Security Implications of Power Management Mechanisms in Modern Processors

Current Studies and Future Trends

Jawad Haj-Yahya

Principal Architect, Rivos Inc.

The International Winter School on Microarchitectural Security (Mic-Sec) Paris, 5 December 2022

Overview

- Billions of new devices are being deployed each year with new services and features that are driving a higher demand for high performance microprocessors, which often have high power consumption

- Despite the failure of Dennard scaling, the slow-down in Moore's Law, and the high power-density of modern processors, power management mechanisms have enabled significant advances in modern microprocessor performance and energy efficiency

- Yet, current power management architectures also pose serious security implications
  - This is mainly because functionality rather than security has been the main consideration in the design of power management mechanisms in commodity microprocessors

High Level Outline

- We will cover:
  - An overview of state-of-the-art power management mechanisms used in modern microprocessors
  - Multiple state-of-the-art security vulnerabilities that exploit power management mechanisms and mitigations to protect against these vulnerabilities
    - Deep dive into our IChannels vulnerabilities (ISCA 2021)

✓

- We will not cover:
  - Power Analysis Attacks
    - Simple power analysis (SPA)
    - Differential power analysis (DPA)
    - Correlation Power Analysis (CPA)
  - Fault injection attacks (FIA) details

✗

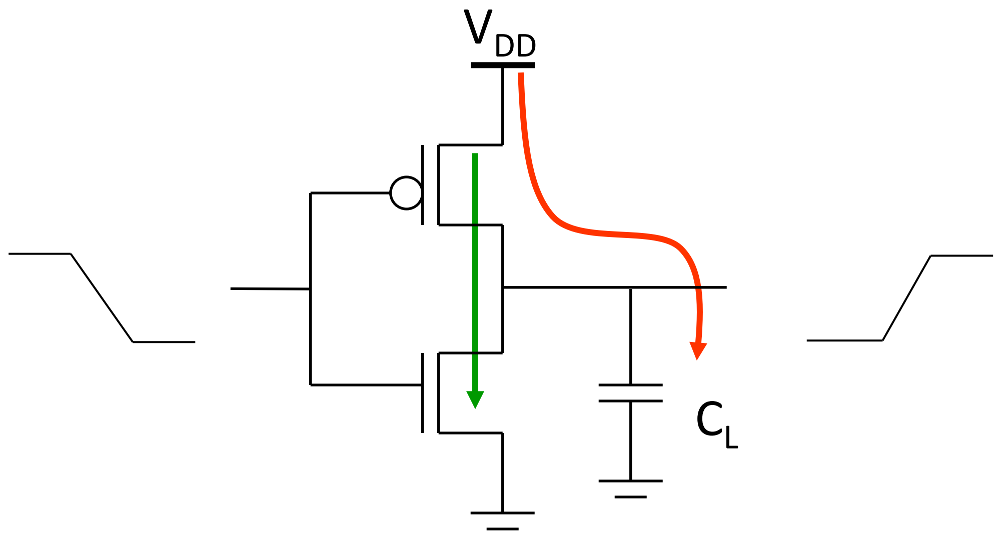
Power Management Mechanisms

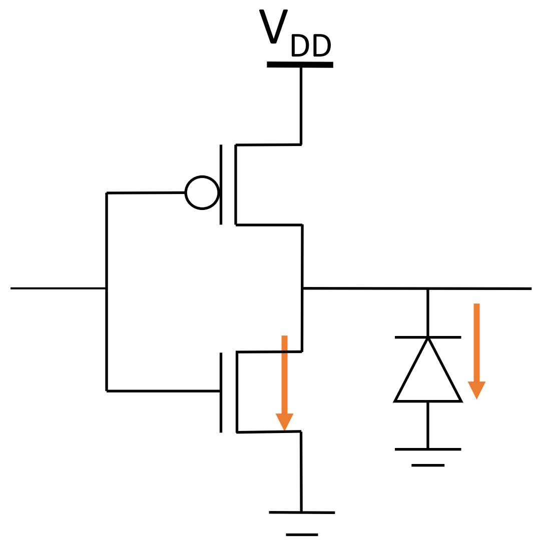
CMOS Power Consumption

- Dynamic Power
  - Signal transitions
    - Logic activity
    - Glitches
  - Short-circuit
- Static Power
  - Leakage

$$P_{total} = P_{dyn} + P_{stat}$$

$$= P_{tran} + P_{sc} + P_{lkg}$$





Dynamic Power Consumption

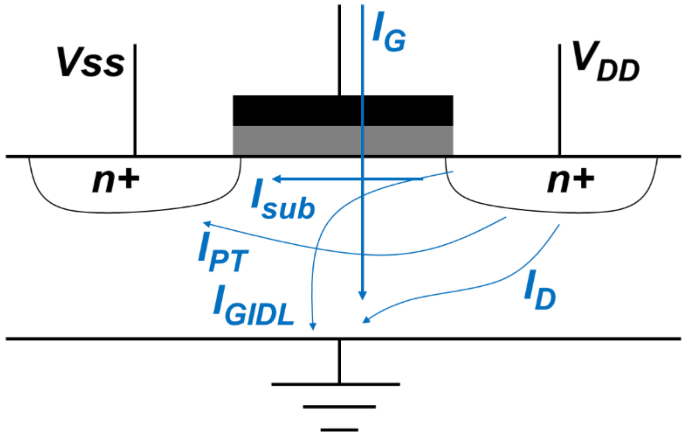
- Dynamic power: $P_{dyn} = \alpha \cdot C_L \cdot f_{clk} \cdot V_{DD}^2$
  - α activity factor (i.e., the probability the given node will change its state from 1 to 0 or vice versa at a given clock tick)
  - C_L total load capacitance
  - f_{clk} clock frequency
  - V_{DD} supply voltage

- Circuit techniques to reduce dynamic power
  - State/bus encoding (↓α)
  - Reduce device size (↓C_L)
  - Pipelining & parallelism (↓ f_{clk}, ↓ V_{DD})

- Run-time techniques to reduce dynamic power
  - Clock-gating (↓α)
  - Dynamic Voltage & Frequency Scaling - DVFS (↓ f_{clk}, ↓ V_{DD})

Leakage Power Consumption

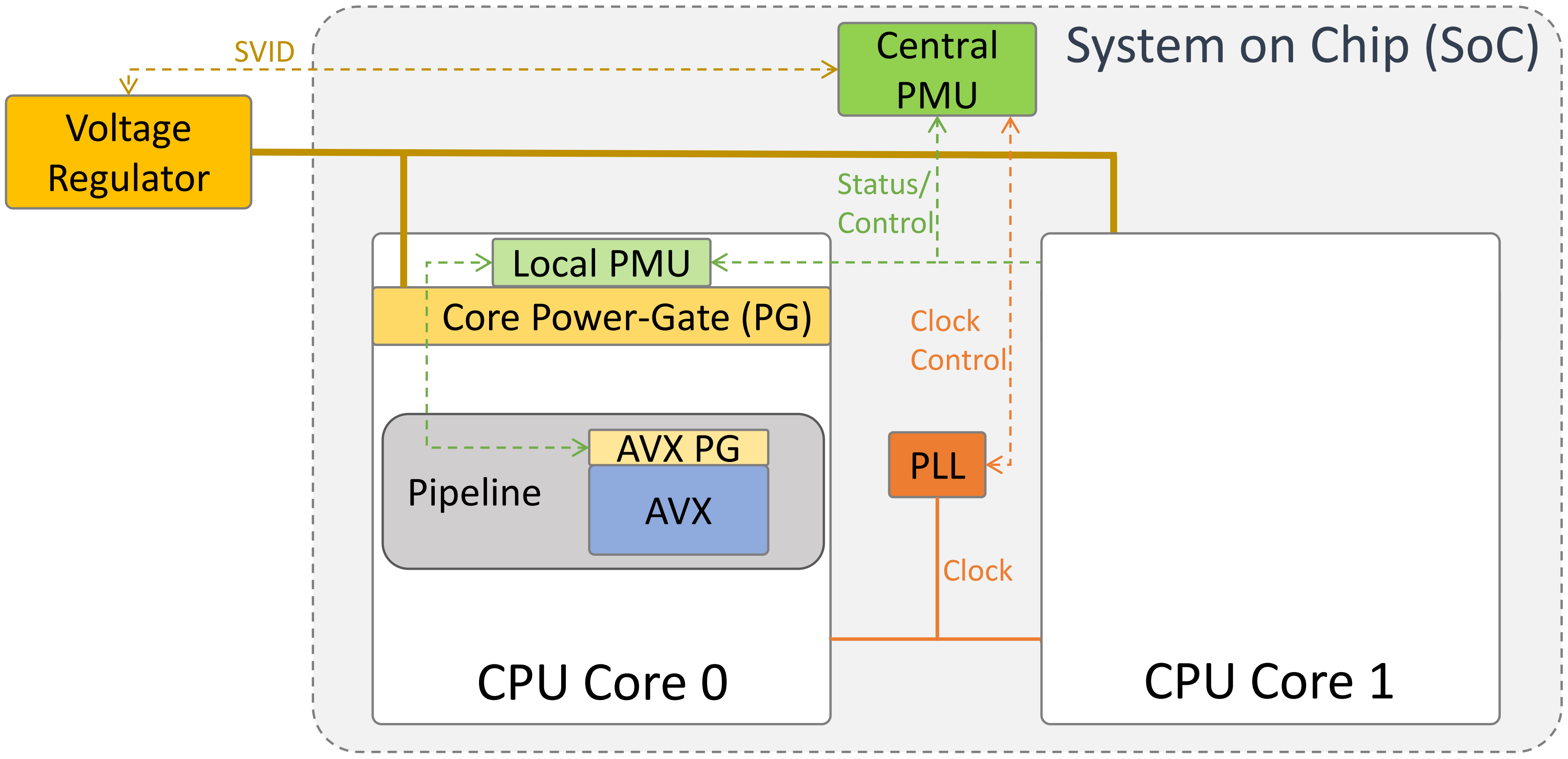
- Static power: $P_{static} = I_{stat} \cdot V_{DD} = (I_{sub} + I_D + I_{GIDL} + I_{PT} + I_G) \cdot V_{DD}$
  - I_{sub} Subthreshold leakage
  - I_D Junction Reverse Bias Current
  - I_{GIDL} Gate Induced Drain Leakage
  - I_{PT} Punch-through Current
  - I_G Gate Tunneling Currents
  - V_{DD} Supply voltage



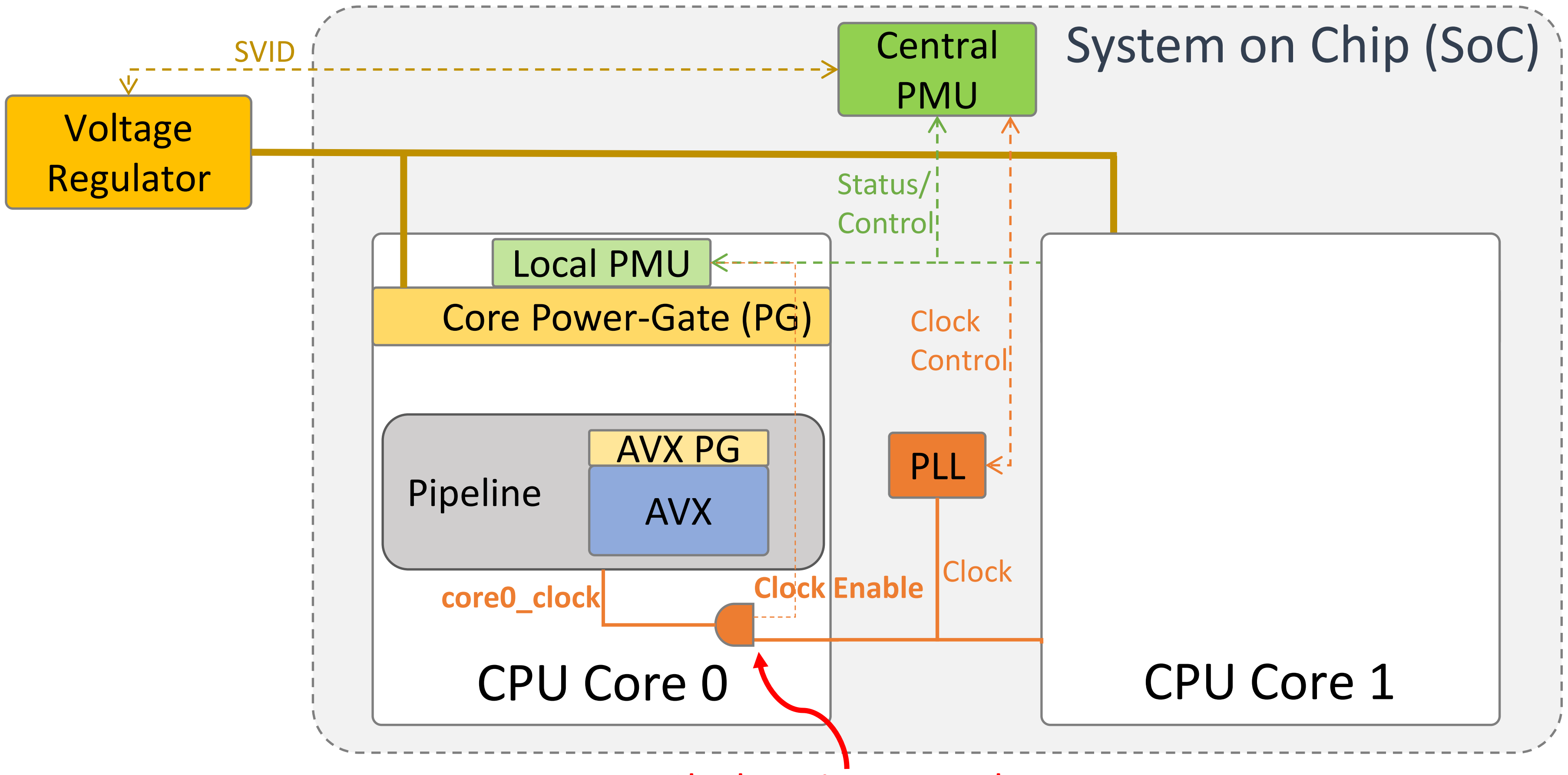
- Circuit techniques to reduce leakage power
  - Increase V_t: use Multiple-threshold (V_t) devices (↓ I_{stat})
    - Use low V_t devices (have high leakage) only in critical circuits

- Run-time techniques to reduce leakage power
  - Reduce idle circuit's voltage to retention (↓ V_{DD})
  - Power-gate idle circuit (↓ V_{DD})
  - Dynamic Voltage & Frequency Scaling - DVFS (↓ V_{DD})

PM Architecture Overview

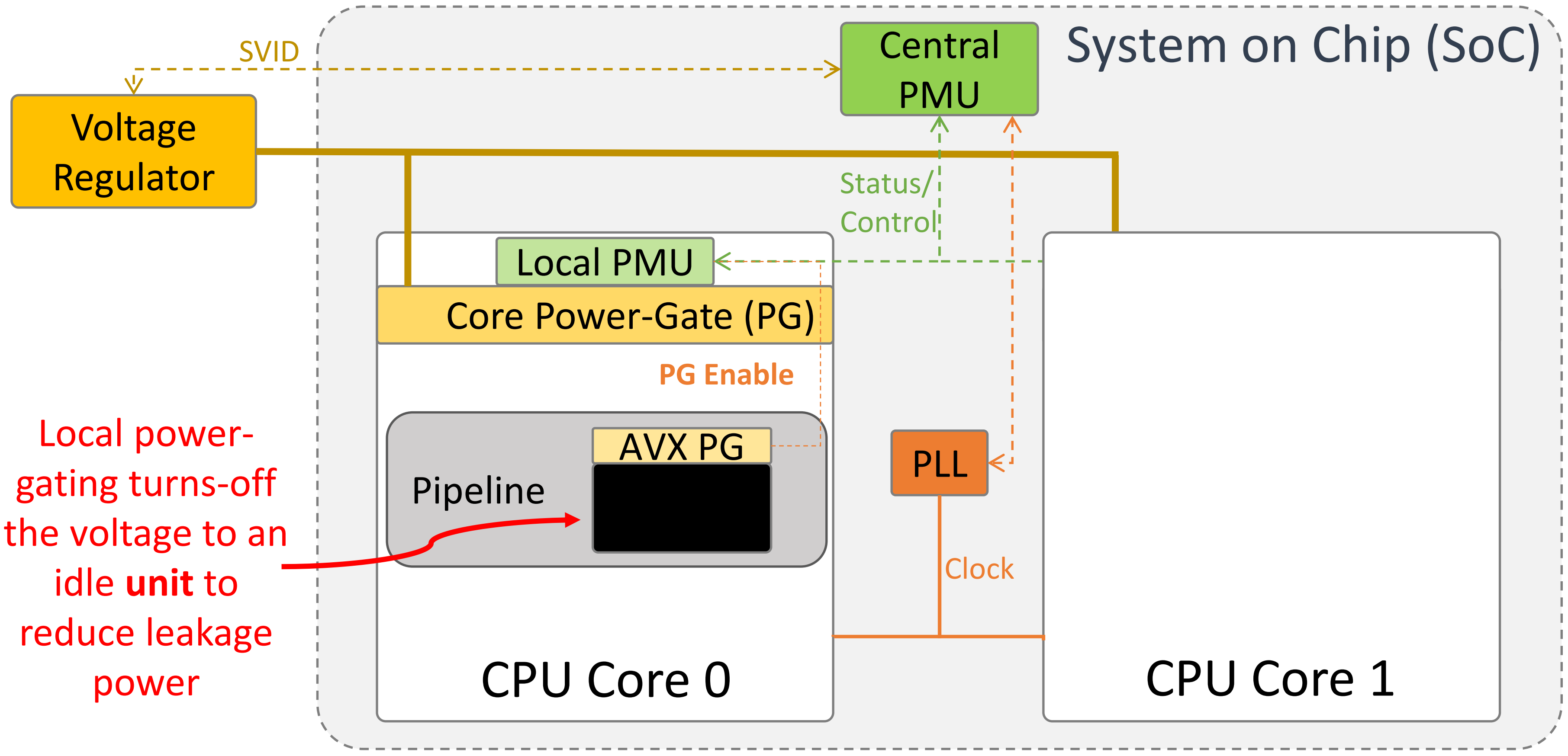


Clock-gating



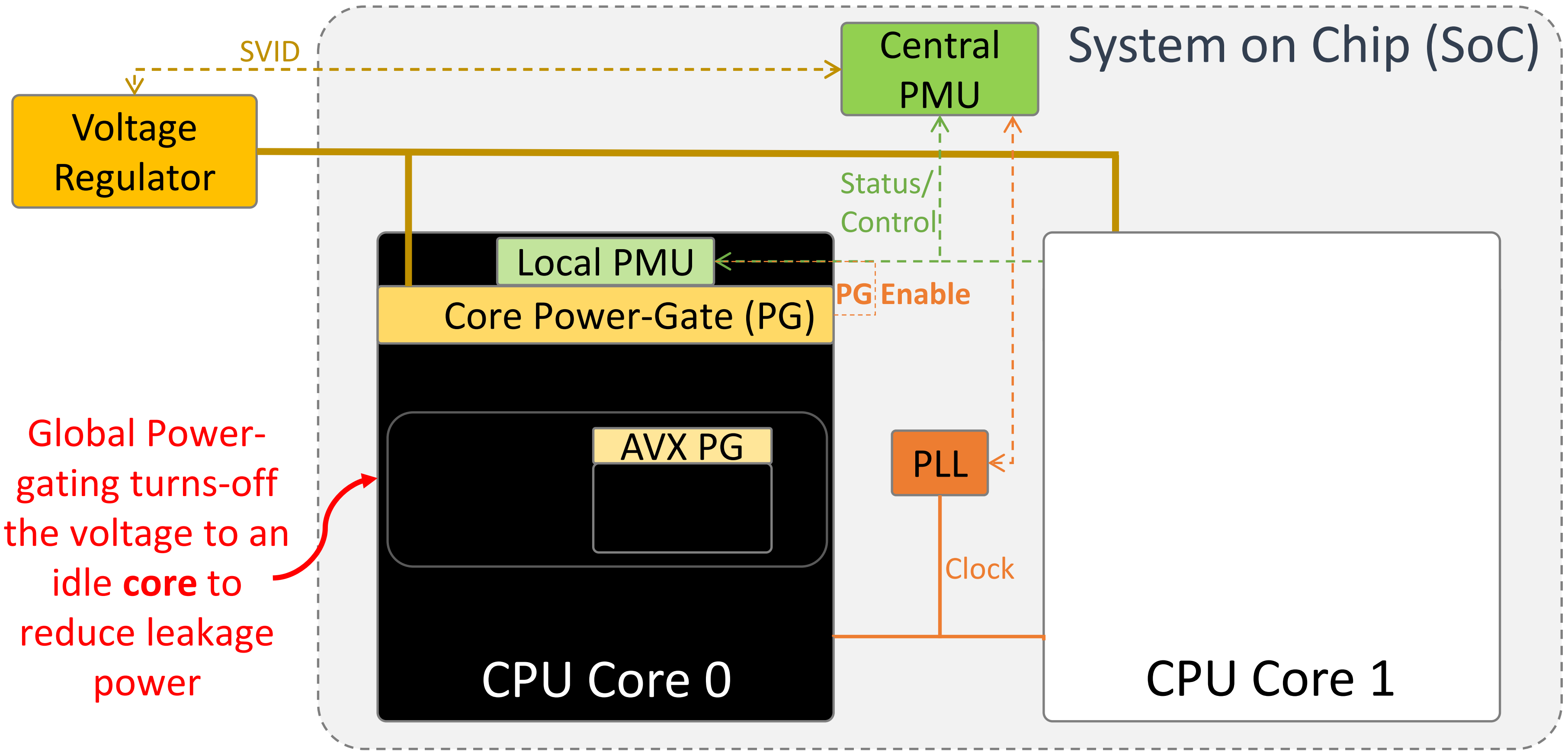
Clock-gating stops the clock to an idle core/unit to reduce dynamic power

Local Power-gating



Local power-gating turns-off the voltage to an idle **unit** to reduce leakage power

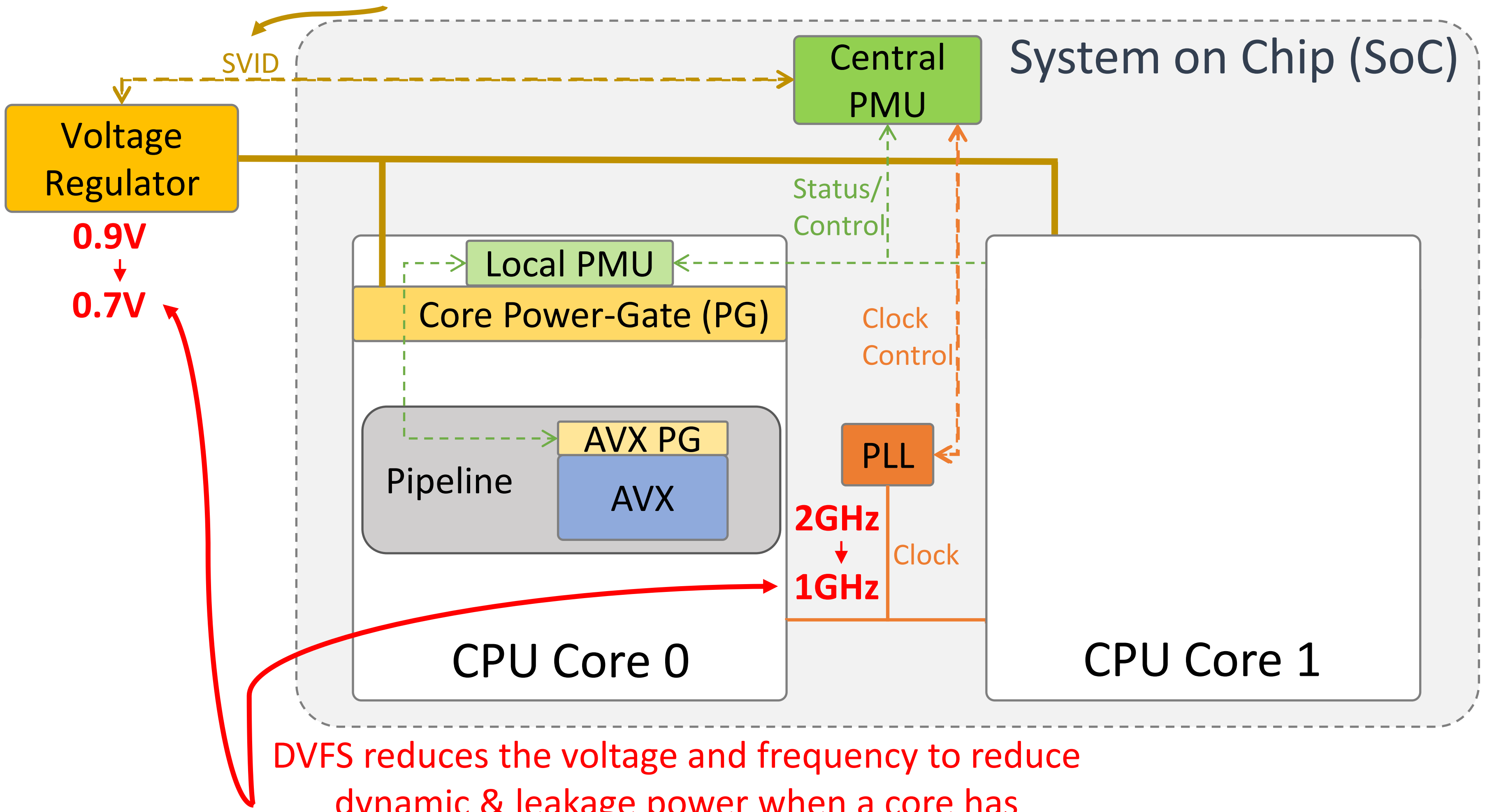
Global Power-gating



Global Power-gating turns-off the voltage to an idle **core** to reduce leakage power

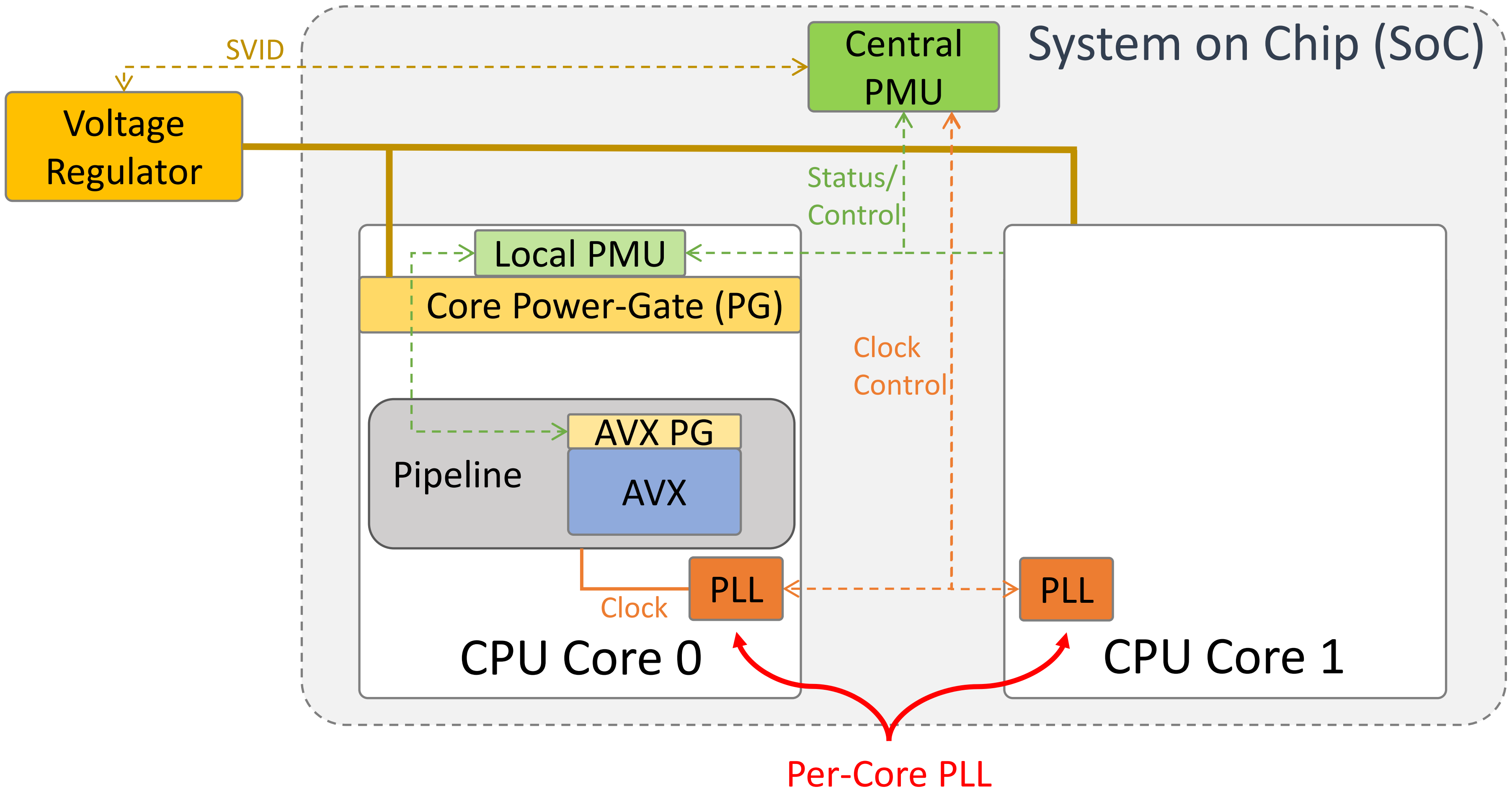
DVFS

The PMU controls the VR using an off-chip serial voltage identification (SVID)



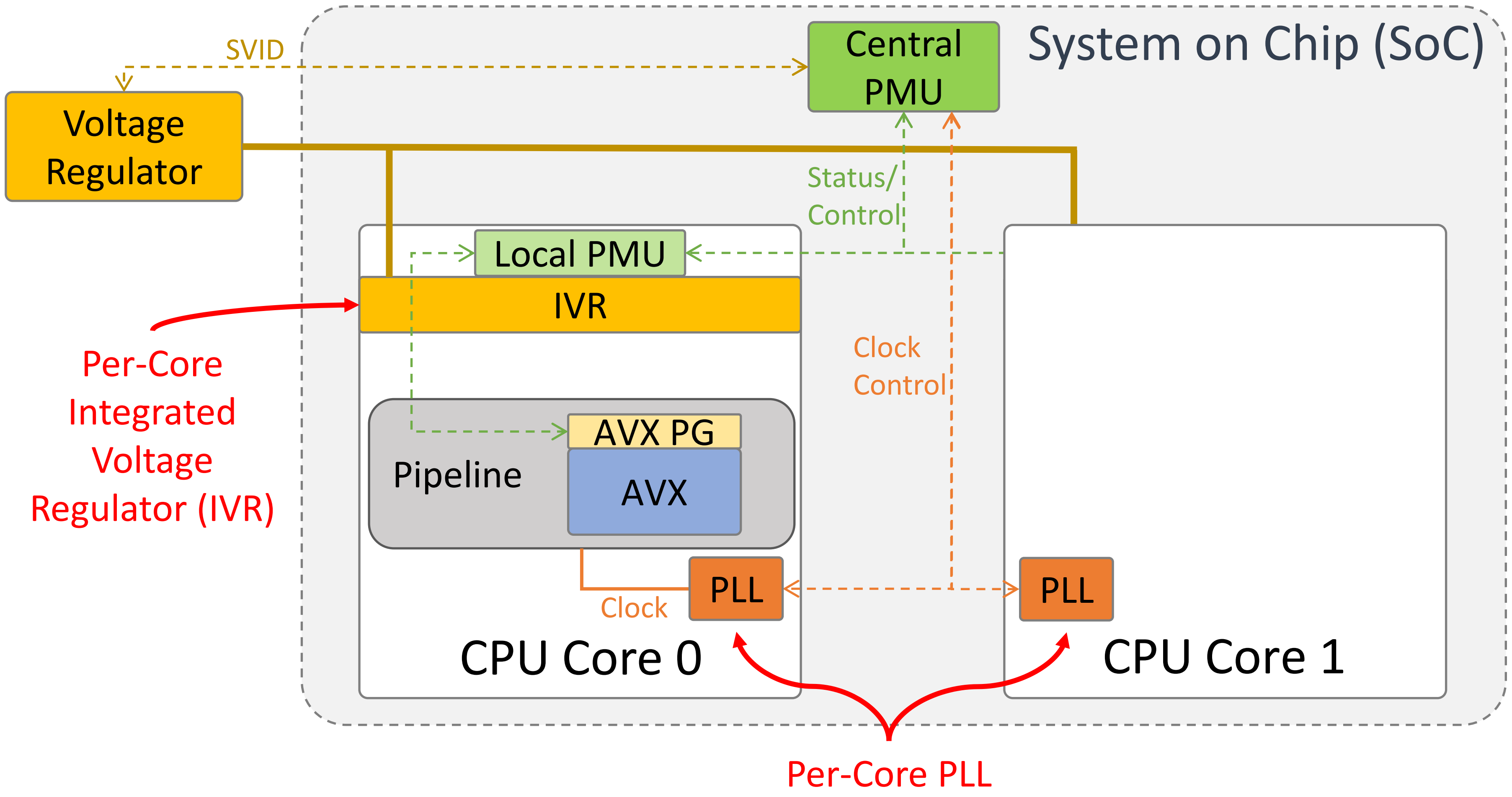
DVFS reduces the voltage and frequency to reduce dynamic & leakage power when a core has
1) low-utilization, or
2) high temperature

Advanced PM Architecture (I)



Per-Core PLL enable *per-cores Dynamic Frequency Scaling (DFS)*

Advanced PM Architecture (II)



Per-Core IVR and PLL enable *per-core DVFS*

More Advanced PM Features

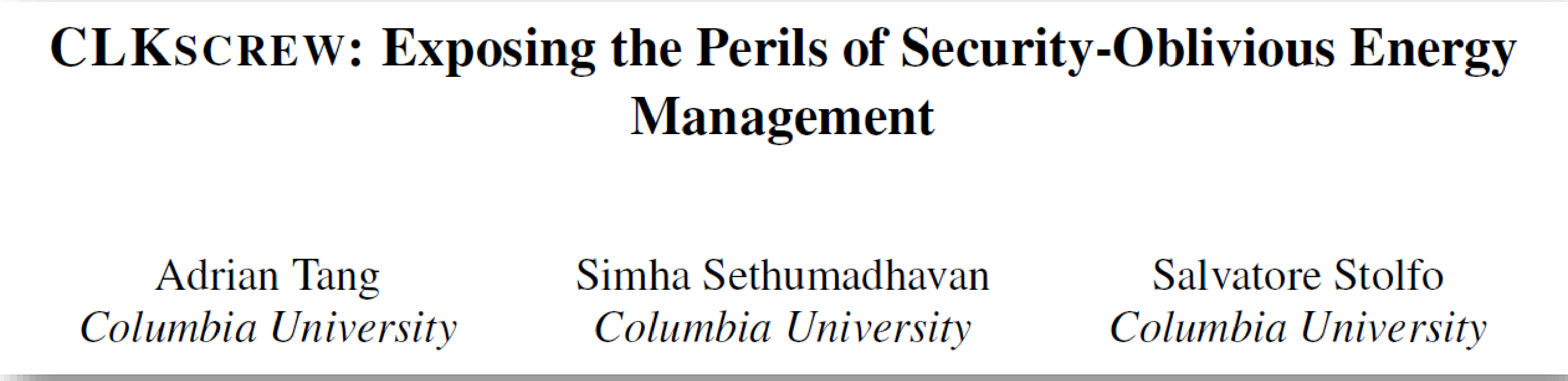
- There are more advanced PM features:
  - Power budget management
  - Computational sprinting (e.g., Turbo)
  - Maximum current limit protection
  - Maximum voltage limit protection
  - Voltage emergency prevention & avoidance
  - Adaptive voltage scaling
  - Reliability degradation mitigation
  - System level idle power-states
  - System level DVFS
  - Race to halt
  - Hardware duty cycling
  - …

Power Management Security Vulnerabilities and Mitigation

Overview

- Many of todays power management (PM) architectures pose serious security vulnerabilities
- Recent works demonstrated security attacks that exploit vulnerabilities in multiple PM components:
  - Frequency
  - Voltage
  - Power
  - Thermal
  - Power supply
  - Power management unit
  - Current management
  - Power management interfaces
  - …
- We will overview some of these attacks and their mitigations

Frequency Vulnerability Example


CLKSCREW: Exposing the Perils of Security-Oblivious Energy Management

Adrian Tang
Columbia University

Simha Sethumadhavan
Columbia University

Salvatore Stolfo
Columbia University

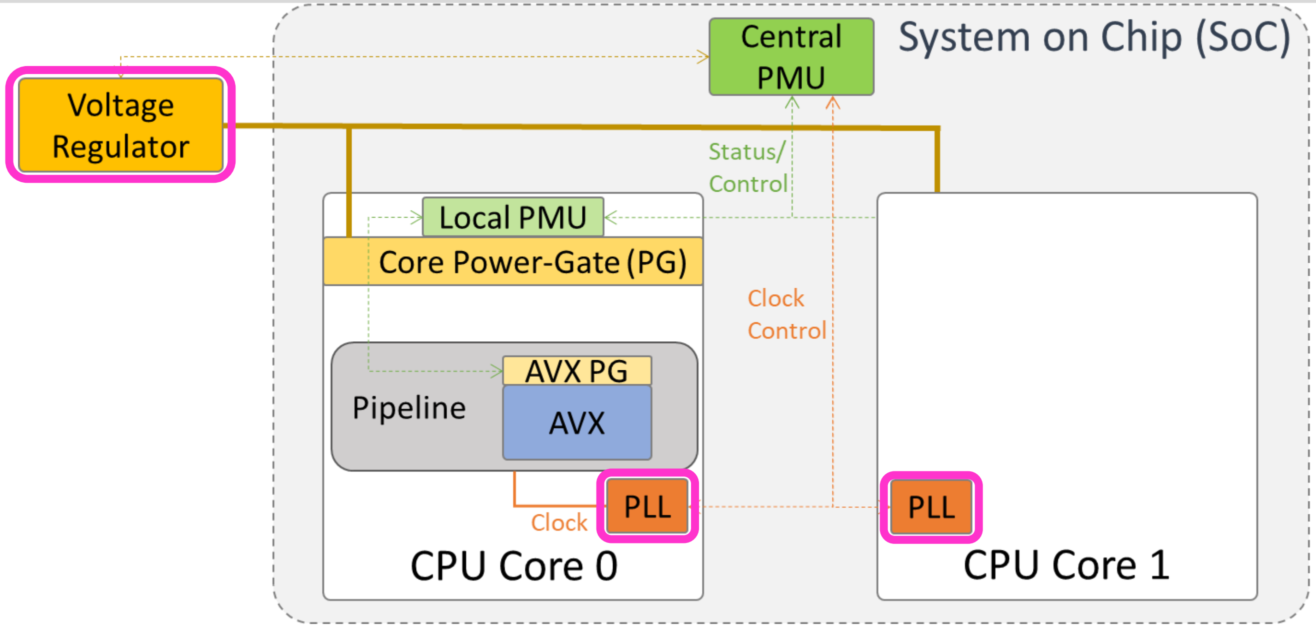
Tang, Adrian et al. CLKSCREW: Exposing the Perils of {Security-Oblivious} Energy Management.“, *USENIX* 2017.

CLKSCREW Overview

- CLKSCREW attack exploits the frequency control to break the security of an ARM/Android device (Nexus 6)
- The attack
  - Exploits software accessible per-core frequency control registers
  - Using these registers, the attacker's code can increase victim core's frequency (overclocking) without increasing the voltage
  - Induces faults (bit-flips) in core running a TrustedZone code, which the attacker can exploit to retrieve a secure key

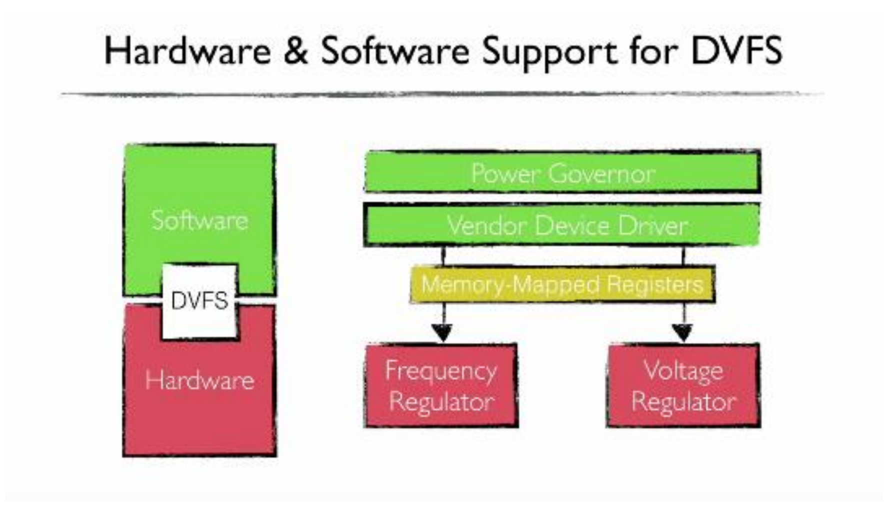
Source: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/tang

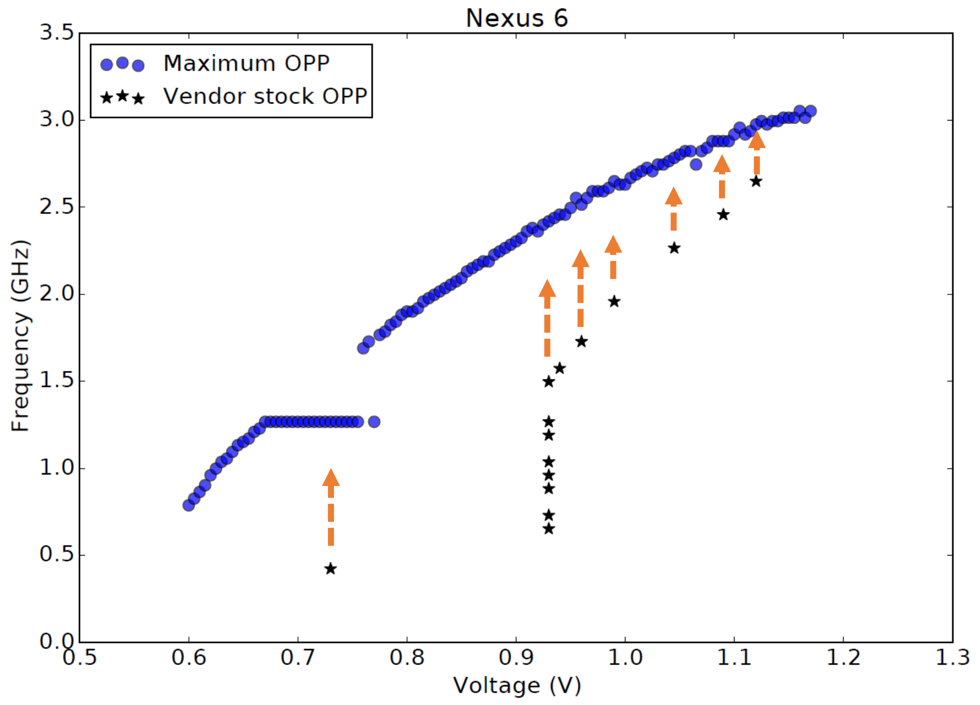
Target Device PM Architecture



- The target (Nexus 6) device PM architecture has
  - All-cores voltage control
  - Per-core frequency control

SW Voltage/Frequency Control

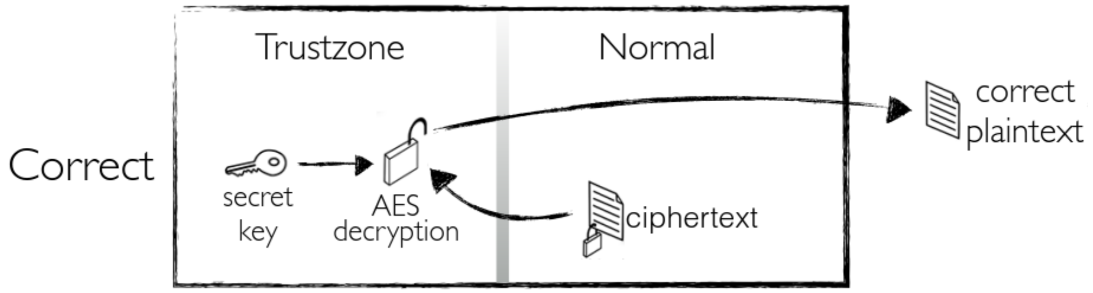




- The software has separate controls of voltage and frequency
  - Without checking the validly of the voltage/frequency operating point
- At a given voltage level, the SW can increase the frequency
  - Beyond the vendor recommended voltage/frequency operating point
  - Inducing computation faults due to circuit timing violation

Source: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/tang

CLKSCREW Key Inference Attack



- The victim app (AES decryption) is executing in the Trustedzone
- The attacker code is running on the Normal world. The attacker code can:
  - Repeatedly invoke the victim app
  - Control the frequency of the core running the victim app
- The attacker induce a fault during AES decryption
- Infer the secret key from pairs of correct and faulty plaintext using differential fault analysis [1]

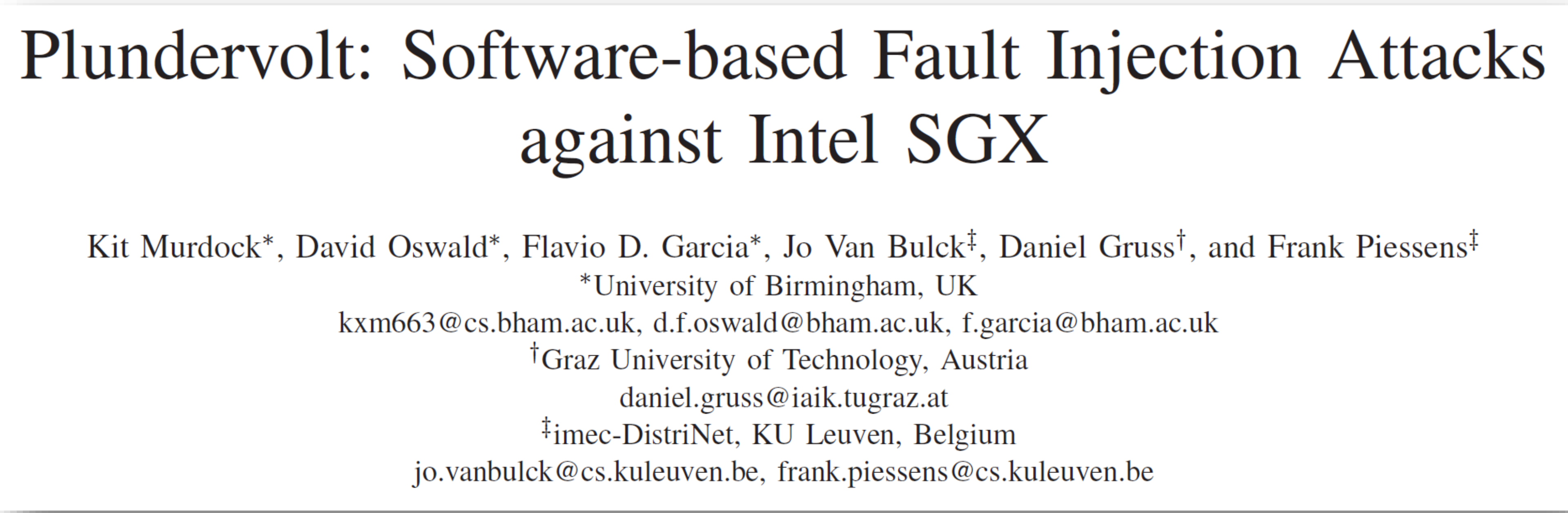
[1] Tunstall, M., Mukhopadhyay, D., And Ali, S. Differential Fault Analysis Of The Advanced Encryption Standard Using A Single Fault. In IFIP International Workshop On Information Security Theory And Practices (2011), Springer, Pp. 224–233.

CLKSCREW Mitigations

- Hardware-Level
  - Limit operating points (V/F) to safe values in hardware
  - Separate operating points mechanism for secure/normal worlds
  - Logic/timing redundancy and recovery mechanisms to mitigate the effects of computation faults
- Software-Level
  - Randomization to the runtime execution to prevent the attacker from gaining clear timing anchor used to induce the faults
  - Code execution redundancy by executing sensitive code multiple times

Tang, Adrian et al. CLKSCREW: Exposing the Perils of {Security-Oblivious} Energy Management.“, *USENIX* 2017.

Voltage Vulnerability Example

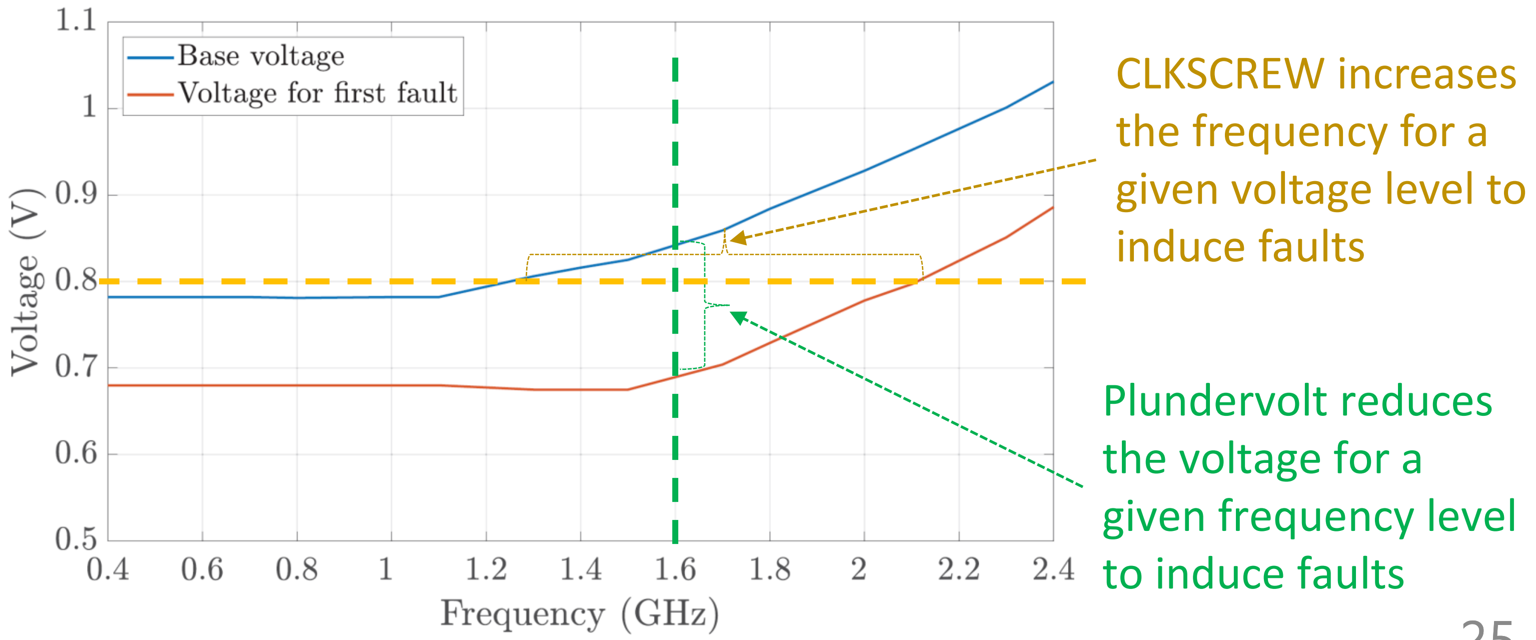

Plundervolt: Software-based Fault Injection Attacks against Intel SGX

Kit Murdock*, David Oswald*, Flavio D. Garcia*, Jo Van Bulck‡, Daniel Gruss†, and Frank Piessens‡
*University of Birmingham, UK
kxm663@cs.bham.ac.uk, d.f.oswald@bham.ac.uk, f.garcia@bham.ac.uk
†Graz University of Technology, Austria
daniel.gruss@iaik.tugraz.at
‡imec-DistriNet, KU Leuven, Belgium
jo.vanbulck@cs.kuleuven.be, frank.piessens@cs.kuleuven.be

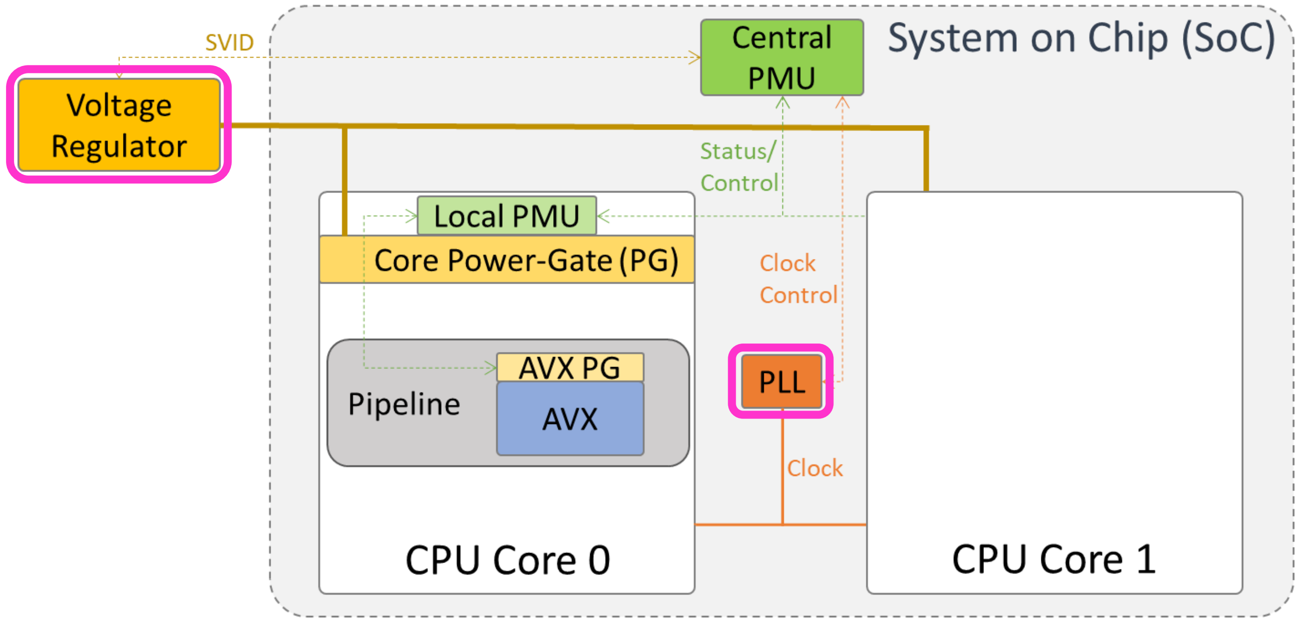
Murdock, K., Oswald, D., Garcia, F.D., Van Bulck, J., Gruss, D. and Piessens, F., 2020, May. Plundervolt: Software-based fault injection attacks against Intel SGX. In *2020 IEEE Symposium on Security and Privacy (SP).*

Plundervolt Overview

- Plundervolt utilizes an undocumented Intel Core voltage scaling software interface to undervolt cores voltage
  - to corrupt the integrity of Intel SGX enclave computations
- Similar to CLKSCREW, Plundervolt violates a circuit timing constrains to induce a fault
  - CLKSCREW uses overclocking (without increasing the voltage)
  - Plundervolt uses undervolting (without reducing the frequency)

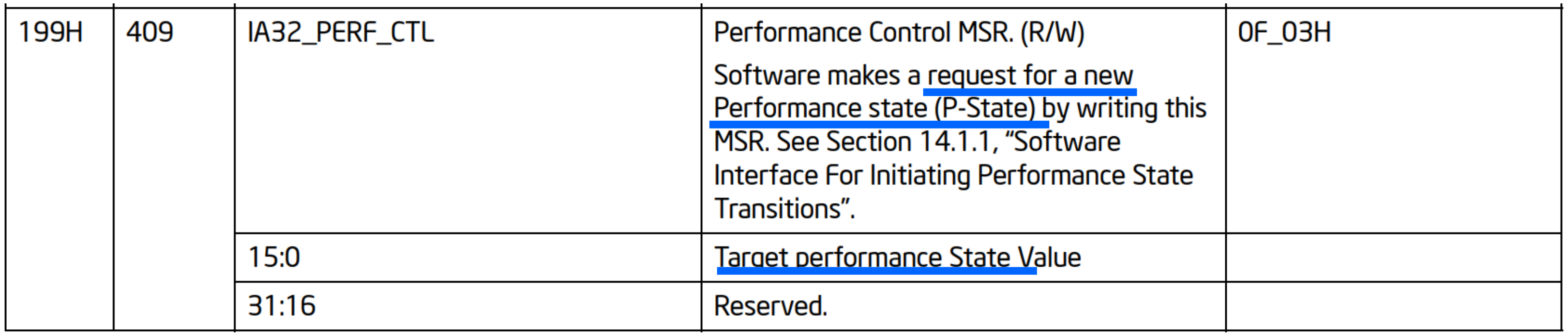


Target Device PM Architecture



- Plundervolt was demonstrated on Intel Skylake, Kaby Lake, and Coffee Lake client processors
- The PM architecture of these processors has:
  - All-cores voltage control
  - All-cores frequency control

SW Voltage/Frequency Control

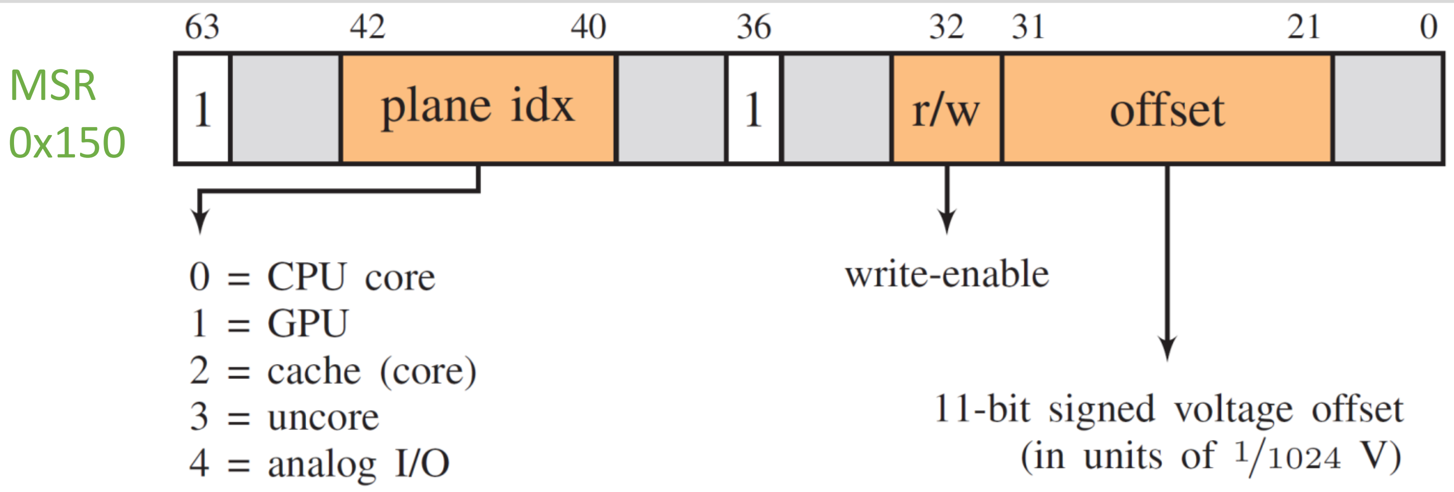


199H	409	IA32_PERF_CTL	Performance Control MSR. (R/W) Software makes a request for a new Performance state (P-State) by writing this MSR. See Section 14.1.1, "Software Interface For Initiating Performance State Transitions".	0F_03H
		15:0	Target performance State Value	
		31:16	Reserved.	

- The OS PM (OSPM) driver can request to change performance state (P-state) by writing to IA32_PERF_CTL register (199H)
  - P-state corresponds to cores operating frequency
- The PMU calculates the appropriate voltage for the requested P-state (frequency) online
  - Based on HW fuses values and other parameters (e.g., temperature)
  - Therefore, the CLKSCREW attack cannot be invoked on Intel SoCs, since SW doesn't have direct control on P-state's voltage
- Plundervolt found another way to control the voltage

Source: Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 4: Model-Specific Registers, 2017

Plundervolt SW Voltage Control



- Plundervolt uses an undocumented Intel SW register (MSR 0x150) to undervolt the operating voltage to inject faults
  - The register is typically used as an overclocking mailbox interface
- The register allows voltage change using two main fields
  - *plane index* - select an SoC components (e.g., CPU core)
  - *voltage offset* - the requested voltage scaling offset (signed)
- After writing to the register, the new operating voltage can be queried from the documented IA32_PERF_STATUS register

Source: Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 4: Model-Specific Registers, 2017

Plundervolt Attacks

- Plundervolt controls the processor's supply voltage (undervolting) during an enclave computation
  - Inducing predictable faults within the processor

- Key recovery PoC attacks against RSA-CRT and AES-NI

- Enable multiple memory safety violations attacks
  - Faulting array index addresses
  - Faulting memory allocation sizes

Murdock, K., Oswald, D., Garcia, F.D., Van Bulck, J., Gruss, D. and Piessens, F., 2020, May. Plundervolt: Software-based fault injection attacks against Intel SGX. In *2020 IEEE Symposium on Security and Privacy (SP).*

Plundervolt Mitigations

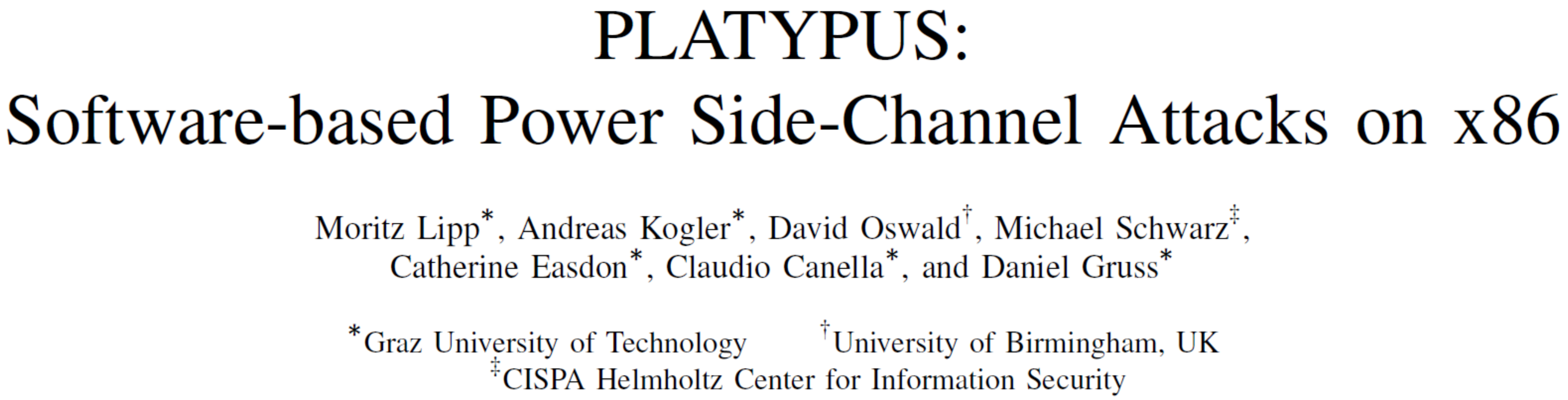
- Hardware-Level
  - Disable the access to MSR 0x150 unless overclocking system is enabled
  - Limit operating points (V/F) to safe values in hardware
  - Separate operating points mechanism for secure/normal worlds
  - Logic/timing redundancy and recovery mechanisms to mitigate the effects of computation faults

- Software-Level
  - Fault-resistant cryptographic primitives
  - Application and compiler hardening by inserting correctness checks
  - Memory Safety Hardening by randomize the enclave memory layout

- Intel mitigated Plundervolt by disabling MSR 0x150 using a BIOS configuration

Murdock, K., Oswald, D., Garcia, F.D., Van Bulck, J., Gruss, D. and Piessens, F., 2020, May. Plundervolt: Software-based fault injection attacks against Intel SGX. In *2020 IEEE Symposium on Security and Privacy (SP).*

Power Vulnerability Example



PLATYPUS: Software-based Power Side-Channel Attacks on x86

Moritz Lipp[*], Andreas Kogler[*], David Oswald[†], Michael Schwarz[‡], Catherine Easdon[*], Claudio Canella[*], and Daniel Gruss[*]

[*]Graz University of Technology [†]University of Birmingham, UK
[‡]CISPA Helmholtz Center for Information Security

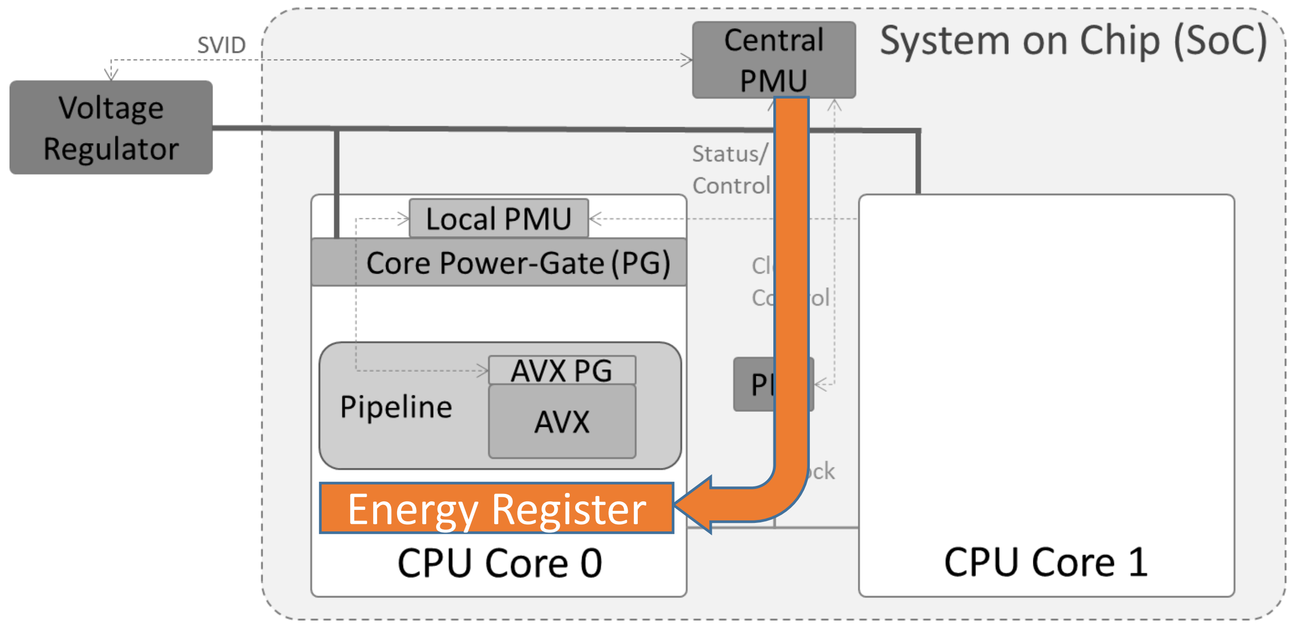
PLATYPUS Overview

- PLATYPUS exploit unprivileged access to the Intel Running Average Power Limit (RAPL) interface
  - RAPL exposes values directly correlated with power consumption
  - RAPL forms a low-resolution side channel

- RAPL allows observing variations in power consumption
  - Distinguish different instructions and different Hamming weights of operands and memory loads
  - Which allows the attacker to monitor the control flow of applications and infer data and extract cryptographic keys

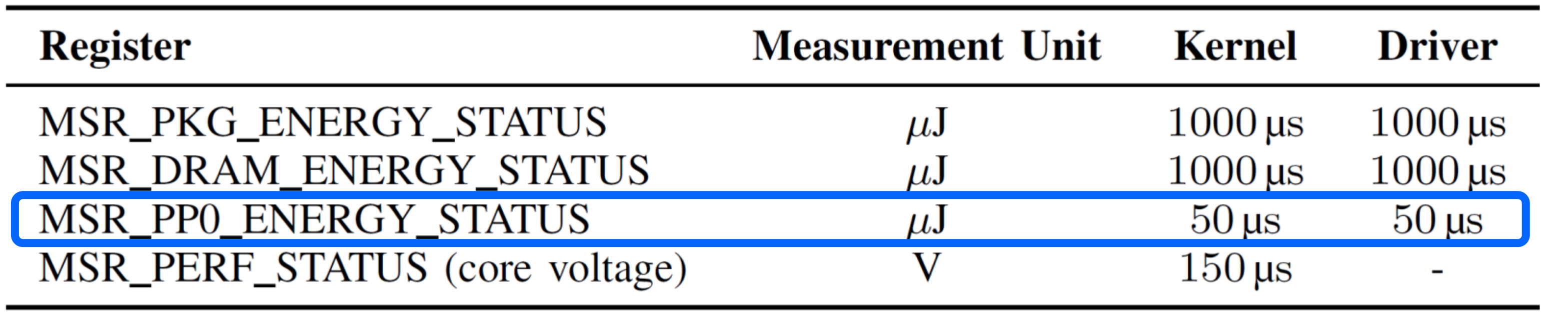
- PLATYPUS demonstrate how to
  - Leak AES-NI secret keys from Intel SGX and the Linux kernel
  - Break kernel address-space layout randomization (KASLR)
  - Infer secret instruction streams and
  - Establish a timing-independent covert channel

Target Device PM Architecture



- PLATYPUS was demonstrated on Intel Skylake, Kaby Lake, Coffee Lake, and Cascade Lake processors
  - The architecture provides energy reading SW interface for multiple domains

PLATYPUS Utilized SW Interface



Register	Measurement Unit	Kernel	Driver
MSR_PKG_ENERGY_STATUS	μJ	1000 µs	1000 µs
MSR_DRAM_ENERGY_STATUS	μJ	1000 µs	1000 µs
MSR_PP0_ENERGY_STATUS	μJ	50 µs	50 µs
MSR_PERF_STATUS (core voltage)	V	150 µs	-

- PLATYPUS utilizes unprivileged energy reading interface - Intel Running Average Power Limit (RAPL)
  - For multiple domain: cores, DRAM, and package
  - The MSR_PP0_ENERGY_STATUS register was mainly used
    - Provides CPU cores energy consumption

Demonstrated RAPL Capabilities (I)

- Measure the power consumption of different CPU core instructions

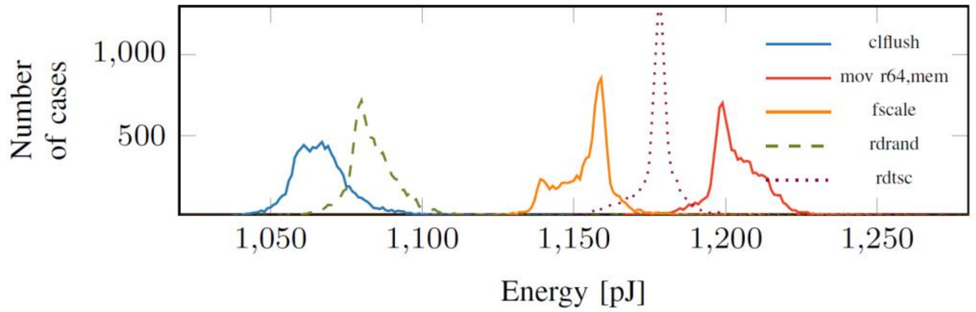


Fig. 1: A histogram of the power consumption of various instructions on the i7-6700K (desktop) system.

- Even inside SGX enclave

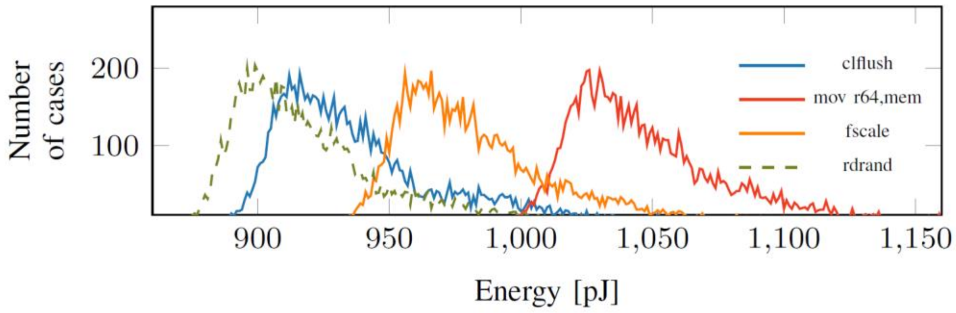


Fig. 2: A histogram of the power consumption of various instructions inside an SGX enclave on our i7-8650U (mobile).

Demonstrated RAPL Capabilities (II)

- Measure the energy consumption of different operands
  - imul instruction

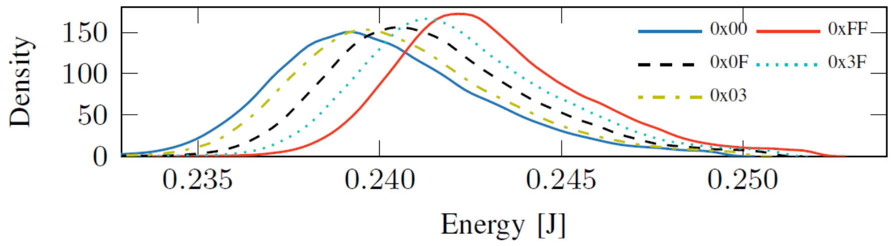


Fig. 3: Measured energy consumption of the `imul` instruction with one operand fixed to 8 and the other varying in its Hamming weight.

- Measure the energy consumption of different load targets
  - Cache hit/miss

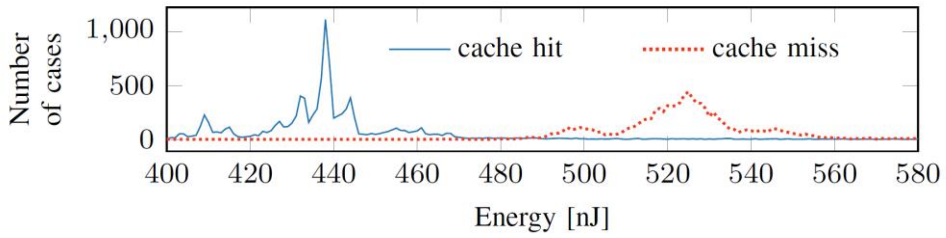


Fig. 6: Using RAPL to distinguish whether the target of a memory load is cached (cache hit) or not (DRAM access).

Source: Lipp, Moritz, et al. "PLATYPUS: Software-based power side-channel attacks on x86." *Symposium on Security and Privacy (SP),* 2021.

PLATYPUS Attacks

- Demonstrate an attack on a cryptographic implementation running in Intel SGX
  - Recovering RSA private keys

- Use Correlation Power Analysis to recover keys from an AES-NI implementation in an SGX enclave
  - Using an unprivileged RAPL attacker

- Break kernel address space layout randomization (KASLR) from user space

Source: Lipp, Moritz, et al. "PLATYPUS: Software-based power side-channel attacks on x86." *Symposium on Security and Privacy (SP),* 2021.

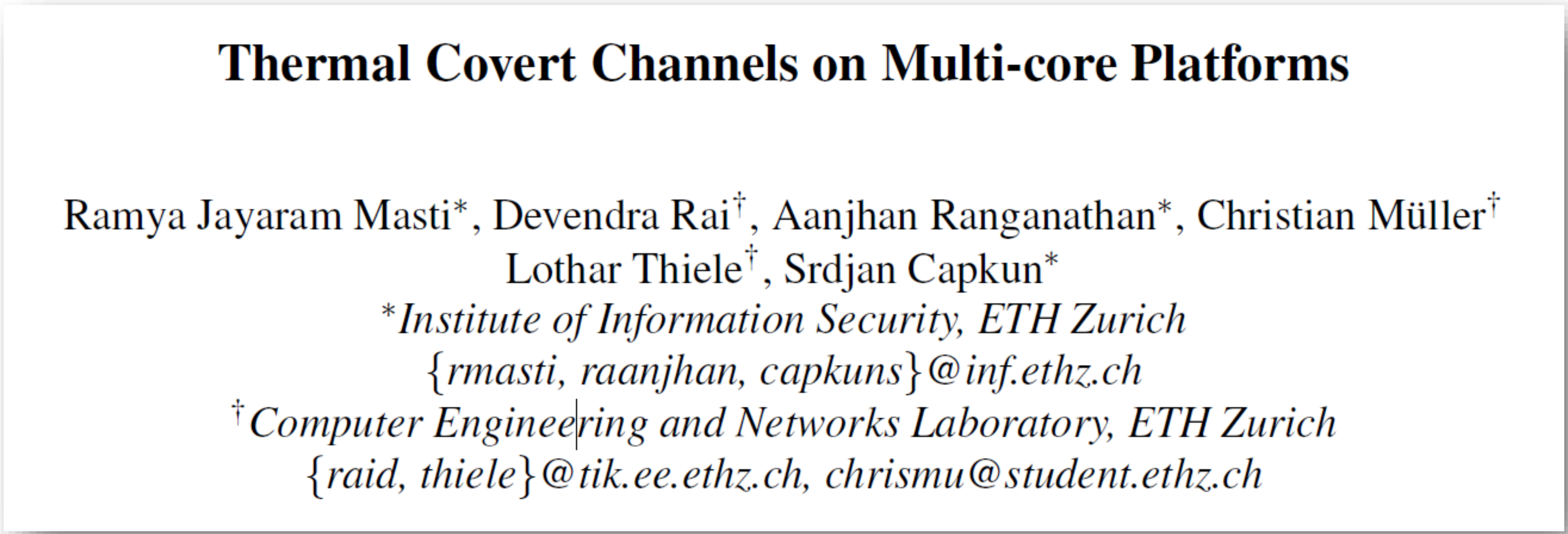
PLATYPUS Mitigations

- Hardware-Level
  - Restricting unprivileged access to RAPL counters
  - Limiting RAPL resolution to make the attacks impractical

- Intel mitigated PLATYPUS by
  1. Disabling unprivileged access to RAPL counters
  2. Adding white noise to reported RAPL readings during SGX

Source: Lipp, Moritz, et al. "PLATYPUS: Software-based power side-channel attacks on x86." *Symposium on Security and Privacy (SP),* 2021.

Thermal Vulnerability Example

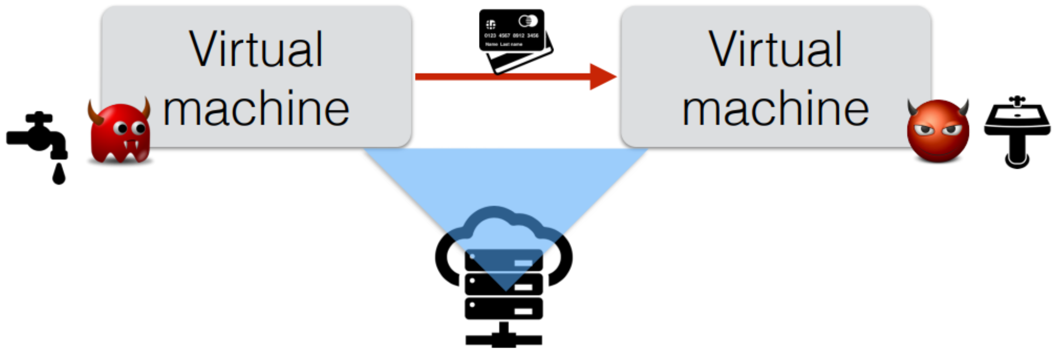


Thermal Covert Channels on Multi-core Platforms

Ramya Jayaram Masti[*], Devendra Rai[†], Aanjhan Ranganathan[*], Christian Müller[†]
Lothar Thiele[†], Srdjan Capkun[*]
[*]*Institute of Information Security, ETH Zurich*
{*rmasti, raanjhan, capkuns*}@*inf.ethz.ch*
[†]*Computer Engineering and Networks Laboratory, ETH Zurich*
{*raid, thiele*}@*tik.ee.ethz.ch, chrismu@student.ethz.ch*

Source: Masti, Ramya Jayaram, et al. "Thermal covert channels on multi-core platforms." USENIX security 2015.

Thermal Covert Channel Overview

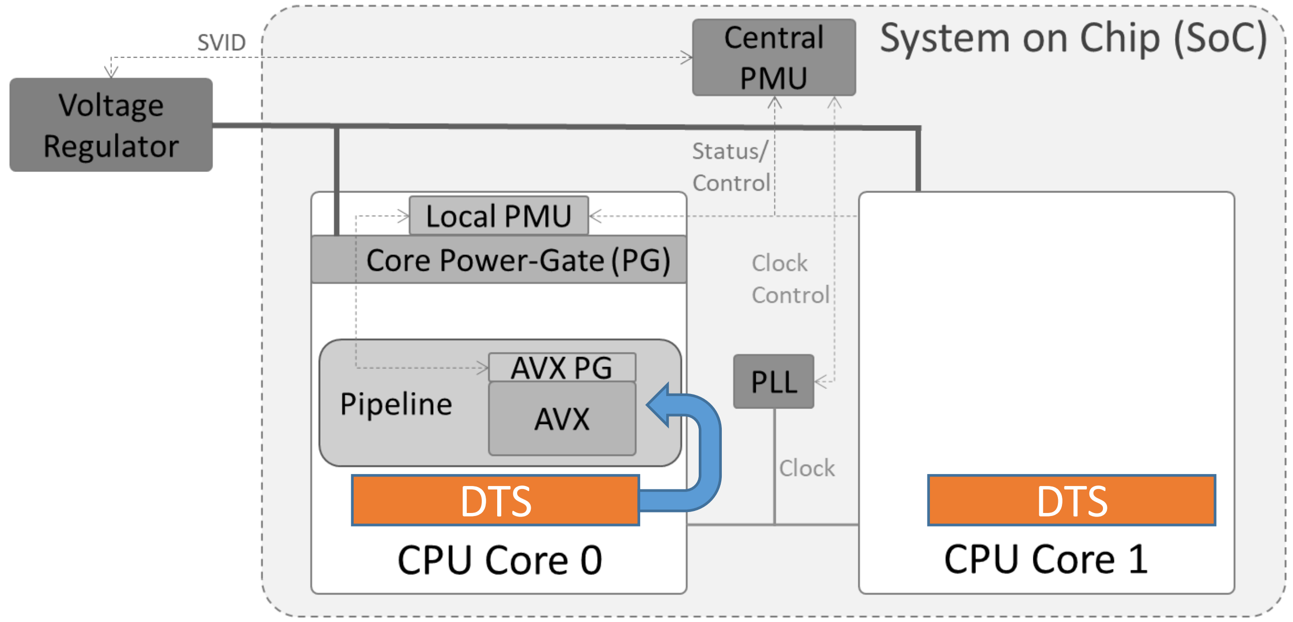


- A covert channel is an attack that creates a capability to transfer information between two processes
  - That are not supposed to be allowed to communicate by the computer security policy

- Thermal Covert Channel demonstrated how thermal channels can be exploited to compromise a system's confidentiality guarantees

Source: https://www.usenix.org/sites/default/files/conference/protected-files/sec15_slides_masti.pdf

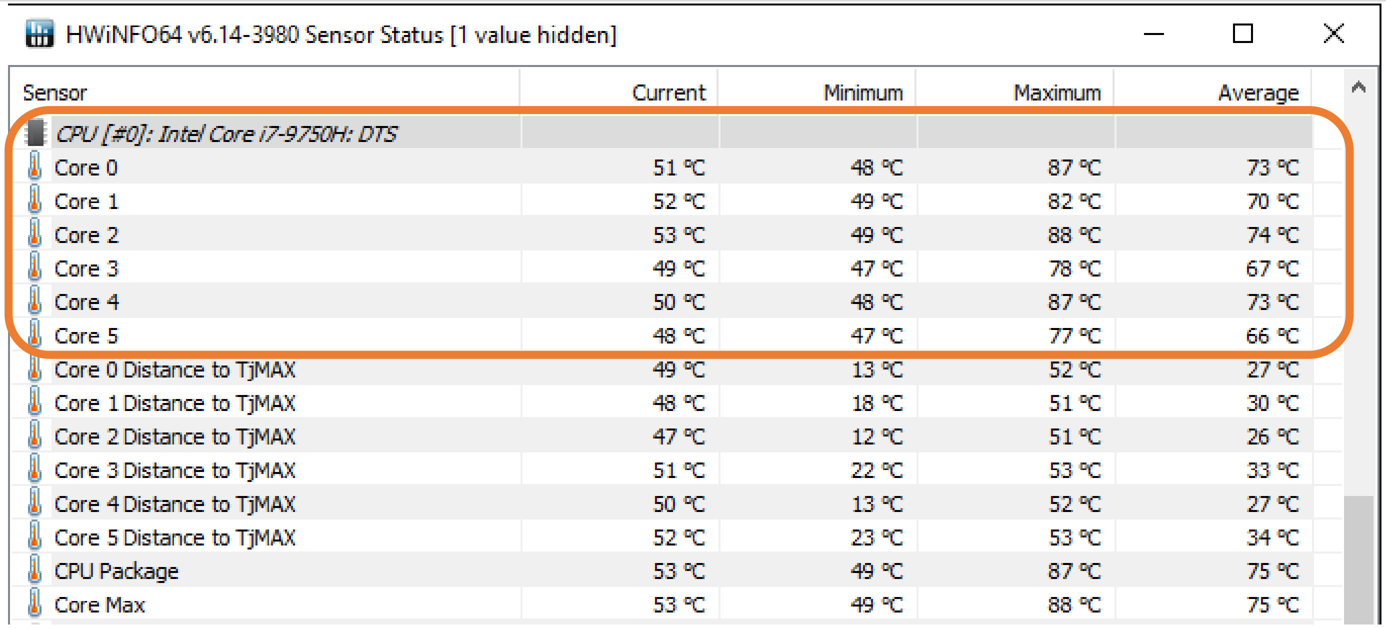
Target Device PM Architecture



- Thermal Covert Channel was demonstrated on Intel Xeon server with 10 cores. The system
  - Includes per-core Digital Thermal Sensors (DTS)
  - Provides SW interface to read core's DTS

Source: https://www.usenix.org/sites/default/files/conference/protected-files/sec15_slides_masti.pdf

DTS SW Interface



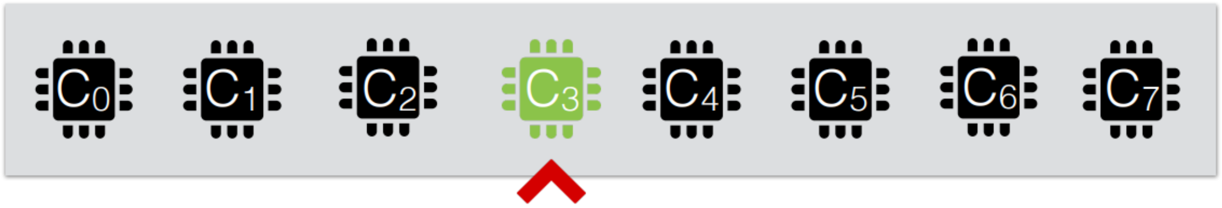
HWiNFO64 v6.14-3980 Sensor Status [1 value hidden]

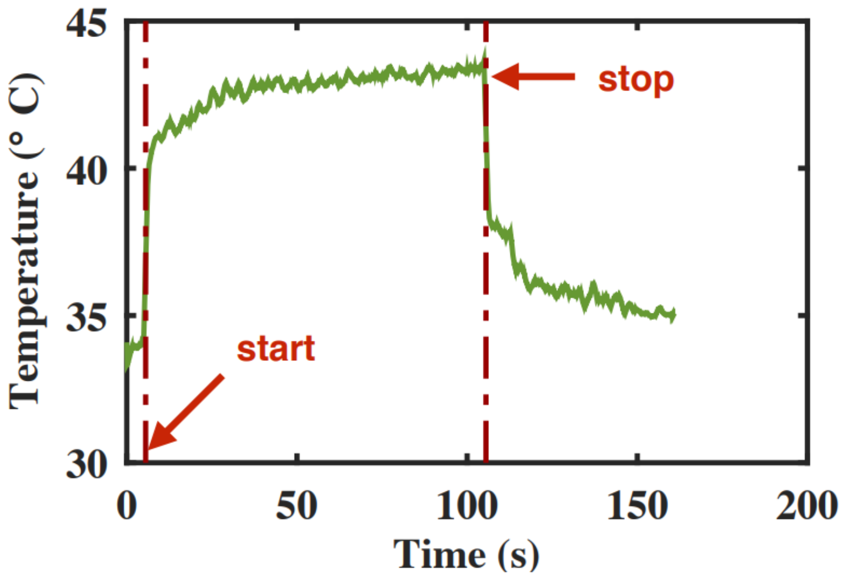
Sensor	Current	Minimum	Maximum	Average
CPU [#0]: Intel Core i7-9750H: DTS				
Core 0	51 °C	48 °C	87 °C	73 °C
Core 1	52 °C	49 °C	82 °C	70 °C
Core 2	53 °C	49 °C	88 °C	74 °C
Core 3	49 °C	47 °C	78 °C	67 °C
Core 4	50 °C	48 °C	87 °C	73 °C
Core 5	48 °C	47 °C	77 °C	66 °C
Core 0 Distance to TjMAX	49 °C	13 °C	52 °C	27 °C
Core 1 Distance to TjMAX	48 °C	18 °C	51 °C	30 °C
Core 2 Distance to TjMAX	47 °C	12 °C	51 °C	26 °C
Core 3 Distance to TjMAX	51 °C	22 °C	53 °C	33 °C
Core 4 Distance to TjMAX	50 °C	13 °C	52 °C	27 °C
Core 5 Distance to TjMAX	52 °C	23 °C	53 °C	34 °C
CPU Package	53 °C	49 °C	87 °C	75 °C
Core Max	53 °C	49 °C	88 °C	75 °C

- Thermal Covert Channel utilizes DTS reading interface
  - All cores' DTS are exposed using the CoreTemp [1] kernel module on Linux systems
  - There are multiple other tools (e.g., HWiNFO [2]) that report the per-core DTS

[1] CoreTemp. http://www.alcpu.com/CoreTemp/ [2] HWiNFO https://www.hwinfo.com/

Demonstrated DTS Capabilities (I)

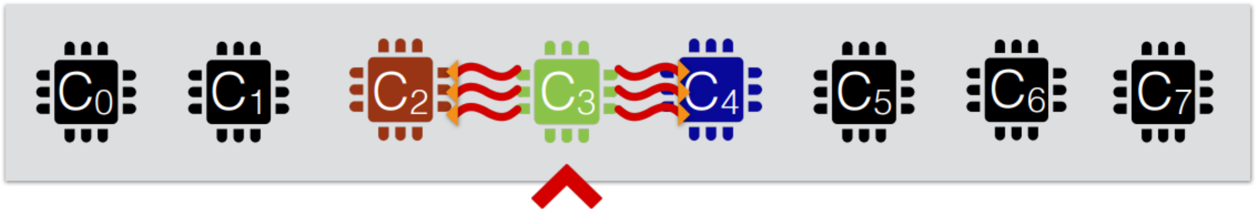




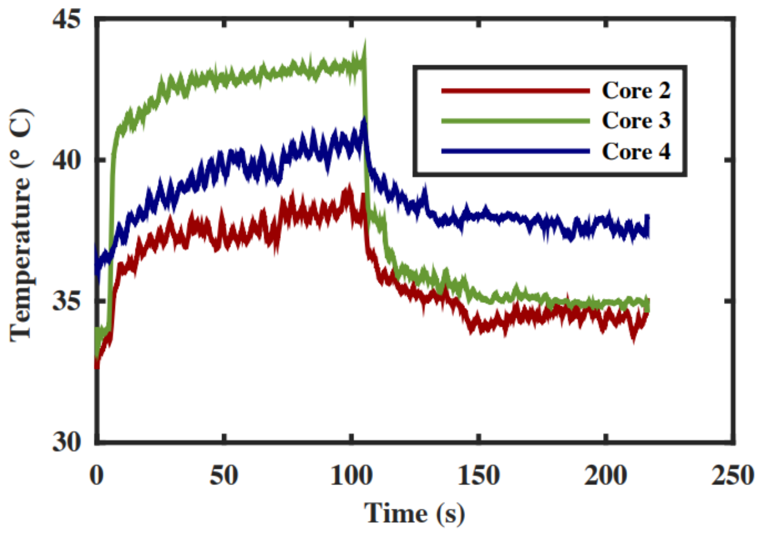
- When starting/stopping a high computational-intensity application for 100us on core 3
    - The temperature raises/fails exponentially

Source: https://www.usenix.org/sites/default/files/conference/protected-files/sec15_slides_masti.pdf

Demonstrated DTS Capabilities (II)



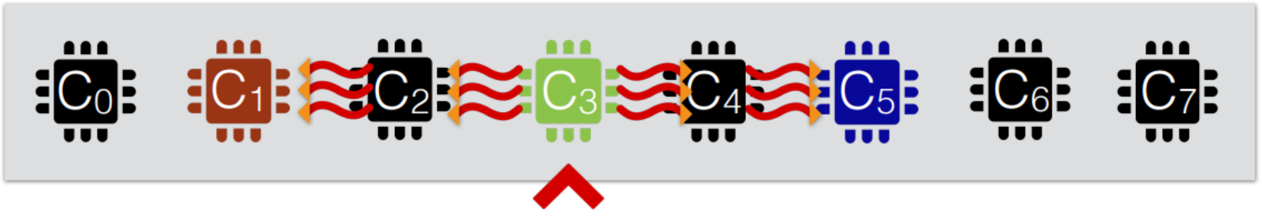
1-hop neighbours

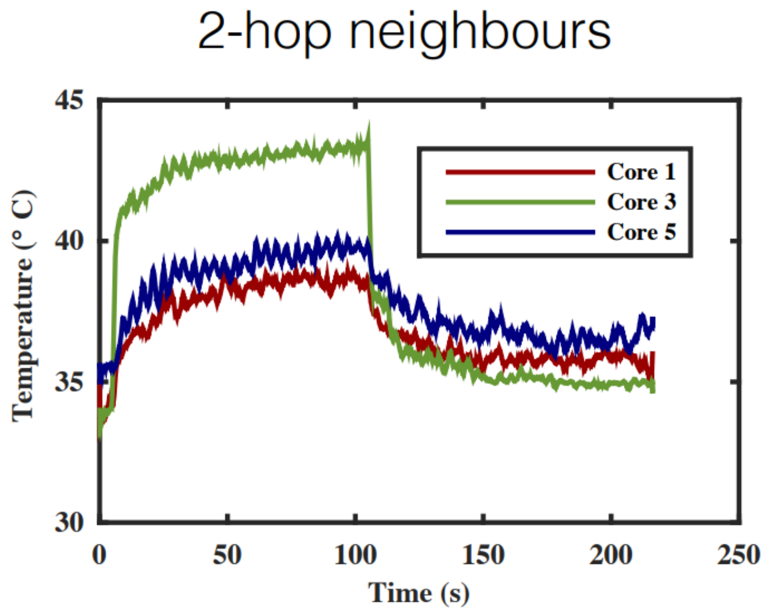


- The heat propagates to 1-hop neighboring cores

Source: https://www.usenix.org/sites/default/files/conference/protected-files/sec15_slides_masti.pdf

Demonstrated DTS Capabilities (III)



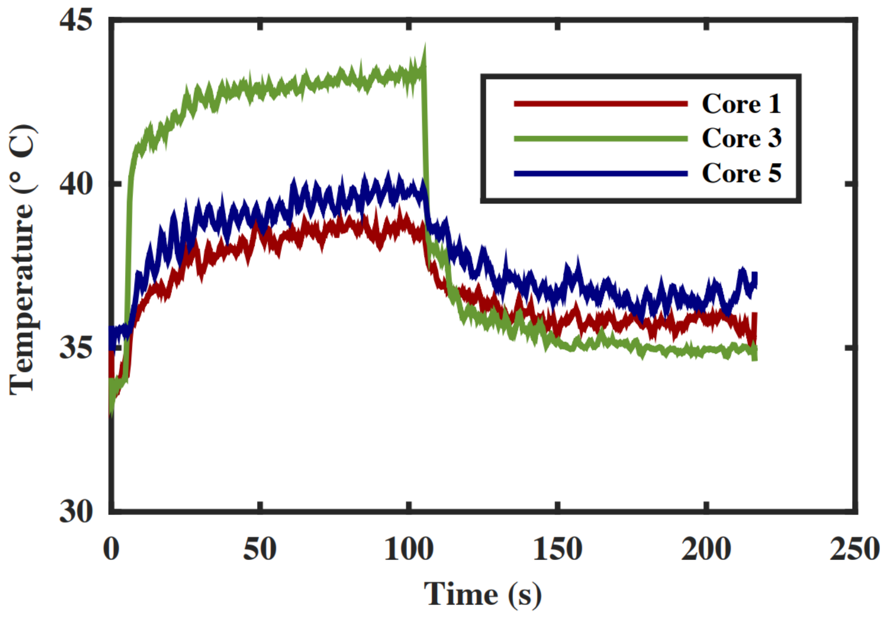


- The heat propagates even to 2-hops neighboring cores

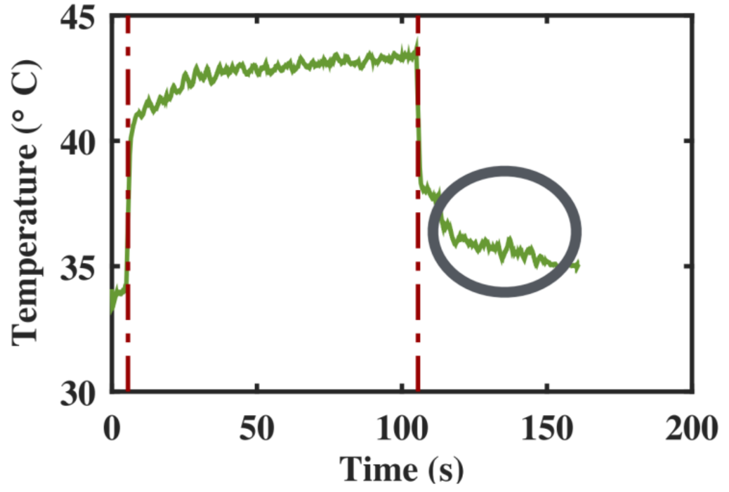
Source: https://www.usenix.org/sites/default/files/conference/protected-files/sec15_slides_masti.pdf

Demonstrated DTS Capabilities (IV)

- Spatial effect: Heat propagates and affects neighboring cores

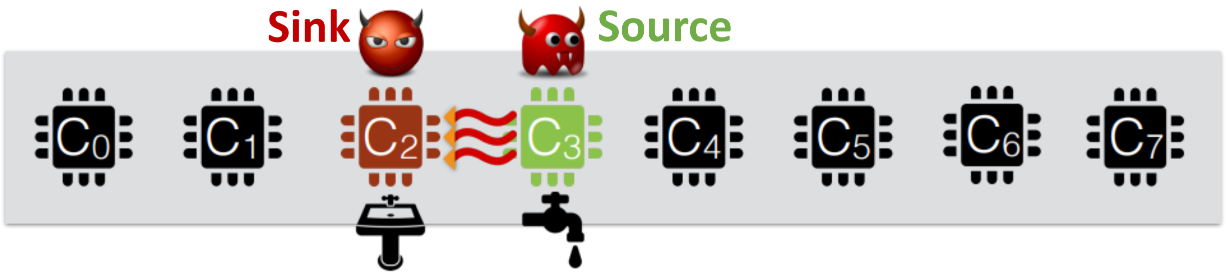


- Temporal effect: There is remnant heat even after computation is completed



Source: https://www.usenix.org/sites/default/files/conference/protected-files/sec15_slides_masti.pdf

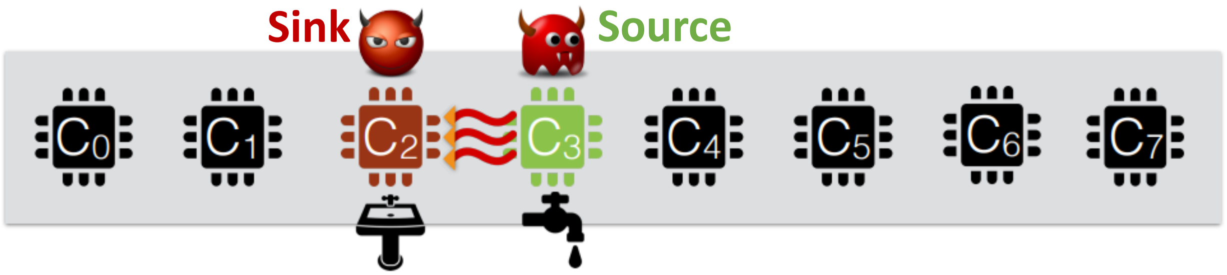
Two Cores Thermal Covert Channel



- Two malicious applications Source/Sink running on cores 3/2
  - The source application is a decryption app that has access to a secure data (credit card details) but *without* internet access permission
  - The sink is a simple (weather) application that can read DTS temperature *with* internet access permission

Source: https://www.usenix.org/sites/default/files/conference/protected-files/sec15_slides_masti.pdf

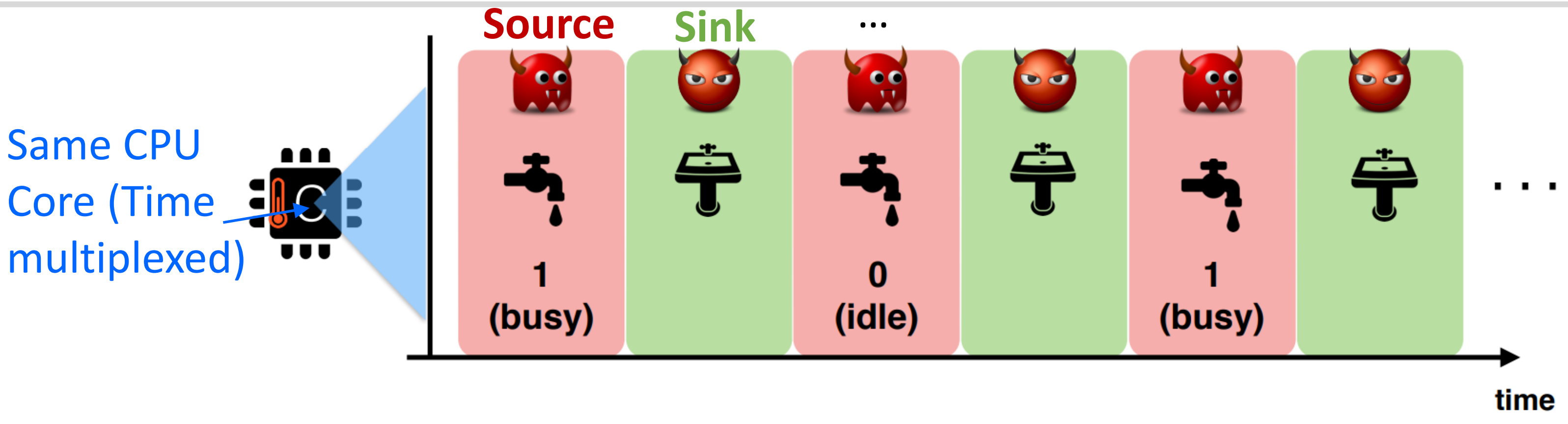
Two Cores Thermal Covert Channel



- The Source/Sink can create a thermal-based covert channel
  - The Source can send a secure data to the Sink
    - The source sends 1 by performing a computation
    - The source sends 0 by remaining idle
  - The Sink can send the secure data to a remote attacker

Source: https://www.usenix.org/sites/default/files/conference/protected-files/sec15_slides_masti.pdf

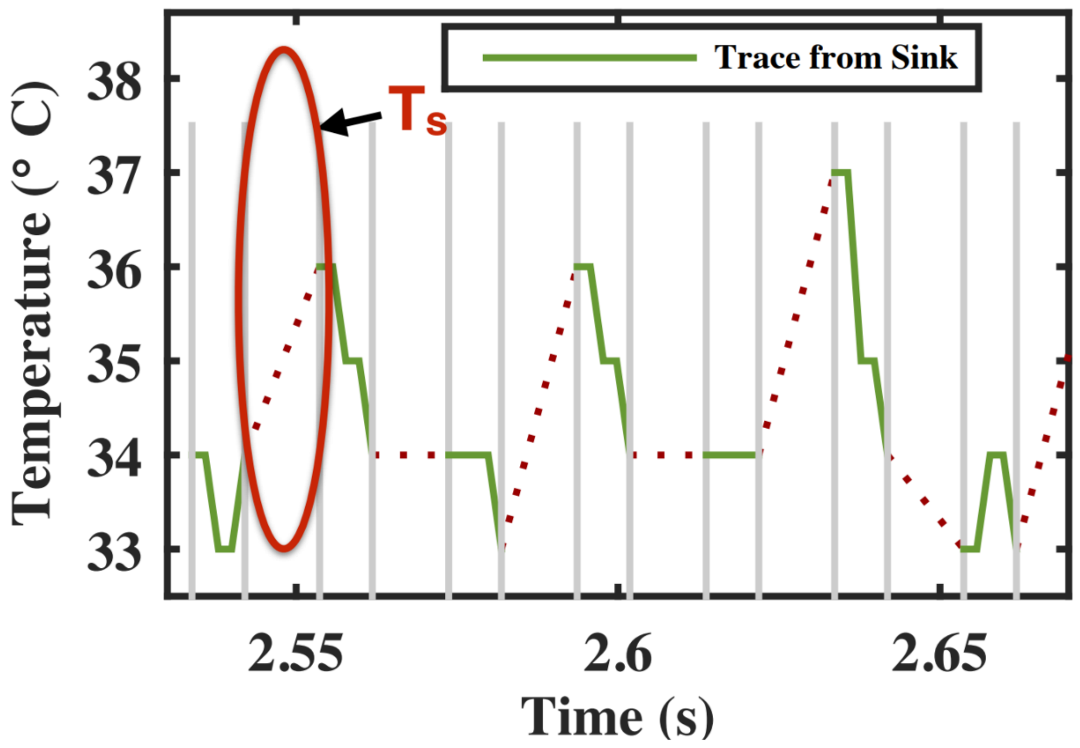
Same Core Thermal Covert Channel



- Two malicious apps Source/Sink run on the same CPU core
  - The source application is a decryption app that has access to a secure data (credit card details) but *without* internet access permission
  - The sink is a simple (weather) application that can read DTS temperature *with* internet access permission
- The Source/Sink can create a thermal-based covert channel

Source: https://www.usenix.org/sites/default/files/conference/protected-files/sec15_slides_masti.pdf

Thermal Covert Channel – Same Core

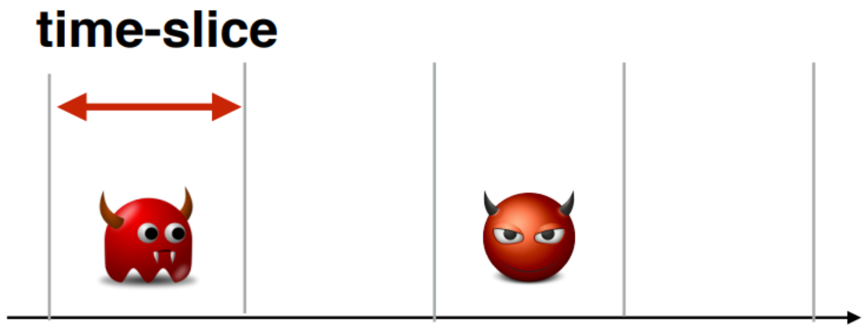


- The source sends
  - 1 by performing a computation
  - 0 by remaining idle
- The Sink can decode the bit by leveraging the remanence heat after the execution on the Source in the same core

Source: https://www.usenix.org/sites/default/files/conference/protected-files/sec15_slides_masti.pdf

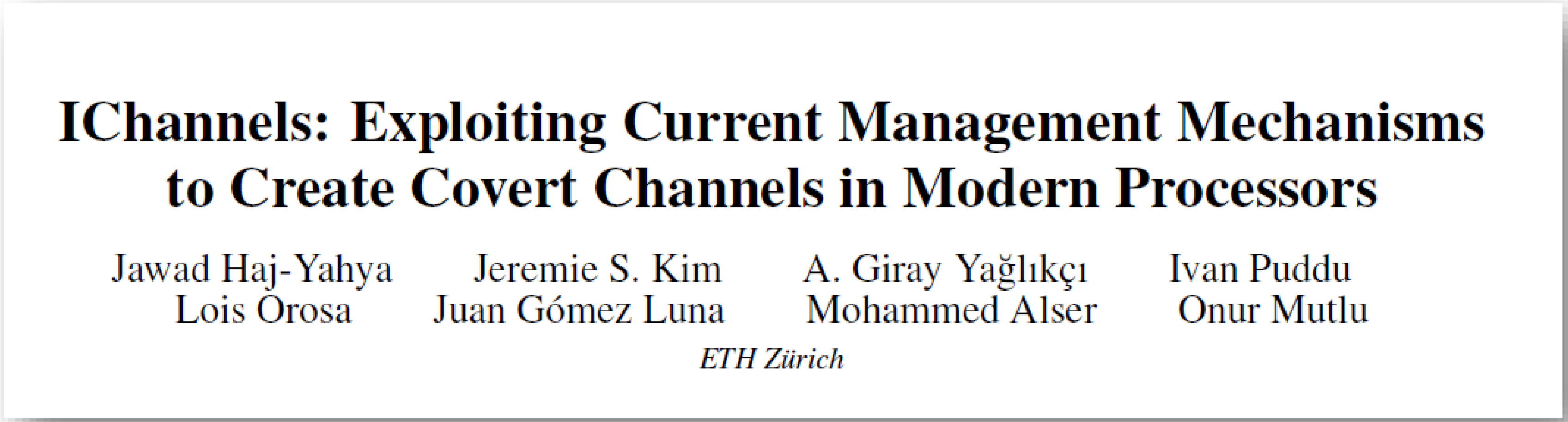
Thermal Covert Channel Mitigations

- Separate the processes
  - Temporally: allow the sink to execute only after the remanence heat decays
  - Spatially: separate the processes far from each other (3 or 4 hops)
- The separation can result in resource wasting
  - Maybe apply in a secure mode only
- Restrict application access to thermal sensors



Source: https://www.usenix.org/sites/default/files/conference/protected-files/sec15_slides_masti.pdf

Current Vulnerability Example


IChannels: Exploiting Current Management Mechanisms to Create Covert Channels in Modern Processors

Jawad Haj-Yahya Jeremie S. Kim A. Giray Yağlıkçı Ivan Puddu
Lois Orosa Juan Gómez Luna Mohammed Alser Onur Mutlu
ETH Zürich

Haj-Yahya, Jawad, et al. "IChannels: exploiting current management mechanisms to create covert channels in modern processors." *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*

IChannels

Exploiting Current Management Mechanisms to Create Covert Channels in Modern Processors

Jawad Haj-Yahya

Jeremie S. Kim   *A. Giray Yağlıkçı*   *Ivan Puddu*   *Lois Orosa*

Juan Gómez Luna   *Mohammed Alser*   *Onur Mutlu*

ISCA 2021

SAFARI   ETH zürich

Executive Summary

Problem: Current management mechanisms throttle instruction execution and adjust voltage/frequency to accommodate power-hungry instructions (PHIs).
These mechanisms may compromise a system's confidentiality guarantees

Goal:
1. Understand the throttling side-effects of current management mechanisms
2. Build high-capacity covert channels between otherwise isolated execution contexts
3. Practically and effectively mitigate each covert channel

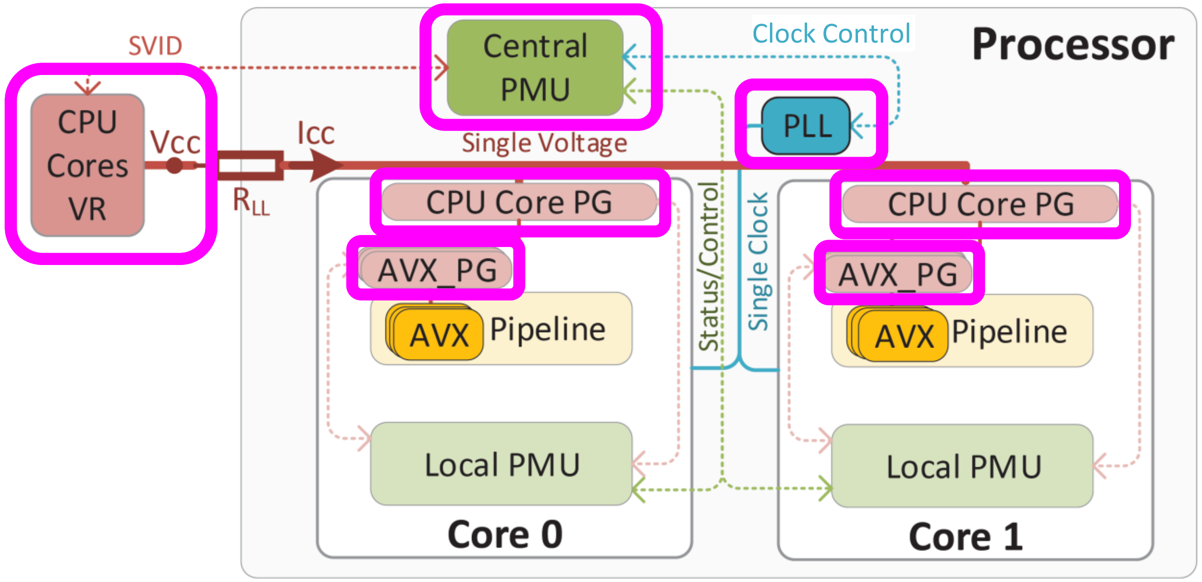
Characterization: Variable execution times and frequency changes due to running PHIs
We observe five different levels of throttling in real Intel systems

IChannels: New covert channels that exploit side-effects of current management mechanisms
- On the same hardware thread
- Across co-located Simultaneous Multi-Threading (SMT) threads
- Across different physical cores

Evaluation: On three generations of Intel processors, IChannels provides a channel capacity
- 2× that of PHIs' variable latency-based covert channels
- 24× that of power management-based covert channels

Presentation Outline

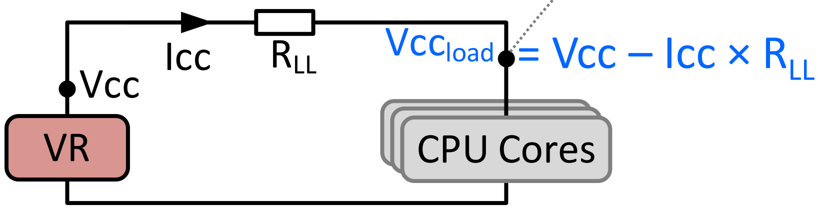
Overview of Client Processor Architectures

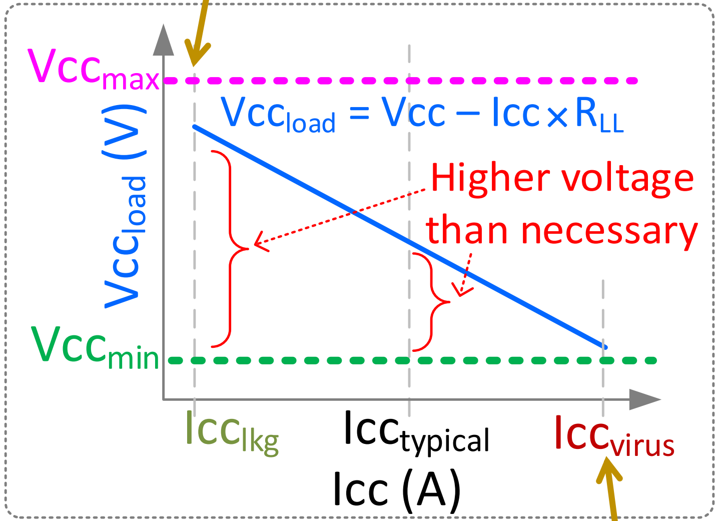


- In many recent processors (e.g., Intel Coffee Lake, Cannon Lake), CPU cores:
  - Share the same voltage regulator (VR) and clock domain
- A central power management unit (PMU) controls:
  - The VR using an off-chip serial voltage identification (SVID) interface
  - The clock phase-locked loop (PLL) using an on-chip interface
- Each CPU core has a power-gate (PG) for the entire core
  - Each SIMD unit (e.g., AVX-256, AVX-512) has a separate PG

Load Voltage and Voltage Guardband

Below the maximum operational voltage (Vcc_{max}) under the lightest load (leakage, Icc_{lkg})





Above the minimum functional voltage (Vcc_{min}) under the most intensive load (power-virus, Icc_{virus})

- The relationship between load voltage (Vcc_{load}), supply voltage (Vcc) and current (Icc) under a given system impedance (R_{LL}) is : $Vcc_{load} = Vcc - Icc \times R_{LL}$
- The PMU adds voltage guardband to Vcc to a level that keeps Vcc_{load} within limits
- For loads with current lower than Icc_{virus}, the voltage drop ($Icc \times R_{LL}$) is smaller than when running a power-virus
  - Results in a higher load voltage Vcc_{load} than necessary
  - Leading to a power loss that increases quadratically with the voltage level

Presentation Outline

Motivation: Limitations of Prior Work

- **NetSpectre** [Schwarz+, ESORICS 2019] exploits the variable execution times of PHIs to create a covert channel. NetSpectre has three limitations:
  - Established only between two execution contexts on the same hardware thread
  - Uses only a single-level throttling side-effect (thread is throttled/unthrottled)
  - Does not identify the true source of throttling
    - Hypothesizes that the throttling is due power-gating of the PHI execution units

- **TurboCC** [Kalmbach+, arXiv 2020] exploits the core frequency throttling when executing PHIs to create cross-core covert channel. TurboCC has two limitations:
  - Focuses only on the slow (milliseconds) side-effect of frequency throttling that happens when executing PHIs at only Turbo frequencies
  - Does not uncover the real reason behind the vulnerability
    - Hypothesizes that the frequency throttling is due to thermal management

Motivation: Limitations of Prior Work

- **NetSpectre** [Schwarz+, ESORICS 2019] exploits the variable execution times of PHIs to create a covert channel. NetSpectre has three limitations:
  - Established only between two execution contexts on the same hardware thread
  - Uses only a single-level throttling side-effect (thread is throttled/unthrottled)
  - Does not identify the true source of throttling
    - Hypothesizes that the throttling is due power-gating of the PHI execution units

Recent works propose limited covert channels and use inaccurate observations

happens when executing PHIs at Turbo frequencies
  - Does not uncover the real reason behind the vulnerability
    - Hypothesizes that the frequency throttling is due to thermal management

Goal

Our goal in this work is to:

1. Experimentally understand the throttling side-effects of current management mechanisms in modern processors to gain several deep insights into how these mechanisms can be abused by attackers

2. Build high-capacity covert channels, IChannels, between otherwise isolated execution contexts located
   - On the same hardware thread
   - Across co-located Simultaneous Multi-Threading (SMT) threads
   - Across different physical cores

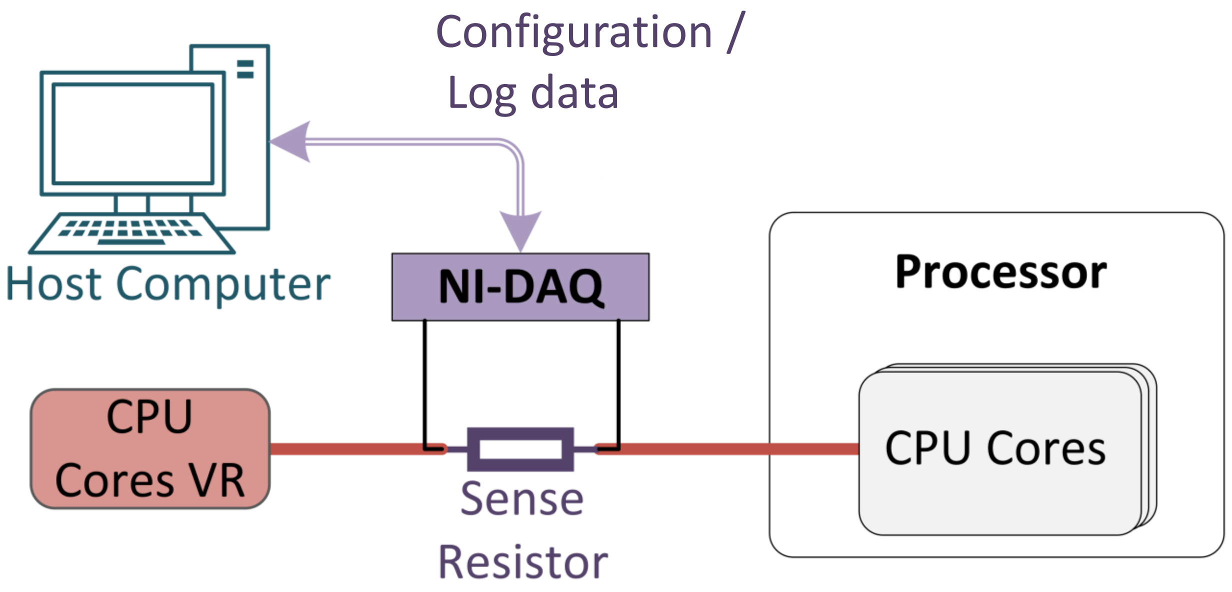
3. Practically and effectively mitigate covert channels caused by current management mechanisms

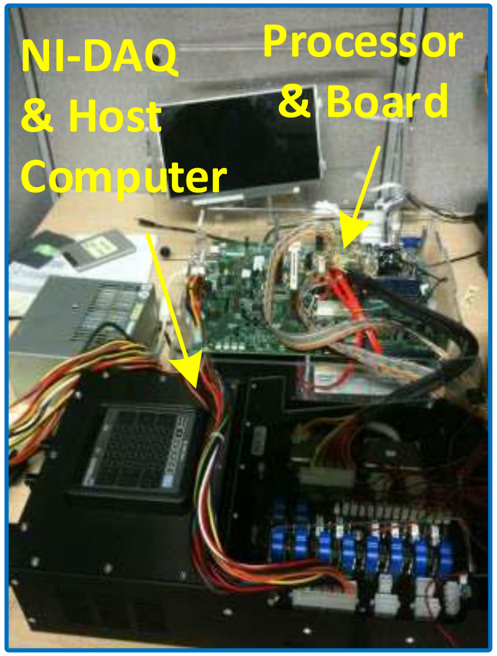
Presentation Outline

Experimental Methodology

- We experimentally study three modern Intel processors
  - Haswell, Coffee Lake, and Cannon Lake

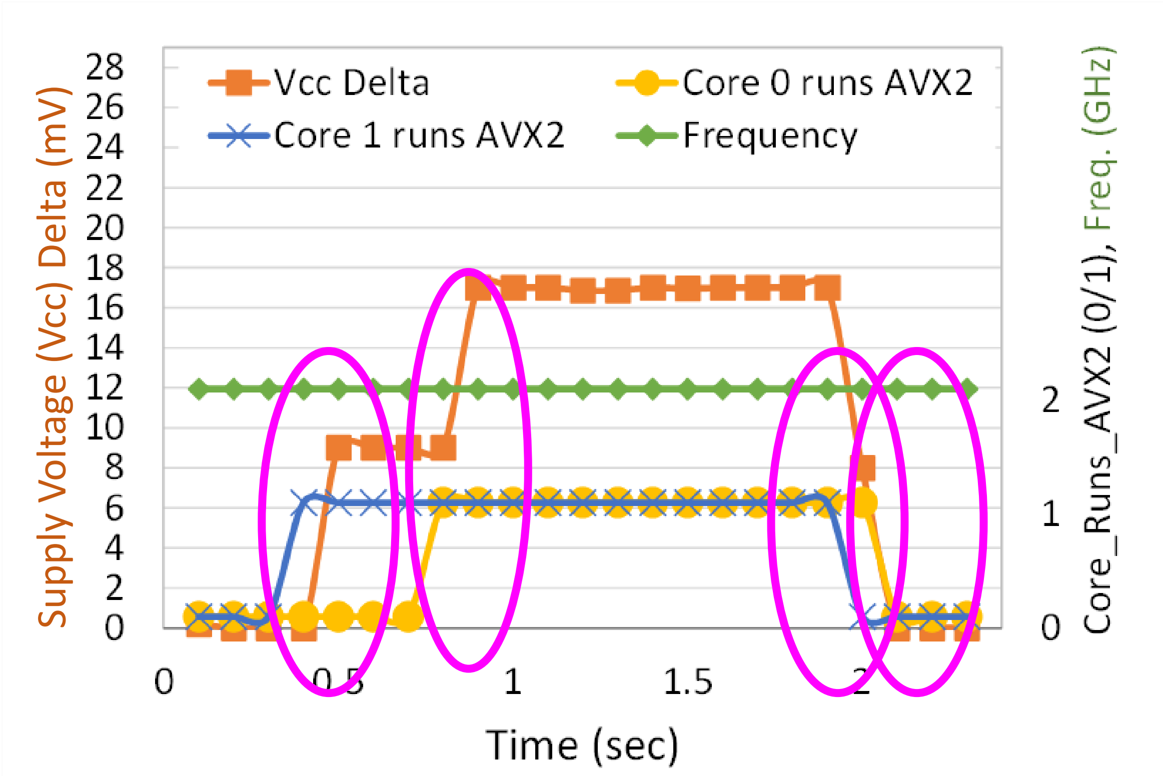
- We measure voltage and current using a Data Acquisition card (NI-DAQ)





Voltage Emergency Avoidance Mechanism

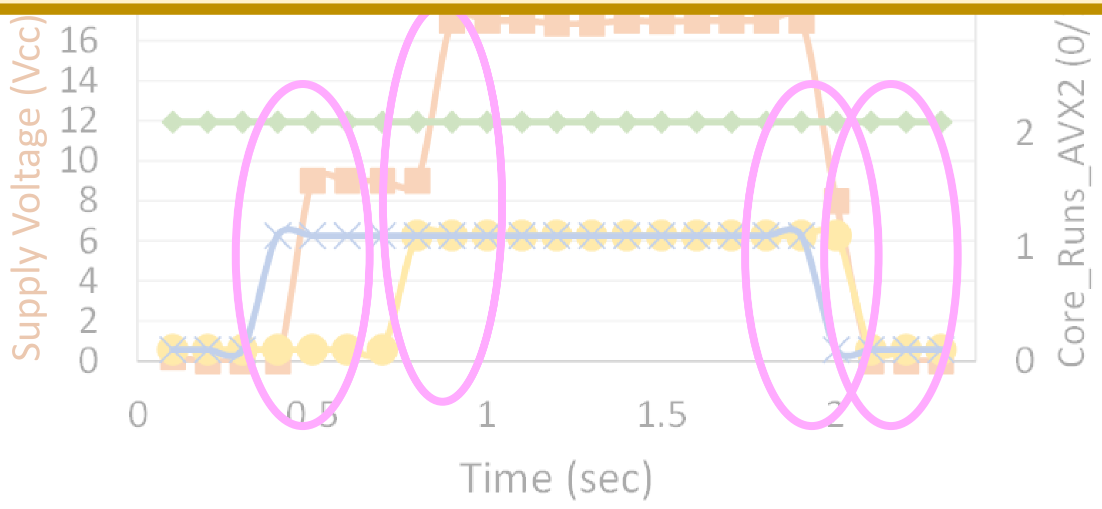
- We study the impact of Power-Hungry Instructions (PHIs) on the CPU core supply voltage (Vcc)
- We track the Vcc change during an experiment on a two-core Coffee Lake system executing code that includes PHI (AVX2) phases
- Vcc increases once a CPU core begins executing AVX2 instructions
  - The more cores executing AVX2 instructions, the higher the Vcc



Voltage Emergency Avoidance Mechanism

- We study the impact of Power-Hungry Instructions (PHIs) on the CPU core supply voltage (Vcc)
- We track the Vcc change during an experiment on a two-core Coffee Lake system executing code that includes PHI (AVX2) phases
- Vcc increases once a CPU core begins executing AVX2 instructions

Voltage emergency avoidance mechanism prevents the core voltage from dropping below the minimum operational voltage limit when executing PHIs



Icc_{max} and Vcc_{max} Limit Protection Mechanisms

- Systems:
  - A single-core Coffee Lake desktop CPU operating at Turbo frequencies (4.9 GHz and 4.8 GHz)
  - A two-core Cannon Lake mobile CPU operating at Turbo frequencies (3.1 GHz and 2.2 GHz)
- Workloads (Non-AVX and AVX2) while measuring current and voltage

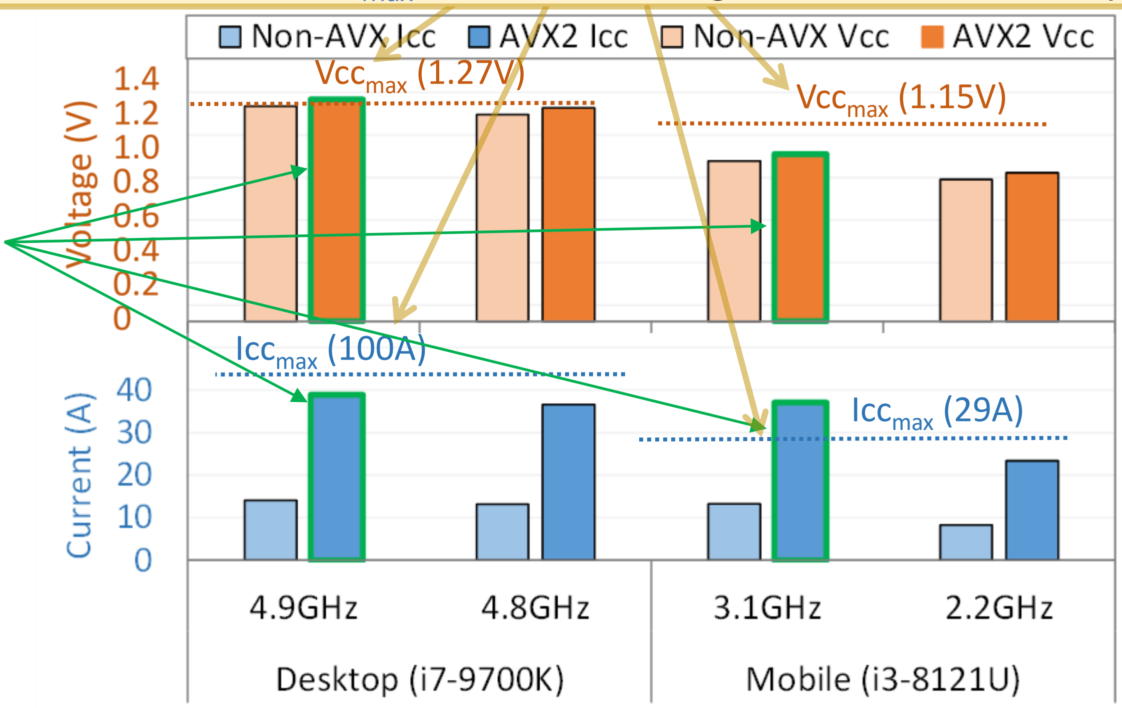
For both **desktop** frequencies, the current (Icc) is below the system limit (Icc_{max})

Vcc will exceed the voltage limit (Vcc_{max}) when executing AVX2 code at a frequency of 4.9 GHz

For both **mobile** frequencies, the voltage (Vcc) is below the system limit (Vcc_{max})

Icc will exceed the current limit (Icc_{max}) when executing AVX2 code at a frequency of 3.1 GHz

The bars with green borders are **projected**



Icc_{max} and Vcc_{max} Limit Protection Mechanisms

- Systems:
  - A single-core Coffee Lake desktop CPU operating at Turbo frequencies (4.9GHz and 4.8GHz)

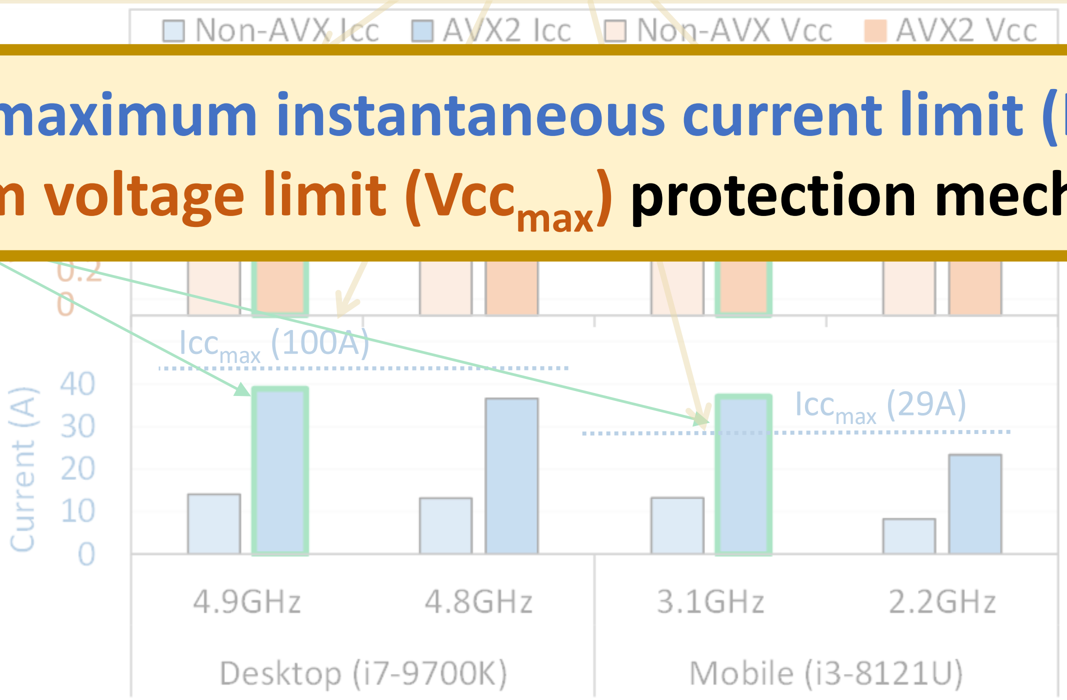
Contrary to the state-of-the-art work's hypothesis:

The core frequency reduction that directly follows the execution of PHIs at the Turbo frequency is not due to thermal management

Icc will exceed the current limit (Icc_{max}) when executing AVX2 code at a frequency of 3.1 GHz

It is due to maximum instantaneous current limit (Icc_{max}) and maximum voltage limit (Vcc_{max}) protection mechanisms

are **projected**



AVX Throttling is Not Due to Power Gating

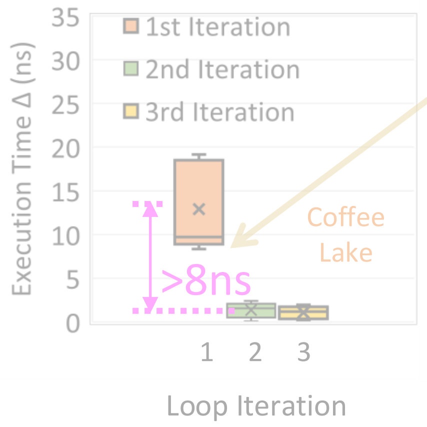
- We study the time it takes to open the AVX power-gate of Coffee Lake
  - By comparing it to Haswell system, which doesn't have an AVX power-gate
- When running AVX2 instructions in a loop
  - Consisting of 300 AVX (VMULPD) instructions that use registers

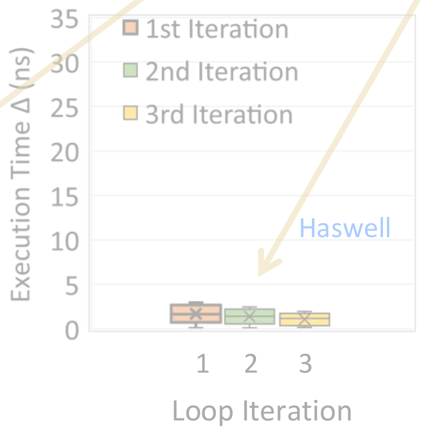
The first iteration of the loop running on Coffee Lake is > 8ns longer than the other two iterations

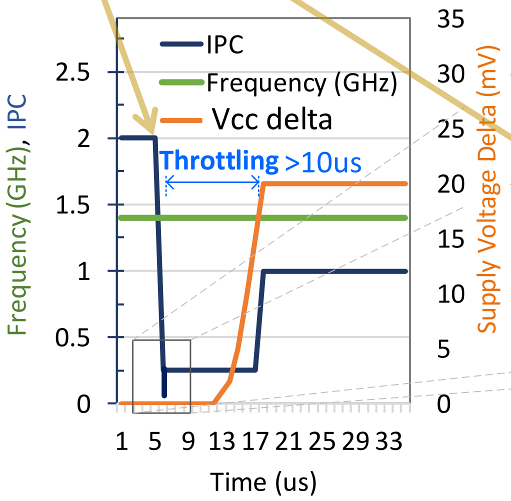
For the Haswell processor all iterations have nearly the same latency

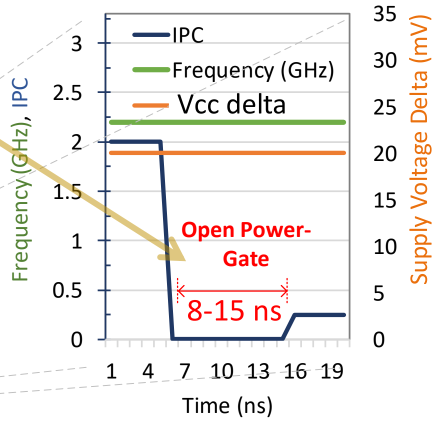
AVX power-gating feature has approximately 8–15 ns of wake-up latency

About 1% of the total throttling time when executing PHIs (>10us)









AVX Throttling is Not Due to Power Gating

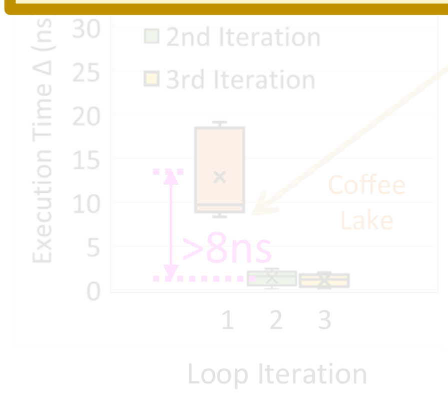
- We study the time it takes to open the AVX power-gate of Coffee Lake
  - By comparing it to Haswell system, which doesn't have an AVX power-gate
- When running AVX2 instructions in a loop

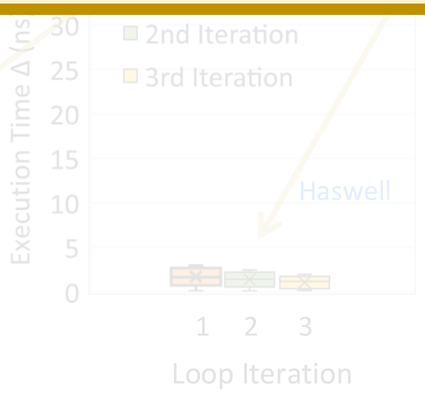
Contrary to the state-of-the-art work's hypothesis:

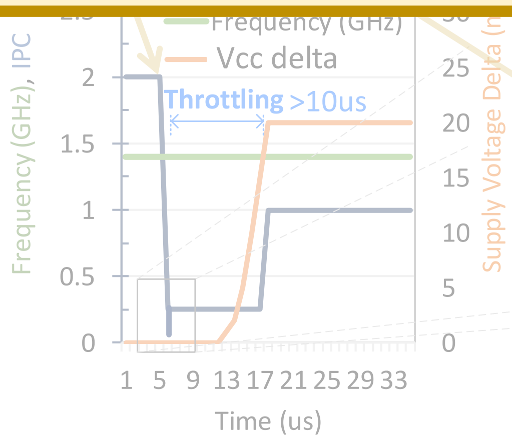
Power-gating AVX execution units accounts for only ~0.1% of the total throttling time observed when executing PHIs

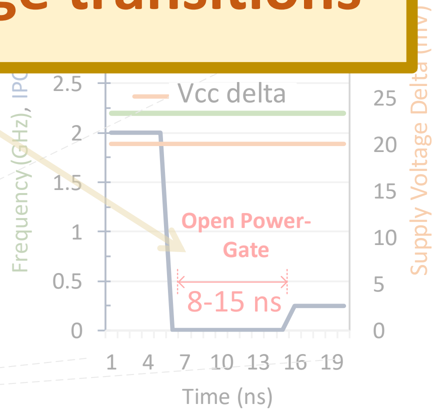
About 1% of the total throttling time when executing PHIs (>10us)

The majority of the throttling time is due to voltage transitions





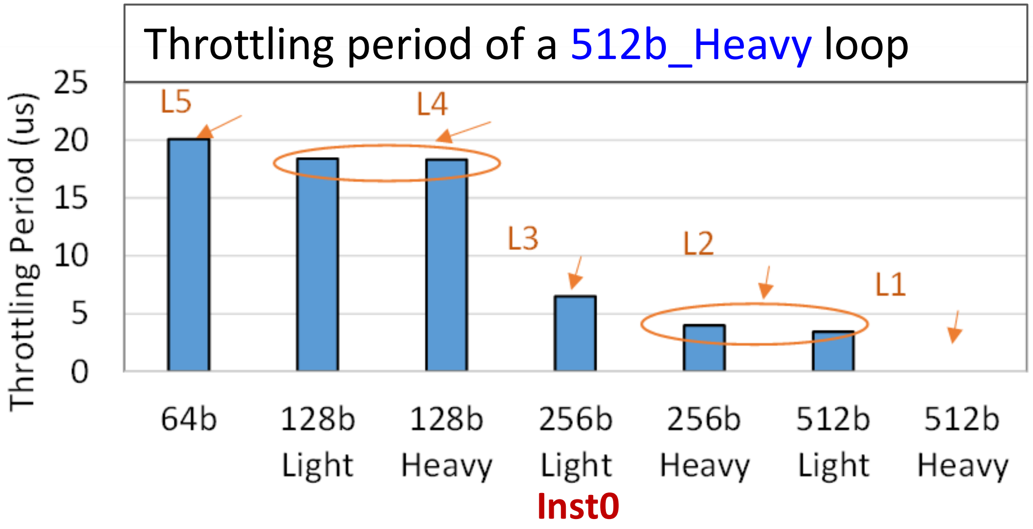




Multi-level Throttling

- We execute one of 7 instruction types in a loop followed by a 512b_Heavy loop
  - Inst0: 64b, 128b_Light, 128b_Heavy, 256b_Light, 256b_Heavy, 512b_Light, and 512b_Heavy
  - Heavy instructions: require the floating-point unit or any multiplication
- The throttling period of the 512b_Heavy loop increases when
  - The computational intensity of the instructions executed in the preceding loop decreases
- The lower the instructions' computational intensity in the preceding loop, the lower the applied voltage guardband to this instruction
  - Hence, the 512b_Heavy loop requires more time to increase the voltage to the required level
- We observe at least five throttling levels (L1–L5) corresponding to the computational intensity of Inst0 instruction types

```
Inst0 loop
...
T0:
512b-Heavy loop
```

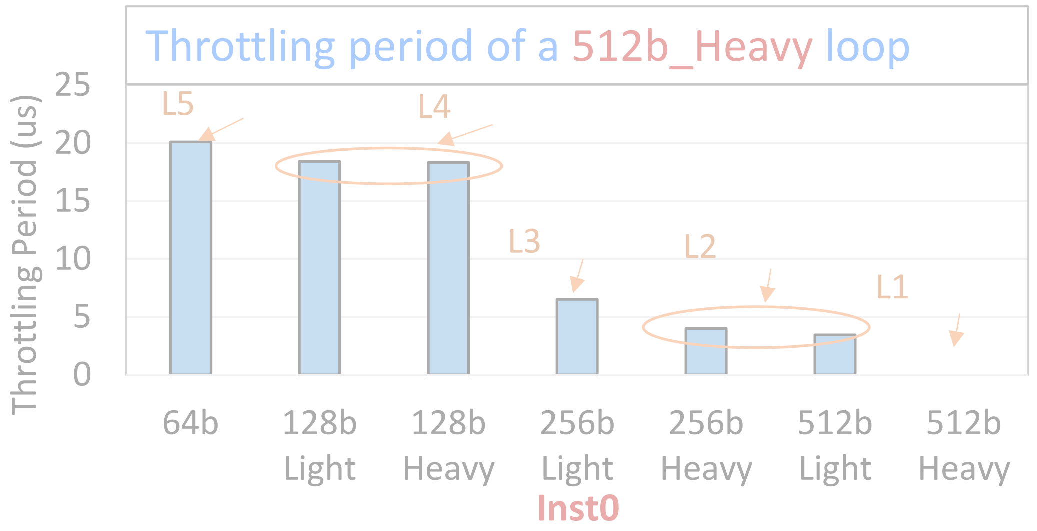


Multi-level Throttling

- We execute one of 7 instruction types in a loop followed by a 512b_Heavy loop
  - Inst0: 64b, 128b_Light, 128b_Heavy, 256b_Light, 256b_Heavy, 512b_Light, and 512b_Heavy
  - Heavy instructions: require the floating-point unit or any multiplication
- The throttling period of the 512b_Heavy loop increases when
  - The computational intensity of the instructions executed in the preceding loop decreases

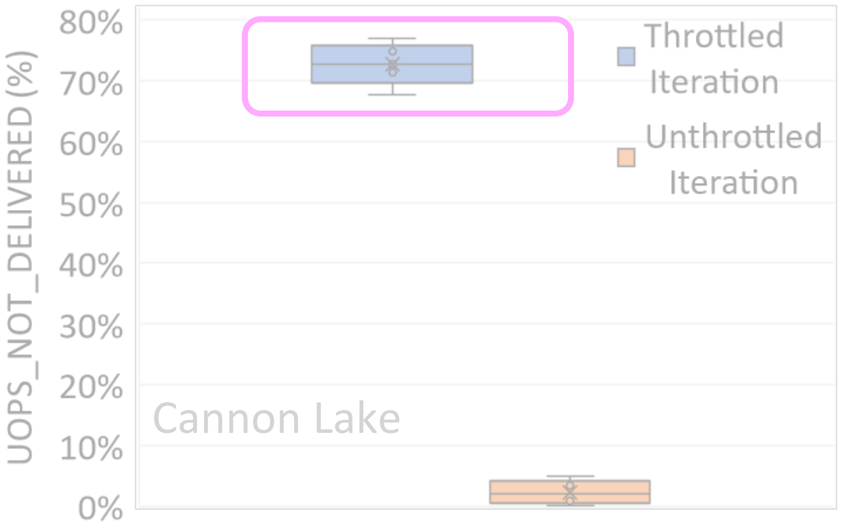
Current management mechanisms result in a multi-level throttling period depending on the computational intensity of the PHIs

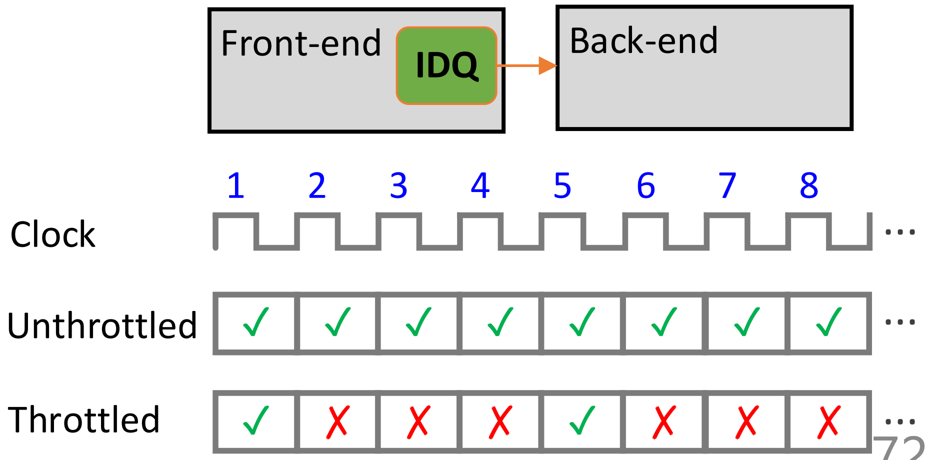
Inst0 loop
…
T0:
512b-Heavy loop



Throttling Affects SMT Threads

- We study the source of throttling and its microarchitectural impact
- We track the number of micro-operations (uops) that the core pipeline delivers from the front-end to the back-end during throttled and non-throttled AVX2 loops
- The front-end does not deliver any uop in approximately three-quarters (~75%) of the core cycles even though the back-end is not stalled
- The core uses a throttling mechanism that limits the number of uops delivered from the front-end to the back-end during a certain time window
- We found that this throttling mechanism affects both threads in Simultaneous Multi-Threading (SMT)





Throttling Affects SMT Threads

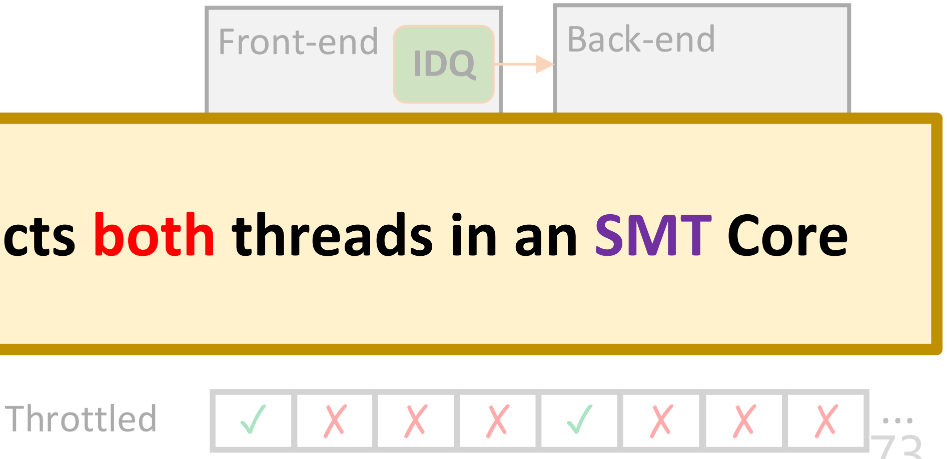
- We study the source of throttling and its microarchitectural impact

Contrary to the state-of-the-art work's hypothesis:

The 4× core IPC reduction that directly follows the execution of PHIs is not due to reduced core clock frequency of 4×

- The core uses a throttling mechanism that limits the number of uops delivered from the front-end to the back-end during a certain time window

It is rather because the core blocks the front-end to back-end uop delivery during 75% of the time



This throttling mechanism affects both threads in an SMT Core

Presentation Outline

IChannels Covert Channels

- Threat model consists of two malicious user-level attacker applications, sender and receiver, which cannot communicate through overt channels
- We build three high-throughput covert channels between sender and receiver that exploit throttling side-effects of current management mechanisms
  - On the same hardware thread
  - Across SMT threads, and
  - Across cores
- Each covert channel sends 2 bits from Sender to Receiver in every transaction
  - Each covert channel should wait for reset-time (~650us) before starting a new transaction
  - We demonstrate the covert channels on real Intel Coffee Lake and Cannon Lake system

Sender

```
case (send_bits[i+1:i])
  00: 128b_Heavy loop()   //L4
  01: 256b_Light_loop()   //L3
  10: 256b_Heavy_loop()   //L2
  11: 512b_Heavy_loop()   //L1
```

Receiver

```
 start = rdtsc
if (same-thread) 512b_Heavy_loop()
if (across-SMT)  64b_loop()
if (across-cores) 128b_Heavy_loop()
 TP = rdtsc - start
 case ( TP )
   L4_range: received_bits[1:0] = 00
   L3_range: received_bits[1:0] = 01
   L2_range: received_bits[1:0] = 10
   L1_range: received_bits[1:0] = 11
```

Covert Channel 1: IccThreadCovert (1/2)

- IccThreadCovert covert channel exploits the side effect of Multi-Throttling-Thread
- Multi-Throttling-Thread: Executing an instruction with high computational intensity results in a throttling period proportional to the difference in voltage requirements of
  - The currently and previously executing instructions

When 512b_Heavy loop is executed, it is first throttled (IPC = 1/4) while ramping the Vcc to accommodate 512b_Heavy

Inst0 loop is throttled (IPC=1/4) while ramping the voltage (Vcc)

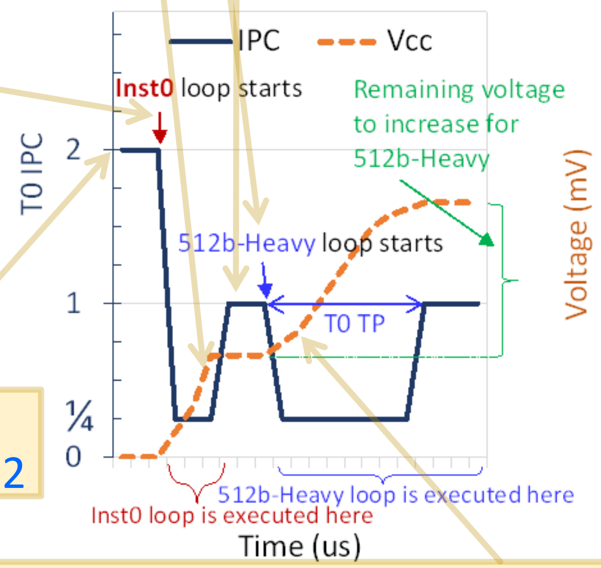
Once the target Vcc is reached, the throttling is stopped (IPC=1)

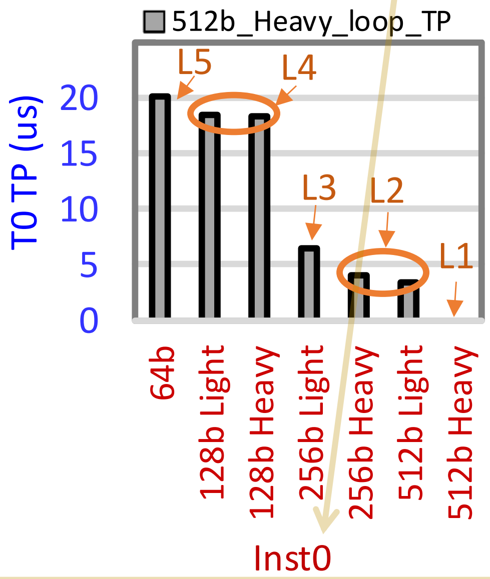
T0 throttling period (TP) dependent on the computational intensity of Inst0 loop

Inst0 loop starts executing with IPC=1

...
T0:

Executes scalar instruction with IPC=2





The remaining voltage required to execute a 512b_Heavy instruction depends on the previous Vcc level that was reached when Inst0 loop was executed

Covert Channel 1: IccThreadCovert (2/2)

Sender

```
case (send_bits[i+1:i])
  00: 128b_Heavy loop()   //L4
  01: 256b_Light_loop()   //L3
  10: 256b_Heavy_loop()   //L2
  11: 512b_Heavy_loop()   //L1
```

Receiver

```
start = rdtsc
if (same-thread) 512b_Heavy_loop()
if (across-SMT)   64b_loop()
if (across-cores) 128b_Heavy_loop()
TP = rdtsc - start
case ( TP )
  L4_range: received_bits[1:0] = 00
  L3_range: received_bits[1:0] = 01
  L2_range: received_bits[1:0] = 10
  L1_range: received_bits[1:0] = 11
```

- **IccThreadCovert** exploits the **Multi-Throttling-Thread** side-effect to build a covert channel between Sender and Receiver:
- The Sender executes a PHI loop with a computational intensity level (L1–L4) depending on the values of two secret bits it wants to send
- The Receiver can infer the two bits sent by the Sender based on the measured TP of the 512b_Heavy loop
  - The higher the power required by the PHI loop executed by the Sender, the shorter the TP experienced by the Receiver will be

Covert Channel 2: IccSMTcovert (1/2)

- IccSMTcovert covert channel exploits the side effect of Multi-Throttling-SMT
- Multi-Throttling-SMT: when a thread is throttled due to executing PHIs, the co-located SMT thread is also throttled
  - We discover that co-located hardware threads are throttled together because the throttling mechanism in the core pipeline blocks the front-end to back-end interface during three-quarters of the TP for the entire core

T1 throttling period (TP) depends on the computational intensity of Inst0 (executed by T0), which determines Vcc level to which the processor needs to increase the supply voltage

T0 starts executing Inst0 loop with IPC=1
T1 starts executing 64b loop with IPC=1

Once the target Vcc is reached, the throttling is stopped (IPC = 1)

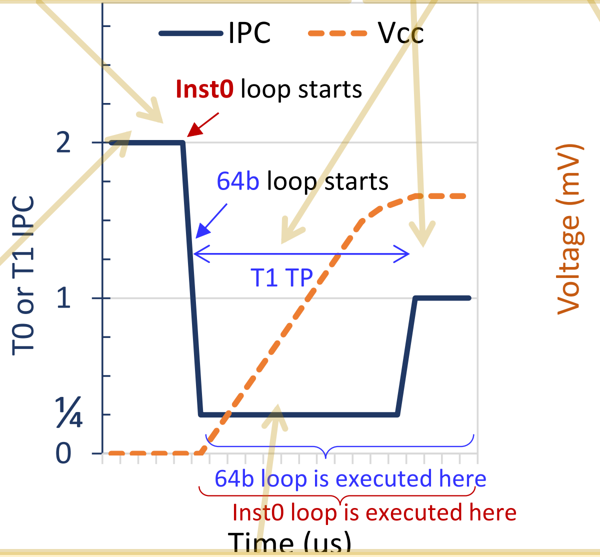
T0 (Core0):

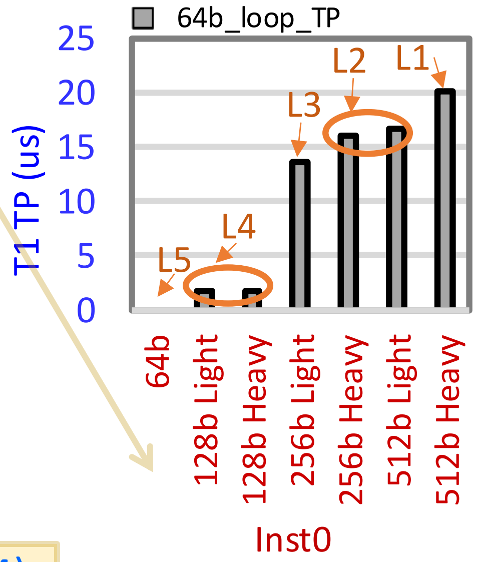
...

T1 (Core0):

...

...





Threads Execute scalar instructions with IPC=2

T0 and T1 loops are throttled (IPC=1/4) while ramping the voltage (Vcc)

Covert Channel 2: IccSMTcovert (2/2)

Sender

```
case (send_bits[i+1:i])
  00: 128b_Heavy loop()   //L4
  01: 256b_Light_loop()   //L3
  10: 256b_Heavy_loop()  //L2
  11: 512b_Heavy_loop()  //L1
```

Receiver

```
 start = rdtsc
if (same-thread) 512b_Heavy_loop()
if (across-SMT)   64b_loop()
if (across-cores) 128b_Heavy_loop()
 TP = rdtsc - start
 case ( TP )
   L4_range: received_bits[1:0] = 00
   L3_range: received_bits[1:0] = 01
   L2_range: received_bits[1:0] = 10
   L1_range: received_bits[1:0] = 11
```

- **IccSMTcovert** exploits the **Multi-Throttling-SMT** side-effect to build a covert channel between Sender and Receiver:
- The Sender executes a **PHI loop** with a computational intensity level (L1–L4) depending on the values of two secret bits it wants to send
- The Receiver can infer the two bits sent by the Sender based on the measured TP of the 64b loop
  - The higher the power required by the PHI loop executed by the Sender, the higher the TP experienced by the Receiver will be

Covert Channel 3: IccCoresCovert (1/2)

- IccCoresCovert covert channel exploits the side effect of Multi-Throttling-Cores
- Multi-Throttling-Cores: when two cores execute PHIs at similar times, the throttling periods (TP) are exacerbated proportionally to the computational intensity of each PHI executed in each core
  - This increase in the TP is because the power management unit (PMU) waits until the voltage transition for core A to complete before starting the voltage transition for core B

T1 TP depends on the computational intensity of Inst0, which determines Vcc level to which the PMU needs to increase the supply voltage before handling T1 voltage transition

T0 and T1 loops are throttled (IPC=1/4)

T1 continues to be throttled since the PMU will not handle T1 voltage transition until T0 voltage target is reached

T0 (Core0):

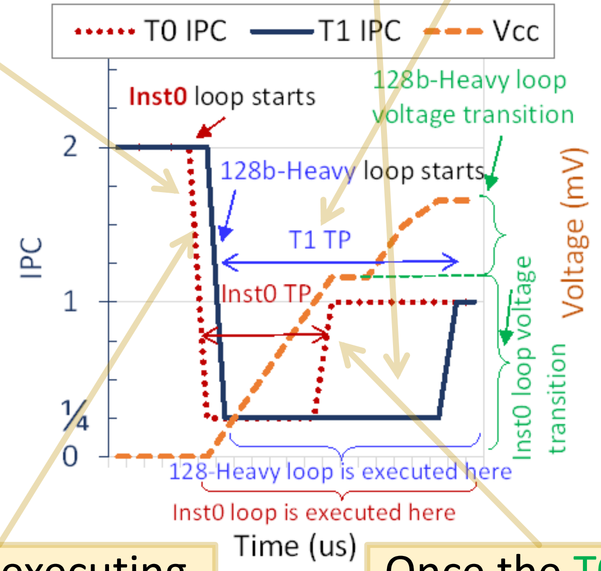
…

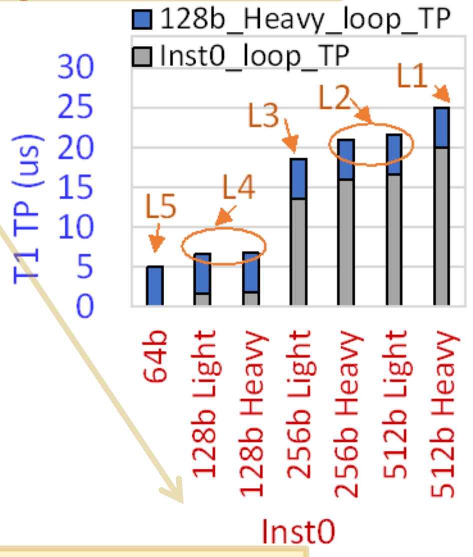
T1 (Core1):

…

)

…





T0/T1 in core0/1 starts executing Inst0/128b-Heavy loop with IPC=1

Once the T0 target Vcc is reached, T0 throttling is stopped (IPC = 1)

Covert Channel 3: IccCoresCovert (2/2)

Sender

```
case (send_bits[i+1:i])
  00: 128b_Heavy loop()   //L4
  01: 256b_Light_loop()   //L3
  10: 256b_Heavy_loop()   //L2
  11: 512b_Heavy_loop()   //L1
```

Receiver

```
start = rdtsc
if (same-thread) 512b_Heavy_loop()
if (across-SMT)   64b_loop()
if (across-cores) 128b_Heavy_loop()
TP = rdtsc - start
case ( TP )
  L4_range: received_bits[1:0] = 00
  L3_range: received_bits[1:0] = 01
  L2_range: received_bits[1:0] = 10
  L1_range: received_bits[1:0] = 11
```

- **IccCoresCovert** exploits the **Multi-Throttling-Cores** side-effect to build a covert channel between Sender and Receiver:
- The Sender executes a PHI loop with a computational intensity level (L1–L4) depending on the values of two secret bits it wants to send
- The Receiver can infer the two bits sent by the Sender based on the measured TP of the 128b_Heavy loop
  - The higher the power required by the PHI loop executed by the Sender, the higher the TP experienced by the Receiver will be

Presentation Outline

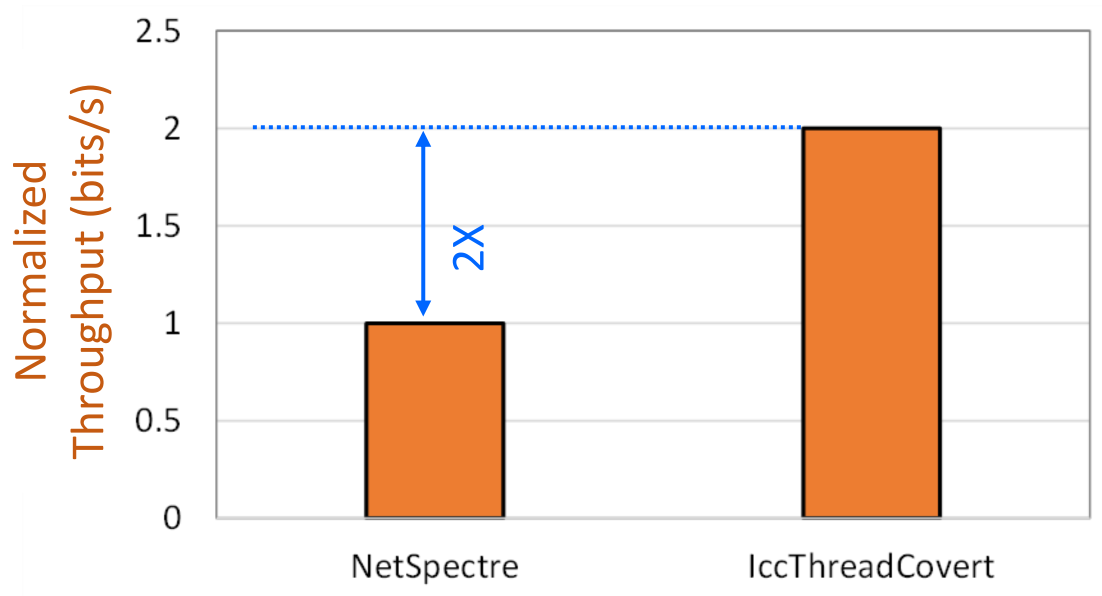
Methodology

- **Framework:** We evaluate IChannels on Coffee Lake and Cannon Lake
  - We test IccThreadCovert and IccCoresCovert on both processors, but we test IccSMTcovert only on Cannon Lake as Coffee Lake does not support SMT

- **Workloads**: Proof-of-concept codes of each of the three IChannels covert channels

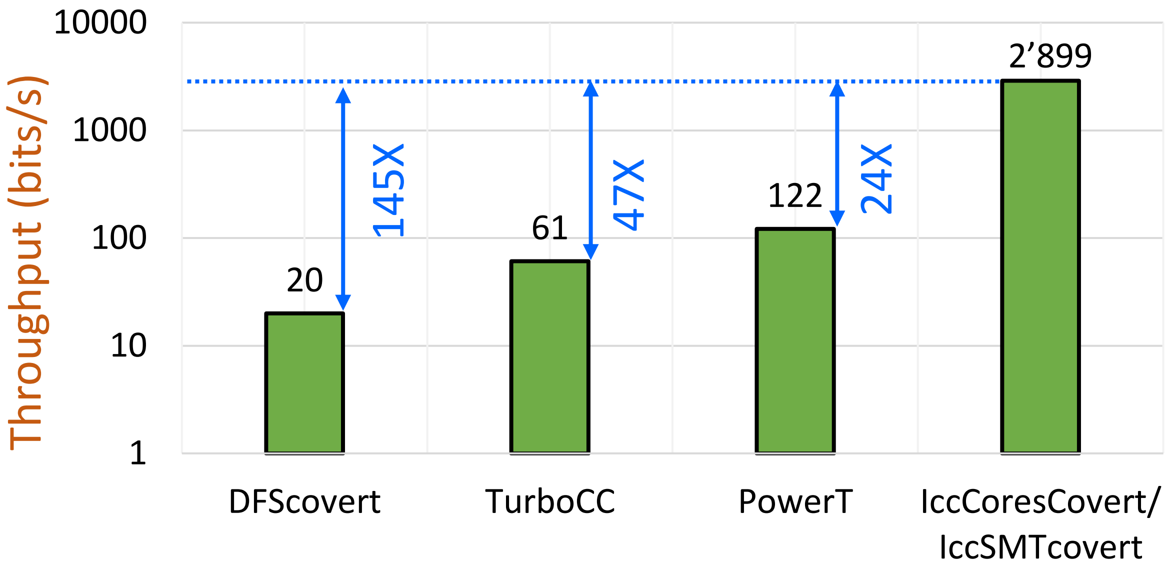
- **Comparison Points**: We compare IChannels to four recent works
  - That exploit different power management mechanisms of modern processors to build covert channels

Results – IccThreadCovert



- We compare IccThreadCovert against NetSpectre
  - The state-of-the-art work that exploits the variable latency of PHIs to create a covert channel between two execution contexts running on the same hardware thread
- The NetSpectre covert channel can send one bit per transaction,
  - IccThreadCovert covert channel can send two bits per transaction

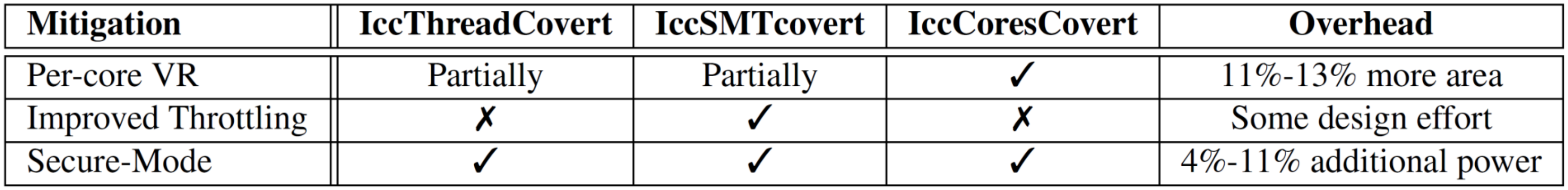
Results – IccSMTcovert & IccCoresCovert



- We compare IccSMTcovert and IccCoresCovert against DFScovert, TurboCC and PowerT
    - The state-of-the-art works that exploit different power management mechanisms of modern processors to build covert channels across cores and SMT threads
- IccSMTcovert/IccCoresCovert throughput is 145×, 47×, and 24×
    - The throughput of DFScovert, TurboCC, and PowerT, respectively
- The three works exploit slow mechanisms (e.g., frequency/thermal changes)
    - Compared to the current management side-effects that our IChannels exploits

IChannels Proposed Mitigation

- We propose practical hardware and software techniques for the mitigation of IChannels covert channels:
  - Fast Per-core Voltage Regulators
    - Each core can change voltage independently of other cores
  - Improved Core Throttling
    - For SMT threads, throttle only the thread that runs PHI
  - New Secure Mode of Operation
    - Before entering this secure mode, transition the voltage to the highest voltage guardband to prevent throttling when executing any PHI type



Mitigation	**IccThreadCovert**	**IccSMTcovert**	**IccCoresCovert**	**Overhead**
Per-core VR	Partially	Partially	✓	11%-13% more area
Improved Throttling	✗	✓	✗	Some design effort
Secure-Mode	✓	✓	✓	4%-11% additional power

Presentation Outline

Conclusion

Problem: Current management mechanisms throttle instruction execution and adjust voltage/frequency to accommodate power-hungry instructions (PHIs).
These mechanisms may compromise a system's confidentiality guarantees

Goal:
1. Understand the throttling side-effects of current management mechanisms
2. Build high-capacity covert channels between otherwise isolated execution contexts
3. Practically and effectively mitigate each covert channel

Characterization: Variable execution times and frequency changes due to running PHIs
We observe five different levels of throttling in real Intel systems

IChannels: New covert channels that exploit side-effects of current management mechanisms
- On the same hardware thread
- Across co-located Simultaneous Multi-Threading (SMT) threads
- Across different physical cores

Evaluation: On three generations of Intel processors, IChannels provides a channel capacity
- 2× that of PHIs' variable latency-based covert channels
- 24× that of power management-based covert channels

IChannels
Exploiting Current Management Mechanisms to Create Covert Channels in Modern Processors

Jawad Haj-Yahya

Jeremie S. Kim *A. Giray Yağlıkçı* *Ivan Puddu* *Lois Orosa*
Juan Gómez Luna *Mohammed Alser* *Onur Mutlu*

SAFARI ETH zürich

Many more PM Vulnerabilities (I)

- There are many more attacks that exploit different PM vulnerabilities
  - Frequency Vulnerabilities and Mitigations:
    - Alagappan, Murugappan, et al. "DFS covert channels on multi-core platforms." 2017 VLSI-SoC.
    - Qiu, Pengfei, et al. "DVFSspy: Using Dynamic Voltage and Frequency Scaling as a Covert Channel for Multiple Procedures." ASP-DAC, 2022.
    - Liu, Chen, et al. "Frequency throttling side-channel attack.“, SIGSAC 2022.
    - Wang, Yingchen, et al. "Hertzbleed: Turning Power Side-Channel Attacks Into Remote Timing Attacks on x86." USENIX Security 2022.
    - Zhang, Sheng, et al. "Blacklist core: machine-learning based dynamic operating-performance-point blacklisting for mitigating power-management security attacks.“, ISLEPD 2018.
  - Power Vulnerabilities and Mitigations:
    - Khatamifard, S. Karen, et al. "A New Class Of Covert Channels Exploiting Power Management Vulnerabilities“, CAL 2018.
    - Lipp, Moritz, Daniel Gruss, and Michael Schwarz. "AMD Prefetch Attacks Through Power And Time“, USENIX Security 2022.
    - Khatamifard, S. Karen, et al. "POWERT Channels: A Novel Class Of Covert Communication Exploiting Power Management Vulnerabilities.“, HPCA 2019.
  - Power-Supply Vulnerabilities and Mitigations:
    - Sehatbakhsh, Nader, et al. "A New Side-channel Vulnerability On Modern Computers By Exploiting Electromagnetic Emanations From The Power Management Unit.“, HPCA 2020.
    - Giechaskiel, Ilias, Kasper Bonne Rasmussen, and Jakub Szefer. "C^{3}APSULe: Cross-FPGA Covert-Channel Attacks through Power Supply Unit Leakage.“, S&P 2020.
  - Power-Management-Unit Vulnerabilities and Mitigations:
    - JayashankaraShridevi, Rajesh, et al. "Catching The Flu: Emerging Threats From A Third-party Power Management Unit.“, DAC 2016.

Many more PM Vulnerabilities (II)

- Thermal Vulnerabilities and Mitigations:
  - Huang, Hengli, et al. "Detection of and Countermeasure against Thermal Covert Channel in Many-core Systems.", TCAD 2021.
  - Tian, Shanquan, and Jakub Szefer. "Temporal thermal covert channels in cloud FPGAs.", FPGA 2019.
  - Kim, Taehun, and Youngjoo Shin. "ThermalBleed: A practical thermal side-channel attack." IEEE Access 2022.
- Voltage Vulnerabilities and Mitigations:
  - Singh, Arvind, et al. "Enhanced Power And Electromagnetic SCA Resistance Of Encryption Engines Via A Security-aware Integrated All-digital LDO." JSSC 2019.
  - Singh, Arvind, et al. "Exploiting on-chip power management for side-channel security.", DATE 2018.
  - Singh, Arvind, et al. "Improved power/EM side-channel attack resistance of 128-bit AES engines with random fast voltage dithering." JSSC 2018.
  - Singh, Arvind, et al. "Mitigating power supply glitch-based fault attacks with fast all-digital clock modulation circuit.", DATE 2019.
  - Schellenberg, Falk, et al. "Remote inter-chip power analysis side-channel attacks at board-level.", ICCAD 2018.
  - Kenjar, Zijo, et al. "V0LTpwn: Attacking x86 Processor Integrity from Software." USENIX Security 2020.
  - Qiu, Pengfei, et al. "VoltJockey: Breaching TrustZone by software-controlled voltage manipulation over multi-core frequencies." SIGSAC 2019.
  - Chen, Zitai, et al. "VoltPillager: Hardware-based fault injection attacks against Intel SGX Enclaves using the SVID voltage scaling interface." USENIX Security 2021.

Conclusion

Conclusion

- In this talk, we provided
  - An overview of state-of-the-art power management mechanisms used in modern microprocessors
  - Multiple state-of-the-art security vulnerabilities that exploit power management mechanisms and mitigations to protect against these vulnerabilities
    - Deep dive into our IChannels vulnerabilities (ISCA 2021)

- We conclude that power management, and resulting security implications are critical and exciting areas to research to make modern systems both more energy-efficient and secure

Q
&
A

Security Implications of Power Management Mechanisms in Modern Processors

Current Studies and Future Trends

Jawad Haj-Yahya
Principal Architect, Rivos Inc.

The International Winter School on Microarchitectural Security (Mic-Sec) Paris, 5 December 2022