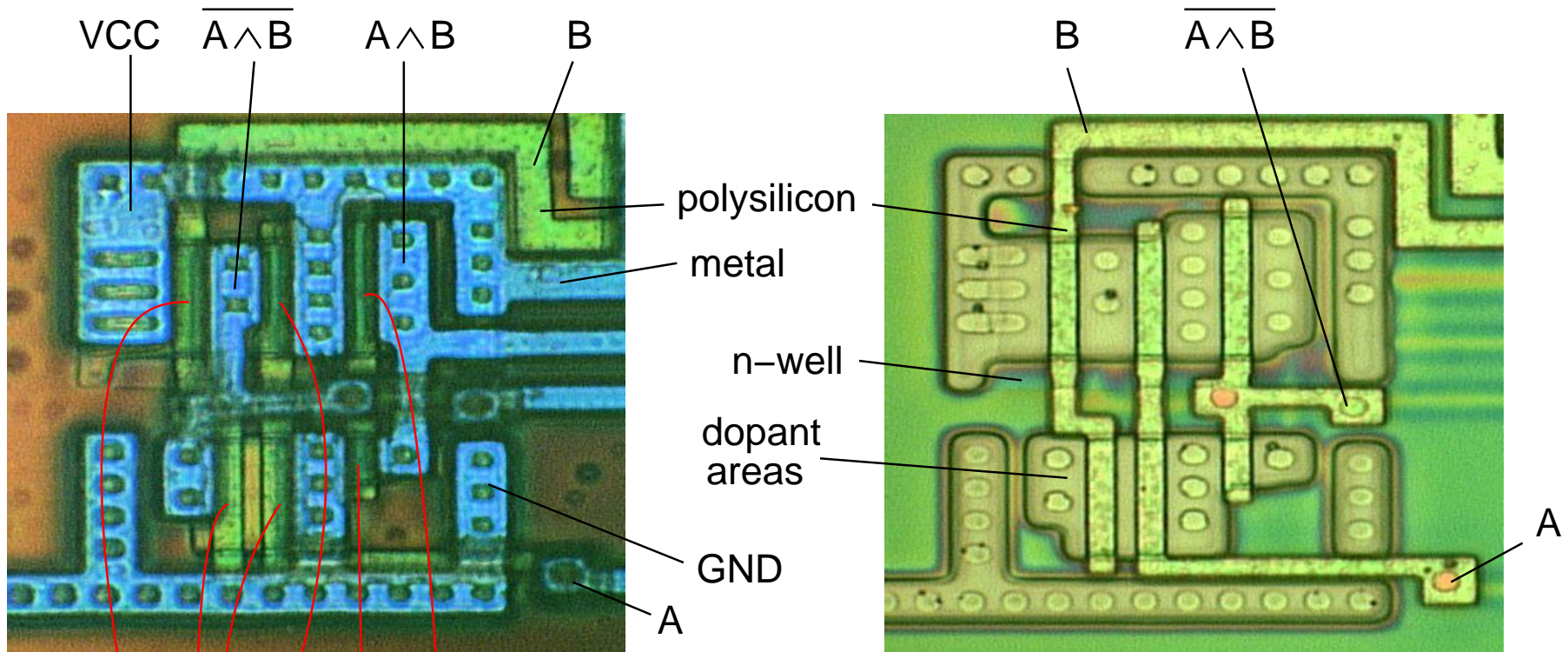
Following slides from Kömmerling/Kuhn: Design principles for tamper-resistant smartcard processors. USENIX Smartcard Security, 1999.

Preparation I: Depackaging the Processor



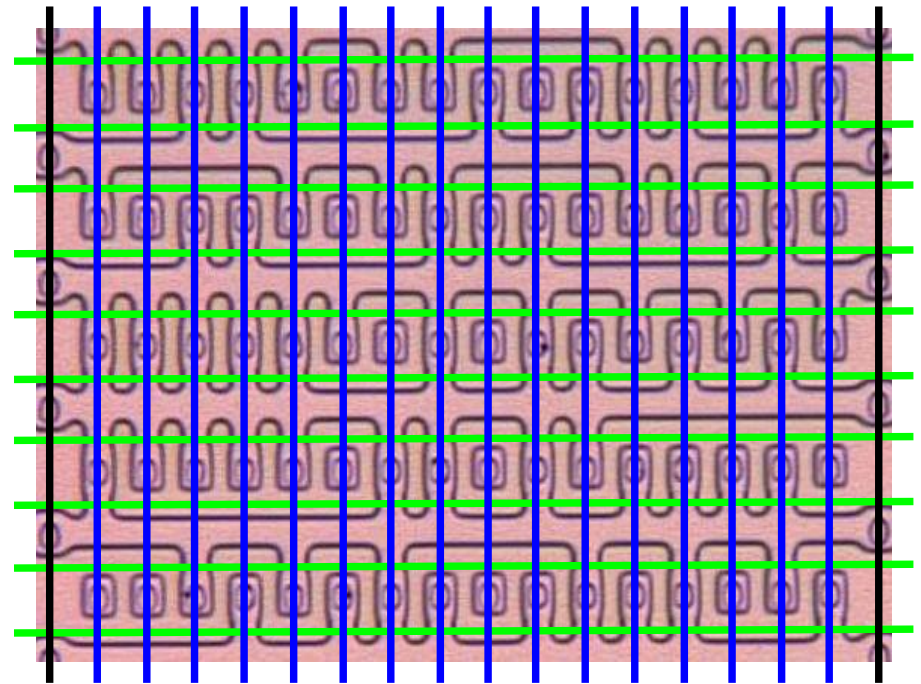
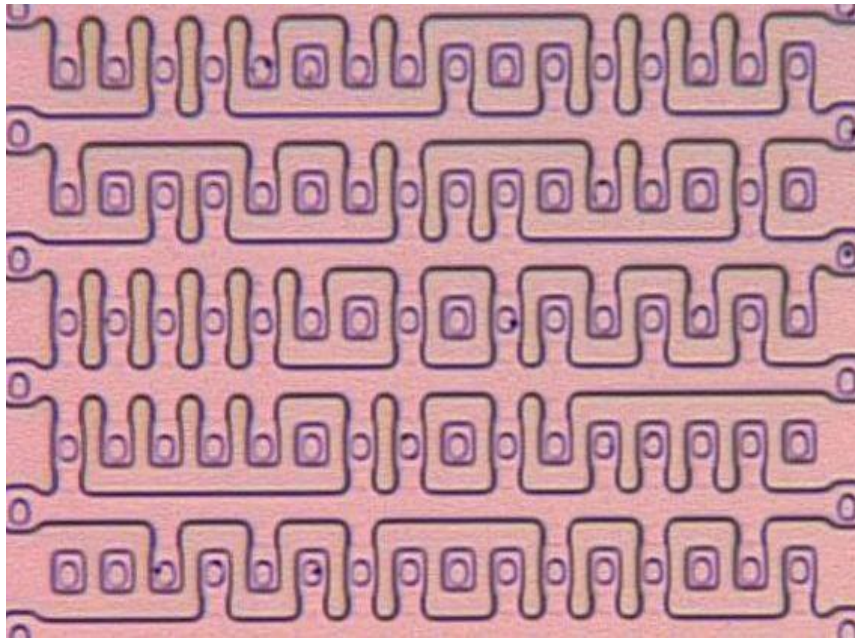
- 1) Heat up card plastic, bend it, and remove chip module
- 2) Dissolve package in 60 °C fuming nitric acid, then wash in acetone, deionized water, and finally isopropanol. The etching should be carried out under very dry conditions.

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.

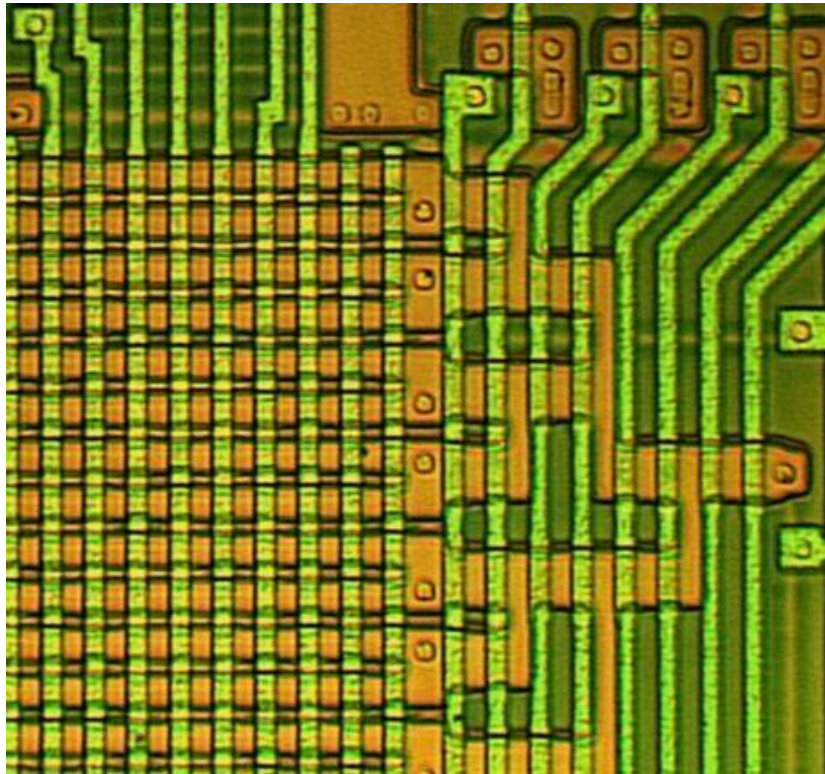
Optical Access to Diffusion Layer ROM Content



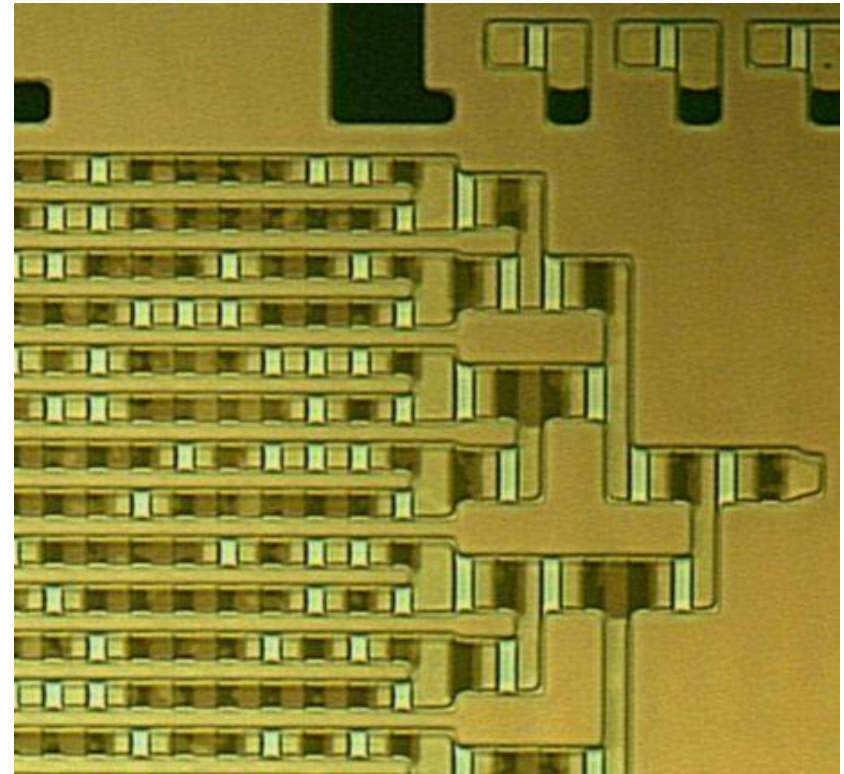
After all covering layers including the surrounding field oxide have been removed with hydrofluoric acid, the shape of the now visible diffusion areas will reveal the ROM content (here 16x10 bits).

- polysilicon row access line
- metal column access line
- ground connection

Optical Reconstruction of Ion Implantation ROM Content



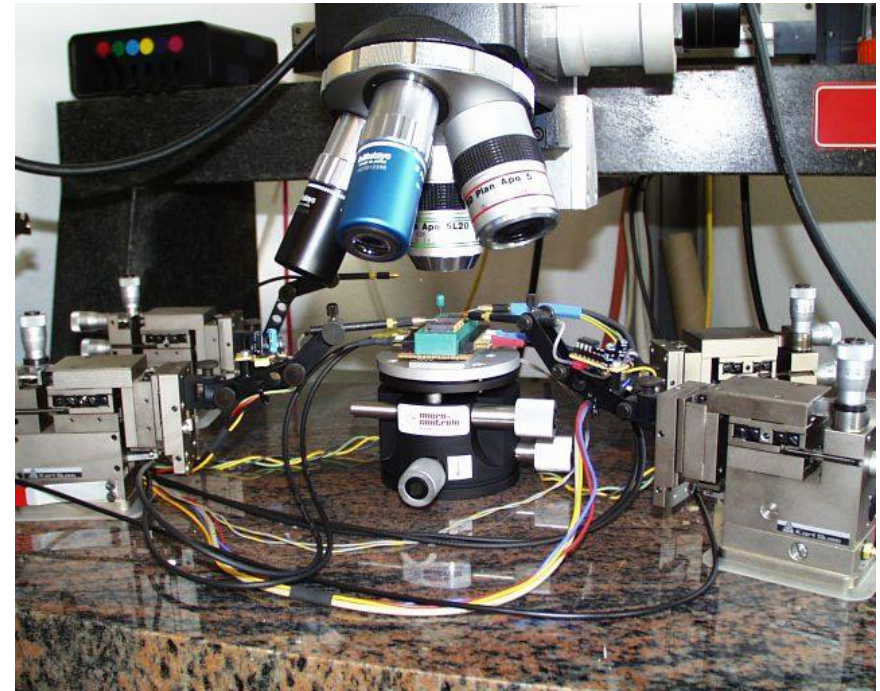
View of ROM with polysilicon intact



Diffusion layer after crystallographic etch

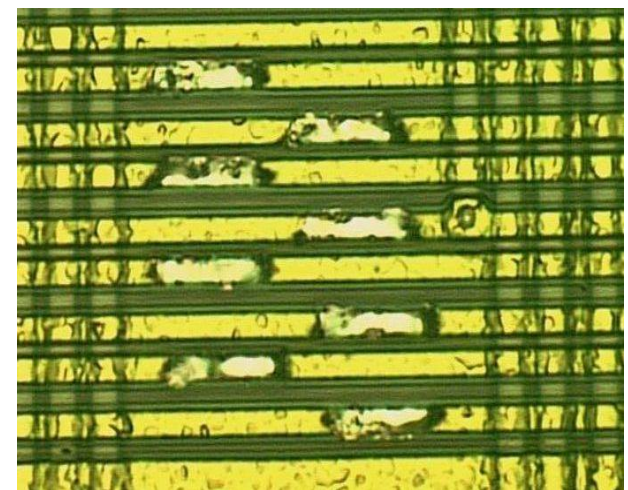
This type of ROM does not reveal the bit pattern in the shape of the diffusion areas, but a crystallographic staining technique (Dash etch) that etches doped regions faster than undoped regions will still show the ROM bits.

Access to CPU Bus via Laser Depassivation and Microprobing



Top: A complete microprobing station consisting of a microscope (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: Fast mask-geometry validation

- Look only for complex Trojan (hundreds of gates added), assumed to be necessary for data-driven remote attacks
- Assume that attacker has shrunk some mask area by a few percent (in one direction) to make space for 0.1% more gates
- position of each transistor predictable better than a few tens of nanometers, otherwise masks wouldn't align (remove linear trend due to mask temperature changes)
- Optically verify location of a sample of all gates from mask geometry data
- Should be much cheaper and more scalable than reverse engineering unknown mask or netlist

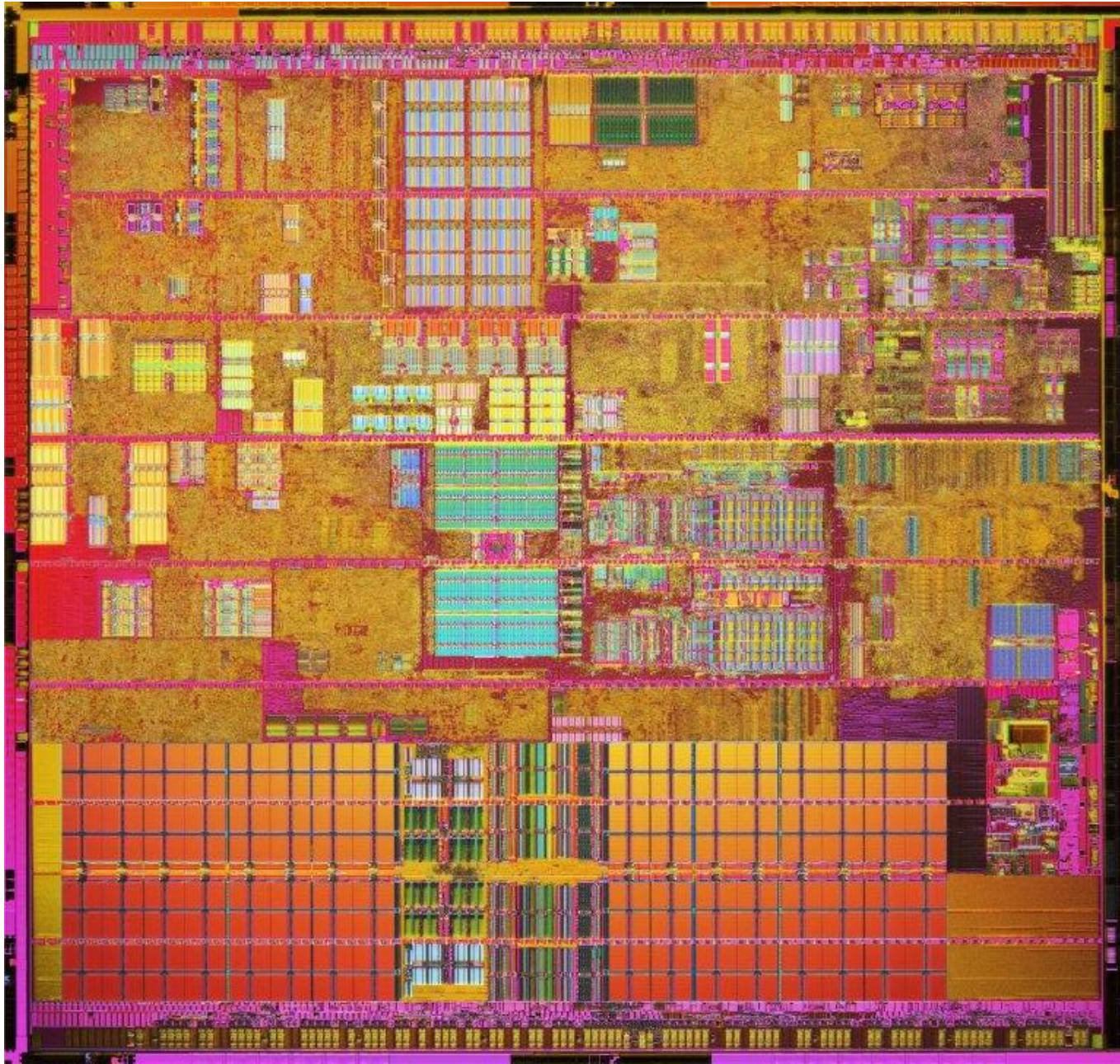
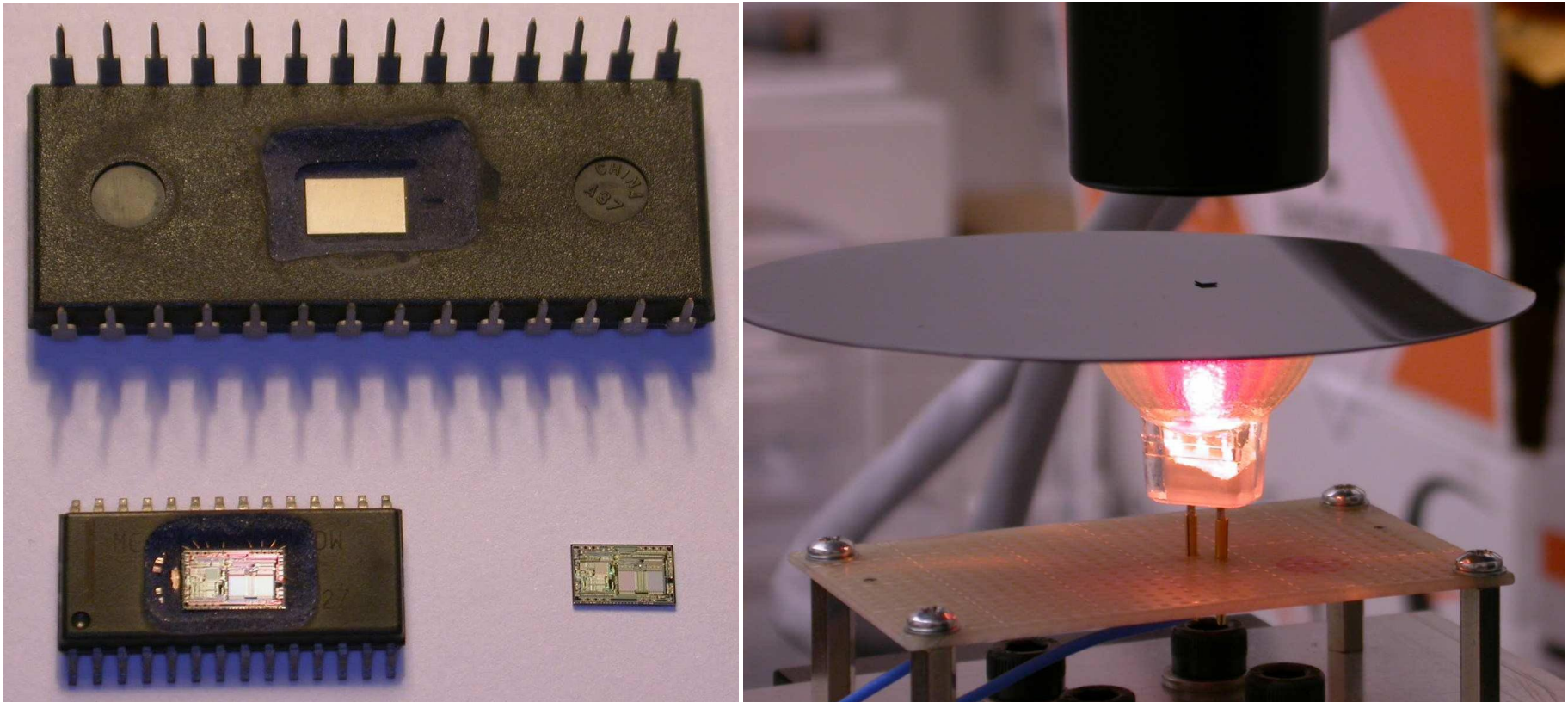


Photo: Smithfield CPU, www.intel.com

IR transmission photography

Reflective surface microscopy reveals only 1–2 top metal interconnect layers.

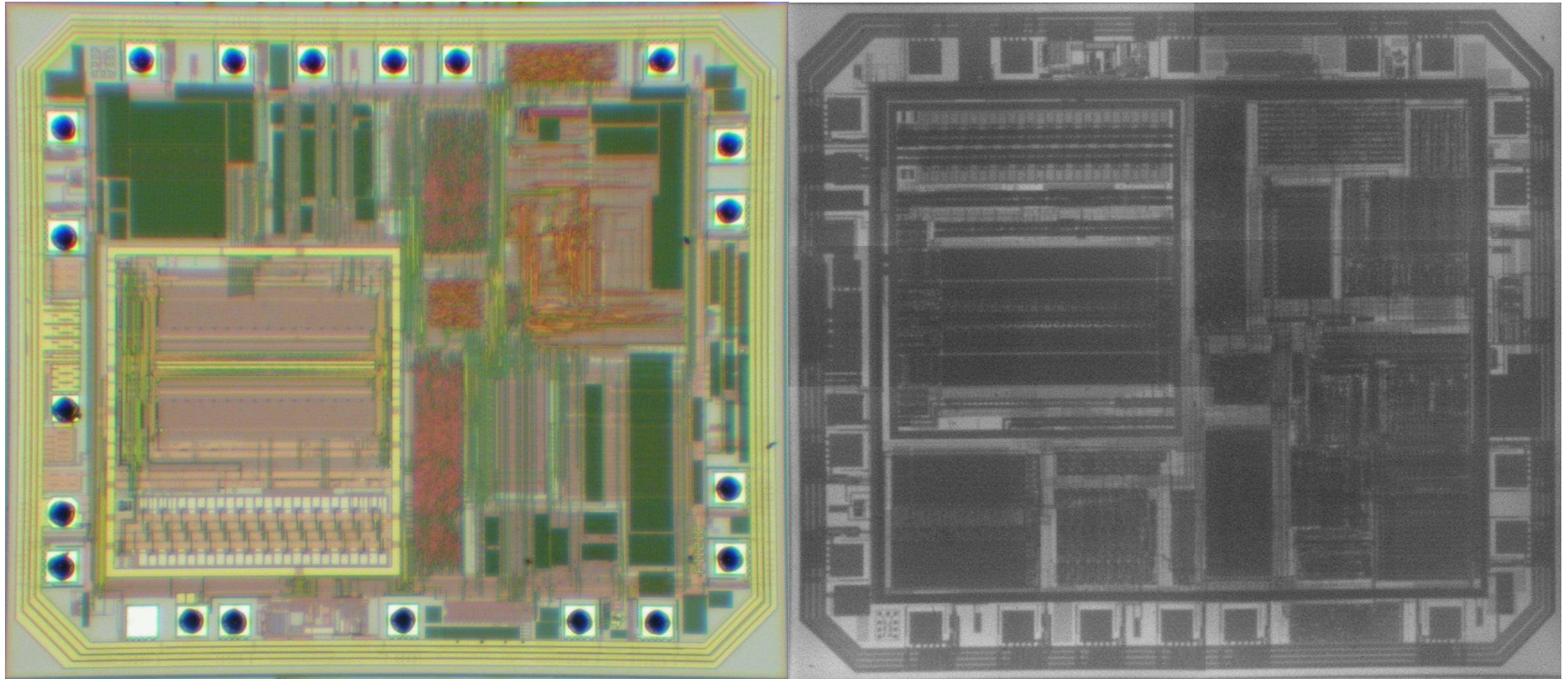
But silicon is transparent to infrared light: transmission microscopy



Photos: Sergei Skorobogatov

<http://www.cl.cam.ac.uk/research/security/posters/sps32-chipimaging.pdf>

Infrared light is not sufficient to resolve modern gates, but it could still reveal substantial changes to lower-layer masks.



Photos: Sergei Skorobogatov

<http://www.cl.cam.ac.uk/research/security/posters/sps32-chipimaging.pdf>

Alternative: X-ray imaging (microscopy difficult, X-ray lenses?)

Detection technique III: Non-invasive device characterization

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

- HF current signature (power analysis)
- EM emissions
- Thermal signature
- 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 IV: 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:

- Available total memory recognizable by inspecting chip types
- Optimize firmware for space efficiency (no loop unrolling, etc.)
- Pad all remaining firmware memory with random bits
- $m = \text{firmware plus random bits}$
- query $f(x)$ repeatedly in challenge-response protocol
- 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.

Could this be extended to gate networks?

- Do semiconductors contain circuitry that is rarely or never used (e.g., to aid debugging of chip) or may not be easily missed by users (e.g., exact cache size) and might be replaced by attacker?
- What mechanisms can we add (software-accessible scan chains rather than JTAG pins, etc.) to make it practical for software to attest in challenge-response protocols the presence of *all* expected hardware functionality?

Detection technique V: 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, watchdog circuit prevents action (halts CPU?)

Other precautions

Inserting a complex Trojan into mask geometry is likely to require substantial design time. Organizational procedures can make this impractical:

- Compartmentalized design process
(attacker within design team can't see watchdog)
- minimize access (people and time) to masks and mask data
- separate mask and chip production
- provide masks only shortly before production run
- avoid reusing all masks unmodified, add many remotely attestable mask serial numbers to design (most practical in short-lived consumer products, laser-cutter modification of masks)