CAD for Security

Mark Tehranipoor and Farimah Farahmandi

Florida Institute for Cybersecurity Research
University of Florida

Link to Slides:
http://farimah.ece.ufl.edu/cadforsecurity

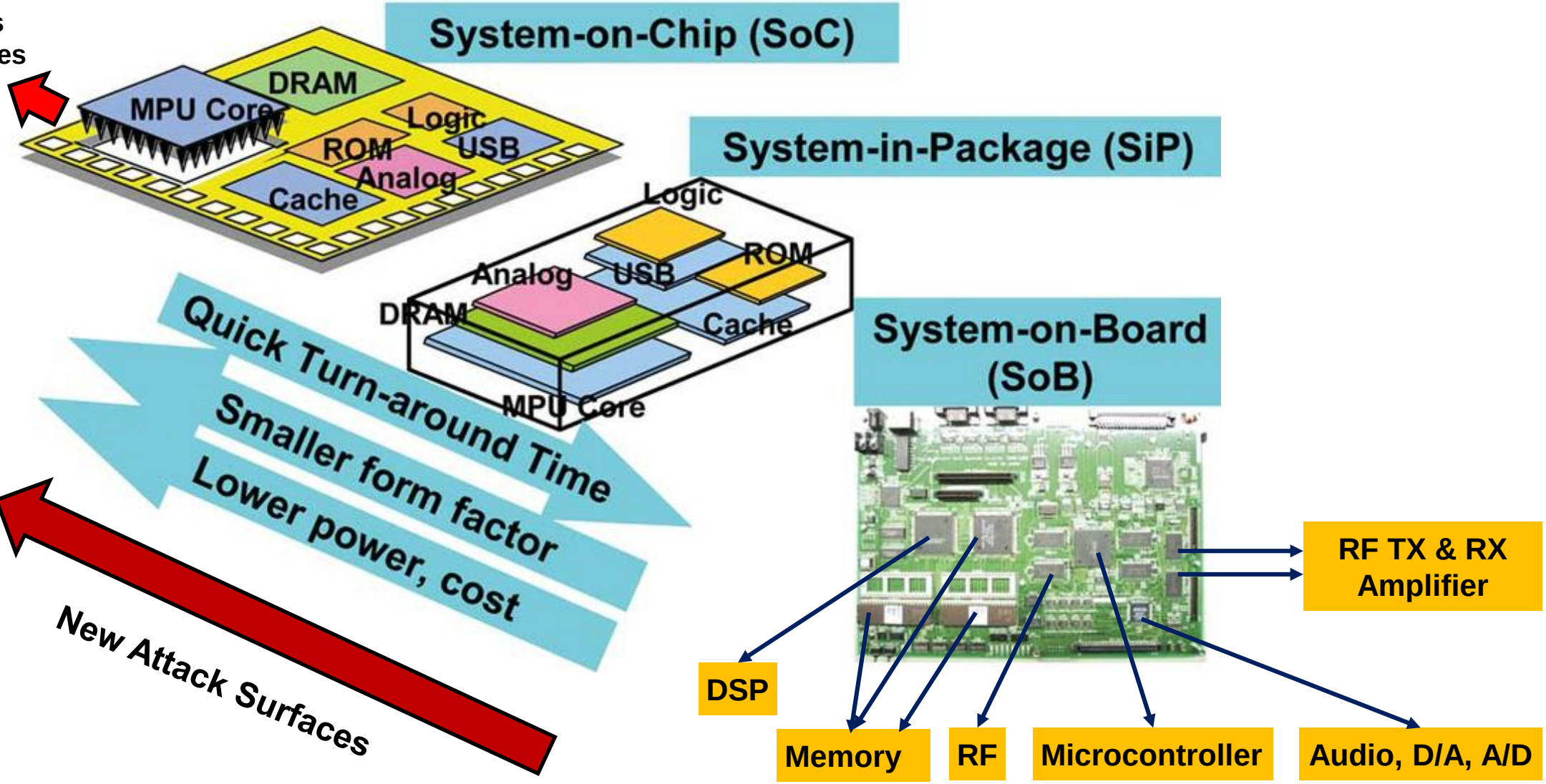
Useful information:
WiFi Network: Hilton-Meeting
Password: HOST2019

Outline

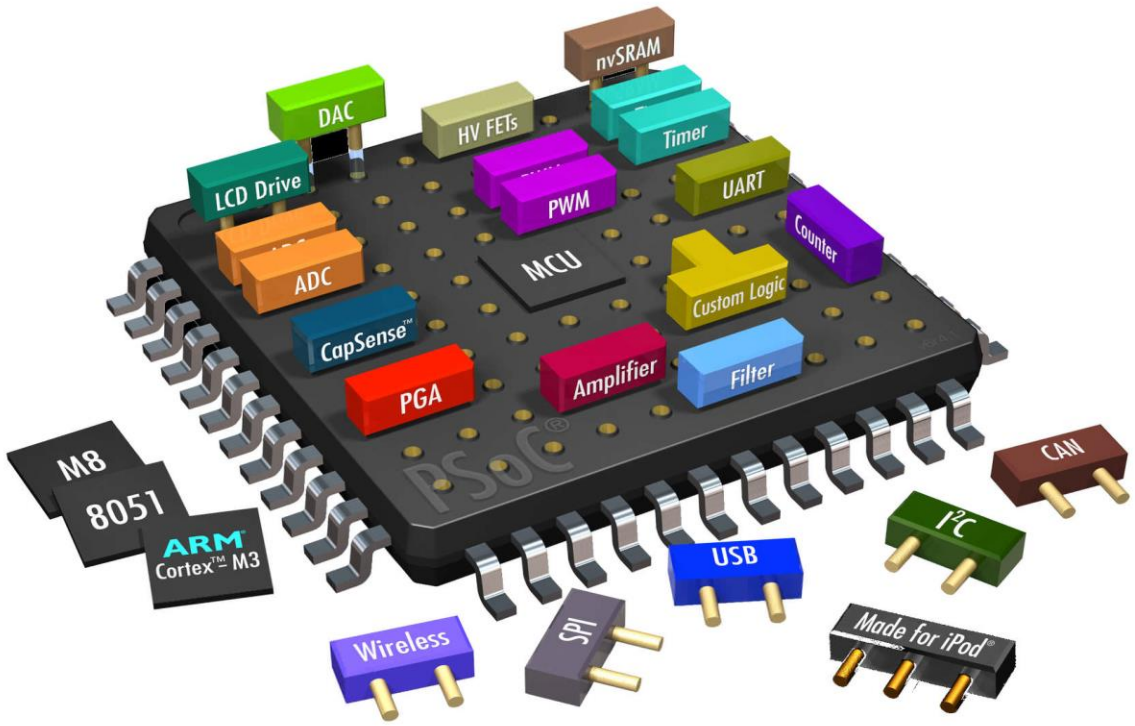
- Problem Statement
- Design Flow
- Supply Chain
- Hardware Attacks
- Need for Automation
- CAD for Security
- Challenges

VLSI Integration

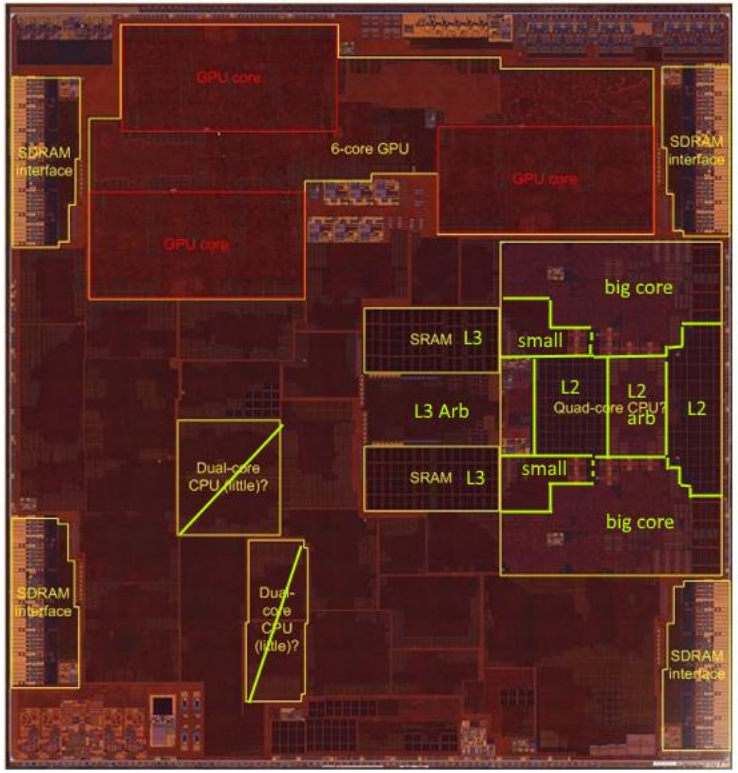
IoT Devices
Mobile Devices



Modern SoCs – Heterogeneous Architecture



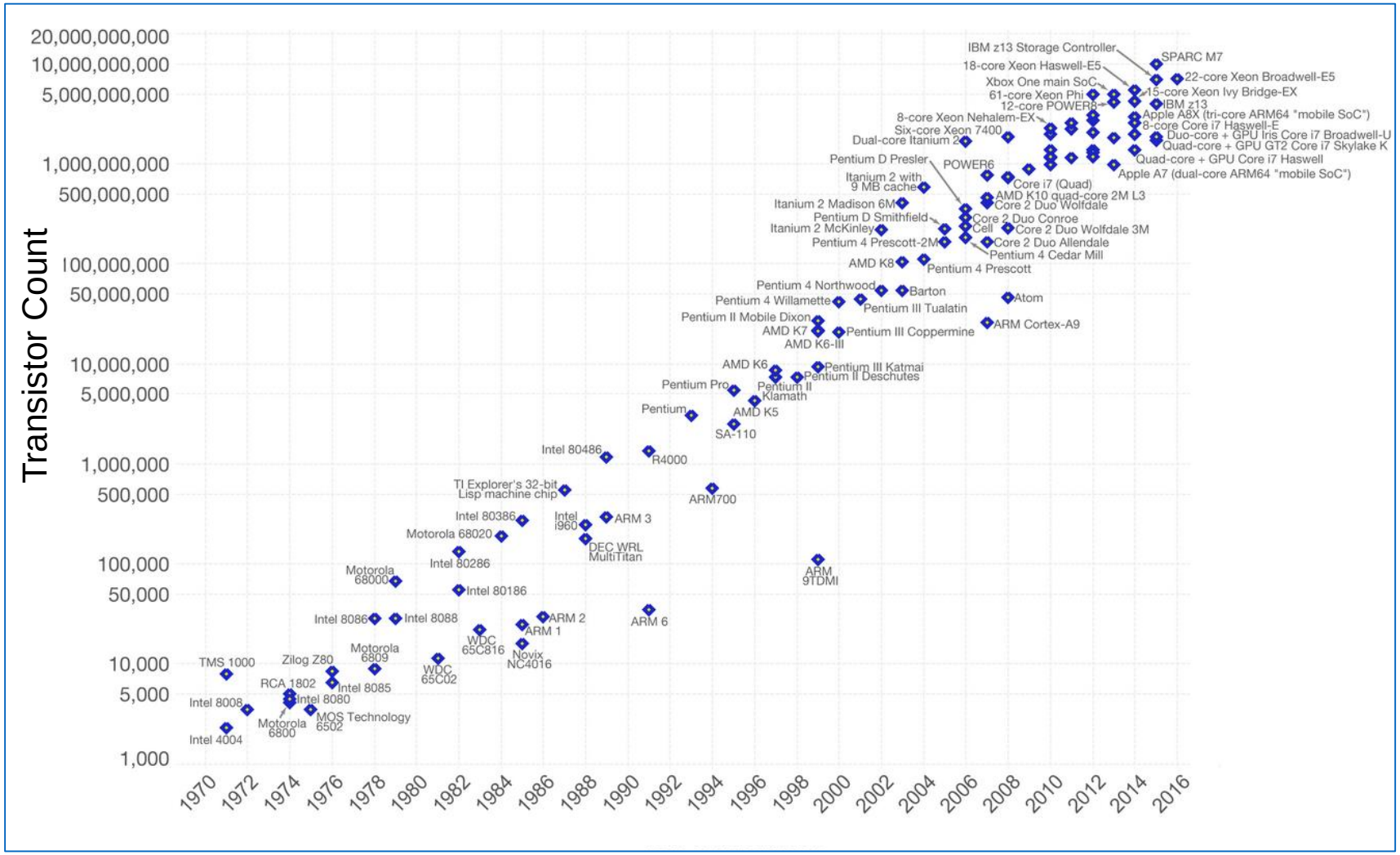
Apple A10 Quad Core SoC



- TSMC's **16 nm** FinFET
- **3.3 billion** transistors
- Die size: 125 mm^2

@Chipworks

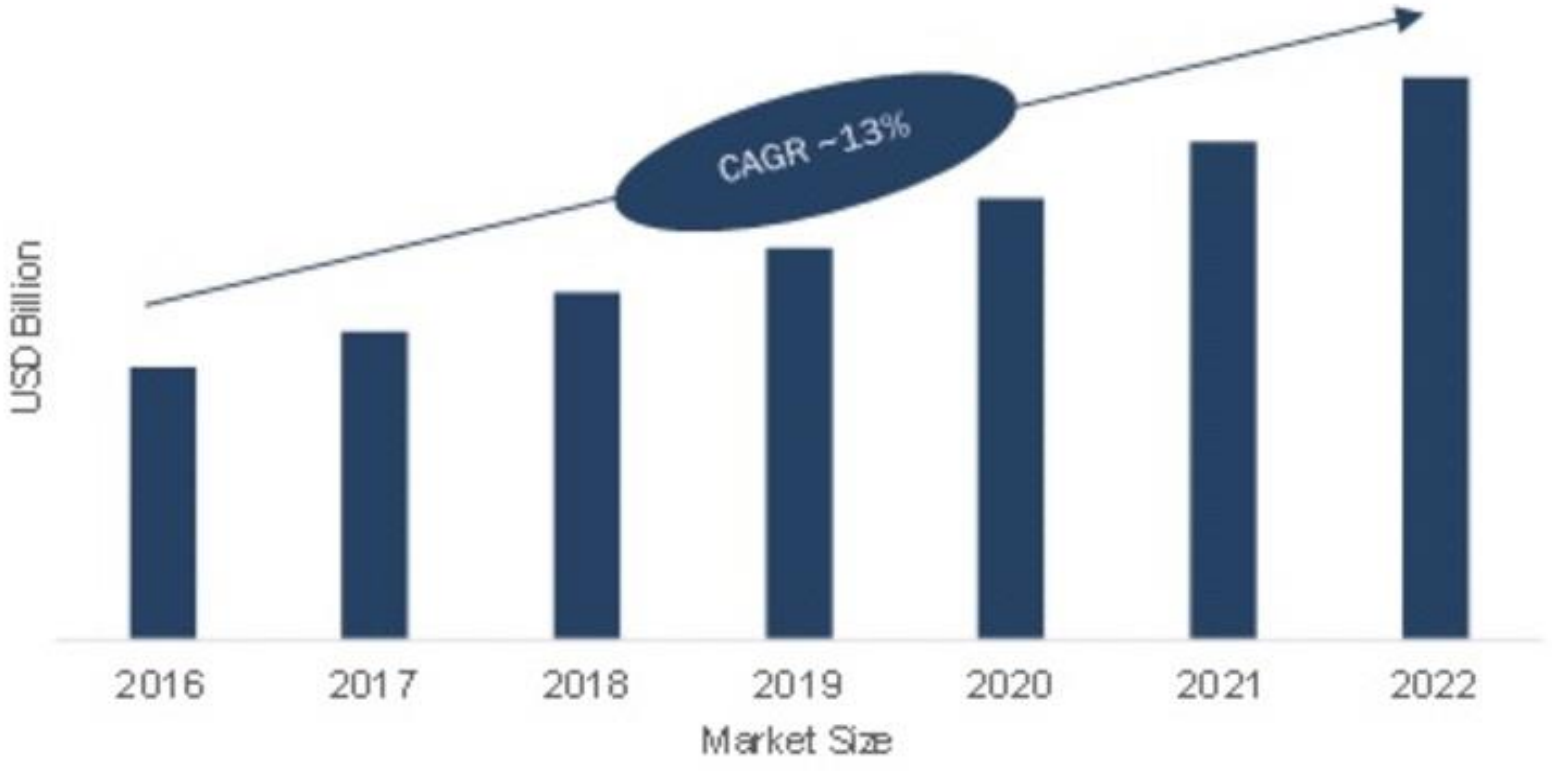
SoC's Growth



Year of Introduction

SoC's Growth

SoC Market Size



Design Challenges

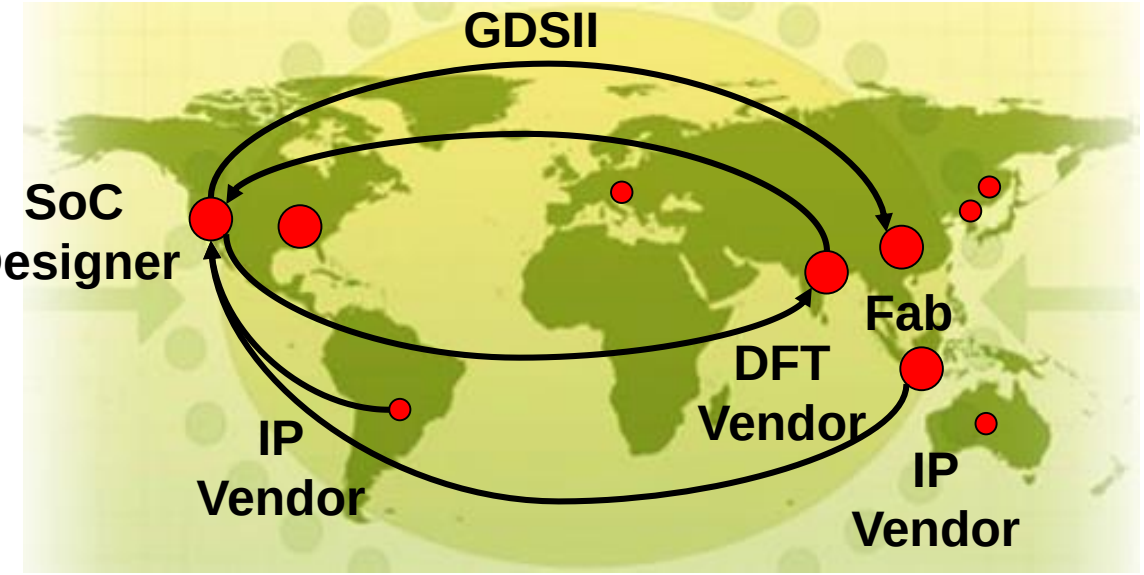
- **High complexity of devices**
  - Tens of billions transistors
- **Aggressive time-to-market requirements**
  - Severely constrains functional validation → vulnerability escapes to silicon or in-field
- **High diversity in computing devices**
  - Security requirements **vary** significantly
  - Cannot be “pre-verified” at the IP level
- **Connectivity**
  - More SoCs being connected → not originally designed to be connected

Contains 3.3 Billions of transistors

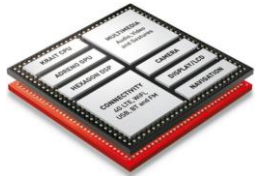
Shrunk to less than a year → mobile device

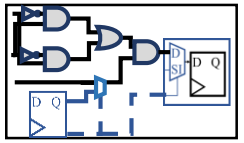
Everything is connect to Internet

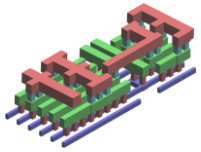
Design Flow

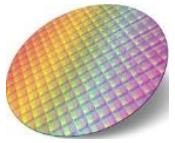


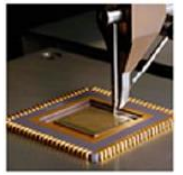

```
ENTITY IP
  port K1: in;
  port K2: in;
end ENTITY
```







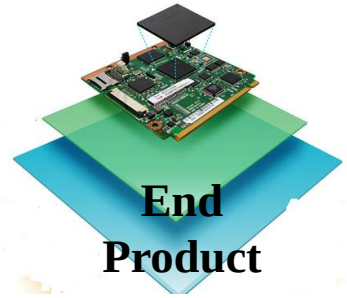




RTL IP 2 ... RTL IP n

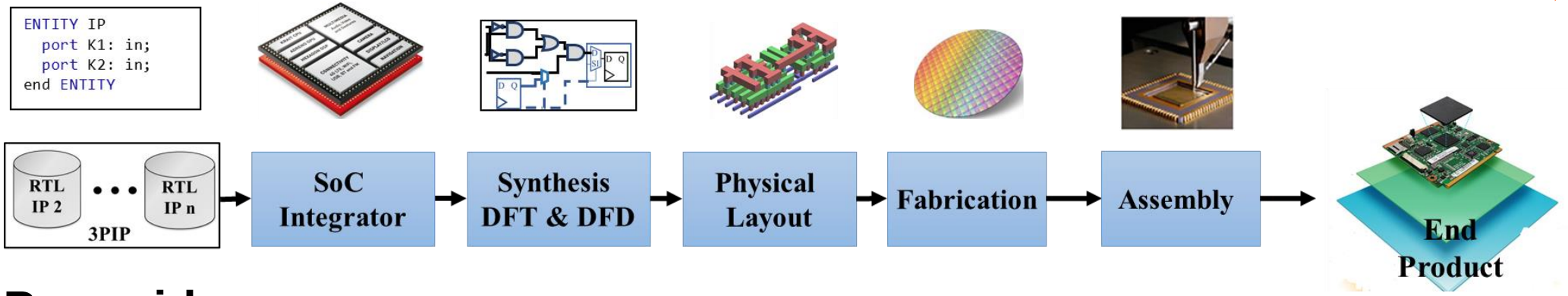
3PIP

SoC Integrator → Synthesis DFT & DFD → Physical Layout → Fabrication → Assembly →



End Product

Security & Trust Issues: Supply Chain



- **3PIP providers**
  - Working under aggressive schedules → **design mistakes, poor IP validation**
  - Can insert malicious implants (hardware **Trojans**)

- **CAD tools**
  - **Not equipped** with understanding security vulnerabilities
  - Vulnerabilities during **optimization, synthesis, DFT**, etc.

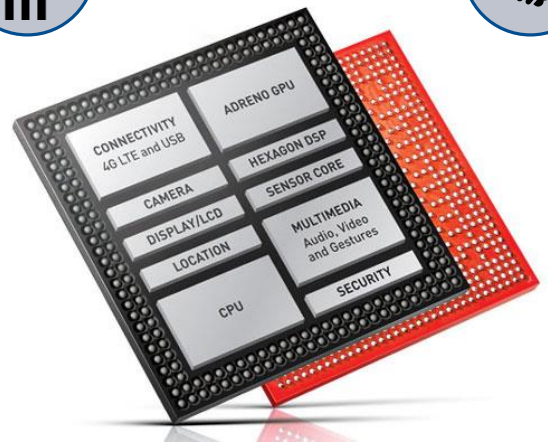
- **Foundry**
  - Access to the entire design → hardware Trojan, Counterfeit
  - **Counterfeits** → low-quality clones, overproduced chips in untrusted foundry

Challenges

Tens of billions transistors

Designed around the globe

Many custom/legacy functionality



Tens of IPs from 3P vendors

Many security critical assets

Aggressive time-to-market

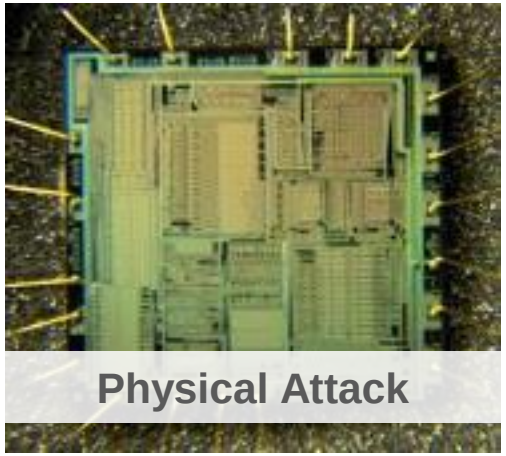
Ensuring security is a challenge

HW Attacks


Trojans

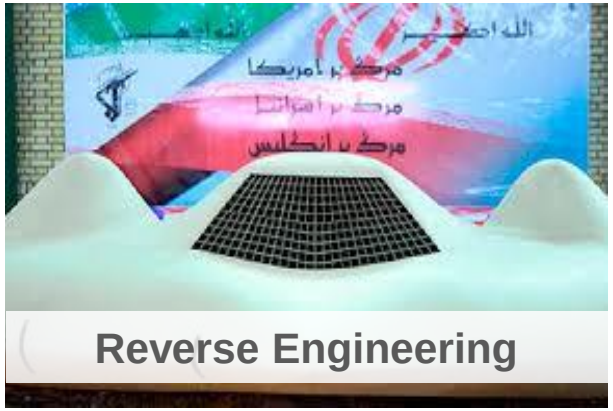

Untrusted Foundry

Counterfeit ICs


Physical Attack

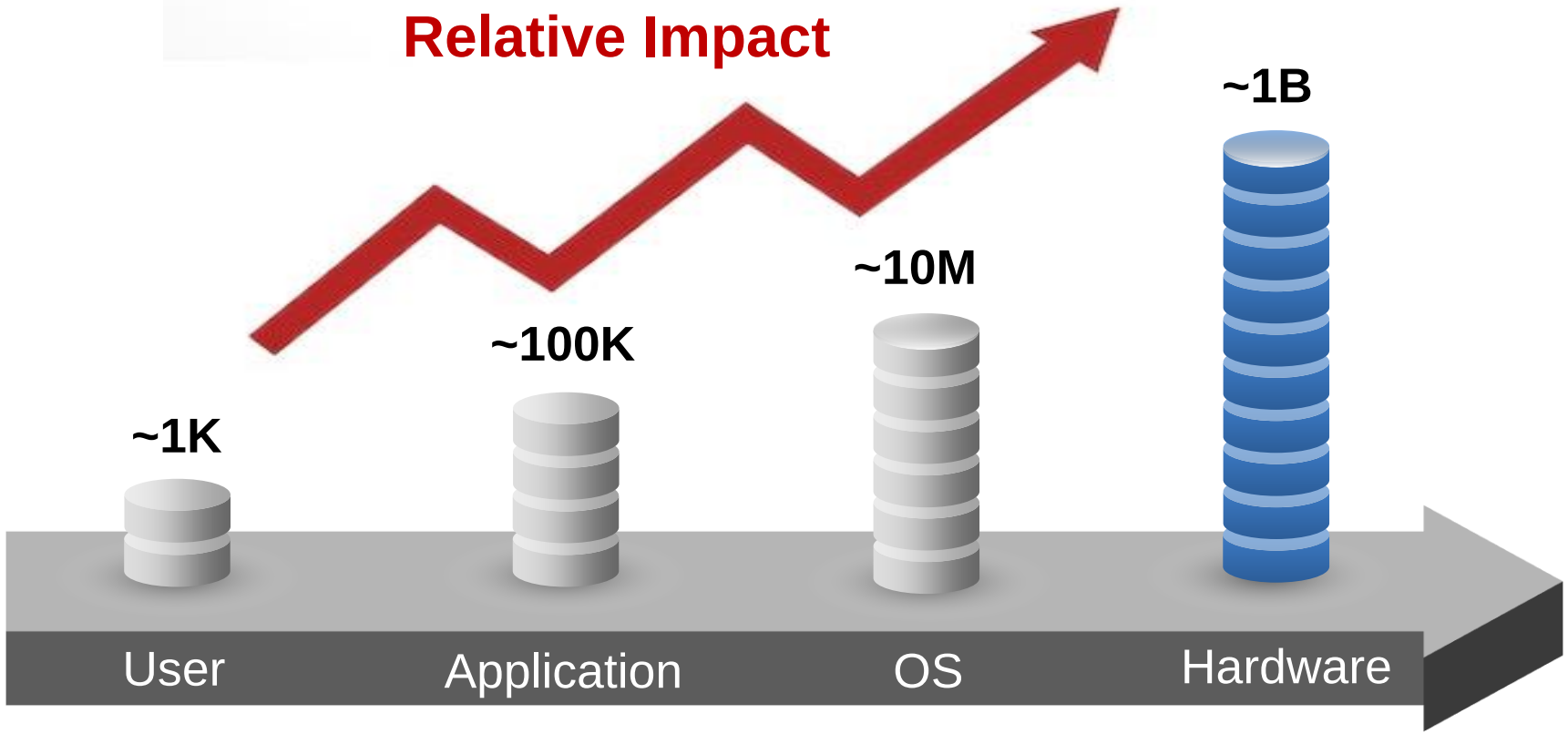

Side-channel Attacks

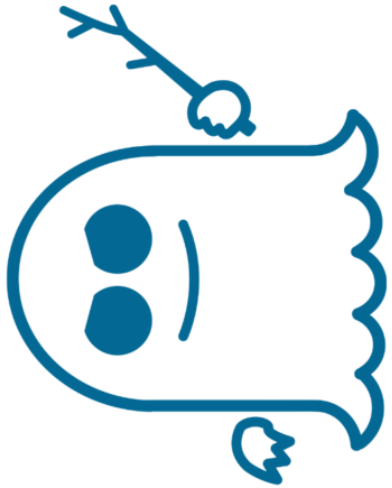
Fault Injection Attacks


Reverse Engineering

Counterfeit/Fake Parts

Impact: HW Security Compromise





Social engineering (phishing)

Malwares (information harvesting)

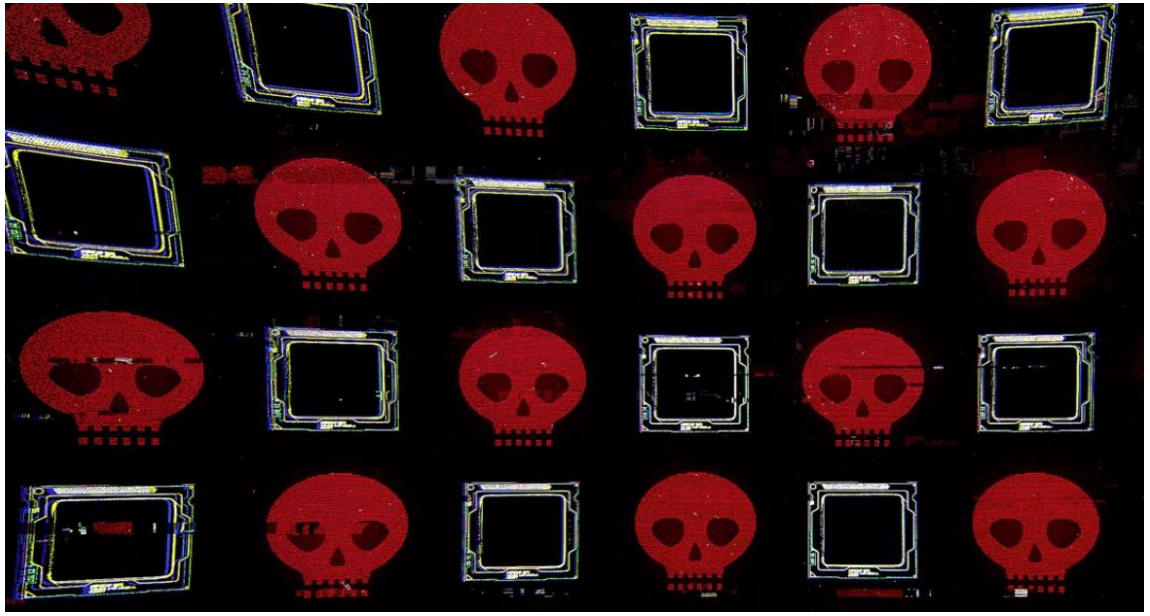
Virus/ Trojan (Hijacking/ DDoS)

Hardware compromise (low grade/ backdoor)

Impact of Hardware Compromise

THE VERGE

Intel Facing 32 Lawsuits Over Meltdown and Spectre CPU Security Flaws



Jan 4, 2018

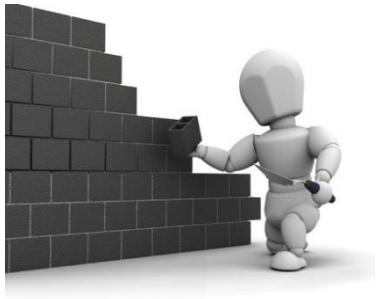
Intel sells off for a second day as massive security exploit shakes the stock

BUSINESS INSIDER

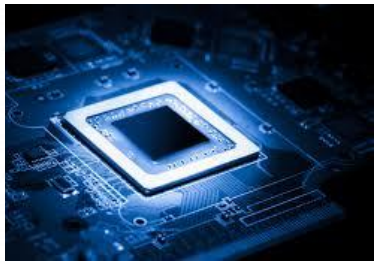
The company accused of selling Apple and Amazon data servers compromised by Chinese spies is getting crushed — it's lost half of its value today

Building a Secure Design

- Consider security from very **beginning**
- Identify what needs to be protected (**assets, IPs,**)
- Evaluate **right level** of security for each asset
  - A door may be sufficient to protect cloths, but a safe should be needed to protect jewelry
- Identify potential **vulnerabilities**
  - Need to develop a vulnerability database
- **Analyze if vulnerabilities exists**
  - **Need to develop CAD tools for security assessment**
- Develop proper **countermeasures**

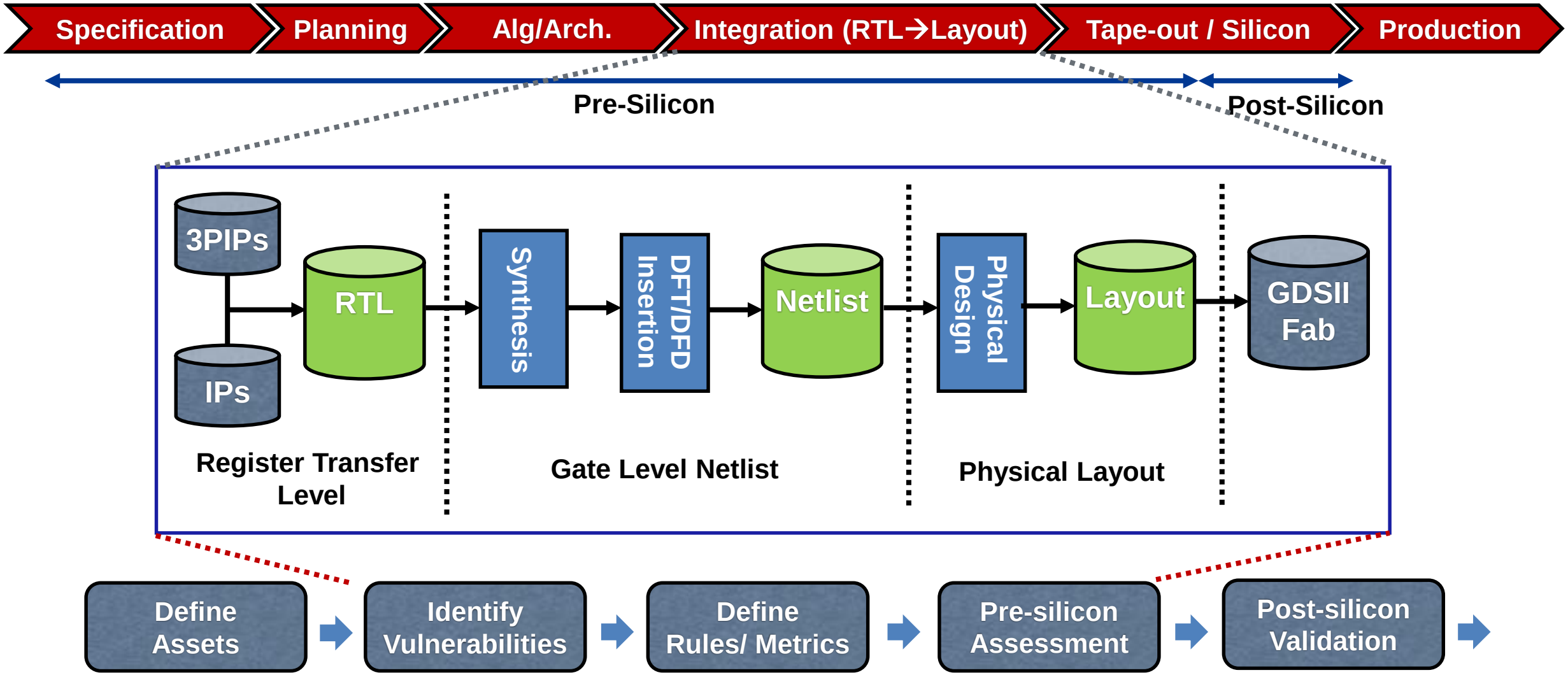


Security from the start



Security assessment

Security along Design Life-cycle

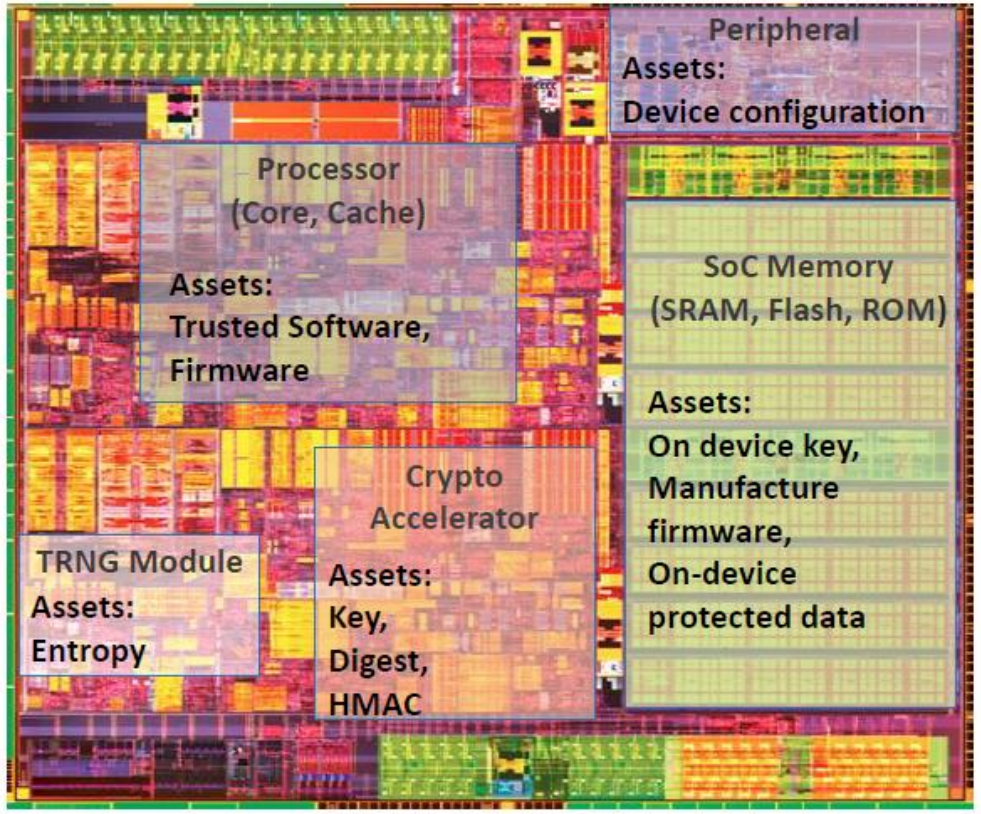


Security Assets

Asset: A resource of value worth protecting from an adversary

Security Assets in SoCs:

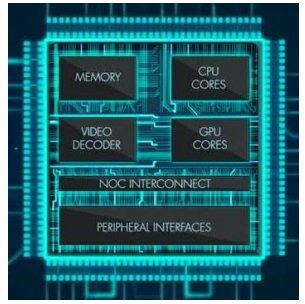
- On-device keys (developer/OEM)
- Device configuration
- Manufacturer Firmware
- Application software
- On-device sensitive data
- Communication credentials
- Random number or entropy
- E-fuse,
- PUF, and more…



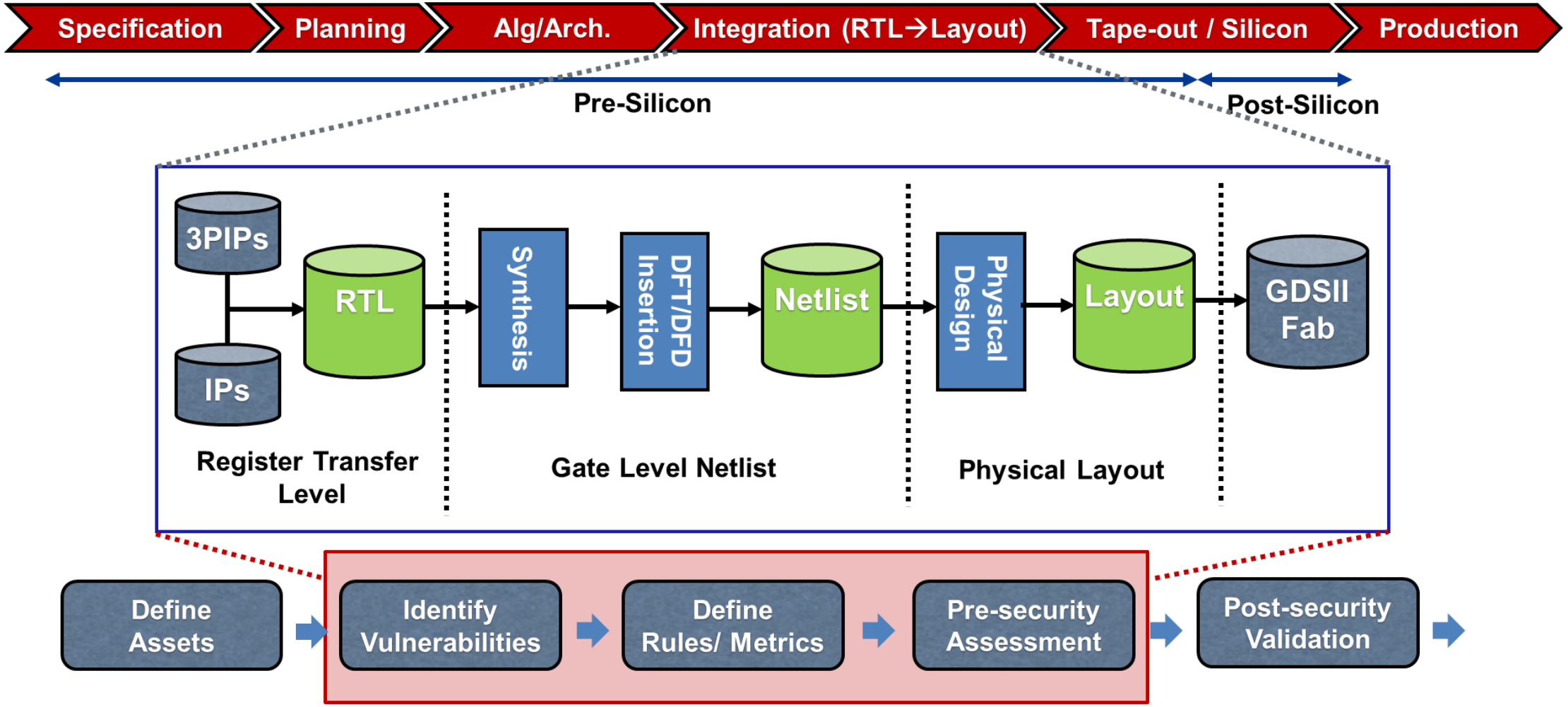
Source: Intel

Assets

- **On device key:** Secret encryption key material permanently embedded on the device
  - Confidentiality violated if compromised
- **Random Number/Entropy:** Cryptographic primitives rely on a good quality and unbiased random number generator
  - Weaken cryptographic algorithms if tampered
- **On-device sensitive data:** Information about the user credential, meter readings, counters
  - Privacy violated if compromised/tampered
- **Chip manufacturer's code:** Low level program instructions, proprietary firmware
  -



Security along SoC Design Life-cycle



CAD for Security

Current Practices

Manual Security Assessment

- **Certification Schemes**: Security verification by an independent official 3rd party
  - Example: payment Card Industry (PCI-DSS and PTS Finance industry)
- **Process overview:**

Security claims

3P Assessment

Final report

- **Suffer from various flaws**
  - Security review depends greatly on the experience
  - No proof that the design is completely secure against all possible attack scenarios

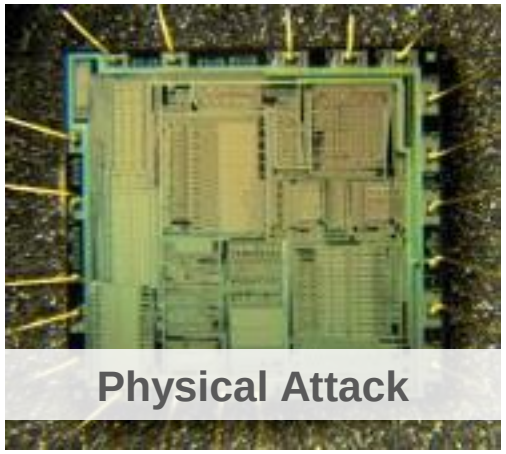
Automation

- Automation made design of modern ICs possible
- Tools made design of chips optimized for different applications possible, i.e., optimized for power, performance, and area
- Metrics played major role
  - Power
  - Performance
  - Area
  - Testability

Automation

- **Security is a generic term**
  - **Vulnerabilities are quite diverse**
  - **No silver bullet and no one size fits all**
  - **Relying on SMEs is no longer possible**
  - **There is a lack of understanding of security issues by designers**
  - **Emerging vulnerabilities**
    - **How quickly one can understand it? Mitigate it?**
    - **Best to be automated**
  - **Focus on the known vulnerabilities**


Fault Injection Attacks

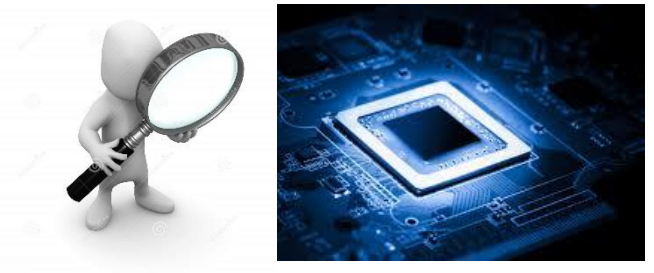

Physical Attack

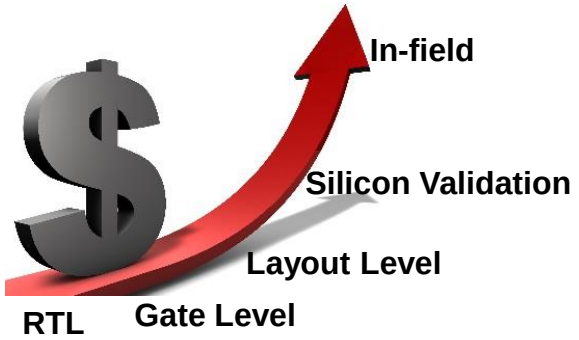

Side-channel Attacks


Untrusted Foundry

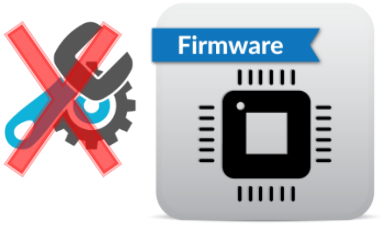
Automation

- ***No comprehensive solution to guide security check for SoCs***
- Cost of fixing vulnerabilities found at later stages is **significantly higher – Rule of 10**
  - Unlike software or firmware → **no flexibility** in changing or releasing post-shipment patches for hardware



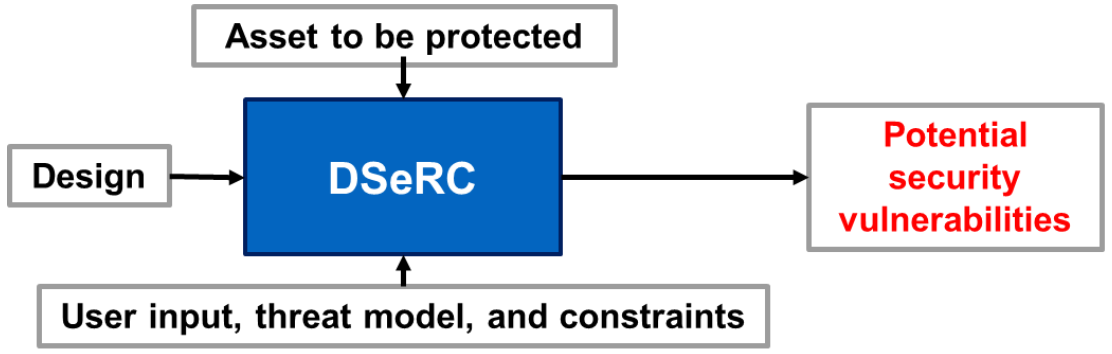


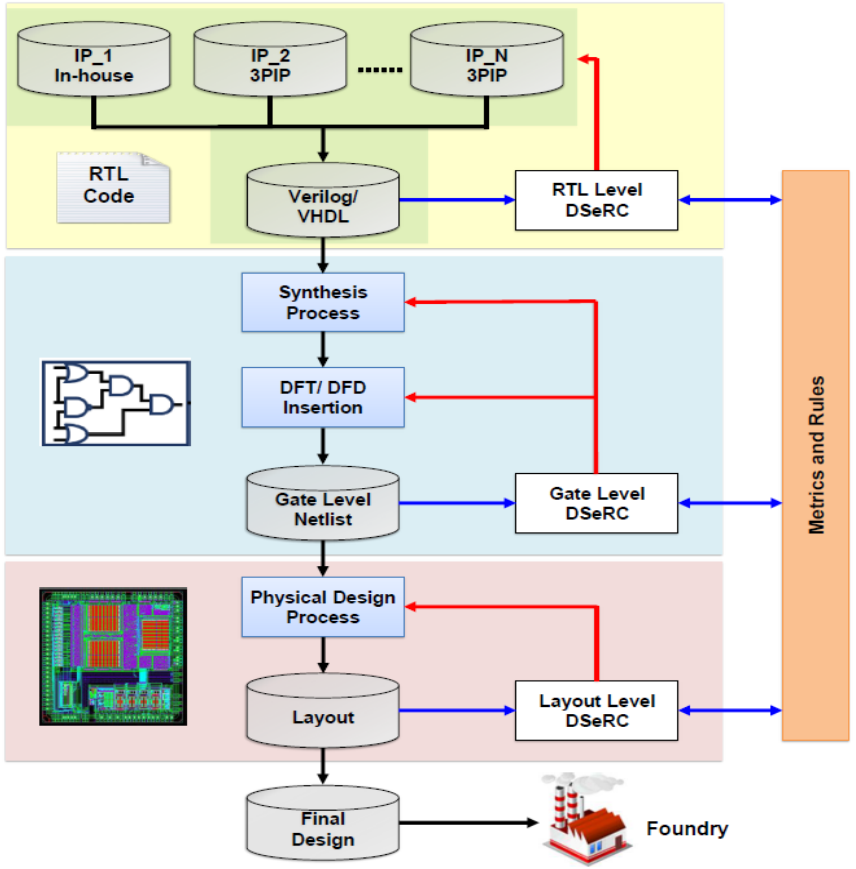
- Identify security issues during **design phase**
- Address them as early as possible in the design process



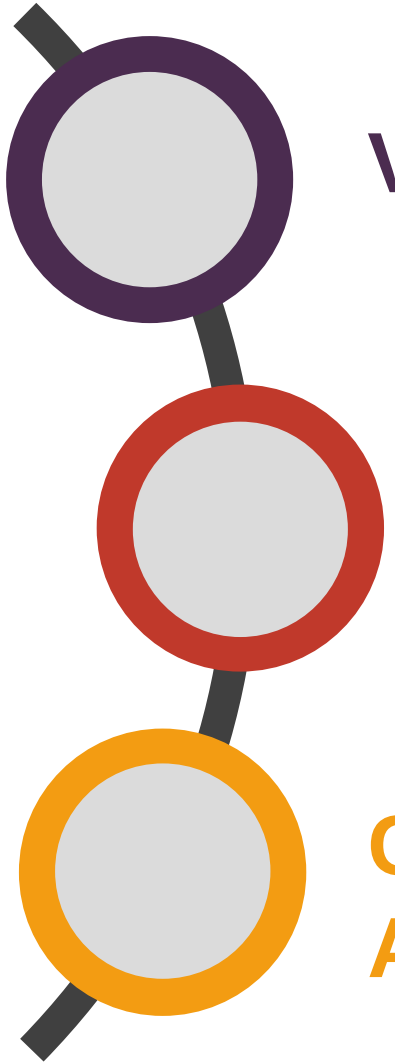
Security Assessment

- A **comprehensive framework** for analyzing **known** security issues in SoCs
- DSeRC framework:
  - **reads** the design files, constraints, threat model, and user input data
  - checks for vulnerabilities at all levels of **abstraction** (RTL, gate, layout, and architectural levels)
- Each vulnerability is tied with a set of **rules** and **metrics** → security can be quantitatively measured





Security Assessment

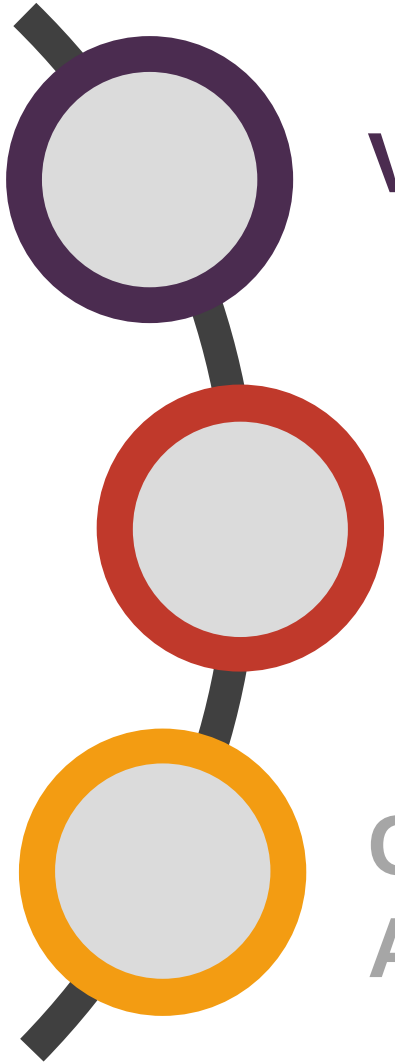


Vulnerabilities

Rules & Metrics

CAD for Security Assessment

Security Assessment

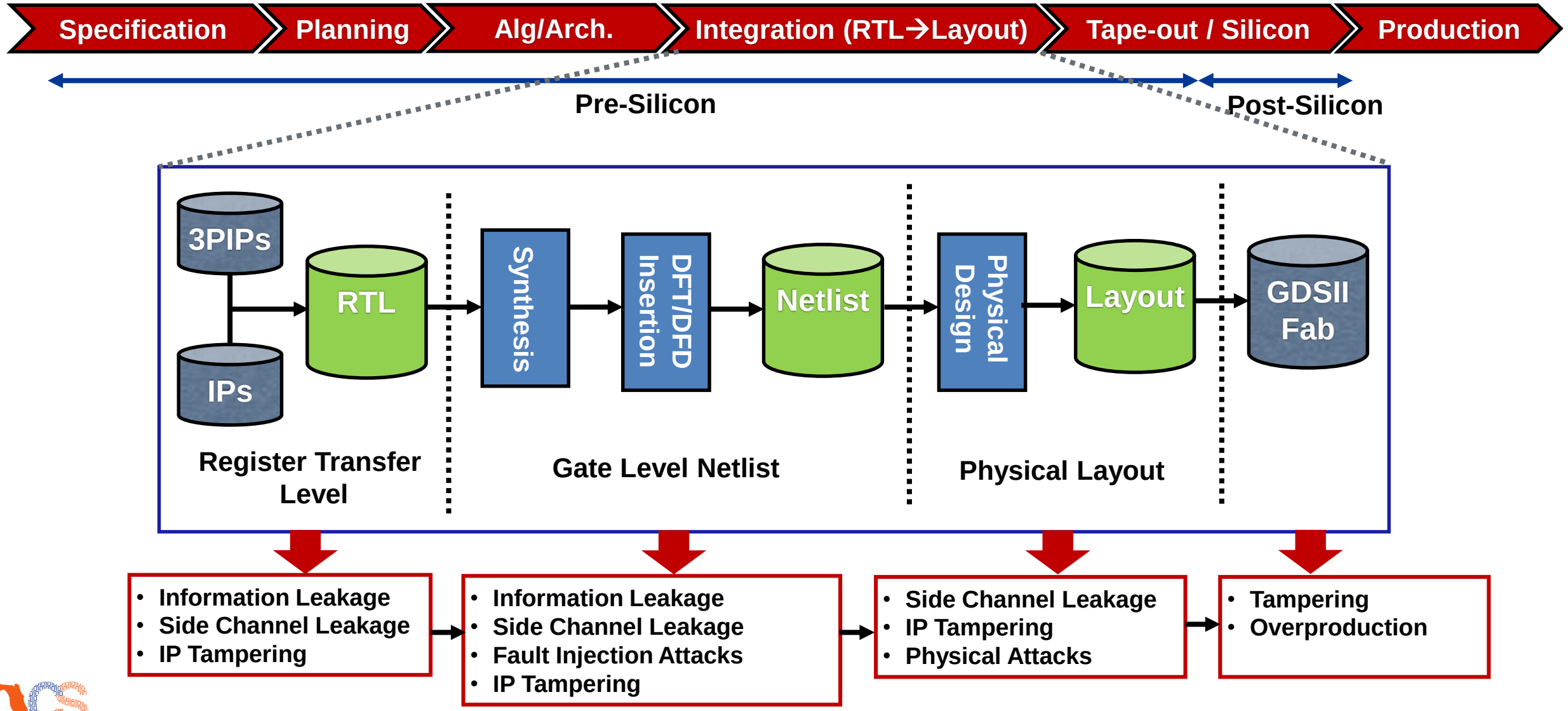


Vulnerabilities

Rules & Metrics

CAD for Security Assessment

Comprehensive Vulnerability Database



Sources of Vulnerabilities

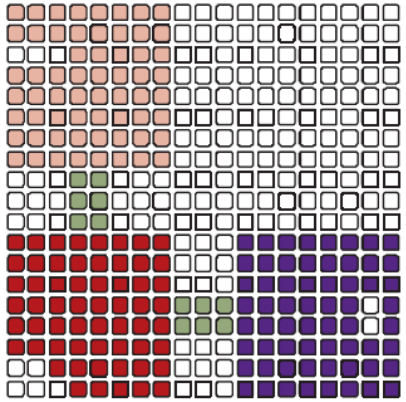
- **Design Issues**
  - Unintentionally created by (i) **designer's mistakes**, (ii) designer's **lack of understanding** of security problems and requirements in a complex SoC.

- **CAD Tools**
  - Tools are designed to focus on **power**, **performance**, and **area**
  - Can introduce vulnerabilities during **optimization/synthesis – leak information**

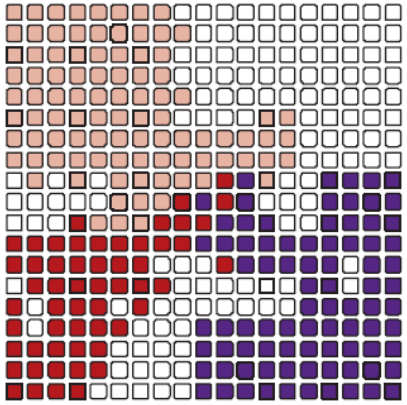
Synthesis tools "melt" the IP cores into one circuit – Circuit Flattening

T. Huffmire et al., Moats and Drawbridges: An Isolation Primitive for Reconfigurable Hardware Based Systems , ieee-sp'07.

RTL Design



Synthesized Design

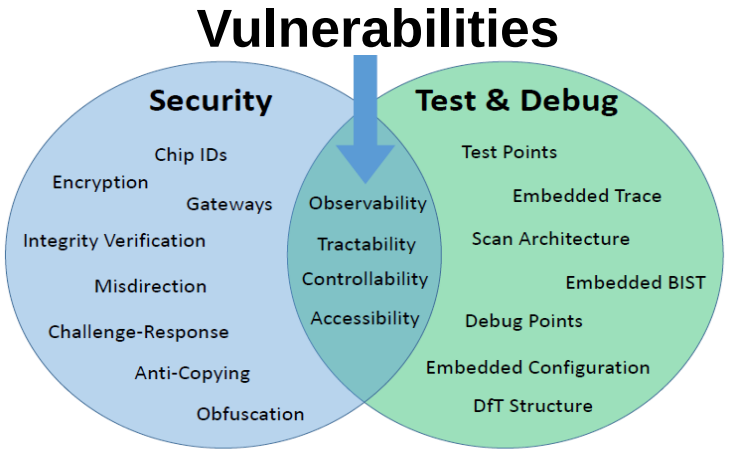


Confidential IP core

Untrusted IP cores

Sources of Vulnerabilities

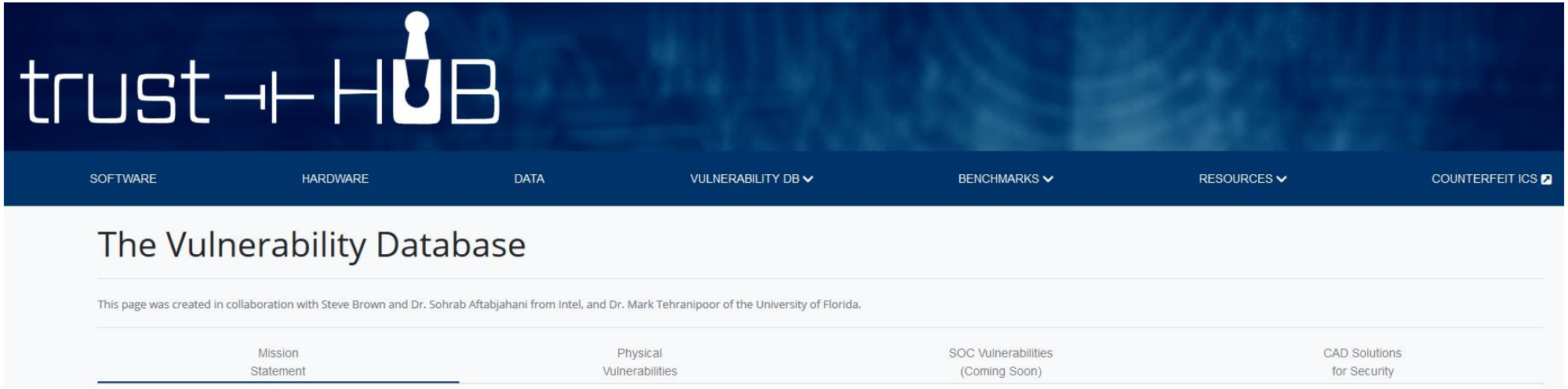
- **DFT and DFD Structures**
  - The increased **controllability** and **observability** added by DFT and DFD structures can create **additional** vulnerabilities



- **Black and White Hats**
  - Side channel attacks, fault injection attacks, information leakage, IP issues, and more

Trust-Hub / TAME Vulnerability Database

- An effort by industry and academic research leaders to provide awareness to researchers and practitioners of hardware security on SoC vulnerabilities
- **Goal**:
  - Develop the National Hardware Vulnerability Database (NHVD) to be shared with the potential of being used as a standard approach for enumerating and screening of various dimensions of security risks for SoCs



Trust-Hub Vulnerability Database

Timing
- Delay Analysis
- Clock Glitching Injection
- Overclocking
- Underclocking

Fault Injection
- Photon(Laser) Induced current
- Ambient / Ultra - violet
- Ionizing Radiation
- E and M Field
- Voltage Spike
- Temperature
- Over / Under Voltage

Side - Channel Observation Methods
- Acoustic
- Photoemission
- Voltage, Charge contrast
- SEM Inspection
- IREM Inspection
- Temperature Imaging
- E or M Fields
- Current & Power Measurement
- Voltage Measurement
- Indirect Voltage Measurement
- Data Remanence
- Black Box I / O

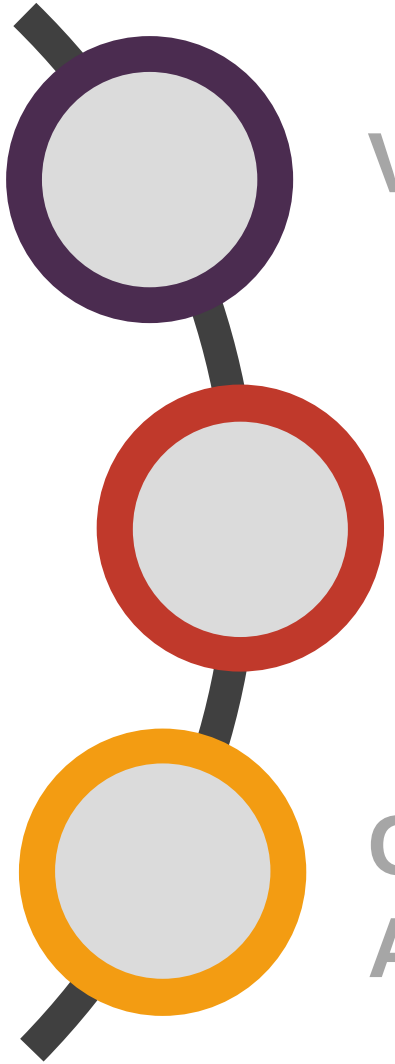
Logical Attacks
- Brute Force Algorithm
- Protocol Attacks

Die Analysis
- Delayering, Netlist Reconstruction
- Grind
- Section
- Dimple Down
- Photon(Laser) Induced Current
- Focused Ion Beam Deposition
- Focused Ion Beam Removal
- Ion Milling
- Direct Metal or Contact Probing
- Light Sensing
- Circuit Parameter Sensing

Board Analysis
- Delayering, Netlist Reconstruction

Design or FAB Injection
- HW Trojan

Security Assessment



Vulnerabilities

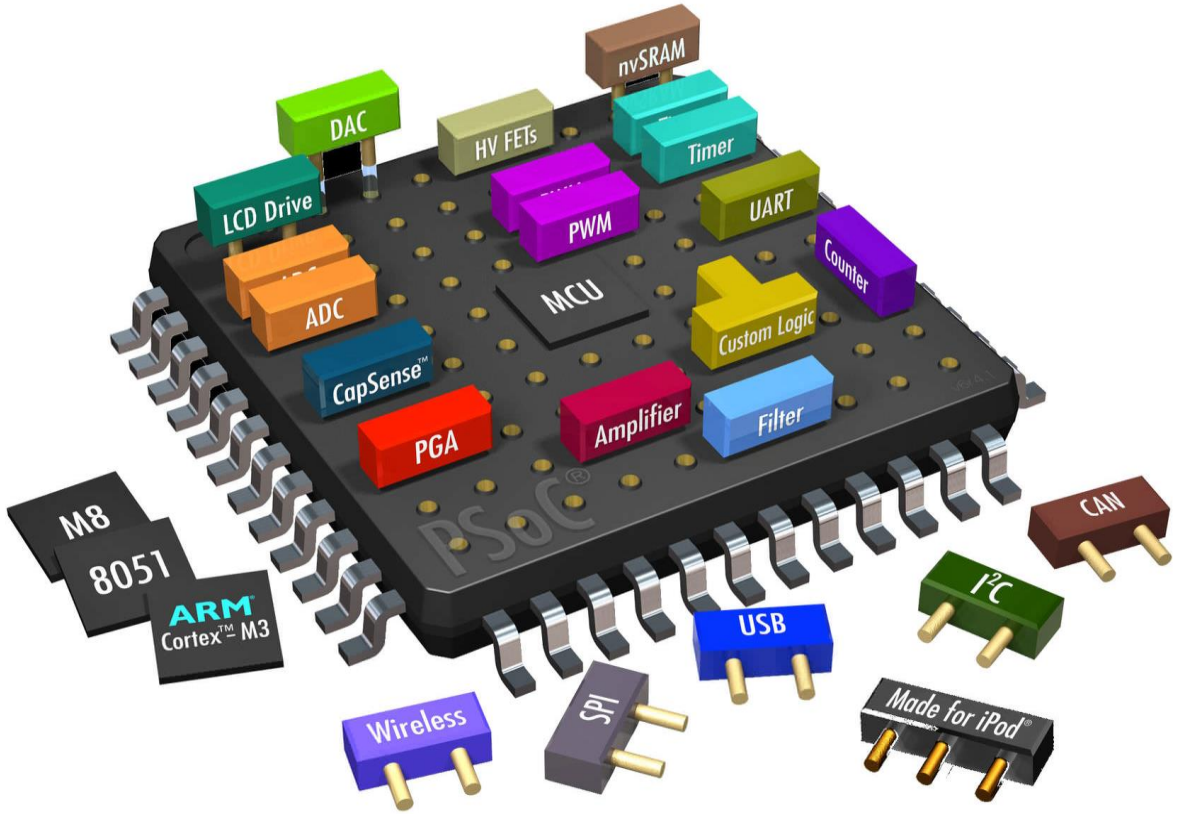
Rules & Metrics

CAD for Security Assessment

Abstraction Levels

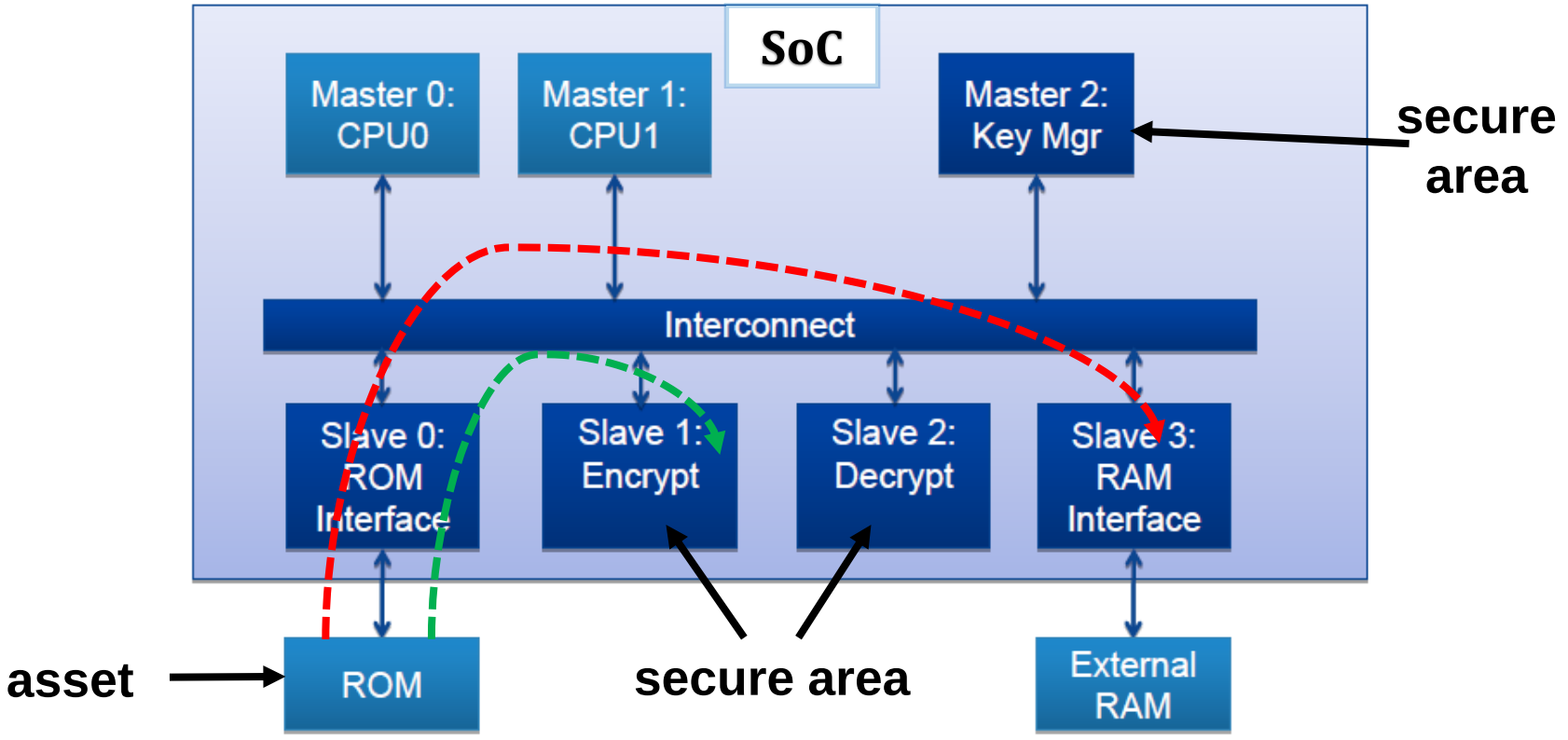
- **IP Level:** Vulnerabilities considered in modular basis at RTL, gate, and physical layout levels

- **SoC Level:** Vulnerabilities considered from system (e.g., SoC) level perspective – interaction between different cores



Vulnerabilities and Rules

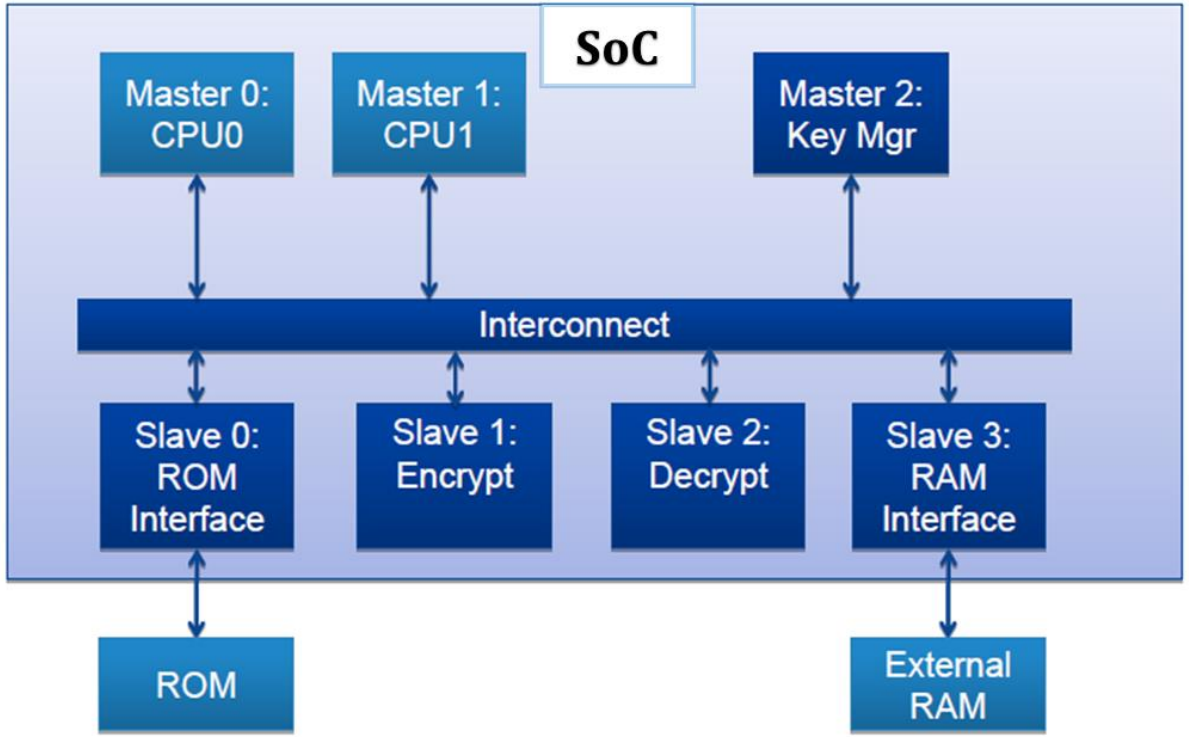
- **Vulnerability:** Asset leakage
- **Rule:** An asset should never propagate to any location where an attacker can observe it



Source: Jasper

More Examples of Rules

- uP in user mode should **never access** OS kernel memory
- During crypto operation reset, reading intermediate results, changing keys, and data operations **are prohibited**
- During cryptographic asset (e.g. key) transfer from the system memory to the crypto-core registers, all other IP accesses to the bus **are disabled**



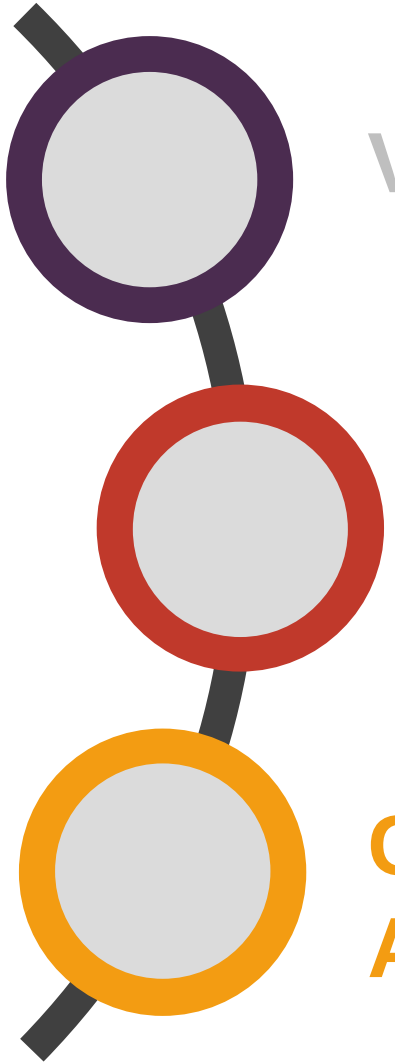
- The power management module **can enable** a modification in the clock frequencies only when the core is not in active mode
- During debug, **no accesses** are allowed to the security critical part of memory

Source: Jasper

Vulnerabilities, Metrics and Rules

	Vulnerability	Metric	Rule	Attack (Attacker)
RTL Level	Dangerous Don't Cares	Identify all 'X' assignments and check if 'X' can propagate to observable outputs	'X' assignments should not be propagated to observable output	Hardware Trojan (Insider)
	Hard-to-control & hard-to-observe Signals	Statement hardness and signal observability	Statement hardness (signal observbility) should be lower (higher) than a predefined threshold	Hardware Trojan (Insider)
	Asset leakage	Structure checking and IFT	Security sensitive assets should not be exposed to observable points	Asset hacking (End user)
	….			
Gate Level	Hard-to-Control & hard-to-observe Nets	Net controllability and observability	Controllability and observability should be higher than a threshold value	Hardware Trojan (Insider)
	Vulnerable FSM	Vulnerability factor of fault injection (VF_{FI}) and Trojan insertion (VF_{Tro})	VF_{FI} and VF_{Tro} should be zero	Fault injection, Hardware Trojan (Insider, end user)
	Asset Leakage	Confidentiality and integrity assessment	Assets should not be leaked through observable points	Asset hacking (End user)
	Design-for-Test (DFT), JTAG/IJTAG Vulnerabilities	Confidentiality and integrity assessment	Assets should not be leaked or accessed through DFT structure	Asset hacking (End user)
	Design-for-Debug structure Vulnerabilities	Confidentiality and integrity assessment	Assets should not be leaked or accessed through DFD structure	Asset hacking (End user)
	…..			
Layout Level	Side-Channel Leakage	Side-channel vulnerability (SCV)	SVF should be lower than a threshold value	Side-channel attack (End user)
	Microprobing Vulnerability	Exposed area of the security-critical nets which are vulnerable to microprobing attack	The exposed area should be lower than a threshold value	Micro-probing attack (Professional attacker)
	Trojan Insertion – unused space	Unused space analysis	Unused space should be lower than a threshold value	Untrusted foundry
	…..			

Security Assessment



Vulnerabilities

Rules & Metrics

CAD for Security Assessment

Trust-Hub CAD for Security

Information Leakage | Hardware Trojan | Probing | Fault Injection | Logic Locking | Side Channel Analysis

Violation of information flow security policies due to design mistakes and/or CAD tools

Academic License
Commercial or academic tool

Plugin Solution

Incorporate to conventional ASIC design flow to asses vulnerabilities due to violation of IFS policies at design stage

Security Metric

Confidentiality Verification (asset leakage) and Integrity Verification (asset tampering)

Description

The tool models an asset (e.g., a net carrying a secret key) as a stuck-at-0 and stuck-at-1 fault and utilizes the automatic test pattern generation (ATPG) algorithm to detect that faults. A successful detection of faults means that the logical value of the asset carrying net can be observed through the observe points or logical value of the asset can be controlled by the control points. The tool works at a gate level netlist.

Contacts

Dr. Mark Tehranipoor, tehranipoor@ece.ufl.edu
Dr. Domenic Forte, dforte@ece.ufl.edu

More Information

https://fics.institute.ufl.edu/

CAD for Security

RTL

- **Susceptibility to Trojan Insertion**
- Power Side-channel Leakage Assessment (RTL-PSC)
- EM Leakage Assessment
- Information Leakage (Jasper)
- Formal Verification of Security Properties

Gate

- Information Flow Security (IFS) Verification
- Power Side-channel Leakage Assessment (SCRIPT)
- Trojan Detection and Localization
- Susceptibility to Fault Injection (AVFSM)

Layout

- Susceptibility of Probing Attacks
- Power Side-channel Leakage Assessment (TVLA)

Susceptibility to Trojan Insertion

- Sections in a circuit with **low controllability and observability** are considered potential areas for implementing **Trojans**
- **Metrics:**
  - **Statement hardness**: Difficulty of executing a statement
  - **Observability**: Difficulty of observing a signal

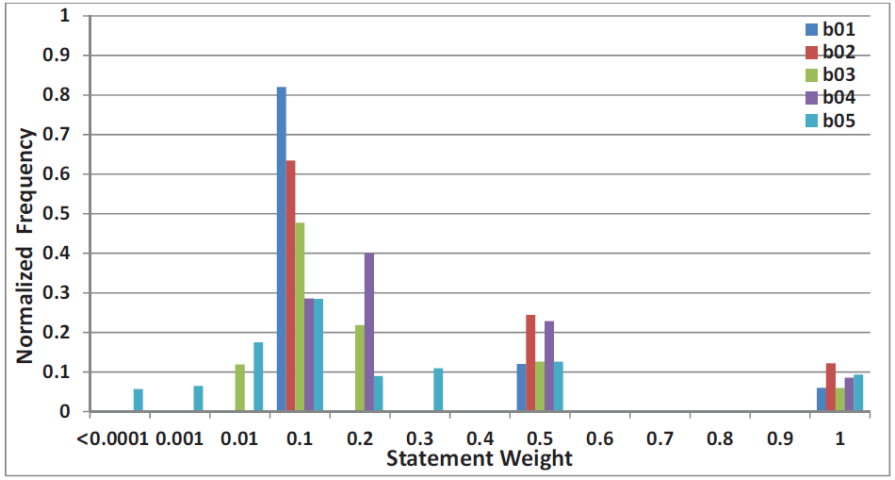
```
13. BEGIN
14.         FOR X IN 0 TO 9 LOOP
15.                 IF ( X < 2 ) THEN
16.                         P := 1 - X;
17.                 ELSIF ( X > 5 ) THEN
18.                         IF ( K = 7) THEN
19.                                 P := 2 * X;
20.                         ELSE
21.                                 P := 2 + K;
22.                         END IF;
23.                 END IF;
24.         END LOOP;
25.         IF ( P < 7 ) THEN
26.                 IF ( X < 7 ) THEN
27.                         IF ( P > 2 ) THEN
28.                                 Z <= P;
29.                         END IF;
30.                 END IF;
31.         END IF;
32. END PROCESS PROC1;
```

High Statement Hardness

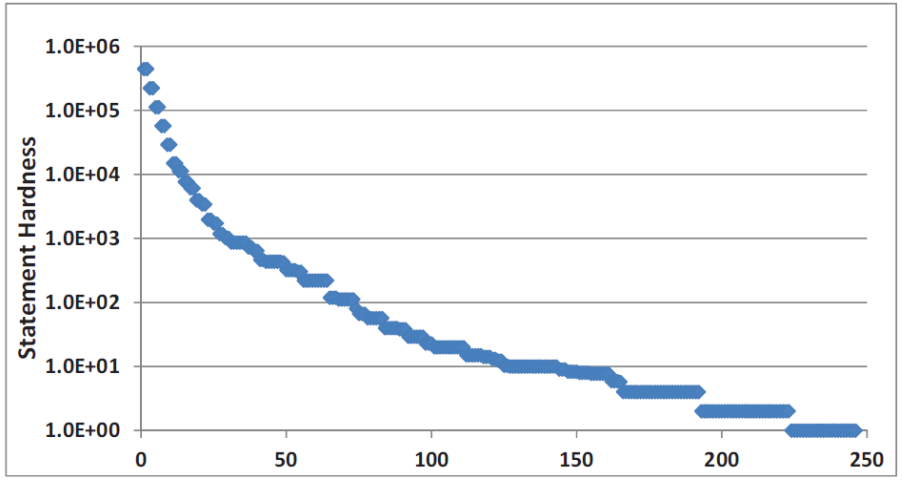
Low Observable Point

- **Rule 1:** Statement hardness of each statement should be lower than a predefined threshold
- **Rule 2:** Observability of each observable signal should be higher than a predefined threshold

Susceptibility to Trojan Insertion



Statement weight analysis.



Statement hardness for b05.

- Application of the Tool:
  - Can be used to determine which parts of a circuit are more susceptible to Trojan insertion
  - Can be used to track and identify malicious part included in the code by a rogue employee (insider threat)

CAD for Security

RTL

- Susceptibility to Trojan Insertion
- **Power Side-channel Leakage Assessment (RTL-PSC)**
- EM Leakage Assessment
- Information Leakage (Jasper)
- Formal Verification of Security Properties

Gate

- Information Flow Security (IFS) Verification
- Power Side-channel Leakage Assessment (SCRIPT)
- Trojan Detection and Localization
- Susceptibility to Fault Injection (AVFSM)

Layout

- Susceptibility of Probing Attacks
- Power Side-channel Leakage Assessment (TVLA)

SCA

- Side-channel attacks have been a major concern to security community.
- Side-channel countermeasures (e.g. masking and hiding) and leakage assessment (e.g. TVLA) have been studied in academia and industry.
- However, they mostly focus on post-silicon side-channel assessment.
  - Difficult to find the leakage sources or modules
  - Too expensive in modifying leakage issues
- Contribution: A frame work to automatically assess PSC vulnerability at the earliest pre-silicon design state, i.e. RTL
  - Technology independent
  - Fine granularity evaluation: Which modules?
  - Fast power estimation
  - Generic framework

RTL-PSC Framework

- **Goal : Identifying vulnerable blocks (modules)**
  - A group of simulation keys are specified.
  - Synopsys VCS simulation
  - Generate SAIF files
  - Localization for each module
  - Estimate power leakage distribution
  - Calculate evaluation metrics; KL div., SR
  - Identify vulnerable modules

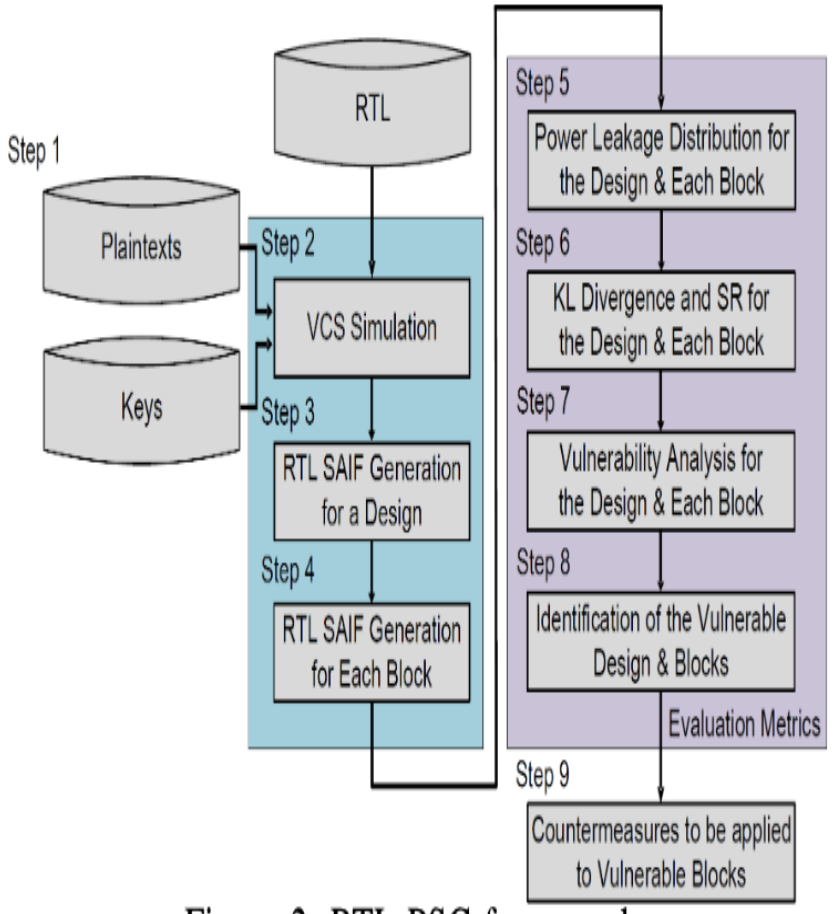
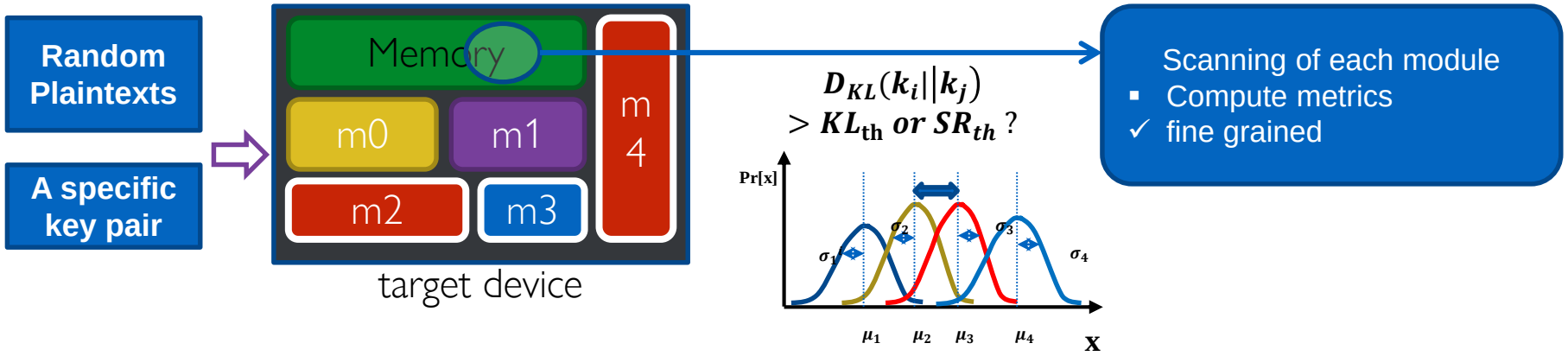


Figure 2: RTL-PSC framework.

Evaluation Metrics



- Kullback-Leibler (KL) divergence
- Success Rate based on the maximum likelihood estimation
- A key pair:
  - Each key consists of the same subkey
  - HD between two subkeys is maximum
  - D_KL increases asymptotically as HD increases

Table I: Keys used in RTL-PSC framework.

Key_0	0x0000_0000_0000_0000_0000_0000_0000_0000
Key_1	0x0000_0000_0000_0000_0000_0000_0000_00FF
	.....
Key_{15}	0x00FF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF
Key_{16}	0xFFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF

AES RTL Implementation

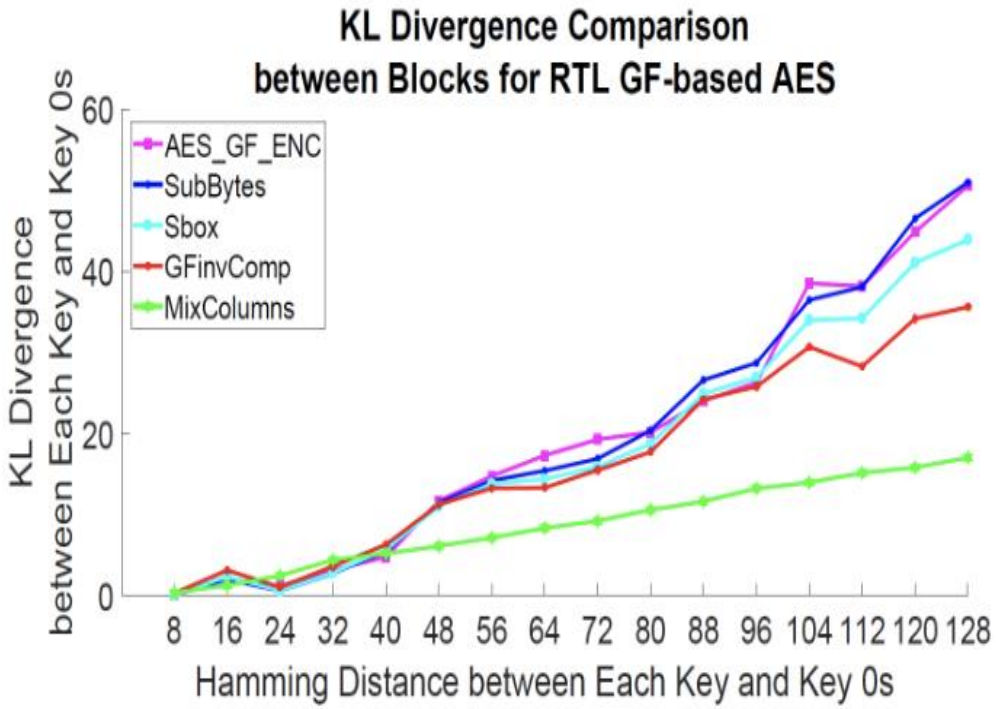
	AES-GF	AES-LUT
SBOX Implementation	Galois-field Arithmetic	Lookup Table
Key expansion and Round operation	Parallel	Serial
# Clocks / encryption	10 clocks (10 rounds)	11 clocks (an Addround + 10 rounds) after key expansion
Blocks	5 SubByte 4 Sbox GFinvComp 4 MixColumn	1 SubWord 1 SubByte 4 MixColumn

VCS Simulation

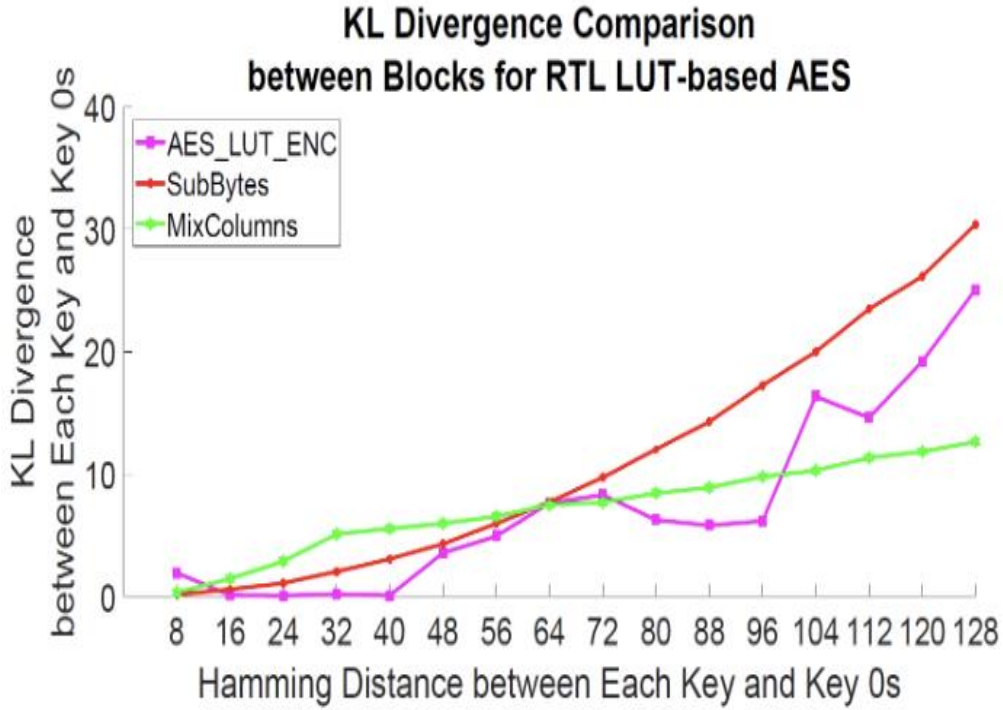
- Input : 17 keys, 1000 random plaintexts per each key
- Output : Switching Activity Interchange Format (SAIF) files
- Calculate # of transitions per block and per clock
- Calculate KL divergence between two different keys
- VCS simulation time : 42X than gate-level simulation
  - AES-GF : 46.3 min
  - AES-LUT : 24.03 min

Key_0	0x0000_0000_0000_0000_0000_0000_0000_0000
Key_0	0x0000_0000_0000_0000_0000_0000_0000_00FF
Key_1	0x0000_0000_0000_0000_0000_0000_0000_FFFF
…	…
Key_{15}	0x00FF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF
Key_{16}	0xFFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF

KL Divergence Comparison between Blocks



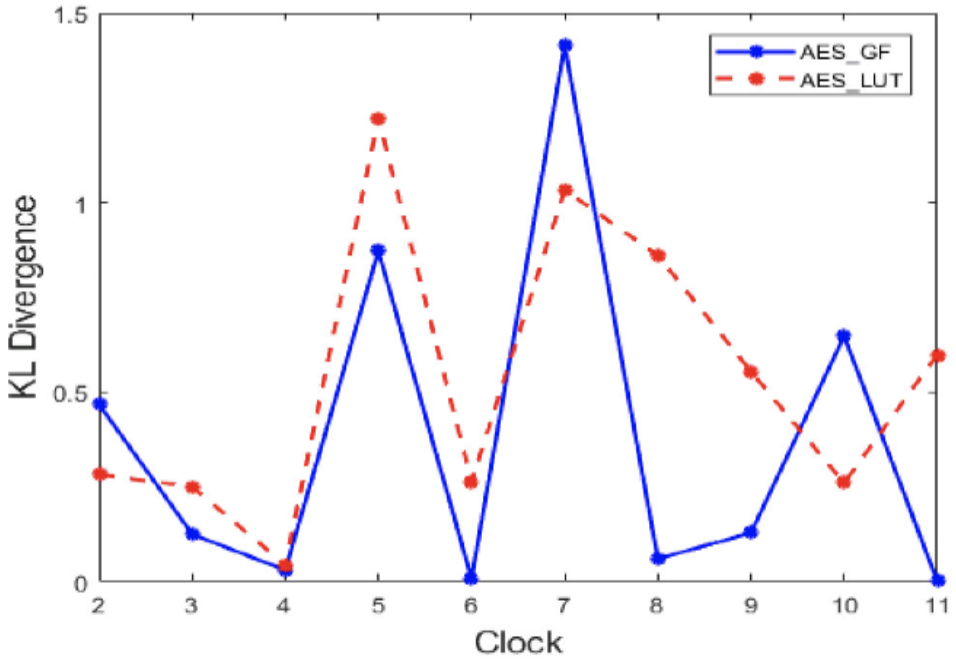
(a) KL divergence comparison between blocks for RTL AES-GF implementation



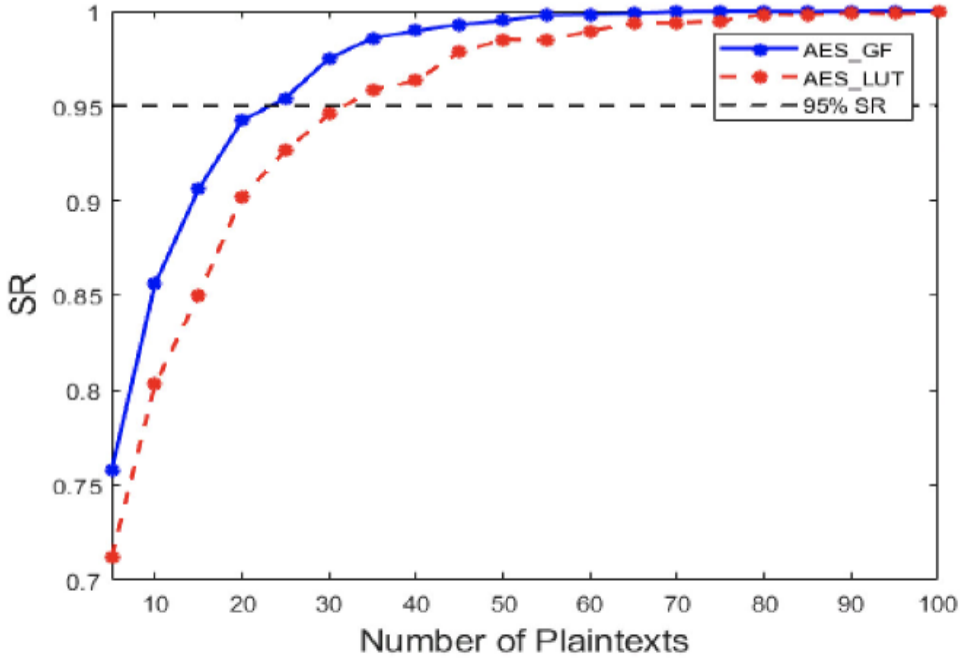
(b) KL divergence comparison between blocks for RTL AES-LUT implementation

Figure 3: KL divergence comparison between blocks for RTL AES-GF and AES-LUT implementations.

KL Divergence and SR



(a) KL divergence per clock cycle for AES-GF and AES-LUT implementations

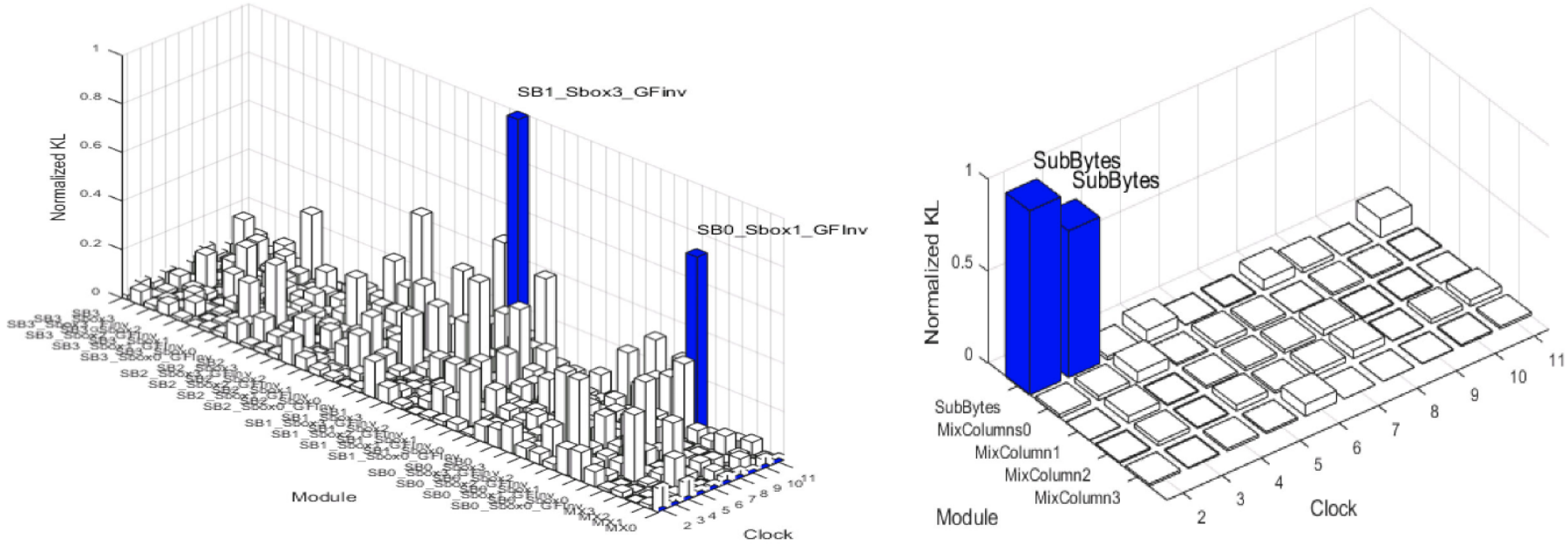


(b) SR_{em} corresponding to KL divergence (0.47 and 0.28) for AES-GF and AES-LUT implementations

Figure 4: KL divergence and SRs for AES-GF and AES-LUT implementations.

Vulnerable Block Identification

- Normalized KL divergence: $KL_{norm} = KL_i / \max(KL_i)$
- KL threshold : $KL_{norm.th} = 0.5$
- The threshold values can be adjusted by the SR vulnerability level.



(a) Normalized KL divergence for AES-GF implementation in both time and spatial/modular domains

(b) Normalized KL divergence for AES-LUT implementation in both time and spatial/modular domains

Figure 5: Normalized KL divergence for vulnerable blocks within AES-GF and AES-LUT implementations ($KL_{norm.th} = 0.5$).

Gate-level Validation

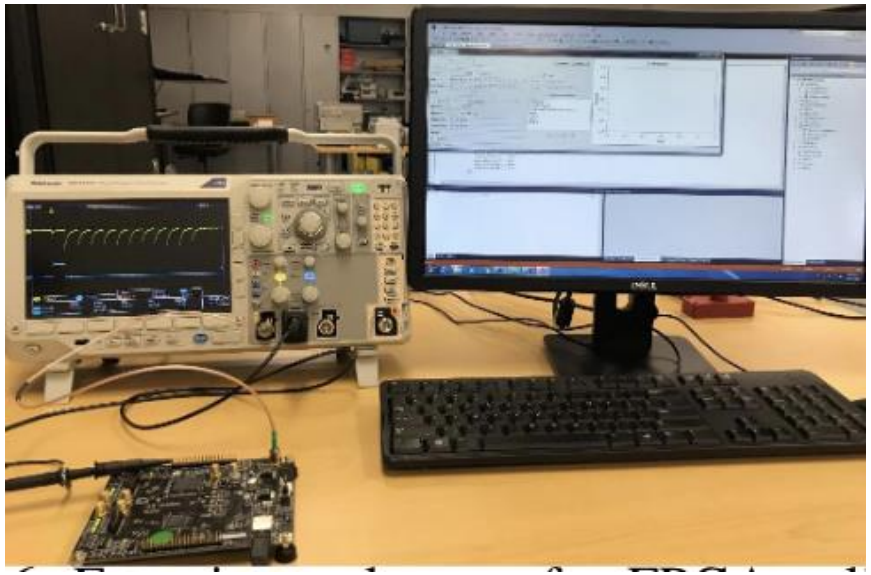
- Logic synthesis using Synopsys Design Compiler with Synopsys standard cell library.
- Power estimation for the entire design and each block using Synopsys PrimeTime.
- Calculation of KL divergence at gate level
- Calculation of Pearson correlation coefficient between the KL divergence between RTL and GTL

AES-GF Blocks, RTL vs GTL				AES-LUT Blocks, RTL vs GTL	
SubByte	Sbox	GFinvComp	MixColumn	SubByte	MixColumn
99.11%	99.55%	99.64%	94.73%	99.71%	96.80%

Gate-level Validation

- For FPGA silicon validation, a SAKURA-G board is used for AES implementations with a 24MHz-clock frequency
- Power measurement setup
  - Tektronic MDO3102 oscilloscope (Sampling rate 500 MS/s, Bandwidth : 250 MHz)
  - Passive probe

Benchmark	RTL vs FPGA
AES-GF	98.83 %
AES-LUT	80.80 %



CAD for Security

RTL

- Susceptibility to Trojan Insertion
- Power Side-channel Leakage Assessment (RTL-PSC)
- **EM Leakage Assessment**
- Information Leakage (Jasper)
- Formal Verification of Security Properties

Gate

- Information Flow Security (IFS) Verification
- Power Side-channel Leakage Assessment (SCRIPT)
- Susceptibility to Fault Injection (AVFSM)
- Susceptibility to Trojan Insertion

Layout

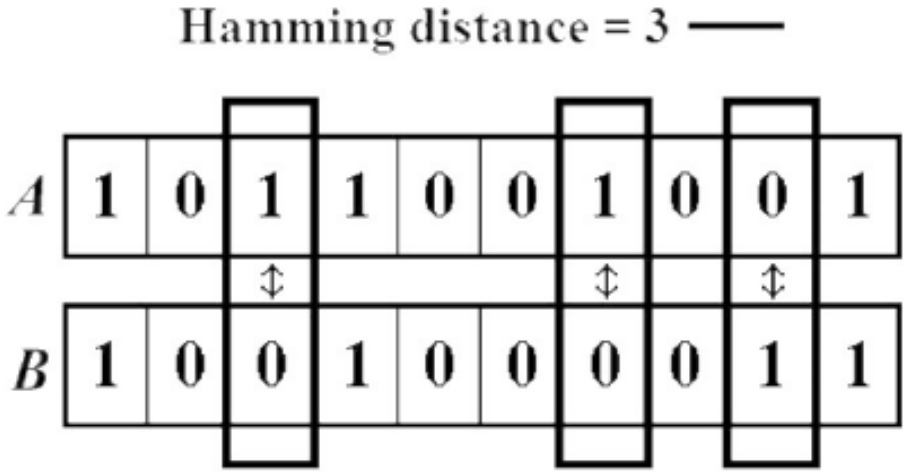
- Susceptibility of Probing Attacks
- Power Side-channel Leakage Assessment (TVLA)

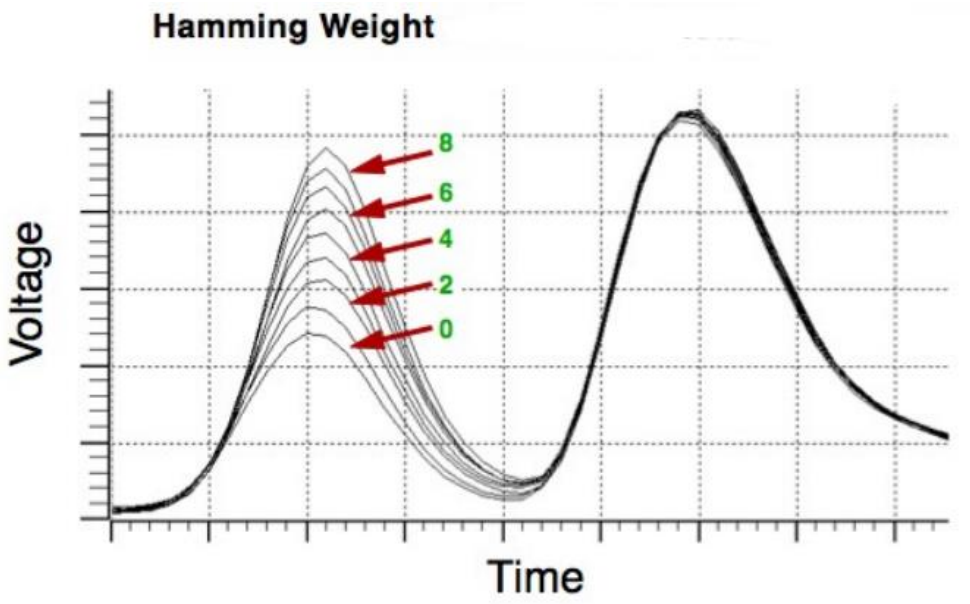
Motivations

- Side-channel threats for the modern cryptographic integrated circuits (IC)
  - Recover the secret information from circuits' manifestations
  - Post-silicon stage security evaluation
  - High cost for removing/evaluating the side-channel vulnerability
- Proposed solutions
  - Design for side-channel security (DFSCS) framework
  - Register Transfer Level (RTL) hardware implementations
  - EM Simulation Model: combine the two models
    - Hamming Distance (HD) model
    - Hamming Weight (HW) model

Information Leakage Model

- Hamming distance model
- Hamming weight model
- Improved Hamming distance/weight model





EM Simulation Model: Hamming Distance

- Hamming Distance (HD) model
  - Def. The number of positions at which the corresponding symbols are different.
  - The minimum number of substitutions required to change one string into the other string
  - Metric to monitor the changes within every time point along the time line

$$D(t) = \sum_{i=1}^{n} F_i \times (A_i \oplus B_i)$$

  - Performs a good match with the EM radiation from FPGA measurements
  - Challenge: Too many mismatches in high frequency range

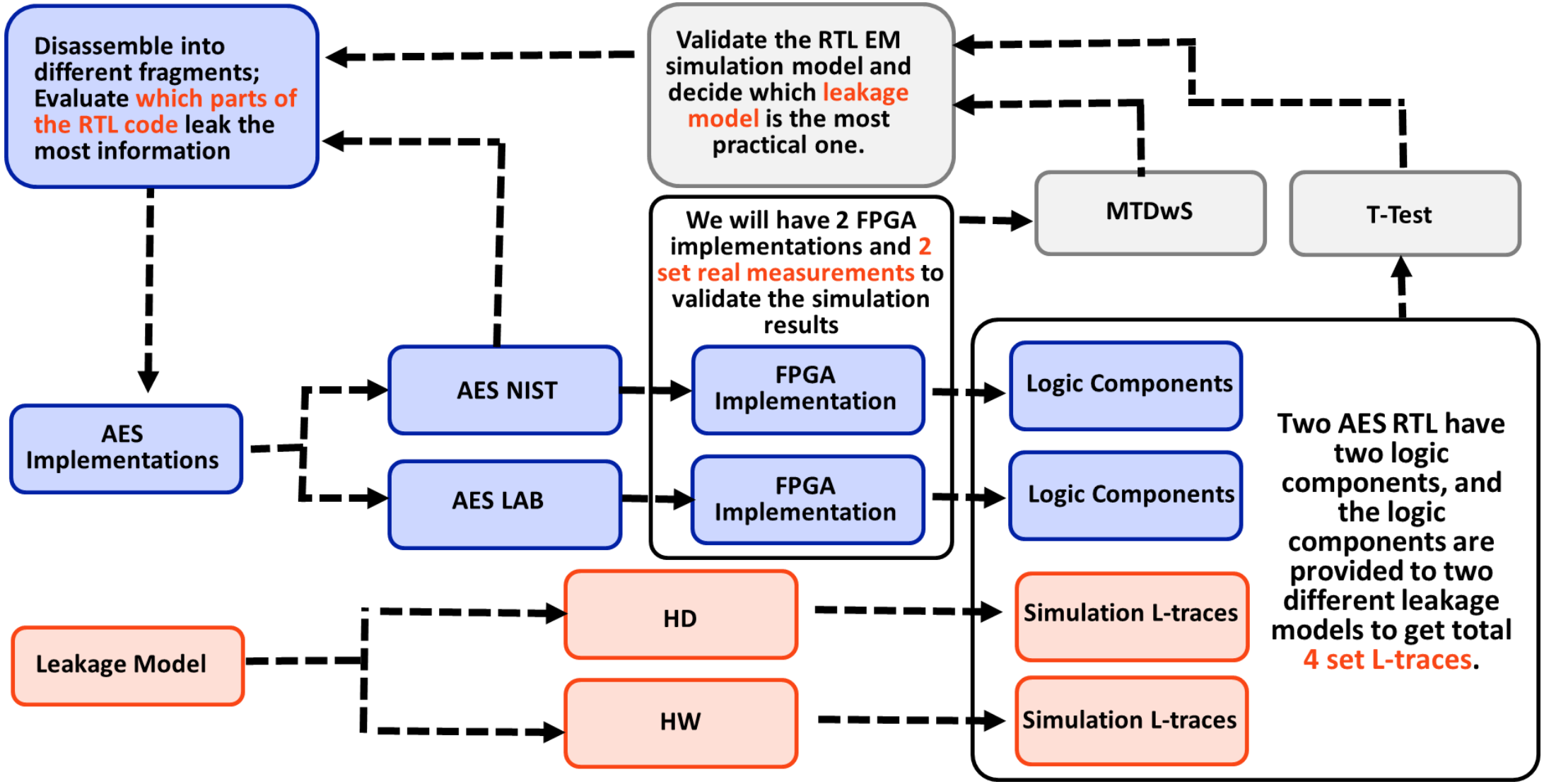
EM Simulation Model: Hamming Weight

- Hamming Weight (HW) model
  - Def. The number of symbols that are different from the zero-symbol
  - The number of 1's in a binary string
  - Metric to calculate the changes along the time line

$$R(iv,t) = \sum_{i=1}^{n} H(A_i)|_{iv}$$

  - Can be leveraged for side channel attack
  - Challenge: Evaluation results are not validated using FPGA measurements
- Proposed EM simulation model:
  - Try both HD and HW models
  - Make an option on the model with better performance

Overall DFSCS Framework



CAD for Security

RTL

- Susceptibility to Trojan Insertion
- Power Side-channel Leakage Assessment (RTL-PSC)
- EM Leakage Assessment
- **Information Leakage (Jasper)**
- Formal Verification of Security Properties

Gate

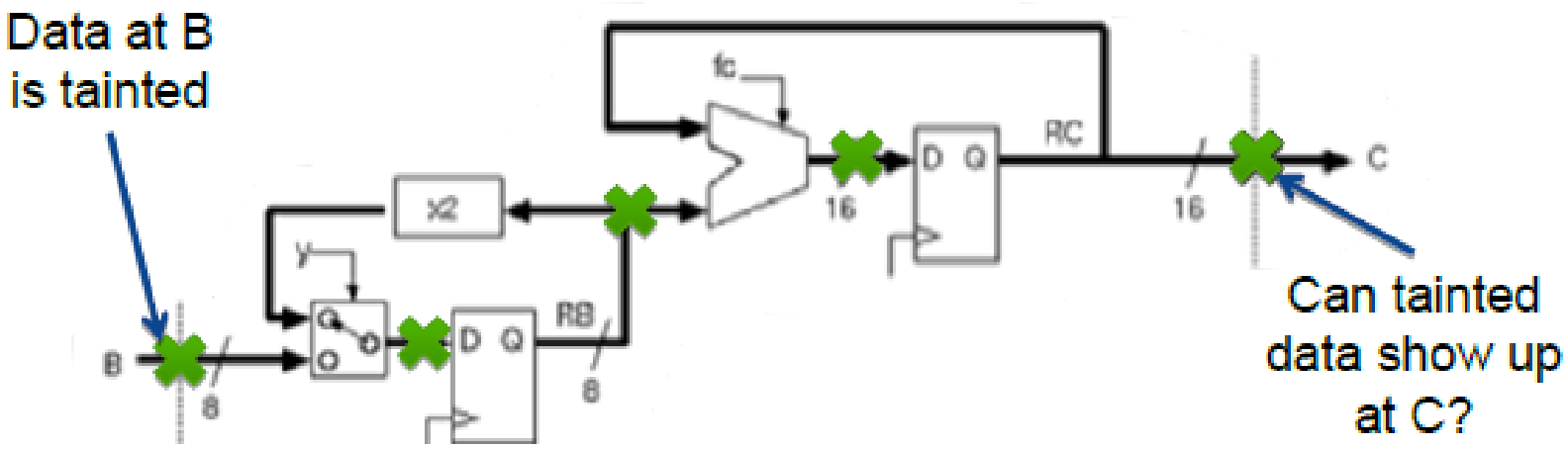
- Information Flow Security (IFS) Verification
- Power Side-channel Leakage Assessment (SCRIPT)
- Trojan Detection and Localization
- Susceptibility to Fault Injection (AVFSM)

Layout

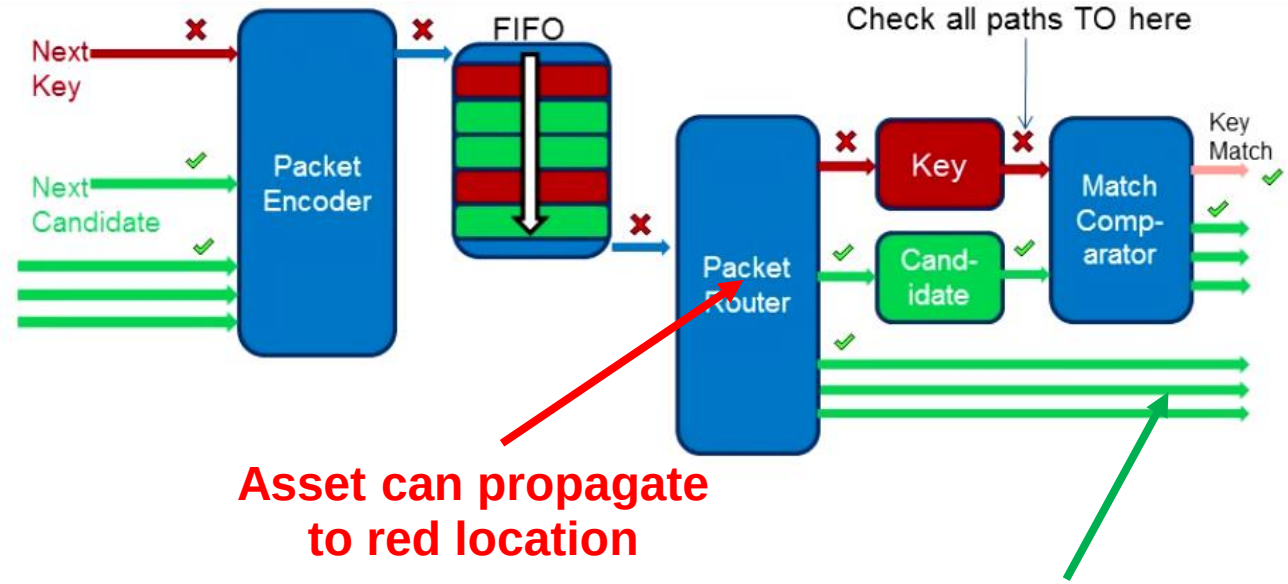
- Susceptibility of Probing Attacks
- Power Side-channel Leakage Assessment (TVLA)

Jasper Security Path Verification

- Jasper SPV accepts RTL containing a specific secure area (memory or registers), and exhaustively proves that secure data:
  - Can't be read illegally (no leaks)
  - Can't be illegally overwritten (sanctity)
- Dynamic methods (simulation) is often ineffective → activation of security bugs depend on the "hacking" ability
- Jasper utilizes unique path sensitization technology to detect security issues



Jasper Security Path Verification



Property Table

Filter on name

	Type	Name
✗	Assert(spv)	leakage_from_next_key_to_mem_wdata
✗	Assert(spv)	leakage_from_next_key_to_s1_encrypt_key
✗	Assert(spv)	leakage_from_next_key_to_s2_decrypt_key
✓	Assert(spv)	leakage_from_next_key_to_key_write_data
✓	Assert(spv)	leakage_from_next_key_to_packet_router_in
✓	Assert(spv)	leakage_from_next_key_to_mem_addr
✓	Assert(spv)	leakage_from_next_key_to_mem_enable
✗	Assert(spv)	leakage_from_next_key_to_mem_wdata_valid

	Name	Task	From	From Precond	To	To Precond
1	leakage_from_next...	security	mem_rdata	mem_enable && !mem_write	s2_decrypt.key	
2	leakage_from_next...	security	mem_rdata		mem_addr	
3	leakage_from_next...	security	mem_rdata		mem_enable	
4	leakage_from_next...	security	rom_data	rom_read && rom_addr <= 1	mem_wdata	mem_enable && mem_write
5	leakage_from_next...	security	mem_rdata		mem_wdata	

CAD for Security

RTL

- Susceptibility to Trojan Insertion
- Power Side-channel Leakage Assessment (RTL-PSC)
- EM Leakage Assessment
- Information Leakage (Jasper)
- **Formal Verification of Security Properties**

Gate

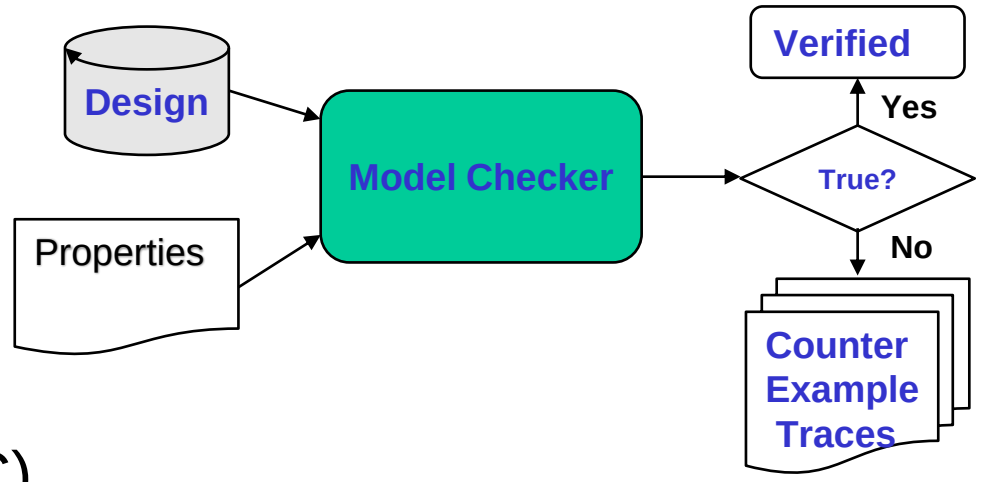
- Information Flow Security (IFS) Verification
- Power Side-channel Leakage Assessment (SCRIPT)
- Susceptibility to Fault Injection (AVFSM)
- Susceptibility to Trojan Insertion

Layout

- Susceptibility of Probing Attacks
- Power Side-channel Leakage Assessment (TVLA)

Security Verification using Model Checkers

- Security properties describe the expected behaviors which a trustworthy design is required to follow.
- Model checkers can be used to ensure safety properties.
- Example: Suppose that the program counter (PC) register is considered as a critical data. Valid ways update it:
  - Reset signal (V1)
  - CALL instruction which increments the PC (V2)
  - Using RET instruction which decrements PC register (V3)



$$Safe\_PC\_change: assert \quad always \quad PC_{access} \notin \mathbb{V} = \{V_1, V_2, V_3\} \rightarrow PC_t = PC_{t-1}$$

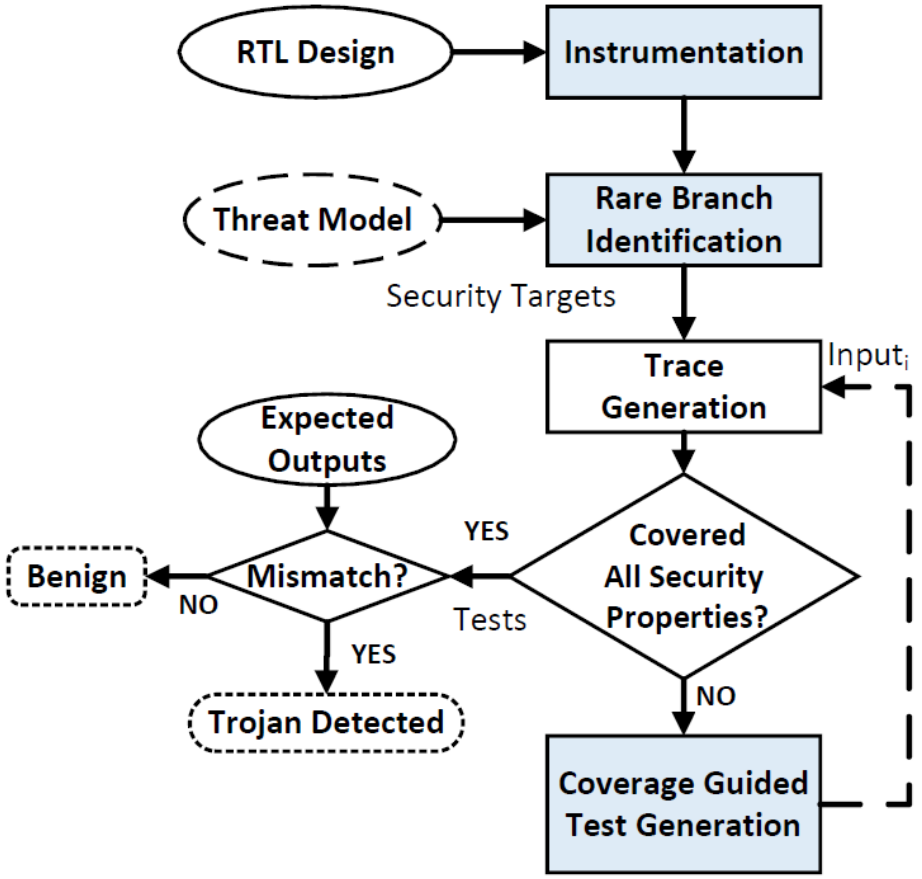
Trojan Activation by Interleaving Concrete Simulation and Symbolic Execution

- Goal: Given a RTL design, we need to generate test for covering all suspicious targets

```
func (a) {
    if (a == 5)
        activate Trojan
    else
        normal operation
}
```

Target

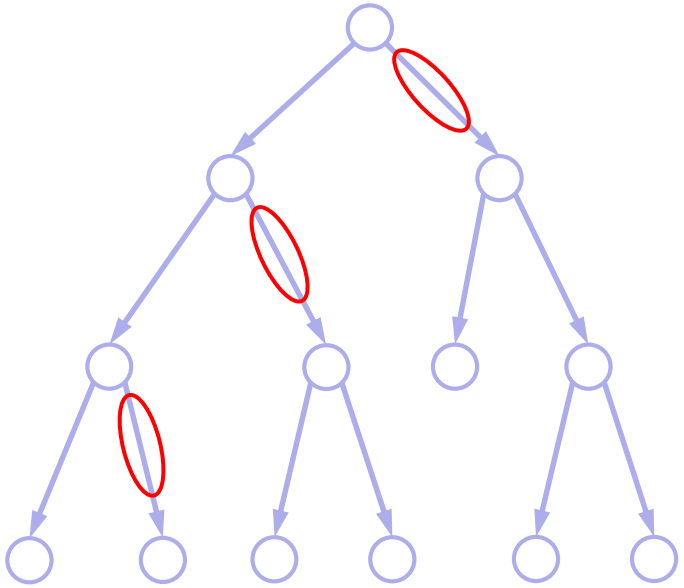
We want to generate test case to cover *target*



Background: Concolic Testing

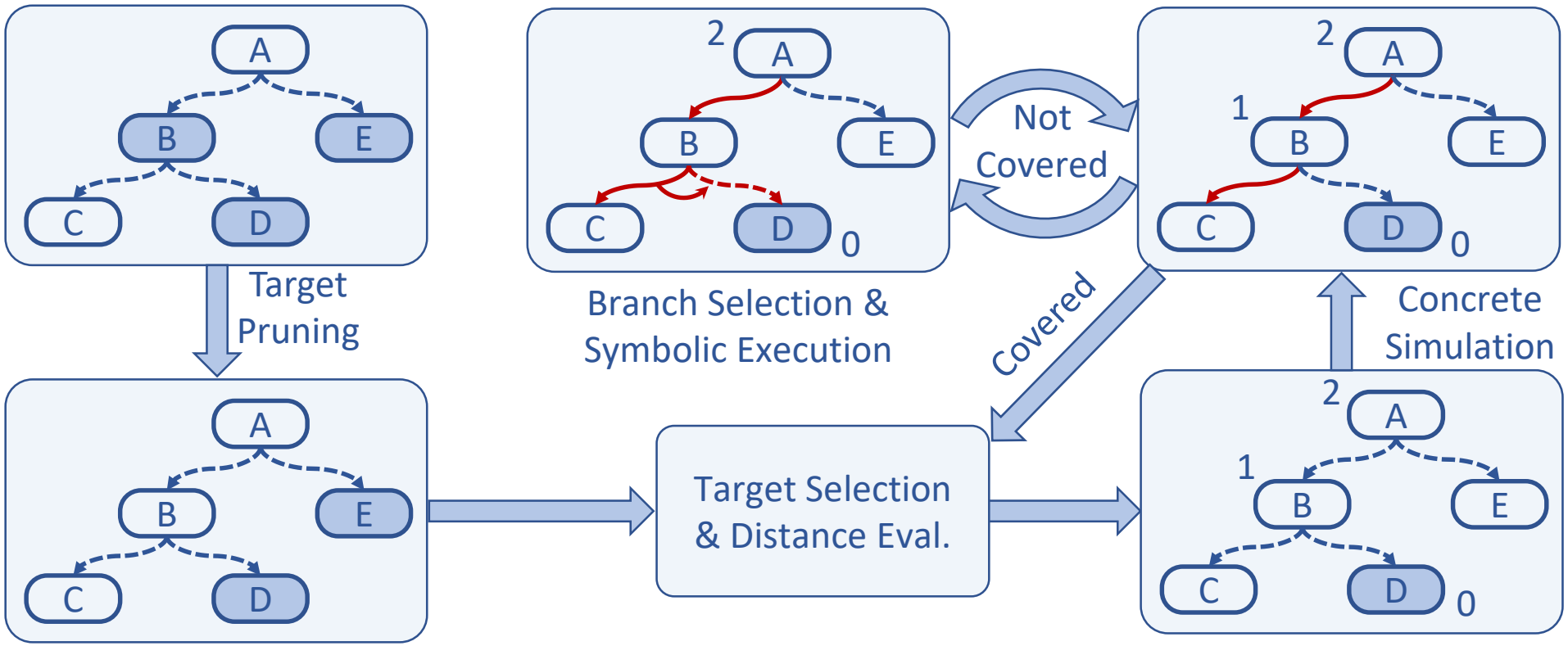
- Used **Concolic** testing:
  - Combines concrete simulation with symbolic execution

- Standard Concolic testing Steps:
  1. Simulate the design
  2. Select an adjacent branch
  3. Solve symbolically to get new input
  4. Repeat 1 with new input



Can we reduce the number of targets?
Which branch to explore next?

Test Generation for Trojan Detection



If target covered:

- Repeat target selection and distance evaluation phase

Evaluation of Coverage

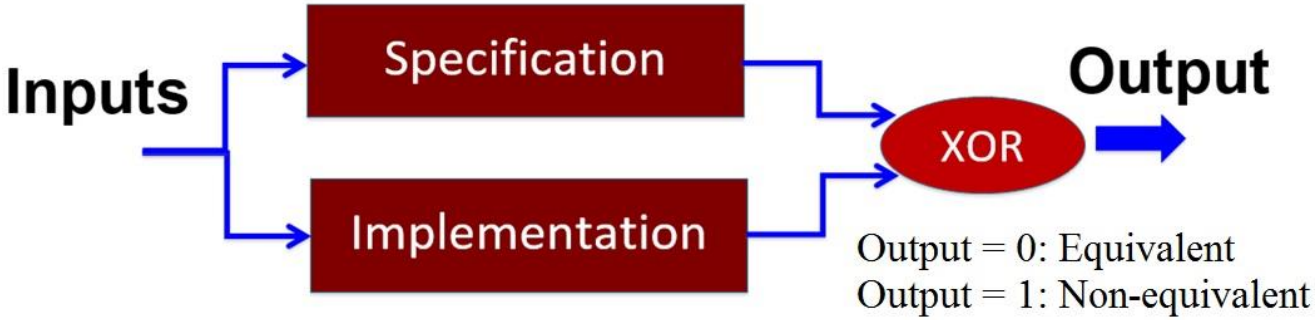
Benchmark	Cycles Unrolled	Lines of Code[1]	#Rare Branches	Rare Branches Coverage	EBMC [63] Time (sec)	EBMC [63] Memory (MB)	Our Approach Time (sec)	Our Approach Mem (MB)	Time Improvement	Memory Improvement
wb_conmax-T200	10	63 k	1	100.00%	8.71	659.5	13.36	124.7	-1.53x	5.29x
wb_conmax-T300	10	63 k	1	100.00%	11.77	1198.9	11.06	118.8	1.06x	10.09x
AES-T500	10	455 k	5	100.00%	67.07	7436	11.67	599	5.74x	12.41x
AES-T1000	10	456 k	2	100.00%	68.37	7441	3.88	525	17.62x	14.17x
AES-T1100	10	544 k	5	100.00%	71.03	7449	11.8	601	6.01x	12.39x
AES-T1300	10	456 k	9	100.00%	68.57	7449	2.65	524	25.87x	14.21x
AES-T2000	10	456 k	6	83.33%	69.27	7554	6.75	600	10.26x	12.59x
cb_aes_01	5	33 k	1	100.00%	1.27	179.4	0.51	55.3	2.49x	3.24x
cb_aes_05	10	167 k	1	100.00%	11.47	1450.3	4.03	244.3	2.84x	5.93x
cb_aes_10	15	334 k	1	100.00%	33.17	4130.6	14.47	502.4	2.29x	8.22x
cb_aes_15	20	501 k	1	100.00%	70.78	8041.2	32.14	778.2	2.20x	10.33x
cb_aes_20	25	668 k	1	100.00%	110.13	13202.8	86.03	1085.5	1.28x	12.16x
cb_aes_25	30	886 k	1	100.00%	-	MO	150.54	1405.3	-	-
cb_aes_30	35	1003 k	1	100.00%	-	MO	243.02	1780.3	-	-
cb_aes_35	40	1169 k	1	100.00%	-	MO	371.23	2112.7	-	-
cb_aes_40	45	1693 k	1	100.00%	-	MO	851.25	2532	-	-

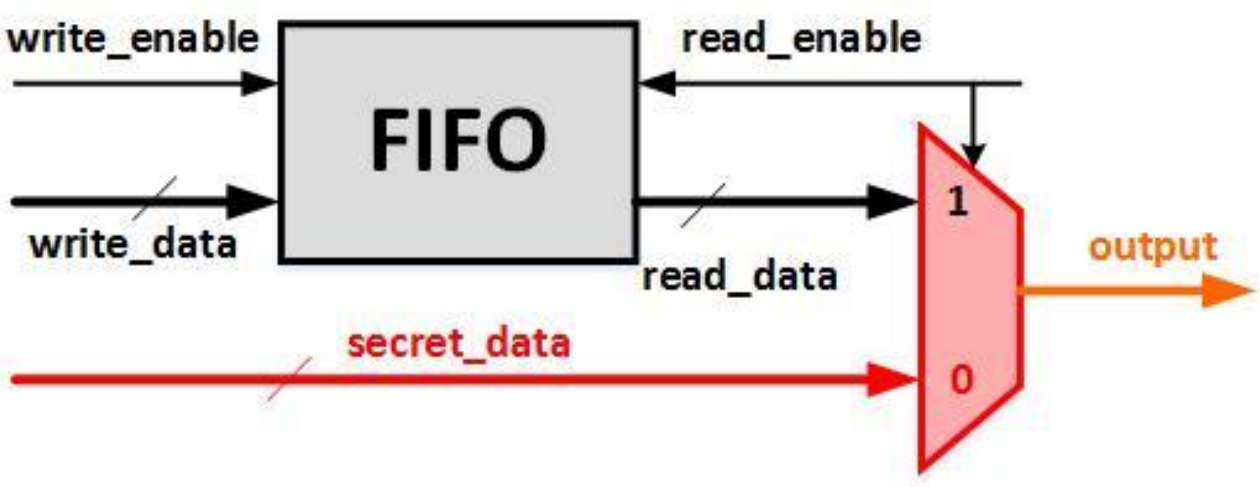
[1] After hierarchy flattening.

Full coverage in all benchmarks except AES-T2000

Trust Validation using Satisfiability Problem

- Check the equivalence between the specification of the circuit and its implementation using SAT-solvers
  - Outputs of the specification and implementation are XORed and CNF formula is generated
- Use SAT solvers to find existing Trojan in unspecified functionality
  - Trojan does not alter the specification





CAD for Security

RTL

- **Susceptibility to Trojan Insertion**
- Power Side-channel Leakage Assessment (RTL-PSC)
- EM Leakage Assessment
- Information Leakage (Jasper)
- Formal Verification of Security Properties

Gate

- **Information Flow Security (IFS) Verification**
- Power Side-channel Leakage Assessment (SCRIPT)
- Trojan Detection and Localization
- Susceptibility to Fault Injection (AVFSM)

Layout

- Susceptibility of Probing Attacks
- Power Side-channel Leakage Assessment (TVLA)

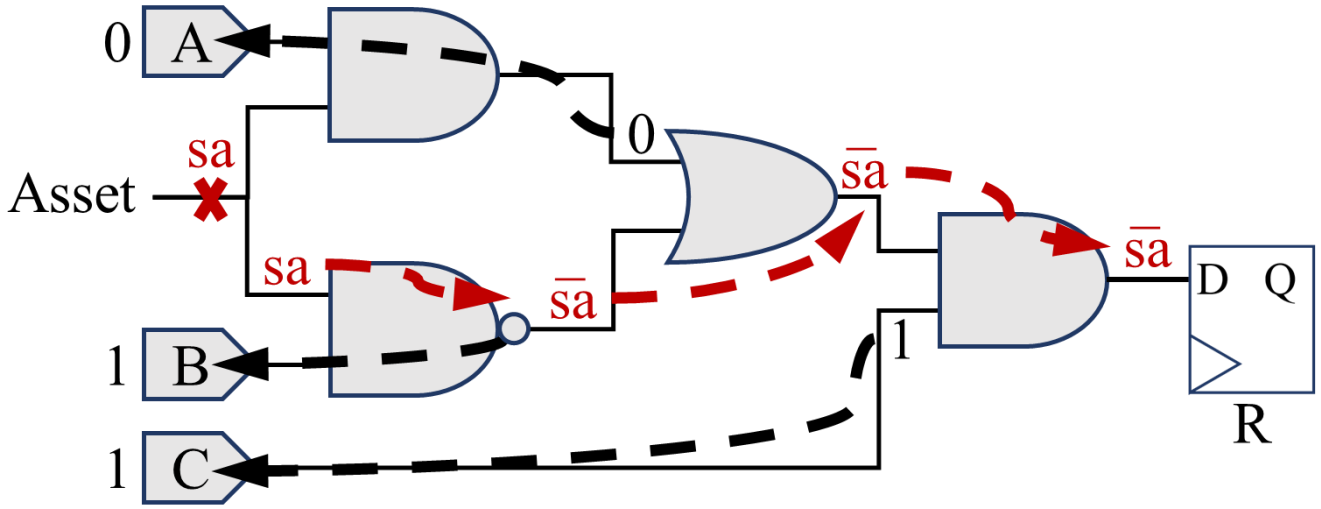
IFS Verification Challenges

- Traditional functional simulation, formal verification → **Not effective** for IFS verification

Functional simulation	Enumerating IFS leakage scenarios	• Non exhaustive • Expertise dependent
Formal verification	Properties for IFS verifications	• Difficult to write properties • False positive/negative • Limited Verification Capability

IFS Verification Framework

- Modeling an asset as **a stuck at fault**
- Utilize automatic test pattern generation (**ATPG)** algorithms to detect that fault
- A **successful** detection → Existence of information flow

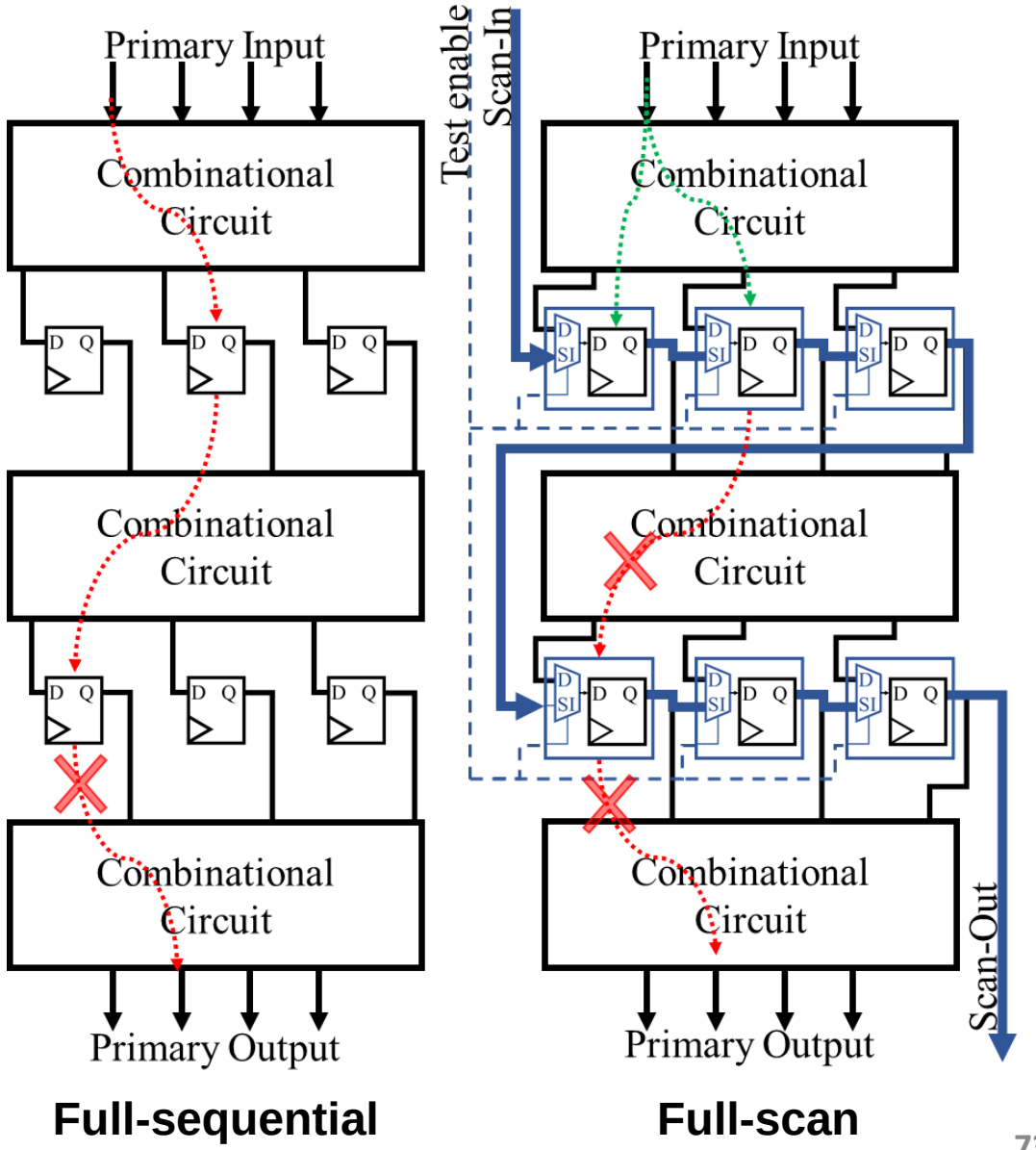


We need to identify all observe points through → Asset can be observed

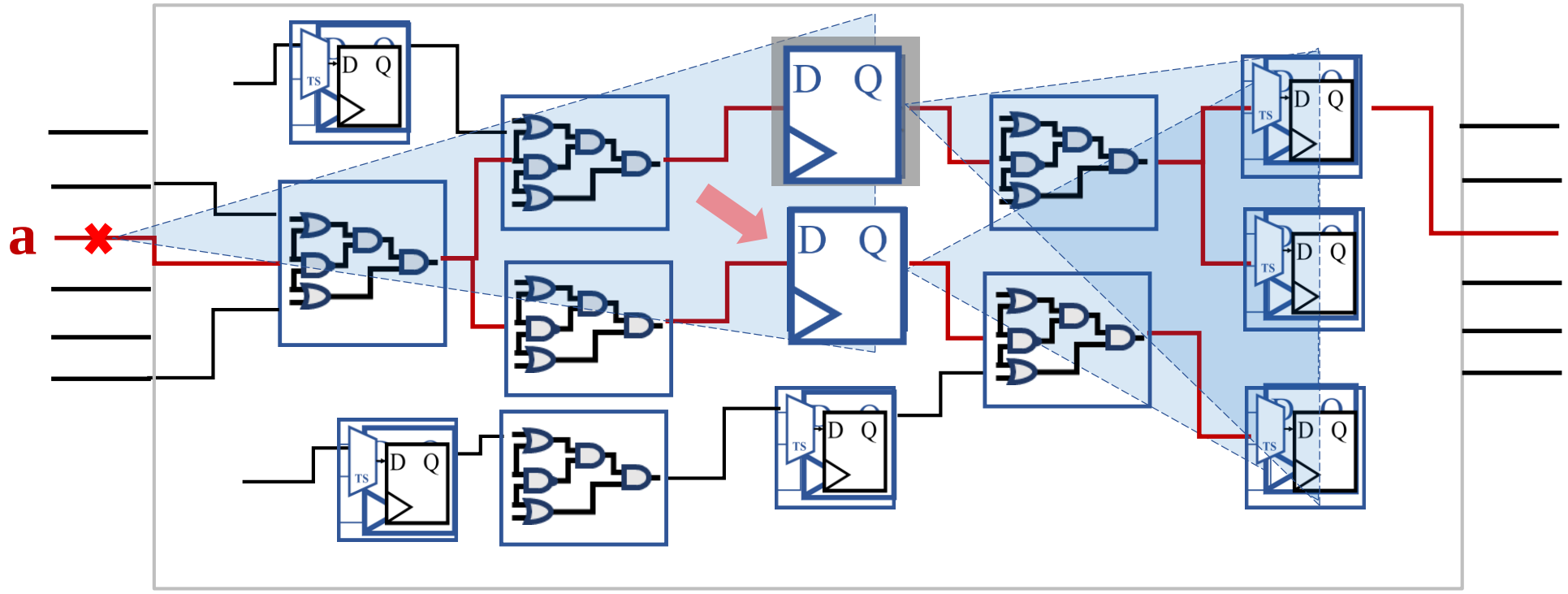
Choice of ATPG

- **Full-scan ATPG** → Detect asset propagation only to the **first level** FFs
  - Asset propagation to the subsequent level of FFs cannot be performed
- **Full-sequential ATPG** → High complexity and low efficiency
  - Cannot identify → **Registers**
- We propose a dynamic **partial-scan ATPG** technique
  - Identify all control/observe points

Partial-scan ATPG technique is only used for our IFS verification purpose



IFS Verification Framework: Confidentiality



- Add fault → a
- Set scan ability → all registers
- Fanout → a
- Add capture masks
- Successful → mark observe point
- Scan ability → off
- Fanout → Vulnerable register
- Run ATPG→ Sequential mode

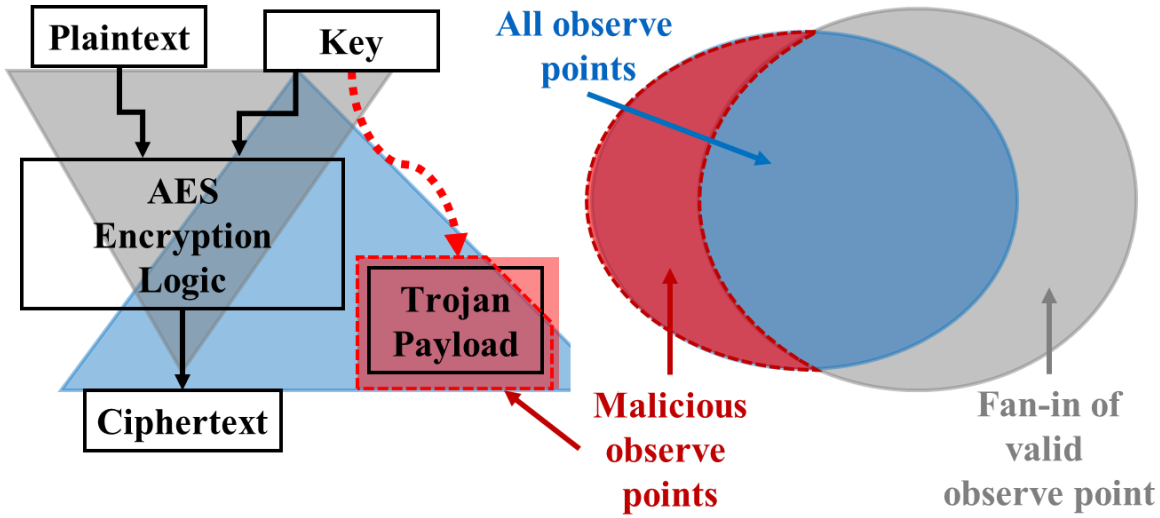
Confidentiality Analysis

Encryption Algorithm	Design	Seq. Elements	Observable Points	Distance		Stimulus	
				Min	Max	Min	Max
AES	high speed	10769	2	2	3	5	7
	small area	2575	4	2	2	6	6
	ultra-high speed	6720	2	0	1	2	3
Single-DES	small area	64	32	11	15	15	17
Triple-DES	small area	128	48	10	12	29	33
	high speed	8808	2	2	2	3	3
RSA	basic	555	32	4	3	6	6
PRESENT	light ware	149	2	2	2	3	3

- **Takeaways**
  - All implementation AES, RSA and PRESENT encryption module **have vulnerability due to DfT insertion**
  - The 'Distance' and 'Stimulus' → **quantitative measure** of vulnerability
  - **Higher** value → **less** vulnerable

IFS Trojan Detection

- A Trojan **violates** IFS policies
- Propagation depth → number of gates an asset propagates before reaching a **observe point**
  - Propagation depth for **Trojan** payload path will be much less
- Intersect Analysis → **malicious** observe points

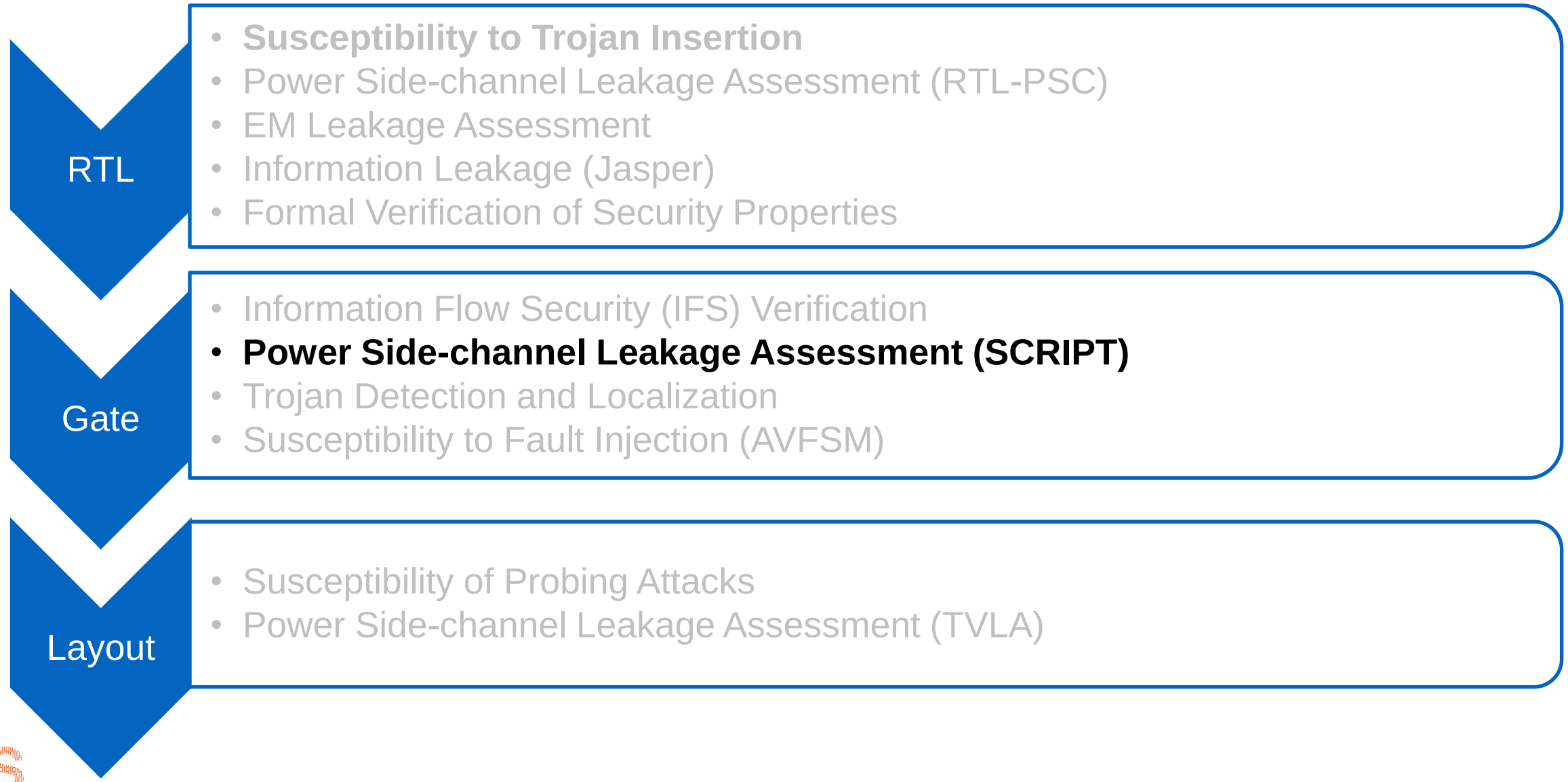


- Trigger Condition Extraction → Stimulus vector to propagate asset to a malicious observe point

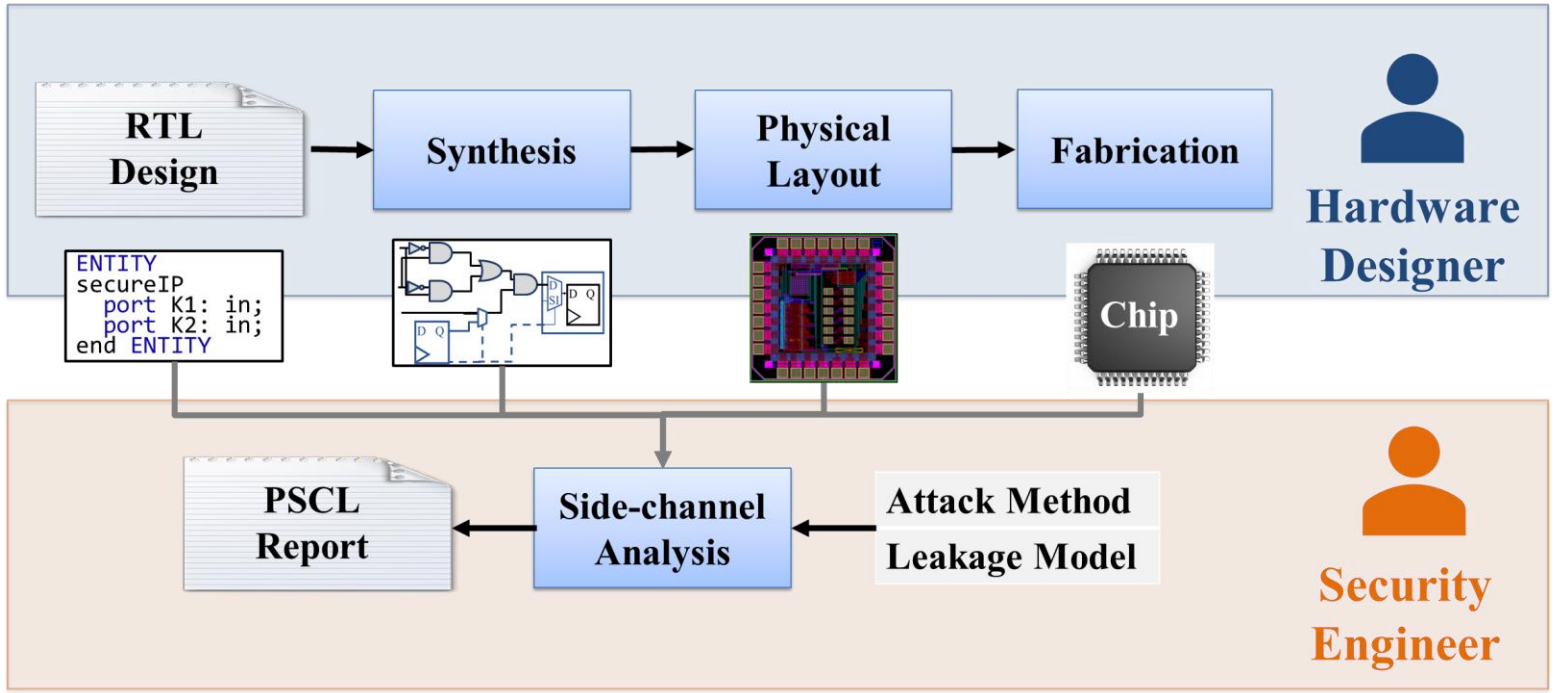
IFS Trojan Detection Results

Benchmark	Trojan payload	Trojan trigger	# of Observe points	# of Malicious points	Time (s)
AES-T100	Leaks the key through covert CDMA	Always on	42	16	251.5
AES-T200	Leaks the key through covert CDMA	Always on	42	16	273.8
AES-T700	Leaks the key through covert CDMA	Specific plaintext	42	16	277.1
AES-T900	Leaks the key through covert CDMA	Counter	42	16	293.7
AES-T1100	Leaks the key through covert CDMA	Plaintext sequence	42	16	362.9
AES-T2000	Leaks the key through shift register	Specific plaintext	35	1	240.5
AES-T2100	Leaks the key through shift register	Plaintext sequence	35	1	350.5
RSA-T100	Leaks the key through output	Specific plaintext	37	2	19.7
RSA-T300	Leaks the key through output	Counter	37	2	20.4

CAD for Security

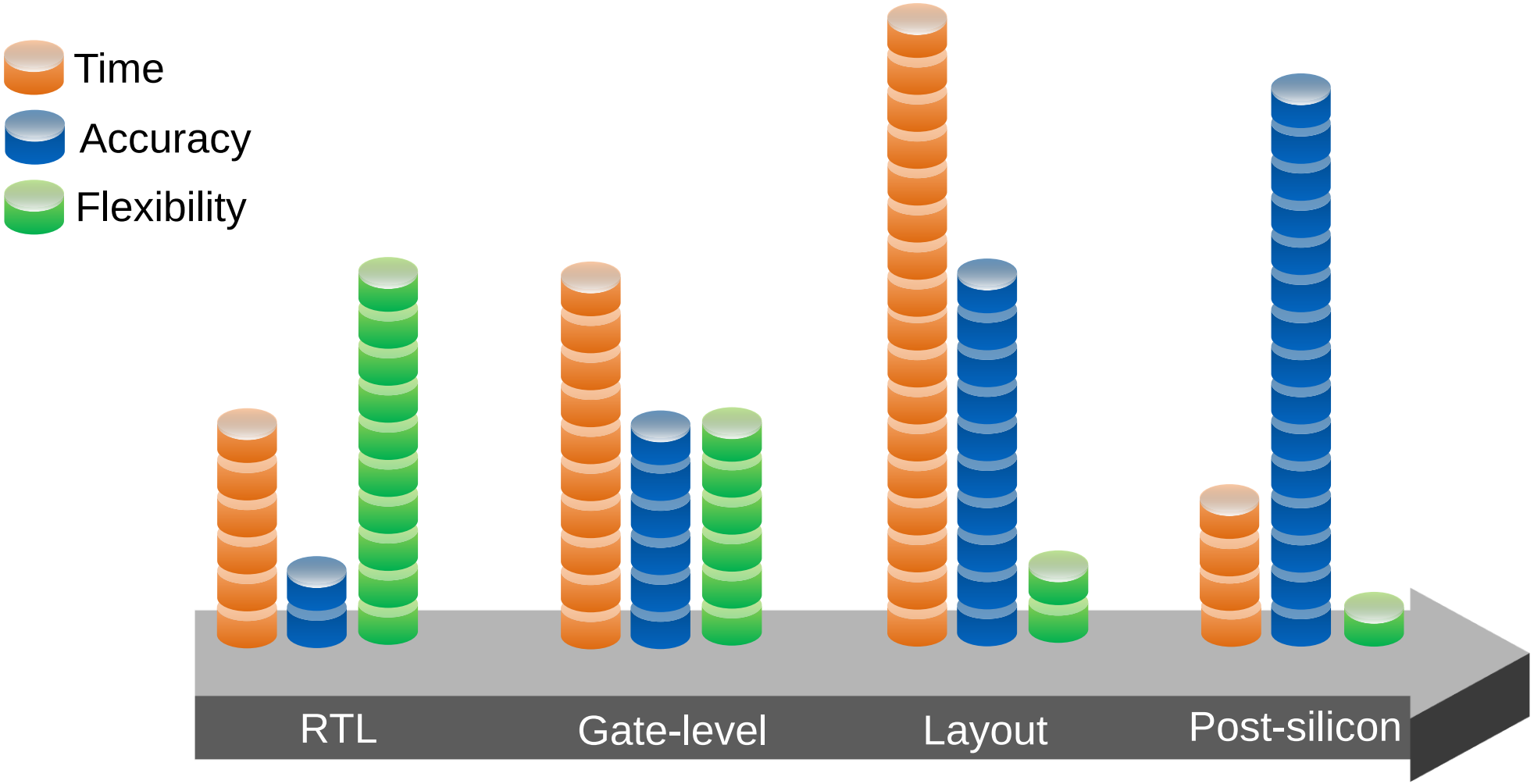


Traditional Power Side-channel Leakage (PSCL) Analysis



- Limitations
  - Security engineers with **significant** knowledge of side-channel attacks
  - Dependent on **expertise**
  - Focused on **post-silicon** assessment
  - Require tens of **thousands** of test vectors

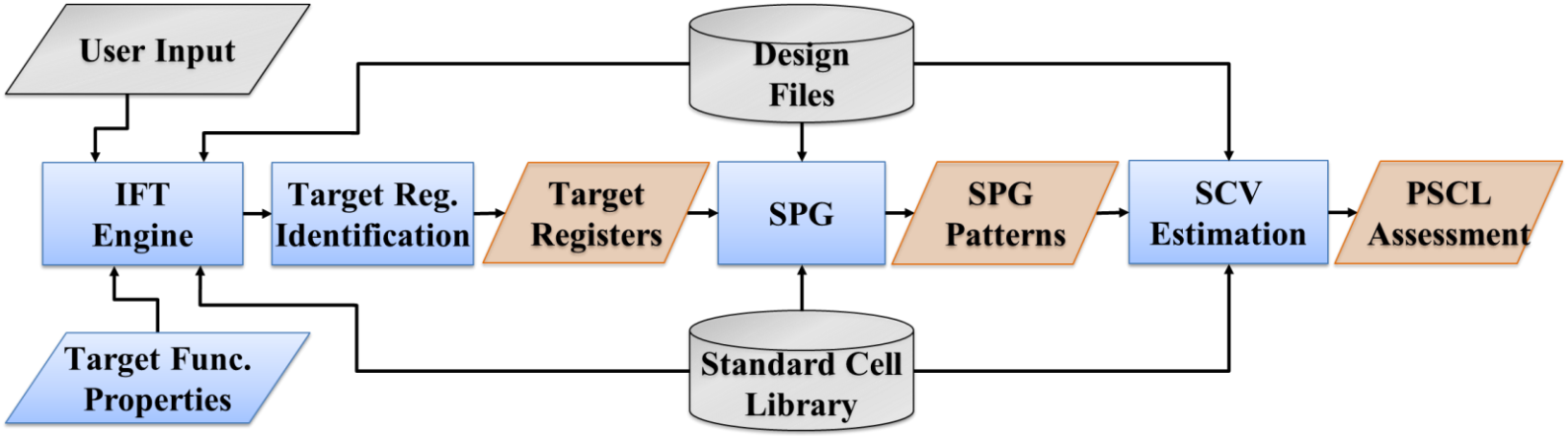
Pre- and Post-silicon PSCL Evaluation



- Need of **fast**, **accurate** PSCL assessment at **RTL/Gate-level**

SCRIPT: PSCL Assessment Tool

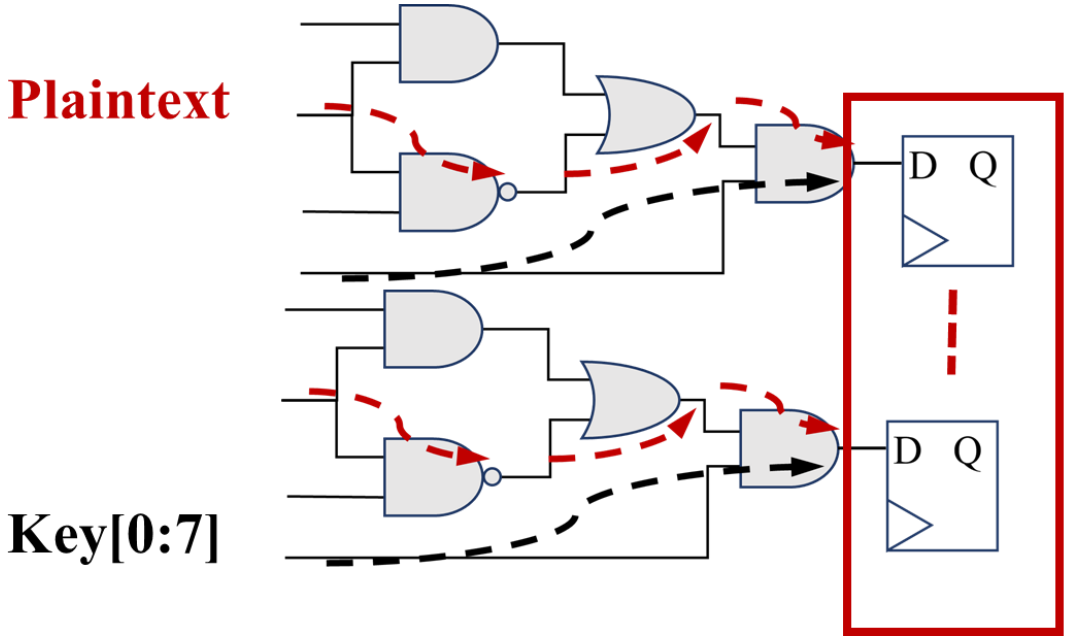
- Underlying **properties** causing side-channel leakage
- **Information flow tracking** → Registers exhibit the properties
  - **Automated** and applicable to **any** hardware design
- **SCV** metric for PSCL assessment
- **Formal verification** techniques to generate patterns for SCV
  - Accurate PSCL assessment with **two** patterns



Target Function

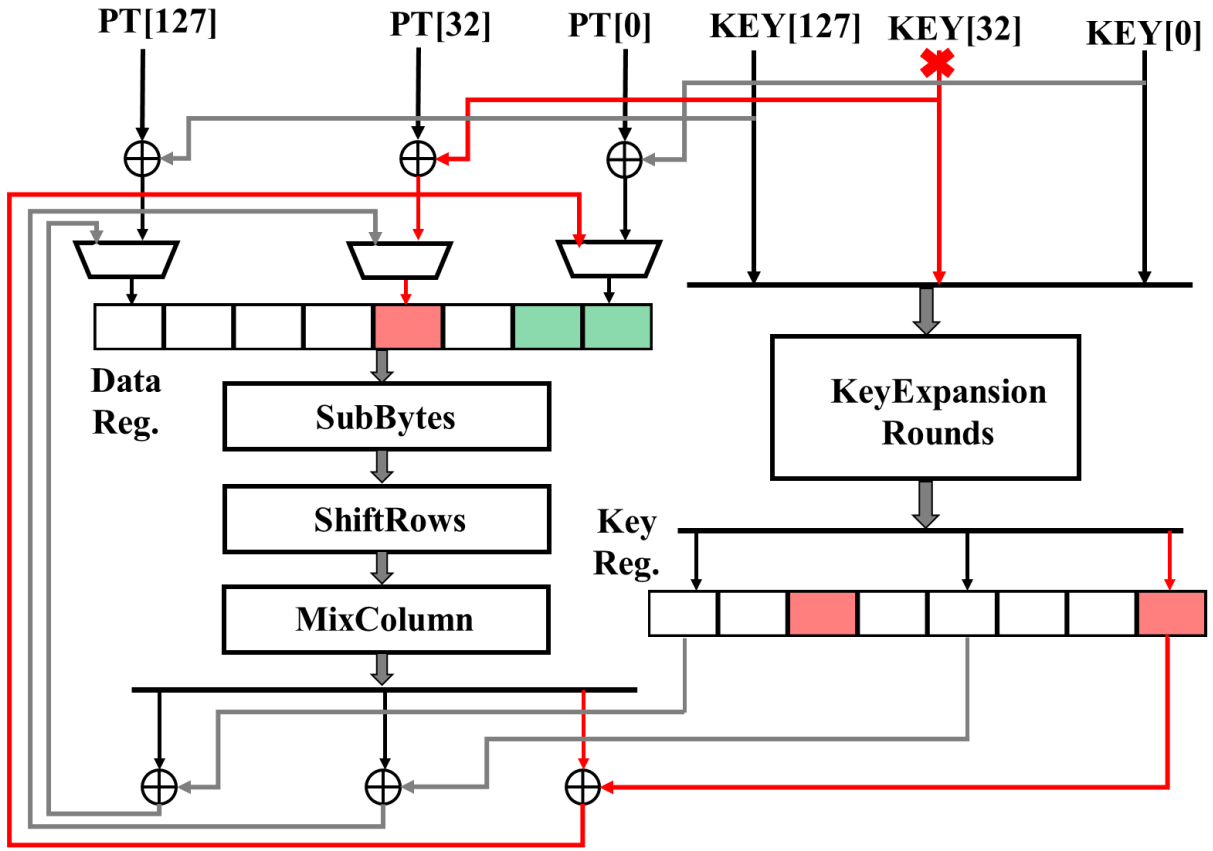
- Target Function **properties**

• **Secret** • Key of encryption operation	• **Controllability** • Plaintext of encryption operation
• **Confusion** • One output bit depends on multiple key bits	• **Divide-and-conquer** • Depends on a subset of key bits



Target Register Identification

- Utilized our **IFS** framework → Identify **Registers** that stores **Target function** (contribute to side-channel leakage)

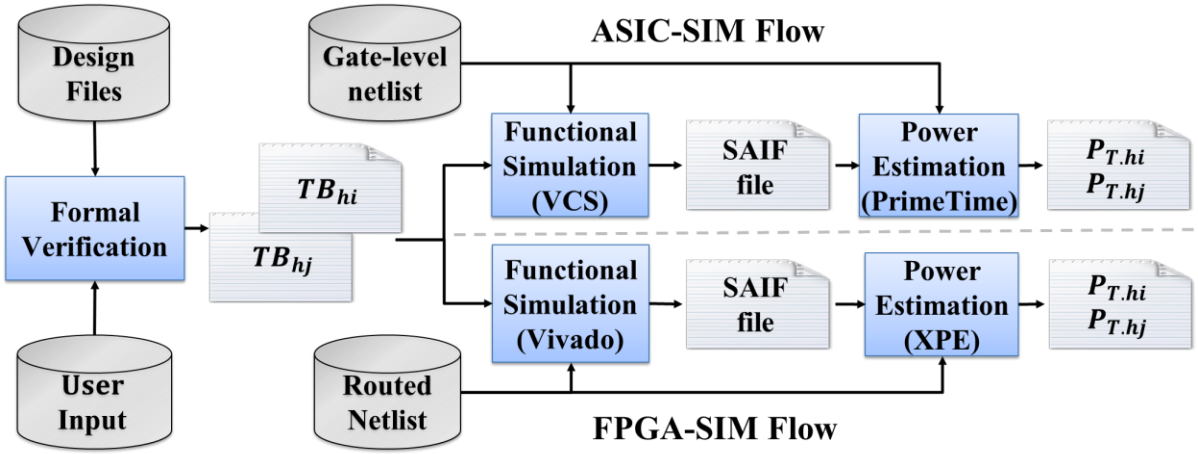


SCV Estimation

- **SCV metric** for PSCL assessment

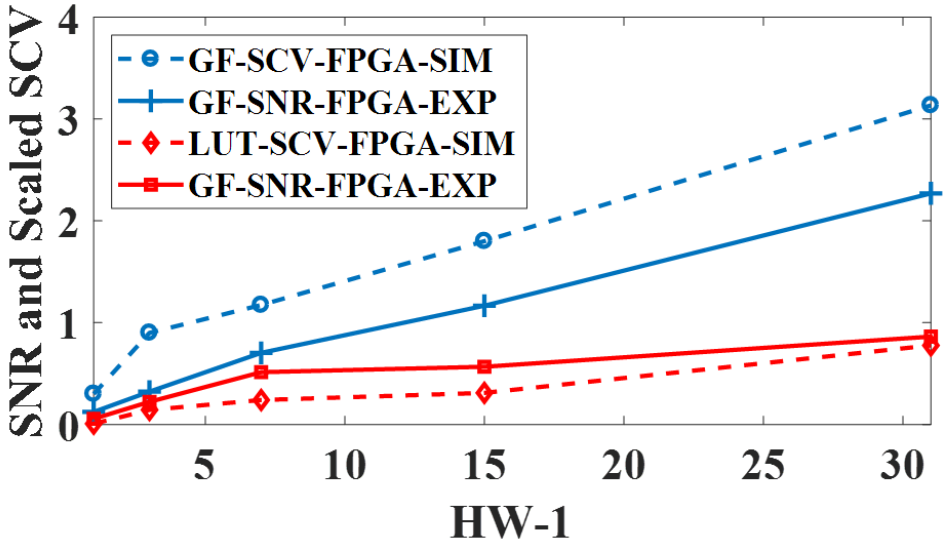
$$SCV = \frac{p_{signal}}{P_{noise}} = \frac{P_{T.hi} - P_{T.hj}}{P_{avg}}$$

- $P_{T.hi}, P_{T.hj}$ power of Target Function when $HW(i)$ and $HW(j)$
- Formal verification for $P_{T.hi}, P_{T.hj}$
  - **Only** introduce switching in logic related to target function
- **Static** power analysis to estimate P_{noise}



Results

- PSCL assessment of AES-GF and AES-LUT
- SCV metric calculated by **SCRIPT** → **2** plaintexts
- Experimentally evaluated SNR metric → **10,000** plaintexts

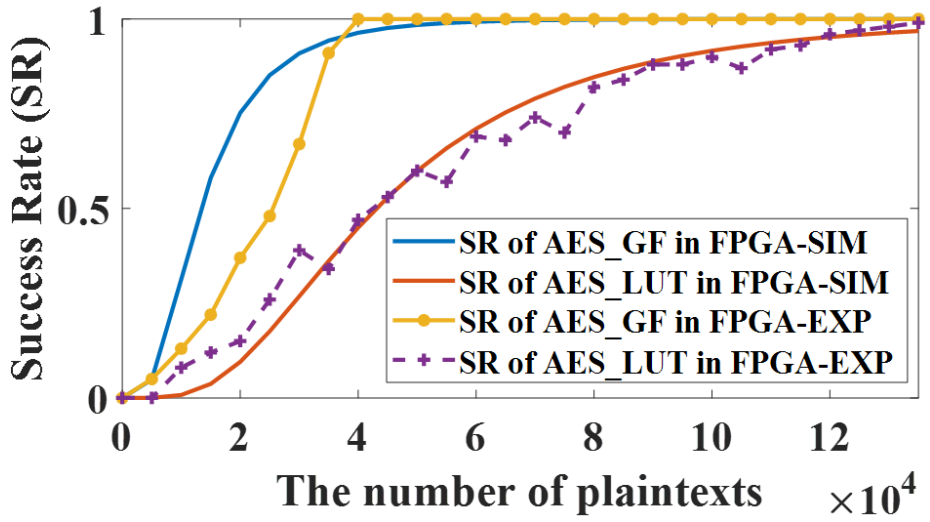


Metric	Stage	Time
SCV	Routed design	**14 mins**
SNR	Routed design	**31 days**

- Takeaways
  - Correlation coefficient → AES-GF is **0.99**, AES-LUT is **0.94**
  - SCV can **accurately** assess power side-channel leakage

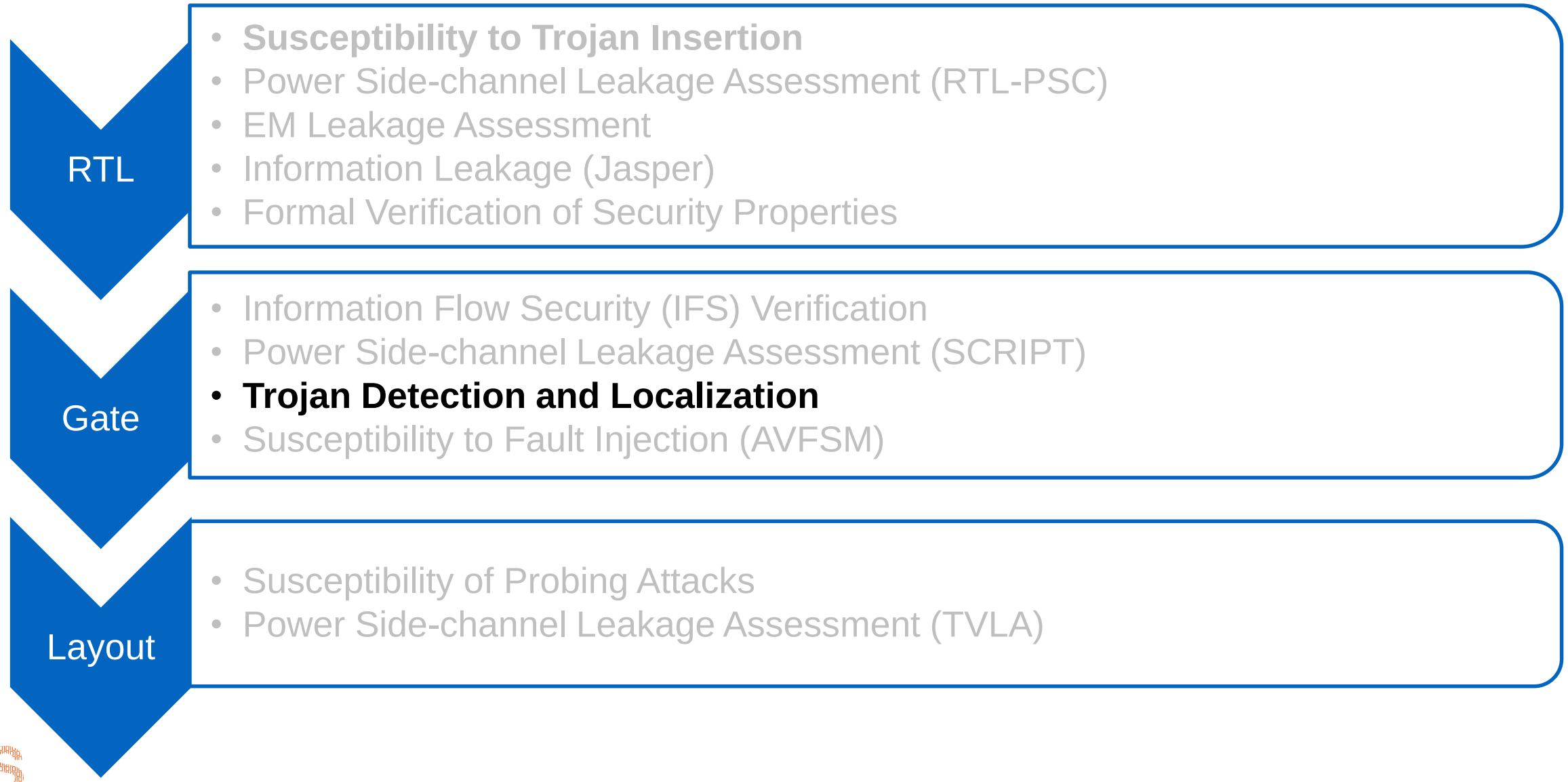
Results

- Derived theoretical attack success rate (SR) from **SCV**
  - Side-channel attack success probability w.r.t. no. of plaintexts
- **Experimentally** evaluated SR from 100 CPA attack

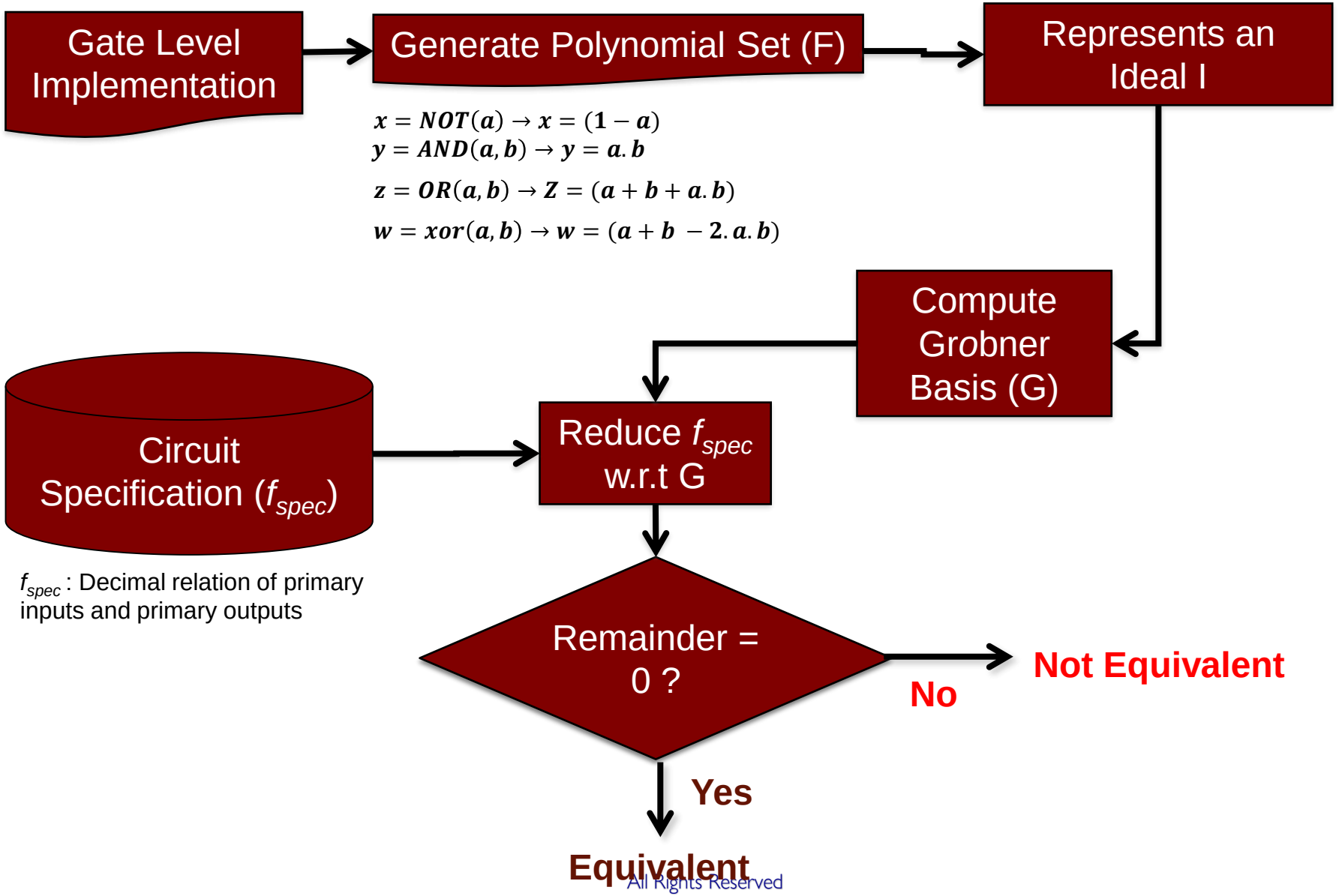


- **Takeaways**
  - Correlation coefficient between SR estimated by SCRIPT and experimentally evaluated SR → AES-GF is **0.93**, AES-LUT is **0.99**

CAD for Security

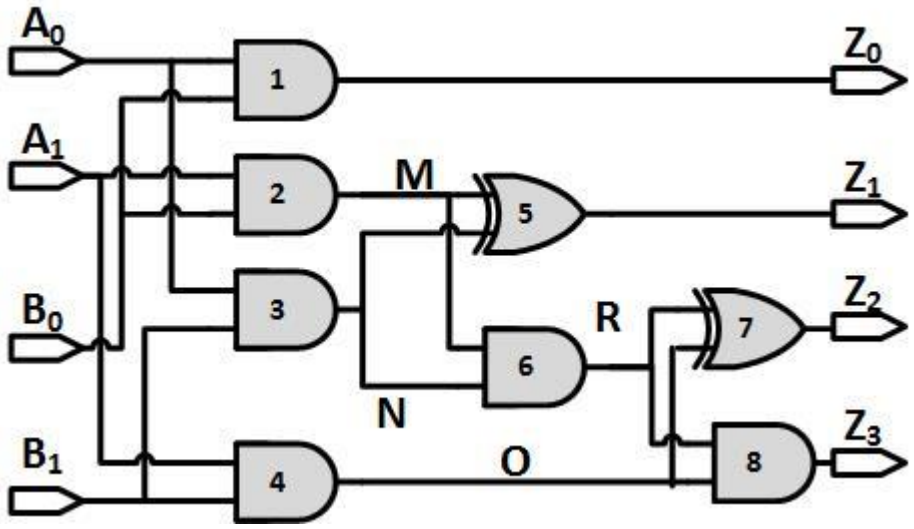


Background: Existing Arithmetic Circuits Verification



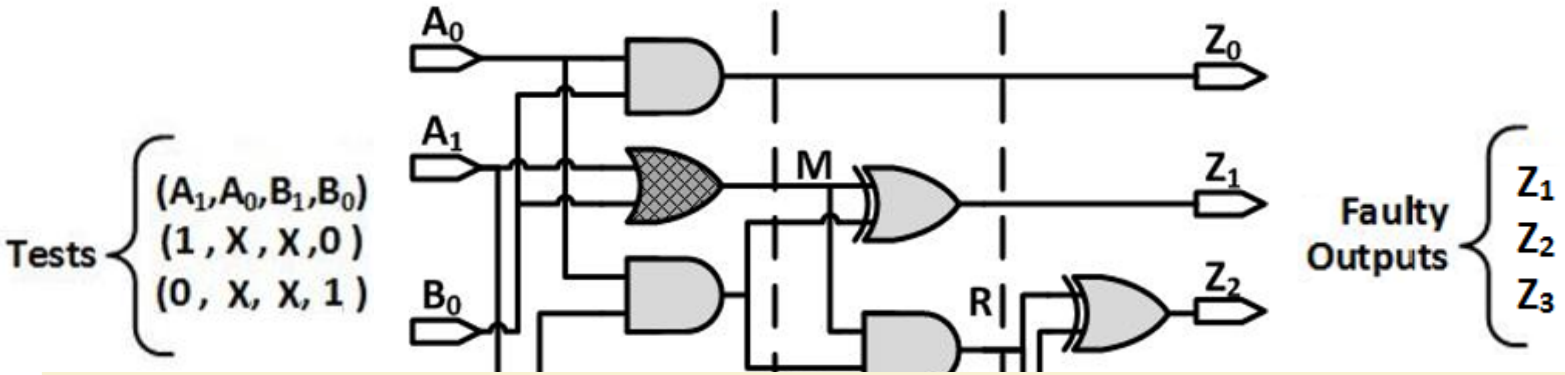
Background: Equivalence Checking using Symbolic Algebra

- Consider a 2-bit Multiplier specification
  - $f_{spec} := Z - (A.B)$
    - $\mathbf{Z} = 8.Z_3 + 4.Z_2 + 2.Z_1 + Z_0$
    - $\mathbf{A} = 2.A_1 + A_0, \mathbf{B} = 2.B_1 + B_0$
- Model gates as polynomials
- Order:
  - $\{Z_2, Z_3\} > \{Z_1, R\} > \{Z_0, M, N, O\} > \{A_1, A_0, B_1, B_0\}$
- Verification Steps:
  - $f_{spec}: 8.Z_3 + 4.Z_2 + 2.Z_1 + Z_0 - 4.A_1.B_1 - 2.A_1.B_0 - 2A_0B_1 - A_0B_0$
- Cancel Z_2 and Z_3
  - Step1: $4.R + 4.O + 2.Z_1 + Z_0 - 4.A_1.B_1 - 2.A_1.B_0 - 2A_0B_1 - A_0B_0$
- Cancel R and Z_1
  - Step 2: $4.O + 2.M + 2.N + Z_0 - 4.A_1.B_1 - 2.A_1.B_0 - 2A_0B_1 - A_0B_0$
- Cancel Z_0, M, N, O
  - Step 3: (remainder): 0



Background: Trojan-inserted Implementation

- Consider a buggy 2-bit Multiplier
  - $f_{spec} := output - (A.B) = 0$
  - $f_{spec} := 8.Z_3 + 4.Z_2 + 2.Z_1 + Z_0 - ((2.A_1 + A_0).(2.B_1 + B_0))$

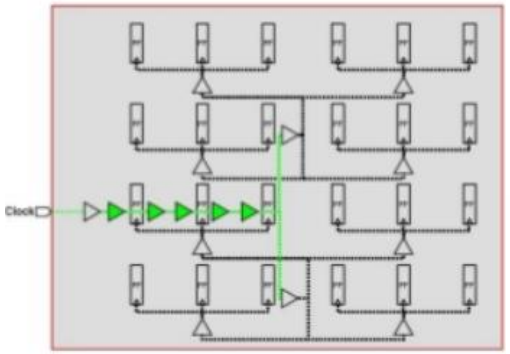


OR gate function with inputs $A_1, B_0 : M = (A_1 + B_0 - A_1.B_0)$
AND gate function with inputs $A_1, B_0 : M^* = (A_1.B_0)$
Difference: $M - M^* = (A_1 + B_0 - 2.A_1.B_0)$

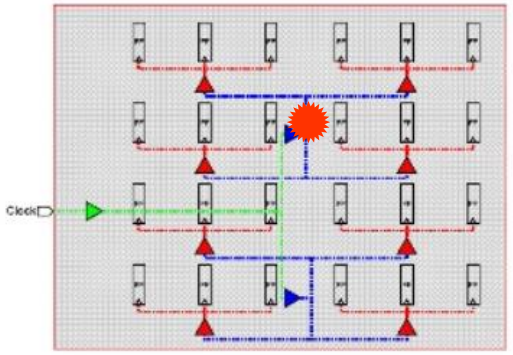
$f_{spec_3}(remiander) : 2.A_1 + 2.B_0 - 4.A_1.B_0$

Trojan Localization using Symbolic Algebra

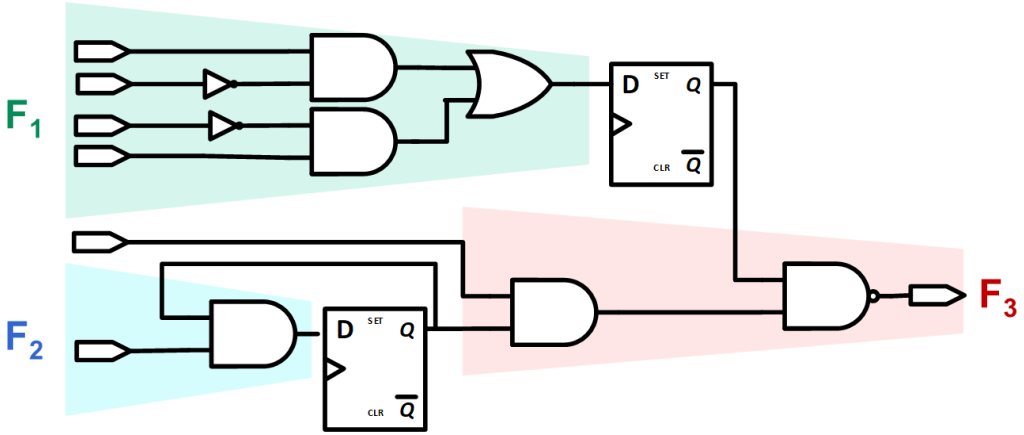
- Design after non-functional changes should be validated to check are not inserted

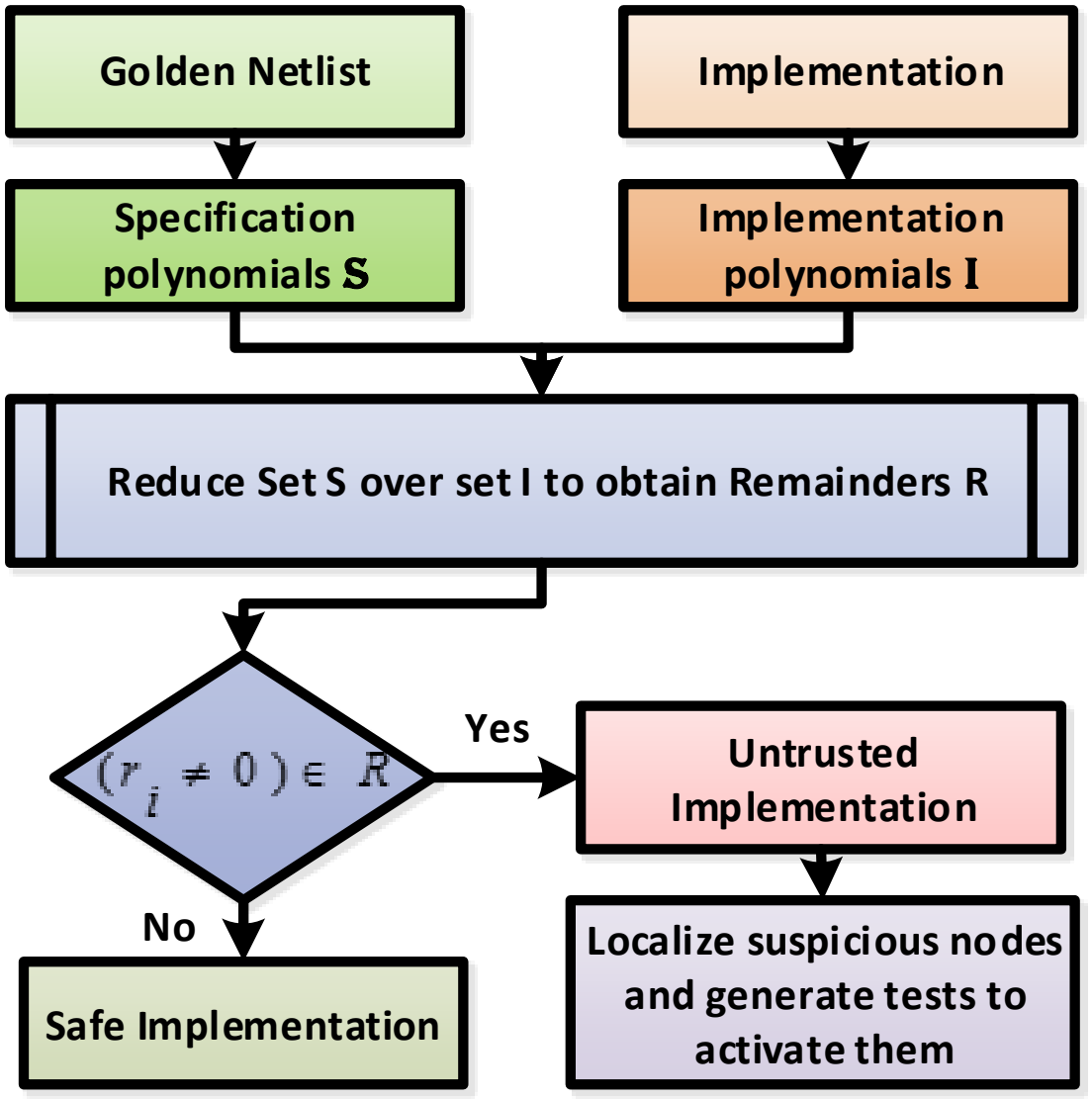


Verified design



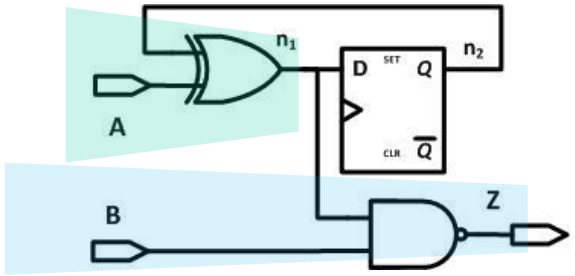
After clock tree insertion





Example: Trojan Localization using Symbolic Algebra

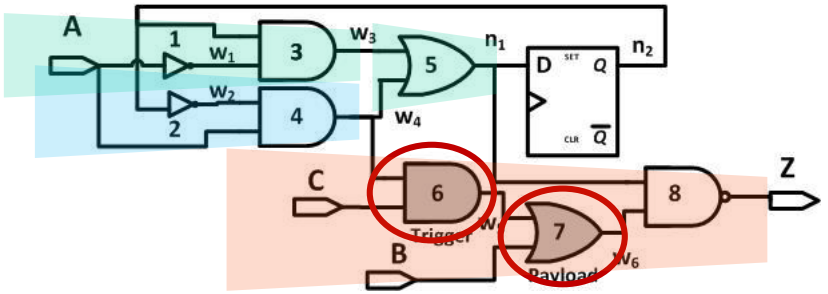
Specification:



Implementation

?

=



Specification Polynomials

F_{spec1}: $n_1 - (A+n_2-2*A*n_2)=0$

F_{spec2}: $Z - (n_1*B)=0$

Implementation polynomials

F_1: $n_1 - (n_2*w_4*A - n_2*w_4 + w_4 - n_2*A)=0$

F_2: $w_4 - (A - n_2*A)=0$

F_3: $Z - (n_1*w_4*C*B - + n_1*w_4*C - n_1*B+1)=0$

- Fspec1 will be reduced to zero
  - Gates {1,2,3,4,5} which construct the Fspec1 are safe
- Reduction of Fspec2 results in a non-zero remainder
  - Gates {2,4,6,7,8} which construct the Fspec2 are suspicious

Results: Trojan Localization

Benchmark			#Suspicious Gates			False Positives	False positive Improvement	
Type	Gates	#Trojan Gates	FANCI	Formality	Ours	Our	FANCI	Formality
RS232-T1000	311	13	37	**214**	13	0	*	*
RS232-T1100	310	12	36	213	14	2	**12x**	**100.5x**
S15850-T100	2456	27	76	710	27	0	*	*
S38417-T200	5823	15	73	2653	26	11	**5.27x**	**239.8x**
S35932-T200	5445	16	70	138	22	6	**9x**	**20.3x**
S38584-T200	7580	9	**85**	**47**	**11**	**2**	**38x**	**19x**
Vga-lcd-T100	70162	5	**706**	**	**22**	**17**	**41.2x**	**

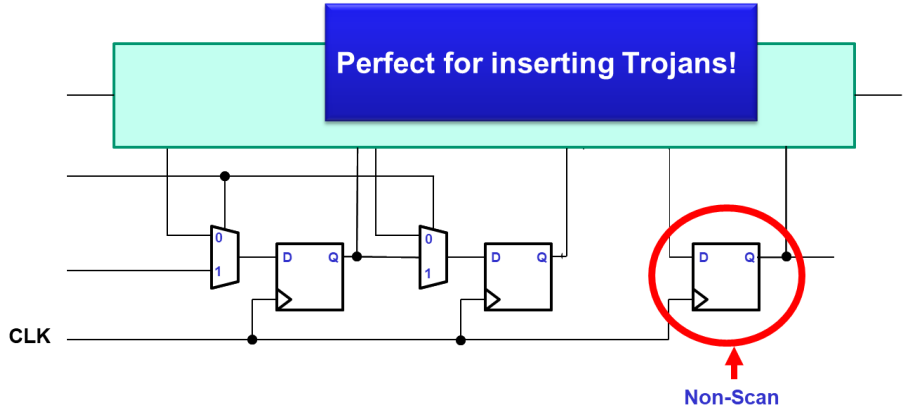
"*" indicates our approach does not produce any false positive gates (infinite improvement)

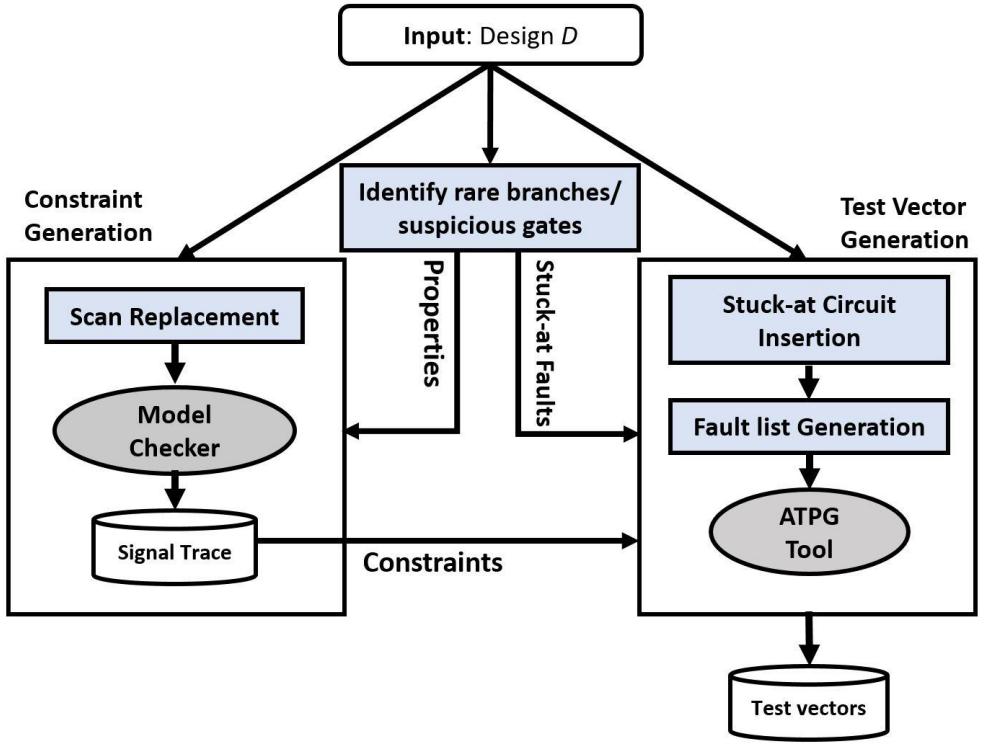
"**" shows the cases that Formality could not detect the Trojans.

[FANCI] A. Waksman et al., "Fanci: Identification of stealthy malicious logic using Boolean functional analysis," in CCS, 2013.

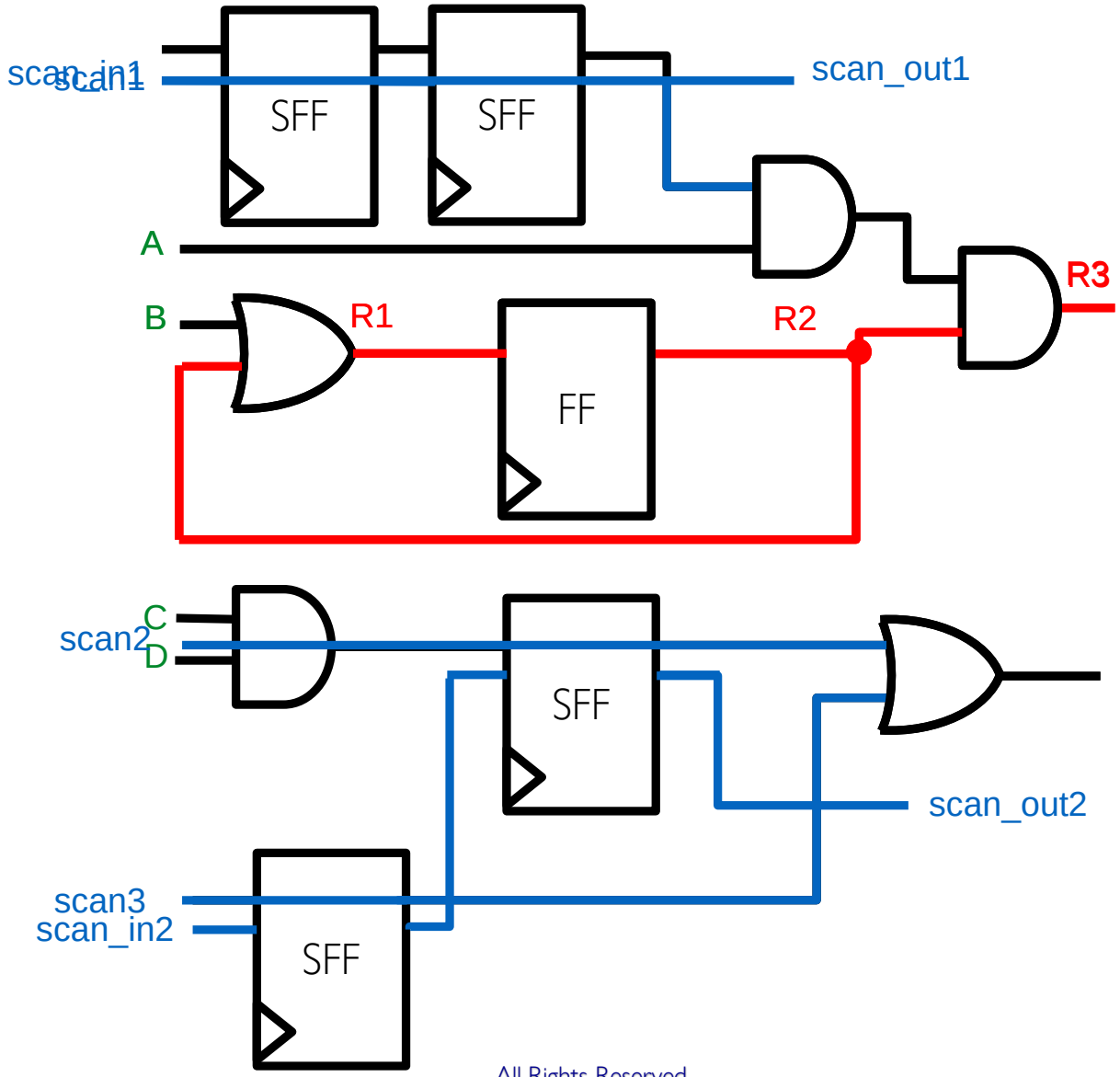
Hardware Trojan Detection using ATPG and Model Checking

- ATPG performance generally suffers in the presence of non-scan sequential elements.
- Model Checking is used to generate constraint structures to be used in ATPG.
  - Mitigate state explosion with scan replacement.
- Rare-node identification:
  - Functional Simulation up to millions of cycles
  - Calculate signal probabilities
  - All nets that are below a threshold value are identified as rare

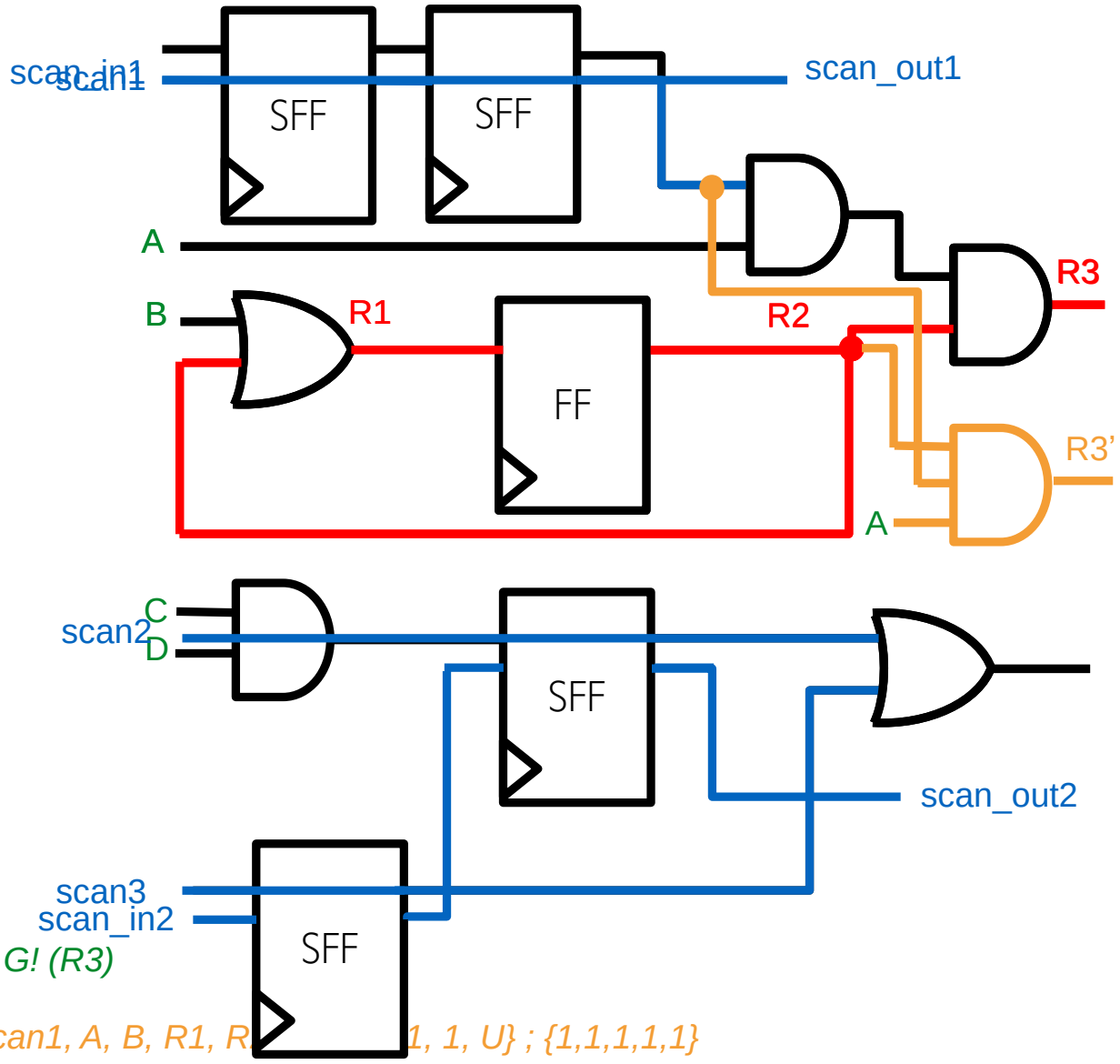




Constraint Generation



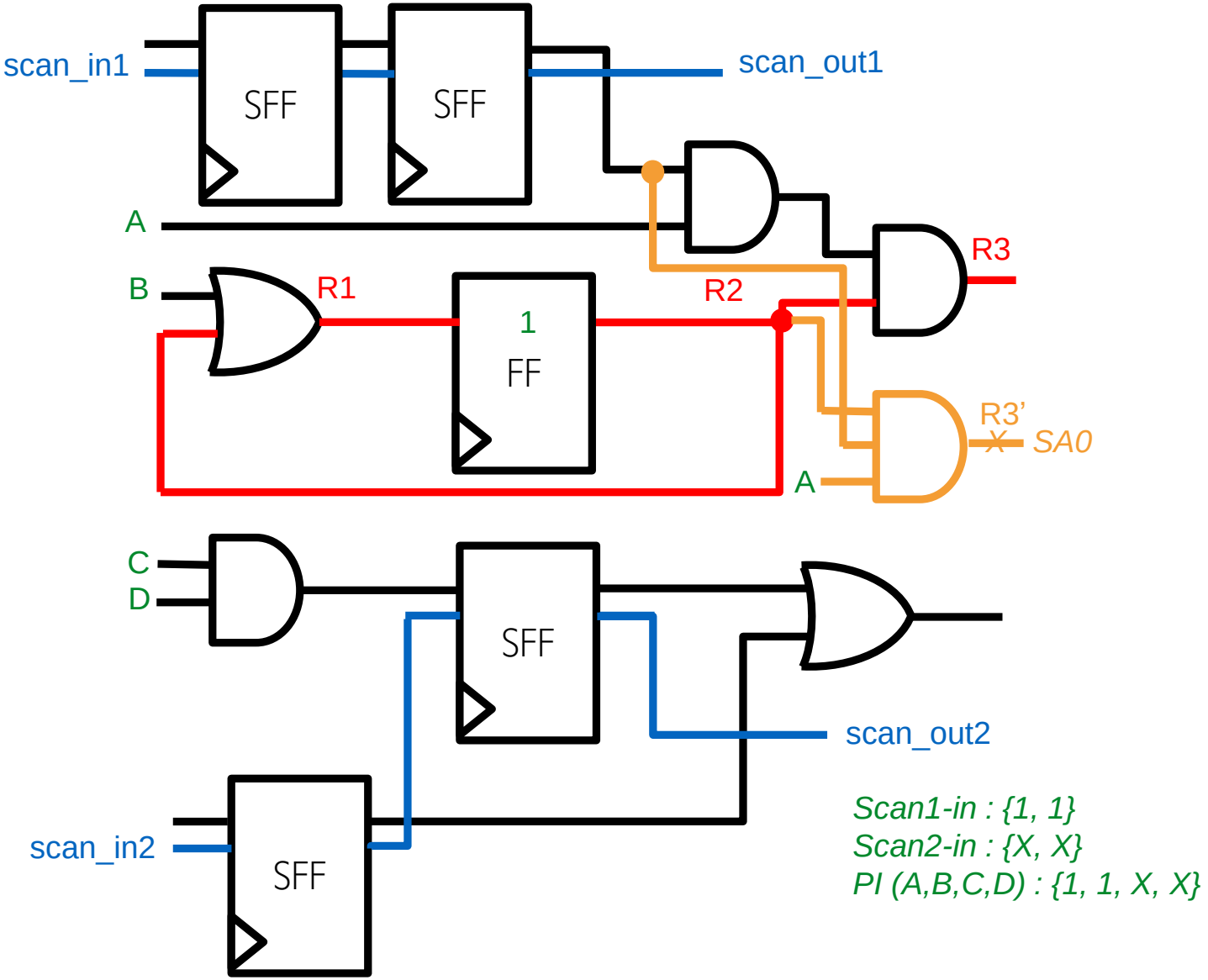
Constraint Generation



Property: assert G! (R3)

Signal Trace: {scan1, A, B, R1, R… 1, 1, U} ; {1,1,1,1,1}

Test Generation

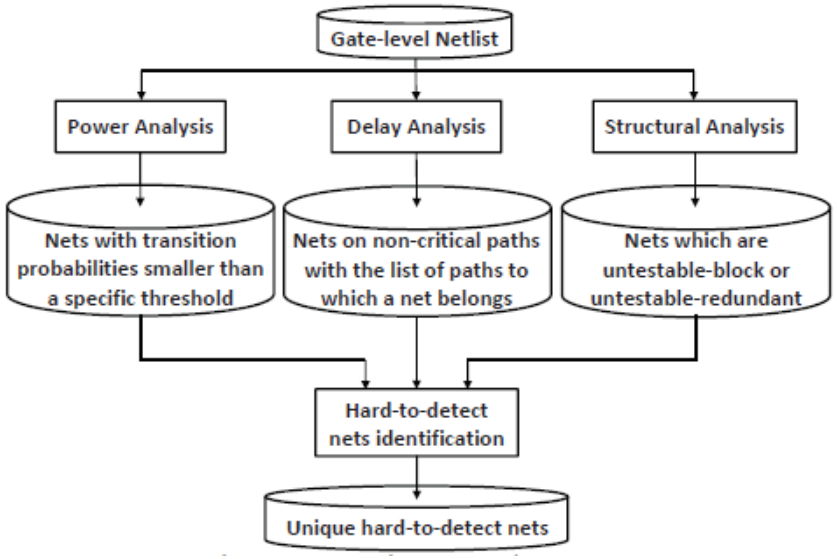


Trojan Detection using Combined Approach

Benchmarks	Scan FFs (Scan/Total)	Test Cov.	# Rare Bran.	ATPG		Model Chk (MC)		MERO		Our Approach	
				Detect	Time	Detect	Time	Detect	Time	Detect	Time
AES-T1000	93%	99%	2	√	0.02s	X*	85.86s	X	TO	√	8.8s
AES-T2000	91%	99%	5	√	0.90s	X*	216.5s	X	TO	√	22s
RS232-T400	51%	97%	2	√	0.24s	√	1 hour	√	2810s	√	0.52s
RS232-T800	45%	97%	1	√	0.06s	√	7.233s	√	3157s	√	0.12s
cb_aes_15	85%	99%	1	√	8 hours	X	BDD limit	X	15720s	√	7.85s
cb_aes_20	93%	99%	1	√	8 hours	X	BDD limit	X	16740s	√	38.3s

Susceptible to Trojan Insertion

- The vulnerability analysis flow reports unique hard to-detect nets
  - Untestable nets with low transition probabilities
  - Nets with low transition probabilities on non-critical paths
- It is expected that Trojan trigger inputs are supplied by these nets to reduce activity inside the Trojan circuit.



Circuit	Power Analysis					Delay Analysis		Fault Analysis
	# Nets	< 0.1	< 0.01	<0.001	<0.0001	Critical path C(pF)	< 70% of Critical Path C	Untestable faults
b19	70,259	14,482	8,389	5,533	4,530	0.377	474,358	8
s38417	5,669	589	291	219	69	0.050	41,901	0
s38584	7,203	817	197	85	30	0.044	27,689	0

CAD for Security

RTL

- **Susceptibility to Trojan Insertion**
- Power Side-channel Leakage Assessment (RTL-PSC)
- EM Leakage Assessment
- Information Leakage (Jasper)
- Formal Verification of Security Properties

Gate

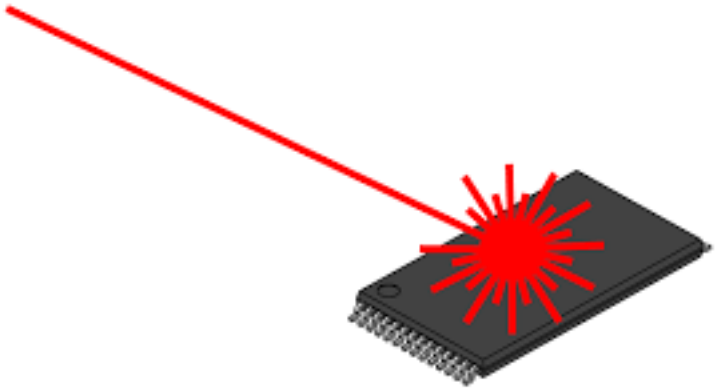
- Information Flow Security (IFS) Verification
- Power Side-channel Leakage Assessment (SCRIPT)
- Trojan Detection and Localization
- **Susceptibility to Fault Injection (AVFSM)**

Layout

- Susceptibility of Probing Attacks
- Power Side-channel Leakage Assessment (TVLA)

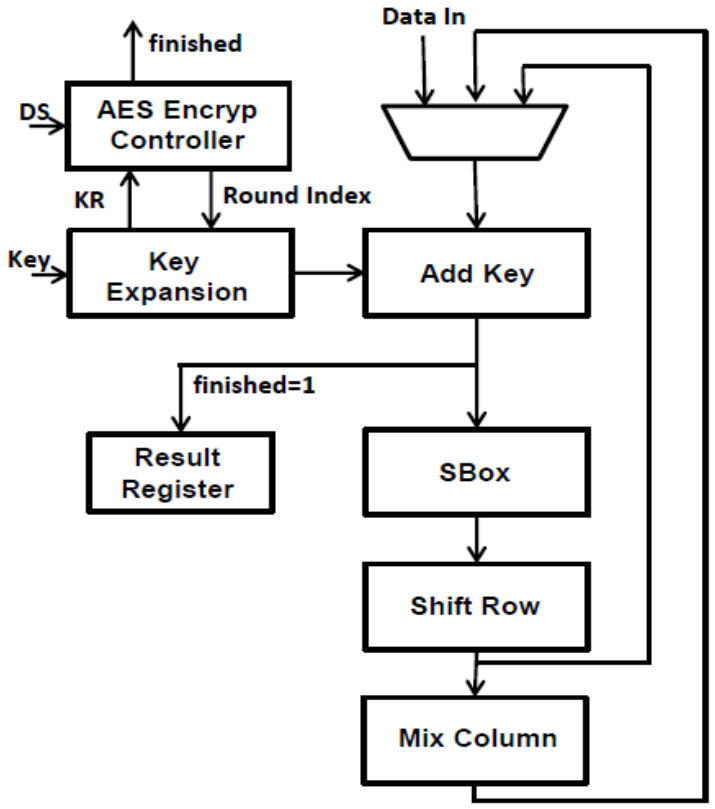
Vulnerability Analysis of FSM

- Finite State Machine → controls overall functionality of most digital systems
- **Attacks on FSM**
  - **Fault Injection Attack:** Inject a fault to cause transition to a protected state from an unauthorized state
  - **Trojan Attack:** Insert a Trojan to go to a protected state
- **Sources of Vulnerabilities**
  - Synthesis tools introduce **don't-care states and transitions** → facilitate fault and Trojan based attacks
  - **Encoding scheme** and **design constraints** → create unintentional vulnerabilities in FSM

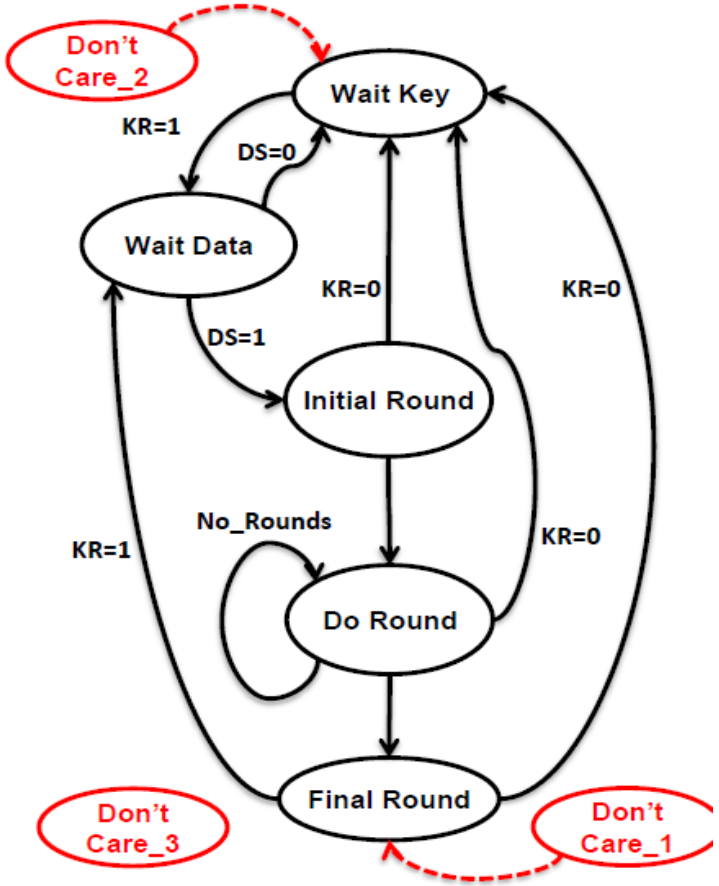


Example: AES Encryption

Attacker's objective:
Bypass the intermediate rounds and go directly to the Final Round.



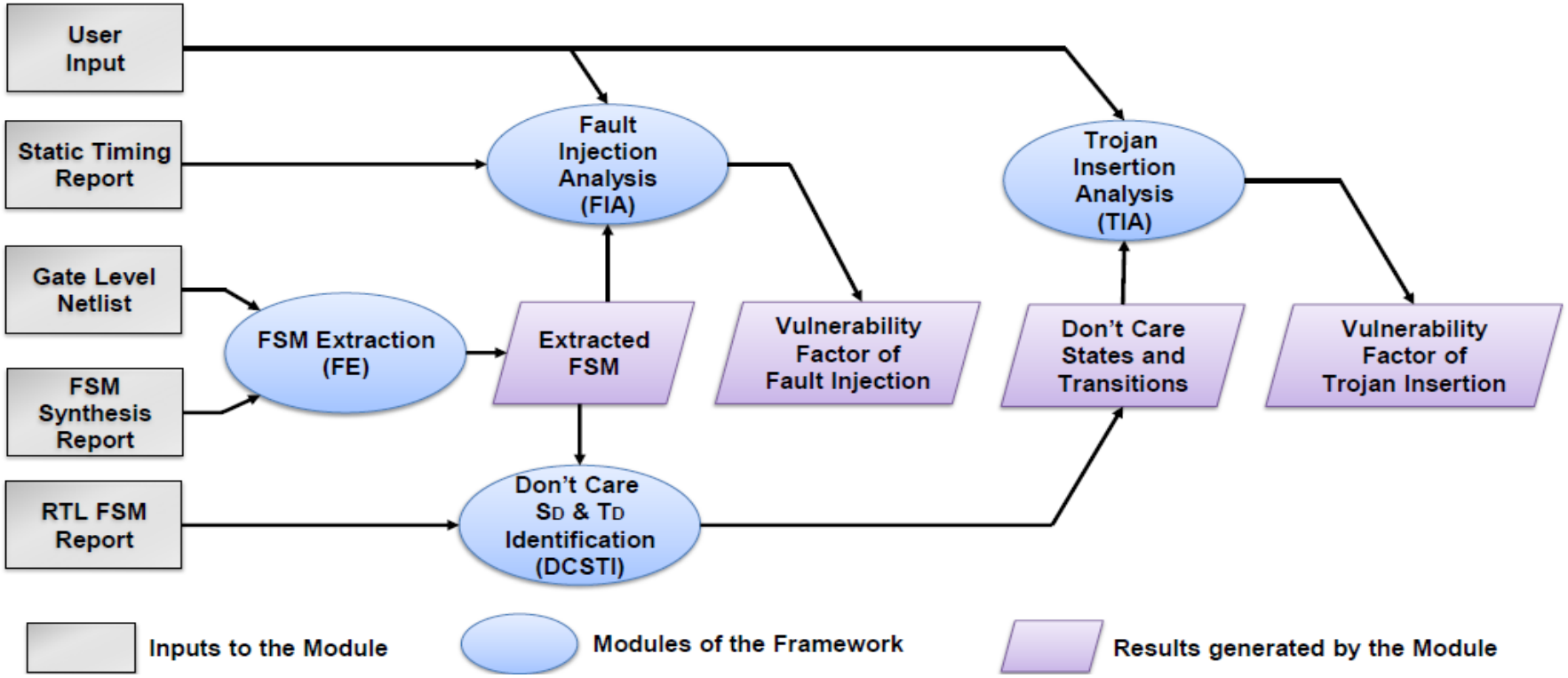
(a)



(b)

Source: Datasheeet AES 128/192/256 (ECB) AVALON

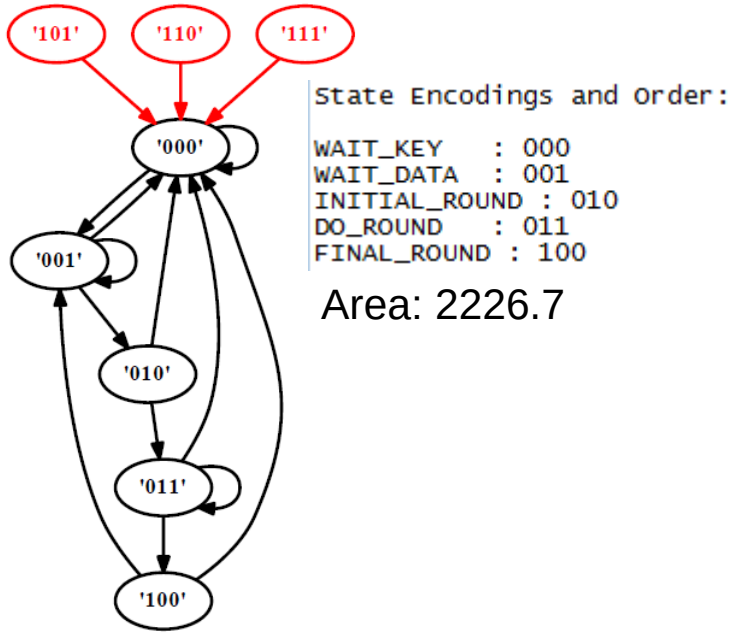
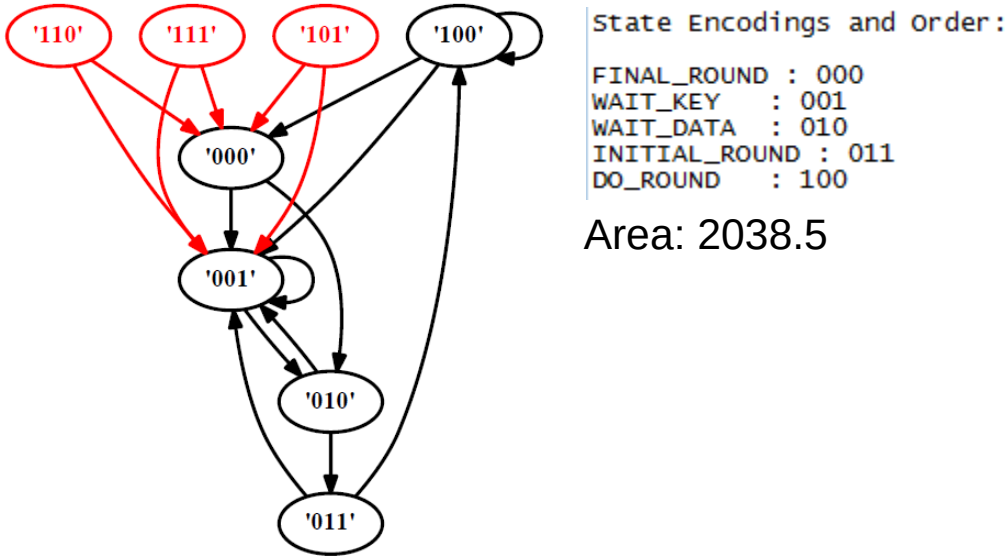
AVFSM Framework



Fault Injection Vulnerability Metric	Trojan Vulnerability Metric
$VF_{FI} = \{PVT(\%), ASF\}$ $PVT(\%) = \frac{Total_{Vulnerable\_Transition}(N_{VT})}{Total_{Transition}}$, $ASF = \frac{\sum_{i=1}^{N_{VT}} SF(i)}{N_{VT}}$	$VF_{Tro} = \frac{Total\ number\ of\ s'}{Total_{Transition}}$

Rule: For Secure FSM VF_{FI} and VF_{Tro} should be zero (or minimized)

Impact of Encoding Schemes

Encoding Scheme 1	Encoding Scheme 2
	

Area: 2226.7

Area: 2038.5

- **Takeaway**
  - **State encodings** impacts the vulnerabilities of a FSM

Vulnerability analysis of AES

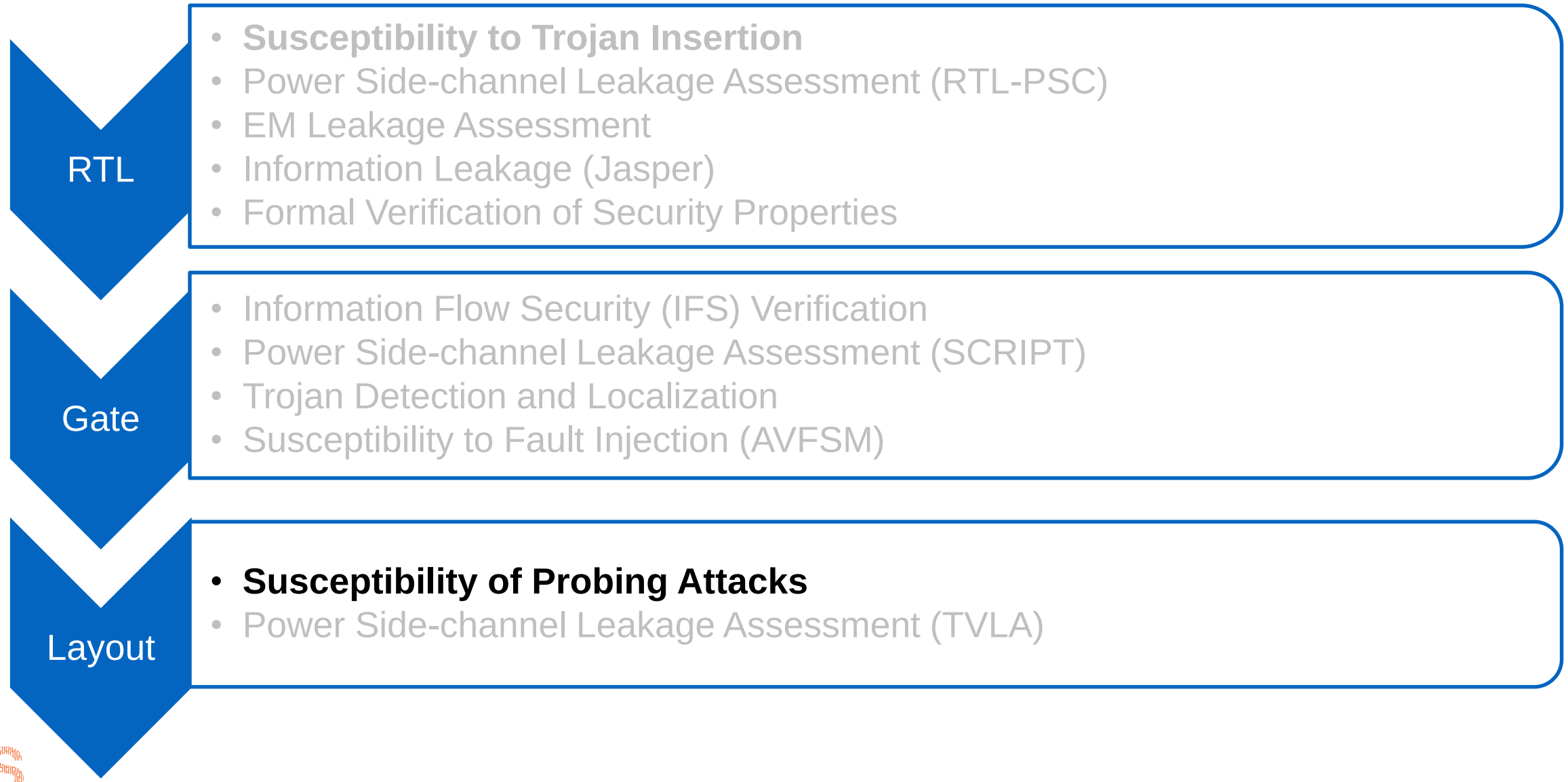
	scheme 1	scheme 2
VF_{FI}	(0,0)	(58.9%,0.15)
VF_{Tro}	0	0.18

Result: FSM Vulnerability Analysis

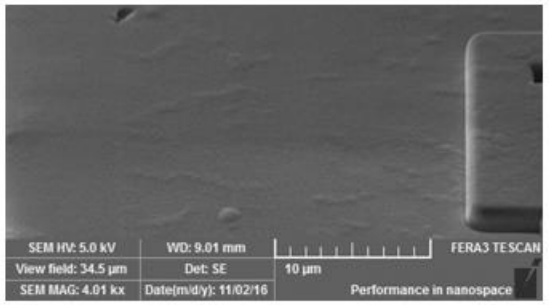
- We evaluated the security, cost and performance of traditional encoding schemes

	Encoding scheme	# states	# state FFs	# Don't cares	Area (μm^2)	Delay (ns)	Security concern	$VF_{FI} = ASF, PVT$
AES	Binary	5	3	3	3068	0.62	Yes	0.23, 0.38
	One-hot		5	27	4380	0.7	No	0, 0
SHA	Binary	7	3	1	4495	2.69	Yes	0.10, 0.35
	One-hot		7	121	6701	3.12	Yes	0.10, 0.07
MIPS	Binary	19	5	13	9346	1.6	Yes	0.42, 0.09
	One-hot		19	5.2e3	19816	1.52	Yes	0.26, 0.07
Mem.	Binary	66	7	62	60039	1.47	Yes	0.09, 0.01
	One-hot		66	7.3e19	68904	1.45	No	0, 0
RSA	Binary	7	3	3	3099	0.55	Yes	0.09, 0.12
	One-hot		7	121	5519	0.69	No	0, 0

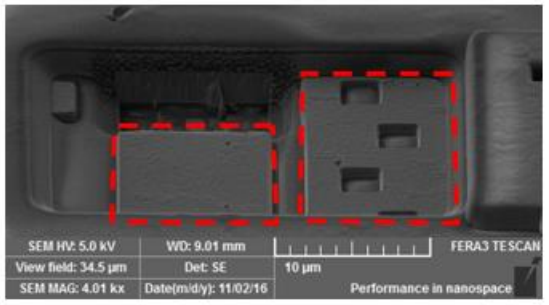
CAD for Security



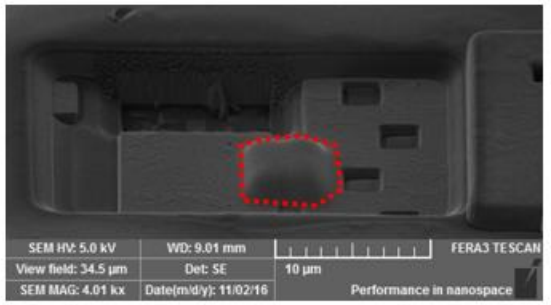
Susceptibility of Probing Attacks



Pre-FIB surface



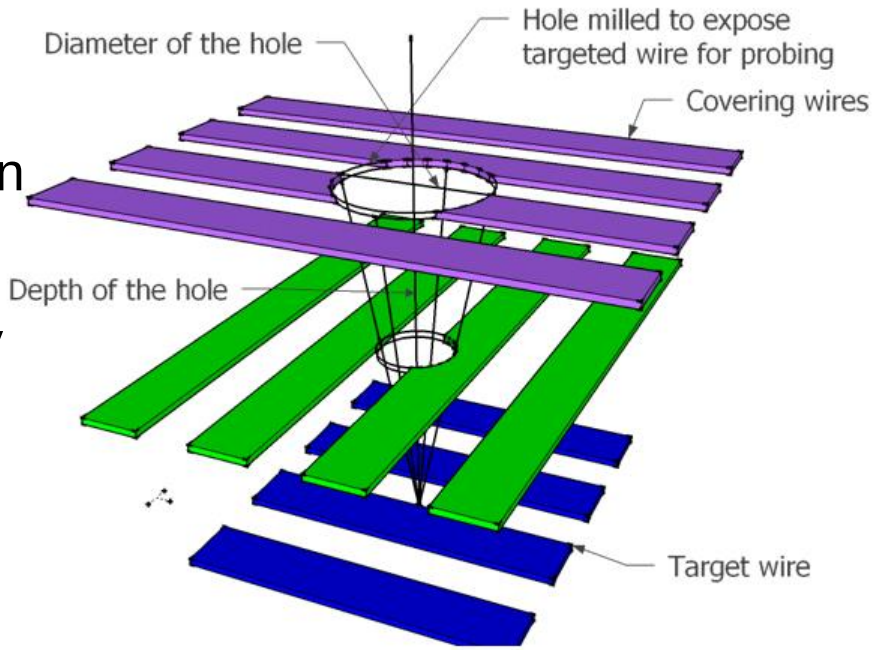
FIB milling to expose adjacent interconnects



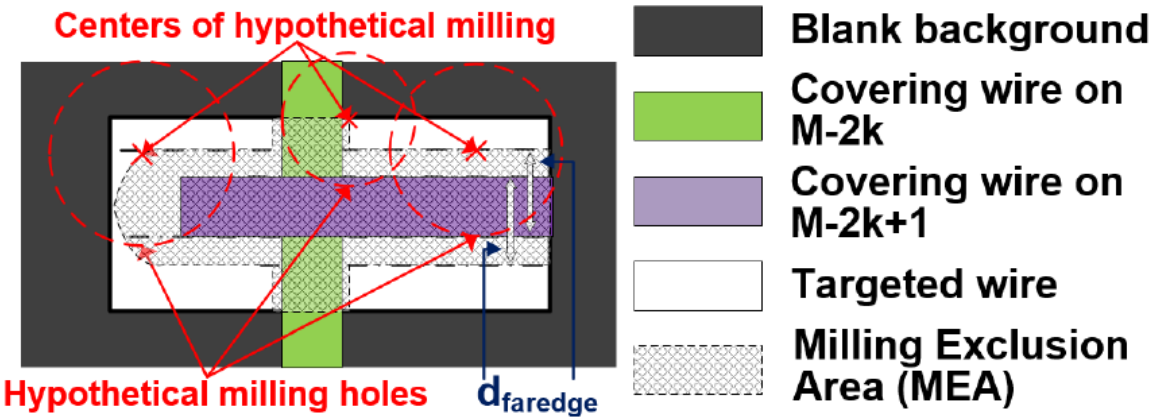
FIB deposition to short adjacent interconnects

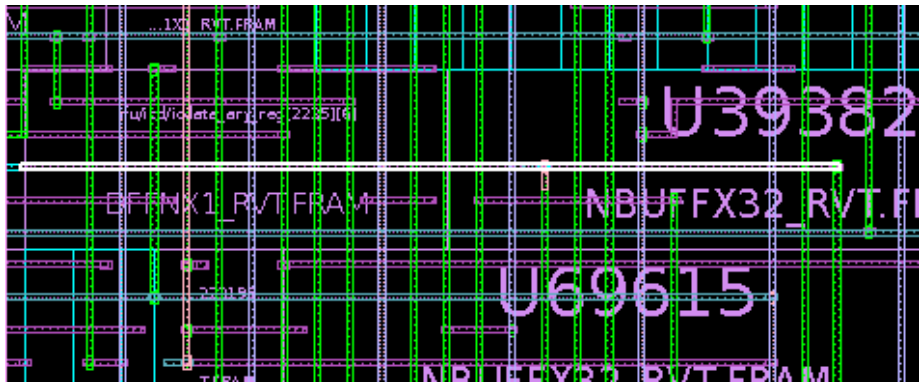
Source: FICS Research (Fera system)

- **Focused Ion Beam (FIB)**
  - A powerful tool commonly used in the development, manufacturing, and editing of ICs in nm level precision
- **Probing Attack**
  - Get physical access to signal wires to extract security critical information
  - Front-side attack and back-side attack
  - Trojan circuit.



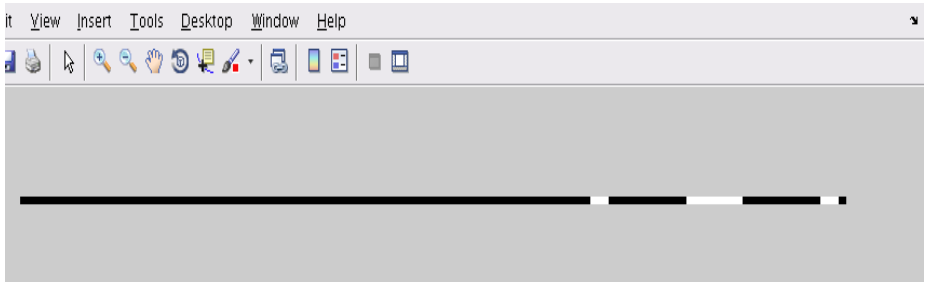
Probing Assessment: Exposed Area





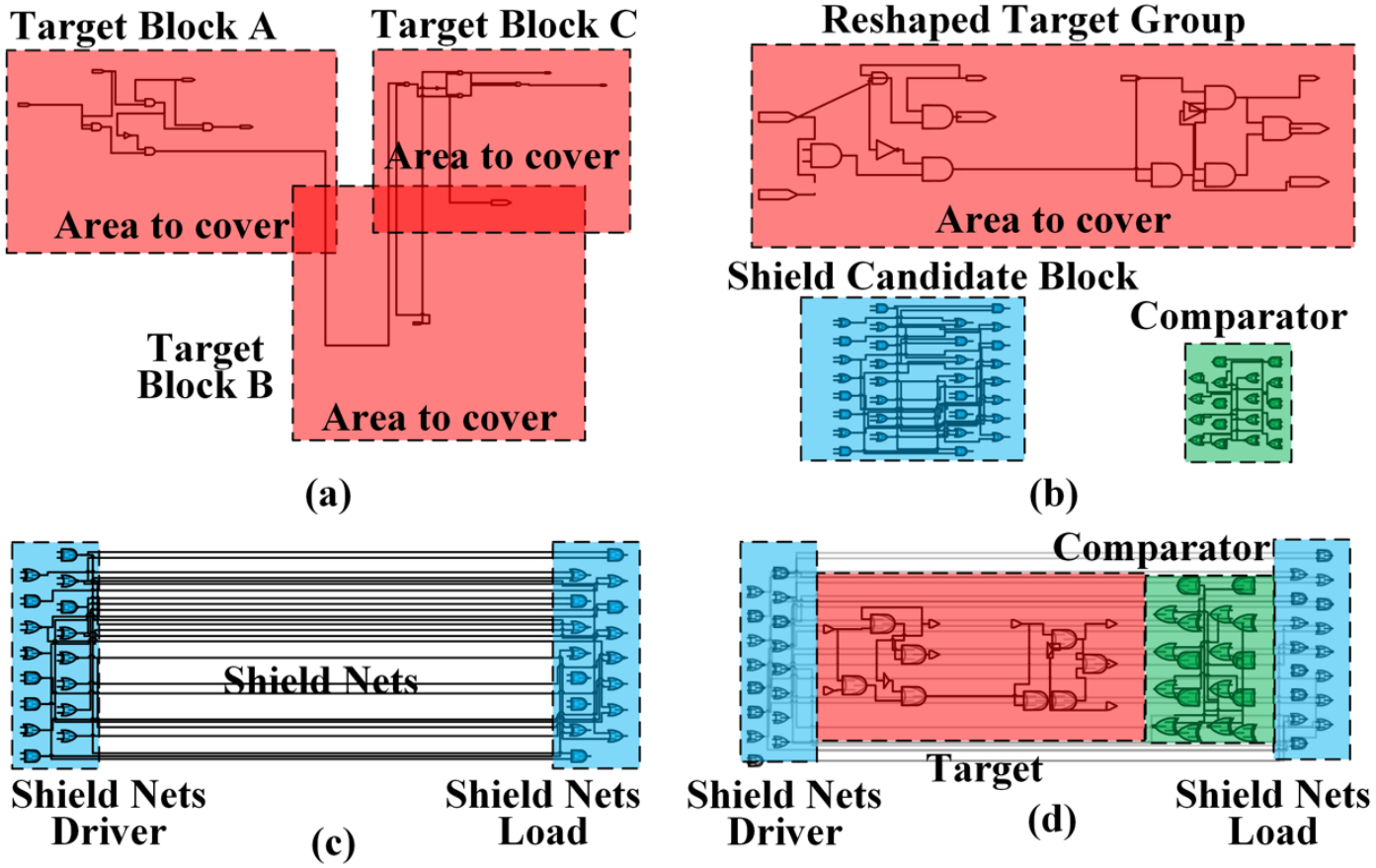
Layout view of targeted wire

- **Milling-exclusion Area (MEA)**
  - If milling center falls in MEA, an covering wire will be completely cut
- **Exposed Area (EA)**
  - Complement of MEA on target wires
  - Free to probe without impacting signal transmission
  - Designs with large exposed area are vulnerable to probing



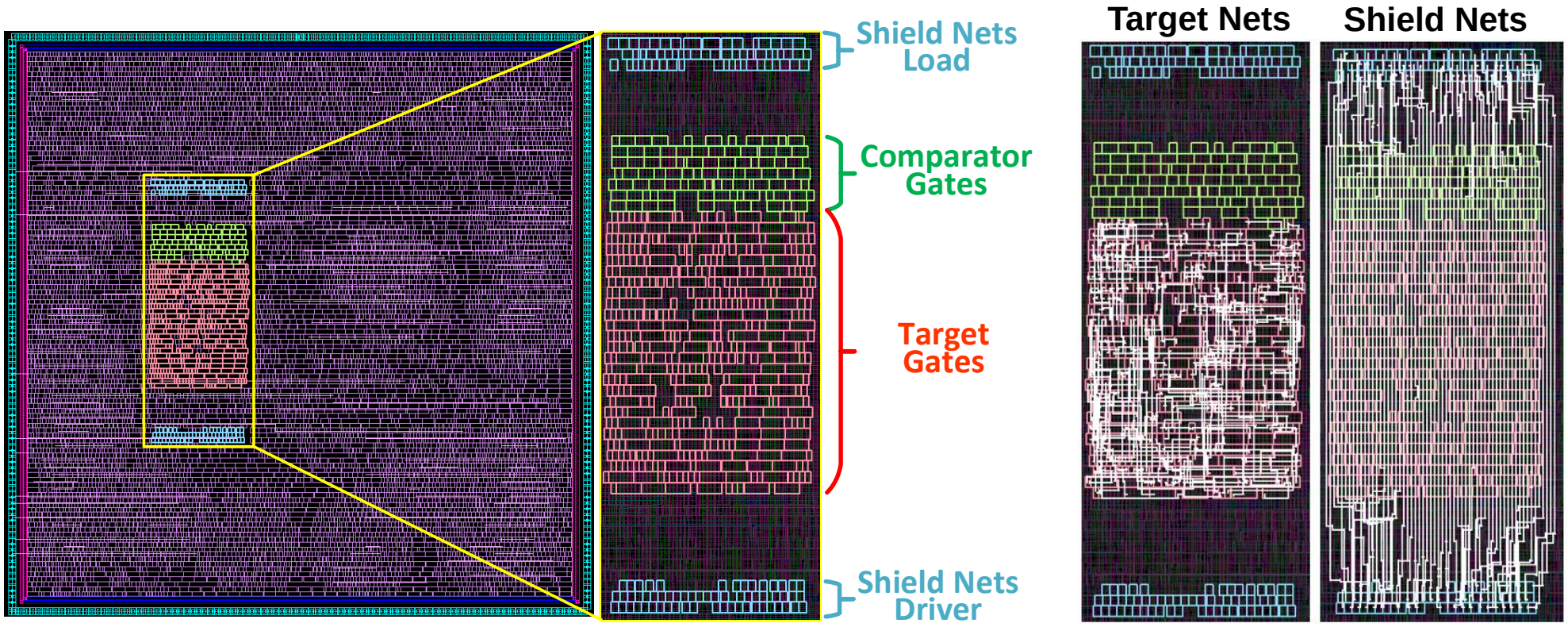
White: Exposed Area -- 11%
Black: Milling-exclusion Area

iPROBE: a CAD–based Internal Shielding Approach



- Automatically **identify** target and shield nets
- **Group** target nets under internal shield by **constrained** place and route
- **Compare** the shield signal and a lower copy to **detect** milling

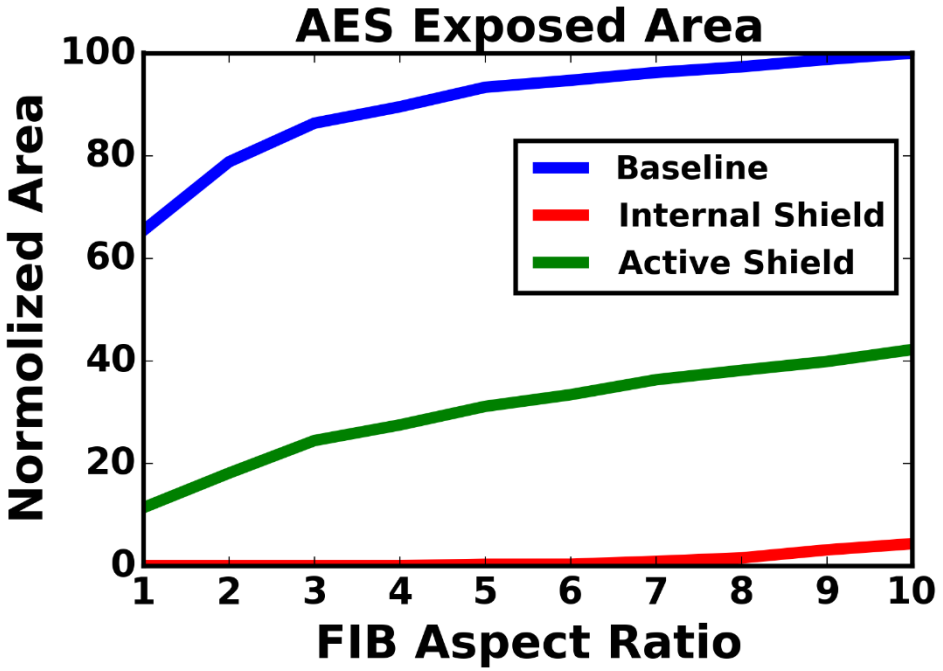
Evaluation on AES

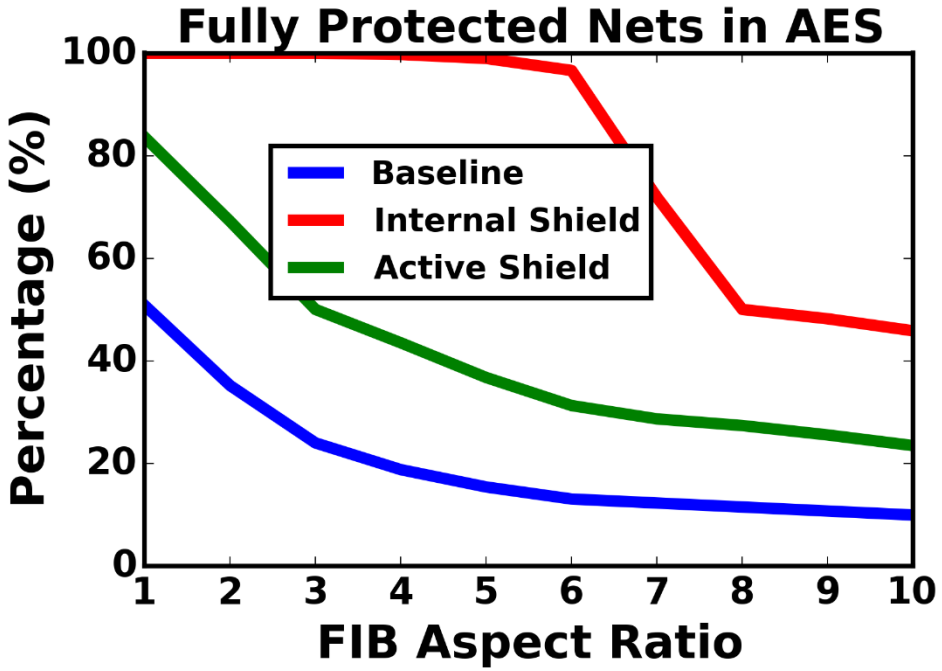


- Two level nets in the fanout of encryption keys are identified as target nets
- Target nets are routed on M1~M4, shield nets are routed on M6
- <1% area overhead, negligible in an SoC

Overhead	timing	power	area
AES	0.32%	2.79%	0.74%

Results: Exposed Area and Fully Protected Nets



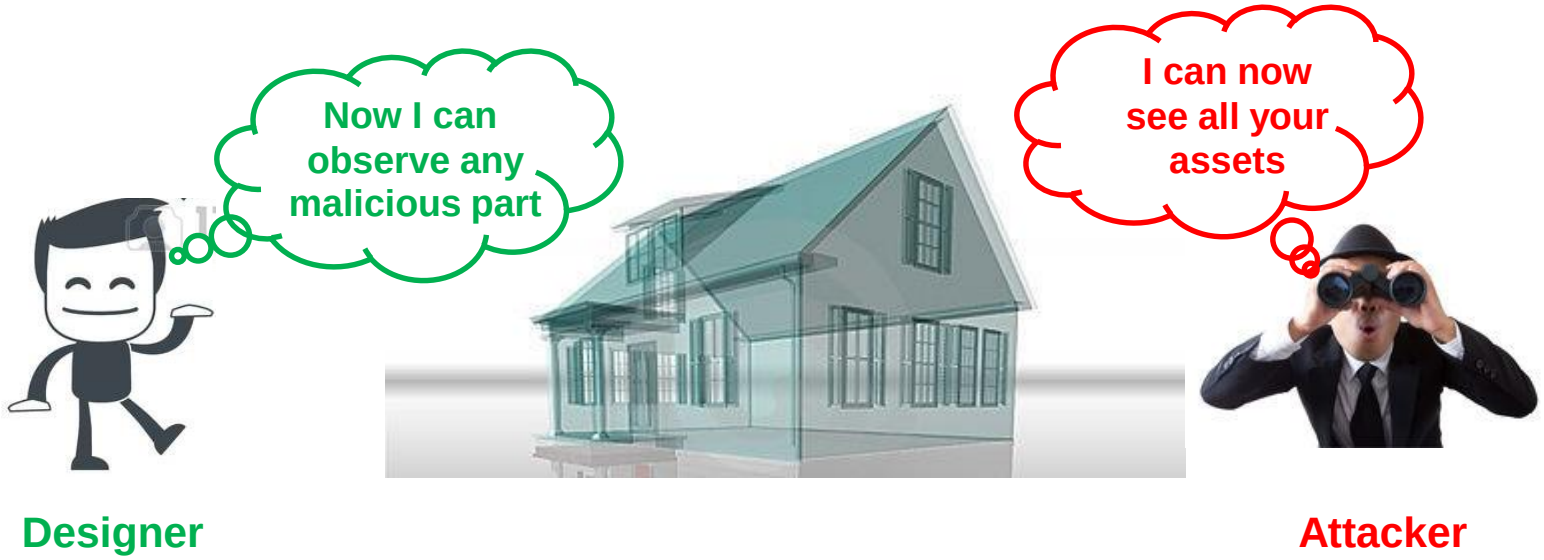


- Exposed Area (EA) could be reduced to 0, and all target nets are fully protected when RFIB< 5
- With advanced FIB (RFIB=10), EA can be reduced by ~95%, and ~50% target nets are fully protected
- Protection from a conventional active shield is inefficient

Challenges

Challenges

- Some rules can have **conflicting requirement**
  - For malicious change detection → high observability is **desired**
  - For asset leakage → high observability (of asset) is a **serious threat**



- **Risk-cost Analysis**: Invest in addressing threats that matters the most within the given budget/risk
  - Blindly applying rules → unnecessary **design overhead and loss of testability**

Challenges

- Need to develop comprehensive **SoC vulnerability database**
  - Effort underway by TAME working group
- **Formally expressing** security policies and rules
- Metrics
- Need to develop **standards** -- IEEE
- Automated security validation
  - Done at higher levels of abstraction, i.e., C/C++ or RTL
  - Evaluation times need to be **scalable** with the design size
  - Outputs generated should be easily **interpretable** by design engineer

Challenges

- **Usable Security**:
  - Development of **design guidelines** for security → avoid some common security problems
  - **Do-s** & **Don't-s** for designers
  - Best security practices
  - **Low-cost countermeasure** techniques for each vulnerability

References

[1] Salmani, Hassan, and Mohammed Tehranipoor. "Analyzing circuit vulnerability to hardware Trojan insertion at the behavioral level." *2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*. IEEE, 2013.

[2] He, Miao., Park, J., Nahiyan, A., Vassilev, A., Jin, Y., & Tehranipoor, M. (2019). RTL-PSC: Automated Power Side-Channel Leakage Assessment at Register-Transfer Level. *IEEE VLSI Test Symposium 2019*. 2019

[3] Y. Jin et al., EMLA: Metrics and Tools for Automated EM-Channel Leakage Analysis at Pre-Silicon, in preparation

[4] Jasper. (2014). JasperGold: Security Path Verification App. [Online].

[5] Contreras, Gustavo K., et al. "Security vulnerability analysis of design-for-test exploits for asset protection in SoCs." *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2017.

[6] Nahiyan, Adib, et al. "Hardware Trojan detection through information flow security verification." *2017 IEEE International Test Conference (ITC)*. IEEE, 2017.

[7] A. Nahiyan, F. Farahmandi, D. Forte, P. Mishra and M. Tehranipoor, \Security-aware FSM Design Flow for Mitigating Vulnerabilities to Fault Injection Attacks", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), submitted.

[8] Nahiyan, A., Xiao, K., Yang, K., Jin, Y., Forte, D., & Tehranipoor, M. (2016, June). AVFSM: a framework for identifying and mitigating vulnerabilities in FSMs. In *Proceedings of the 53rd Annual Design Automation Conference* (p. 89). ACM.

References

[9] H. Salmani, M. Tehranipoor, R. Karri, On design vulnerability analysis and trust benchmarks development, in: Computer Design (ICCD), 2013 IEEE 31st International Conference on, IEEE, pp. 471–474.

[10] Wang, Huanyu, et al. "Probing Assessment Framework and Evaluation of Antiprobing Solutions." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (2019).

[11] F. Farahmandi and P. Mishra. Automated test generation for debugging arithmetic circuits. In 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), pages 1351– 1356. IEEE, 2016.

[12] F. Farahmandi, Y. Huang, and P. Mishra. Trojan localization using symbolic algebra. In Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific, pages 591–597. IEEE, 2017.

[13] N. Fern, I. San, C. K. Koc¸, and K.-T. T. Cheng. Hardware trojans in incompletely specified on-chip bus systems. In Proceedings of the 2016 Conference on Design, Automation & Test in Europe, pages 527–530. EDA Consortium, 2016.

[14] X. Guo, R. G. Dutta, Y. Jin, F. Farahmandi, and P. Mishra. Pre-silicon security verification and validation: A formal perspective. In ACM/IEEE Design Automation Conference (DAC), 2015.

[15] J. Rajendran, V. Vedula and R. Karri. Detecting malicious modifications of data in third party intellectual property cores. In ACM/IEEE Design Automation Conference (DAC), pages 112–118, 2015.

[16] Jonathan Cruz, Farimah Farahmandi, Alif Ahmed, and Prabhat Mishra, “Hardware Trojan Detection using ATPG and Model Checking,” International Conference on VLSI Design (VLSI Design), pages 91-96, Pune, India, January 6 – 10, 2018.

See Trust-Hub to access benchmarks, tools, hardware platforms, etc.
www.trust-hub.org

SoC Security
http://trust-hub.org/vulnerability-db/cad-solutions

Mark Tehranipoor, tehranipoor@ece.ufl.edu
Farimah Farahmandi, farimah@ece.ufl.edu