Emulating Power Attacks with *gem5*

Carlos Andres LARA-NINO

Journée thématique « sécurité matérielle et open-source »

Paris, France

November 13th, 2023

Université Jean Monnet Saint-Etienne, CNRS, Laboratoire Hubert Curien
UMR 5516, F-42023, SAINT-ETIENNE, France

Acknowledgements

This work has been supported by the French government through the *Agence Nationale de la Recherche* under project ARCHISEC (ANR-19-CE39-0008) and in the framework of the *France 2030* initiative under project ARSENE (ANR-22-PECY-0004).

Outline

Motivation

Power attacks

Power attacks leverage the power distribution network of the target.

Attack goals

- Power analysis
- Covert data transmission
- Fault injection and denial of service

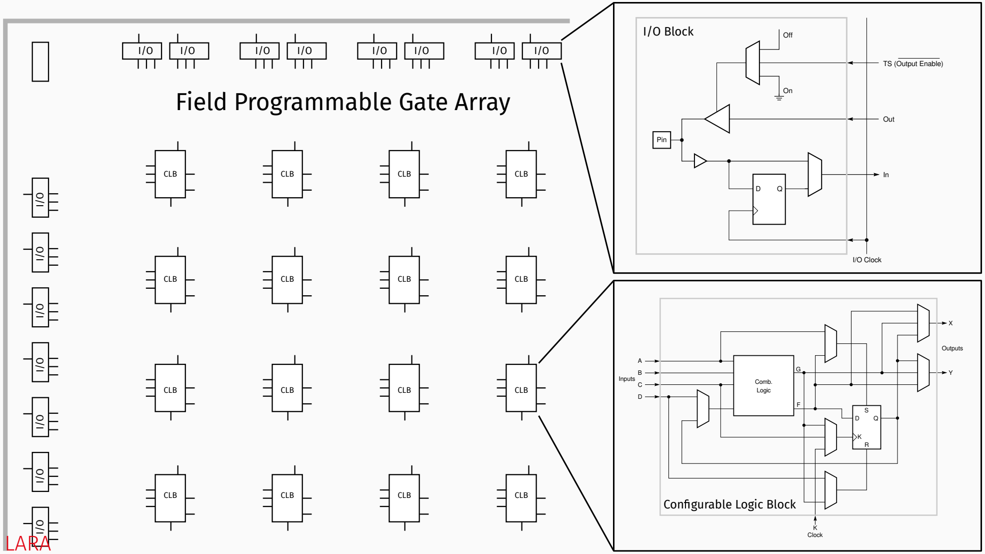
Power attacks

Power attacks leverage the power distribution network of the target.

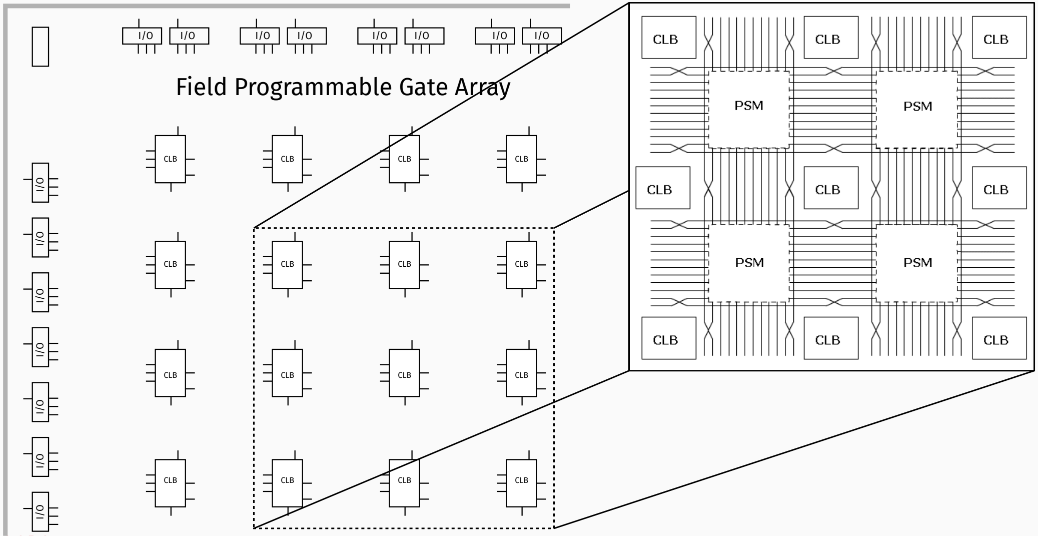
Attack goals

- Power analysis
- Covert data transmission
- Fault injection and denial of service

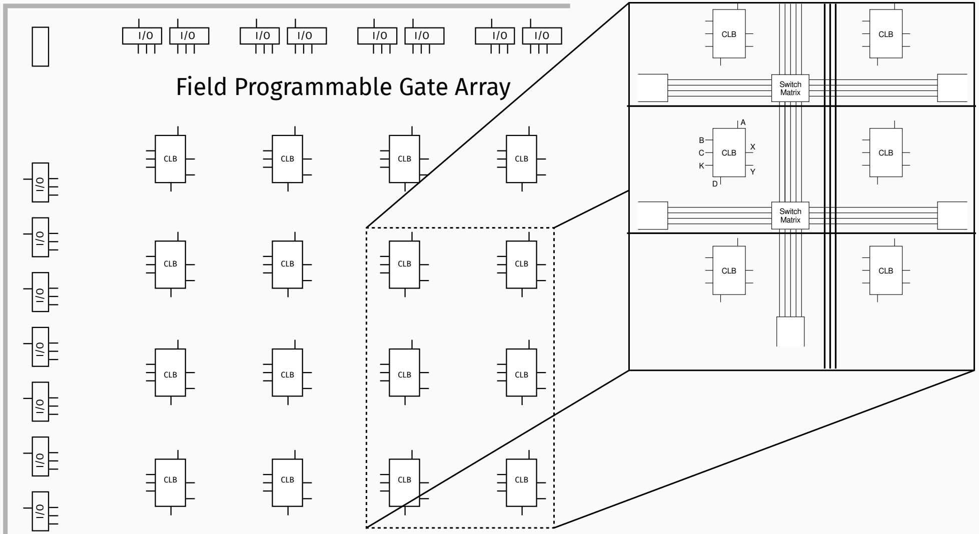
Power distribution networks: FPGAs



Power distribution networks: FPGAs



Power distribution networks: FPGAs



Attack scenarios

Conventional

- physical access to the device
- sophisticated equipment
- physical tampering **and** logical tampering

Attack scenarios

Remote

- connected devices
- leverages the internal components of the chip (reconfigurable logic, memories)
- software vulnerabilities **and** logical tampering

State of the Art

Remote power analysis

2018	Zhao, M., & Suh, G. E. **FPGA-based remote power side-channel attacks**.
2021	Schellenberg, F., Gnad, D. R., Moradi, A., & Tahoori, M. B. **An inside job: Remote power analysis attacks on FPGAs**.

Internal sensors

2010 Franco, J. J. L., Boemo, E., Castillo, E., & Parrilla, L.
Ring oscillators as thermal sensors in FPGAs: Experiments in low voltage.

2013 Zick, K. M., Srivastav, M., Zhang, W., & French, M.
Sensing nanosecond-scale voltage attacks and natural transients in FPGAs.

2019 Gravellier, J., Dutertre, J. M., Teglia, Y., & Loubet-Moundi, P.
High-speed ring oscillator based sensors for remote side-channel attacks on FPGAs.

2019 Giechaskiel, I., Eguro, K., & Rasmussen, K. B.
Leakier wires: Exploiting FPGA long wires for covert-and side-channel attacks.

2021 Gravellier, J., Dutertre, J. M., Teglia, Y., & Moundi, P. L.
Sideline: How delay-lines (may) leak secrets from your SoC.

2023 Spielmann, D., Glamočanin, O., & Stojilović, M.
RDS: FPGA Routing Delay Sensors for Effective Remote Power Analysis Attacks.

State of the Art

Power covert channels

2010	Ziener, D., Baueregger, F., & Teich, J. **Using the power side channel of FPGAs for communication**.
2021	Gnad, D. R., Nguyen, C. D. K., Gillani, S. H., & Tahoori, M. B. **Voltage-based covert channels using FPGAs**.

State of the Art

Combined internal attacks

2023 Fellah-Touta, A., Bossuet, L., & Lara-Nino, C. A.
Combined Internal Attacks on SoC-FPGAs: Breaking AES with Remote Power Analysis and Frequency-based Covert Channels.

Countermeasures

Masking

2013 Prouff, E., & Rivain, M.
Masking against side-channel attacks: A formal security proof.

Isolation

2019 Krautter, J., Gnad, D. R., Schellenberg, F., Moradi, A., & Tahoori, M. B.
Active fences against voltage-based side channels in multi-tenant FPGAs.

Inspection of binaries

2018 Gnad, D. R., Rapp, S., Krautter, J., & Tahoori, M. B.
Checking for electrical level security threats in bitstreams for multi-tenant FPGAs.

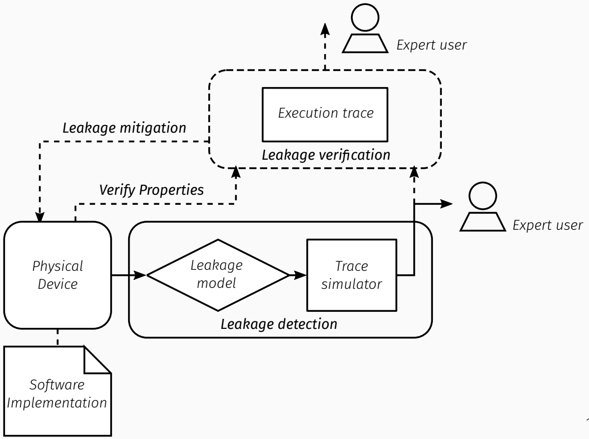
2020 La, T. M., Matas, K., Grunchevski, N., Pham, K. D., & Koch, D.
FPGAdefender: Malicious self-oscillator scanning for Xilinx UltraScale+ FPGAs

Challenges

- Some of these techniques must be adjusted to the algorithm
- Require extensive testing to confirm their effectiveness
- Some protections might be more effective in a specific platform

Tooling for the study of power attacks

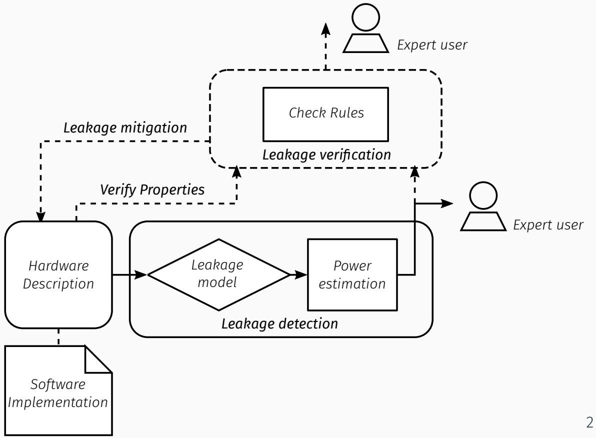
Post-silicon analysis



[1]

[1] Buhan, I., Batina, L., Yarom, Y., & Schaumont, P. (2022, May). SoK: Design tools for side-channel-aware implementations. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security (Asia CCS)* (pp. 756-770).

Pre-silicon tooling



[2]

[2] Buhan, I., Batina, L., Yarom, Y., & Schaumont, P. (2022, May). SoK: Design tools for side-channel-aware implementations. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security (Asia CCS)* (pp. 756-770).

Leakage verification

2013	S. A. Huss, M. Stöttinger, and M. Zohner. **AMASIVE: An Adaptable and Modular Autonomous Side-Channel Vulnerability Evaluation Framework**.
2014	D. Walters, A. Hagen, and E. Kedaigle. **SLEAK: A Side-Channel Leakage Evaluator and Analysis Kit**.
2018	Y. L. Corre, J. Großschädl, and D. Dinu. **Micro-architectural Power Simulator for Leakage Assessment of Cryptographic Software on ARM Cortex-M3 Processors**. (MAPS)
2019	P. Slpsk, P. K. Vairam, C. Rebeiro, and V. Kamakoti. **KARNA: A Gate-Sizing based Security Aware EDA Flow for Improved Power Side-Channel Attack Protection**.

State of the Art

Leakage verification

2020 A. Nahiyan, J. Park, M. T. He, Y. Iskander, F. Farahmandi, D. Forte, and M. Tehranipoor.
SCRIPT: A CAD Framework for Power Side-channel Vulnerability Assessment Using Information Flow Tracking and Pattern Generation.

2021 A. Jahania and V. Samadi Bokharaie.
Power side-channel leakage assessment and locating the exact sources of leakage at the early stages of ASIC design process. (PATCH)

2021 B. Gigerl, V. Hadzic, R. Primas, S. Mangard, and R. Bloem.
Coco: Co-Design and Co-Verification of Masked Software Implementations on CPUs.

State of the Art

Leakage detection

2019	M. He, J. Park, A. Nahiyan, A. Vassilev, Y. Jin, and M. Tehranipoor. **RTL-PSC: Automated Power Side-Channel Leakage Assessment at Register-Transfer Level**.
2020	M. A. K. F, V. Ganesan, R. Bodduna, and C. Rebeiro. **PARAM: A Microprocessor Hardened for Power Side-Channel Attack Resistance**
2020	Y. Yao, T. Kathuria, B. Ege, and P. Schaumont. **Architecture Correlation Analysis (ACA): Identifying the Source of Side-channel Leakage at Gate-level**.
2020	D. Sijacic, J. Balasch, B. Yang, S. Ghosh, and I. Verbauwhede. **Towards efficient and automated side-channel evaluations at design time**. (CASCADE)

State of the Art

Leakage detection					
Tool	**Year**	**Input**	**Target device**	**Masking**	**Open source**
RTL-PSC	2019	RTL	AES	No	No
PARAM	2020	Gate layout	RISC-V	No	No
ACA	2020	Gate layout	RISC-V	No	No
CASCADE	2020	Gate layout	Generic ASIC	No	Yes

Leakage verification					
AMASIVE	**2013**	**RTL**	**Generic**	**No**	**No**
SLEAK	2014	ISA	ARM Cortex A8	Yes	No
MAPS	2018	ISA	ARM Cortex M3	Yes	Yes
KARNA	2019	Gate layout	AES, Simon	No	No
SCRIPT	2020	Gate layout	AES	No	No
PATCH	2021	Gate layout	AES	No	No
COCO	2021	Gate layout	RISC-V	Yes	Yes

The *gem5* simulator

About

- A modular platform for computer-system architecture research
- Encompasses system-level architecture as well as processor microarchitecture
- Precursors :
  - m5 from the University of Michigan
  - GEMS from the University of Wisconsin
- Used by ARM Research, AMD Research, Google, Micron, Metempsy, HP, Samsung[3]

[3] *https://www.gem5.org/about/*

Why *gem5*?

- Open source, free for use[4]
- Can emulate the operation of multiple processors including ARM and RISC-V
- Can be extended to model memory[5] and network[6] activity

[4] Lowe-Power, J., Ahmad, A. M., Akram, A., Alian, M., Amslinger, R., Andreozzi, M., ... & Zulian, É. F. The gem5 simulator: Version 20.0+. arXiv:2007.03152.

[5] Kim, Y., Yang, W., & Mutlu, O. Ramulator: A fast and extensible DRAM simulator. *IEEE Computer architecture letters*, 15(1), 45-49.

[6] Agarwal, N., Krishna, T., Peh, L. S., & Jha, N. K. GARNET: A detailed on-chip network model inside a full-system simulator. In *2009 IEEE international symposium on performance analysis of systems and software* (pp. 33-42). IEEE.

Use in the literature

Study of microarchitectural attacks

2021 France, L., Bruguier, F., Mushtaq, M., Novo, D., & Benoit, P.
Implementing Rowhammer Memory Corruption in the gem5 Simulator.

2021 Ayoub, P., & Maurice, C.
Reproducing spectre attack with gem5: How to do it right?

2023 Bossuet, L. & Lara-Nino, C.A.
Emulating Covert Data Transmission on Heterogeneous SoCs.

Why not *gem5*

- Shallow learning curve
- Long simulation time for complex simulations
- May produce large volumes of data
- The code is updated very often (some of it), the documentation is not

Methods

Modeling the power dissipation of a system

- A *gem5* simulation creates a large set of statistics associated with the operation of the system
- These data can be used to study the different components in the simulated platform
- As the activity of the multiple underlying components would affect the power consumption of the entire system, we propose that the simulation statistics can be used to approach the electrical behavior of a simulated platform

$$\begin{aligned} power &= dynamic + static \\ dynamic &= voltage \times (2A \times ipc + 3pA \times dentry\_misses) \\ static &= 4 \times temperature \end{aligned} \tag{1}$$

Simulation environment

- Our work builds on the *gem5* simulator publicly available[7]
- We focus on ARM platforms and use the ***`opt`*** level to conduct our experiments
- We employ the ***`full simulation`*** as it produces statistics which are closer to those observed in a physical device

[7] *`https://www.gem5.org/`*

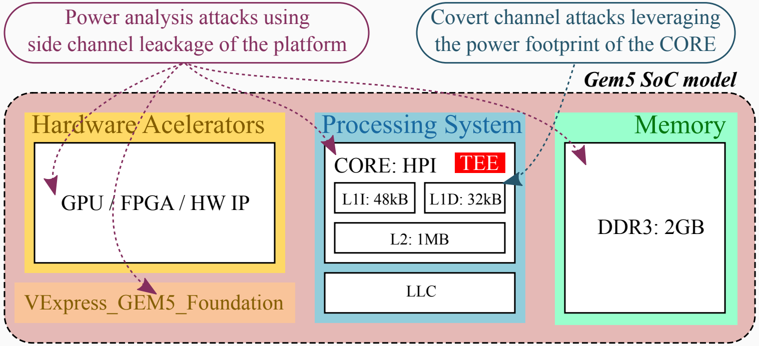
Simulation environment (cont.)

- From every execution of the simulation, *gem5* dumps statistics with a rate of $1E - 3$
- We modified this dump rate to $2.5 \times freq\_core$
- It was possible to use a core frequency of 10MHz and an acquisition rate of 25MHz
- We modulated the core frequency to segment the statistics

The simulated platform

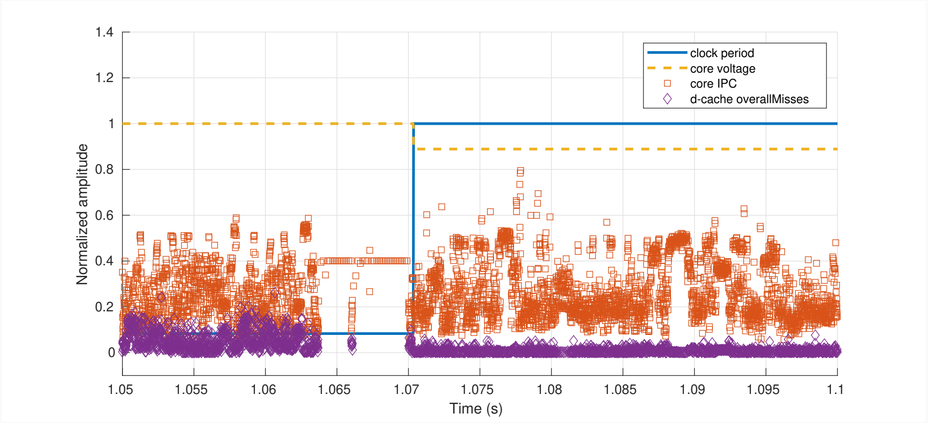
- *`High performance in-order`* processor model (Cortex A-53)
- Single core to reduce the complexity of the simulation
- 2GB *`DDR3_1600_8x8`* unit which was emulated using Ramulator
- L1 and L2 caches as well as a memory management unit
- *`VExpress_gem5_Foundation`* machine type which allows to execute the *`ArmTrustedFirmware`* workload
- Included the OP-TEE runtime in the simulation

The simulated platform (cont.)



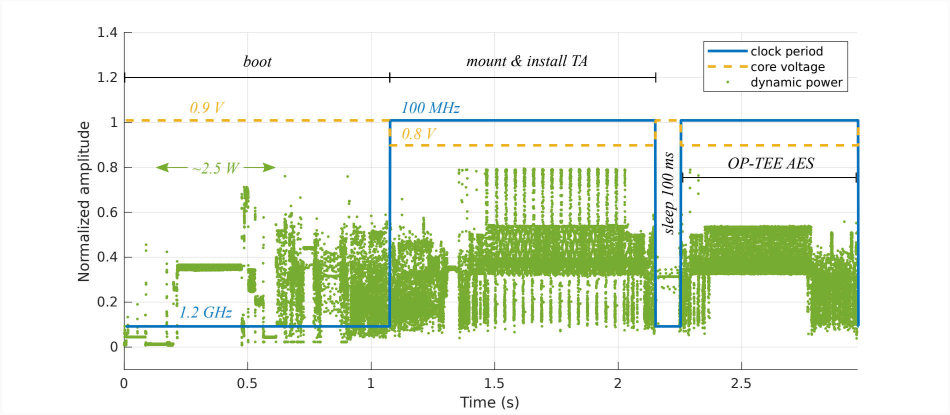
Modeling the power dissipation of a system (cont.)

$$dynamic = voltage \times (2A \times ipc + 3pA \times dentry\_misses)$$



Modeling the power dissipation of a system (cont.)

$$dynamic = voltage \times (2A \times ipc + 3pA \times dentry\_misses)$$



Correlation Power Analysis

Experimental setup (i)

- We used as case study the power analysis of AES
- We simplified the implementation to the essential operations $c = SBOX(p \oplus k)$
- We tested two key values (`0x2B, 0x7E`) to analyze whether it was possible to find useful information in the *gem5* output

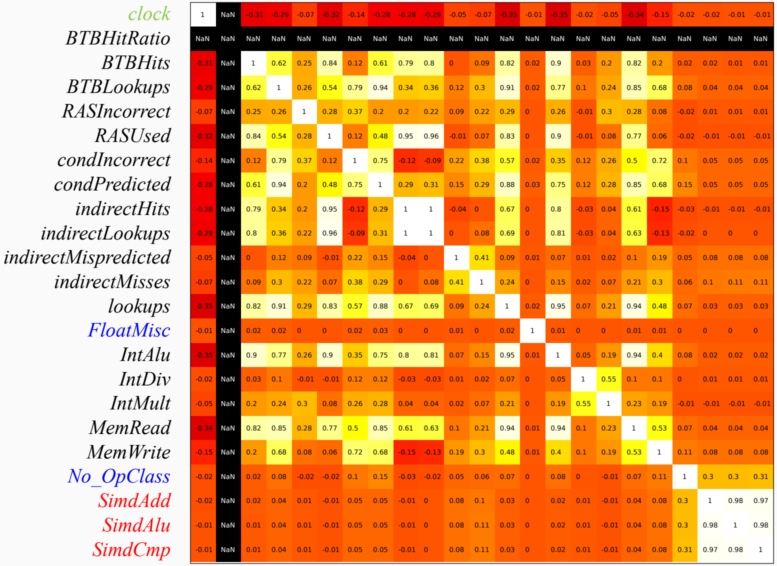
First round model

```
Require: $k$, an 8-bit random integer
Require: SBOX, the substitution box of AES
Require: $f, f'$, two frequencies of the core with $f' < 2f_s$
  for $i = 0$ to 255 do
    cpu_freq $\leftarrow f'$                                   {Pull trigger}
    SBOX($i \oplus k$)
    cpu_freq $\leftarrow f$                                    {Release trigger}
    wait
  end for
```

Challenges

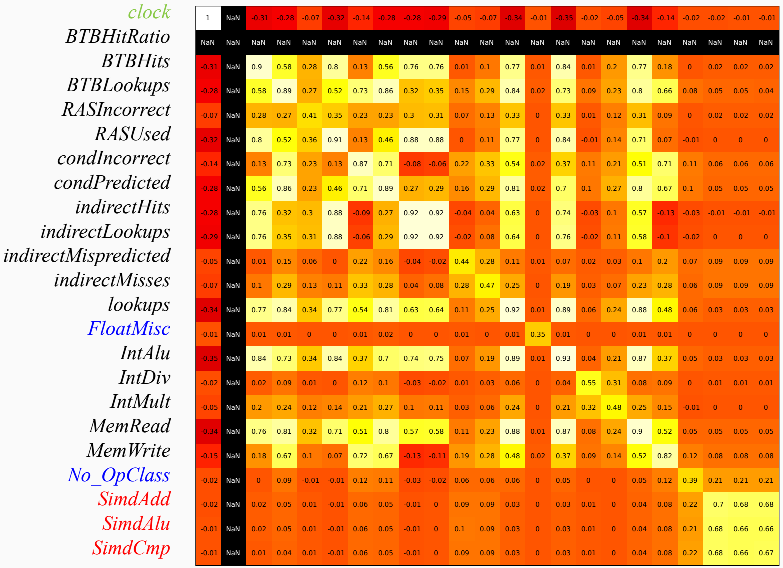
- We obtained over 800 files holding multiple metrics from the simulation
- It was not practical to process all these data with the usual statistical techniques
- We devised a statistical analysis to identify statistics of interest

Auto-correlation analysis



clock	1	NaN	-0.31	-0.29	-0.07	-0.32	-0.14	-0.28	-0.28	-0.29	-0.05	-0.07	-0.35	-0.01	-0.35	-0.02	-0.05	-0.34	-0.15	-0.02	-0.02	-0.01	-0.01
BTBHitRatio	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
BTBHits	-0.31	NaN	1	0.62	0.25	0.84	0.12	0.61	0.79	0.8	0	0.09	0.82	0.02	0.9	0.03	0.2	0.82	0.2	0.02	0.02	0.01	0.01
BTBLookups	-0.29	NaN	0.62	1	0.26	0.54	0.79	0.94	0.34	0.36	0.12	0.3	0.91	0.02	0.77	0.1	0.24	0.85	0.68	0.08	0.04	0.04	0.04
RASIncorrect	-0.07	NaN	0.25	0.26	1	0.28	0.37	0.2	0.2	0.22	0.09	0.22	0.29	0	0.26	-0.01	0.3	0.28	0.08	-0.02	0.01	0.01	0.01
RASUsed	-0.32	NaN	0.84	0.54	0.28	1	0.12	0.48	0.95	0.96	-0.01	0.07	0.83	0	0.9	-0.01	0.08	0.77	0.06	-0.02	-0.01	-0.01	-0.01
condIncorrect	-0.14	NaN	0.12	0.79	0.37	0.12	1	0.75	-0.12	-0.09	0.22	0.38	0.57	0.02	0.35	0.12	0.26	0.5	0.72	0.1	0.05	0.05	0.05
condPredicted	-0.28	NaN	0.61	0.94	0.2	0.48	0.75	1	0.29	0.31	0.15	0.29	0.88	0.03	0.75	0.12	0.28	0.85	0.68	0.15	0.05	0.05	0.05
indirectHits	-0.28	NaN	0.79	0.34	0.2	0.95	-0.12	0.29	1	1	-0.04	0	0.67	0	0.8	-0.03	0.04	0.61	-0.15	-0.03	-0.01	-0.01	-0.01
indirectLookups	-0.29	NaN	0.8	0.36	0.22	0.96	-0.09	0.31	1	1	0	0.08	0.69	0	0.81	-0.03	0.04	0.63	-0.13	-0.02	0	0	0
indirectMispredicted	-0.05	NaN	0	0.12	0.09	-0.01	0.22	0.15	-0.04	0	1	0.41	0.09	0.01	0.07	0.01	0.02	0.1	0.19	0.05	0.08	0.08	0.08
indirectMisses	-0.07	NaN	0.09	0.3	0.22	0.07	0.38	0.29	0	0.08	0.41	1	0.24	0	0.15	0.02	0.07	0.21	0.3	0.06	0.1	0.11	0.11
lookups	-0.35	NaN	0.82	0.91	0.29	0.83	0.57	0.88	0.67	0.69	0.09	0.24	1	0.02	0.95	0.07	0.21	0.94	0.48	0.07	0.03	0.03	0.03
FloatMisc	-0.01	NaN	0.02	0.02	0	0	0.02	0.03	0	0	0.01	0	0.02	1	0.01	0	0	0.01	0.01	0	0	0	0
IntAlu	-0.35	NaN	0.9	0.77	0.26	0.9	0.35	0.75	0.8	0.81	0.07	0.15	0.95	0.01	1	0.05	0.19	0.94	0.4	0.08	0.02	0.02	0.02
IntDiv	-0.02	NaN	0.03	0.1	-0.01	-0.01	0.12	0.12	-0.03	-0.03	0.01	0.02	0.07	0	0.05	1	0.55	0.1	0.1	0	0.01	0.01	0.01
IntMult	-0.05	NaN	0.2	0.24	0.3	0.08	0.26	0.28	0.04	0.04	0.02	0.07	0.21	0	0.19	0.55	1	0.23	0.19	-0.01	-0.01	-0.01	-0.01
MemRead	-0.34	NaN	0.82	0.85	0.28	0.77	0.5	0.85	0.61	0.63	0.1	0.21	0.94	0.01	0.94	0.1	0.23	1	0.53	0.07	0.04	0.04	0.04
MemWrite	-0.15	NaN	0.2	0.68	0.08	0.06	0.72	0.68	-0.15	-0.13	0.19	0.3	0.48	0.01	0.4	0.1	0.19	0.53	1	0.11	0.08	0.08	0.08
No_OpClass	-0.02	NaN	0.02	0.08	-0.02	-0.02	0.1	0.15	-0.03	-0.02	0.05	0.06	0.07	0	0.08	0	-0.01	0.07	0.11	1	0.3	0.3	0.31
SimdAdd	-0.02	NaN	0.02	0.04	0.01	-0.01	0.05	0.05	-0.01	0	0.08	0.1	0.03	0	0.02	0.01	-0.01	0.04	0.08	0.3	1	0.98	0.97
SimdAlu	-0.01	NaN	0.01	0.04	0.01	-0.01	0.05	0.05	-0.01	0	0.08	0.11	0.03	0	0.02	0.01	-0.01	0.04	0.08	0.3	0.98	1	0.98
SimdCmp	-0.01	NaN	0.01	0.04	0.01	-0.01	0.05	0.05	-0.01	0	0.08	0.11	0.03	0	0.02	0.01	-0.01	0.04	0.08	0.31	0.97	0.98	1

Cross-correlation analysis



clock	1	NaN	-0.31	-0.28	-0.07	-0.32	-0.14	-0.28	-0.28	-0.29	-0.05	-0.07	-0.34	-0.01	-0.35	-0.02	-0.05	-0.34	-0.14	-0.02	-0.02	-0.01	-0.01
BTBHitRatio	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
BTBHits	-0.31	NaN	0.9	0.58	0.28	0.8	0.13	0.56	0.76	0.76	0.01	0.1	0.77	0.01	0.84	0.01	0.2	0.77	0.18	0	0.02	0.02	0.02
BTBLookups	-0.28	NaN	0.58	0.89	0.27	0.52	0.73	0.86	0.32	0.35	0.15	0.29	0.84	0.02	0.73	0.09	0.23	0.8	0.66	0.08	0.05	0.05	0.04
RASIncorrect	-0.07	NaN	0.28	0.27	0.41	0.35	0.23	0.23	0.3	0.31	0.07	0.13	0.33	0	0.33	0.01	0.12	0.31	0.09	0	0.02	0.02	0.02
RASUsed	-0.32	NaN	0.8	0.52	0.36	0.91	0.13	0.46	0.88	0.88	0	0.11	0.77	0	0.84	-0.01	0.14	0.71	0.07	-0.01	0	0	0
condIncorrect	-0.14	NaN	0.13	0.73	0.23	0.13	0.87	0.71	-0.08	-0.06	0.22	0.33	0.54	0.02	0.37	0.11	0.21	0.51	0.71	0.11	0.06	0.06	0.06
condPredicted	-0.28	NaN	0.56	0.86	0.23	0.46	0.71	0.89	0.27	0.29	0.16	0.29	0.81	0.02	0.7	0.1	0.27	0.8	0.67	0.1	0.05	0.05	0.05
indirectHits	-0.28	NaN	0.76	0.32	0.3	0.88	-0.09	0.27	0.92	0.92	-0.04	0.04	0.63	0	0.74	-0.03	0.1	0.57	-0.13	-0.03	-0.01	-0.01	-0.01
indirectLookups	-0.29	NaN	0.76	0.35	0.31	0.88	-0.06	0.29	0.92	0.92	-0.02	0.08	0.64	0	0.76	-0.02	0.11	0.58	-0.1	-0.02	0	0	0
indirectMispredicted	-0.05	NaN	0.01	0.15	0.06	0	0.22	0.16	-0.04	-0.02	0.44	0.28	0.11	0.01	0.07	0.02	0.03	0.1	0.2	0.07	0.09	0.09	0.09
indirectMisses	-0.07	NaN	0.1	0.29	0.13	0.11	0.33	0.28	0.04	0.08	0.28	0.47	0.25	0	0.19	0.03	0.07	0.23	0.28	0.06	0.09	0.09	0.09
lookups	-0.34	NaN	0.77	0.84	0.34	0.77	0.54	0.81	0.63	0.64	0.11	0.25	0.92	0.01	0.89	0.06	0.24	0.88	0.48	0.06	0.03	0.03	0.03
FloatMisc	-0.01	NaN	0.01	0.01	0	0	0.01	0.02	0	0	0.01	0	0.01	0.35	0.01	0	0	0.01	0.01	0	0	0	0
IntAlu	-0.35	NaN	0.84	0.73	0.34	0.84	0.37	0.7	0.74	0.75	0.07	0.19	0.89	0.01	0.93	0.04	0.21	0.87	0.37	0.05	0.03	0.03	0.03
IntDiv	-0.02	NaN	0.02	0.09	0.01	0	0.12	0.1	-0.03	-0.02	0.01	0.03	0.06	0	0.04	0.55	0.31	0.08	0.09	0	0.01	0.01	0.01
IntMult	-0.05	NaN	0.2	0.24	0.12	0.14	0.21	0.27	0.1	0.11	0.03	0.06	0.24	0	0.21	0.32	0.48	0.25	0.15	-0.01	0	0	0
MemRead	-0.34	NaN	0.76	0.81	0.32	0.71	0.51	0.8	0.57	0.58	0.11	0.23	0.88	0.01	0.87	0.08	0.24	0.9	0.52	0.05	0.05	0.05	0.05
MemWrite	-0.15	NaN	0.18	0.67	0.1	0.07	0.72	0.67	-0.13	-0.11	0.19	0.28	0.48	0.02	0.37	0.09	0.14	0.52	0.82	0.12	0.08	0.08	0.08
No_OpClass	-0.02	NaN	0	0.09	-0.01	-0.01	0.12	0.11	-0.03	-0.02	0.06	0.06	0.06	0	0.05	0	0	0.05	0.12	0.39	0.21	0.21	0.21
SimdAdd	-0.02	NaN	0.02	0.05	0.01	-0.01	0.06	0.05	-0.01	0	0.09	0.09	0.03	0	0.03	0.01	0	0.04	0.08	0.22	0.7	0.68	0.68
SimdAlu	-0.01	NaN	0.02	0.05	0.01	-0.01	0.06	0.05	-0.01	0	0.1	0.09	0.03	0	0.03	0.01	0	0.04	0.08	0.21	0.68	0.66	0.66
SimdCmp	-0.01	NaN	0.01	0.04	0.01	-0.01	0.06	0.05	-0.01	0	0.09	0.09	0.03	0	0.03	0.01	0	0.04	0.08	0.22	0.68	0.66	0.67

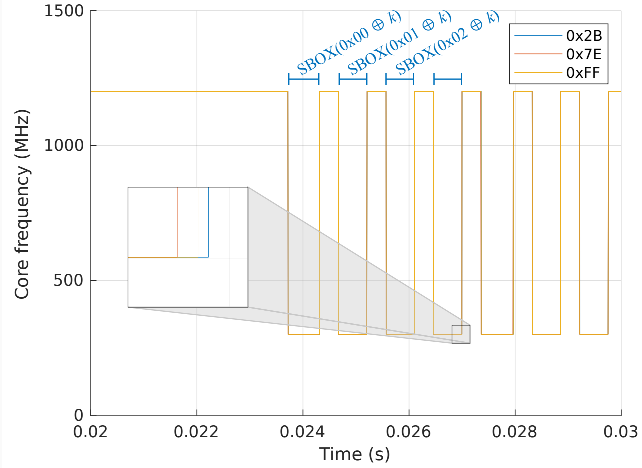
Statistics of interest

- The auto-correlation discards statistics which are similar
- The cross-correlation finds statistics which change with the secret value
- Low cross-correlation statistics:
    - dcache.tags.totalRefs
    - dtb_walker_cache.tags.totalRefs
    - icache.tags.totalRefs
    - mmu.alignFaults
    - mmu.itb.flushedEntries
    - mmu.prefetchFaults

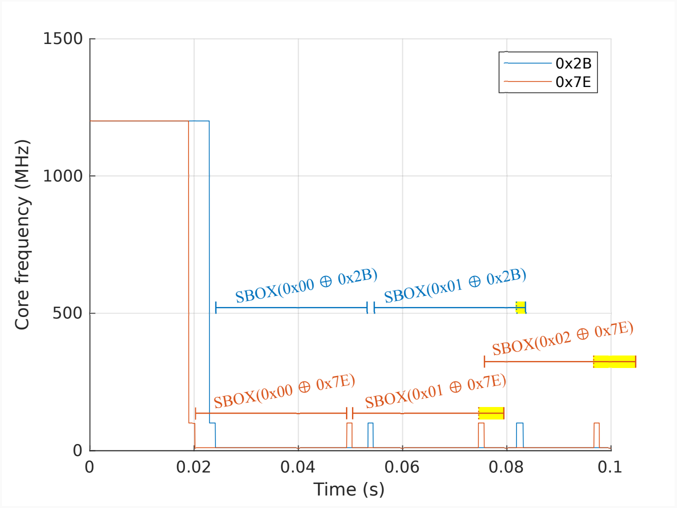
Other challenges

- When we set $f' = 10MHz$ and $f_s = 25MHz$ the fidelity of the analysis decreased
- Capturing 20 traces took over 72h for each key value and yielded 1TB of statistics in each case

Typical behavior



Low frequency behavior



Leakage assessment

- We performed a ***Student's t-test*** to see if the leakages' distribution differed from one another
- The secret values were the key values ***0x2B*** and ***0x7E***
- Using the raw leakages, no statistics overtook the leakage detection threshold of 4.5
- This confirmed the null hypothesis of the ***t-test*** that both sets followed the same distribution

Potential improvements

- Get a larger set of traces for each key
- Apply signal processing techniques to improve the synchronization of the traces
- Consider attacks on various statistics at the same time

Simple Power Analysis

Experimental setup (ii)

- As we noticed some accuracy in the timing behavior we attempted to perform timing analysis
- It is quite costly to simulate RSA or ECC
- We analyzed a simple binary-field double-and-add 1024-bit multiplication implemented in C
- Goal : Determine weather we could identify the power footprint of large arithmetic operations

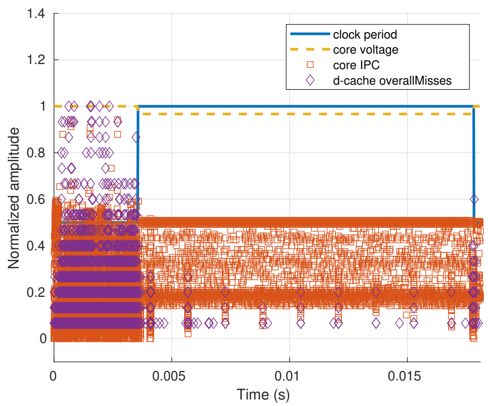
Binary multiplication model

```
Require: $g(t)$, an irreducible polynomial for $\mathbb{K}$
Require: $a, b \in \mathbb{K}$, two random integers
Require: $f, f'$, two frequencies of the core with $f' < 2f_s$
   cpu_freq $\leftarrow f'$                                   {Pull trigger}
   for $i = 0$ to 5 do
      $a \times b \in \mathbb{K}$
   end for
   cpu_freq $\leftarrow f$                                    {Release trigger}
```

Our approach

- We set the core frequency to 600 MHz and used an acquisition rate of $4E - 8$
- This allowed us to see the activity of the core during processing

Simulation traces

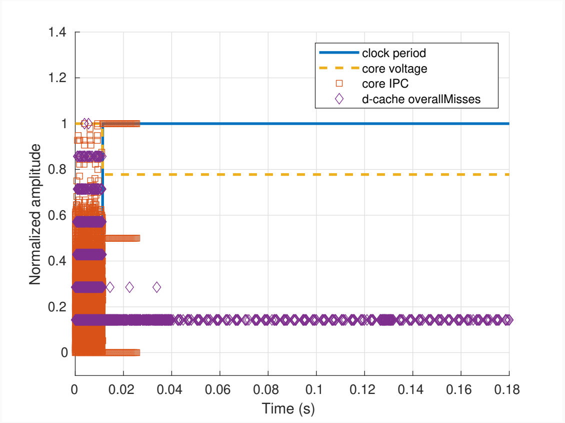


We could not find any timing pattern which corresponded with the execution of the multiplications.

Our approach

- We decreased the core frequency
- This would improve the sampling ratio

Simulation traces



The quality of the statistics made them unusable.

Covert data transmission

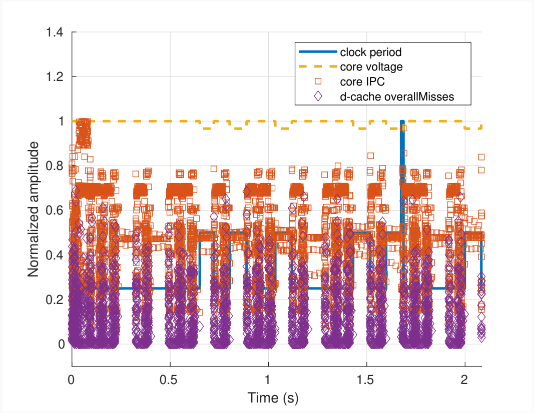
Intrinsic channels

- It is possible to use the power footprint of the platform to transfer information covertly
- The adversary leverages the activation of modules in the platform to produce a discernible pattern in the energy consumption of the circuit
- The receiver is generally another component within the system which would not normally be allowed to share a communications channel with the sender

Experimental setup (iii)

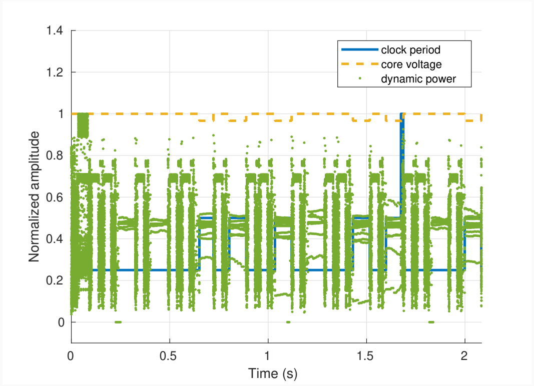
- Employ a trusted application (OP-TEE) to induce a power fluctuation on the platform
- Execute the `optee_example_aes` with a basic “test message” input once or twice

Simulation traces



The operations performed by the AES TA have a significant impact on the statistics.

Simulation traces



Which would produce a discernible pattern in the power footprint of the system.

Improving the simulation statistics

Filter statistics

How it started (2020)

"When `--stats-root` is given, only stats that are under the root *SimObjects* get dumped."

How it is going (2023)

1. Argument `--stats-root` is defined in *common/Options.py*
   - Even when *Options.py* is imported :
     "error: unrecognized arguments: –stats-root"
2. Parameter "stats_root" used in the `run` method of *common/Simulation.py*
   - Even though every simulation calls `run()` :
     Stats are not filtered even when `--stats-root` is is defined.

Filter statistics

How to do it

1. Define the `--stats-root` argument in the simulation script :

```
parser.add_argument("--stats-root", action="append", default=[])
```

2. Implement the stat filtering in the simulation script :

```
def main():
...
    instantiate(options)

    stat_root_simobjs = []
    for stat_root_str in options.stats_root:
        stat_root_simobjs.extend(root.get_simobj(stat_root_str))
    m5.stats.global_dump_roots = stat_root_simobjs
    ...
    run()
```

Filter statistics

Limitations

1. Can enable statistics at the level of *SimObjects* (cpu, l2, mmu)

```
--stats-root='system.bigCluster.clk_domain'
--stats-root='system.bigCluster.cpus[0]'
--stats-root='system.bigCluster.l2'
```

2. Custom *SimObjects* must define the `path_list` attribute

```
    def path_list(self):
        if self._parent:
            return self._parent.path_list() + [self._name]
        else:
            # Don't include the root node
            return []
```

Impact on the simulation

Using the Binary multiplication application, core frequency 100/200 MHz, dump rate of 1E-3.

Data volume

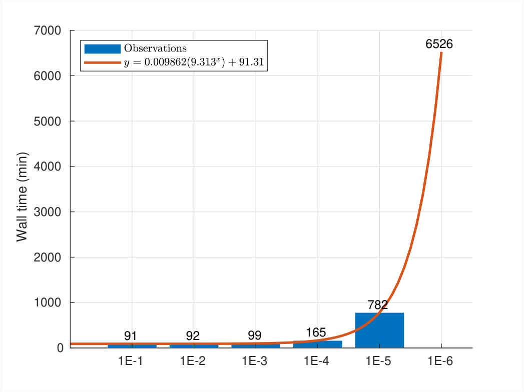
1GB $\rightarrow$ 864 MB

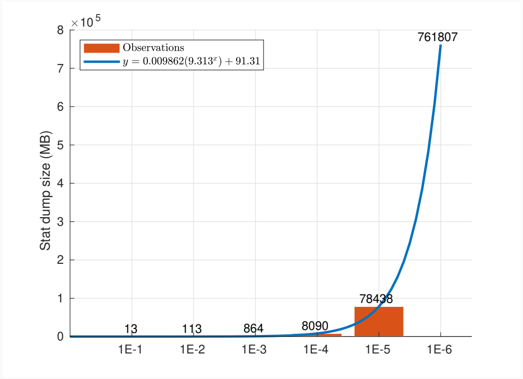
Simulation time

103 min $\rightarrow$ 99 min

Impact on the simulation

Using the Binary multiplication application, core frequency 100/200 MHz.





Some thoughts

Filtering

Create a *SimObject* whose attributes are the statistics of interest. Only whitelist this *SimObject*.

Dump rate

Power analysis does not require a large number of samples, just some samples at the precise time.

Final remarks

Conclusions

- We have investigated the feasibility of using the *gem5* simulator to emulate SCAs on microprocessor systems
- The use of a simulator would like *gem5* allow us to solve problems found in the study of power attacks
- However, some challenges must be overcome to improve the usability of this platform:
    - improve the production of statistics to make their reporting more concise
    - implement mechanisms for enabling or disabling statistics with fine granularity

Data availability

The multiple sets of statistics used in our experiments as well as the scripts created for processing the data can be accessed freely on *`https://github.com/CarlosAndresLARA/power-gem5`*

Thanks !