A2: Analog Malicious Hardware

Kaiyuan Yang, Matthew Hicks, Qing Dong, Todd Austin, Dennis Sylvester
University of Michigan, Ann Arbor

Published in: IEEE Symposium on Security and Privacy 2016

Presented by: Felix Stöger

Paper overview

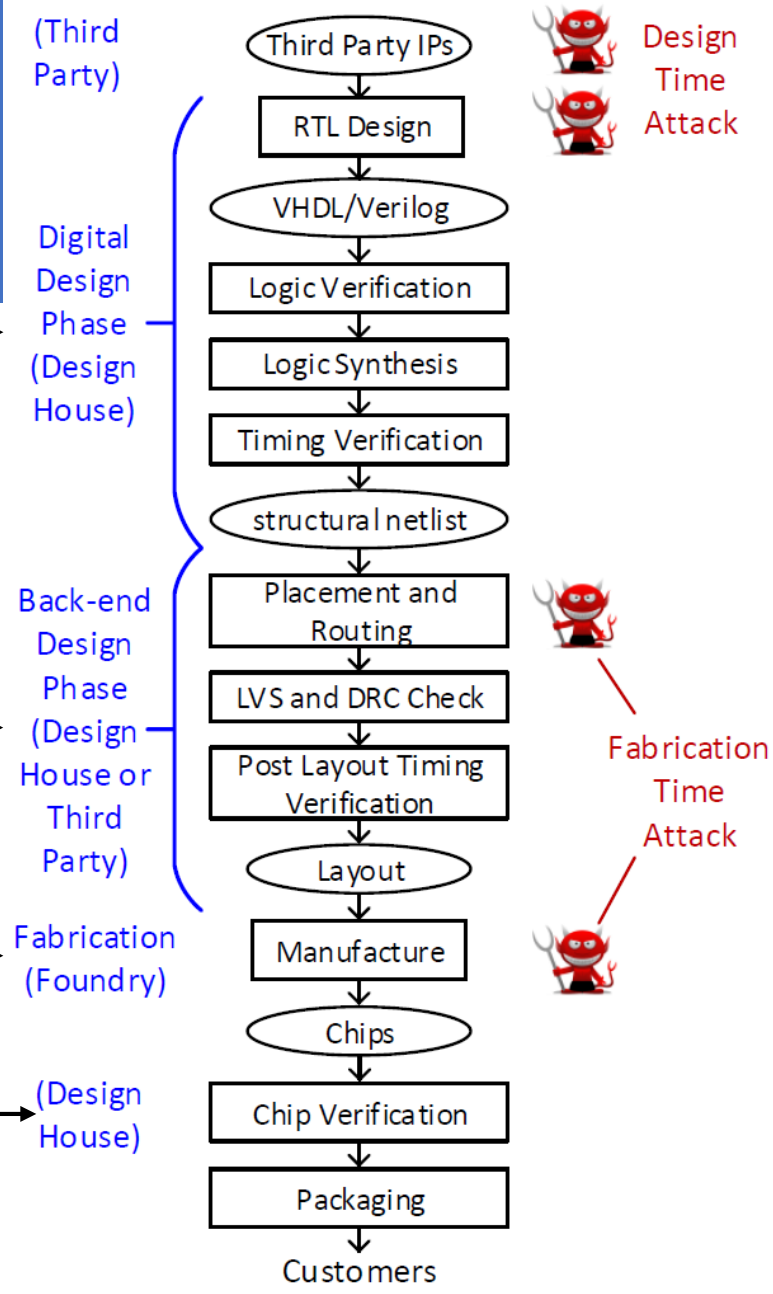
- **Problem:** Current fab-time attacks are either non-deterministic (=> non-targetable) or rely on large, complex digital circuits (=> easy to detect)
- **Goals:**
  - Propose a novel and hard to detect fabrication-time attack
  - Discuss (economical) means of protecting against fabrication-time attacks
- **Observations:**
  - Today‘s means of post-fabrication verification are very limited
  - Analog components can be used for stealthy yet powerful fab-time attacks
- **Mechanisms:**
  - SPICE (Simulation Program with Integrated Circuit Emphasis) simulation
  - Manufacturing of malicious chips to validate simulations and test proposed attack
- **Results:**
  - Creation of virtually undetectable hardware exploit using only **one gate** and occupying just one standard cell

Outline

1. **Background**
2. A2: Analog fab-time attack
3. A2: Evaluation
4. Current protections against Fab-time attacks
5. Critique, Q&A and discussion

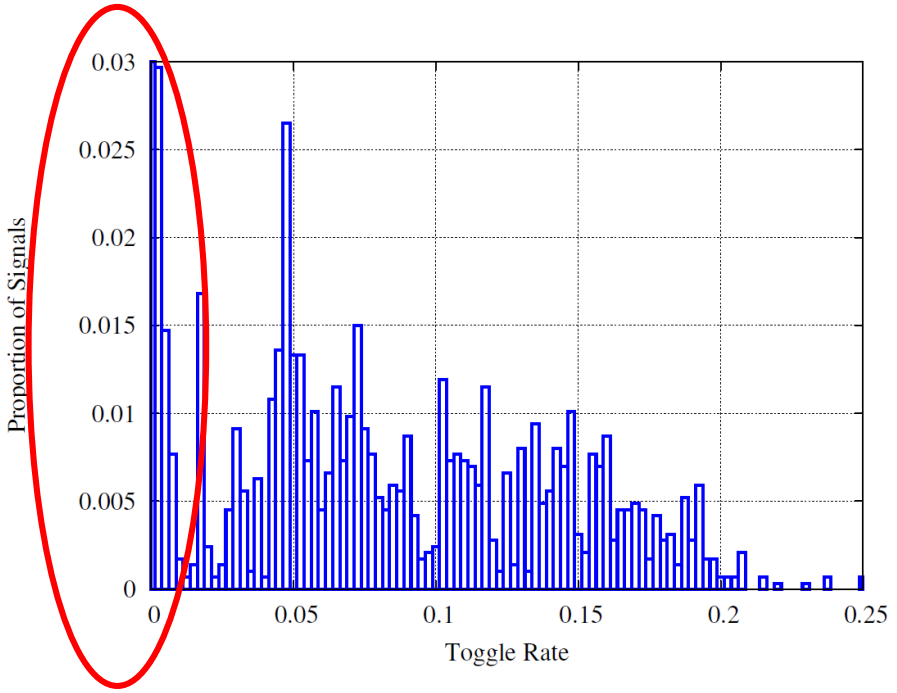
Chip manufacturing today

1. **Digital design: In-house**
   1. Add custom logic, interconnects, …
   2. Logic verification, synthesis and timing verification
2. **Back-end design:** In-hourse or Outsourced
   1. Placement of components
   2. LVS, DRC and timing checks
3. **Fabrication:** Outsourced
   1. Can make slight unnoticed changes
   2. Actual production of chip
4. **Post-fabrication verification:**
   1. Check for defects-/errors in chip
   2. Packaging and shipping



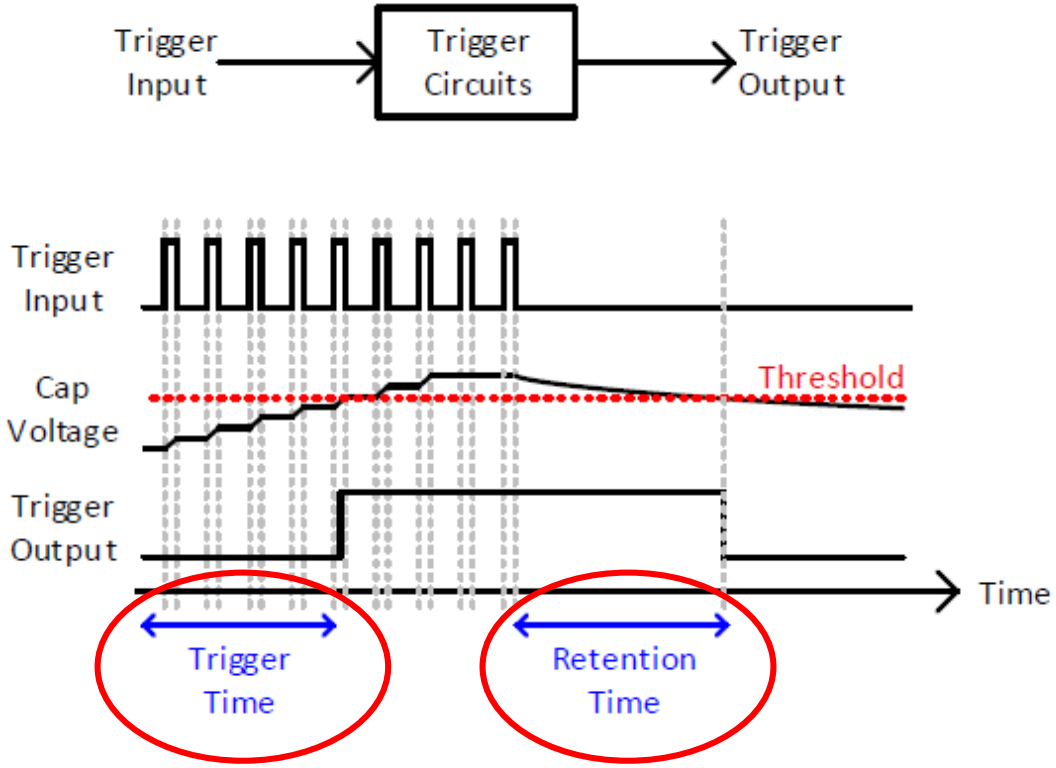
Observations of current chip manufacturing

- Modern ASIC designs consist of „standard“ cells
- ~60% - 70% of die area occupied by standard cells, remainder used for routing
  - Can use free space to siphon on wires
- On the test chip OR1200, ~ 3% of wires have almost no activity in *MiBench basicmath*



Counter-based hardware attack principles

- Attach capacitor to a wire that…
  - … is never-/infrequently toggled by regular programs
  - …can be easily toggled by attacker program
- With every toggle, capacitor's charge increases slightly
  - Implements a crude counter
  - After (roughly) **n** toggles, threshold reached
    => Detector triggered
- 2 central values:
  - Trigger time: time until threshold reached
  - Retention time: once reached, how long does it stay above threshold

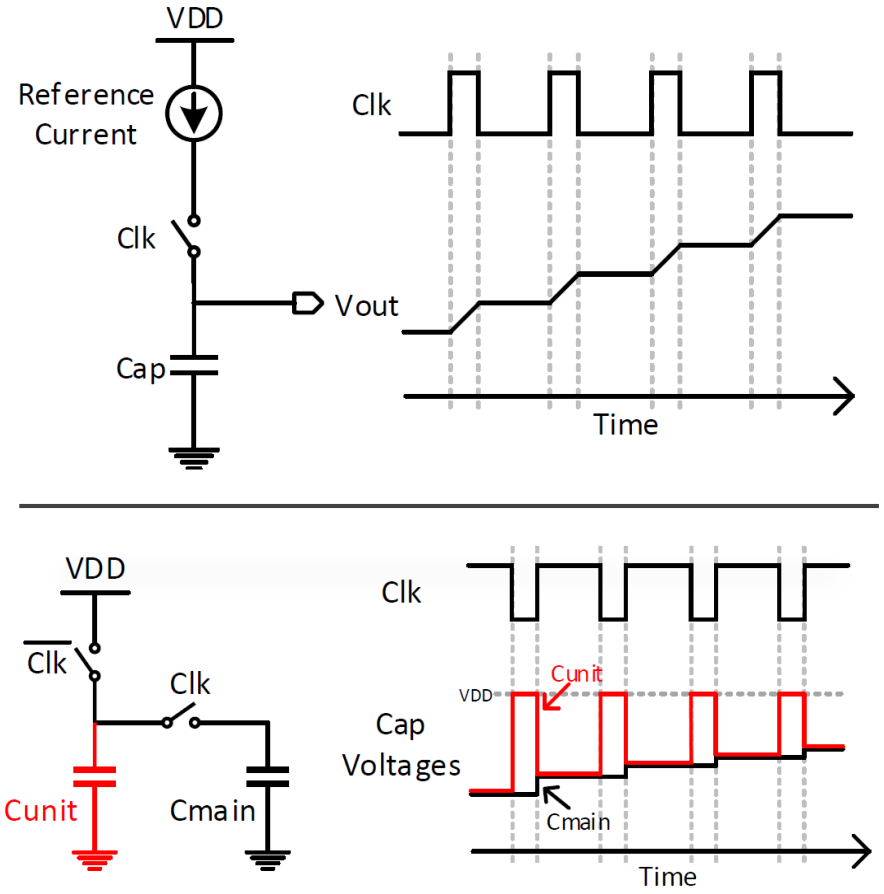


Outline

1. Background
2. **A2: Analog fab-time attack**
3. A2: Evaluation
4. Current protections against Fab-time attacks
5. Critique, Q&A and discussion

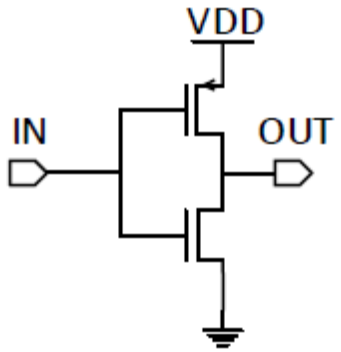
Analog counter using capacitors

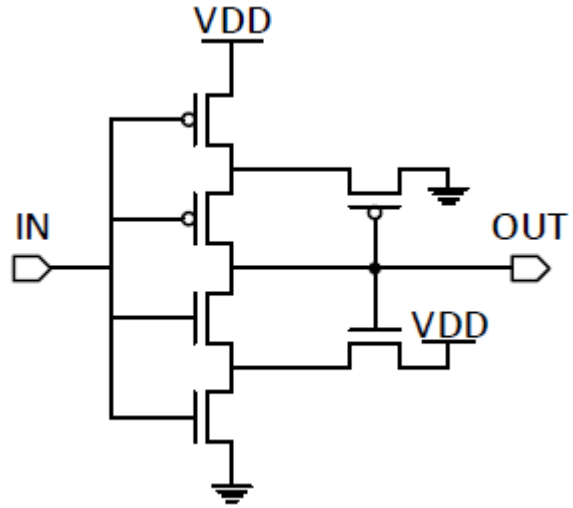
- Naive:
  - Capacitor connected to target wire and GND
  - Wire toggled => Capacitor accumulates charge
  - Charge leakage => automatic reset (prevents erroneous triggers)
  - Problem: Process variations make chosing capacitance difficult, constant high wire
- Sophisticated:
  - Add second capacitor to share charge
  - Gives edge triggered behavior

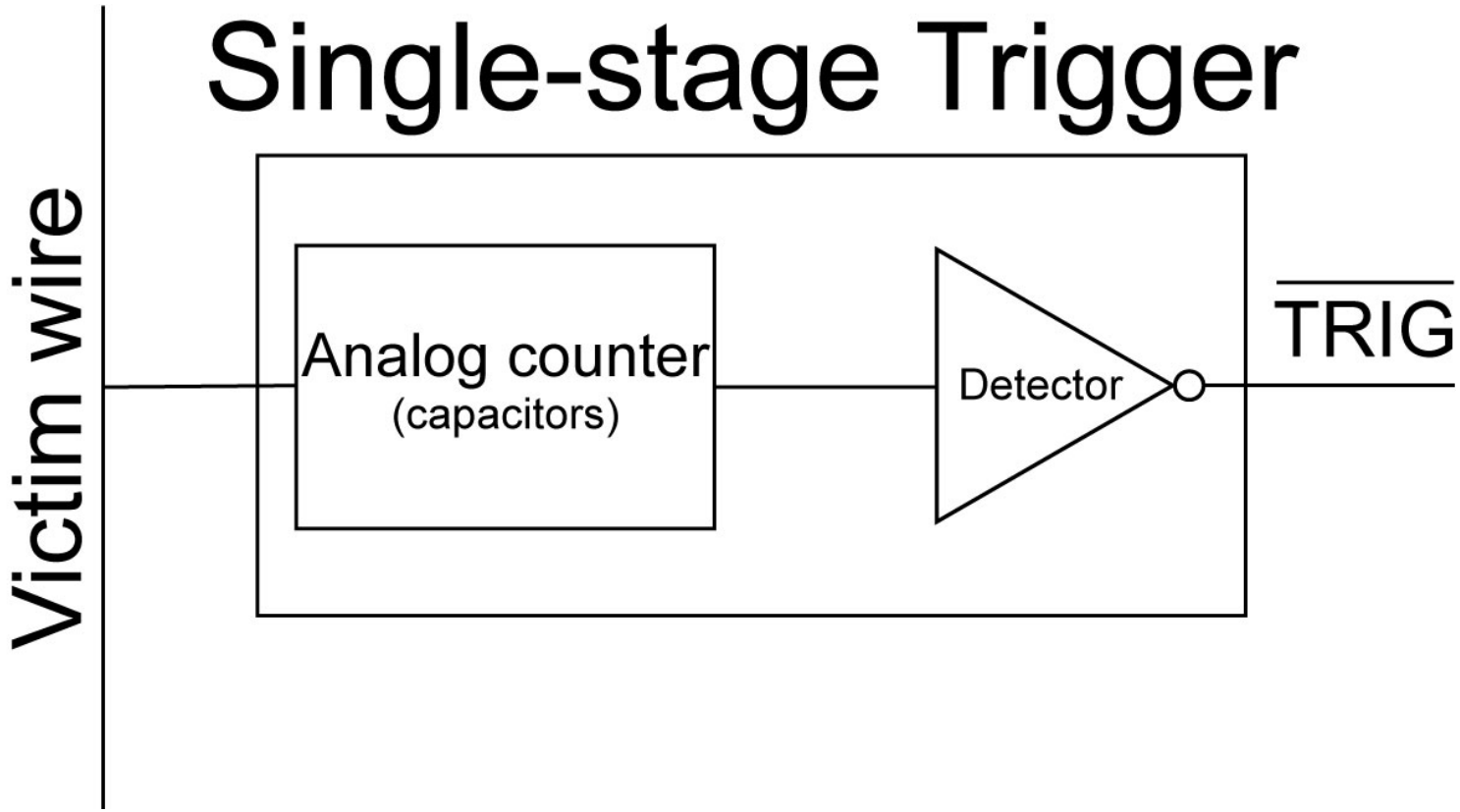


Detector

- Need to translate analog voltage level of capacitor to digital 1-/0
- Effectively need 1-bit ADC
- Skewed Inverter:
  - Counter output connected to IN
  - Skew ≙ Threshold offset
- Schmitt Trigger:
  - Comparator with asymmetric hysteresis
- **Detectors have inverted output**






Single-stage Trigger
Victim wire
Analog counter
(capacitors)
Detector
TRIG

Implementation challenges

Choice of capacitance

High capacitance

- High trigger-/retention time
- Needs more cycles
- Holds charge longer

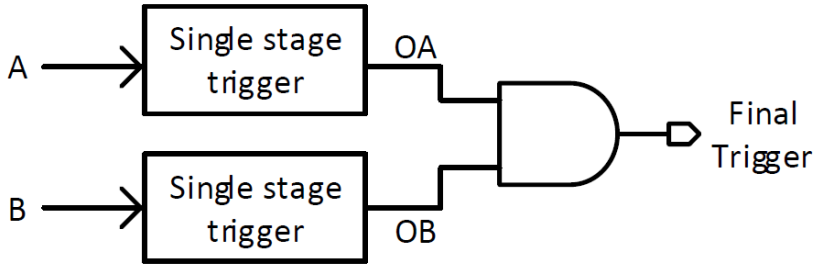
- **Low capacitance**
  - Low trigger-/retention time
  - Less toggles necessary to trigger
  - Loses charge quicker

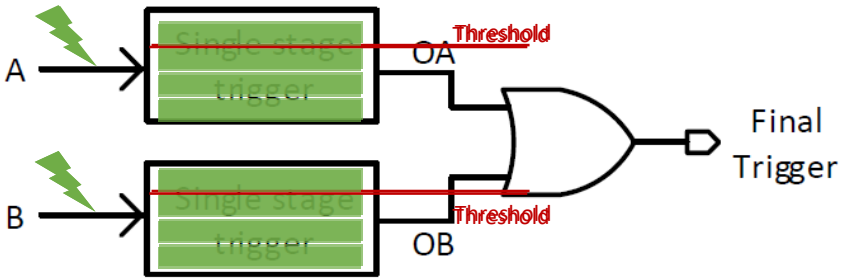
PVT variations

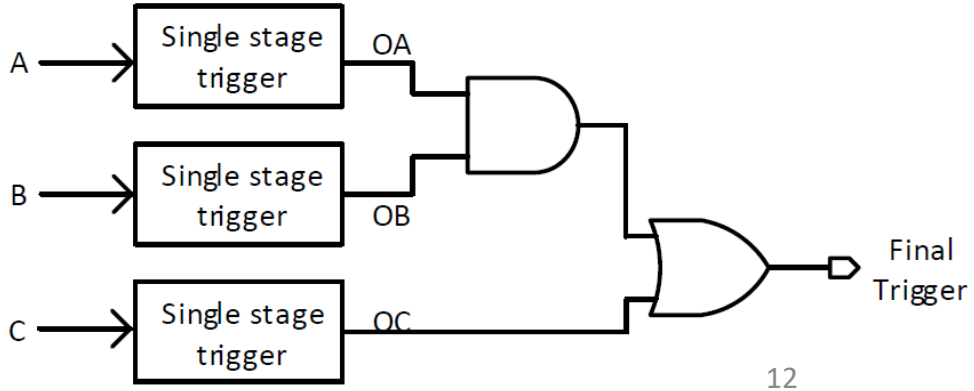
- Include headroom for process tolerances
- Exponential change of capacitance with:
  - Voltage
  - Temperature
  - Gate oxide thickness
- Need to accomodate for environment
- Knowledge of target usecase critical

Reducing false triggering by multi-stage triggers

- **Problem 1:** False triggering of single-stage triggers
- **Solution:** Attach two SST on different wires and connect them via OR gate *(inverted logic => NOR)*
  - Activated ⇔ both SST triggered
  - Can build arbitrarily complex trigger circuits
  - Makes it very unlikely to be falsely triggered
- **Problem 2:** How can one trigger mutually exclusive wires at once
  - One after another
  - Interleaved





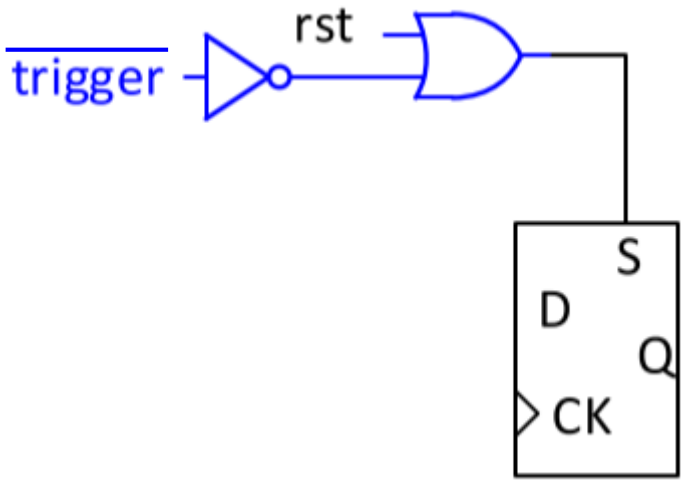


Finding victim wires and targets

- Foundries only have access to final chip layout
- Finding **victim wires** requires using **physical verification tools**
  - Produce Netlists (list of logical connections) used for simulations
  - Need to simulate circuit with sample loads to find possible targets
- Finding **target registers** requires use of carefully created test-cases
- Additionally, need to ensure that chosen implementation doesn‘t violate any **power** or **timing constraints**

Attack implementation

- **Base design:** OR1200, 65nm open-source RISC processor
- **Attack types:**
  - **Trojans** for validating end-to-end operation.
    - **Target:** Reset wire of special register
  - **Pin-level attacks** to investigate trigger behavior.
- **Procedure:**
  - Reverse engineer netlist from routes
  - Find victim wire and target register with simulations
  - Add malicious circuit
  - Fabricate chips



Attack implementation

Single-stage attack

- **Victim wire:** Divide by zero-flag
  - Few programs continuously divide by zero
  - Easy to trigger in low-level languages
- **Target register:** Supervision Register
  - Target processor has a privilege bit determining whether program runs in **Kernel** vs **Usermode**

```
{r0 is a non-zero register but reads as zero in user mode}
Initialize SR[0]=0                    {initialize to user mode}
while Attack_Success==0 do
  i ← 0
  while i < 500 do
    z ← 1/0
    i ← i + 1
  end while
  if read(special register r0) ≠ 0 then
    Attack_Success ← 1
  end if
end while
```

Multi-stage attack

- **Victim wire:** Signed and unsigned division flags
  - Few programs continuously switch between signed-/unsigned division
  - Easy to implement even in high level languages
- **Target register:** Supervision Register

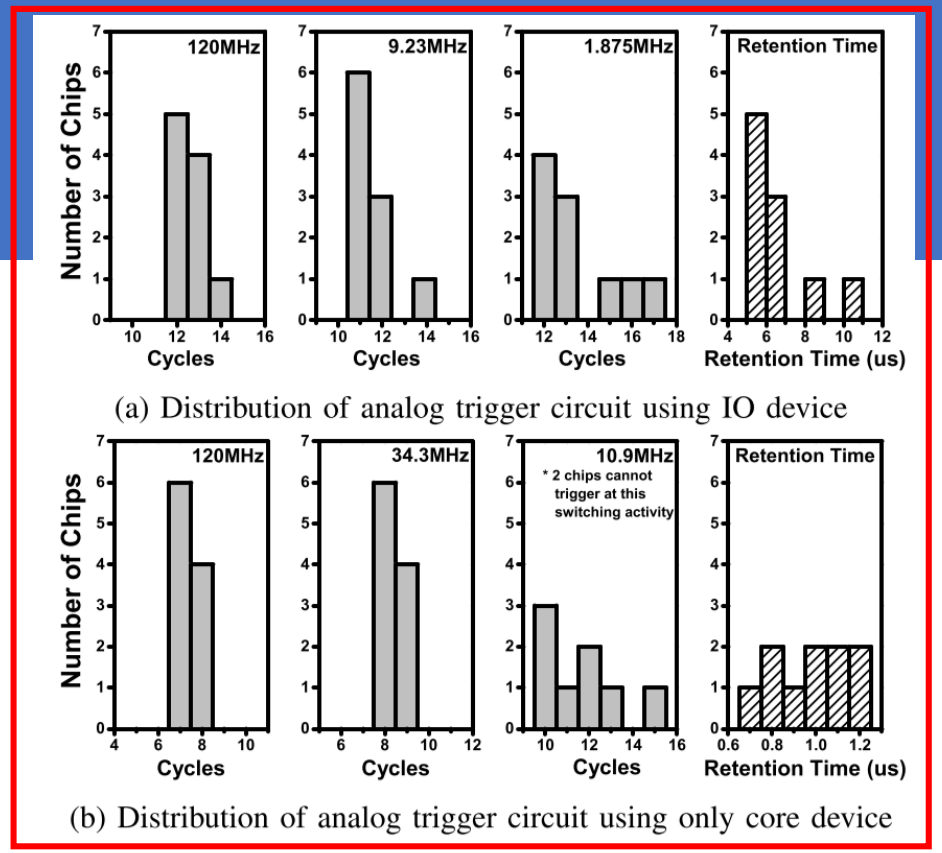
```
{r0 is a non-zero register but reads as zero in user mode}
Initialize SR[0]=0                    {initialize to user mode}
while Attack_Success==0 do
  i ← 0
  while i < 500 do
    z ← a/b                           {signed division}
    z ← c/d                           {unsigned division}
    i ← i + 1
  end while
  if read(special register r0) ≠ 0 then
    Attack_Success ← 1
  end if
end while
```

Outline

1. Background
2. A2: Analog fab-time attack
3. **A2: Evaluation**
4. Current protections against Fab-time attacks
5. Critique, Q&A and discussion

Testing and validation

- **End-to-End testing:** Testing of Trojan
  - **Result:** 100% success rate at any temperature
- **Analog characteristics:**
  - # Toggles necessary
    - Tests VS simulation
  - Retention time
    - **Simulated** – non-IO: $1\ \mu s$, IO: $25\ \mu s$
    - **Measured** – non-IO: ~$1\ \mu s$, IO: $5\ \mu s$



(a) Distribution of analog trigger circuit using IO device

(b) Distribution of analog trigger circuit using only core device

Trigger Circuit	Toggle Rate (MHz)	Measured (10 chip avg)	Simulated (Typical corner)
w/o IO device	120.00	7.4	7
w/o IO device	34.29	8.4	8
w/o IO device	10.91	11.6	10
w/ IO device	120.00	12.6	14
w/ IO device	9.23	11.6	13
w/ IO device	1.88	13.5	12

Comparison of # cycles needed in tests VS simulation

Testing detectability

- **Testing for spurious activations:**
  - **Methodology:** 5 MiBench bechmarks run to check for erroneous activations
  - **Result:** No activations from Single Stage and Multi Stage Triggers
- **Detection via power consumption:**
  - **Simulated power consumption:** $5.3nW$ for trigger with IO devices, $0.5\ \mu W$ for trigger without IO devices
    Attack power consumption hidden within operational noise
- **Detection via timing deviations:**
  - Measurements show:
  - Reference edge rising and falling delay: $19.76ps$ and $17.18ps$
  - Post-insertion edge rising and falling delays: $20.66ps$ and $18.45ps$

Program	Power (mW)
Standby	6.210
Basic math	23.703
Dijkstra	16.550
FFT	18.120
SHA	18.032
Search	21.960
Single-stage Attack	19.505
Two-stage Attack	22.575
Unsigned Division	23.206

Outline

1. Background
2. A2: Analog fab-time attack
3. A2: Evaluation
4. **Current protections against Fab-time attacks**
5. Critique, Q&A and discussion

Current protection against fab-time attacks

Side channels-/ Chip characterization

- Model runtime behavior by measuring golden (trusted) chip
- Run same tests on production sample and compare results
- Self-referencing comparisons mitigate need of golden chip

  - Activation of malicious circuitry necessary to detect

On-Chip sensors

- Always-on sensors
- Measure delays, temperatures over time
- If measurement outside tolerance => Possible malice detected

  - Only detects large malicious circuits

Filling empty space

- ASIC designs mostly utilize only 60% - 70% space
- BISA (built-in self authentication) aims to utilize every free cell with useful circuitry
  - Idea: Every cell serves a purpose => Altering even a single cell detectable

  - Hard to implement “perfect“ BISA. Filling empty space inhibits routing

Conclusion

- **A2:**
  - First analog fab-time attack
  - Almost undetectable, modular and highly dangerous
  - Had 100% success rate in end-to-end testing
- **Protection against fab-time attacks:**
  - Incredibly expensive
  - Often destructive
  - Most effective against large digital exploits
  - Detection of A2-style attack highly unlikely
- **Simulation VS testing:**
  - Test results close to simulation
  - Paves the way for future purely simulated fab-time attack research

Outline

1. Background
2. A2: Analog fab-time attack
3. A2: Evaluation
4. Current protections against Fab-time attacks
5. **Critique, Q&A and discussion**

Strengths and weaknesses

Strengths

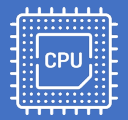
- Novel implementation: First fabrication-level attack using analog triggers
  - Modular design: Trigger independent of payload
- End-to-end approach: Went from abstract concept to concrete workingexploit
- Actually manufactured the chip
- Methodology of comparing simulations with testing
  - Showed that simulations accurate => allows future work solely based on simulation
- Relevant: With many design houses being fabless, outsorcing production becomes more popular

Weaknesses

- **Bias: Hardly any limitations mentioned**
  - **Possible issues covered in just a few sentences**
- Repetitive: Concepts were introduced at multiple points in paper
- OR1200: Obscure architecture
- Badly written:
  - Countless grammar and spelling mistakes
  - Sloppiness: *"This method of attack has four disadvantages: **1)** the attack has limited scope **2)** the attacker must have physical access to the victim device and **3)** the attack is always-on, making it more detectable and uncontrollable."*

Questions

Discussion



Could such an exploit happen on CISC, e.g. x86?

Limitations of A2-style attacks

Can you come up with similar attacks?

Possible protection against such attacks

Possible usecases

Further reading

- Extended abstract: Designer's hardware Trojan horse

  Yousra Alkabani, Farinaz Koushanfar
  2008 IEEE International Workshop on Hardware-Oriented Security and Trust

- Stealthy Dopant-level Hardware Trojans

  G. T. Becker, F. Regazzoni, C. Paar, and W. P. Burleson, "Stealthy Dopant-level Hardware Trojans," in International Conference on Cryptographic Hardware and Embedded Systems, ser. CHES. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 197–214.

- Trojan Side-Channels: Lightweight Hardware Trojans through Side-Channel Engineering

  Lin L., Kasper M., Güneysu T., Paar C., Burleson W. (2009) Trojan Side-Channels: Lightweight Hardware Trojans through Side-Channel Engineering. In: Clavier C., Gaj K. (eds) Cryptographic Hardware and Embedded Systems - CHES 2009. CHES 2009. Lecture Notes in Computer Science, vol 5747. Springer, Berlin, Heidelberg