

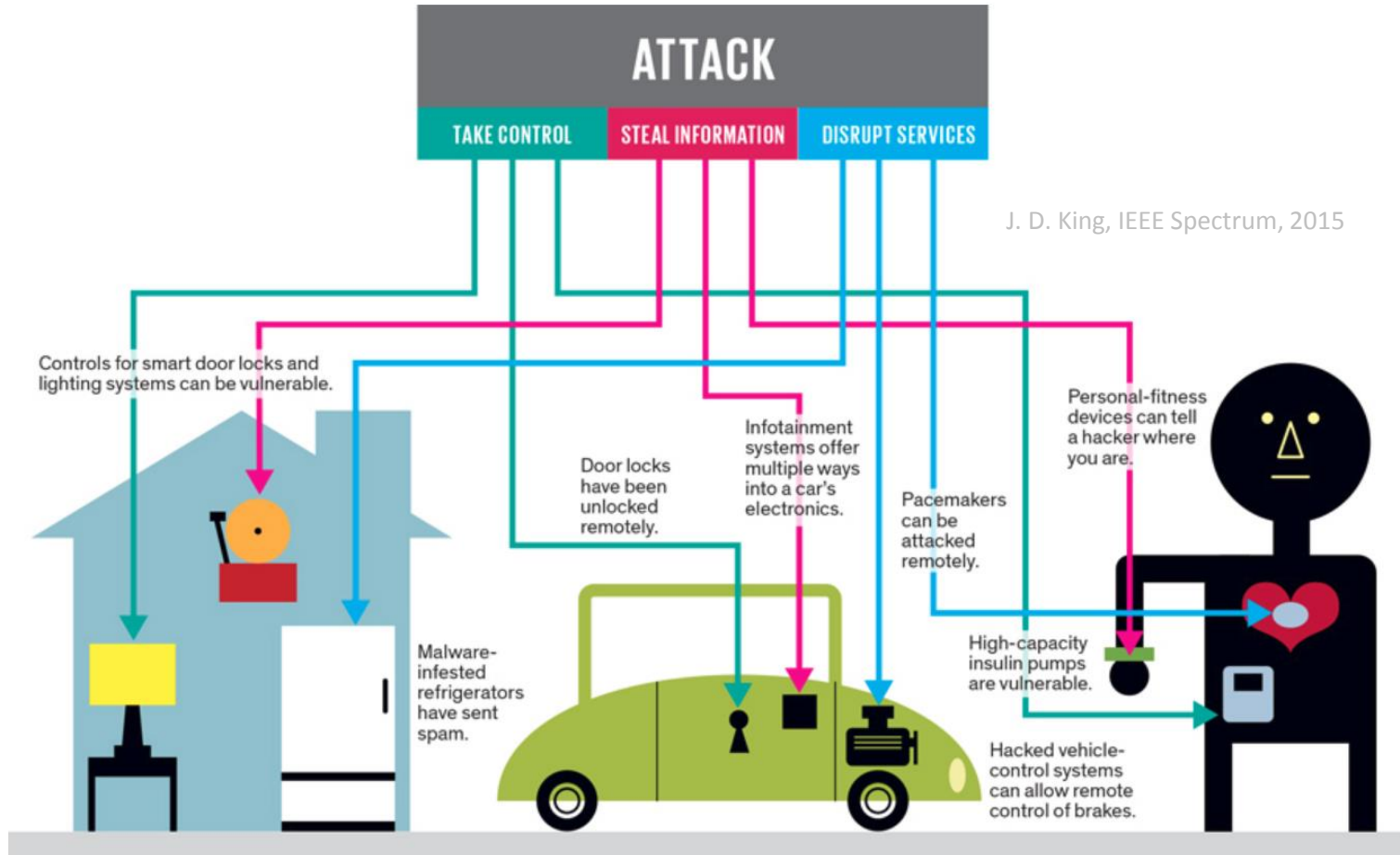


SCHLOSS DAGSTUHL  
Leibniz-Zentrum für Informatik

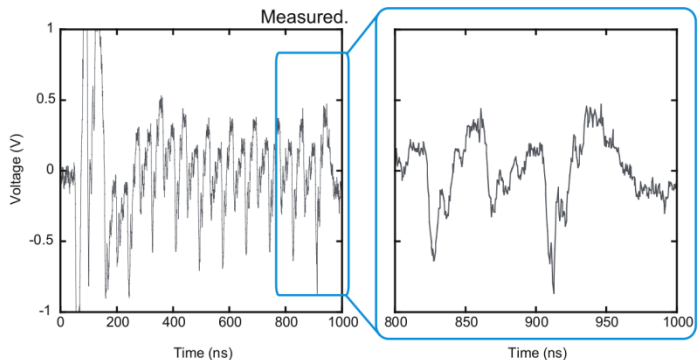
# Towards Secure Composition of Integrated Circuits and Electronic Systems: On the Role of EDA

Johann Knechtel, Elif Bilge Kavun, Francesco Regazzoni, Annelie Heuser, Anupam Chattopadhyay,  
Debdeep Mukhopadhyay, Soumyajit Dey, Yunsi Fei, Yaacov Belenky, Itamar Levi, Tim Güneysu,  
Patrick Schaumont, and Ilia Polian

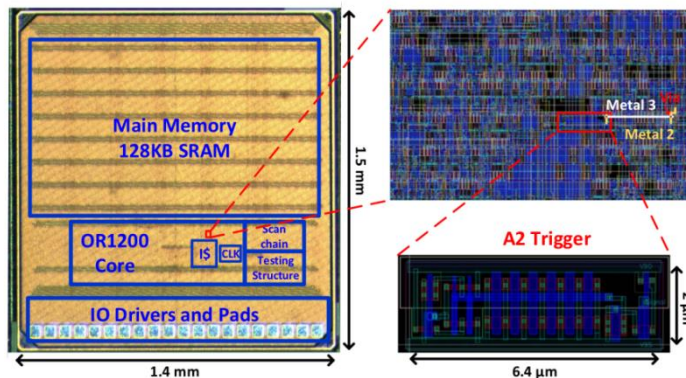
DATE 2020 – Originated at Dagstuhl Seminar 19301,  
“Secure Composition for Hardware Systems,” July 21–26, 2019



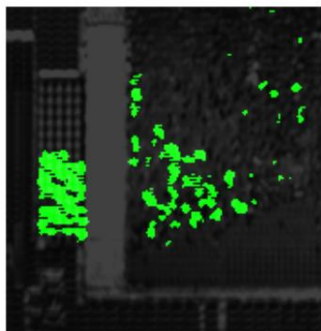
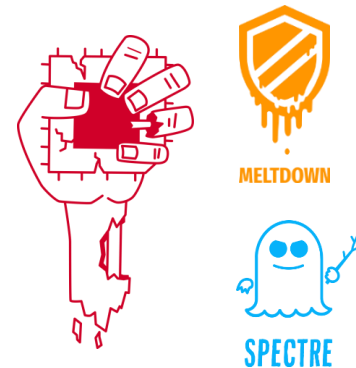
# Security Threats – Affecting Data and Circuits Itself



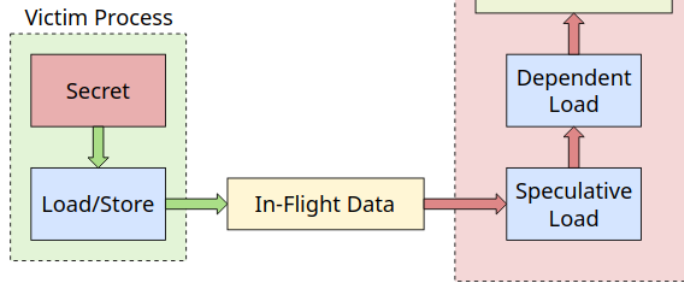
Fujimoto et al., EMC 2014



Yang et al., SP 2016



Tajik et al., CCS, 2017



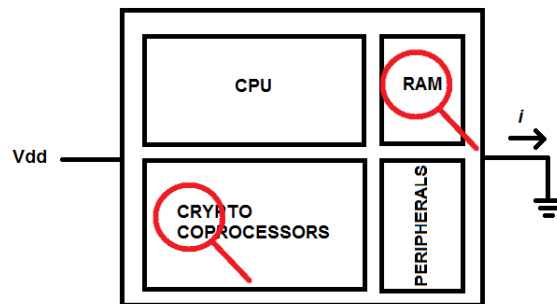
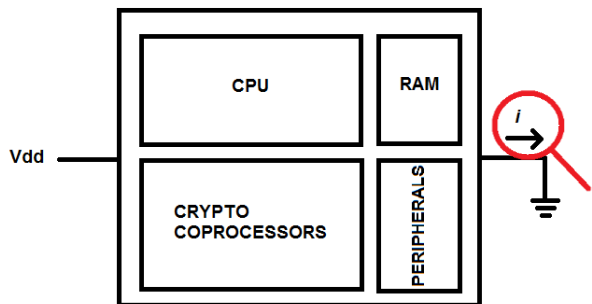
<https://www.bleepingcomputer.com>, 2019

# Motivation and Scope

- Electronic design automation (EDA) focused traditionally on power, performance, area (PPA)
- Due the rise of hardware-centric security threats, we argue that EDA must also adopt security notions
  - Secure by design
  - Secure composition of hardware
- Objective and scope for today:
  - Introduction to hardware security for the EDA community
  - Discussion of security-centric EDA stages for evaluation, implementation
  - Challenges and strategies toward secure composition of circuits and systems

## Side-Channel Attacks

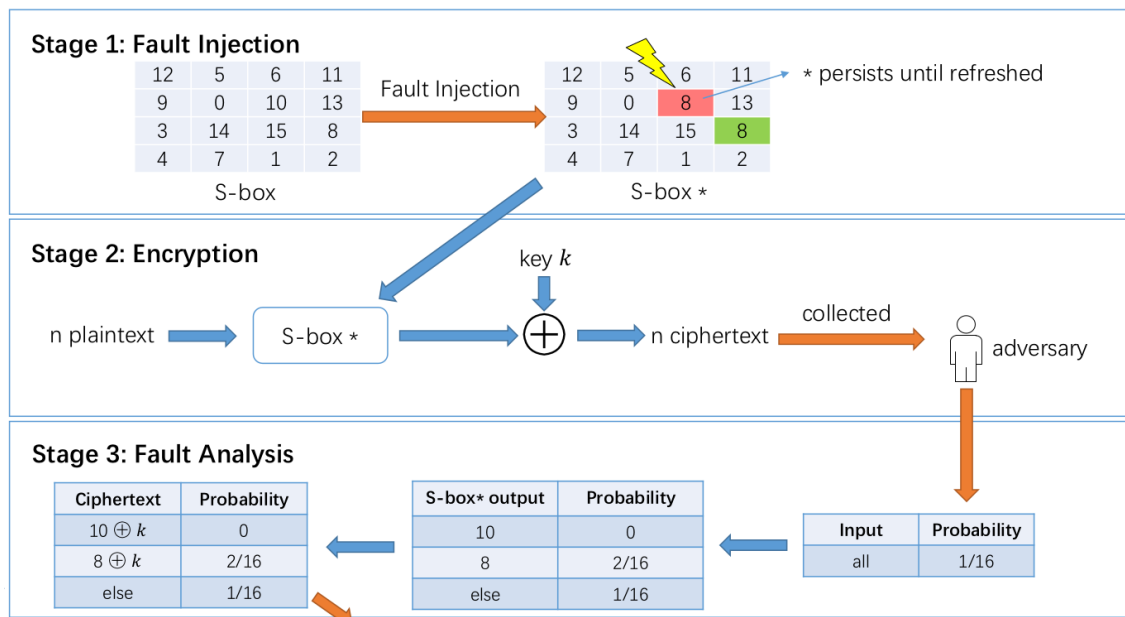
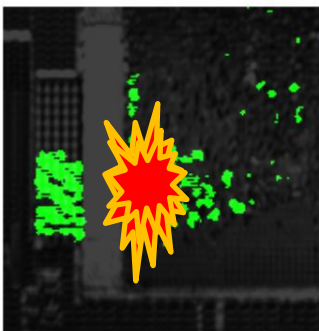
- Side channels: power consumption, timing behavior, electromagnetic emission
  - Information leakage due to physical reality
  - Statistical analysis on collected samples; various well-established and effective types of attacks



- Countermeasures: masking, i.e., diffusion of information leakage

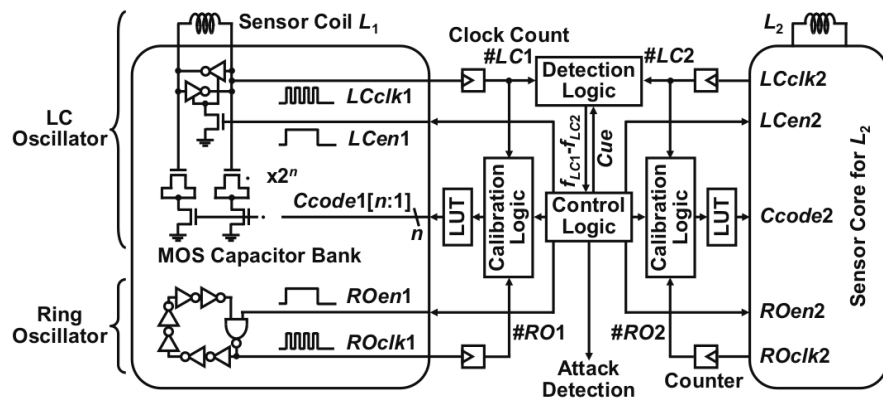
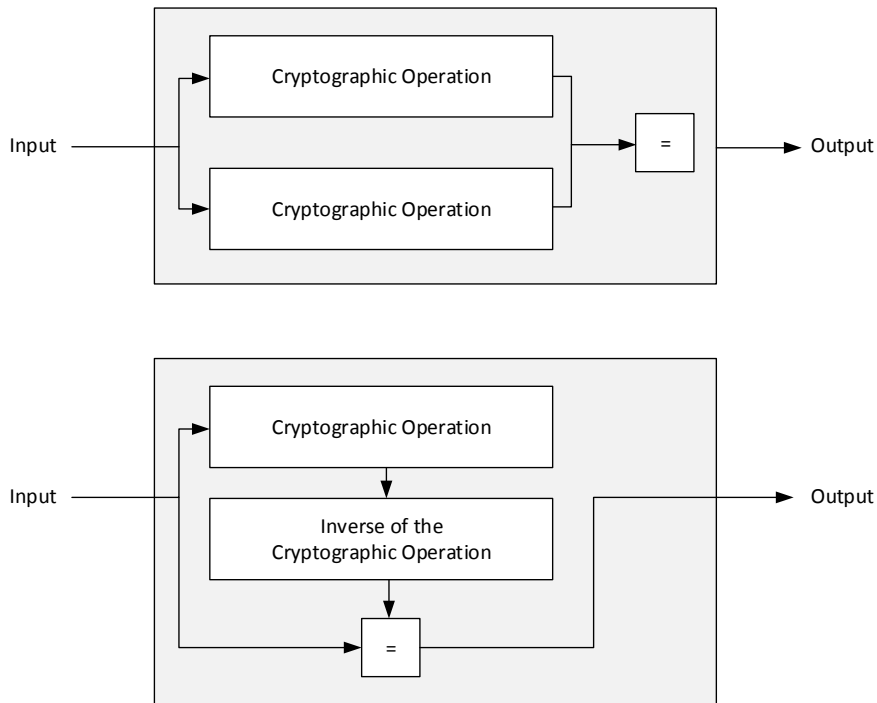
# Fault-Injection Attacks

- Fault injection to deduce sensitive information or interrupt circuit features
  - Direct, invasive fault injection, e.g., by laser light or electromagnetic waves
  - Indirect, architectural fault injection, e.g., by repetitive writing to particular memory locations



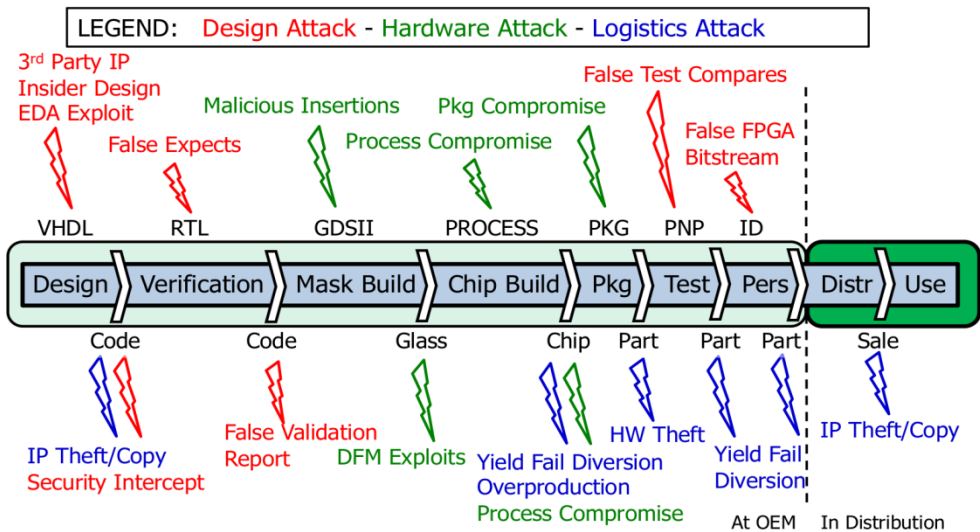
# Fault-Injection Attacks

- Countermeasures: mitigation, detection



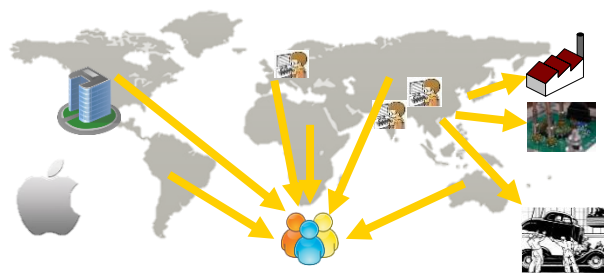
Homma et al., CHES, 2014



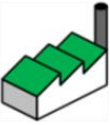


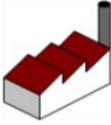


# Piracy of Chip IP, Counterfeiting of ICs



Kerry Bernstein, DARPA, 2016

- Countermeasures: IP protection schemes, physically-unclonable functions (PUFs)

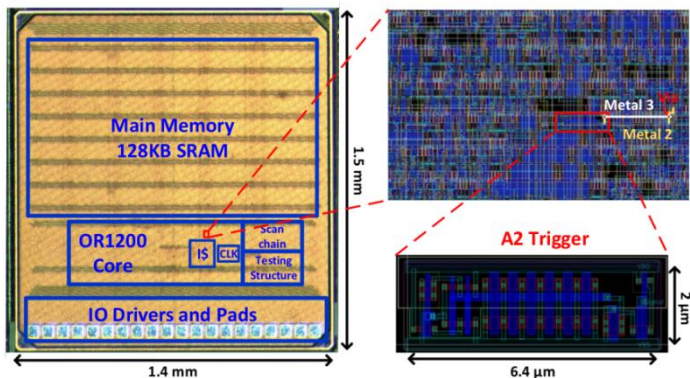


|  |  |   |
|--|--|---|
|  | <br>Trusted user        | <br>Untrusted user   |
| <br>Trusted foundry   | <br>Trusted user        | <br>Camouflaging     |
| <br>Untrusted foundry | <br>Split manufacturing | <br>Logic Encryption |

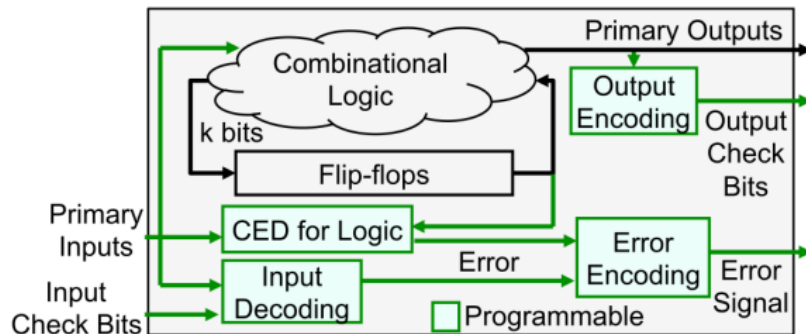


# Hardware Trojans

- Trojans are malicious modifications that are
  - Targeted at the system level, RTL, gate level, or transistor/physical level;
  - Introduced by untrustworthy 3<sup>rd</sup> party IP, adversarial designers, “hacking” of design tools, during packaging of ICs, or (less likely) during manufacturing;
  - Seeking to leak information, reduce the performance, or disrupt the IC;
  - Always on, triggered internally, or triggered externally; etc.
- Countermeasures: detection (pre- and post-silicon), mitigation



Yang et al., SP 2016

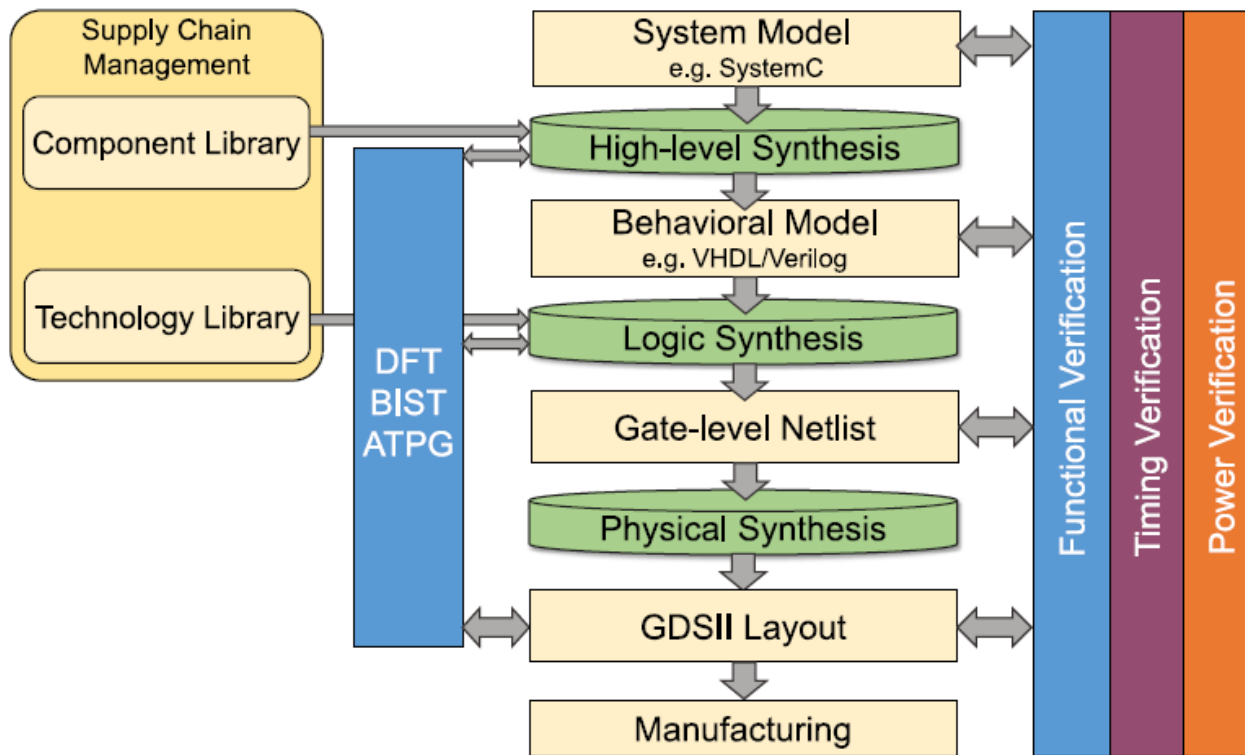


Wu et al., TCAD 2016

## Selected Security Threats and Roles of EDA

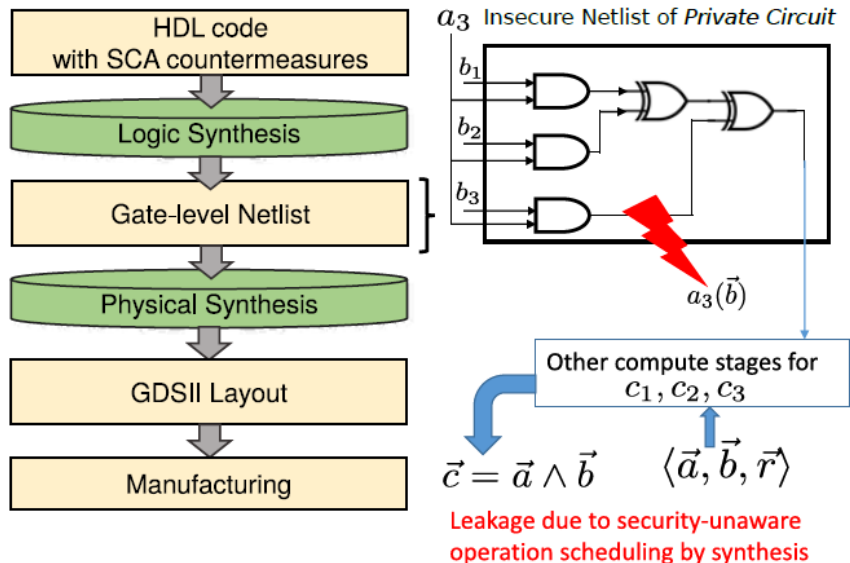
| Threat Vector   | Time of Attack   | Role of EDA   |
|---|--|---|
| <ul style="list-style-type: none"><li>Side-channel attacks</li></ul>  | <ul style="list-style-type: none"><li>Runtime</li></ul>                            | <ul style="list-style-type: none"><li>Evaluation</li><li>Mitigation at design time</li></ul>  |
| <ul style="list-style-type: none"><li>Fault-injection attacks</li></ul>   | <ul style="list-style-type: none"><li>Runtime</li></ul>                            | <ul style="list-style-type: none"><li>Evaluation</li><li>Mitigation at design time</li></ul>  |
| <ul style="list-style-type: none"><li>Piracy of design intellectual property (IP)</li><li>Counterfeiting of ICs</li></ul> | <ul style="list-style-type: none"><li>Manufacturing</li><li>In the field</li></ul> | <ul style="list-style-type: none"><li>Mitigation at design time</li></ul>   |
| <ul style="list-style-type: none"><li>Hardware Trojans</li></ul>  | <ul style="list-style-type: none"><li>Design</li><li>Manufacturing</li></ul>       | <ul style="list-style-type: none"><li>Mitigation at design time</li><li>Verification at design time</li><li>Preparing for testing, inspection</li></ul> |

# Generic EDA Flow



# Generic EDA Flow: An Example for Security Fallacies

- Private circuits
  - Information can be encoded as vector, e.g., for bit  $a$  as  $(a_1, a_2, a_3)$
  - Separate computation of shares, incorporate random bits  $r_{i,j}$
  - E.g.,  $c = a \wedge b$  is computed as:
    - $c_1 = a_1b_1 \oplus r_{1,2} \oplus r_{1,3}$
    - $c_2 = a_2b_2 \oplus (r_{1,2} \oplus a_1b_2) \oplus a_2b_1 \oplus r_{2,3}$
    - $c_3 = a_3b_3 \oplus (r_{1,3} \oplus a_1b_3) \oplus a_3b_1 \oplus (r_{2,3} \oplus a_2b_3) \oplus a_3b_2$
  - Synthesis could compile  $c_3$  such that  $a_3b_1 \oplus a_3b_2 \oplus a_3b_3 = a_3(b)$  is derived first and random bits  $r_{i,j}$  are added later (as  $\oplus$  is commutative)
  - Then, circuit will leak the value of  $b$

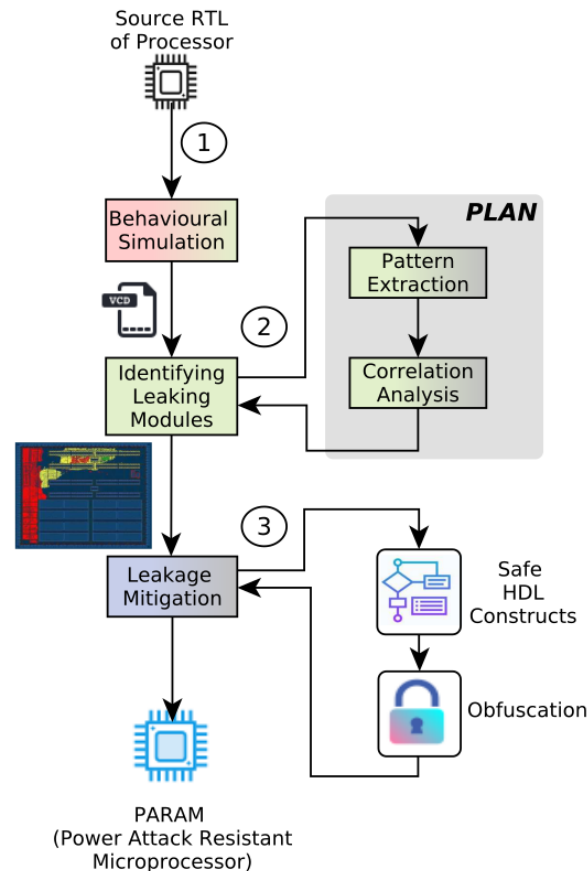


# Selection of Security Schemes Promising for Integration with EDA Tools

| Design Stage                         | Threat Vectors   |  |   |  |
|--------------------------------------|--|--|---|--|
|                                      | Side-Channel Attacks   | Fault-Injection Attacks  | IP Piracy and Counterfeiting  | Trojans  |
| High-Level Synthesis                 | <ul style="list-style-type: none"> <li>Information-flow tracking</li> <li>Integration of masking</li> <li>Register flushing</li> </ul> | <ul style="list-style-type: none"> <li>Error-detecting architectures</li> <li>Infective countermeasures</li> </ul> | <ul style="list-style-type: none"> <li>Metering IP (including PUFs)</li> </ul>                                | <ul style="list-style-type: none"> <li>Self-authentication</li> </ul>                      |
| Logic Synthesis                      | <ul style="list-style-type: none"> <li>Gate-level protections</li> <li>Identification of leaking gates</li> </ul>                      | <ul style="list-style-type: none"> <li>Automatic fault analysis</li> </ul>   | <ul style="list-style-type: none"> <li>Camouflaging</li> <li>Logic locking</li> </ul>                         | <ul style="list-style-type: none"> <li>Automatic insertion of security monitors</li> </ul> |
| Physical Synthesis (Place and Route) | <ul style="list-style-type: none"> <li>Information leakage analysis (TVLA, etc.)</li> </ul>  | <ul style="list-style-type: none"> <li>Embedding sensors</li> <li>Shielding</li> </ul>                             | <ul style="list-style-type: none"> <li>Split manufacturing</li> <li>Entropy primitives</li> </ul>             | <ul style="list-style-type: none"> <li>Embedding sensors</li> </ul>                        |
| Functional Validation                | <ul style="list-style-type: none"> <li>Identification of architectural covert channels</li> </ul>                                      | <ul style="list-style-type: none"> <li>Validation of error-detection properties</li> </ul>                         | <ul style="list-style-type: none"> <li>Correctness of locked logic</li> <li>De-obfuscation attacks</li> </ul> | <ul style="list-style-type: none"> <li>Proof-carrying hardware</li> </ul>                  |
| Timing and Power Verification        | <ul style="list-style-type: none"> <li>Pre-silicon power/timing simulation</li> </ul>  | <ul style="list-style-type: none"> <li>Detailed modelling of fault injections</li> </ul>                           | <ul style="list-style-type: none"> <li>Validation of low-level properties of PUFs</li> </ul>                  | <ul style="list-style-type: none"> <li>Fingerprinting</li> </ul>                           |
| Testing (ATPG, DFT, BIST)            | <ul style="list-style-type: none"> <li>Securing DFT against read-out (scan-chain attacks, etc.)</li> </ul>                             | <ul style="list-style-type: none"> <li>DFX architecture to handle malicious/natural failures</li> </ul>            | <ul style="list-style-type: none"> <li>IP protection integrated into DFX infrastructure</li> </ul>            | <ul style="list-style-type: none"> <li>Pattern generation for Trojan detection</li> </ul>  |

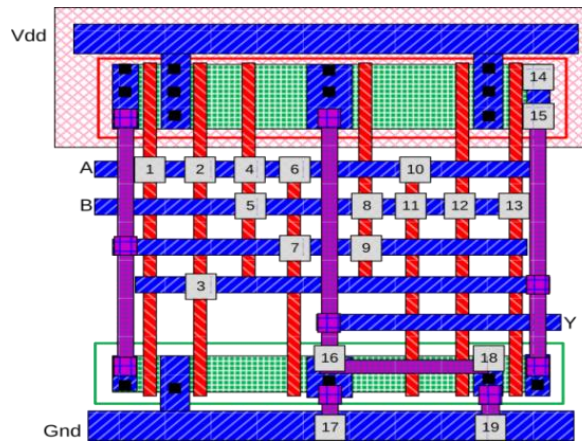
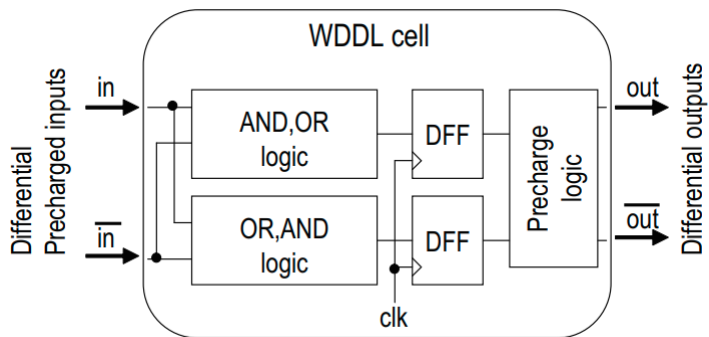
# Security-Driven High-Level Synthesis

- Side-channel attack countermeasures
  - Randomly flush/overwrite registers after use
  - Information flow tracking via dedicated HDL like Caisson, SecVerilog, QIF-Verilog
  - Works on automated synthesis of masking for generic software exist, but for EDA still WIP
- Evaluation, countermeasure implementation typically focused on later stages



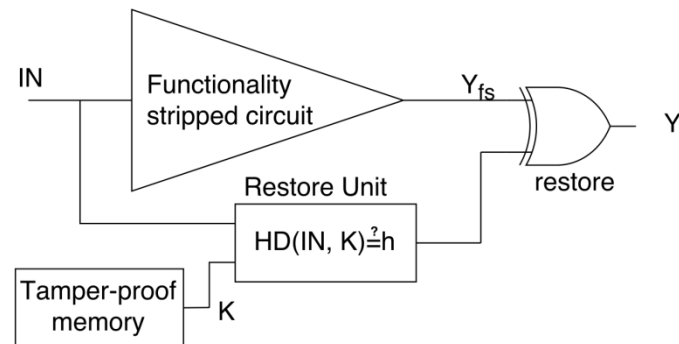
# Security-Driven Logic Synthesis

- In general, instantiate security primitives or circuitry
- Side-channel attack countermeasures
  - E.g., wave dynamic differential logic (WDDL) paradigm
- IP protection: Camouflaging
  - Multi-functional, obfuscated primitives; well supported by synthesis



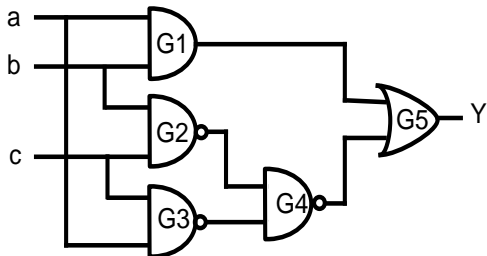
# Security-Driven Logic Synthesis

- IP protection: Logic locking
  - Similar to example of private circuits, synthesis is problematic (yet essential)
  - Transformations via re-synthesis to hide key value; transformations can be machine-learned
  - Structural traces for locking structures may remain; can be re-traced

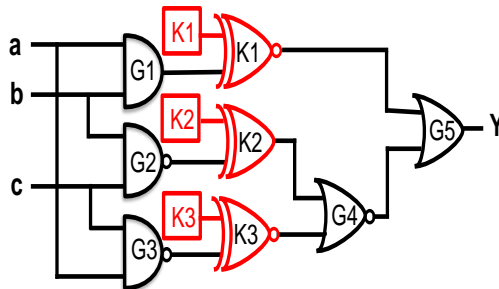


Yang et al., 2019, TIFS

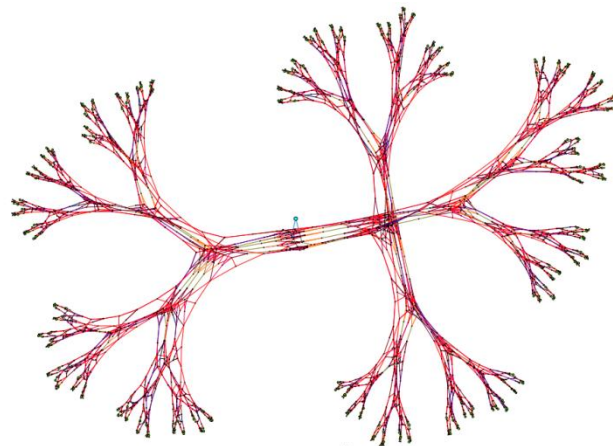
**Original circuit**



**Locked circuit**



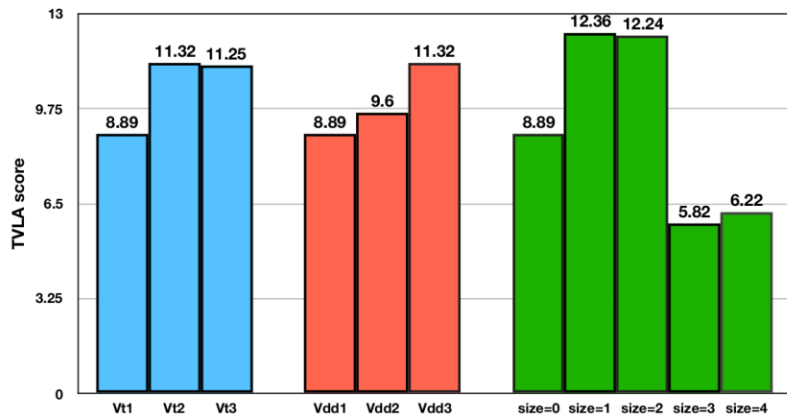
**Correct key: 110**



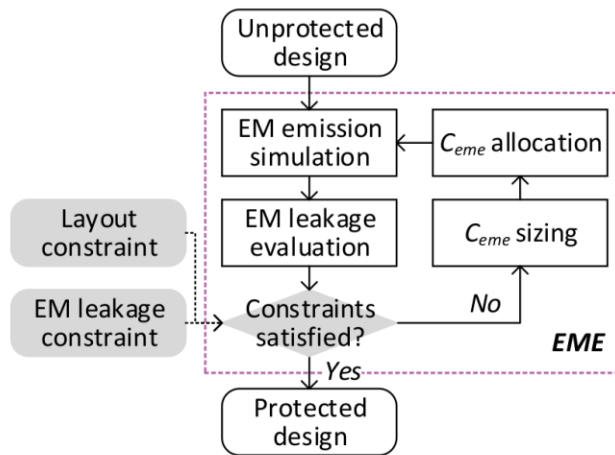


# Security-Driven Physical Synthesis

- Side-channel countermeasure: Re-design physical layout based on test vector leakage assessment (TVLA) or other evaluation schemes
  - Also requires power, timing verification stages
  - Modeling assumptions versus attacker's capabilities (e.g., noise distribution)
- Similarly, works for fault-injection countermeasures



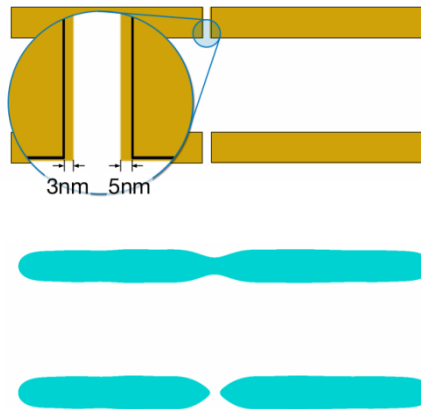
SLPSK et al., ICCAD, 2019



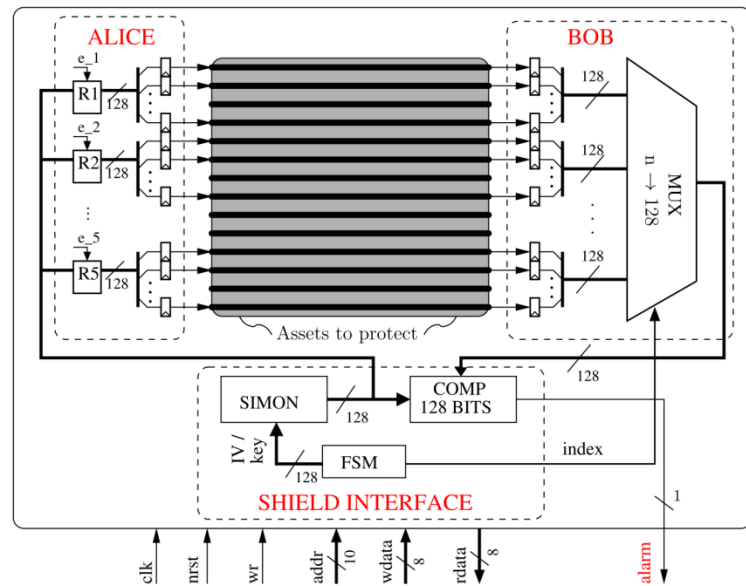
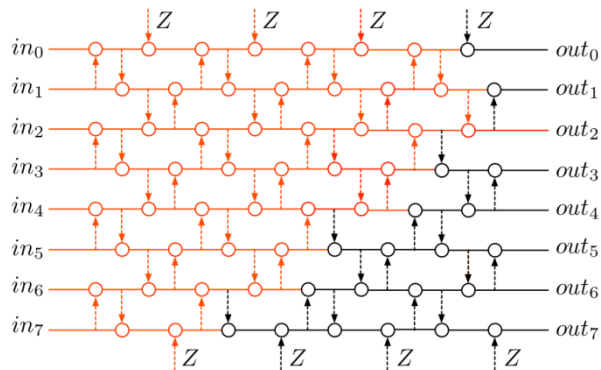
Wang et al., ICCAD, 2018

# Security-Driven Physical Synthesis

- Employ security primitives and account for their physical aspects
  - PUFs, RNGs, shields, sensors, etc.
  - Entropy essential for PUFs and RNGs which comes from physical circuit structures; synthesis is essential for proper implementation



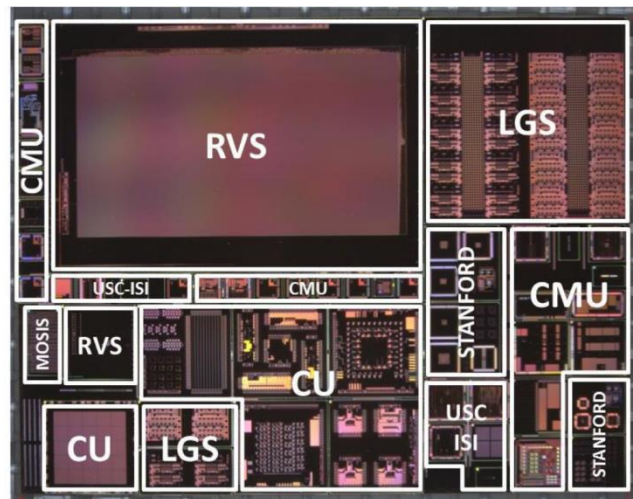
Miao et al., TCAD, 2017



Ngo et al., TC 2017

# Security-Driven Physical Synthesis

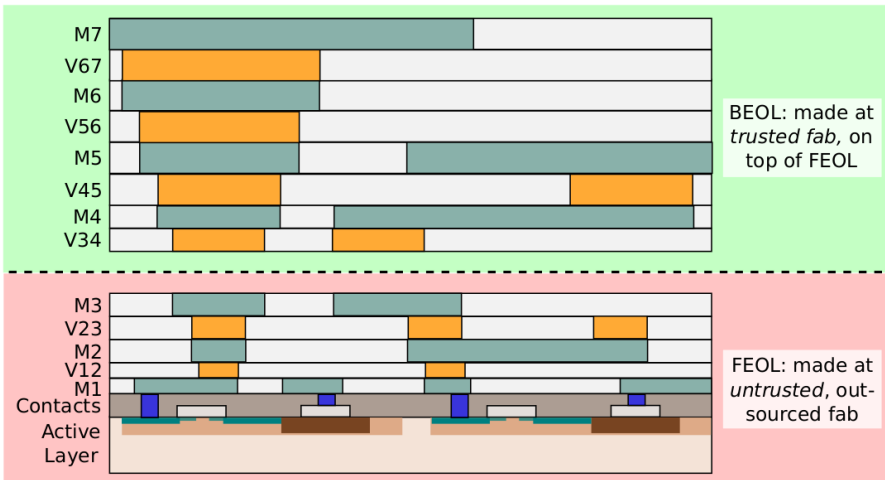
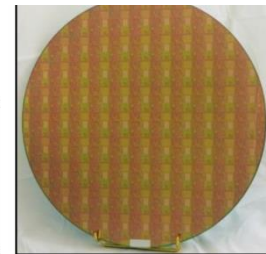
- IP protection: Split manufacturing
  - Benefit from latest technology, without giving away design IP
  - Practical; has been demonstrated in 2D ICs for 28nm and older nodes, promising also for 3D ICs



IARPA multi-user test chip December 2015 fabricated jointly between Samsung (Korea) and Samsung (Austin).

McCants, IARPA, 2016

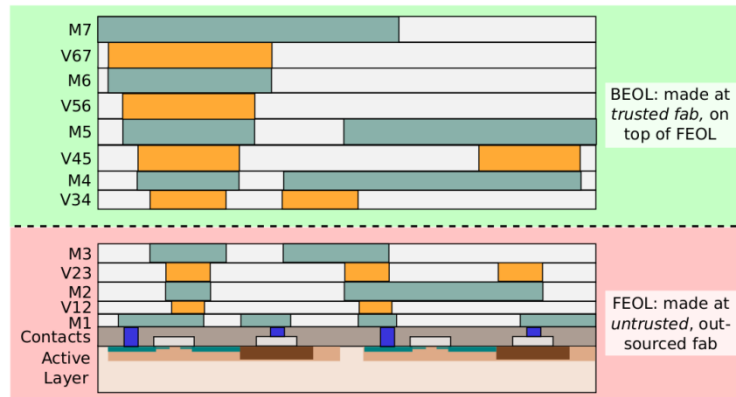
**TIC 65nm  
MPW-1 300 mm Wafer  
Global Foundries / IBM**



Patnaik et al., ASPDAC, 2018

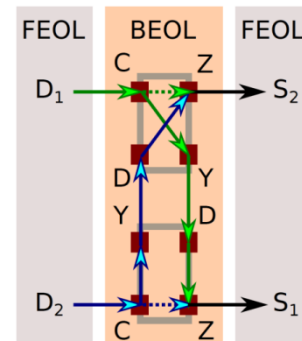
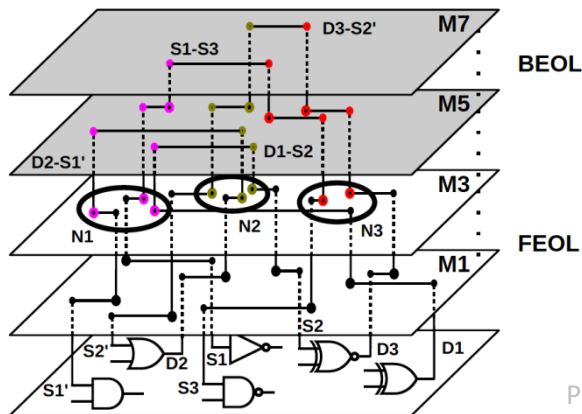
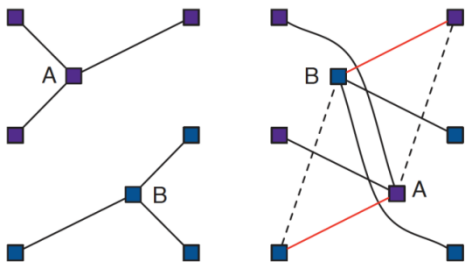
# Security-Driven Physical Synthesis

- IP protection: Split manufacturing
  - Regular synthesis works on FEOL, BEOL at once; information leakage via gate proximity, wires
  - Attacks become challenged for large circuits; also applies for ML



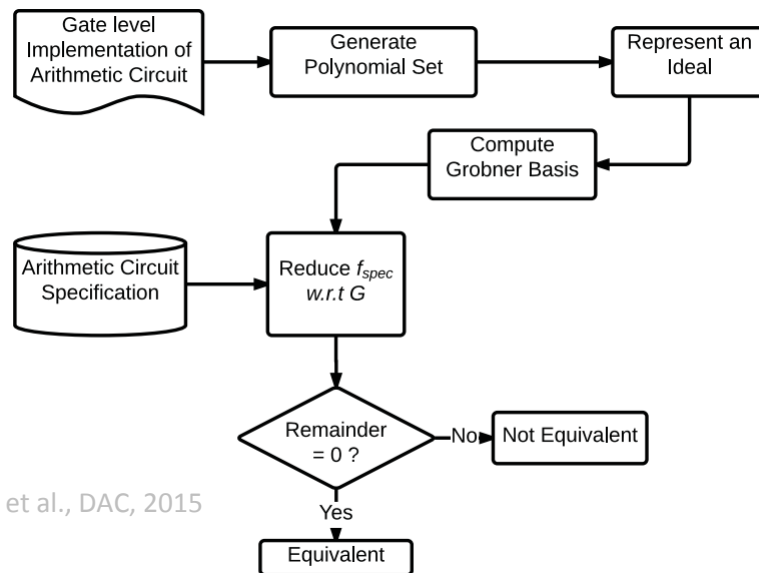
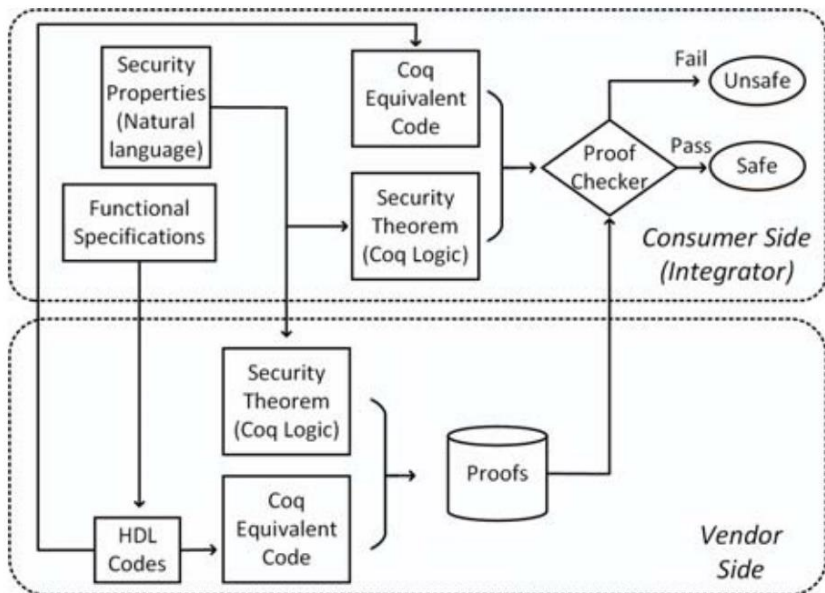
- Countermeasures: Placement and routing perturbation

- Can be well supported by synthesis
- Altering routing more effective; shown to render seminal attacks futile



# Security-Driven Functional Validation

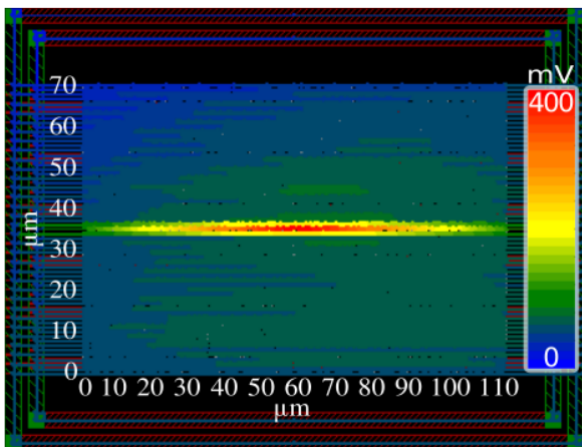
- Validation of security circuitry for error detection, logic locking, etc.
- Internal red-team vs. blue-team evaluation by running functional attacks
- Validation of information-flow tracking, proof-carrying hardware



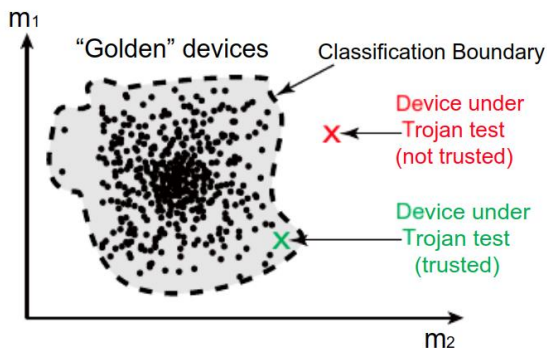
Guo et al., DAC, 2015

# Security-Driven Timing and Power Verification

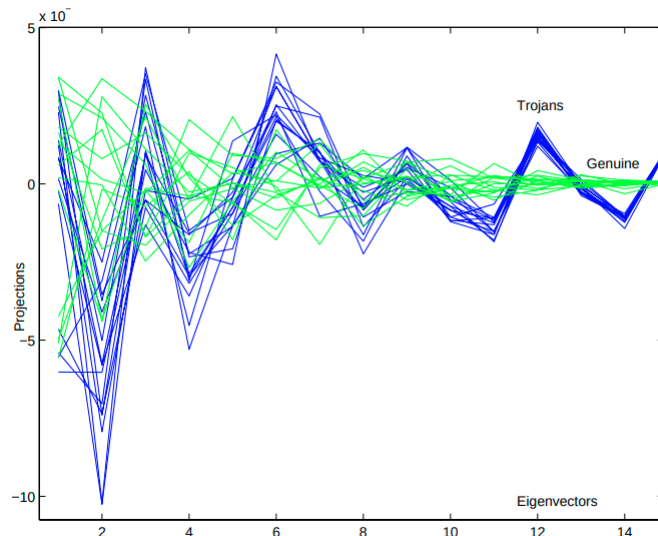
- Evaluation of side-channel, fault-injection vulnerability and countermeasures
  - Modeling efforts (detailed SPICE to fast gate-level), accuracy, runtime versus attacker's capabilities (e.g., noise distribution) versus effectiveness of countermeasure
  - Glitches influence information leakage; but may not remain present at runtime
- Trojans: Fingerprinting, i.e., statistical sampling of “golden” devices subject to variations



Viera et al., ISPD, 2018



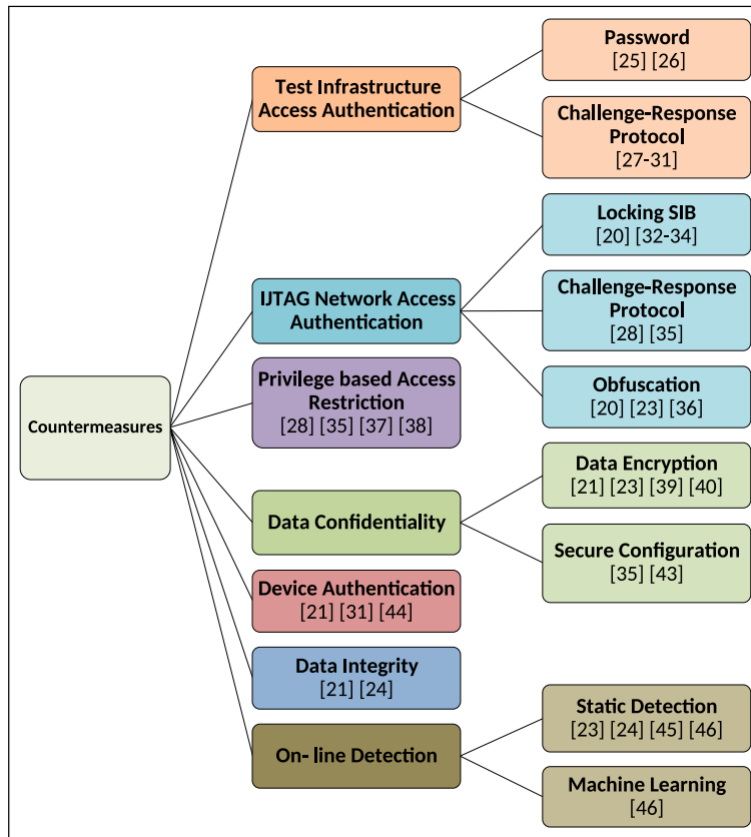
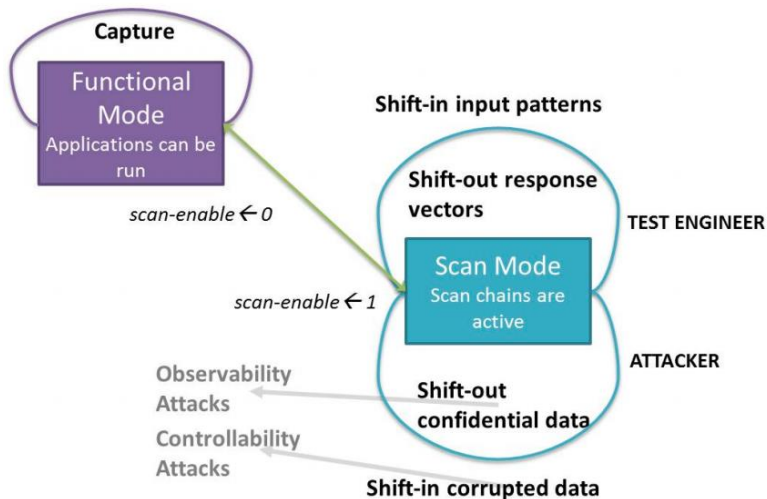
Liu et al., DAC, 2014



Agrawal et al., SP, 2007

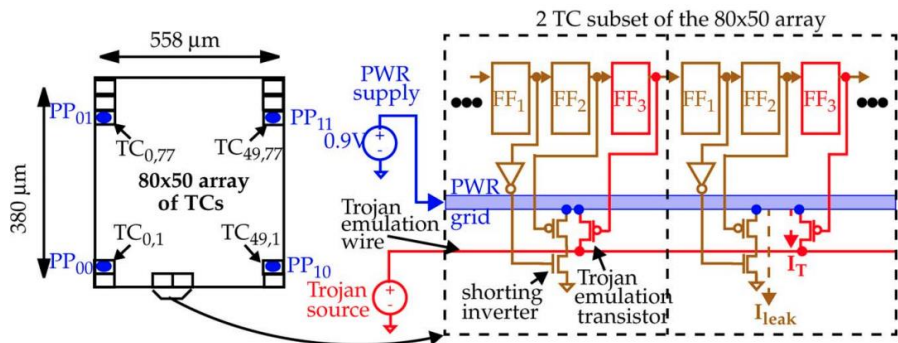
# Security-Driven Testing

- Testing, dbg infrastructure
  - Can be misused, but also protected
  - Various countermeasures
  - Further aspects, e.g., fault-injection detection and runtime reconfiguration

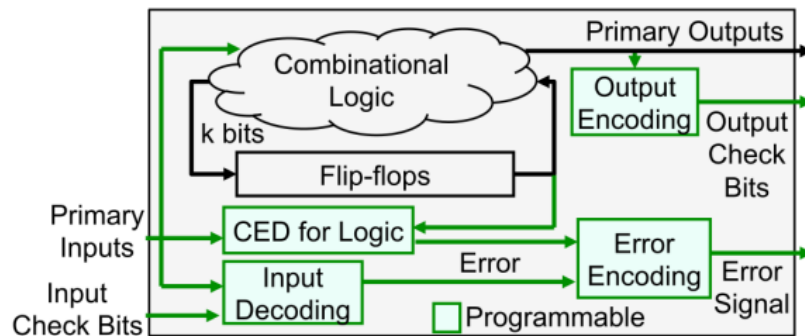


# Security-Driven Testing

- Trojans
  - Functional tests: triggering Trojans, parametric tests: fingerprinting
    - Both can be integrated in ATPG
  - Runtime monitoring infrastructures



Da Rolt et al., TETC, 2014



Wu et al., TCAD 2016



# Challenges Towards Secure Composition Using EDA Tools

- Security subject to “weakest link” – very complex problem to tackle all threats at once
  - But EDA is traditionally focused on multi-dimensional optimization problems; potential
- Threat modeling is often done on high level, ignoring physical realities
  - Once threat models are defined properly, they have to be “translated” into specific metrics and countermeasures compatible with EDA stages
  - Even then, computational efforts can become impractical; modeling versus accuracy versus attackers’ capabilities
- Not all types/implementations of countermeasures are composable
  - E.g., error-detecting logic can help side-channel attacks

## Strategies Towards Secure Composition Using EDA Tools

- Use of security-relevant metrics; varying level of detail for different EDA stages
  - Consider that metrics scale differently than PPA – e.g., a transient fault that's extremely unlikely to occur may be ignored traditionally, but for fault-injection attacks, this very fault might be leveraged
- Effective means for translation, compilation of assumptions, constraints for security schemes, all the way from system level down to “bare metal”
- Automated, holistic synthesis of countermeasures, without inducing negative cross-effects
- (Initial thoughts for further research efforts)



SCHLOSS DAGSTUHL  
Leibniz-Zentrum für Informatik

# Thanks!

## **Towards Secure Composition of Integrated Circuits and Electronic Systems: On the Role of EDA**

Johann Knechtel, Elif Bilge Kavun, Francesco Regazzoni, Annelie Heuser, Anupam Chattopadhyay,  
Debdeep Mukhopadhyay, Soumyajit Dey, Yunsi Fei, Yaacov Belenky, Itamar Levi, Tim Güneysu,  
Patrick Schaumont, and Ilia Polian

**DATE 2020** – Originated at Dagstuhl Seminar 19301,  
“Secure Composition for Hardware Systems,” July 21–26, 2019