Hardware-Oriented Security

Hardware Trojans

Prof. Dr. Stefan Katzenbeisser

Lehrstuhl für Technische Informatik, FIM

Outline

- **What are HW Trojans?**
- Which types of HW Trojans exist?
- How do HW Trojans look?
- How can HW Trojans be detected?

HW Trojans: Definition

- Malicious modification of an integrated circuit during its design flow that adds or removes functionality or reduces reliability

- Applications:
  - Evasion of control
  - Industrial espionage
  - Military
  - …

- HWT vs. Backdoor:
  - HWT pretends to perform specified/harmless function
  - HWT can activate/control Backdoor

Targets for HW Trojans

- Military products
  - Production of almost all chips is outsourced to third countries
  - US DoD „Trusted Foundries Program“
  - Kill switches?
- Critical Infrastructures
  - Can be affected in a „cyberwar“
- Consumer Electronics
  - Sabotage of competitor
  - Compromises of privacy

HW Trojans: Purposes

Information extraction:

- Data: personal information, parameters, log files
- Cryptographic keys
- Seeds for Pseudo-Random Number Generators
- Identity, behavior
- Code, Firmware, apps, net lists

Information insertion:

- Code: Malware (Kill switches, dysfunctional software/hardware)
- Data: modified parameters, functionality (DoS, access)
- Known or weak keys for cryptographic use
- Known seeds for PRNG
- Compromising data (false flag ops)

Known cases: Syrian radar (2007)

01 May 2008 | 19:57 GMT

The Hunt for the Kill Switch

Are chip makers building electronic trapdoors in key military hardware? The Pentagon is making its biggest effort yet to find out

By **Sally Adee**

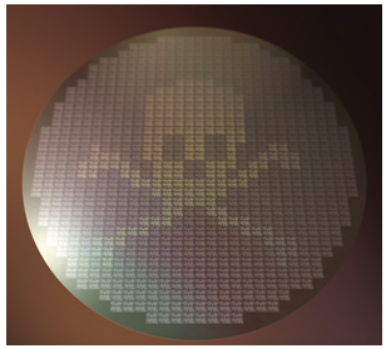


Image: James Archer/AnatomyBlue

Last September, Israeli jets bombed a suspected nuclear installation in northeastern Syria. Among the many mysteries still surrounding that strike was the failure of a Syrian radar—supposedly state-of-the-art—to warn the Syrian military of the incoming assault. It wasn't long before military and technology bloggers concluded that this was an incident of electronic warfare—and not just any kind.

Post after post speculated that the commercial off-the-shelf microprocessors in the Syrian radar might have been purposely fabricated with a hidden "backdoor" inside. By sending a preprogrammed code to those chips, an unknown antagonist had disrupted the chips' function and temporarily blocked the radar.

[ISN] Israel Shows Electronic Prowess

InfoSec News | Mon, 26 Nov 2007 22:28:35 -0800

http://www.aviationweek.com/aw/generic/story_channel.jsp?channel=defense&id=news/aw112607p2.xml

By David A. Fulghum, Robert Wall and Amy Butler
Aviation Week & Space Technology
Nov 25, 2007

The U.S. was monitoring the electronic emissions coming from Syria during Israels September attack; and-although there was no direct American help in destroying a nuclear-reactor there was some advice provided beforehand, military and aerospace industry officials tell Aviation Week & Space Technology.

That surveillance is providing clues about how Israeli aircraft managed to slip past Syrian air defenses to bomb the site at Dayr az-Zawr. The main attack was preceded by an engagement with a single Syrian radar site at Tall al-Abuad near the Turkish border. It was assaulted with what appears to be a combination of electronic attack and precision bombs to enable the Israeli force to enter and exit Syrian airspace. Almost immediately, the entire Syrian radar system went off the air for a period of time that included the raid, say U.S. intelligence analysts.

There was no U.S. active engagement other than consulting on potential target vulnerabilities, says a U.S. electronic warfare specialist.

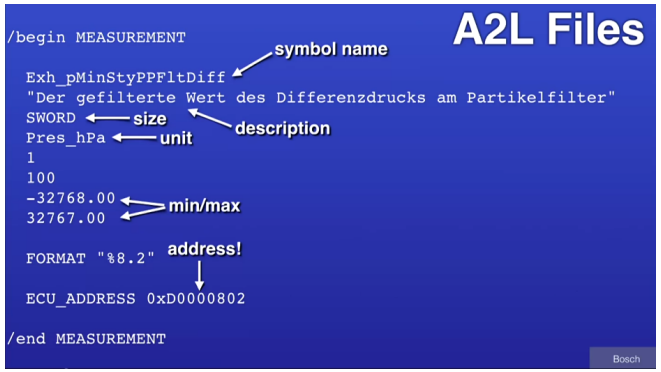
Elements of the attack included some brute-force jamming, which is still an important element of attacking air defenses, U.S. analysts say. Also, Syrian air defenses are still centralized and dependent on dedicated HF and VHF communications, which made them vulnerable. The analysts dont believe any part of Syrias electrical grid was shut down. They do contend that network penetration involved both remote air-to-ground electronic attack and penetration through computer-to-computer links.

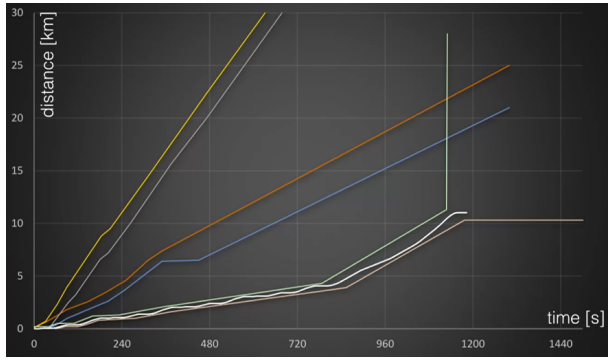
Sources:
https://spectrum.ieee.org/semiconductors/design/the-hunt-for-the-kill-switch
https://www.mail-archive.com/isn@infosecnews.org/msg02216.html

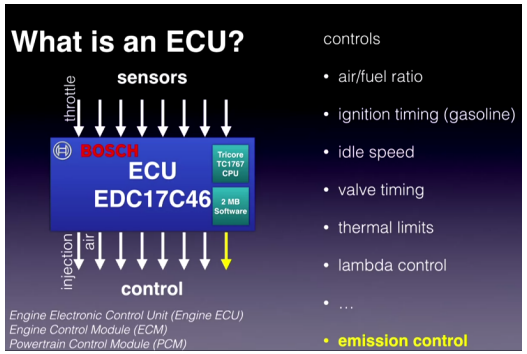
Known cases: Fake parts in military supply chain

Sources:
http://dangerousprototypes.com/blog/2012/08/02/counterfeit-parts-found-on-p-8-posiedons/
https://edition.cnn.com/2011/11/07/us/u-s-military-bogus-parts/index.html

Known cases: „Dieselgate“



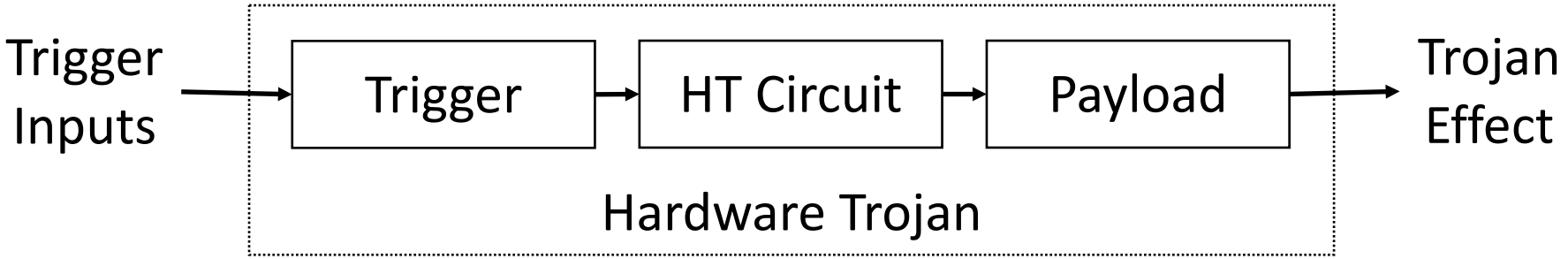




Source:https://media.ccc.de/v/32c3-7331-the_exhaust_emissions_scandal_dieselgate

HW Trojans: Properties

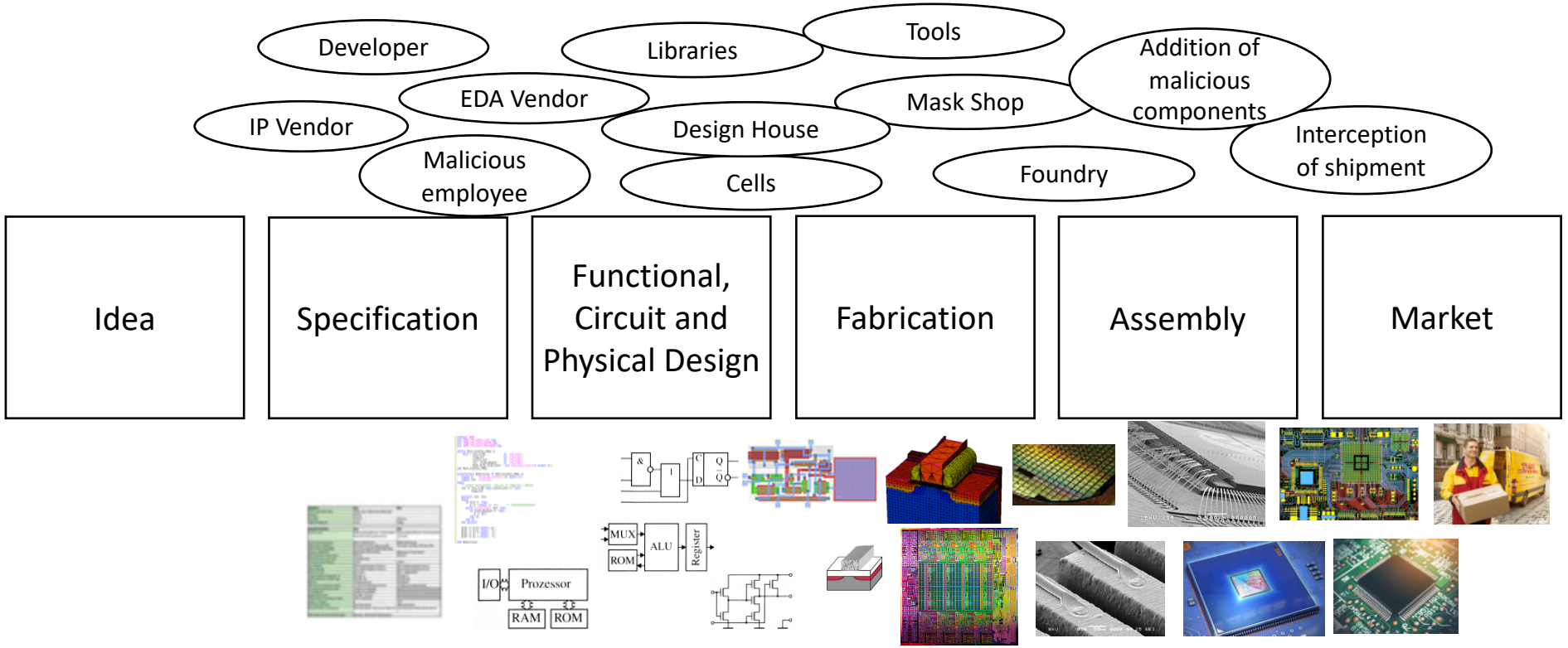
- Hidden in Integrated Circuits
- Silent most of the time: activated by rare signals and events that do not arise during design simulation and testing
- Activation mechanism: trigger
- Malicious function: payload



Outline

- What are HW Trojans?
- **Which types of HW Trojans exist?**
- How do HW Trojans look?
- How can HW Trojans be detected?

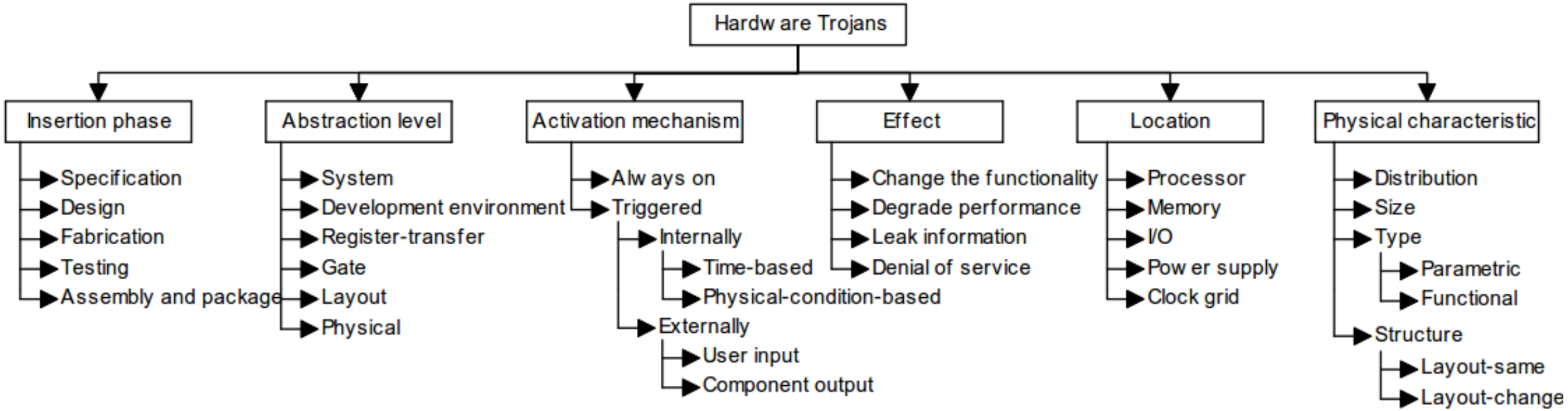
IC design and semiconductor fabrication pipeline



Trojan insertion: Why does it work?

- Outsourcing of the fabrication of the ICs.
- Use of 3rd Party IPs as an intermediate to design an SoC.
- Use of tools for electronic design automation (EDA) to manage complexity.
- Difficult to ensure the trust in all the steps of the design flow.
- A malicious circuitry can be introduced in the design by addition of small number of gates.

Taxonomy of HW Trojans (1)



Taxonomy of HW Trojans (2)

Insertion Phase: when to insert Trojan:

- *Specification:* intentionally “weaken“ specification (limited robustness / security)
- *Design:* use of untrustworthy tools, libraries, IP cores, ... (Trojan can directly be added to the circuitry)
- *Fabrication:* untrustworthy masks during fabrication, change of physical characteristics (e.g. dopants)
- *Assembly and packaging:* add another malicious component to package

Taxonomy of HW Trojans (3)

Abstraction Level: at which design level are Trojans inserted:

- *System:* change of specifications / designs (e.g. weaken PRNGs)
- *Development environment:* untrustworthy tools or scripts modify the design
- *Gates:* addition or deletion of gates, changes of timing behavior
- *Transistors:* change sizes, dopants, add wires or short-circuits

Taxonomy of HW Trojans (4)

Activation mechanisms: how are Trojans activated:

- *Always on:* no special trigger, Trojan consists only of payload
- *Internal activation:* activation upon a certain internal condition (timer, loop conditions, clock, …)
- *External activation:* activation through special input values or special ambient conditions

Taxonomy of HW Trojans (5)

Effect of the Trojan:

- *Change the functionality:* invalid functions, wrong computations, …
- *Degrade the performance:* reduce the availability of critical (real-time) systems, drain power
- *Leak information:* leak critical information (keys, passwords, code,…) to the outside
- *Denial of service:* kill switches

Taxonomy of HW Trojans (6)

Location of the Trojan:

- *Processor, memory, power supply, clock*: Trojans may be present in very different components of a system

Physical characteristics of the Trojan implementation:

- *Distribution:* location of Trojan in the IC
- *Size:* smaller Trojans are harder to detect
- *Structure:* Trojans may change the layout or not

Outline

- What are HW Trojans?
- Which types of HW Trojans exist?
- **How do HW Trojans look?**
- How can HW Trojans be detected?

Trojans: Behavioral/Functional level

- Code written in high-level language.
- Example: insertion of malicious conditions, which may trigger payload at later time.

Example:

```
counter += (value == 0x42424242) ? 1 : 0;
// some lines later ...
sum = (counter > 3) ? (sum += var) : 0;
```

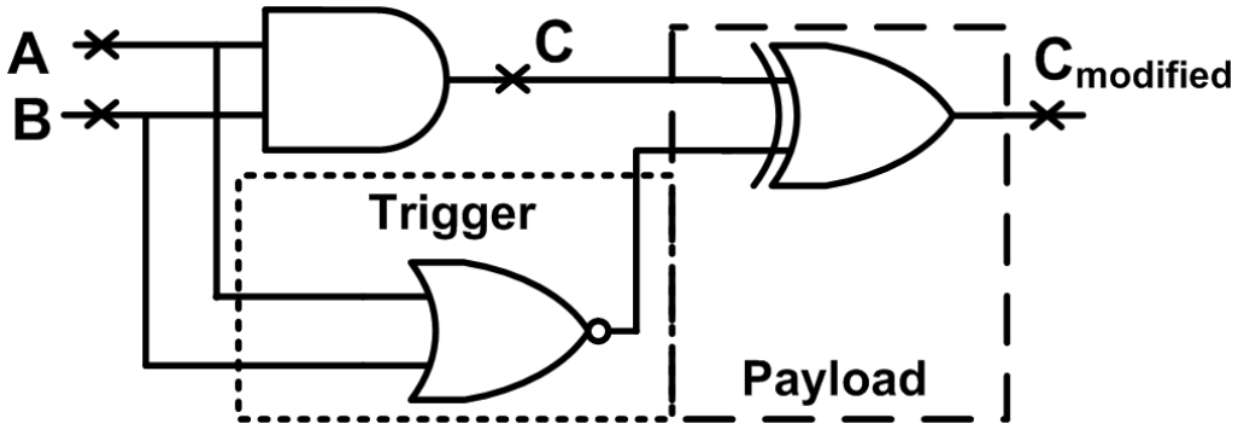
Counter only incremented if conditions met

Value of sum may later trigger payload

Trojans: Gate level (1)
Combinatorial Trigger

- Trigger implemented by additional logic
- Typically: trigger amounts to „rare“ condition that does rarely happen

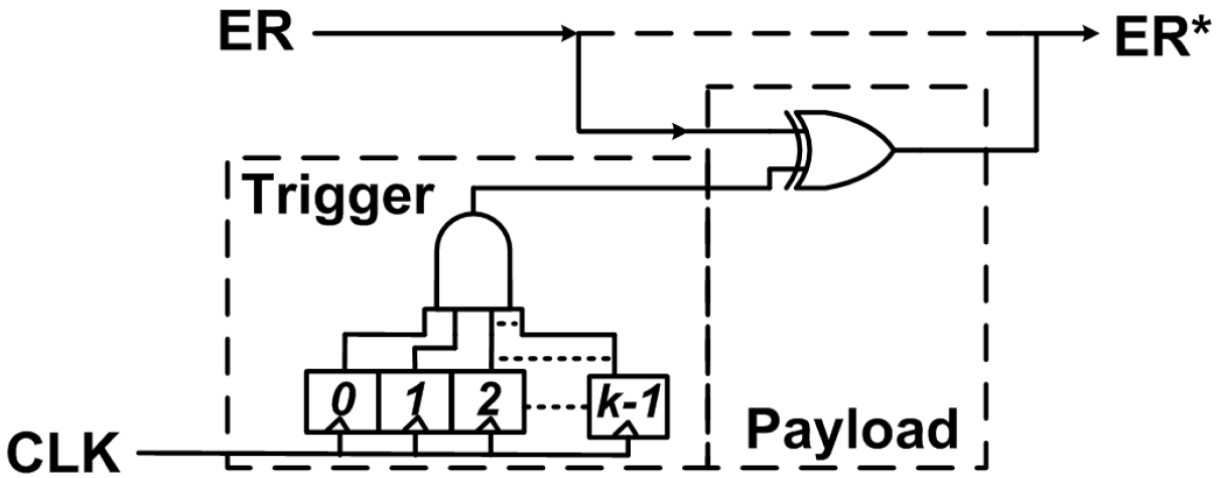
Example: if A=0 and B=0, modify output C



Trojans: Gate level (2) Synchronous Counter

- Also called “time bombs”
- Simplest implementation: counter counts up to a certain value, trigger checks for presence of certain value

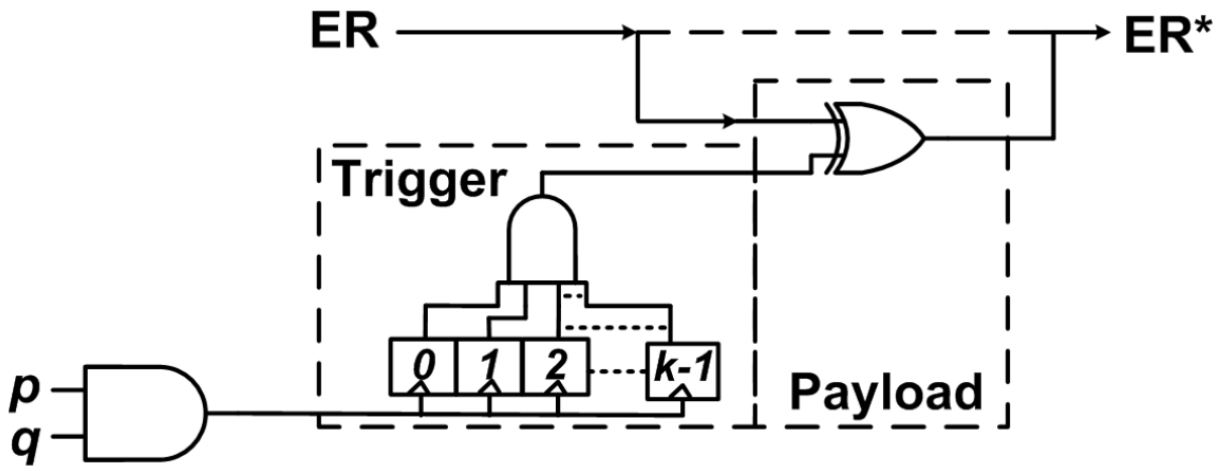
Example: k-bit counter, which modifies output once value *(2^k)-1* reached



Trojans: Gate level (3) Asynchronous Counter

- Counter not directly triggered by clock, but by presence of certain condition

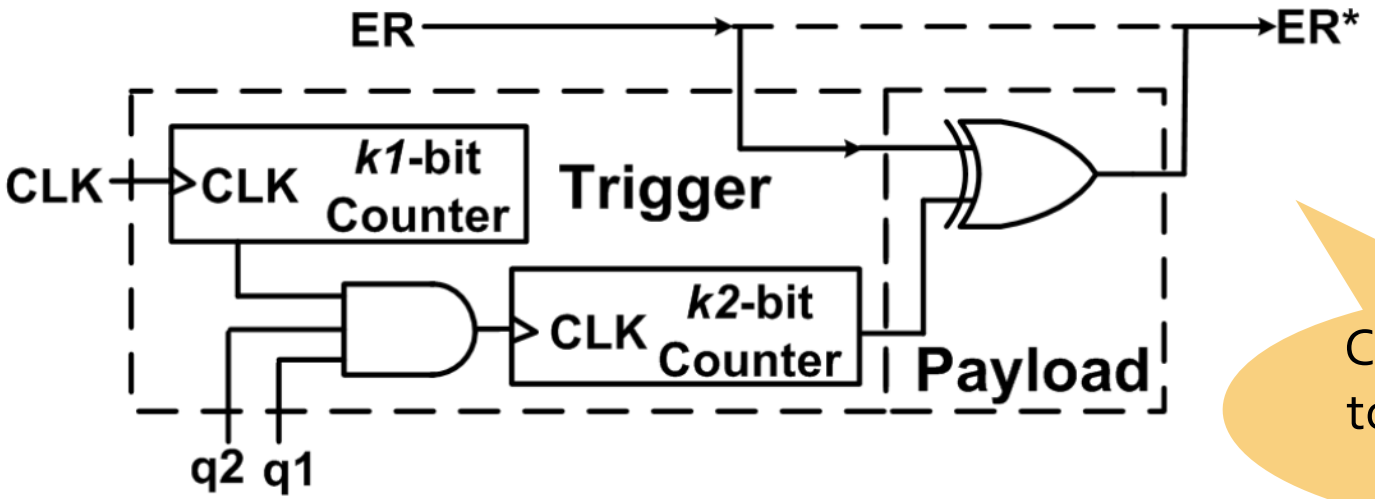
Example: Counter increments whenever *p=q=1.*



Trojans: Gate level (4)
Hybrid Counter

- Counter based both on clock and on number of occurring special events
- Combination of last two implementations

Example: Trigger depends on time and on relation between *q1,q2.*

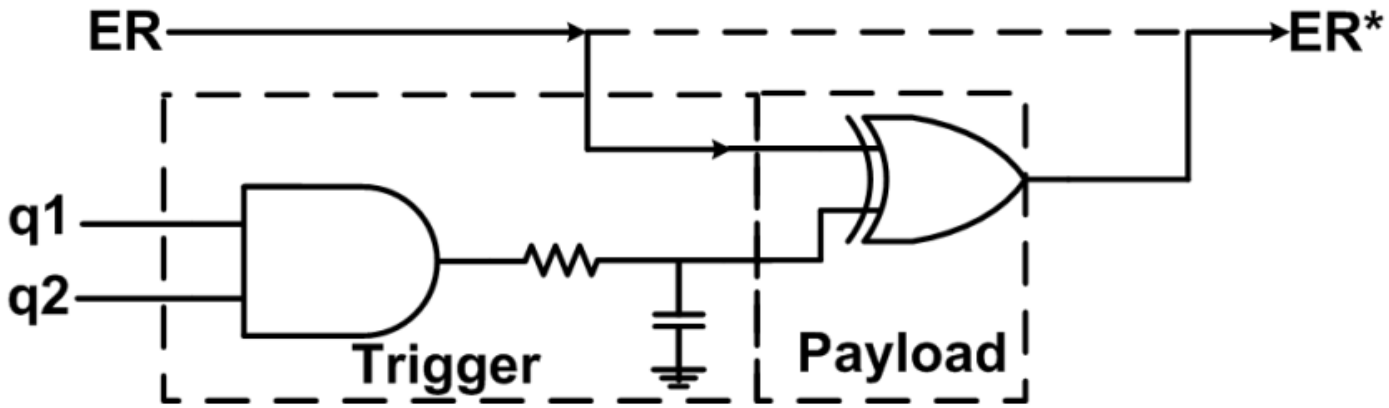


Conditions are hard to detect using test cases!

Trojans: Gate level analog triggers (1) Trigger based on logic values

- Trigger checks for condition among various variables
- If condition is met repeatedly, a capacitor is charged, which will ultimately result in a logic 1

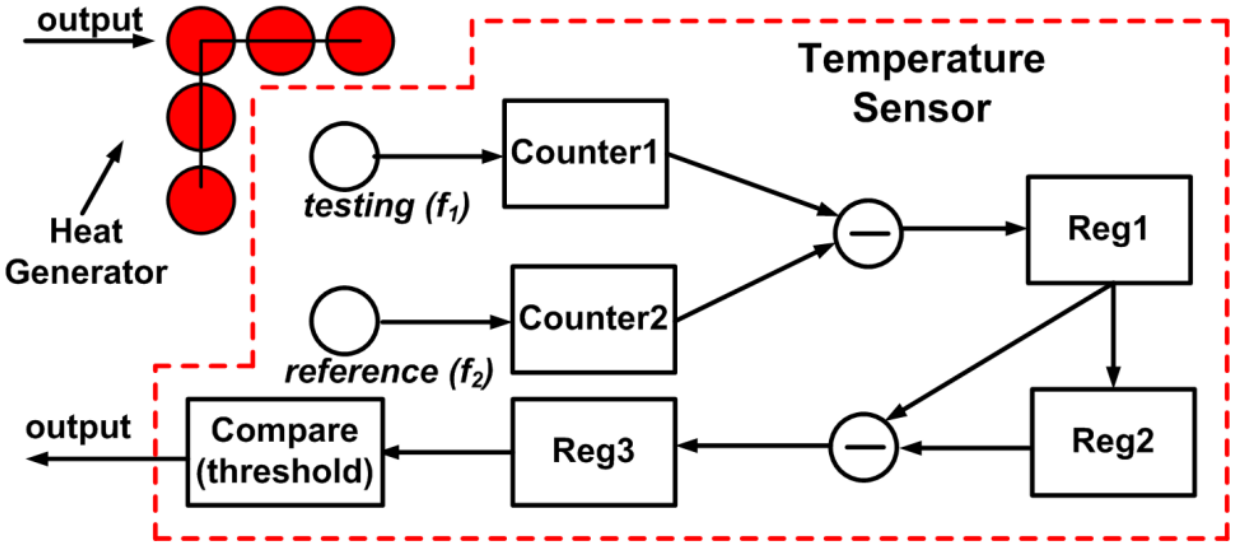
Example:



Trojans: Gate level analog triggers (2) Trigger based on sensed values

- Sensor checks for environmental conditions
- If conditions are met, payload is triggered

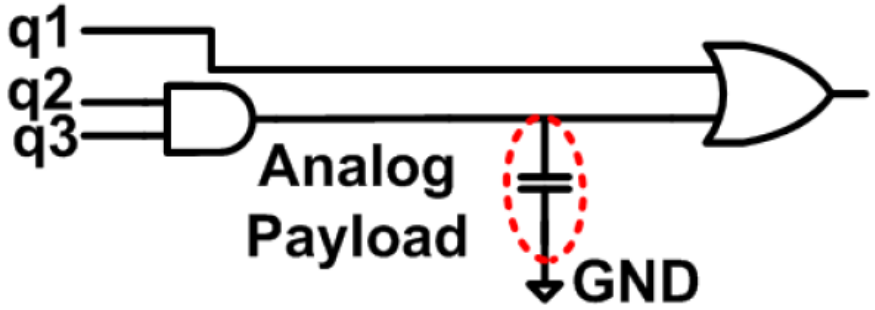
Example: Ring-oscillators as “temperature sensor”

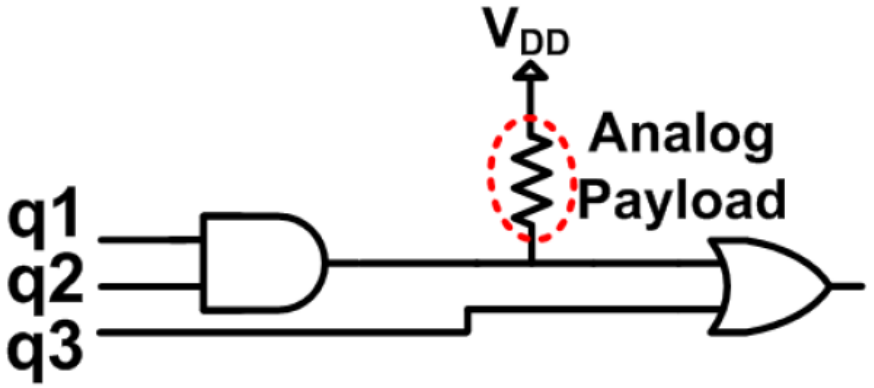


Trojans: Gate level analog triggers (3) Analog payload

- Deliberately insert “errors” (such as short circuits) into the system
- Example: connect wires permanently to GND or VDD
- May intentionally “disrupt” algorithms; may change delays

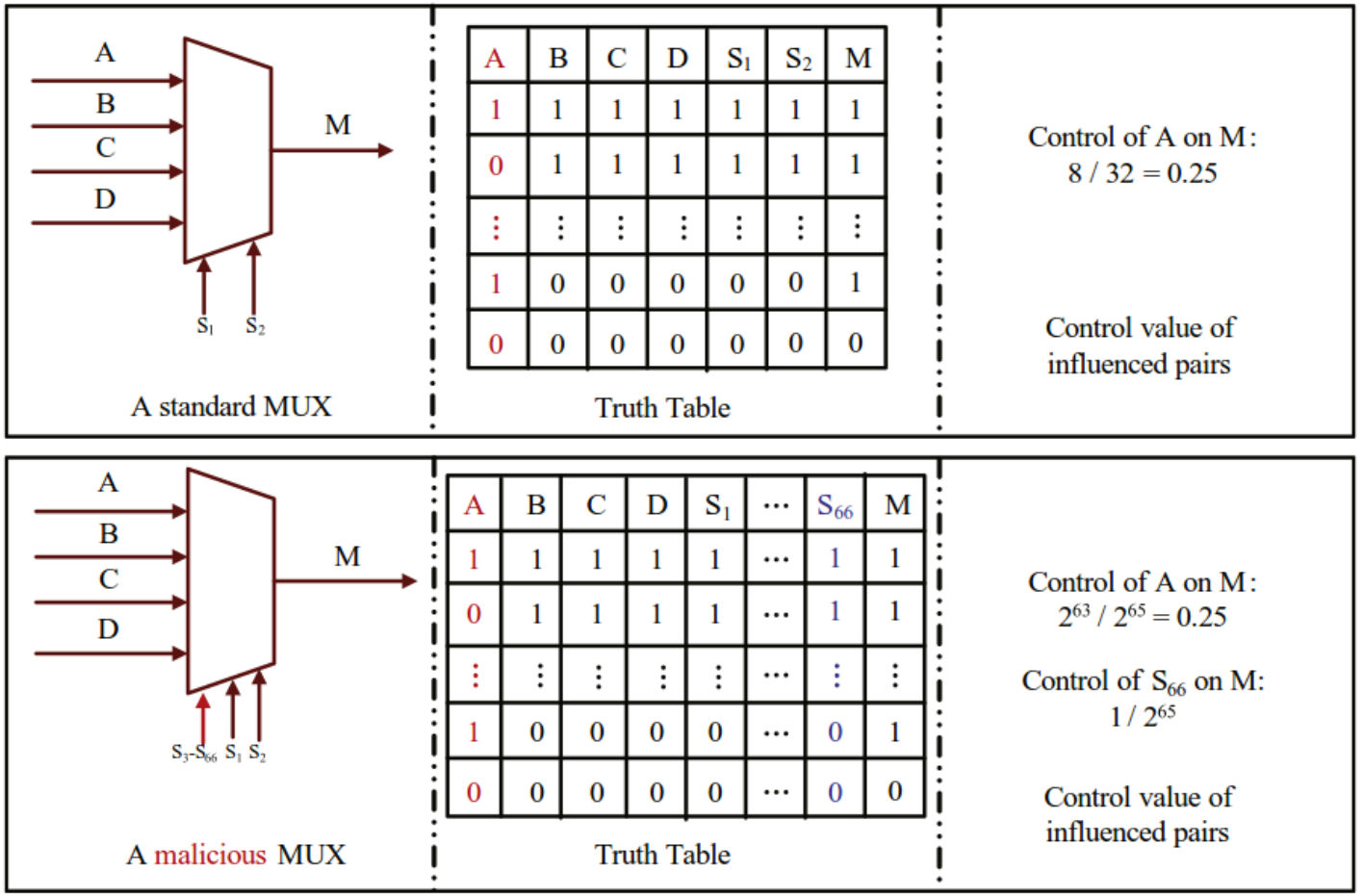
Examples:





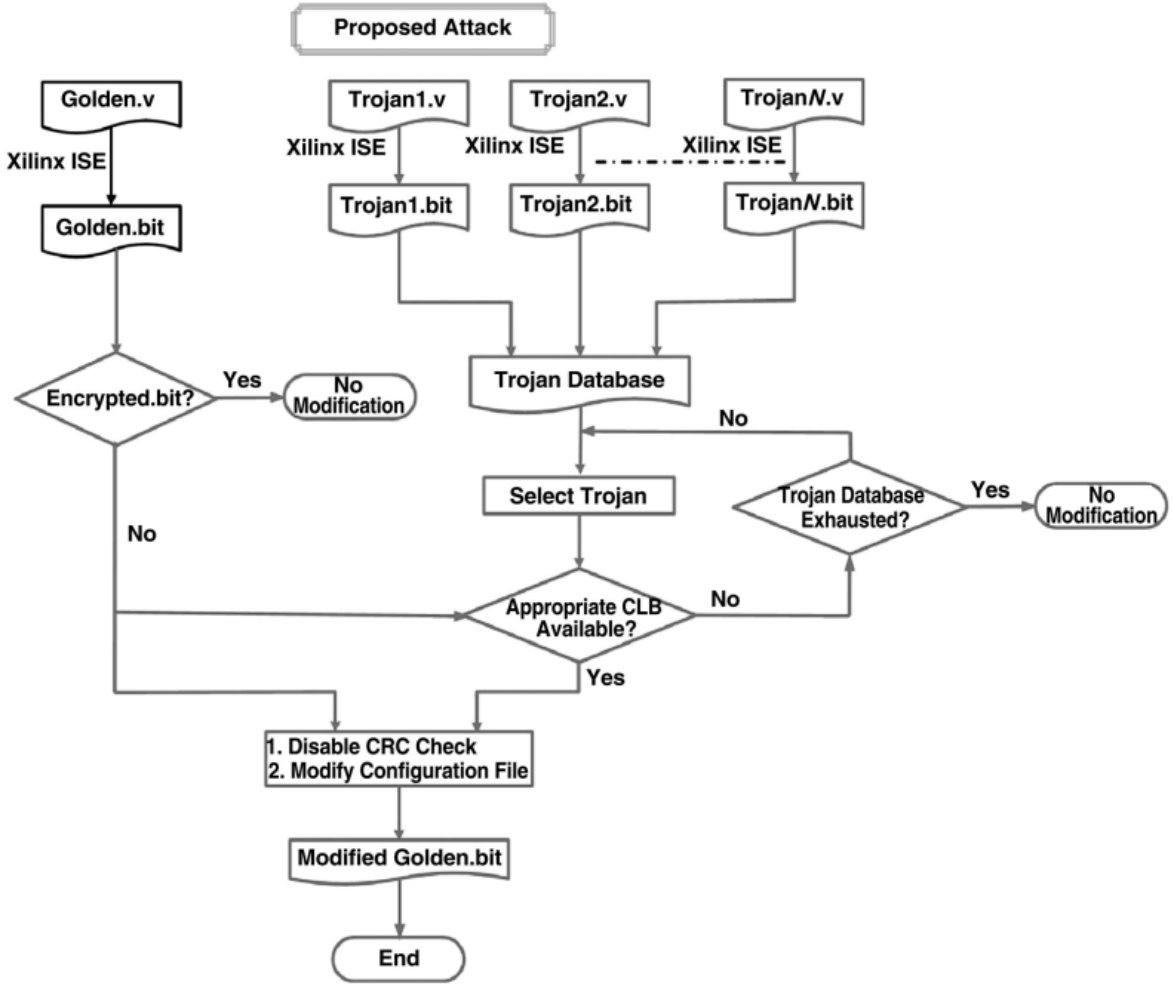
Gate Level Trojans: Multiplexer

- Idea: increase the number of multiplexer inputs
- Alter behavior of the MUX if *one certain* value is present in the superfluous wires
- Hard to find test cases!



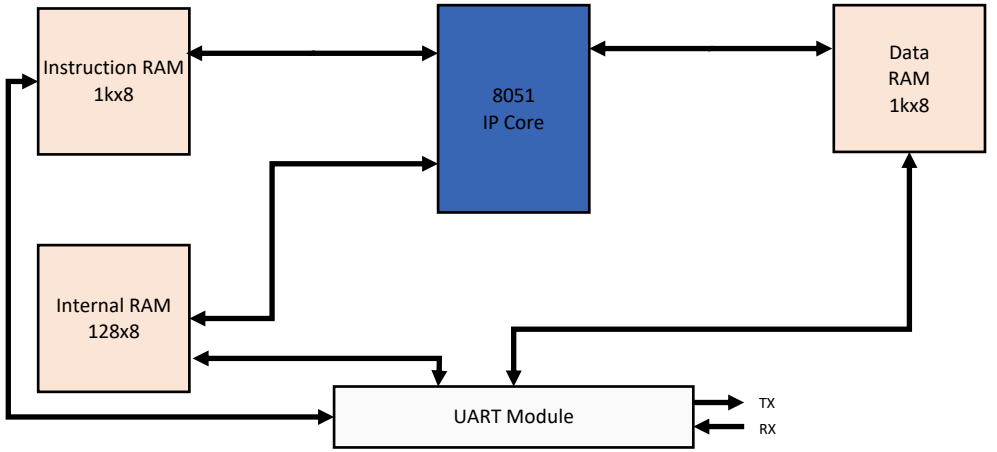
Trojans: Addition Through Malicious Synthesis Tools

- Example of a modified FPGA synthesis tool.
- Tool takes unmodified design…
- … and iterates over a database of possible Trojans.
- Tool automatically adds Trojan and disables integrity checks.



Trojans: Addition during System Design, Example (1)

- Hardware Trojan in an Intel 8051
- Assumption: Malicious Designer
- Architecture is used in some USB µControllers
- Purpose of Hardware Trojan:
  - Detect start of RC5 encryption
  - Store store secret key
  - Transmit secret key

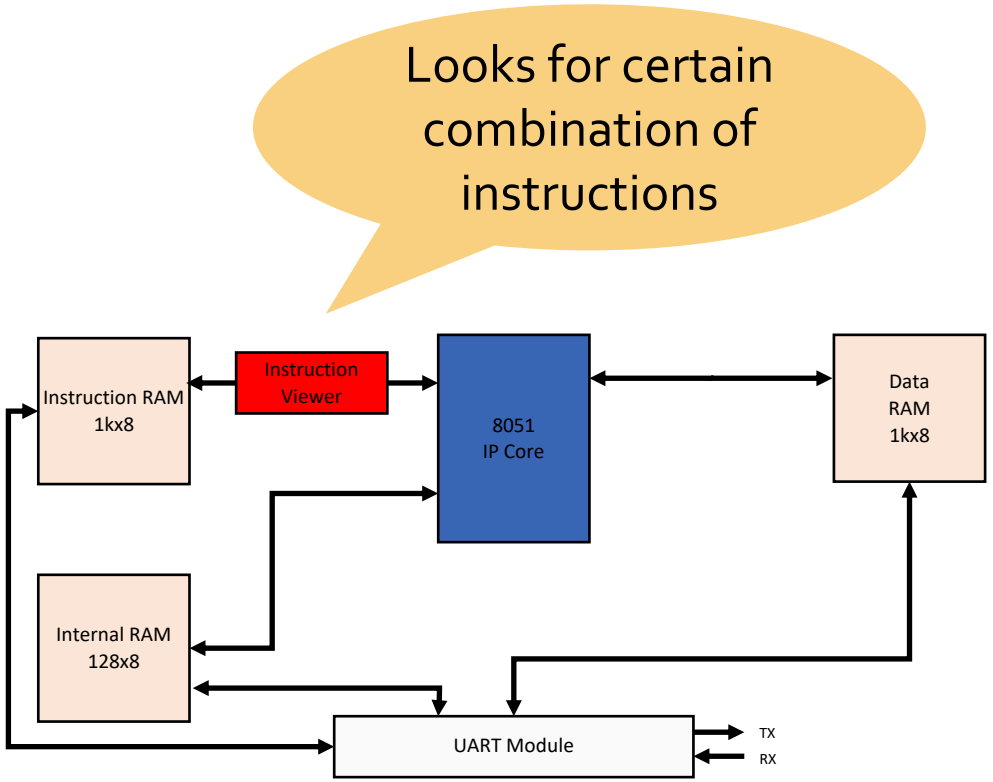


Trojans: Addition during System Design, Example (2)

- Trigger: Detect RC5 encryption
- Based on RC5 algorithm

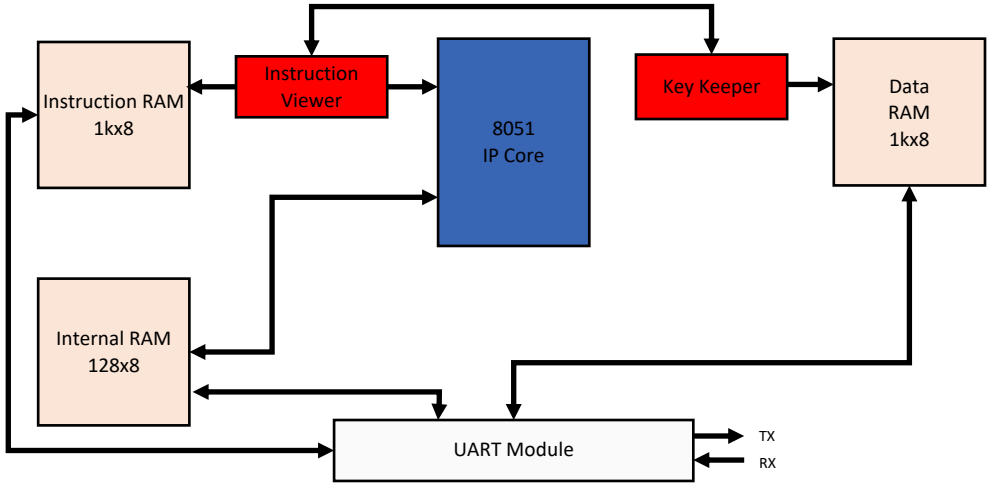
```
A = A + S[0];
B = B + S[1];
for i=0 to r do
   A = ((A ⊕ B) <<< B) + S[2*i];
   B = ((B ⊕ A) <<< A) + S[2*i+1];
```

- Detect access to extended key
  - Look for instructions: `mov` followed by `add`



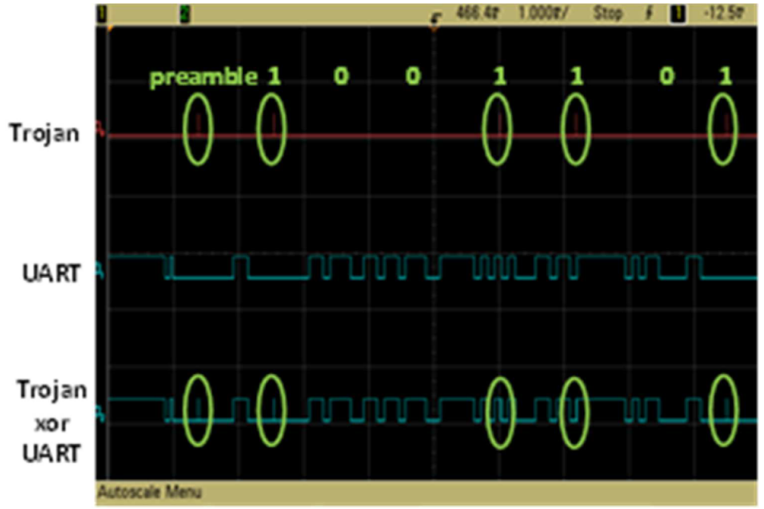
Trojans: Addition during System Design, Example (3)

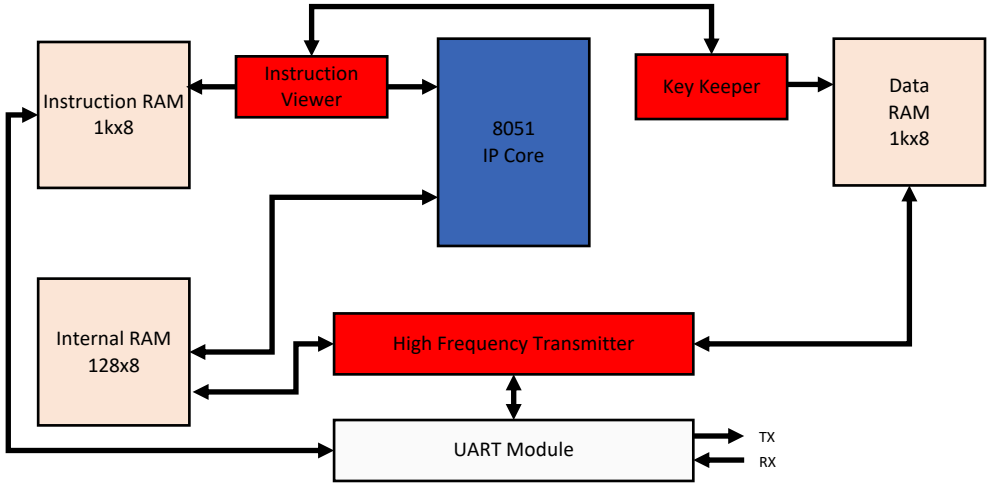
- Store (extended) key
- Reuse of processor memory
  - Low area overhead
  - Requires control circuitry
  - Controlled by instruction viewer on successful interception of key
  - Otherwise: all reads/writes to memory are passed through otherwise



Trojans: Addition during System Design, Example (4)

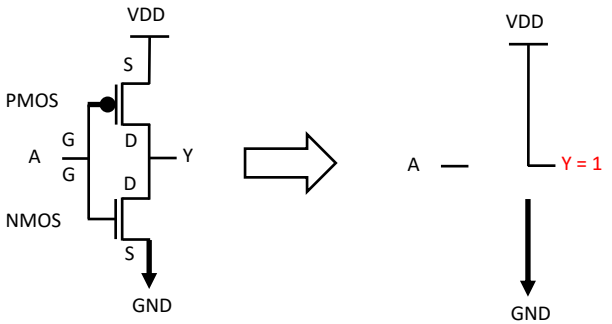
- Transmit key by reusing serial link
- Data construction should be easy
- Original link should not be disturbed
  - Addition of high frequency signal
- Final design including HWT
  - Total area overhead: 2%





Trojans: Physical Level (1)
Dopant Trojans

A	Y
0	1
1	0

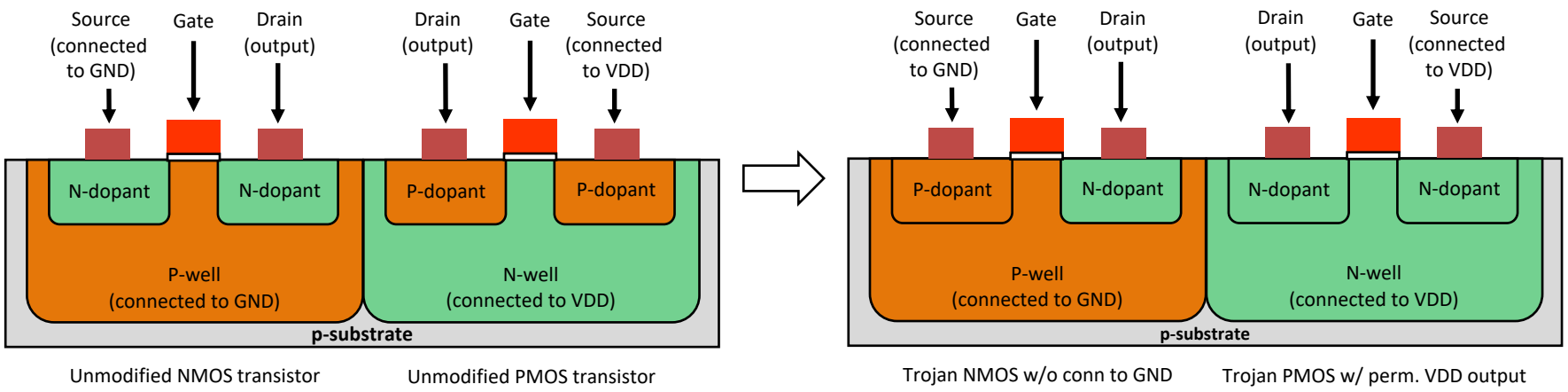


PMOS Transistor permanently closed

NMOS Transistor permanently open

- Example: CMOS inverter consisting of NMOS and PMOS transistors
- Modifications change behavior of circuit in predictable way
- Inverter gate always outputs $V_{DD,}$ logical 1

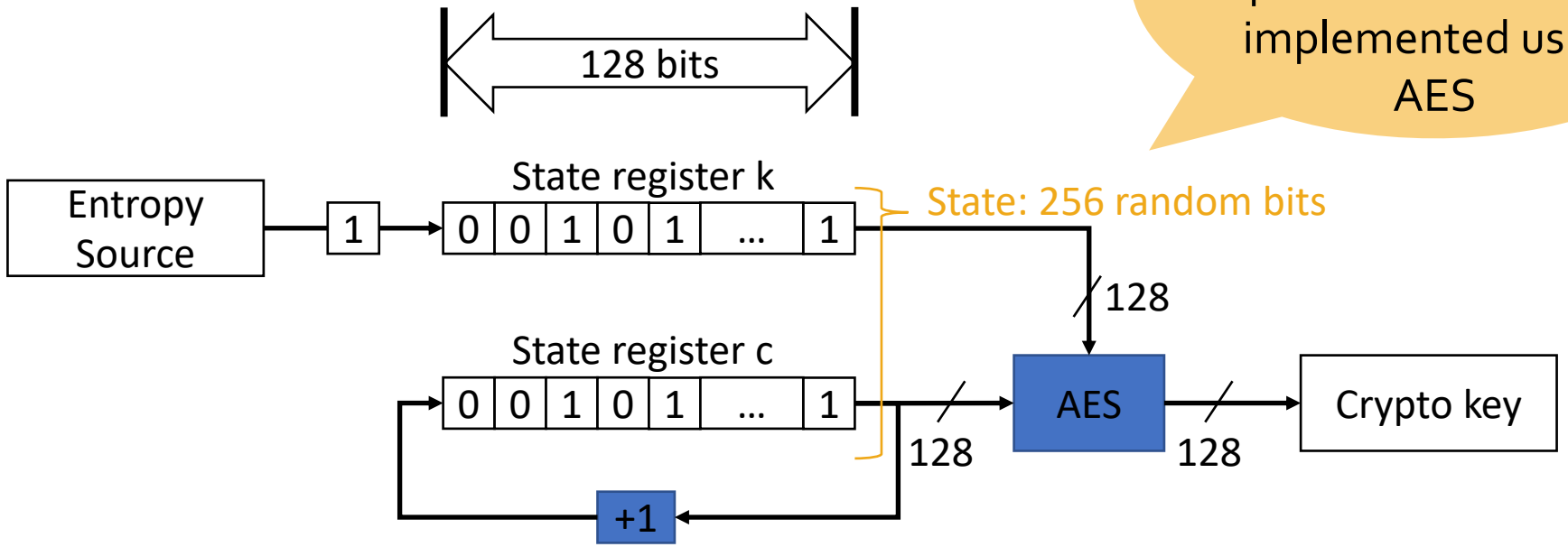
Trojans: Physical Level (2)
Dopant Trojans



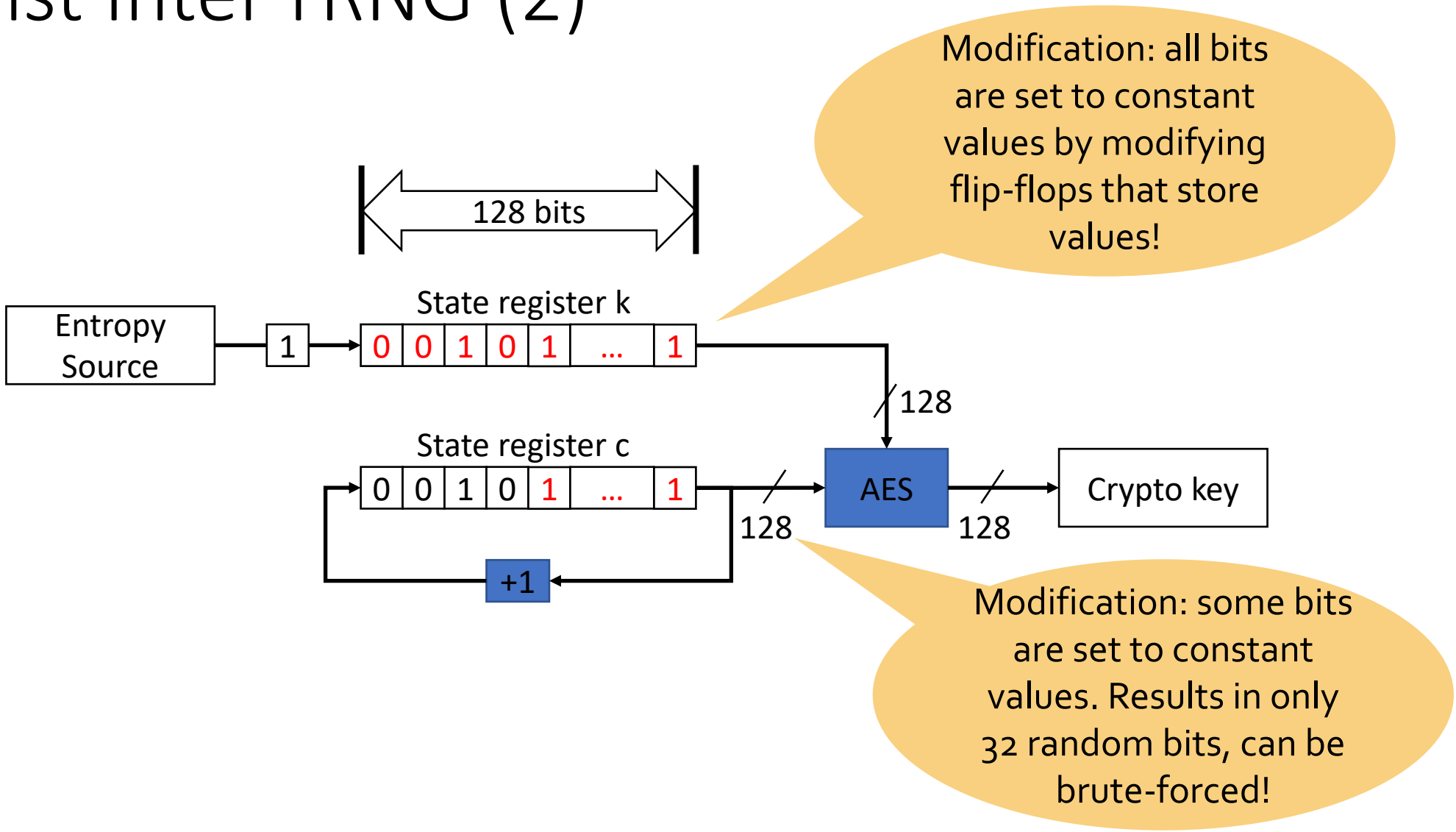
- Gate is modified by applying different dopant polarity to parts of gate's active area
- Constant connection between PMOS drain and V_{DD}
- Interrupted connection between NMOS drain and GND
- Hard to detect in practice!

Example: Physical-level attack against Intel TRNG (1)

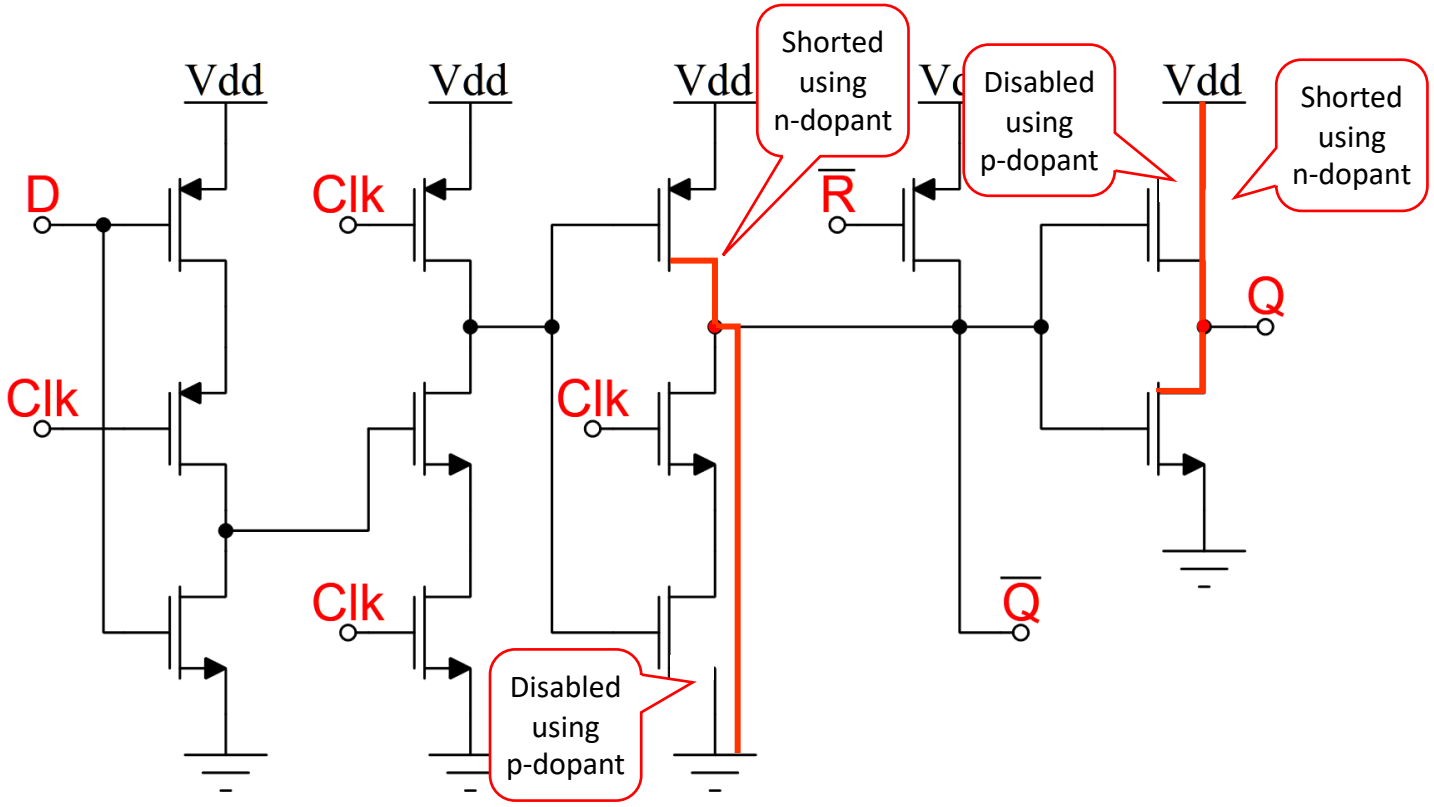
Attack against the "conditioning" phase of the TRNG implemented using AES



Example: Physical-level attack against Intel TRNG (2)

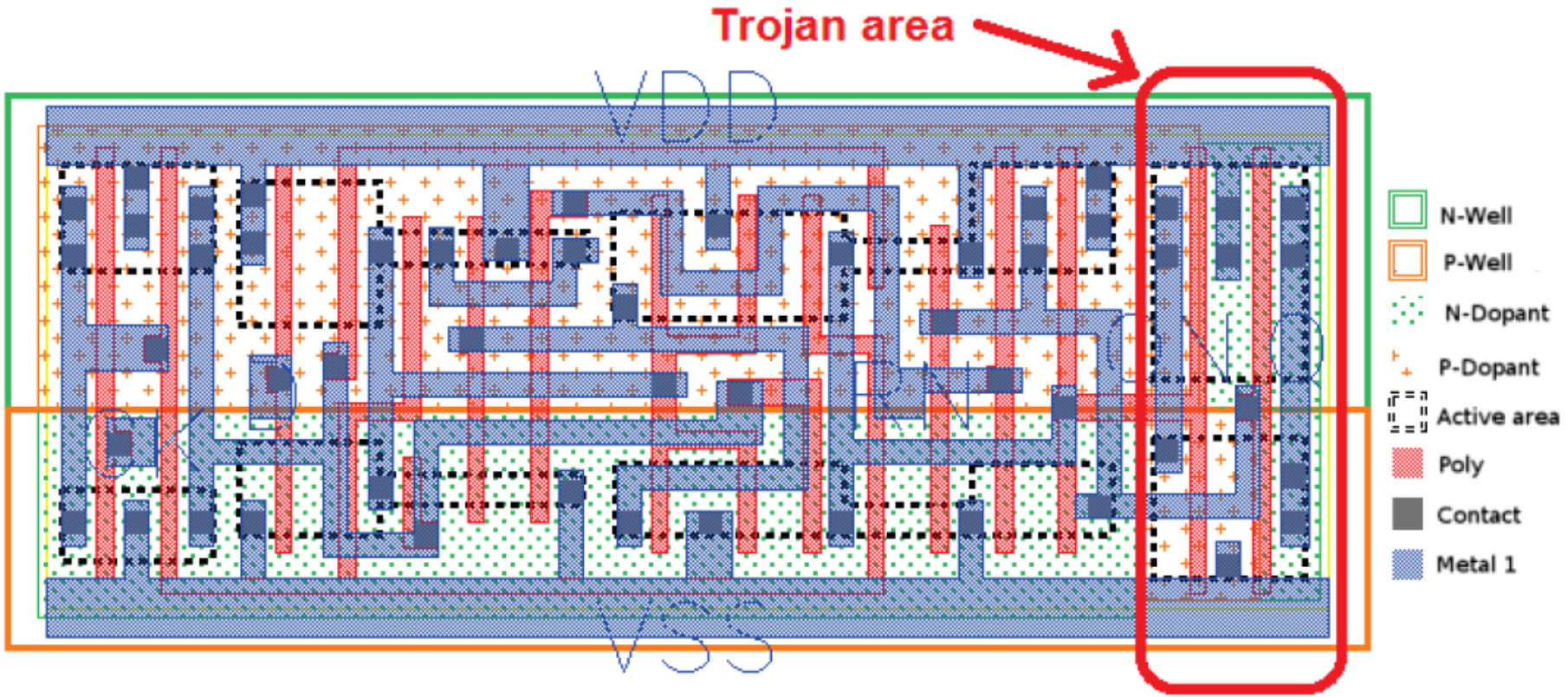


Example: Physical-level attack against Intel TRNG (3)



- Modification of a flip-flop so that it always stores a one
- Analogous: storing a ONE by changing roles of pMOS and nMOS

Example: Physical-level attack against Intel TRNG (4)



Outline

- What are HW Trojans?
- Which types of HW Trojans exist?
- How do HW Trojans look?
- **How can HW Trojans be detected?**

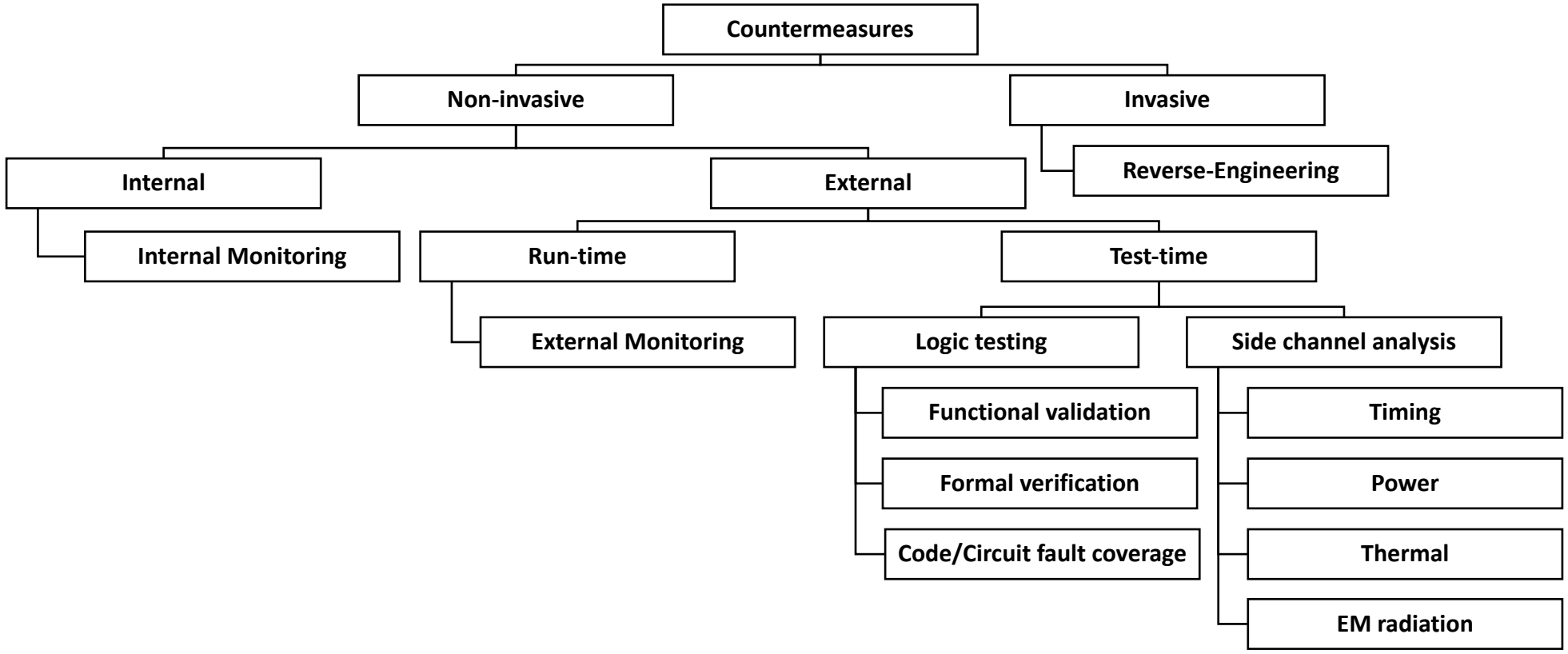
Countermeasures: Disclaimer

- No “silver bullet“ method available yet to detect all classes of Trojans
  - Trojans can get arbitrarily complex!
- Most methods
  - assume a particular Trojan model
  - cannot guarantee detection → provide confidence levels
  - prone to false positives
  - lack of resolution to outright show Trojan location
  - have not been validated experimentally (only simulations, homebrew Trojans)
  - have unacceptable overhead

Types of Countermeasures

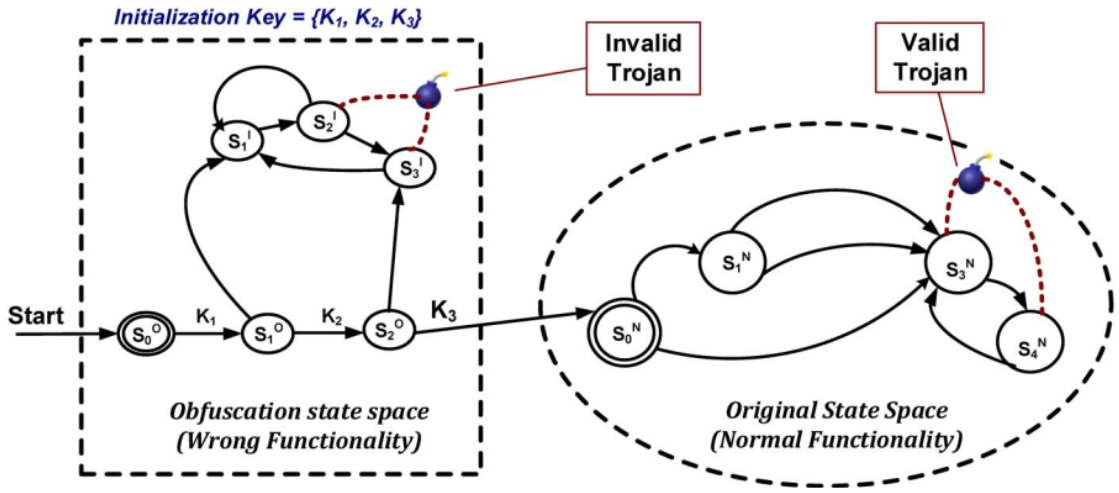
- Detection
  - determines whether any HWTs exist in the circuit
- Diagnosis
  - distinguishes the HWTs in the ICs in terms of their types, locations, and characteristics
- Prevention
  - increases the difficulties of HWT insertion
  - enhances the efficiency of the HWT detection and diagnosis methods
  - prevents the successful HWT insertion during IC development, provide trust

Taxonomy of Countermeasures



Countermeasures: Prevention

- Prevent free space in an IC/FPGA
  - Insertion of Trojans requires space
  - Does not help if attacker uses better logic optimization and placement techniques

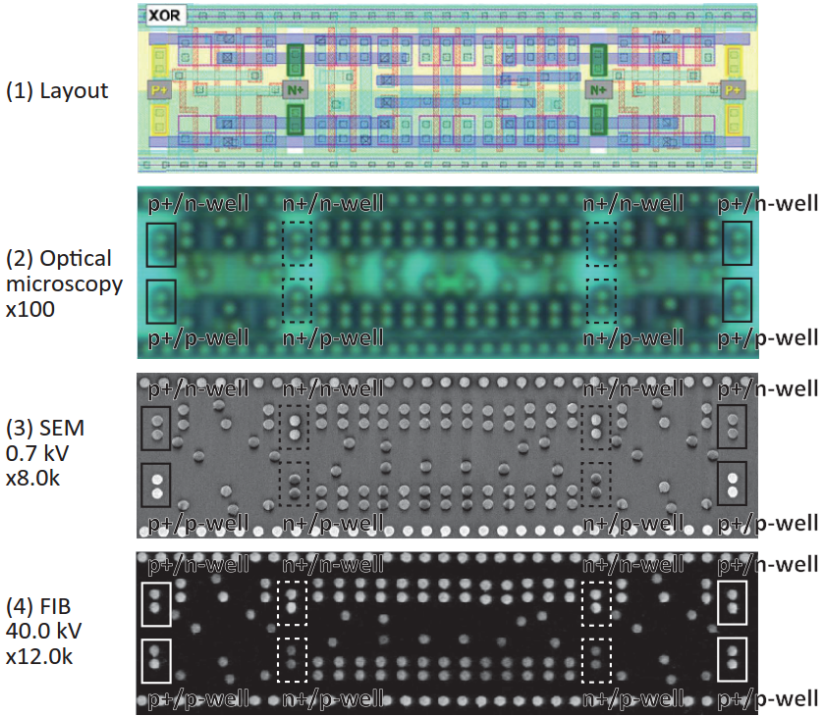


- Obfuscation of circuit functionality
  - At design stage, the attacker requires an estimation of signal probabilities in order to bind the trigger to rare signals
  - Effectiveness depends on attacker position and abilities

Countermeasures: Destructive / Reverse-Engineering

- Use sample of manufactured ICs
  - Perform de-metallization using Chemical and Mechanical Polishing (CMP)
  - Image re-construction and analysis using Scanning Electron Microscope (SEM)

- Limitations:
  - Extremely expensive and time-consuming (single chip takes several months)
  - Doesn't scale well → transistor integration density
  - Results of sample cant be extrapolated to all ICs
  - Adversary might have affected only a small number



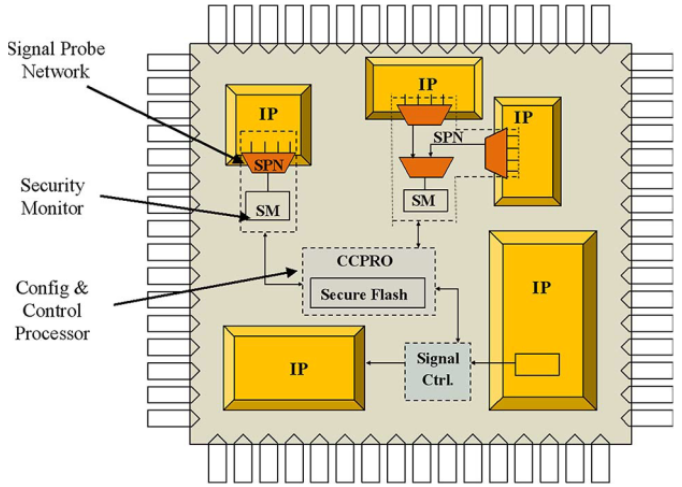
Countermeasures: Runtime Testing (1)

Transparency of operation

- Key-enabled special mode on demand, controls rare events
- Probabilistically leads to Trojan detection when it disrupts operations in mode
- Cannot guarantee Trojan detection

Hardware assisted functionality monitoring

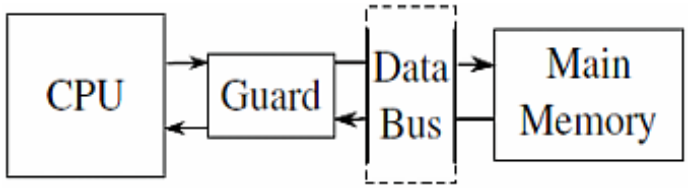
- Reconfiguration framework for run-time functionality monitoring
- Triggers counter-measures on deviation
- Commercial design tools to implement methodology
- Overhead unknown



Countermeasures: Runtime Testing (2)

Hardware + Software Approach

- Addition of “Hardware guard” module outside CPU + enhancement for OS
- Effective against DoS and privilege escalation attacks
- 2.2% average performance overhead for SPECint 2006 benchmarks



BlueChip

- Identification of unused circuit portions in pre-fab as potential Trojans
- Portions replaced by exception generation HW (EGHW)
- When activated, EGHW delivers exception to software layer
- SW emulates instruction that generated the exception
- 5% run-time overhead, 1.5% area overhead, 0.5% power overhead for a FPGA
- Challenge: Based on verification, complete circuit behavior coverage difficult

Countermeasures: Logic Testing

Multiple Excitation of Rare Occurrence (MERO)

- Complete enumeration of all possible Trojans infeasible
- Difficulty of exciting multiple nodes at their rare values
  - Manufacturing defects (stuck-at-faults) ≠ HT effects

- Approach
  - Enumerate rare nodes in a given netlist (Search using program, filter false-positives)
  - Excite potential Trojan trigger nodes multiple times to their rare values individually
  - Generate a compact set of test vectors

- Bypass difficulty of directed test generation to trigger Trojans
- Limitations:
  - Limited to a class of Trojans
  - Statistical technique → cannot guarantee 100% detection coverage

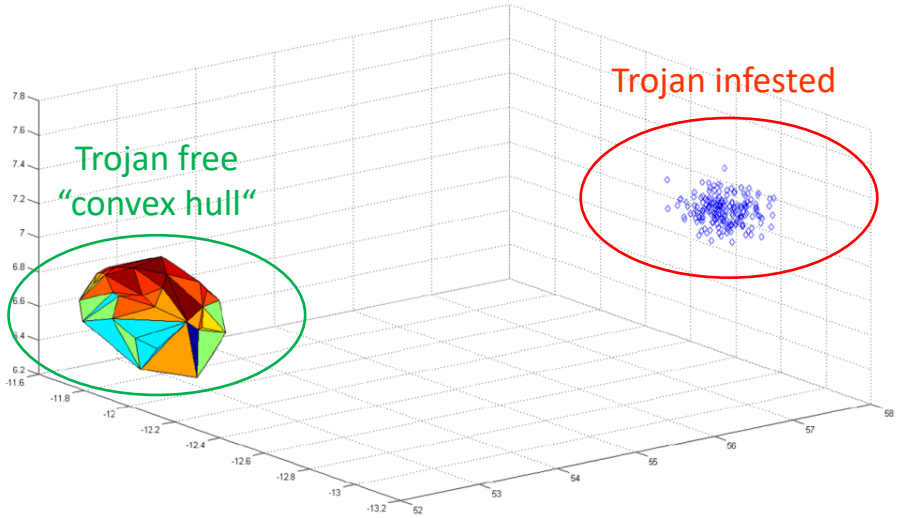
Countermeasures: Fingerprinting through Side Channels (1)

Core idea: IC fingerprinting

- Signature (fingerprint) associated with IC
- Uses „golden model“:
  - represents the anticipated behavior of HT-free IC
  - If the suspect IC deviates sufficiently from golden model, it is considered HT-infected
  - difficult to acquire with the increasing complexity of ICs
- Usually path delay or power trace, supplemented by de-noising techniques
- Can detect Trojans with 0.01% of circuit area having ±7.5% process variation
- Limitations:
  - So far only simulation results
  - No actual measurements and real noise

Countermeasures: Fingerprinting through Side Channels (2)

- Path-delay Fingerprint
  - Multiple paths considered
  - Extensive statistical characterization
  - Detects Trojans with 0.13% area, under 7.5% process variation



- Gate-level Characterization
  - Path delay + leakage current considered
  - Effective for smaller circuits
  - Limitation: Computationally challenging for larger circuits

Summary: Hardware Trojans

- Modern IC design and manufacturing practices are inherently insecure
  - Third-party IPs and off-shore manufacturing
  - Potentially untrusted parties pay a major role, trend likely to increase
- Hardware Trojans are malicious circuit modifications
  - Small overhead, hugely destructive impact
  - Difficult to detect by traditional testing means
- State-of-the-art
  - Both design and detection techniques have been proposed
  - Effectivity of detection techniques limited to the particular types of Trojans
  - Most techniques have not been validated experimentally in-field