CLKSCREW: Exposing the Perils of Security-Oblivious Energy Management

Adrian Tang, Simha Sethumadhavan and Salvatore Stolfo,

Columbia University

USENIX Security Symposium, 2017, Vancouver, Canada

presented by Sarah Tröndle

Outline

- Problem, Goal and Novelty
- Background
- Key Challenges and Solutions
- Mechanism
- Summary
- Strength and Weaknesses
- Takeaways
- Discussion

Problem, Goal and Novelty

Energy Management

- Commodity devices, such as phones, capable of extremely power intensive computations
- Need to preserve energy when not using maximal performance

→ Energy Management is essential

source picture: https://www.ccp.com.au/b-lithiumbatterychargingadvice/

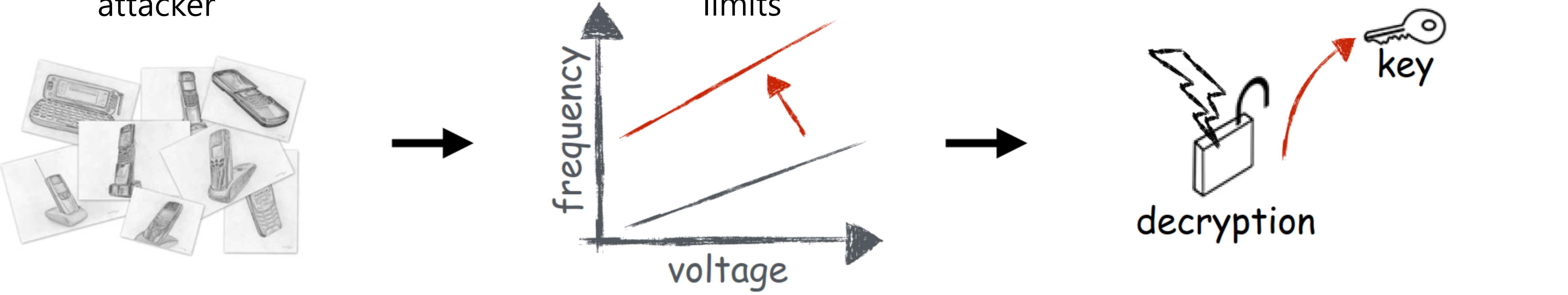
Energy Management and Security

- Today’s energy management:
  - is essential and everywhere
  - usually security is not a big consideration in it’s designs

    → might impose risk on most devices

Goal

- Show importance of security in energy management
- Do so by example attack on ARM Trustzone of Nexus 6 device



Software-based attacker

Stretch operational limits

Induce faults

Novelty

- First security review of energy management technique: Dynamic Voltage and Frequency Scaling (DVFS)
- Fault attack purely from software
- New class of exploitations: induce fault by scaling frequency → CLKscrew

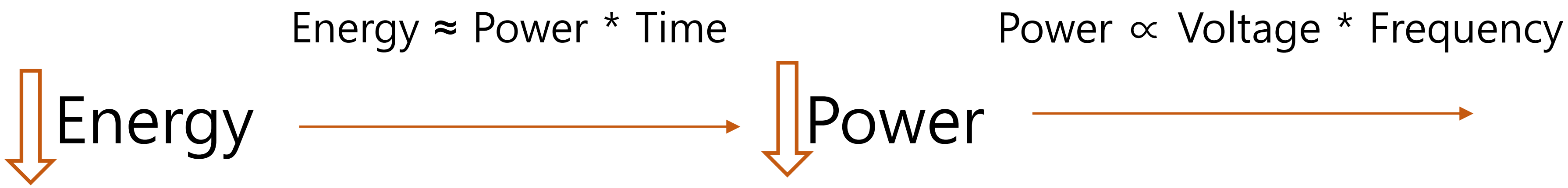
Background

Dynamic Voltage & Frequency Scaling (DVFS)

Energy ≈ Power * Time

⇩ Energy ⟶

Dynamic Voltage & Frequency Scaling (DVFS)



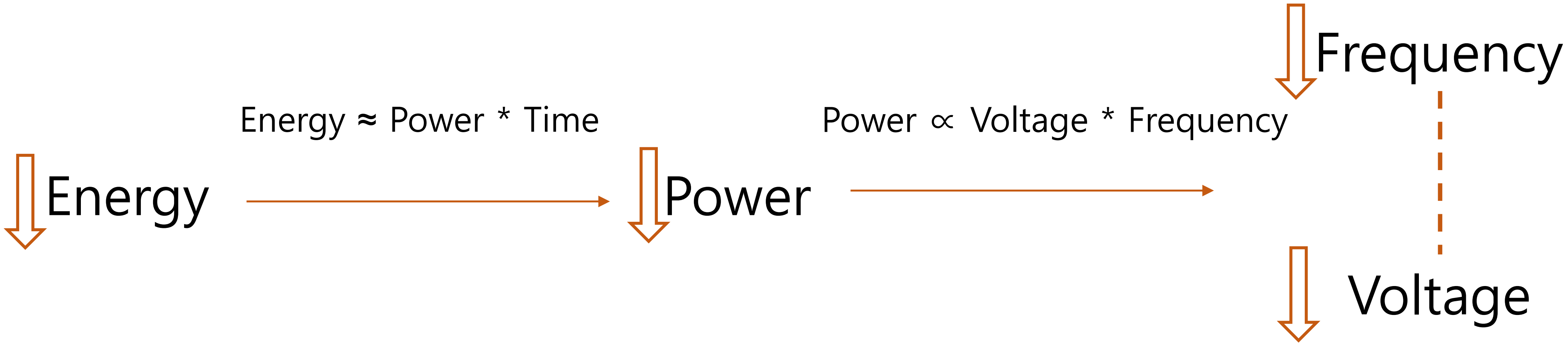
Energy ≈ Power * Time

Power ∝ Voltage * Frequency

Energy

Power

Dynamic Voltage & Frequency Scaling (DVFS)



Energy

Energy ≈ Power * Time

Power

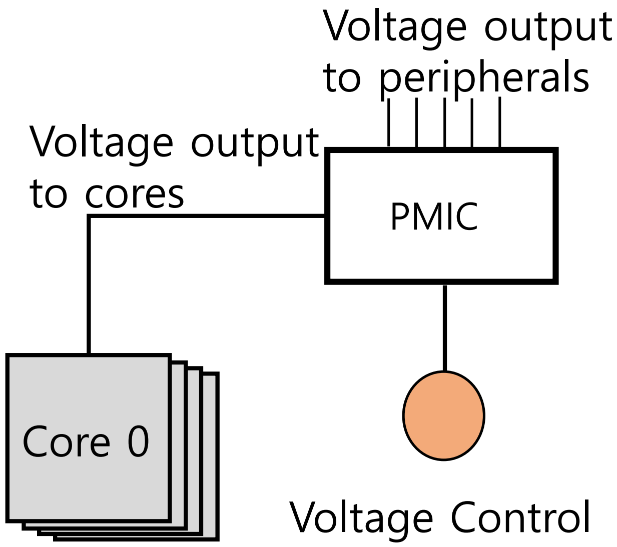
Power ∝ Voltage * Frequency

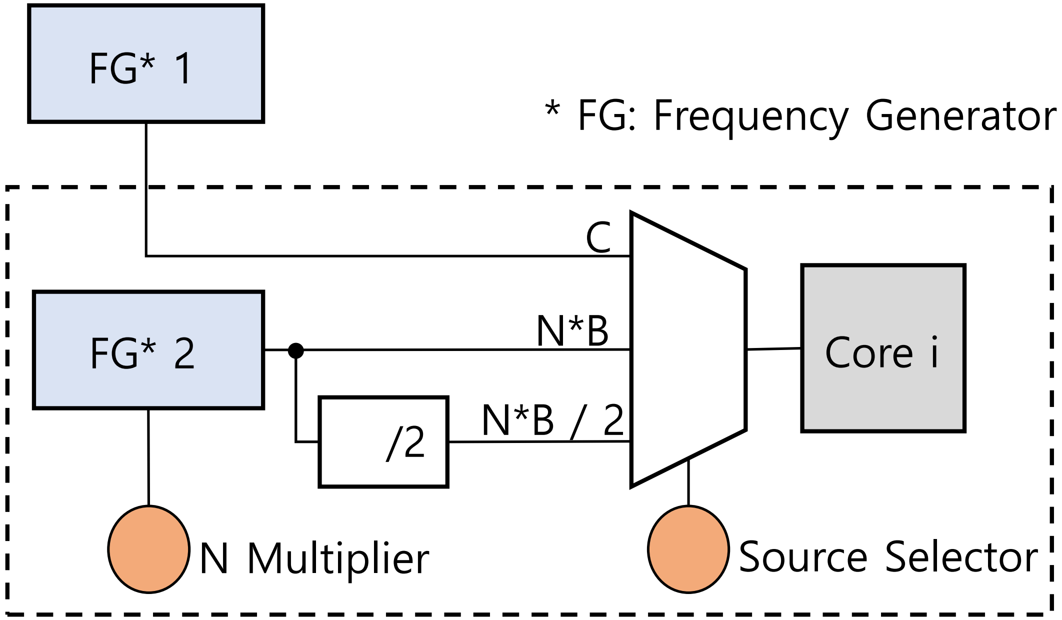
Frequency

Voltage

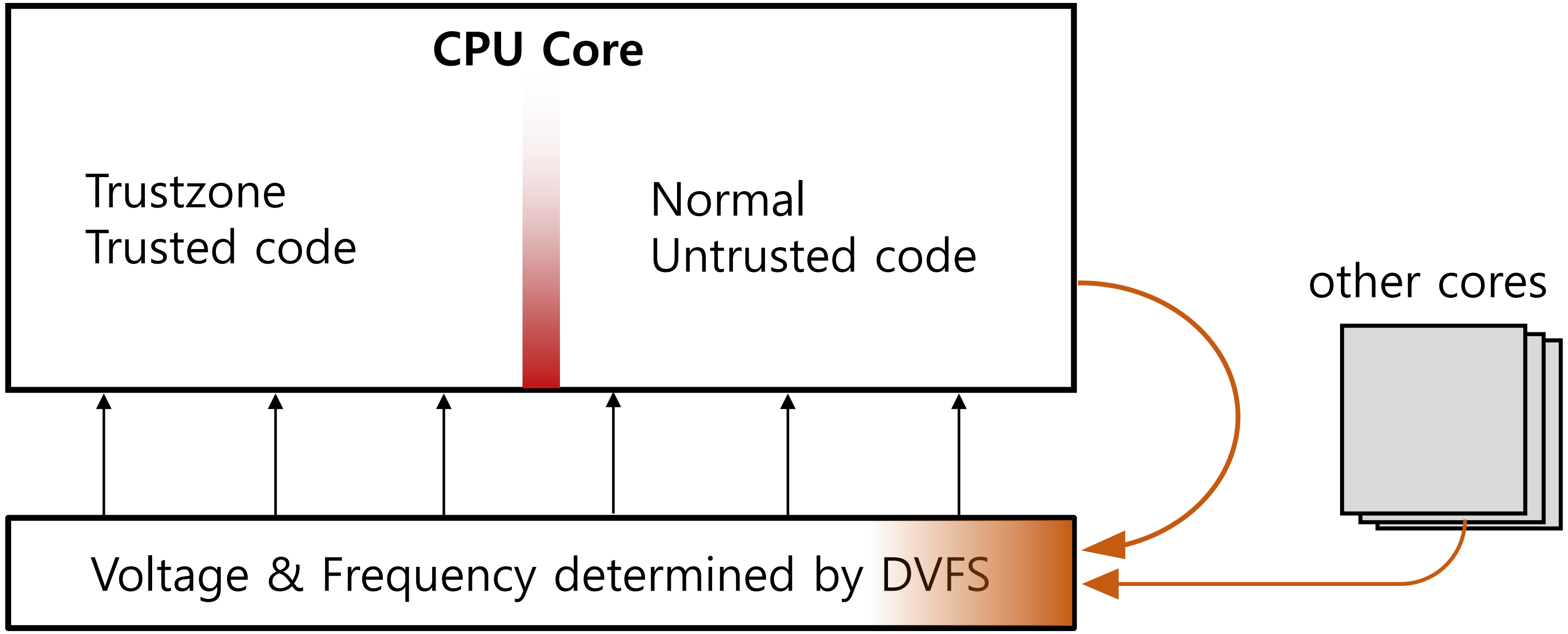
Dynamic Voltage & Frequency Scaling

- DVFS allows software control of voltage and frequency





DVFS and Trustzone



CPU Core

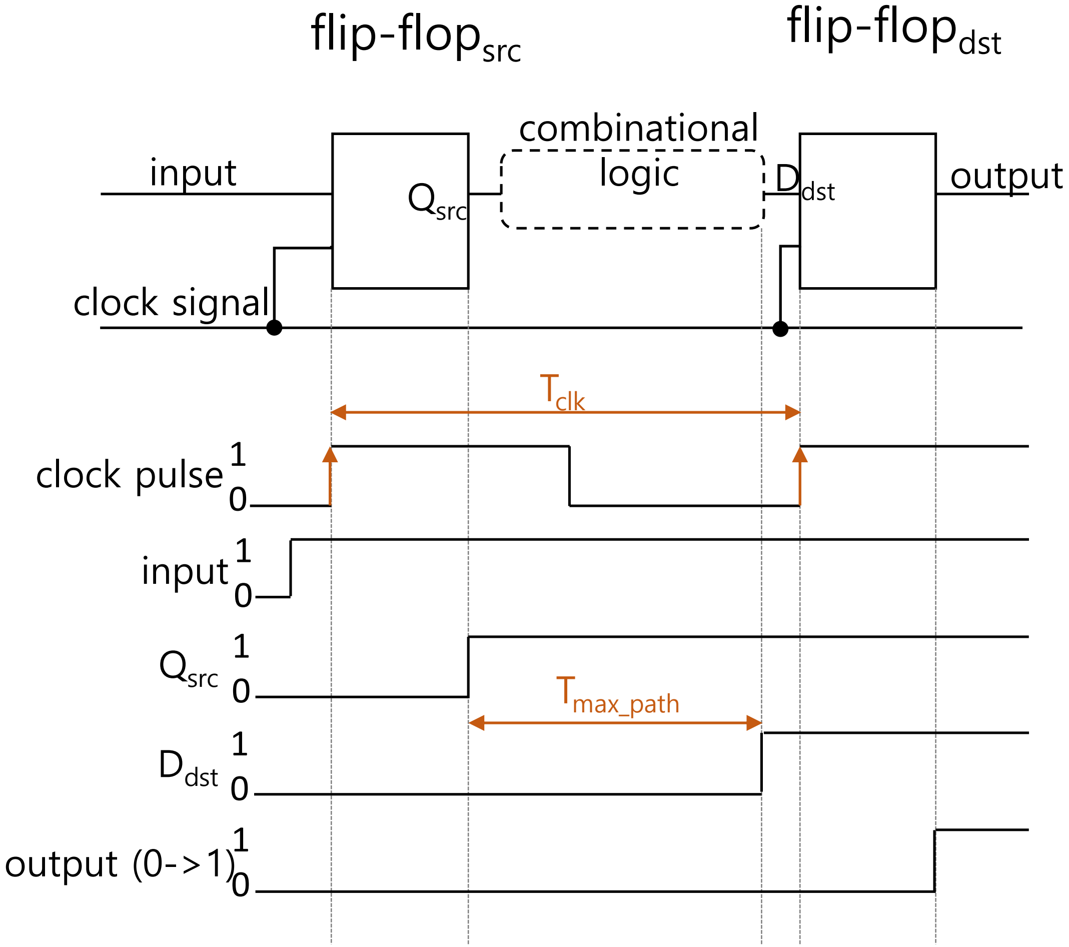
Trustzone
Trusted code

Normal
Untrusted code

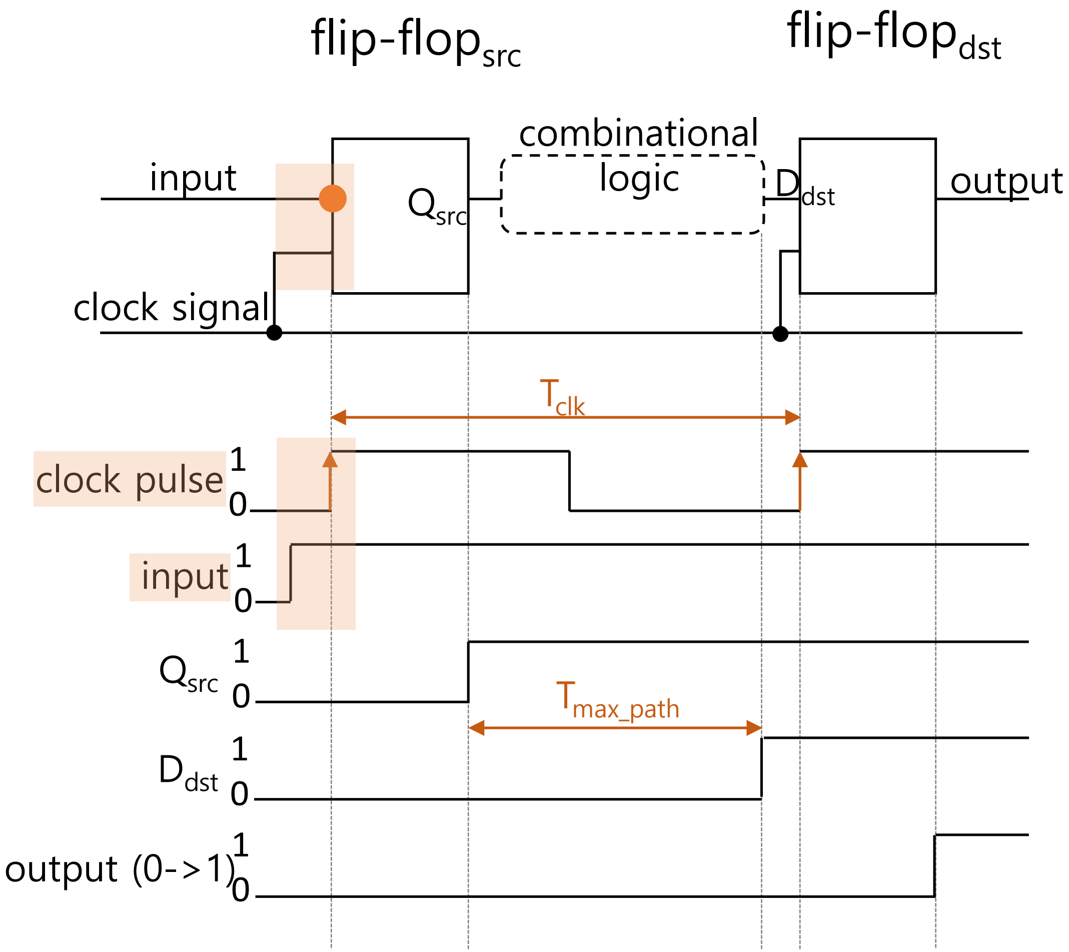
other cores

Voltage & Frequency determined by DVFS

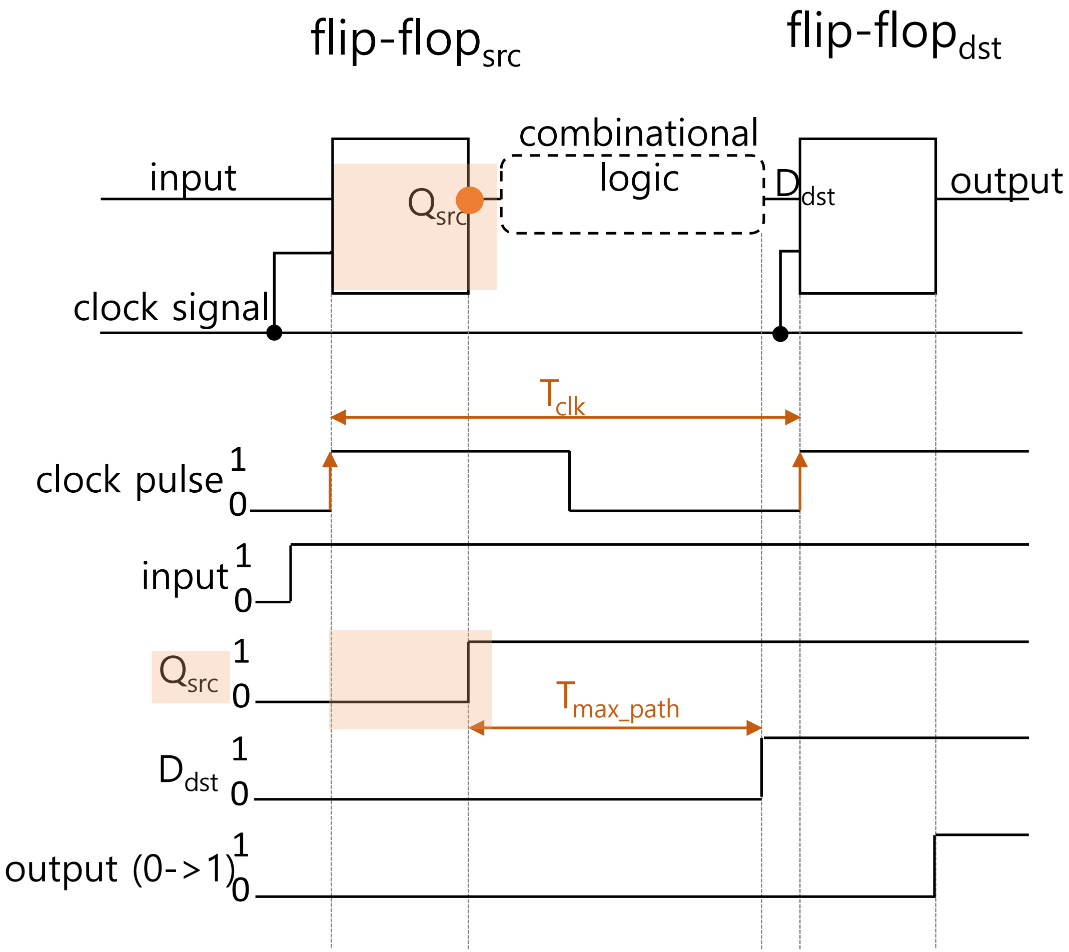
Overclocking and Undervolting



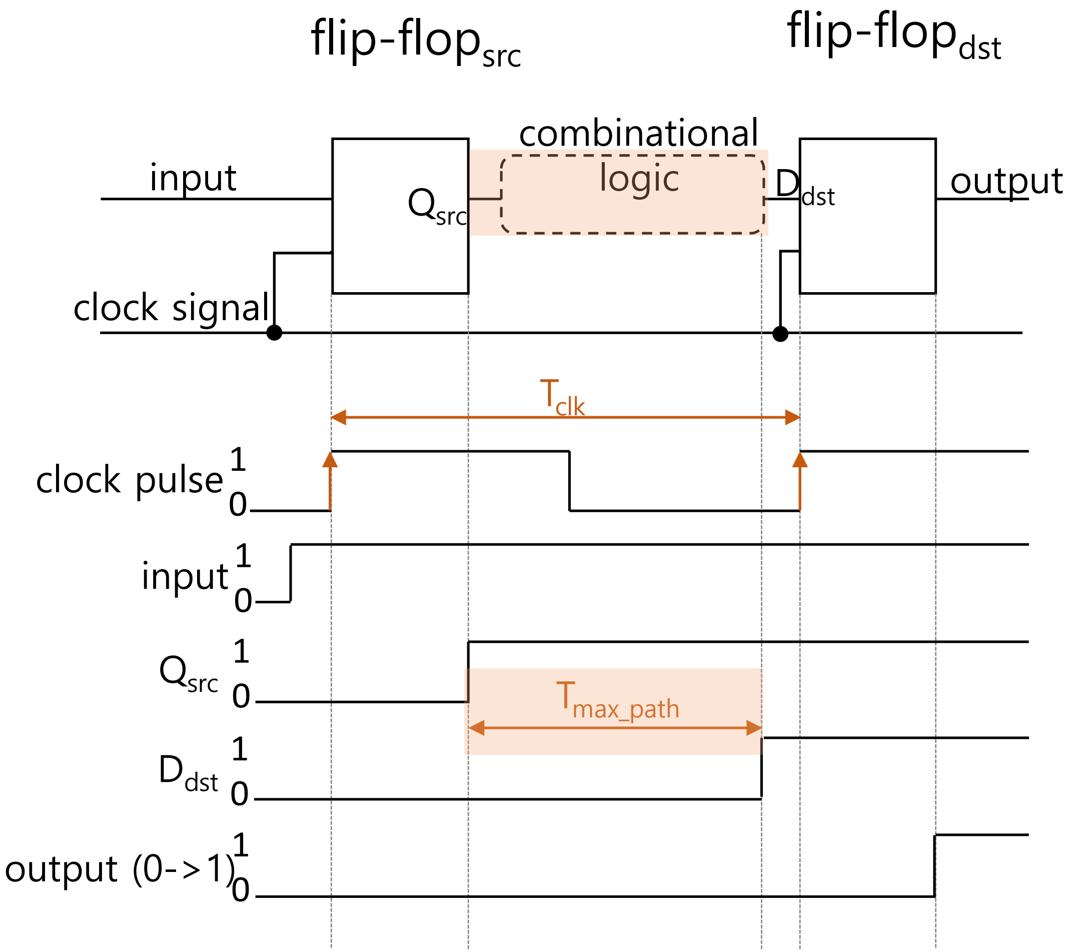
Overclocking and Undervolting



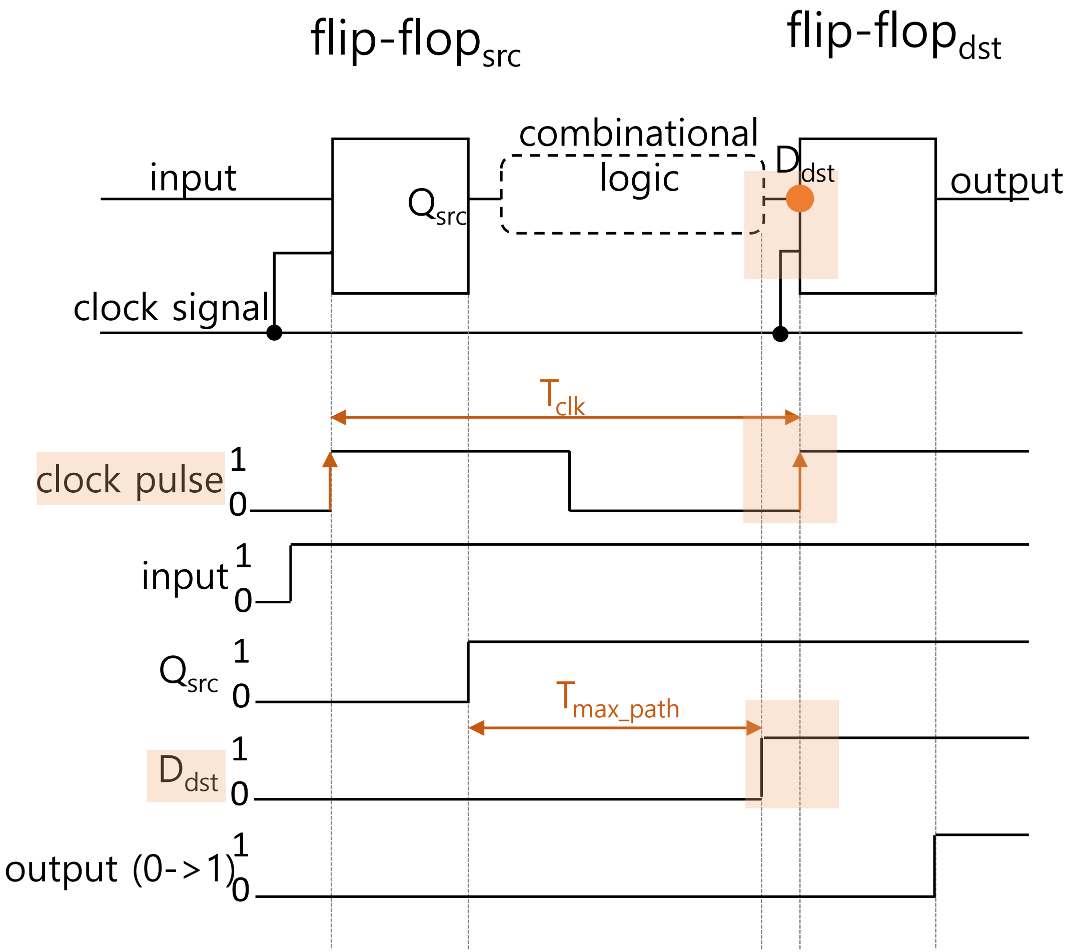
Overclocking and Undervolting



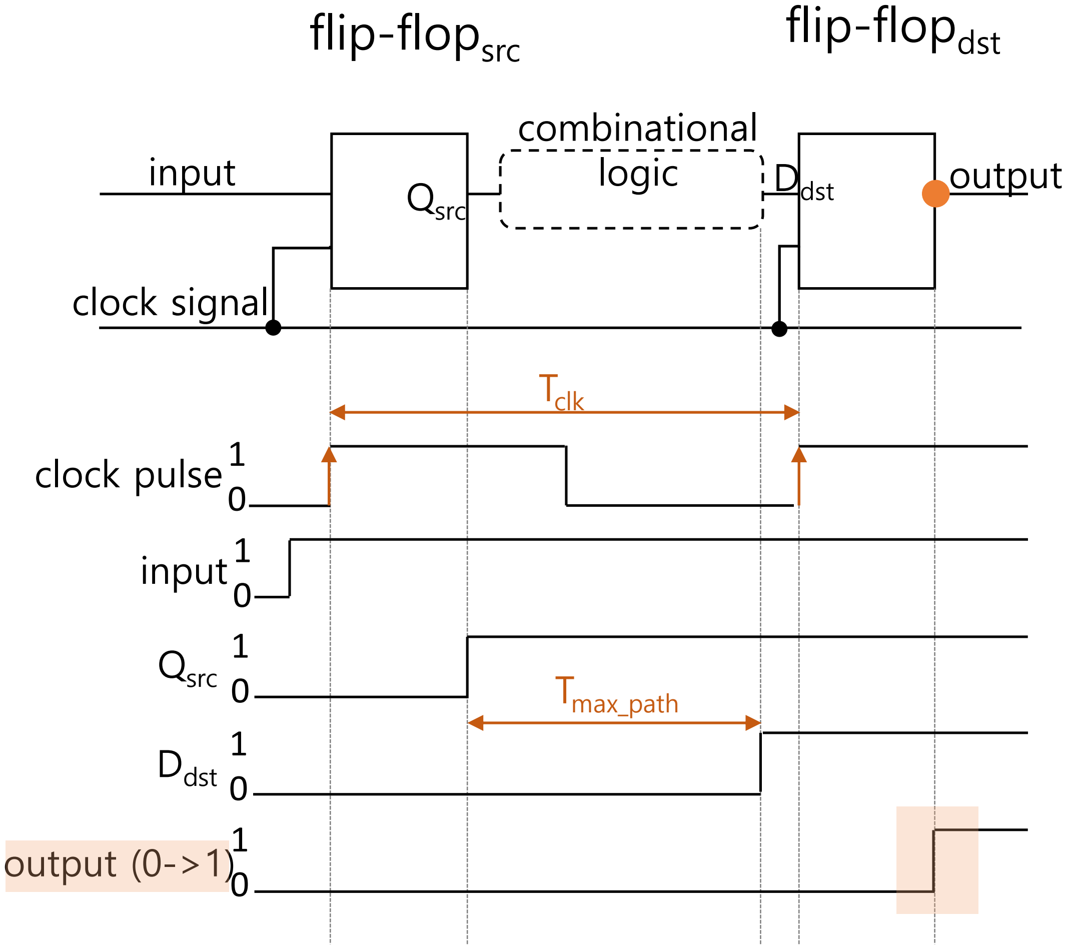
Overclocking and Undervolting



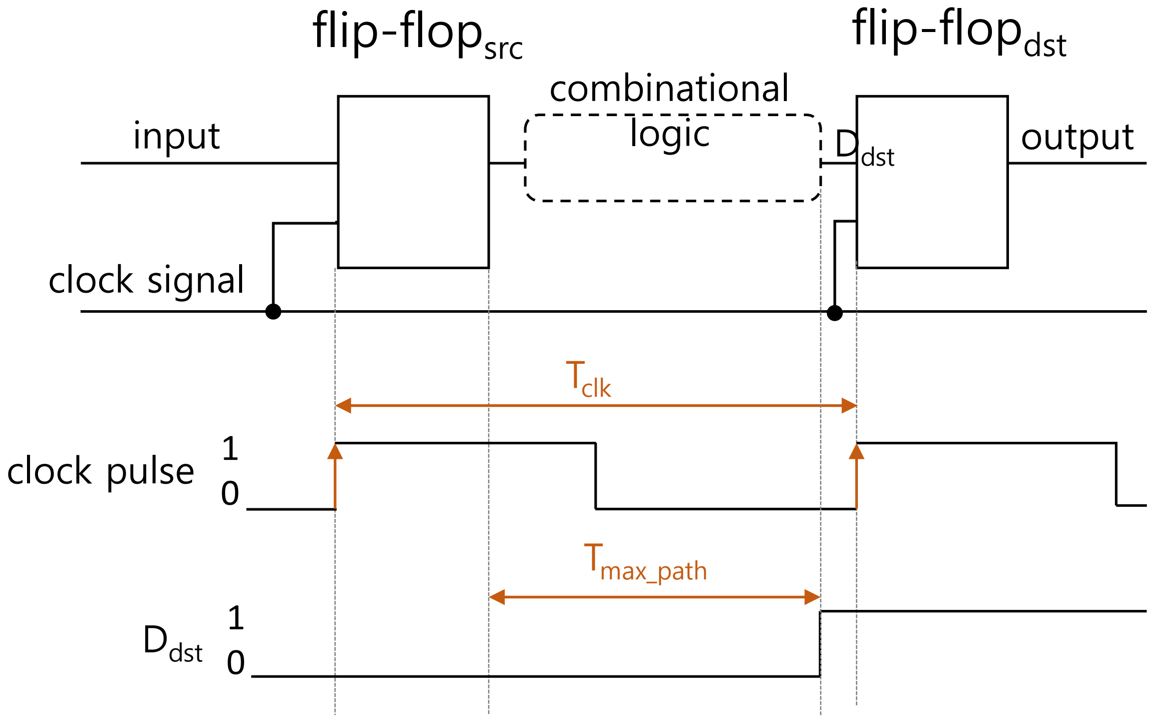
Overclocking and Undervolting



Overclocking and Undervolting

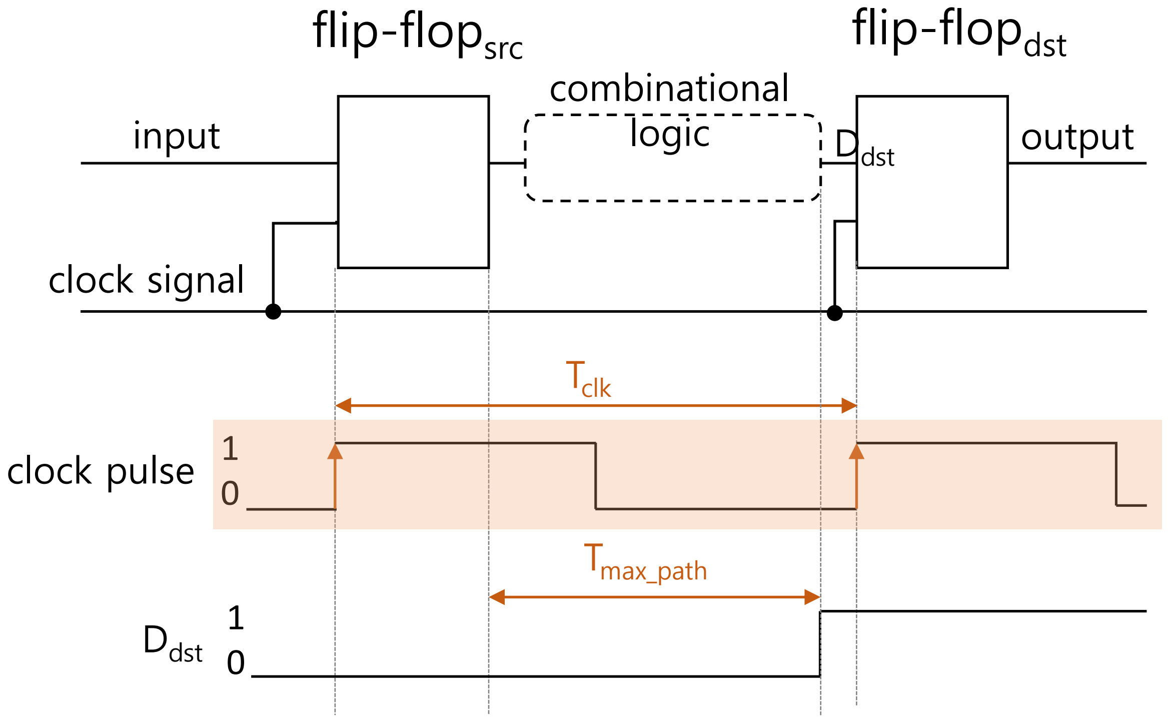


Overclocking and Undervolting



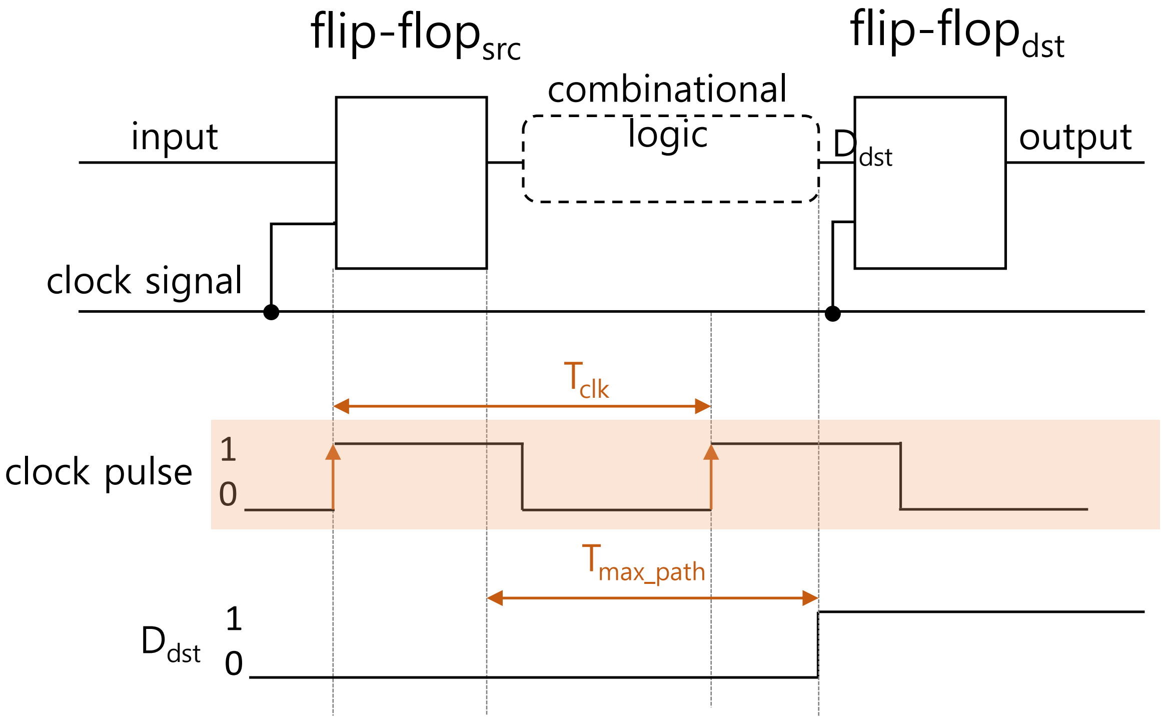
output (0->1)

Overclocking and Undervolting



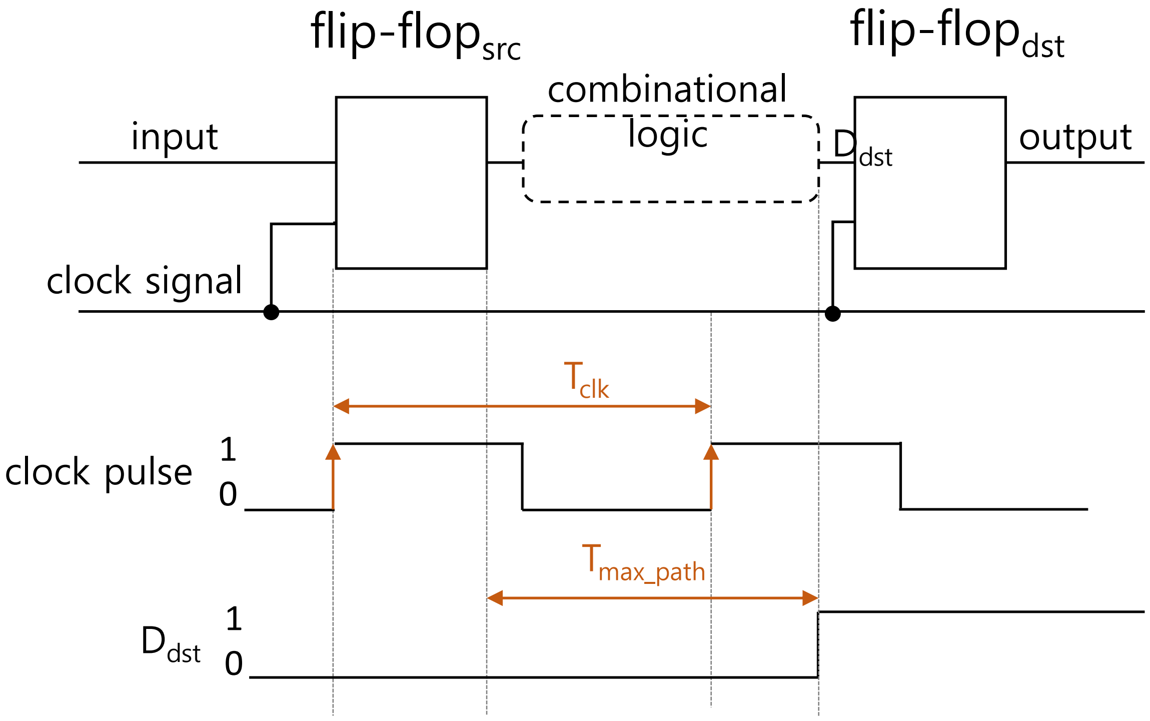
output (0->1)

Overclocking and Undervolting



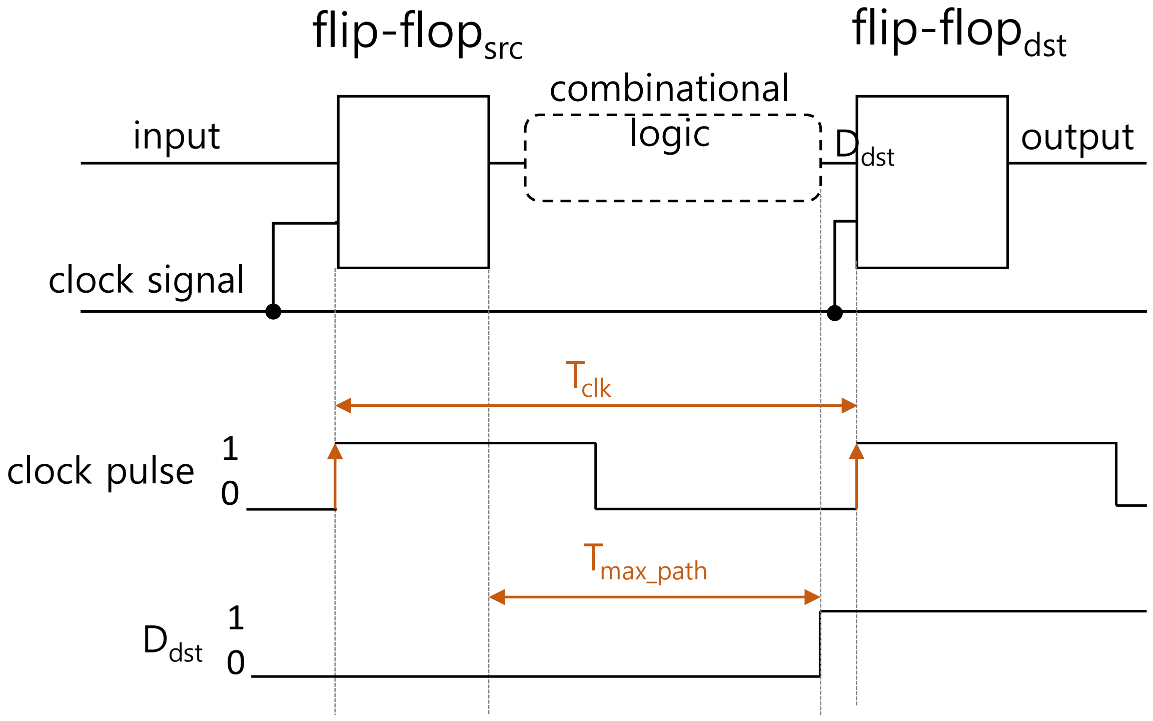
output (0->0)

Overclocking and Undervolting



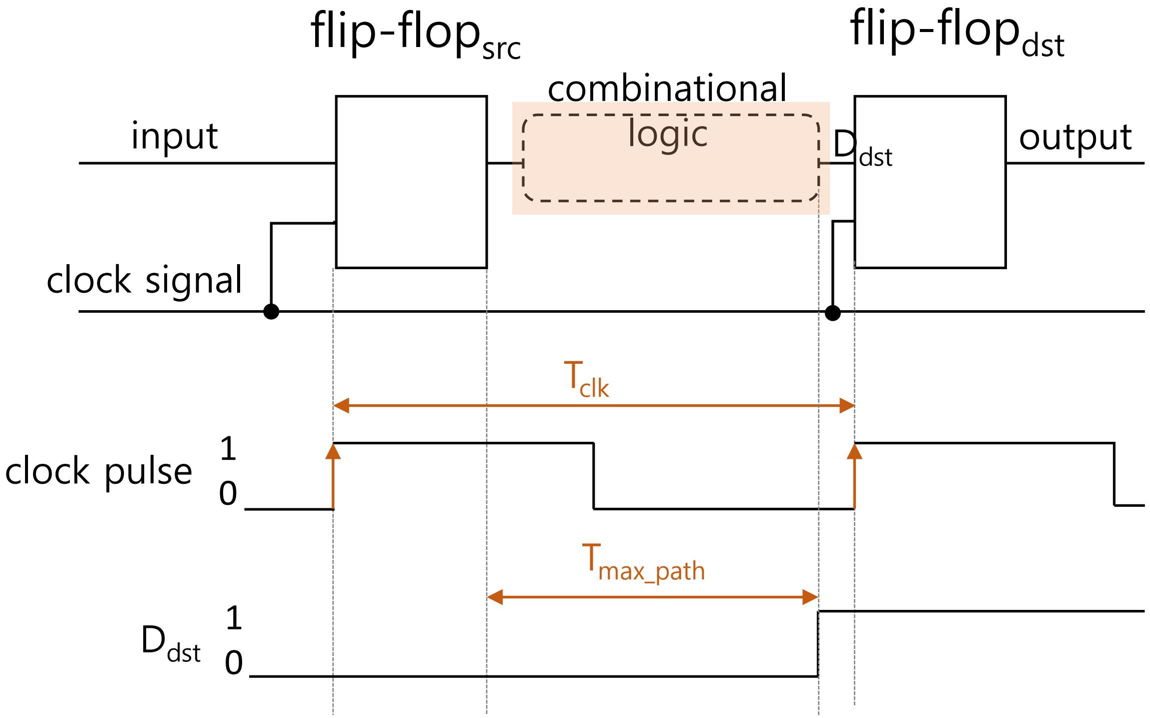
output (0->0)

Overclocking and Undervolting



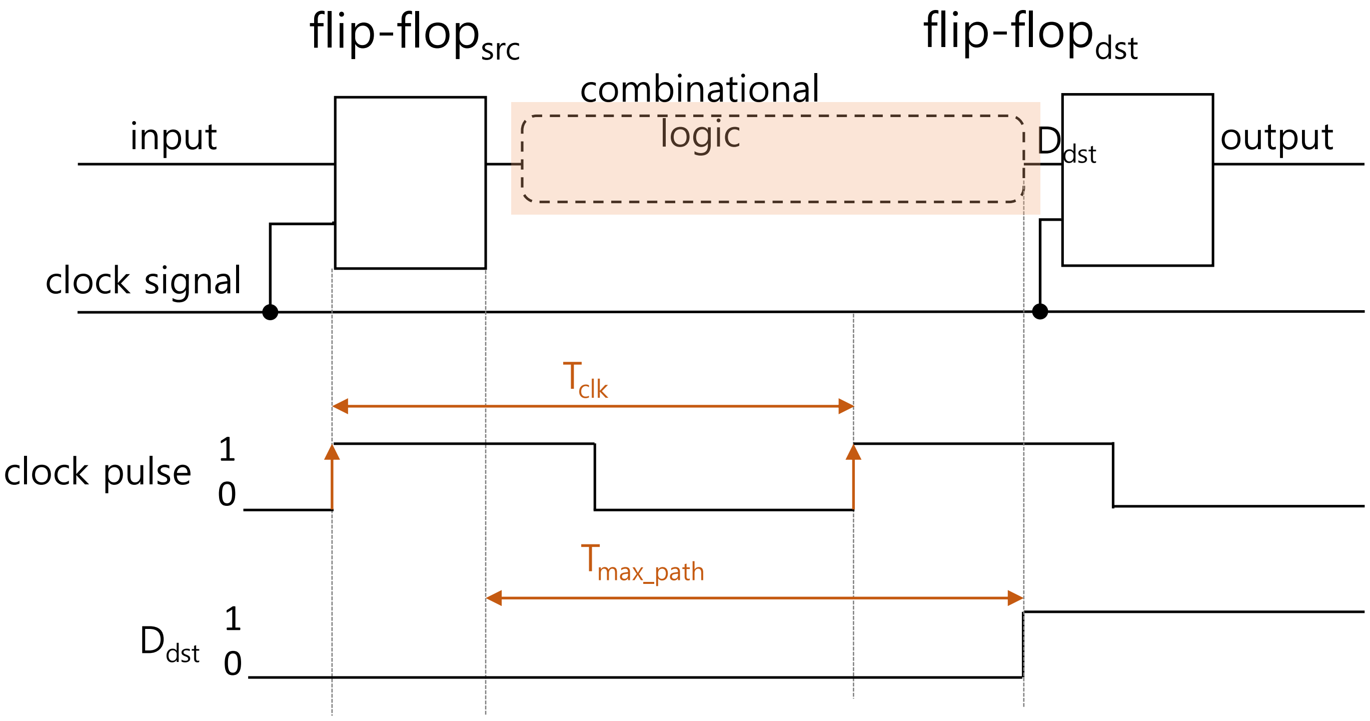
output (0->1)

Overclocking and Undervolting



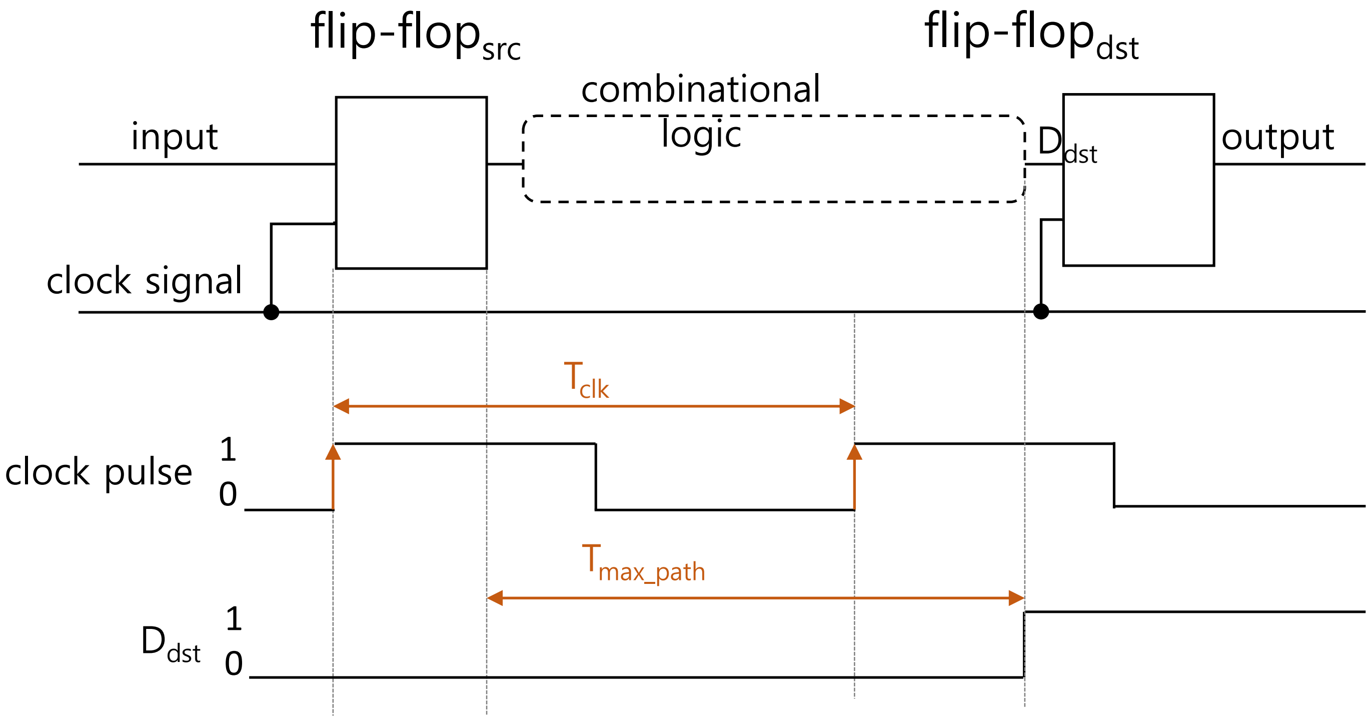
output (0->1)

Overclocking and Undervolting



output (0->0)

Overclocking and Undervolting



output (0->0)

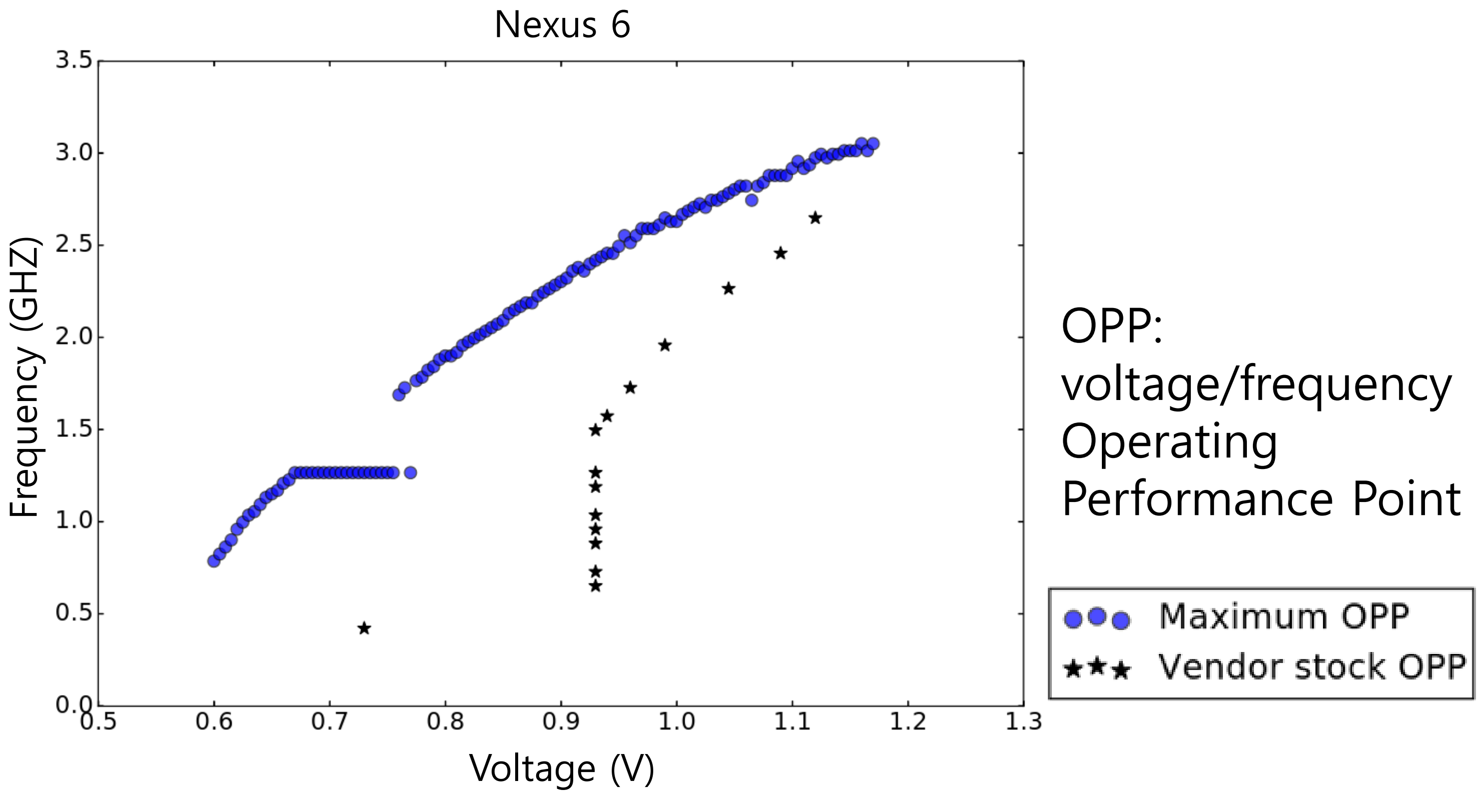
Key Challenges and Solutions

Challenges

- Voltage and Frequency operating limits?
- Self-containment: how to cause fault for victim without an error in the attacker?
- Can attack run without other things interfering?
- How to time attack correctly?

Solutions

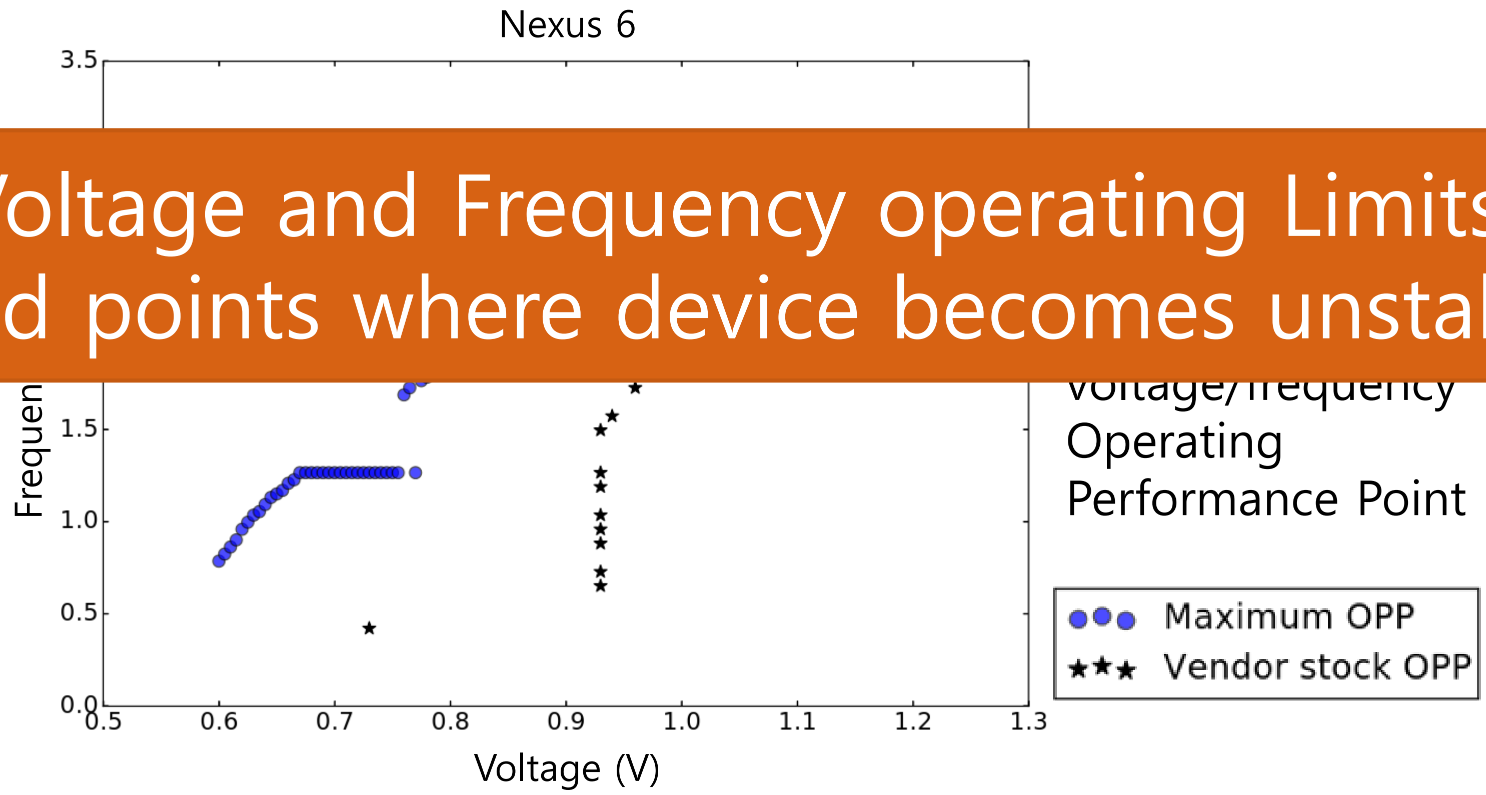
Voltage and Frequency Operating Limits?



OPP:
voltage/frequency
Operating
Performance Point

Solutions

Voltage and Frequency Operating Limits?

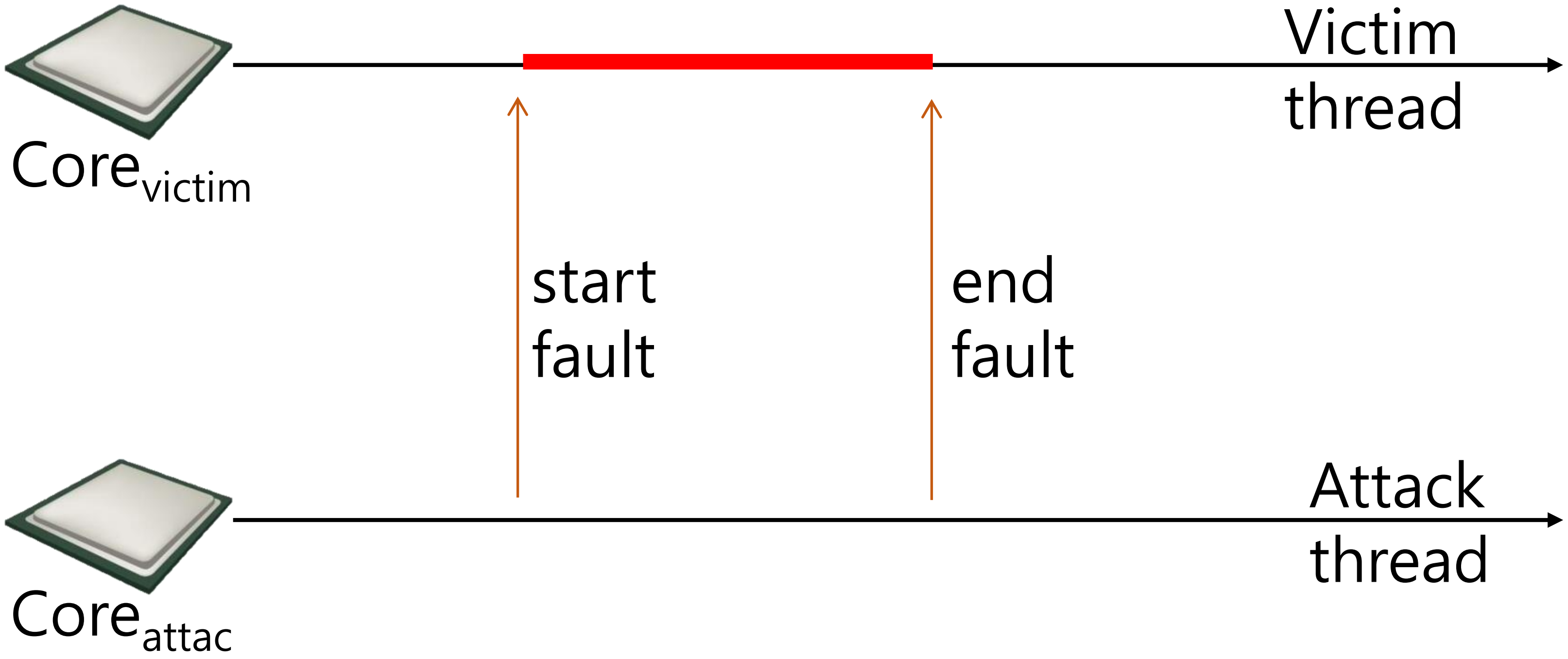


Voltage/frequency Operating Performance Point

No Voltage and Frequency operating Limits. Found points where device becomes unstable.

Solutions

Self-Containment



Victim thread

Core$_{\text{victim}}$

start fault

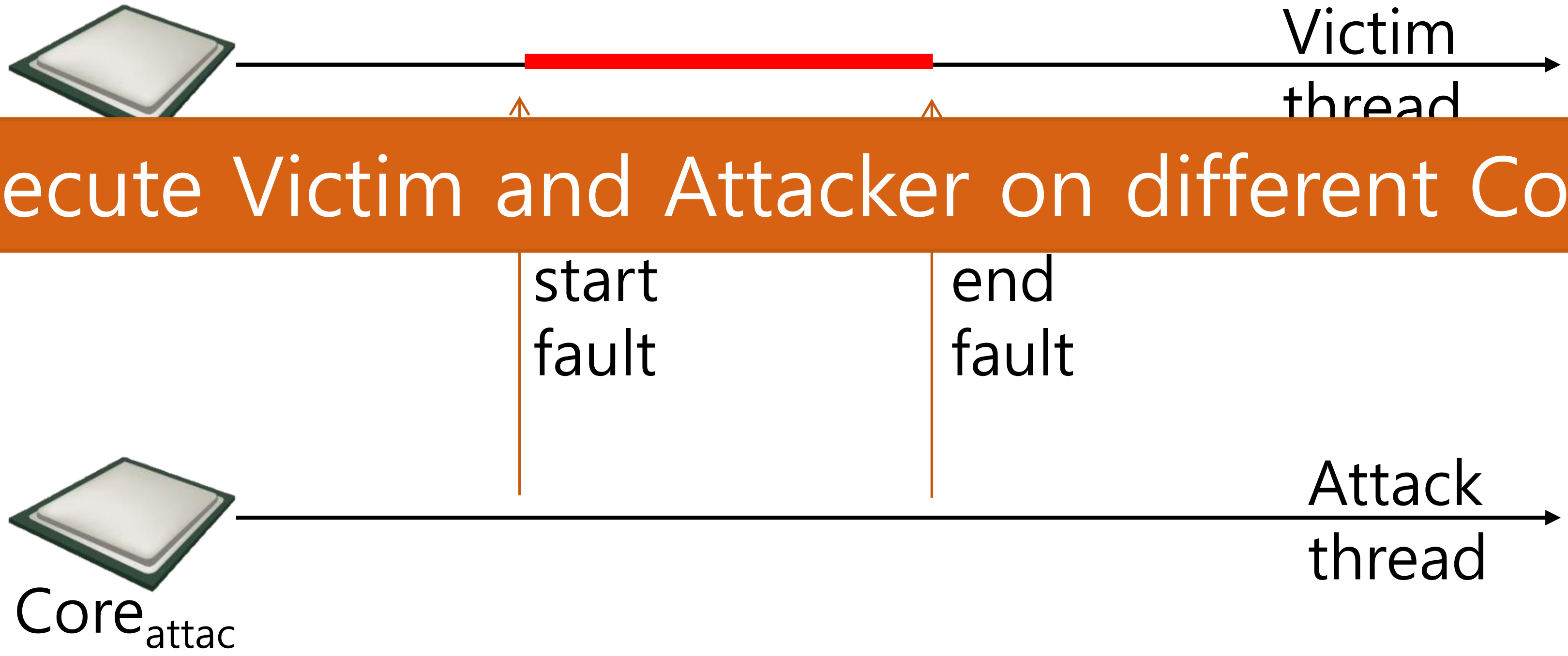
end fault

Attack thread

Core$_{\text{attac}}$

Solutions

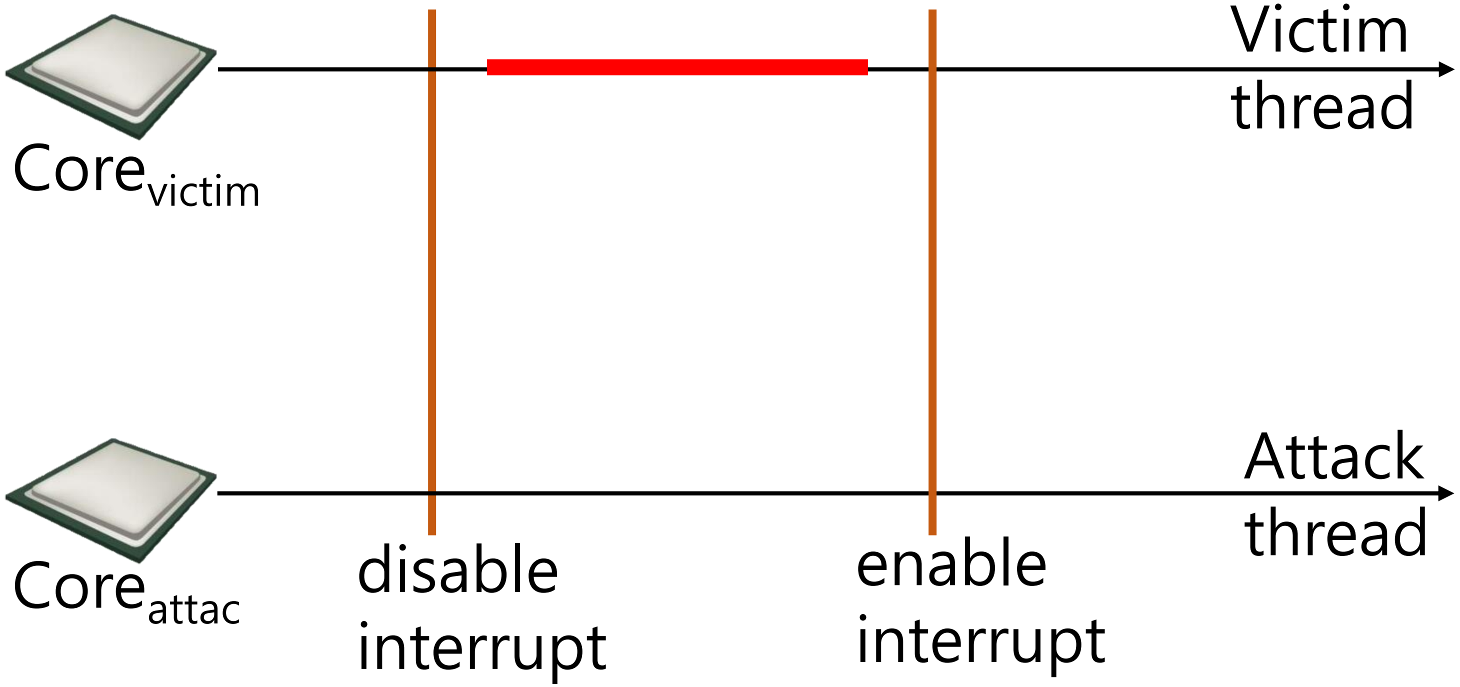
Self-Containment



Execute Victim and Attacker on different Cores

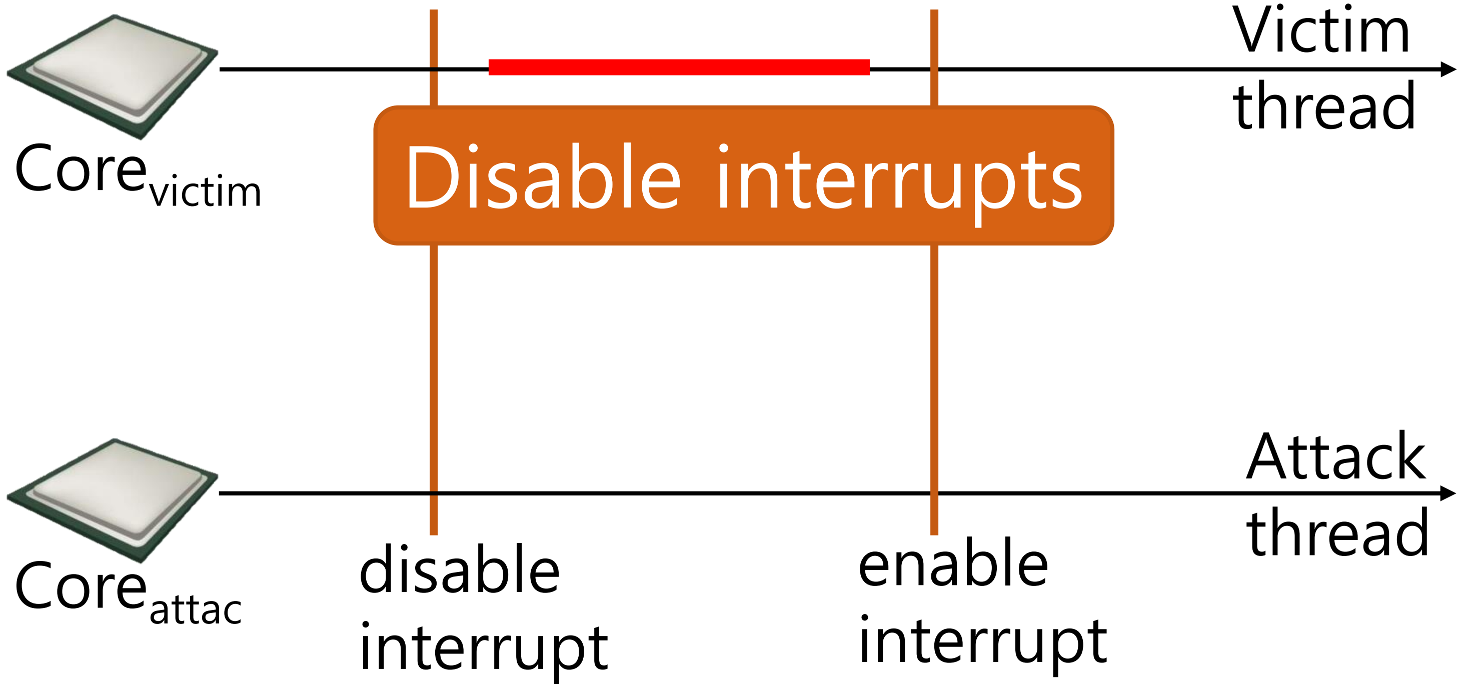
Solutions

Run Attack without Interferences



Solutions

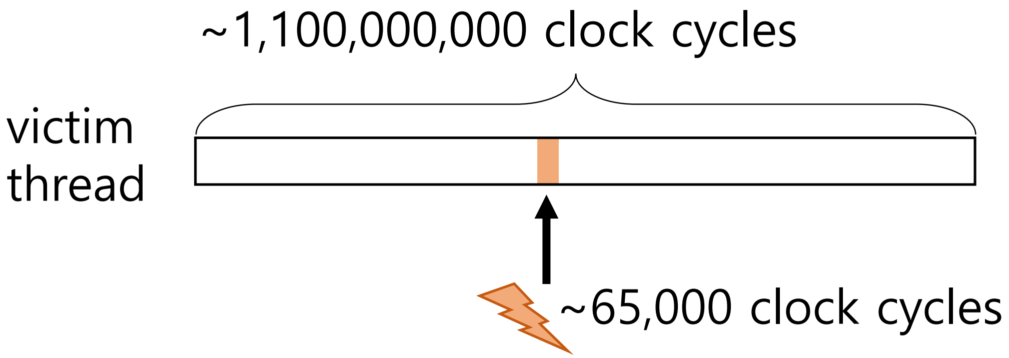
Run Attack without Interferences



Solutions

Timing

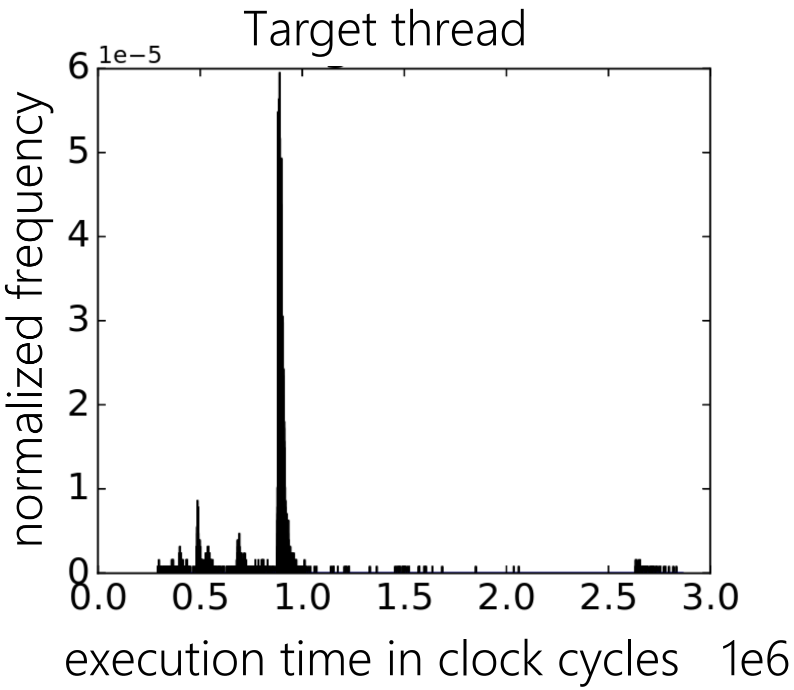
- Need a way to do precise timing



Solutions

Timing

- Use hardware cycle counter to do timing profiling
- Insert no-ops to hit targeted cycle
- Insert anchor times when necessary

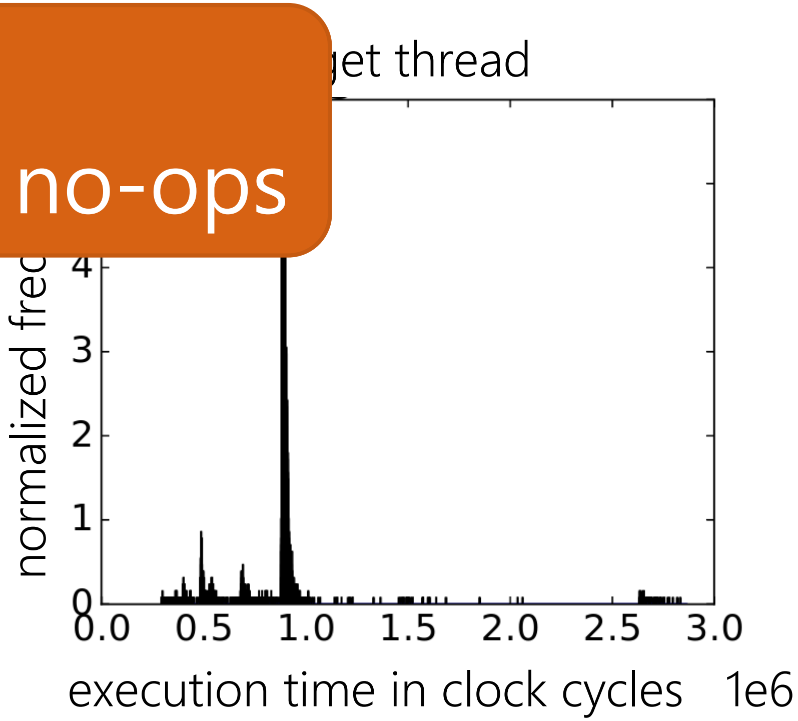


Solutions

Timing

- Use hardware cycle counter to do timing profiling
- Insert no-op
- Insert anchor times when necessary

Do Profiling
Use Anchor Time and no-ops

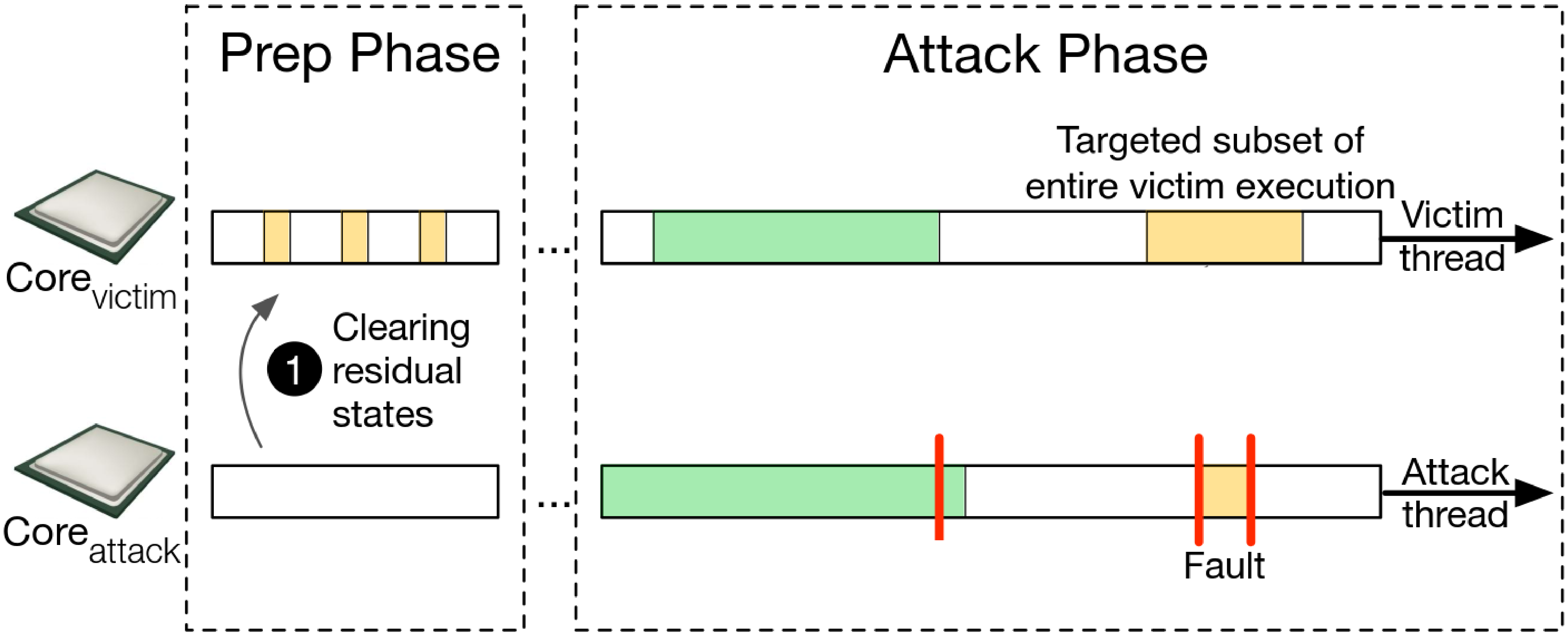


Challenges

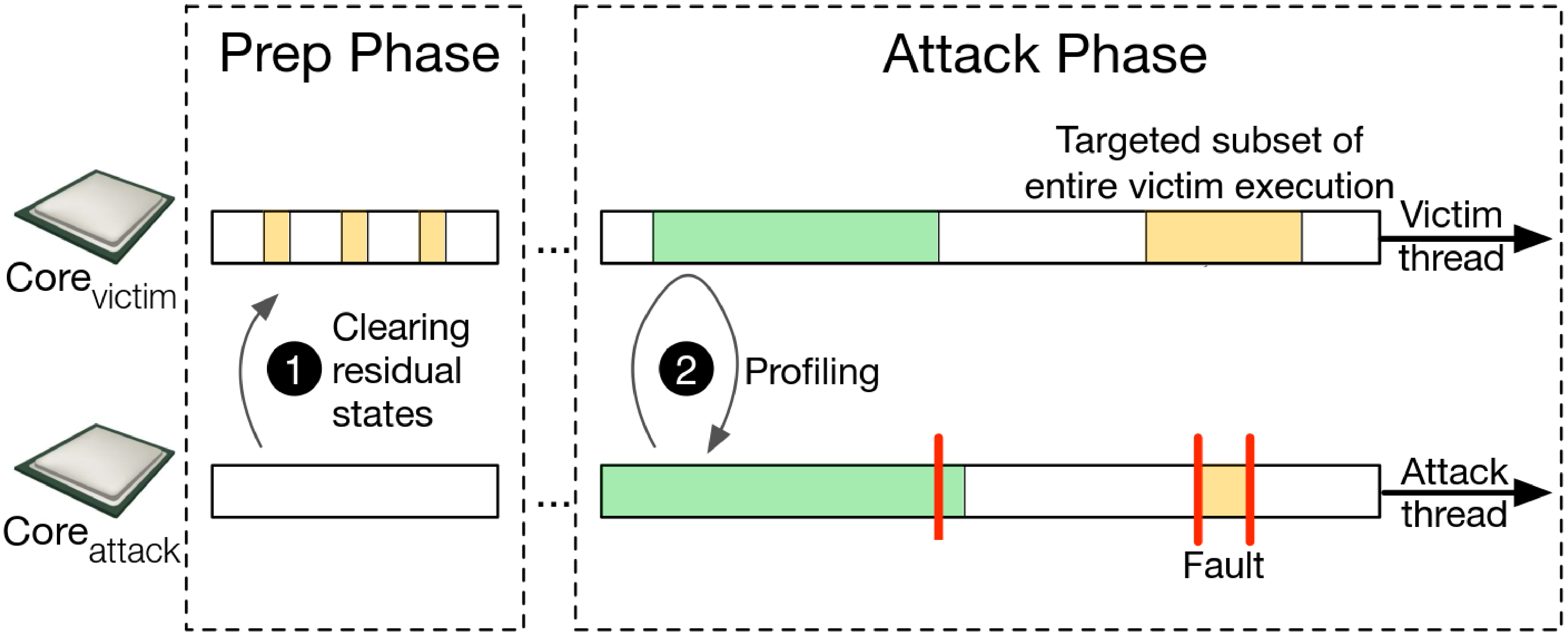
- Voltage and Frequency operating limits?
- Self-containment: how to cause fault for victim without an error in the attacker?
- Can attack run without other things interfering?
- How to time attack correctly?

Mechanisms

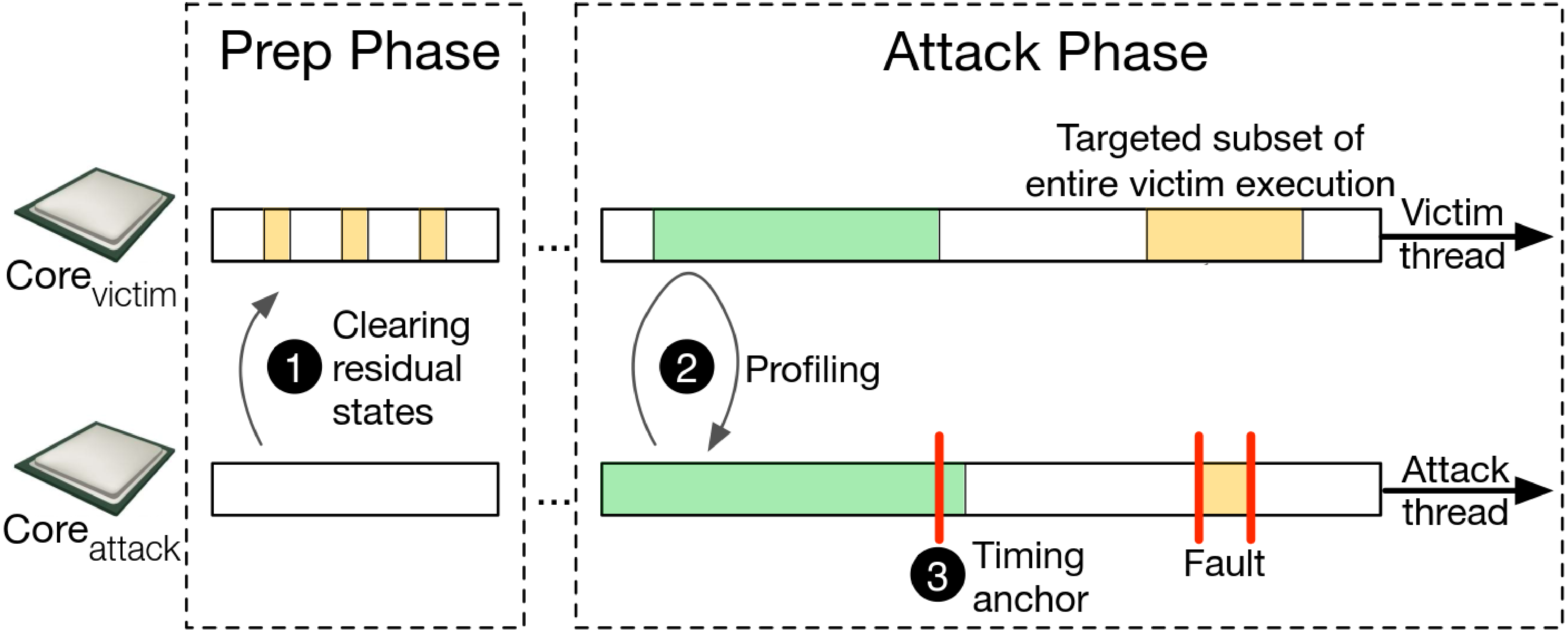
CLKSCREW fault injection setup



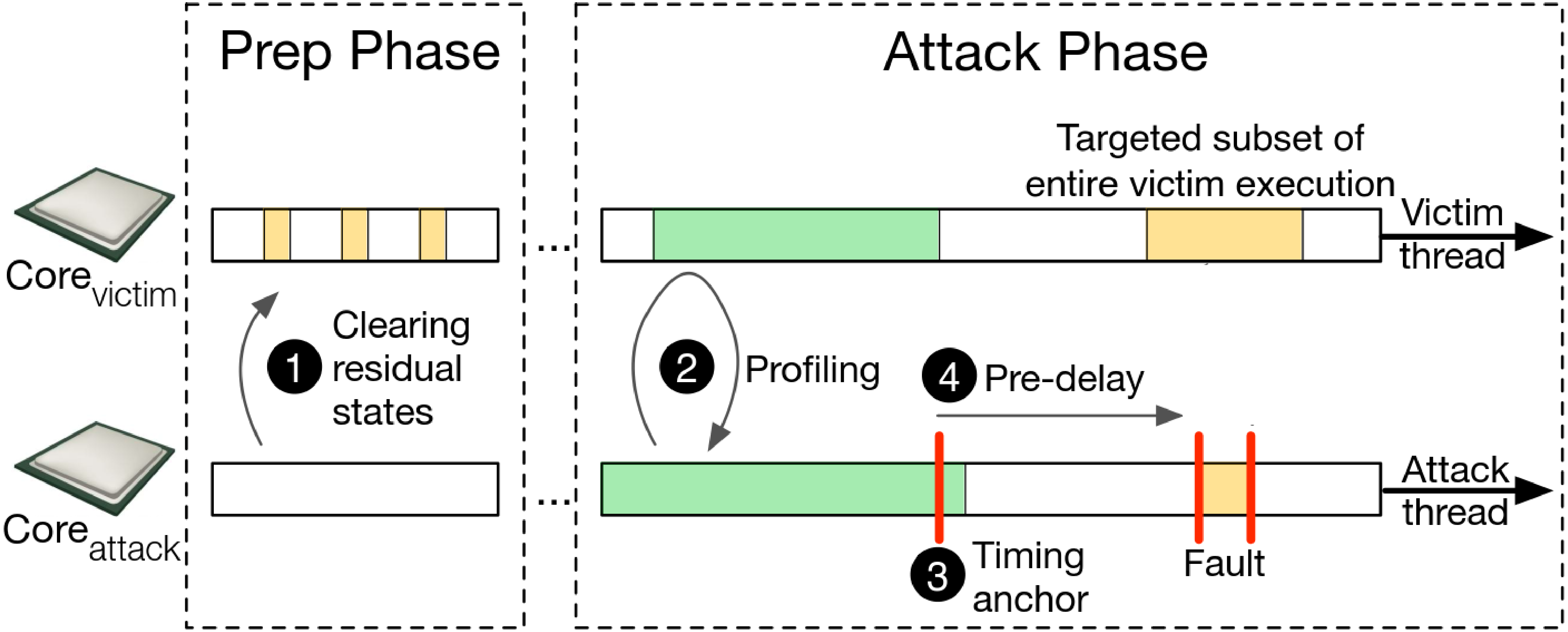
CLKscrew fault injection setup



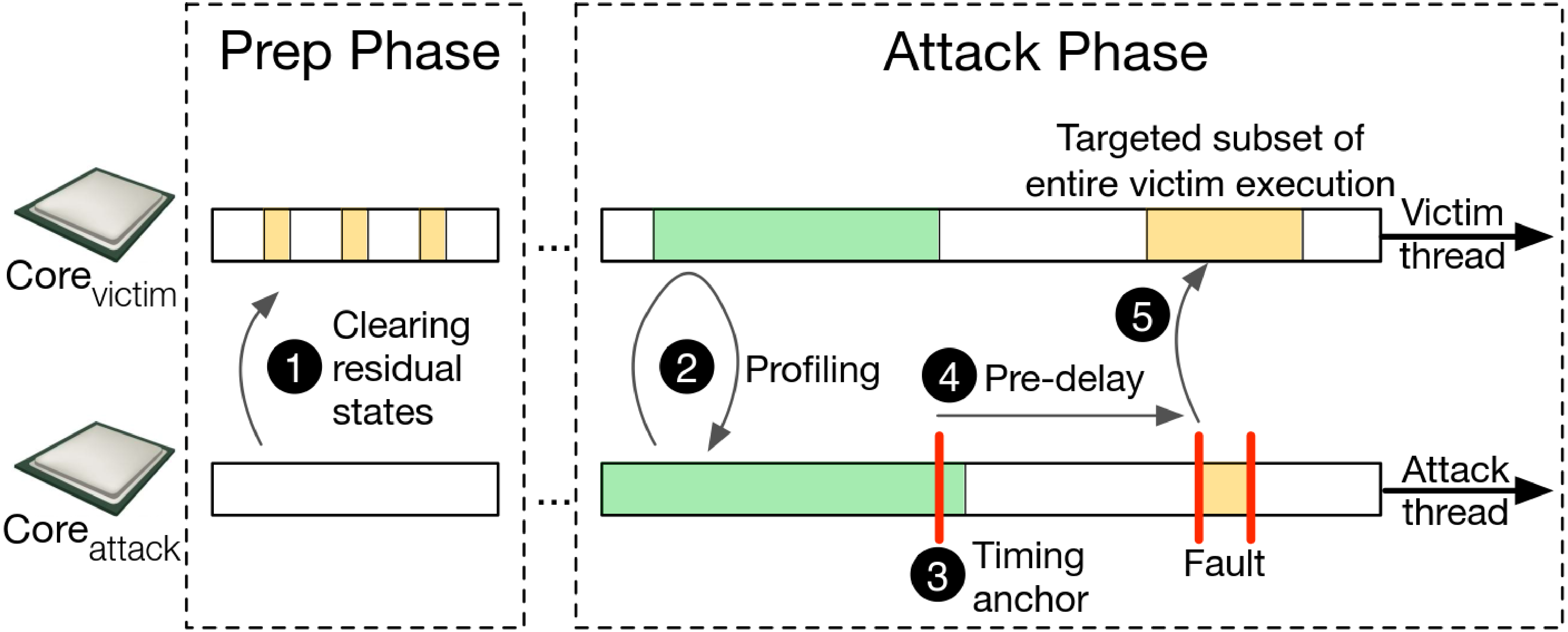
CLKscrew fault injection setup



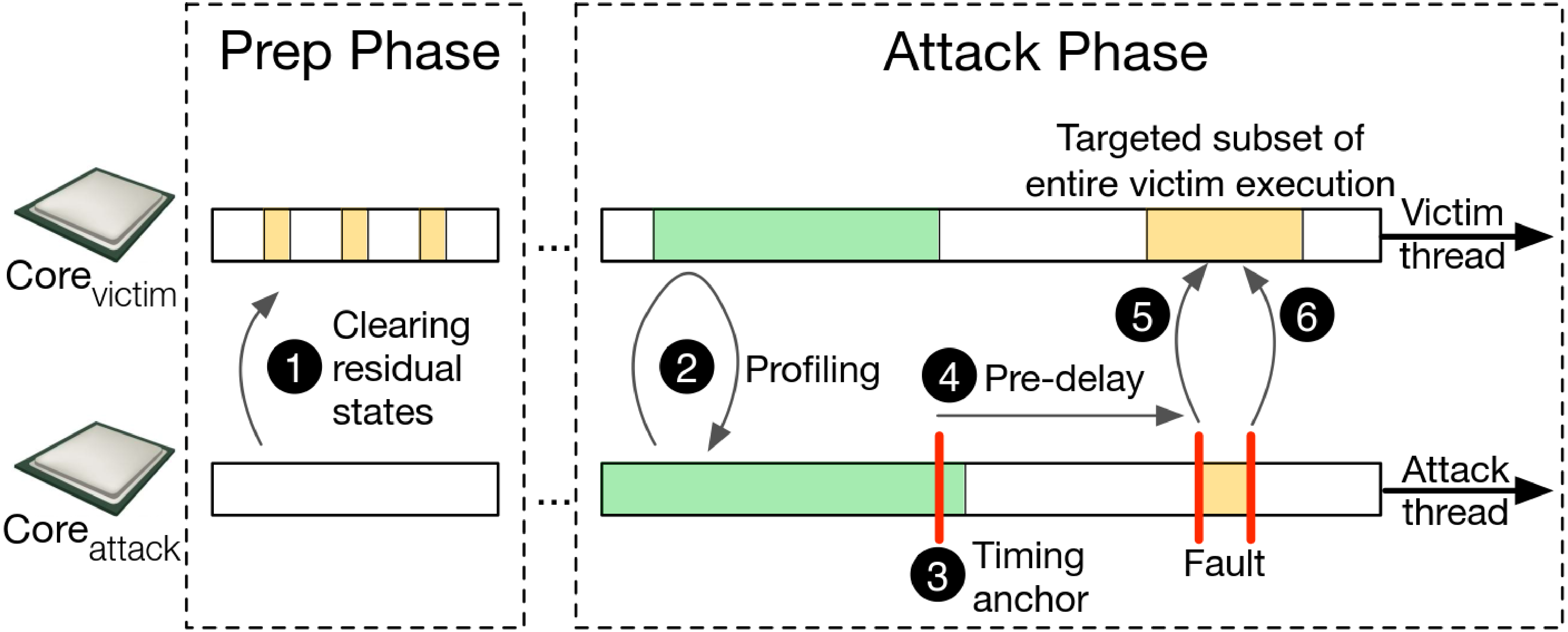
CLKSCREW fault injection setup



CLKscrew fault injection setup

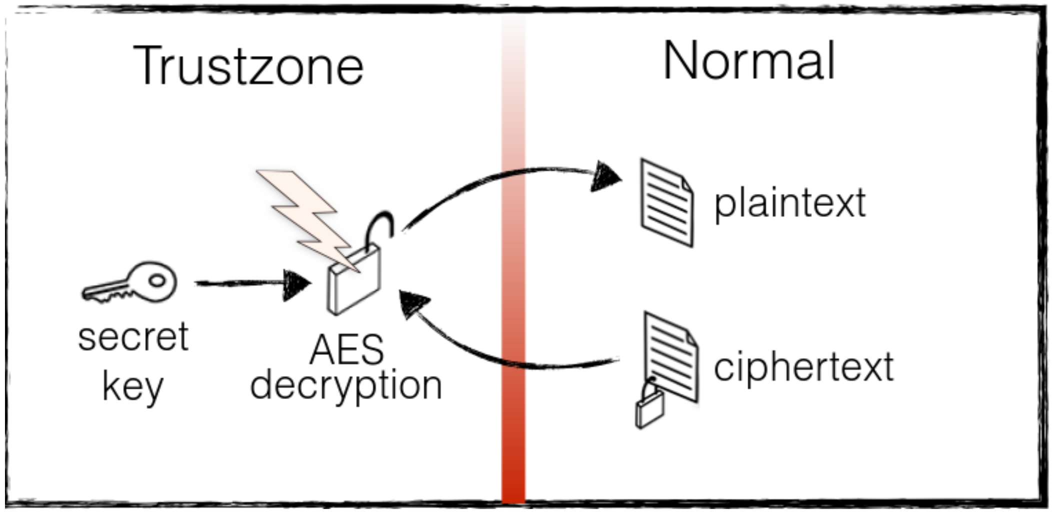


CLKscrew fault injection setup

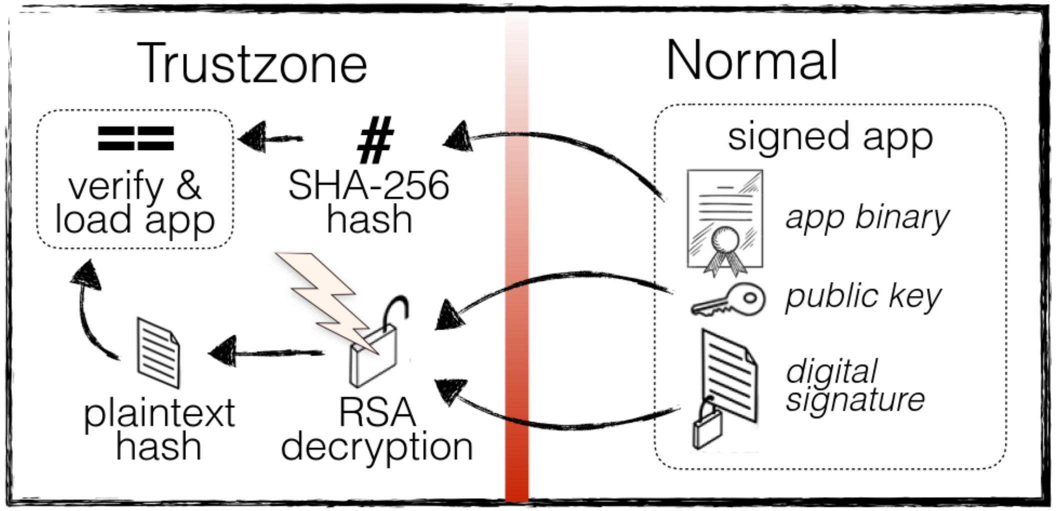


Example Attacks

- TZ Attack #1: Inferring AES Keys



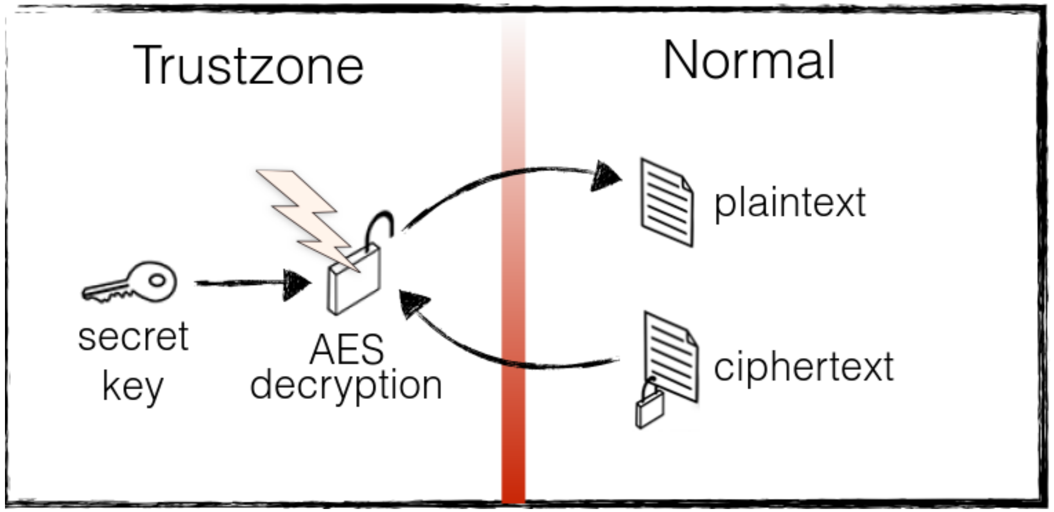
- TZ Attack #2: Loading Self-Signed Apps



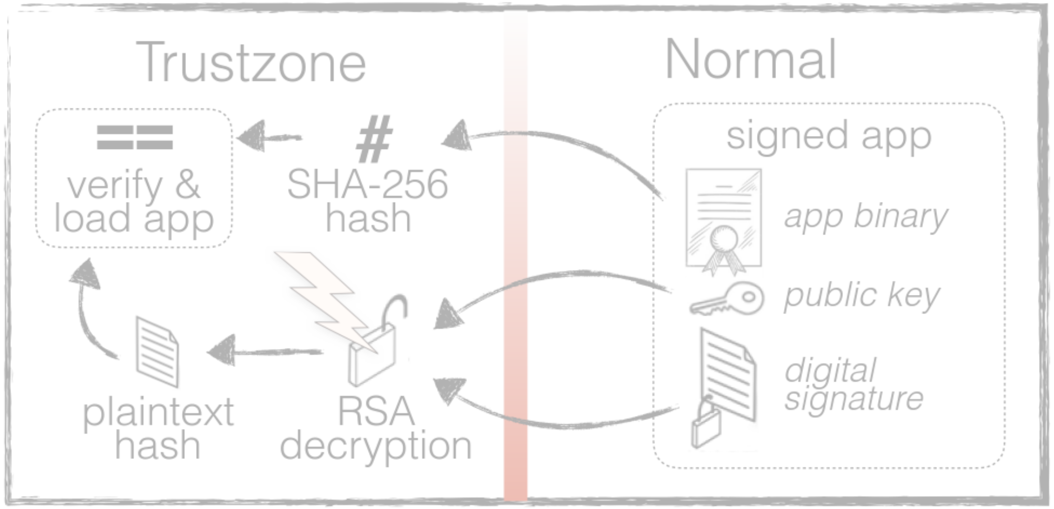
source graphics: CLKSCREW presentation USENIX, Adrian Tang, 2017

Example Attacks

- TZ Attack #1: Inferring AES Keys



- TZ Attack #2: Loading Self-Signed Apps

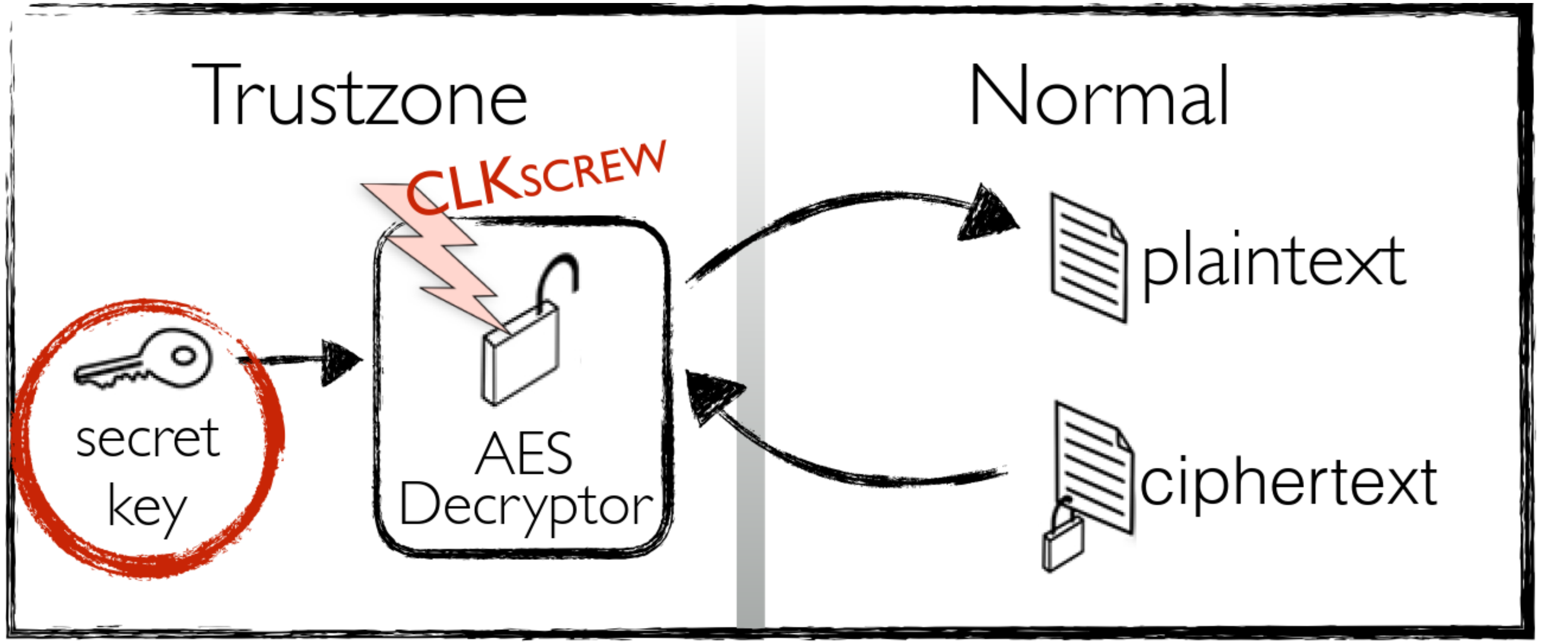


source graphics: CLKSCREW presentation, Adrian Tang, 2017

Key Inference Attack: Threat Model

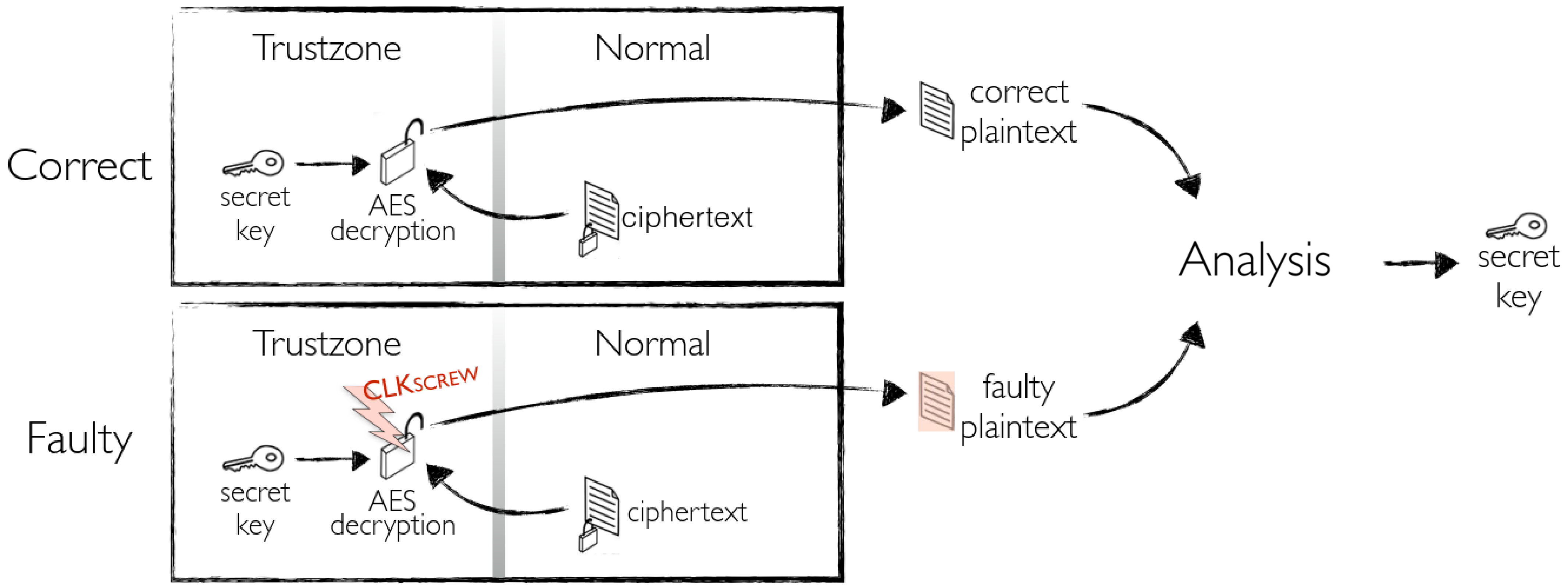
- Victim app: AES decryption app executes in Trustzone
- Attacker's goal: Get secret AES key from outside Trustzone
- Attackers capabilities:
    1. Can repeatedly invoke decryption app
    2. Has software access to hardware regulators

Key Inference Attack: Threat Model



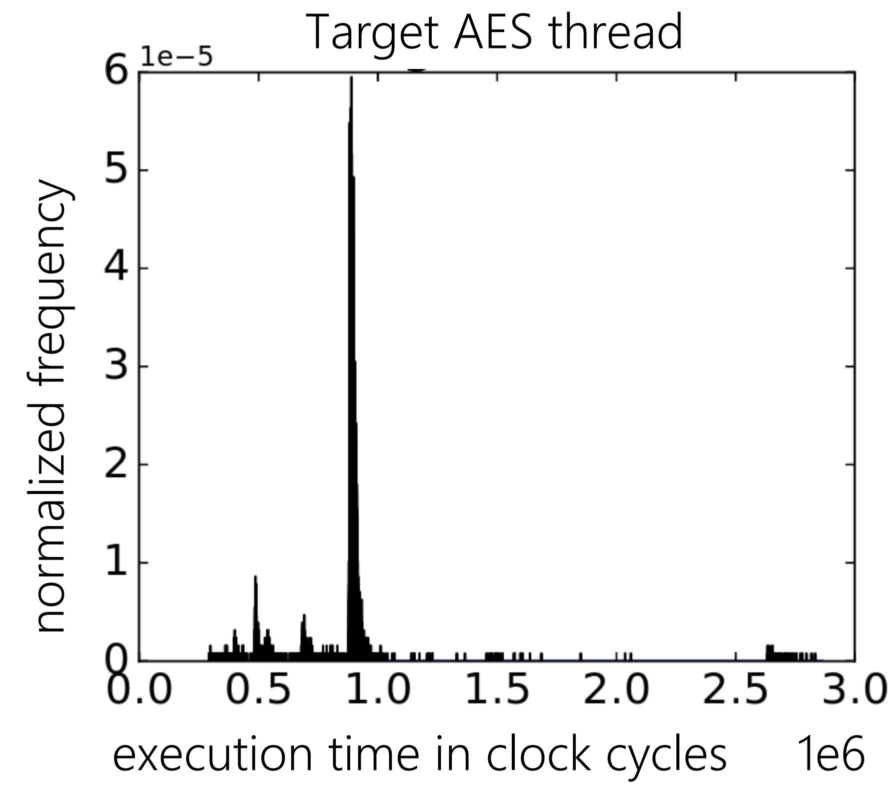
source graphics: CLKSCREW presentation, Adrian Tang, 2017

Key Inference Attack: Overview



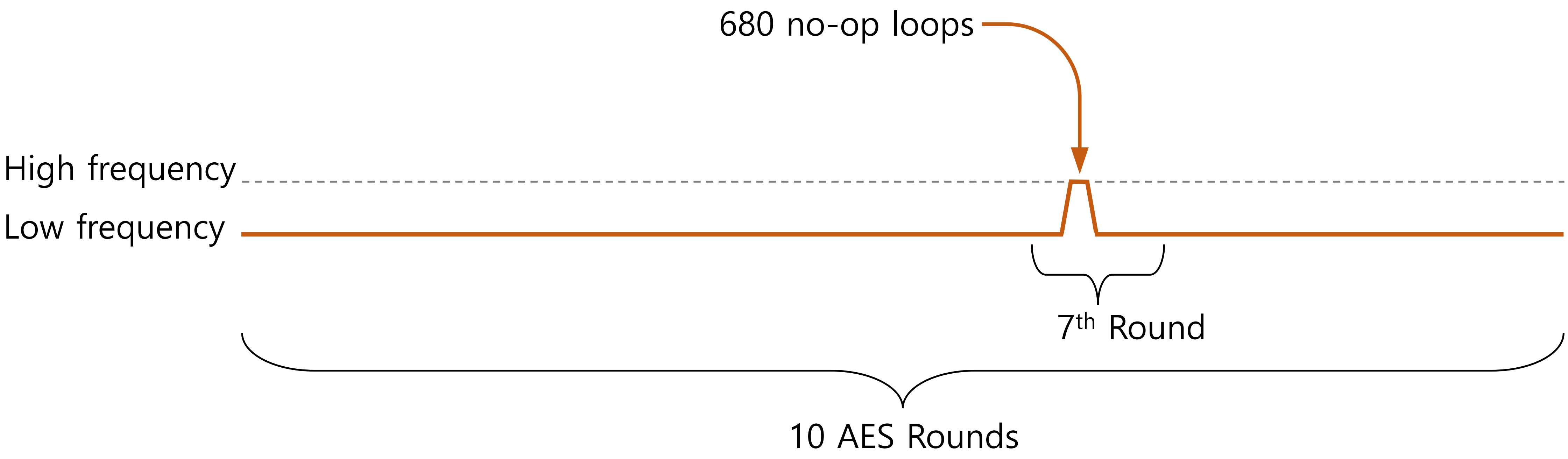
source graphics: CLKSCREW presentation, Adrian Tang, 2017

Key Inference Attack: Timing Profiling



Key Inference Attack: Timing Profiling

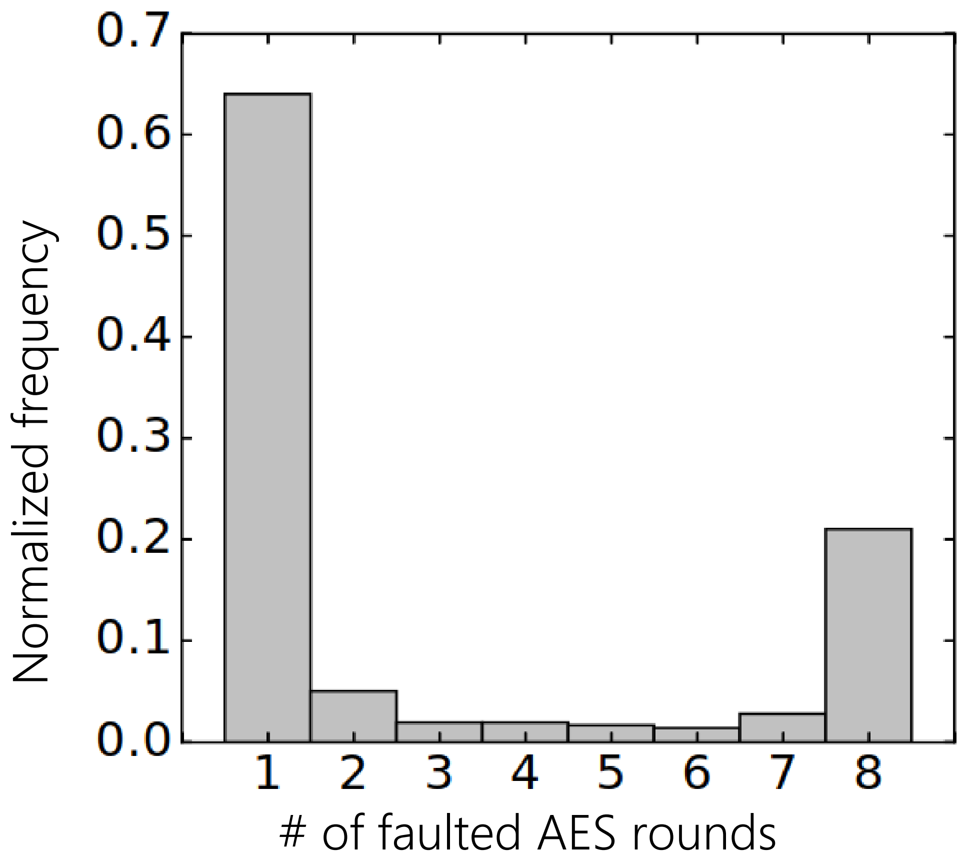
- Induce fault of one byte at 7th AES round



source graphics: CLKSCREW presentation USENIX, Adrian Tang, 2017

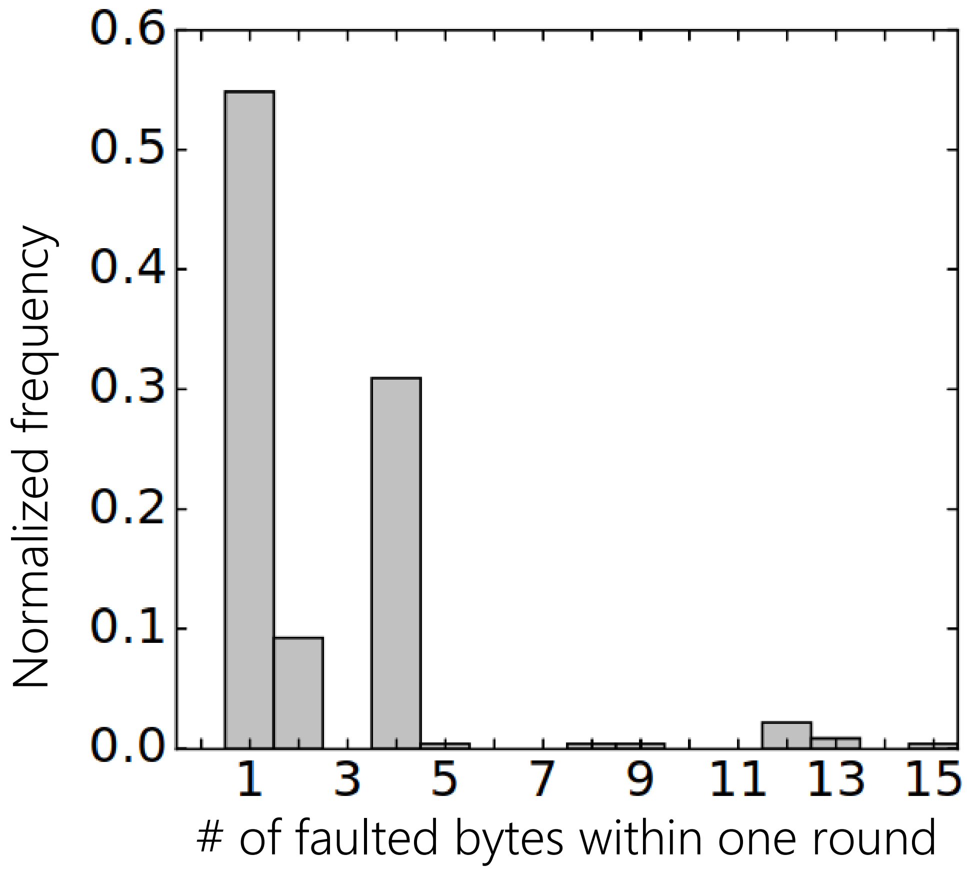
Key Inference Attack: Precision

- Over 60% of generated faults corrupt exactly one AES round



Key Inference Attack: Precision

- Over 60% of generated faults corrupt exactly one AES round
- Of those over 50% corrupt exactly one byte



Summary

Summary

- First security review of DVFS
- DVFS leaves Trustzone vulnerable
- CLKscrew attacks can be timed very precisely
- Can get AES key from outside Trustzone
- Can load untrusted app into Trustzone

Strengths and Weaknesses

Strengths

- First security review of a DVFS
- Managed to do fault attacks purely from software
- Tested two example attacks
  - managed to get the AES key
- only used publicly available knowledge
- Give ideas for possible solutions
- Well written

Weaknesses

- Tested with self written AES decryption app
- Used self written kernel driver to have victim and attacker on different cores.
- Assumed access to hardware regulators
- Tested attacks only on one Nexus 6 device

Takeaways

Takeaways

- New attack surface: Energy management software interface
- Not because of bug but because of fundamental design flaw
- Example attacks on ARM Trustzone
- Energy management designs must take security into consideration

Discussion

Discussion

- Ideas on possible solutions?
    - Hardware?
    - Software?
- What else could be done by exploiting DVFS
    - can you think of specific attacks?

Discussion

Blacklist Core: Machine-Learning Based Dynamic Operating-Performance-Point Blacklisting for Mitigating Power-Management Security Attacks

Sheng Zhang, Adrian Tang, Zhewei Jiang, Simha Sethumadhavan, Mingoo Seok,
Columbia University, 2018

Discussion

- Ideas on possible solutions?
  - Hardware?
  - Software?
- What else could be done by exploiting DVFS?
  - can you think of specific attacks?

Discussion

- How widely spread is this energy management issue?
- How important will this be for the future?
  - will it be considered enough? does it have to?
- General thoughts on the paper?
  - Additional strength, weaknesses, ideas?