CLKSCREW: Exposing the Perils of Security-Oblivious Energy Management (August 2017)

Adrian Tang, Simha Sethumadhavan, Salvatore Stolfo

(Columbia University)

Presented at 26th USENIX Security Symposium (Vancouver, Canada)

Presented by Dominic Weibel

Problem & Goal

Problem & Goal

- Security review of *Dynamic Voltage and Frequency Scaling* (DVFS)
- Problem:
  - Induce errors in trusted environments *(by stretching frequency / voltage beyond operational limits)*
- Goal:
  - Attack trustzone code execution *(via software control of DVFS)*
- Result:
  - Two practical trustzone attacks
  - Implemented on *Nexus 6*

Nexus 6: http://www.extremetech.com/wp-content/uploads/2014/10/nexus-6-product-photo.jpg

Background

- DVFS
- ARM TRUSTZONE
- OVERCLOCKING & UNDERVOLTING

Energy management

- Essential and complex
- Reduces cost, increases battery life and improves portability
- Most systems allow software control of energy management

DVFS

ARM TRUSTZONE

OVERCLOCKING & UNDERVOLTING

Energy consumption

operating frequency ✖ voltage = power

DVFS

ARM TRUSTZONE

OVERCLOCKING & UNDERVOLTING

Dynamic Voltage & Frequency Scaling (DVFS)

- Omnipresent in almost all commodity devices
- Regulates frequency and voltage according to runtime demands
- Hardware and software support

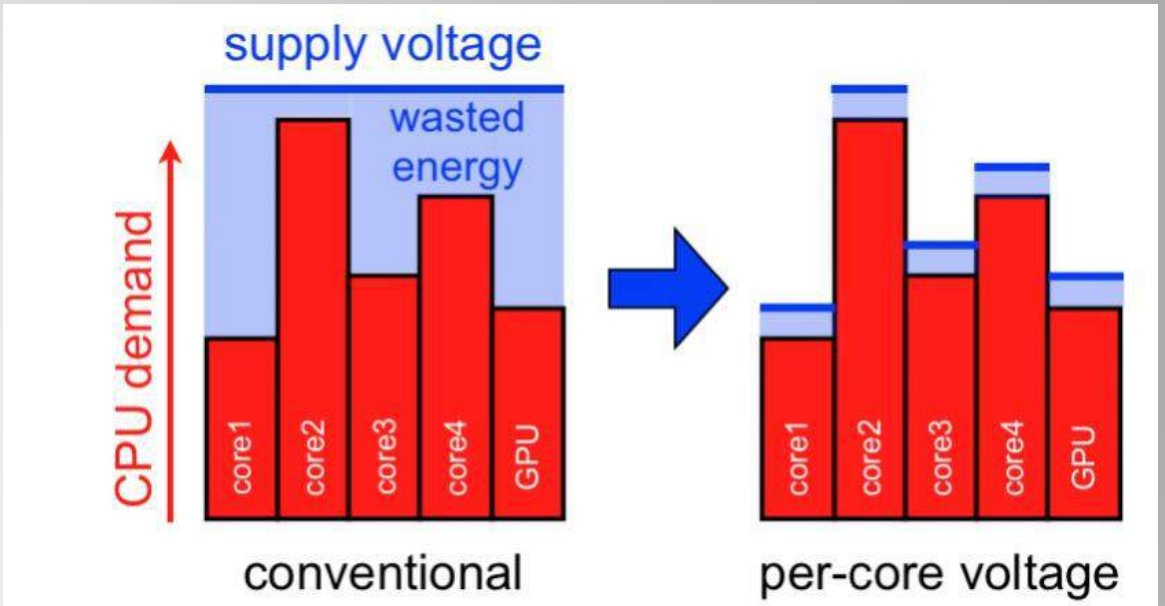


Figure 1. Per-core DVFS can reduce wasted energy due to shared supply voltages.

✓ DVFS

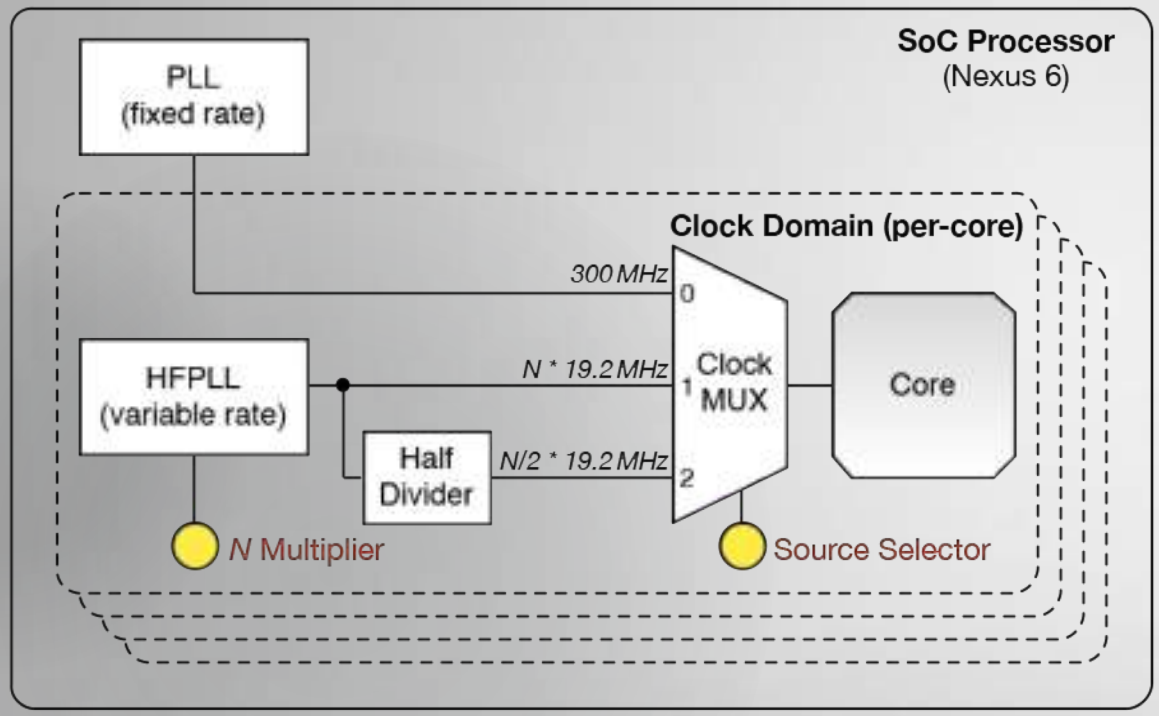
ARM TRUSTZONE

OVERCLOCKING & UNDERVOLTING

Figure 1: http://www.semiwiki.com/forum/content/694-reducing-soc-power-consumption-using-integrated-voltage-regulators.html

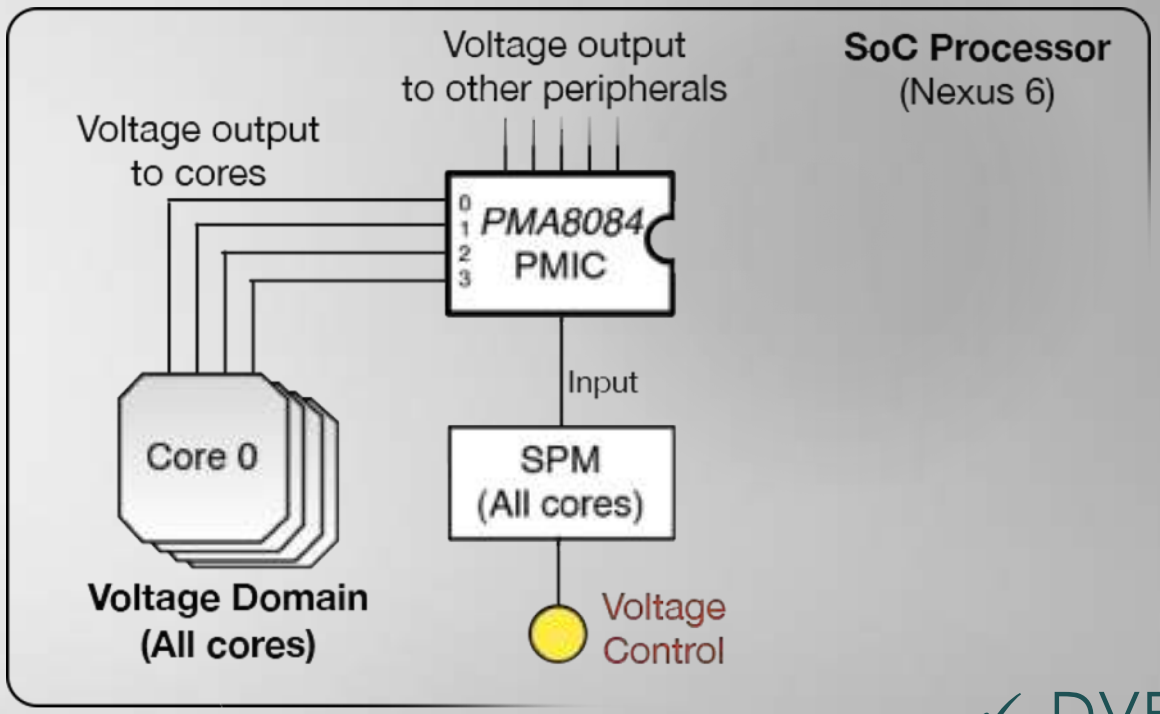
Frequency Regulator

- Provide clock source to each core



Voltage Regulator

- Supply power to various components



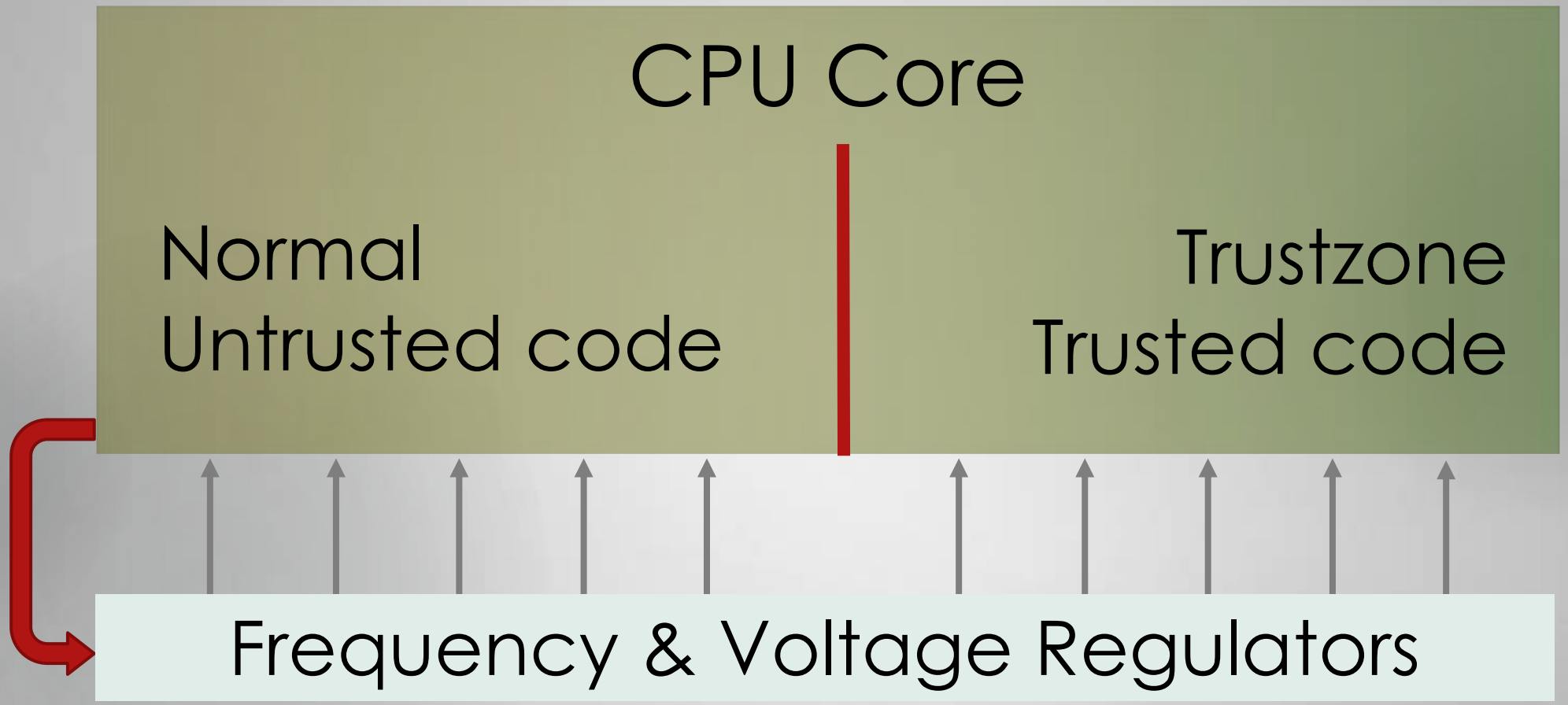
✓ DVFS

ARM TRUSTZONE

OVERCLOCKING & UNDERVOLTING

=> Security is rarely considered

ARM Trustzone & Intel SGX

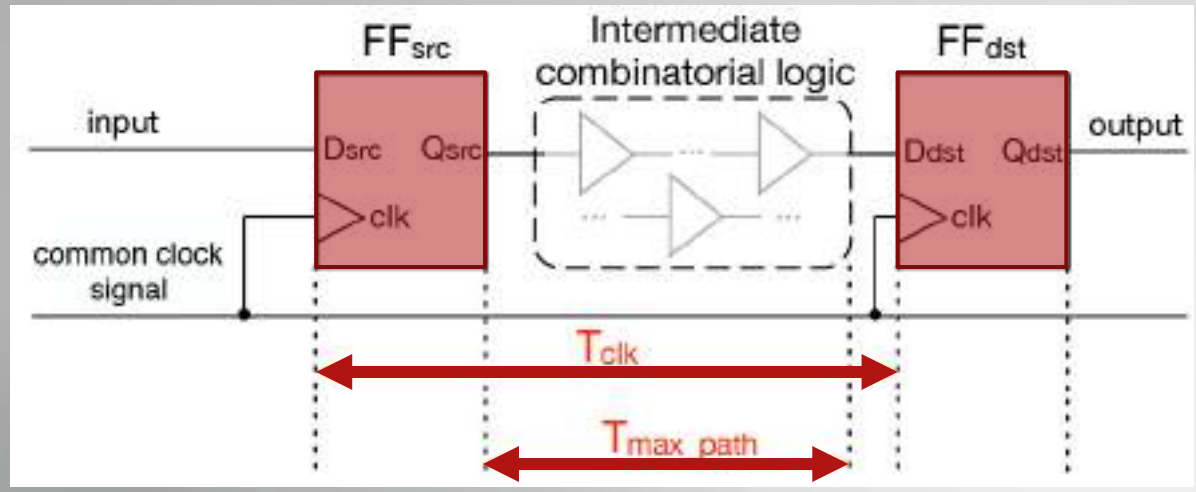


✓ DVFS

✓ ARM TRUSTZONE

OVERCLOCKING & UNDERVOLTING

Overclocking & Undervolting



- **FF_{src} / FF_{dst}** : Flip-Flops
- **T_{clk}** : clock cycle period
- **T_{max_path}** : Max time for signal to propagate from FF_{src} to FF_{dst}

- *Overclocking*: reduce **T_{clk}** cycle period

=> Less time to propagate

- *Undervolting*: increase **T_{max_path}**

Wrong Data at output

✓ DVFS

✓ ARM TRUSTZONE

✓ OVERCLOCKING & UNDERVOLTING

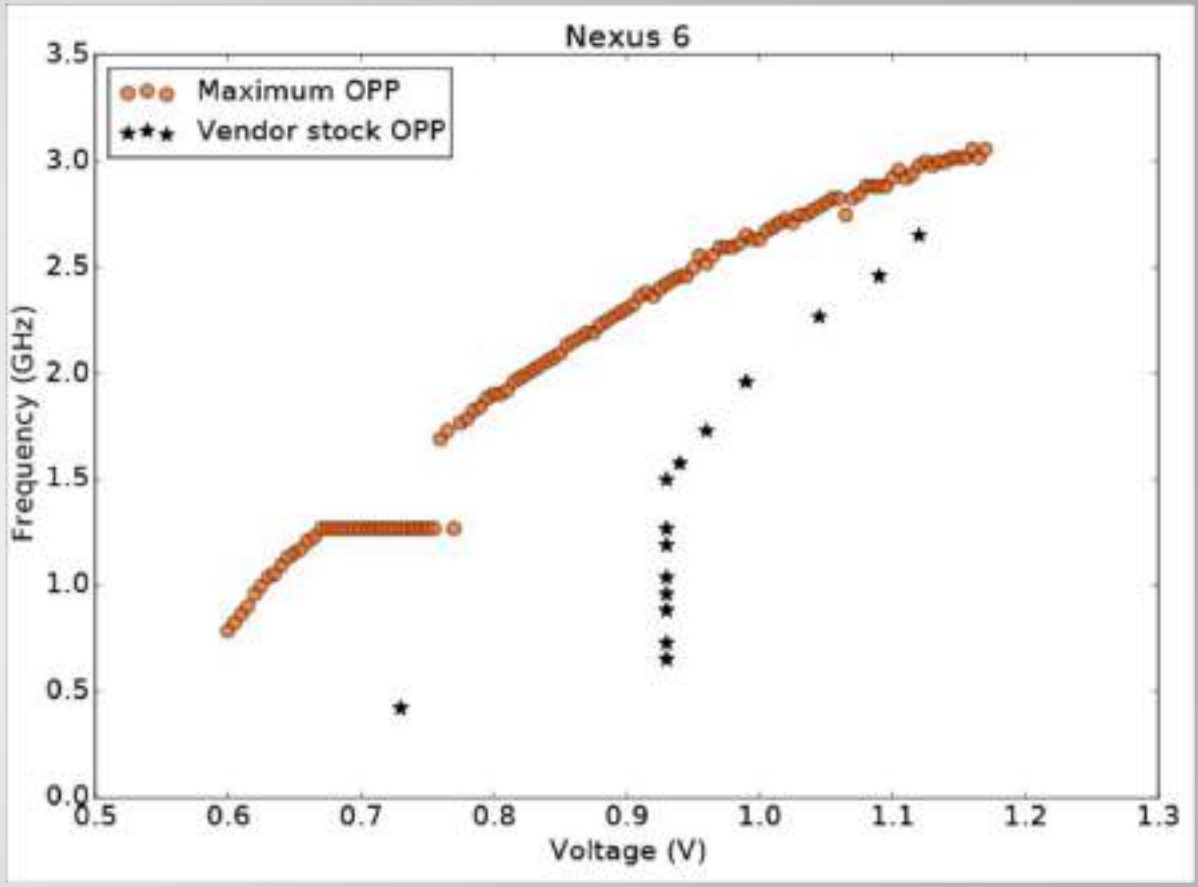
Challenges & Key Insights

Challenges

- Regulator operating limits

=> There are no safeguard limits in hardware

Key Insights



Challenges

- Regulator operating limits
- Self containment within same device

Key Insights

- Use core pinning

Challenges

- Regulator operating limits
- Self containment within same device
- Noisy complex OS environment

Key Insights

- Disable interrupts

Challenges

- Regulator operating limits
- Self containment within same device
- Noisy complex OS environment
- Precise profiling

Key Insights

- Use hardware cycle counter to track execution duration

Challenges

- Regulator operating limits
- Self containment within same device
- Noisy complex OS environment
- Precise profiling
- Fine grained timing

Key Insights

- Use no-op loop

Possible Exploits

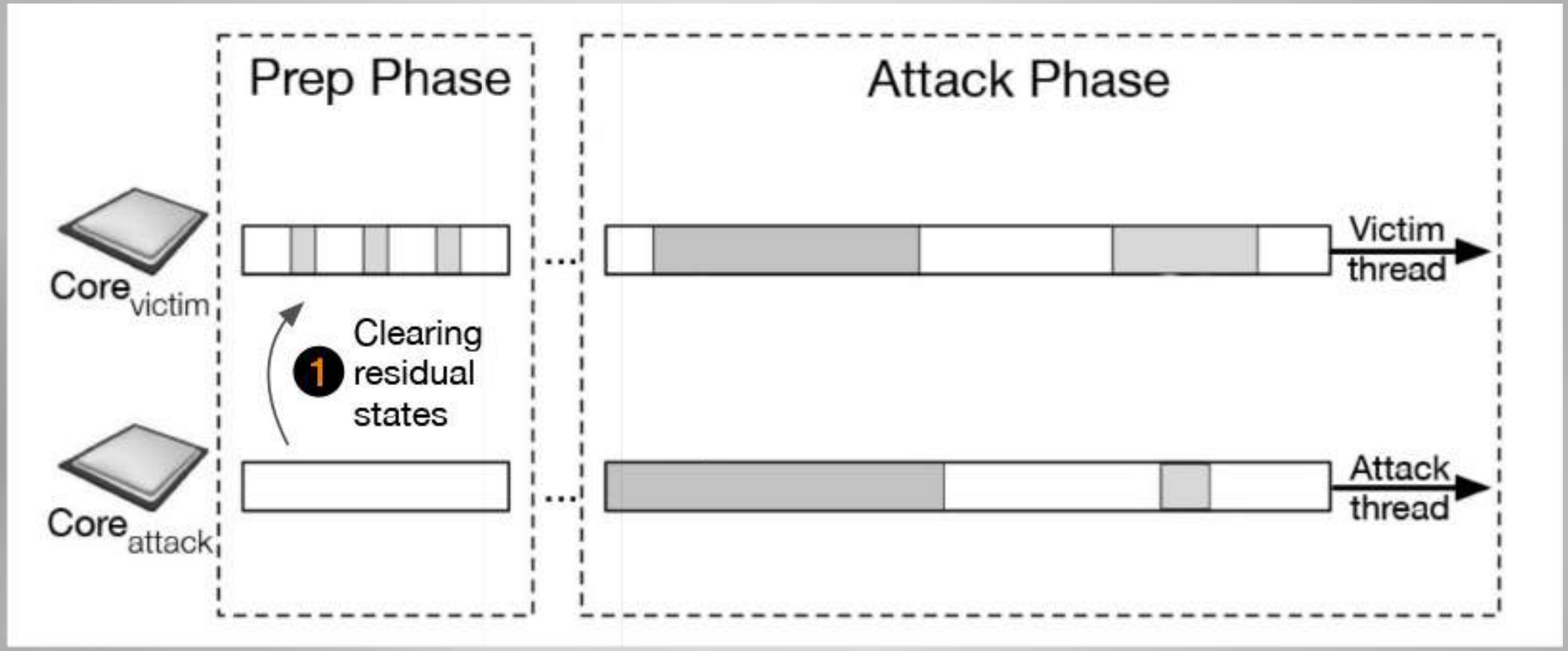
- Attack integrity / availability
- Break (safety) constraints
- Encryption / Decryption
  - Induce faults to get additional knowledge

Key Mechanism of the CLKScrew Attack

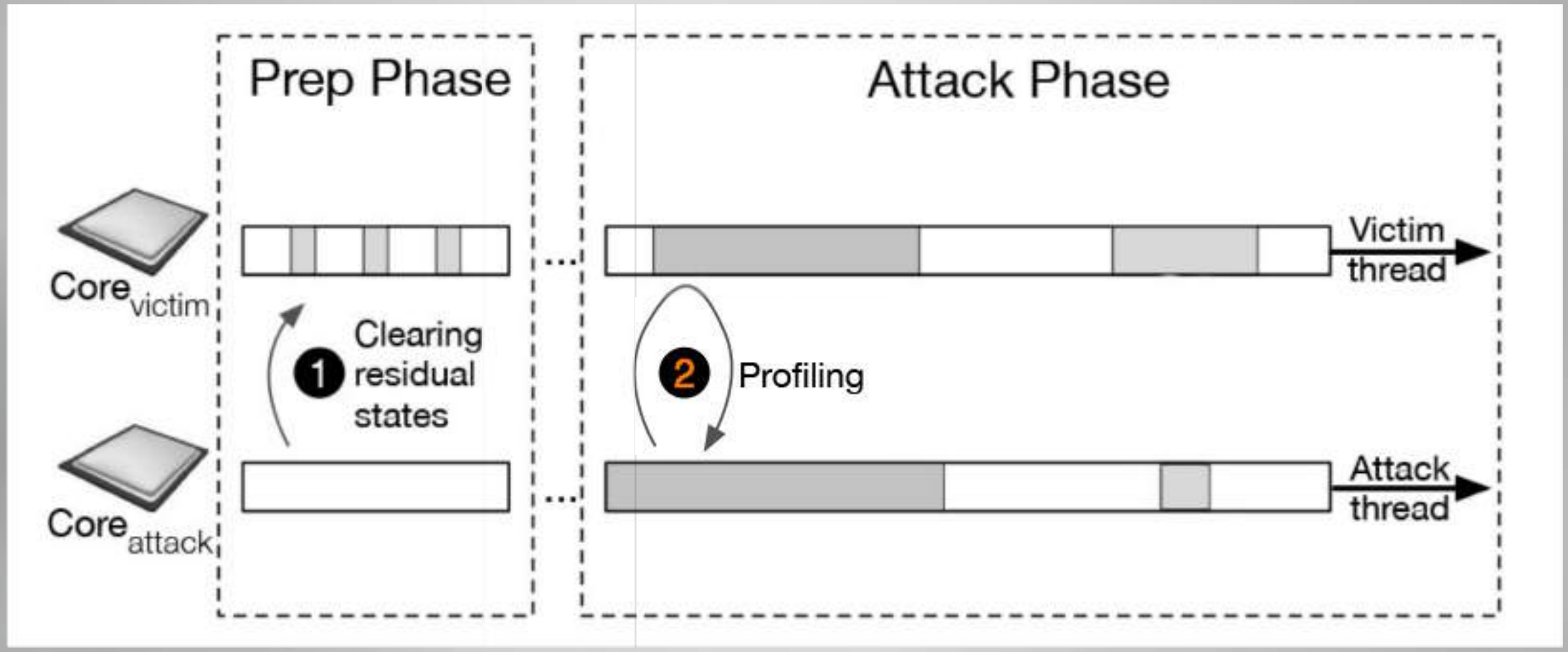
Key mechanism

Key mechanism

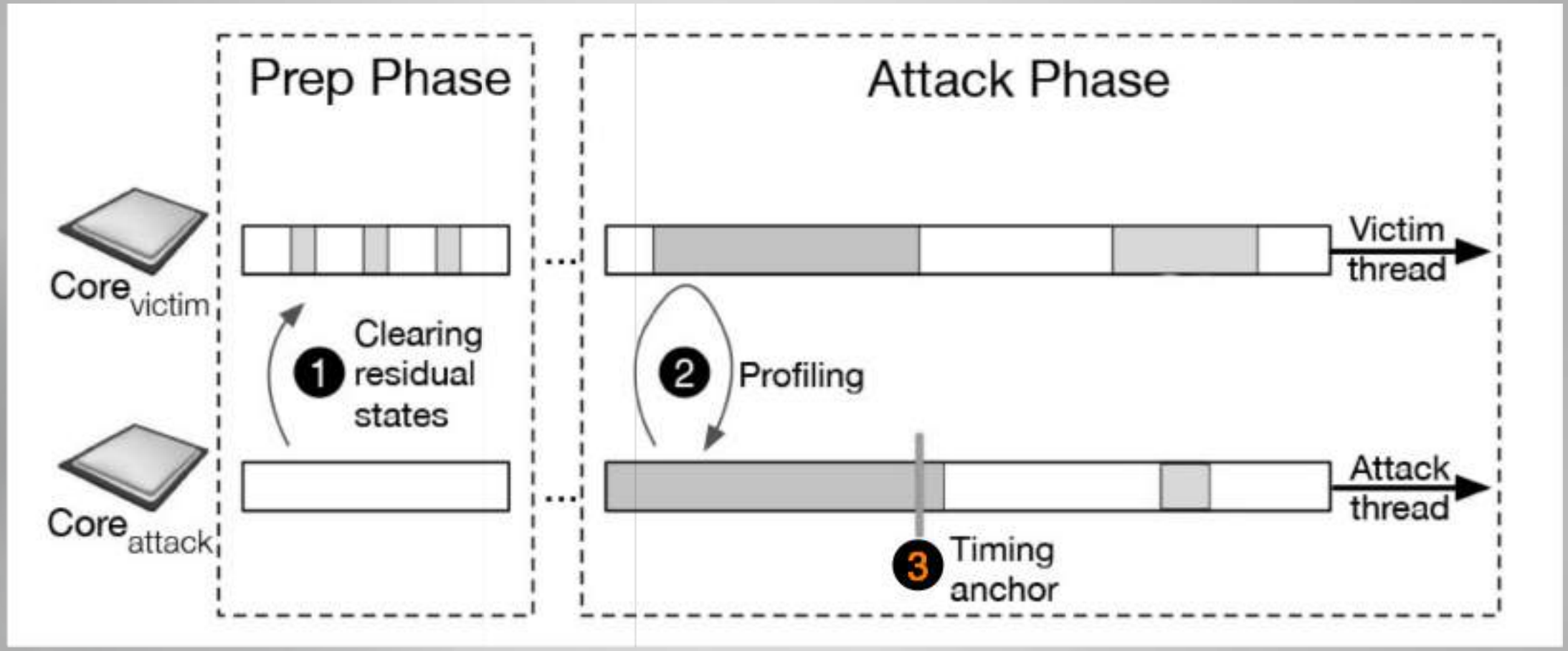
Key mechanism



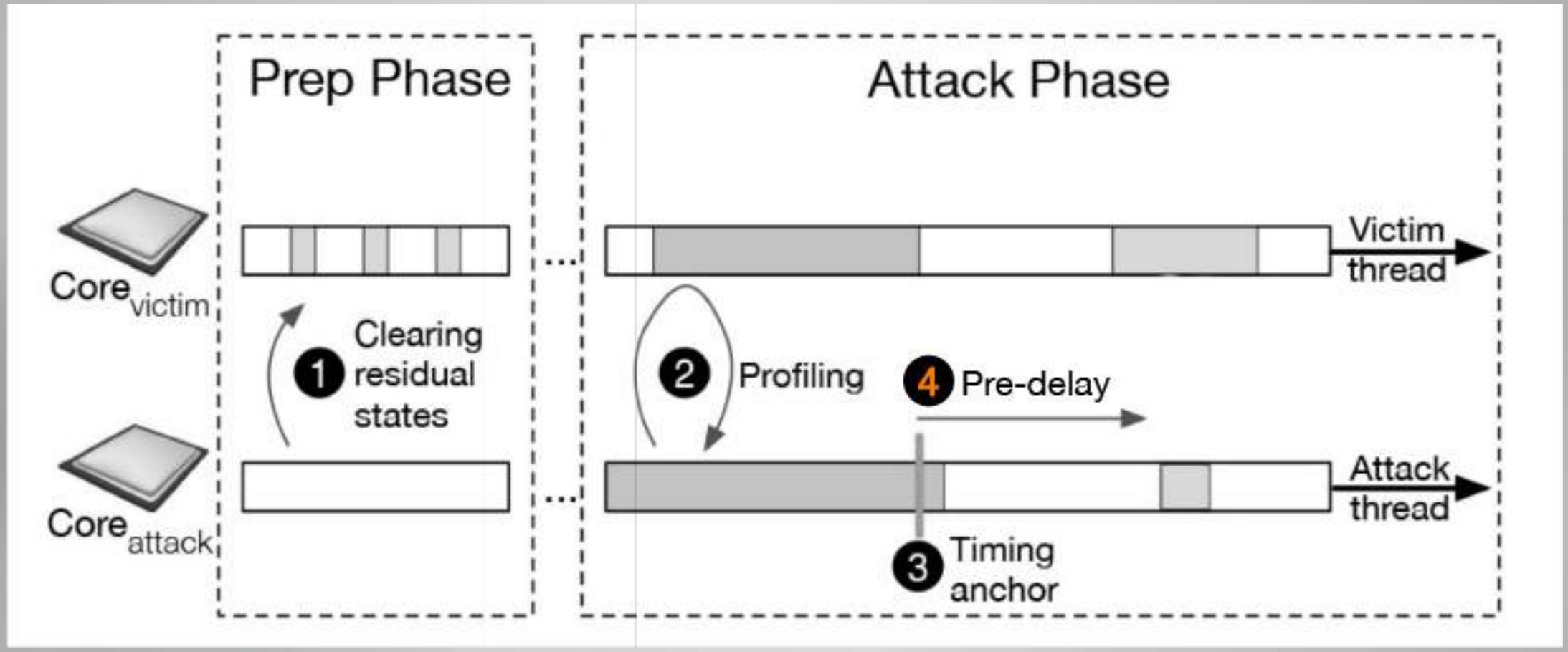
Key mechanism



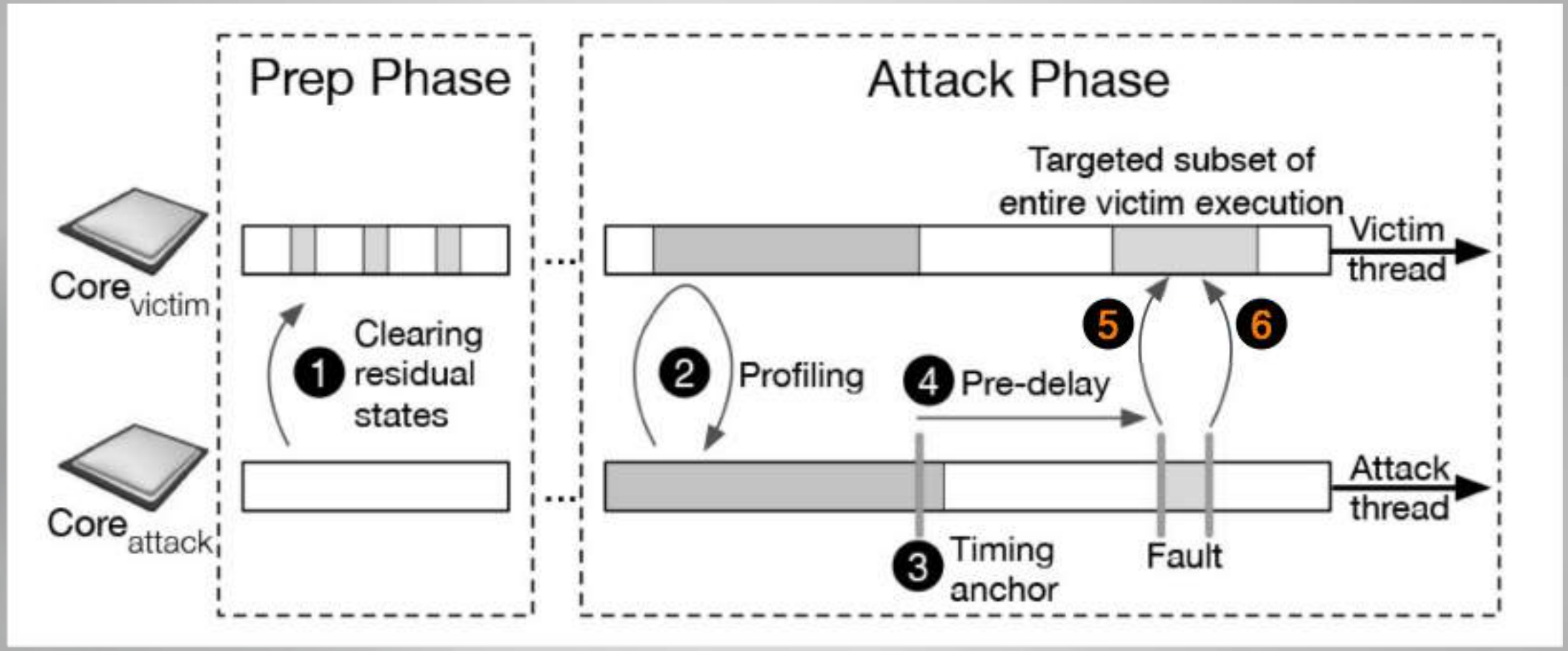
Key mechanism



Key mechanism



Key mechanism



Key Results: Methodology & Evaluation

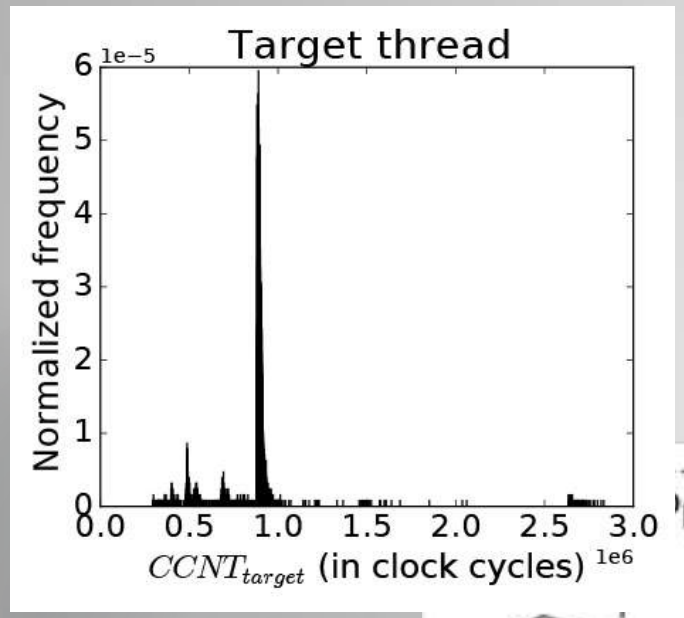
Trustzone Attacks

- TZ Attack #1: Inferring AES Keys
- TZ Attack #2: Loading Self Signed Apps

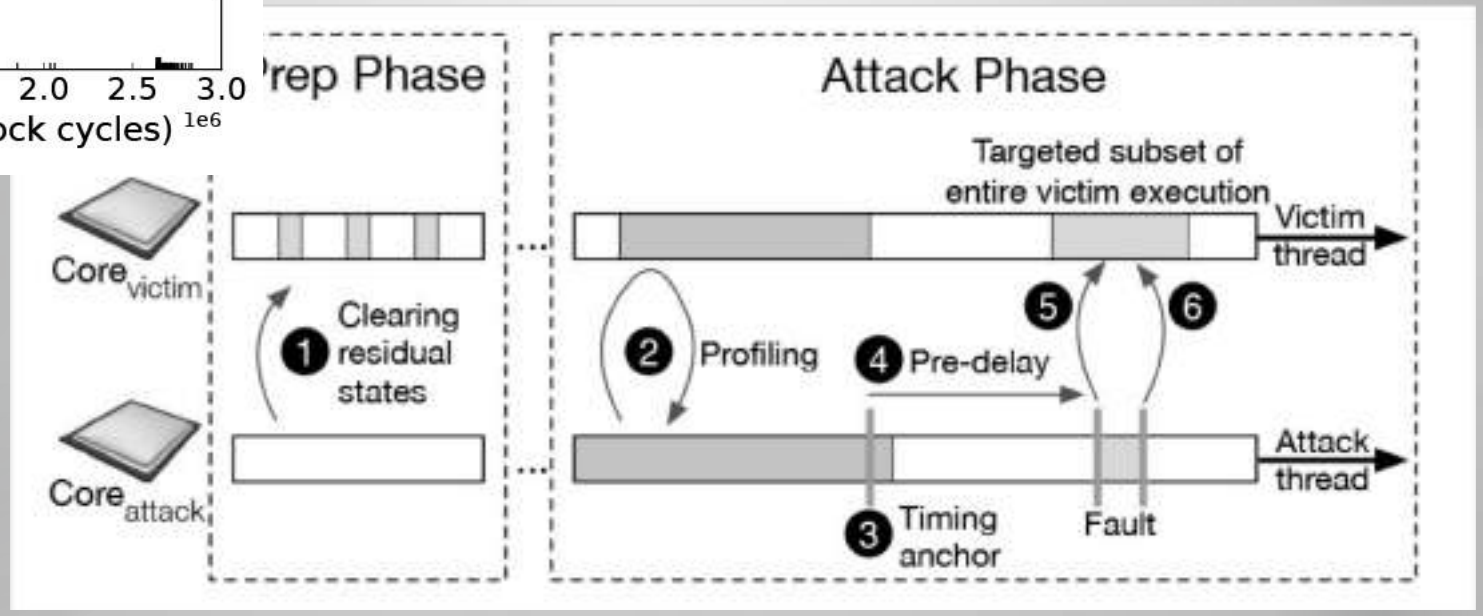
Attack Model – TZ Attack #1

- Trustzone App provisions **AES keys** (and stores them within the Trustzone)
- Attacker can
  - repeatadly invoke the Trustzone App
  - control *frequency and voltage* regulators via memory mapped control registers
- Attackers Goal: infer the stored AES keys

TZ Attack #1: Inferring AES Keys



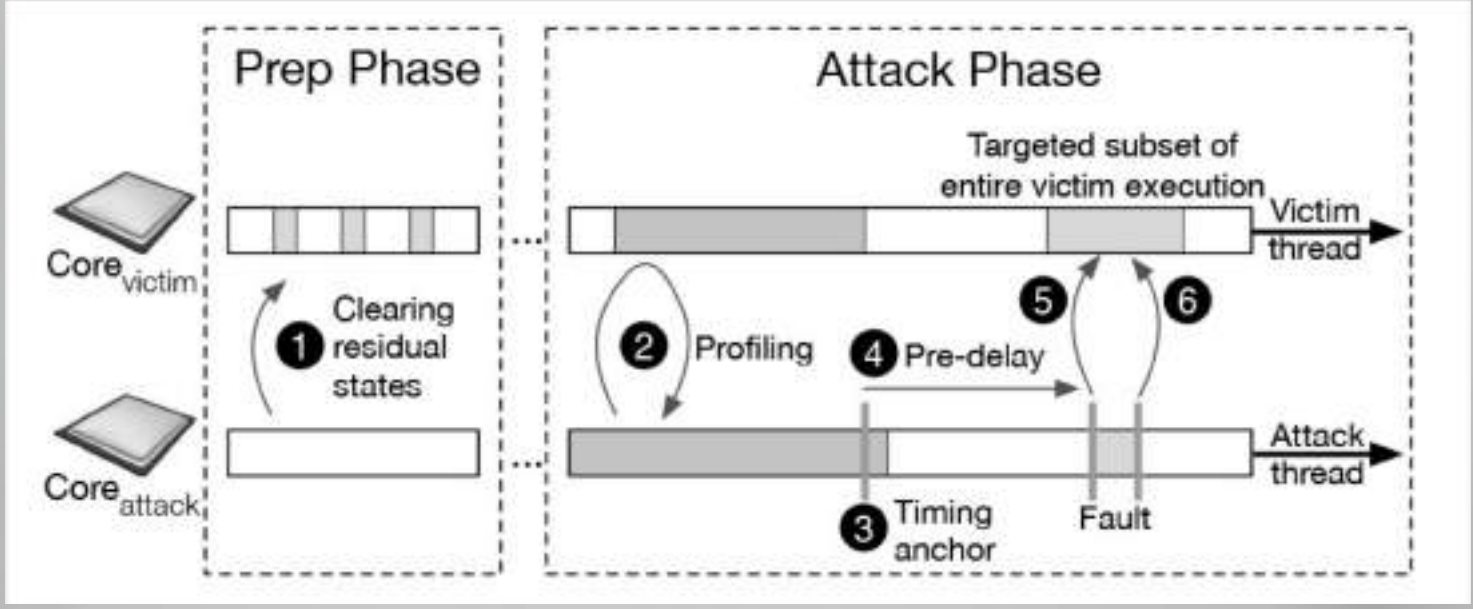
- Search for parameters: F_{freq_hi} and F_{dur}
- Remove timing anchor
- → It took on average of 20 faulting attempts to induce the desired error



TZ Attack #1: Inferring AES Keys

- Breaks confidentiality

- Load victim app into Trustzone
- Induce fault during AES decryption
- Infer key from correct / faulty plaintext pair via DFA



TZ Attack #1: Inferring AES Keys

Differential Fault Analysis

- Induce faults to reveal the internal state of AES
- Exploit properties of encryption / decryption function
- Induce error of one byte in seventh or eighth round of decryption
- Reduces possible keys from 2^{128} to 2^{12} (brute-force)

TZ Attack #2: loading self signed apps

- Subvert RSA signature chain verification
- Load firmware images into Trustzone (verified via RSA signature)
- Goal: Provide an arbitrary attack app within Trustzone (self signed)
- Achieved via inducing fault during decrypt signature code execution

Novelty, Summary & Takeaways

Novelty & Summary

- Attack without physical access or any special equipment
- Only software based (malicious kernel driver)
- Neither software NOR hardware bug => fundamental design flaw
- Only publicly available knowledge used

Strengths

Strengths

- Paper introduces an important problem, motivates further research
- Shows two different practical ways to exploit this design flaw
- Suggest solutions
- Interestingly written

Weaknesses

Weaknesses

- Only tested on Nexus 6
- Relies on being able to track execution cycles
- Security review: not at all complete
- How long would it take in the real world?
- Structure of Paper: badly structured
- Use of self-implemented AES decryption app

Takeaways & Further Work

Takeaways

- New attack surface: Energy Management
- No hardware OR software bug, but a fundamental design flaw
- Applies to other leading architectures (i.e. cloud computing)
- Use findings for future energy management designs

  → Take security into account

Further Work

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

Autoren	Sheng Zhang, Adrian Tang, Zhewei Jiang, Simha Sethumadhavan, Mingoo Seok
Publikationsdatum	2018/7/23
Konferenz	Proceedings of the International Symposium on Low Power Electronics and Design

Thoughts, Ideas & Open Discussion

Open Discussion

What are possible ways to defend a systems against the CLKScrew attack?

Software

- Randomization
- Code execution redundancy

Hardware

- Seperate cross-boundary regulators
- Microarchitectural redundandency

Open Discussion

Do you have other ideas how to exploit energy management?

- Break software safety constraints
- Make unstable
- Block complete process (no progress)

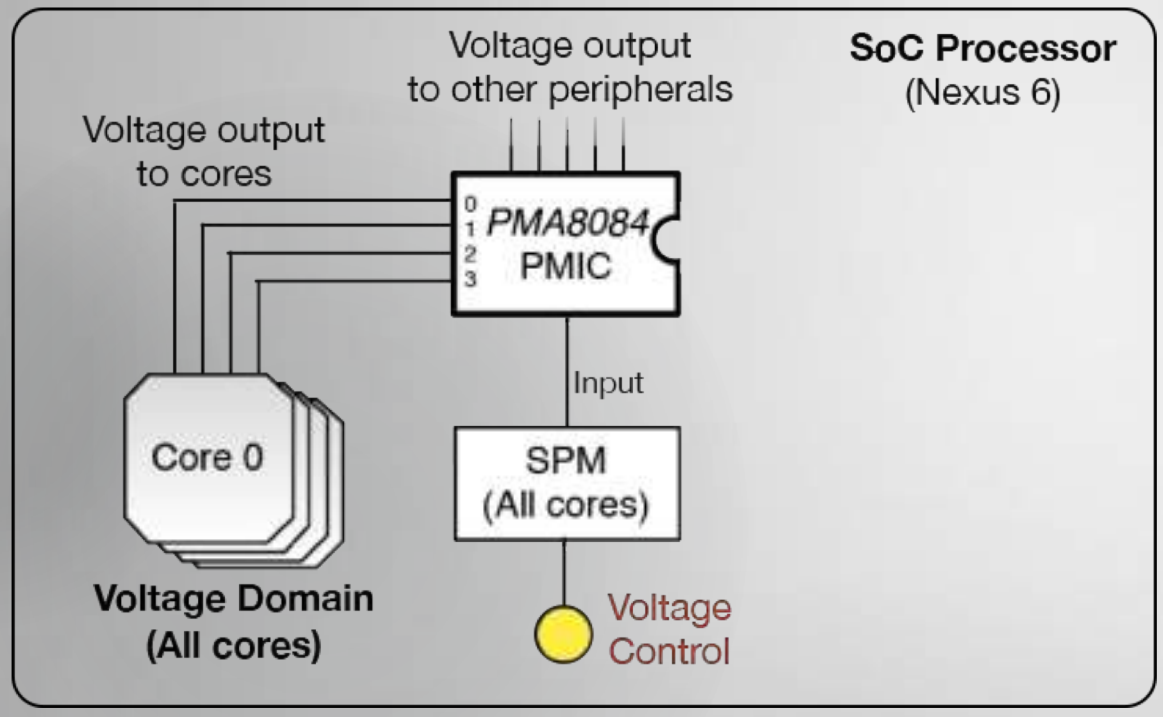
Open Discussion

What are possible ways to prevent such new attack surfaces in the future?

Thank you for your attention!

Backup Slides

Voltage regulators

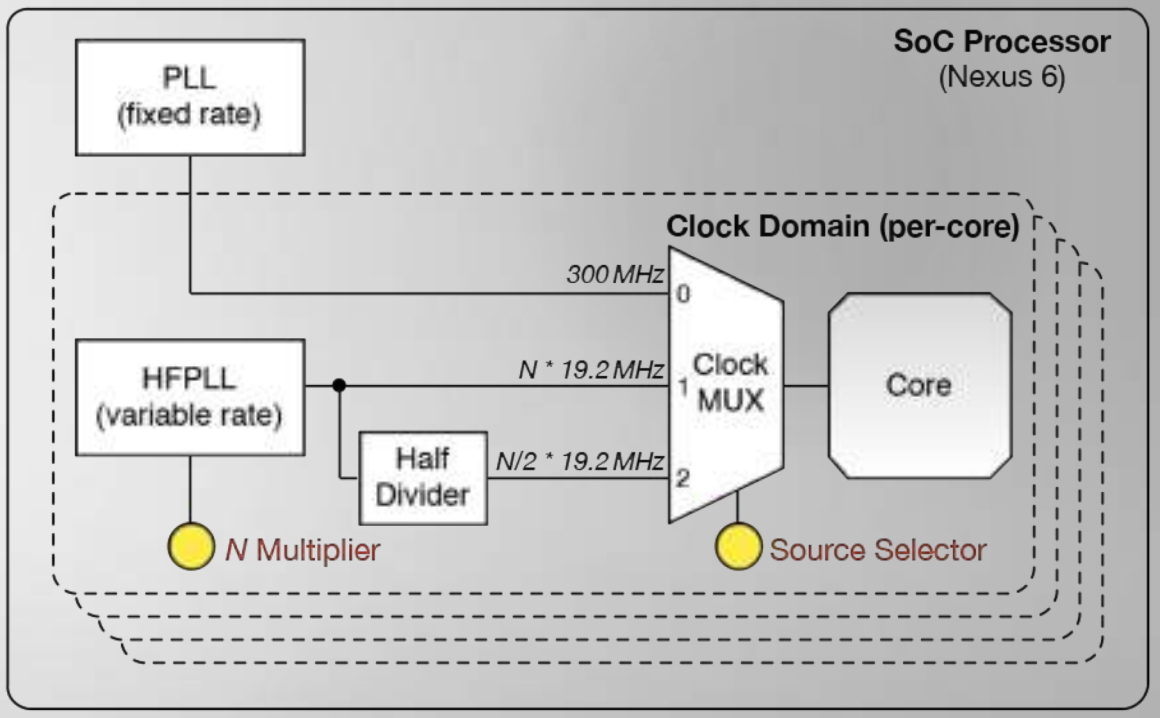


- Supply power to various components
- Integrated within the Power Management Integrated Circuit (PMIC)
- Software interface (control registers) via Subsystem Power Manager (SPM)

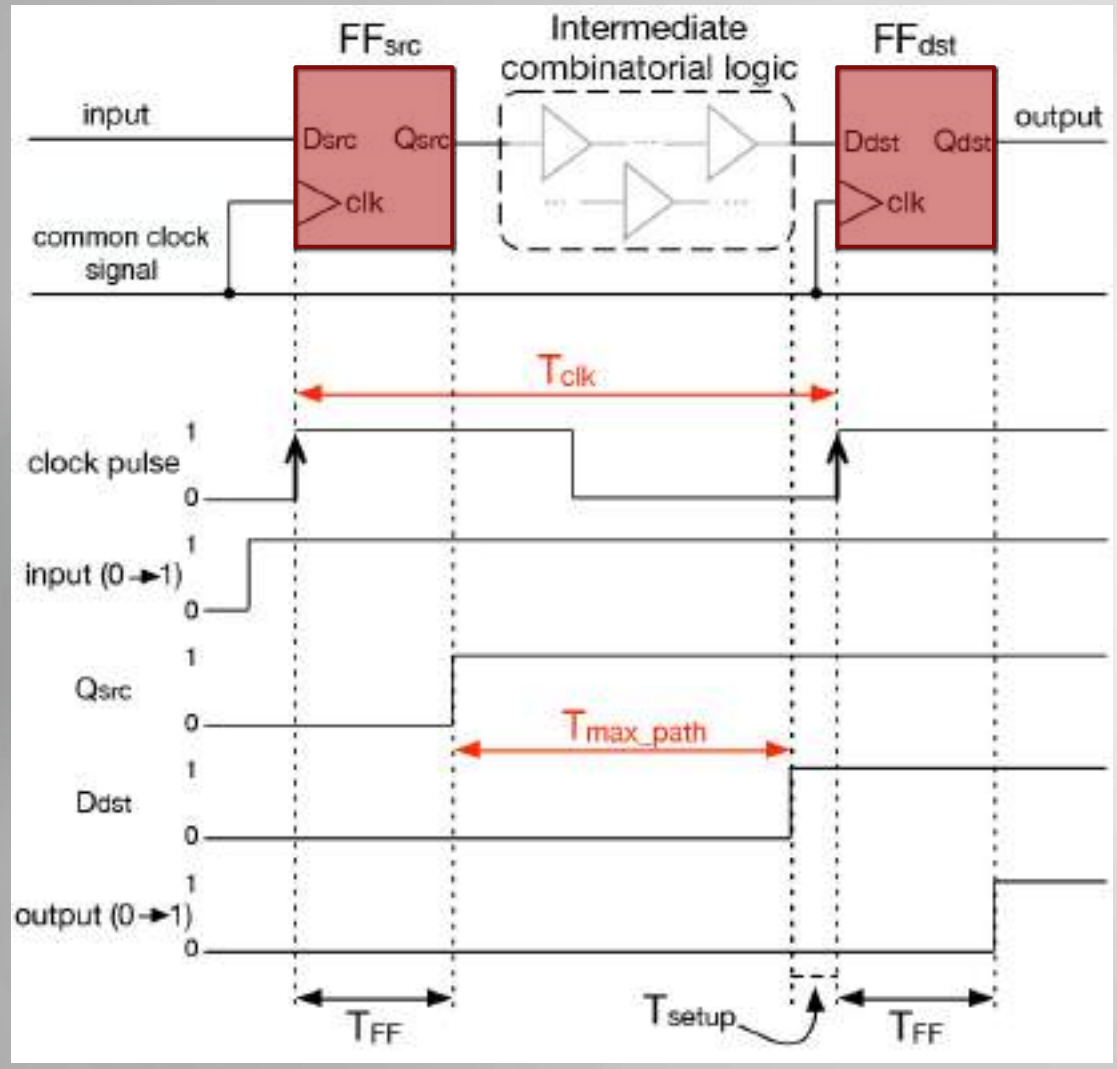
Frequency regulators

Each core can operate on 3 possible clock sources:

- Phase Lock Loop (PLL): fixed clock signal
- High Frequency PLL (HFPLL): variable clk signal
- Same HFPLL, half of signal



Overclocking & Undervolting



A Flip-Flop should:

- Hold stable *incoming signal* for $\mathbf{T_{setup}}$
- Hold stable *input signal* for $\mathbf{T_{FF}}$ within itself

Signal takes $\mathbf{T_{max\_path}}$ to propagate

$$T_{clk} \geq T_{FF} + T_{max\_path} + T_{setup} + K$$

Parameters to optimize

$$F_{\theta|T_{anchor}} = \{F_{volt}, F_{pdelay}, F_{freq\_hi}, F_{dur}, F_{freq\_lo}\}$$

Parameter	Description
F_{volt}	Base operating voltage
F_{pdelay}	Number of loops to delay/wait before the fault
F_{freq_hi}	Target value to raise the frequency *to* for the fault
F_{freq_lo}	Base value to raise the frequency *from* for the fault
F_{dur}	Duration of the fault in terms of number of loops