SHF:Medium:Making Analog Side Channels a First-Class Consideration in Architecture-Level Design

1. Introduction

Analog side-channels (power, electromagnetic, acoustic, etc.) have long been a potential source of attacks that circumvent traditional protections and security measures [2, 8, 9, 11, 13, 18, 31]. This is especially a problem for cyber-physical systems (CPS) and Internet-of-Things (IoT) systems which often contain sensitive data, such as sensor data, login information for over-the-network management of the system and/or accessing back-end cloud infrastructure, and are often placed in publicly accessible locations. For some side-channels, such as electromagnetic (EM) emanations, physical proximity can be leveraged to attack systems that are considered to be physically secure but are located near publicly accessible locations, e.g., in-wall "smart building" sensors, security cameras, etc [16, 30, 33, 81]. Many such attacks have been demonstrated over the past several decades, followed by countermeasures that prevent specific attacks by modifying the software that has been demonstrated to leak sensitive information. However, the PIs recent analog side channel attacks have shown that both attacks and mitigation are becoming increasingly dependent on microarchitectural behavior and potentially fragile to future microarchitectural changes [2, 3]. Unfortunately, early-design tools such as cycle-accurate simulators [7, 12, 17, 42, 43, 53, 58, 73] do not model analog side channel signals, so these side channels can only be considered when they can be physically measured on already-fabricated chips. At that time, however, time-to-market concerns prevent introduction of overall design changes that would adjust the design tradeoffs in a more desirable direction. Additionally, most software developers have neither the know-how nor the equipment to assess their software's potential vulnerability to analog side channels, so such considerations are typically either absent or qualitative/abstract when software is designed, giving first-mover advantage to attackers, and resulting in mitigation via localized patches, which themselves are becoming increasingly microarchitecture-dependent.

Ideally, the potential for information leakage through analog side channels and "breaking" existing software mitigation approaches would be considered in early stages of design for both hardware and software, guided by tools that can predict the impact a specific design has on analog side channels. This would be analogous to how performance and power consumption are predicted by cycle-accurate simulators, which allows the tradeoff between performance, power, and cost to be investigated at design time, years before the first prototype of that processor is fabricated [7, 12, 17, 42, 43, 53, 58, 73] If such efficient-yet-highly-accurate simulation would exist for analog side channels, hardware designers and architects could include analog side-channel leakage among their design considerations [4, 6, 49, 56, 64, 80], compilers could use simulation models to optimize for reduced leakage [32, 44, 55], software designers could detect and mitigate information leakage problems for security-sensitive applications [19, 72, 74], etc.

While there are some tools and metrics to quantify analog side-channel leakages [10, 15, 24, 46, 77], they are limited due to following reasons: *First*, they are mainly focused on developing metrics to estimate the information leakage itself, i.e., mutual information between the signal and the program secrets, rather than modeling the actual analog signal. Relying only on these metrics rather than analyzing the actual signal, may not be sufficient since these metrics inherently make assumptions about the aspects of the signal the attacker may exploit, i.e., they may not reveal *all* of the information the signal may contain. *Secondly*, existing methods only model the system at *architecture-level*, i.e., associating a (leakage) value to individual instructions based on the ISA, and ignore the micro-architecture activities such as pipeline stages, stall cycles, etc. on the signal. As

we have demonstrated in our seminal paper on analog side channel modeling [**?**], this can lead to significant inaccuracy, mainly because the model, by staying at the ISA-level, neglects to account for how that instruction interacts with other instructions and the underlying hardware. *Thirdly*, by neglecting the impact of micro-architecture, these methods implicitly assume that the entire hardware design is a *single source*, and then only model the analog emanations based on this single source. Such an assumption can lead to large inaccuracies since different micro-architecture components (e.g., cache, register-file, etc.) generate different electromagnetic waves with different polarization and/or phase, and hence, they may have *constructive or destructive* impacts on each other and the overall received signal.

The PIs first attempt to address these challenges and develop microarchitecture-level analog side-channel model was presented in [**?**]. The model was a proof of concept that very precise estimation of leakage not only for individual instructions but also for sequences of instructions (when the goal is to assess and improve leakage from a particular piece of code on a set of hardware platforms) is feasible. This model is also the first to asses leakage from a particular part of the system (when the goal is to make the design less "leaky") while maintaining the performance advantages of a cycle-accurate simulation relative to a physics-based model. While the proposed model is good enough for demonstrating possibility of modeling analog side channels, this model has many limitations: 1) it does not scale well across different architectures and processor clock frequencies, 2) Needs prohibitively large number of training sequences to cover all possibilities among opcodes and registers, 3) has large number of parameters that need to be estimated from measurements.

1.1. Proposed Research Work

To address these issues and consider jointly tradeoff between performance, power, and analog side channels we propose to develop:

- techniques that allow architecture-level simulators to efficiently generate estimated side channel signals, to help computer architects, researchers, and software developers assess the impacts of microarchitectural and software changes on the tradeoff between performance, power, and side channel leakage.
- methods for efficient circuit-level exploration of caches and functional units that can be integrated into architecture-level simulators, analogous to how Cacti and McPat are used to obtain per-event latency and power estimates in cycle-accurate simulators.
- **Machine learning methods for analog side channel modeling**. In this task we propose to develop machine learning techniques that will allow for analog side channel modeling that is not dependant on collecting measurements on particular processor and are scalable across different processor and memory speeds. Furthermore, we propose to create neural network structure that does not require to observe all possible combinations of instructions in order to model larger sequences of instructions. Finally, we propose to develop methods for "calibration" of simulation parameters against measured signals from real systems.

1.2. Broader Impact

We expect that our results will help the inclusion of analog side channels among early design considerations and will help reduce the cost of side-channel resistant designs by addressing side-channel-related problems early in the design process, when side-channel resilience may be improved (or preserved) with little or no sacrifice in performance, power, cost, weight, etc. The proposed

work is inherently interdisciplinary, combining expertise in computer architecture and circuits, hardware security, electromagnetic, and signal processing. Thus this research has the potential to improve the state of the art and have broader impacts in all these areas. Also, participating students will be working in a truly multidisciplinary context, which will broaden their expertise in ways otherwise not possible as well as **broaden participation in computing**. The proposal also includes 1) developing an interactive demonstrator for the general public, to educate and raise awareness about several key cyber-security concepts and issues, 2) visits and activities in local schools to improve K-12 education and participation of women and minorities in STEM, and 3) course and curriculum development activities at the undergraduate and graduate level.

2. Related Work

There is a large body of work focused on preventing particular side-channel attacks, e.g., [2, 3, 8, 24, 34, 36, 45, 50, 63], either by removing the tie between sensitive information and the side-channel signal, or by trying to make the signal more difficult to measure. However, such work mostly focuses on preventing a particular side channel attack in a very specific piece of code and are less focused about the fundamental relationship between the hardware, software, and the side-channel signal.

Strategies for quantifying potential side channel exposure at the micro-architectural and architectural levels are still an open problem. Existing work proposed different methods and/or metrics to estimate the leakage either for a specific type of side-channel (e.g., cache, power, EM, etc.) or alternatively, as a generic framework to estimate the overall leakage for any given side-channel.

Side-Channel Vulnerability Factor (SVF) [24] measures how the side-channel signal correlates with high-level execution patterns (e.g., program phase transitions). While this metric allows overall assessment of the "leakiness" of a particular system and application over a given side-channel, it provides limited insight to 1) computer architects about which architectural and microarchitectural features are the strongest leakers, and to 2) software developers about how to reduce the side-channel leakiness of their code.

To address these limitations, Signal Available to Attacker (SAVAT) method [15] was proposed. SAVAT measures the side-channel signal (particularly EM and power from laptops) created by a specific single-instruction difference in program execution, i.e., the amount of signal made available to a potential attacker who wishes to decide whether the program has executed instruction/event A or instruction/event B. These measurements can be used to determine the potential for information leakage when execution of individual instructions or even sections of code depend on sensitive information. Unfortunately, SAVAT only models the system at ISA-level and ignores the underlying relation of each instruction to the hardware or other instructions in the sequence. Extensions of SAVAT that include modeling instructions overlapping in pipeline have been proposed in [76], [79].

Similar to SAVAT, McCann *et al.* [46] proposed a modeling technique capable of producing a leakage metric at instruction-level for power (and/or EM) side-channel signals on ARM M0/M4 cores. To estimate the leakage for individual instructions, the proposed method only requires knowledge about different characteristics of the system at ISA-level such as data-dependent effects of neighboring instructions in a sequence, register effects, bit-flips, etc. Similar to SAVAT, while the method proposed by McCann *et al.* [46] provides interesting insights about possible sources of leakage, it also ignores the effects of micro-architecture events such as cache miss, branch miss-prediction, etc. on the signal.

The method proposed by Barenghi and Pelosi [10] calculates the leakage for individual instructions by measuring the power consumption between two consecutive cycles and employs the Pearson

correlation coefficient between the two measurements. To calculate the leakage, in addition to leveraging the ISA-level information, pipeline model was also used. However, the framework did not consider any micro-architecture events, nor pipeline stalls and only accounts the number of cycles that takes for each instruction to execute. It also did not model the individual effect of each stage on the others and the overall signal.

Another approach to quantifying side channel leakage is to use information theory and estimate capacity of analog side channels. Millen was the first to establish a connection between Shannon's information theory and information flow models in computer systems [48] and calculated the capacity (maximum possible data rate) of such a side-channel. However, that model assumes a synchronous channel (where information is transmitted at a constant rate that is known to the receiver), and this is not a realistic assumption for analog side-channels in computer systems, where the timing of execution in the spy program varies due to a number of hardware features (e.g., pipeline stalls, dynamic scheduling of instructions, cache hits and misses, branch prediction, etc.). Additionally, analog side-channels often include insertion, deletion, and errors, e.g., interrupts and other system activities often insert activity into the timeline of the spy program's execution. There are many papers that discuss bounds on the capacity of channels corrupted by synchronization errors [71], [5], [22], [23], [69], [41], [37], bounds on the capacity of channels corrupted with synchronization and substitution errors [70], [54], [47], or bounds on the capacity when codewords have variable length but no errors in the channel [70], [62], none of them provides the answer to how much information is "transmitted" by execution of particular sequence of instructions that do not have equal timing and are transmitted through erroneous channel. The first attempts to answer this question were presented in [75, 78], where covert channels are generated, and upper and lower leakage capacities were derived. In [76], a side channel leakage capacity is derived for a discrete memoryless channel where it was assumed that each transmitted quantum of information (i.e., instruction in the code) is mutually independent but do not have equal length. Although all these papers make an important step toward assessing information leakage from side-channels, they fall short of considering the relationship among sequence of instructions, which is a result of program functionality as well as a processor pipeline depth, which impacts how much signal energy will be emanated. In [79], side-channel information capacity created by execution of series of instructions (e.g., a function, a procedure, or a program) in a processor is derived. To model dependence among program instructions in a code, we use Markov Source model, which includes the dependencies that exist in instruction sequence since each program code is written systematically to perform a specific task. The presented framework considers processors as the transmitters of a communication system with multiple antennas. The antennas correspond to different pipeline stages of any processor. Moreover, inputs of the transmitter show dependency based on a Markov model which reflects the practicality of a program. Using this setup, we have obtained the channel capacity of a communication system which represents the severity of the side channels. However, none of these approaches provide enough details to model analog side channels taking into consideration microarchitecture details.

Another body of work related to this proposal are the cycle-accurate models/tools to simulate power and/or microarchitecture [7, 12, 17, 42, 43, 53, 58, 73]. While these models can accurately model the power consumption at each cycle, they are different from this work and hence may not be a proper tool for simulating analog side-channel signals for two main reasons: *First*, while these methods do consider the activity factor to calculate power, they often treat all the bit-flips equally. However, as shown in [**?**], depending on the design, not all flips equally contribute to the overall signal. Ignoring this fact can lead to inaccurate modeling. *Second*, depending on the architecture, different stages

might have different effect on each other and the overall signal. Without properly modeling these effects, the overall signal can not be modeled.

Also related to this work are work on leveraging EM signals for program profiling, tracking, and analysis [14,25,26,32,59,60,67]. Spectral Profiling methods [14,59,60,67] tracks program execution at different granularity from loops [60], to basic blocks [14, 59] and individual instructions [67]. EMProf profiles memory [25] and Primer profiles interrupts [26], both leveraging electromagnetic (EM) emanations from devices. By continuously analyzing these EM emanations, EMProf identifies where in the signal's timeline each period of stalling begins and ends, allowing it to both identify the memory events that affect performance the most (LLC misses) and measure the actual performance impact of each such event (or overlapping group of events). Because EMProf is completely external to the profiled system, it does not change the behavior of the profiled system in any way, and requires no hardware support, no memory or other resources, and no instrumentation on the profiled system. Similarly, Primer analyzes the device's external EM side-channel signal in real-time, without any interference with the device's program execution, while providing a detailed analysis of not only the overall overhead created by interrupts, but also their distribution over time (i.e., exact occurrence of interrupts in program execution time-line). As the CPU follows a generic procedure to handle such asynchronous system events, our approach can be generalized and applied to all non-deterministic interrupts across multiple platforms. More details on both methods are presented in following sections. All these methods are complimentary to this proposal and combined with proposed modeling tool can enhance leakage estimation, compiler development, etc.

However, none of them address the need for developing microarchitecture-level analog side-channel tool such that allows for integration of such modeling into a cycle-accurate simulator. This, in turn, would allow analog side channels to become a first-class consideration, along with performance and power, in processor designs, allowing computer architects to avoid introducing significant new vulnerabilities and "breaking" existing software mitigation, and possibly even to reduce leakage and/or enable new mitigation.

3. Preliminary Results

The PIs are the first to attempt to address these challenges and develop microarchitecture-level EM side-channel model [**?**].

Fundamentally, EM side-channel signals are created due to *bit-flips* at the transistor-level [68, 81]. In principle, all transistors and metal-layer interconnect components contribute to the signal, thus the signal could be modeled using all transistors and on-chip wires as predictor variables, which *should* be highly accurate but is practically infeasible. As a result, the main challenge in model-building is to select (or discard) potential predictors in a systematic manner, to achieve a trade-off where feasibility (or even efficiency) is achieved without a major sacrifice in accuracy.

To accurately simulate EM signals, we model micro-architectural components as ***independent*** sources of EM emanations and then further group these units in each pipeline stage as an individual source. We use pipeline stages as the sources mainly because we observed that each instruction has different footprint in each cycle, and the side-channel generated at each cycle is a combination of these activities in *ALL* stages. Using this methodology, we model a multi-input (pipeline stages), single-output (EM signal) system (MISO).

Leveraging this approach, the **challenges** are ***a)*** *how to model the signal for individual sources*, and ***b)****how to properly combine the signals* generated by each source to accurately form the side-channel signal.

3.1. Signal Amplitude for Individual Sources

In practice, there are two contributors in creating EM side-channel signals for each pipeline stage. The first group of contributors, which we call *instruction-dependent* activities, are caused by the switching activities of micro-architectural units (e.g., register-file, ALU, etc.) that are utilized in that stage (e.g., whether the register-file is being written or not).

The second group, *data-dependent* activities, are created due to bit-flips on the data-bus, address-bus, and any other registers that hold operand's values. These bit-flips are independent from the instruction-type but are dependent to the previous state of the bus. In the following, we will describe how we *independently* measure each of these two groups.

Instruction-Dependent Activities. To independently measure these groups, we first minimize the effect of the *data-dependent* activities by setting all the operands, addresses, and immediate values to zero. This approach enables us to measure the *baseline* signal for each stage which is *only* created by the switching activities of the micro-architectural units used in that stage.

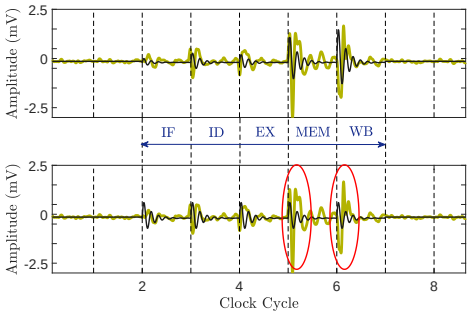

Figure 1: The signal amplitude for an `ADD` as it progress in the pipeline (while all other instructions are `NOP`). The actual signal is shown in light color (green). Darker color (black) shows the simulated signal when considering each pipeline stage as a separate source (top), and when considering the entire processor as a single source (bottom), and the largest differences between the two are pointed out using red ellipses.

After decoupling the data-related activities from the signal, the second challenge is to minimize the effects of other stages on the generated EM signal. Recall that we mentioned ALL pipeline stages contribute to the overall signal, however, ideally we want to be able to measure the effect of each stage *separately* so that we can use them as the basic-blocks to reconstruct the overall signal. To achieve that, we use `NOP` instruction as the *baseline* since it has the minimum possible switching activity, and then create `NOP` $\rightarrow$ `inst` $\rightarrow$ `NOP` instruction sequence (for all instructions), while operands for `inst` are all set to $r1$ (and $r1 = 0$). Using this method, no data/operand-dependent bit-flips are created, but register-file, ALU, etc. may be used (depending on the instruction type). We then measure the signal amplitude for all instructions and every pipeline stages. We call this ***baseline hardware amplitude*** or A.

Figure 1 shows how the (actual) EM signal (shown in green/light color) changes as an `ADD` instruction progresses through the pipeline while all the other instructions are `NOP`. Using Equ. 1 [**?**] and `NOP`$\rightarrow$ `inst`$\rightarrow$ `NOP` instruction sequence, we used our simulator to generate the signal.

$$y(t) = \sum_{n=0}^{+\infty} x[n] \sin(\frac{2\pi(t-nT)}{T_0}) e^{-\theta(t-nT)} \mathtt{u}(t-nT). \quad (1)$$

This equation represents two main effects we observed in the analog side-channel signal: 1) *switching activity in a processor is synchronized to its clock* and most of the switching happens

right after the positive/negative edge of the clock, which implies that activity is not evenly spread over a cycle and a decaying function can be used to model clock activities; 2) *the received signal is also exposed to oscillations with decreasing magnitude*, i.e., small signal variations can be modeled as sinusoidal oscillations. Further, to show why individual stages should be modeled separately, Figure 1 (bottom) shows the simulated signal when the "average" amplitude is used for all stages. As can be seen, failing to model each stage individually (as used in previous work [46]) can lead to significant inaccuracies in some stages (note that using *max* instead of *average* also leads to similar inaccuracies).

Data-Dependent Activities. Once the baseline amplitude is measured, the next step is to find how this amplitude changes as the number of bit-flips changes due to value/operand used in the instruction and the previous state of the bus/register. Intuitively, the more bit-flips, the higher the amplitude should be thus we define *activity-factor*, α, as a ***scaling factor*** to the baseline activity, A. To find α, we first treat each bit-flip *equally*, and assume that each bit-flip has similar effect on the signal amplitude. We then calculate α as:

$$\alpha = 1 + \frac{(flips_{new} - flips_{base})}{flips_{total}}, \tag{2}$$

where $flips_{new}$ is the total number of flips for the current instruction, $flips_{base}$ is the total number of flips when previous instruction is `NOP`, and $flips_{total}$ is the maximum possible number of flips for the current instruction. Using this equation, we then define $A' = \alpha \times A$, and use it to simulate the signal. Figure 2 (bottom) shows the original signal (shown in light green), and the simulated signal using this approach (shown in black) for the similar `NOP`→ `inst`→ `NOP` instruction sequence discussed in the previous section. As can be seen in the figure (bottom), this *"averaging"* modeling can not accurately predict the amplitude of the signal which indicates that *not all the bit-flips have the similar impact on the amplitude*. Our further investigation confirmed this theory. Particularly, we found that flips in the output of the ALU and memory have the most significant impacts on the signal. We believe this difference is mainly due to the different physical parameters of transistors and/or lengths of the connecting wires.

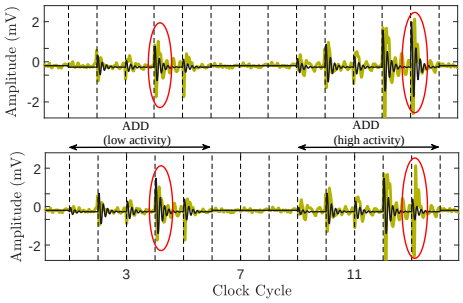

Figure 2: Effect of the *activity factor* on the amplitude. The actual signal shown in green. The simulation is shown in black when activity factor is modeled using a linear regression model (top) and when an *average* activity is used (bottom).

Using this observation, to systematically calculate the activity factors, we use a *linear regression* model:

$$\alpha = \delta + \mathcal{T} \times c + \epsilon, \tag{3}$$

where $\mathcal{T}$ is a vector of transition bits across all the existing registers in the targeted pipeline stage, δ and ϵ are the vector of scalar intercept and error terms respectively, and c is the vector of activity factors to be predicted by the model. As mentioned before, α is the scaling factor for the baseline amplitude, A, thus $\alpha = A_{meas}/A_{simul}$. Note that to find $\mathcal{T}$, a detailed micro-architecture model is needed to track all the bit-flips for every gate in the processor (except cache/memory). However, to significantly reduce the complexity and simulation time, the size of $\mathcal{T}$ can be reduced using the step-wise regression method [35] where, iteratively,

the size of the fitted model (i.e., α and $\mathcal{T}$ in our case) is reduced using standard statistical metrics such as F-tests [35]. In other words, since *not all the bit-flips have statistically significant impact on the emanated signal*, the non-contributing factors can/should be removed from the model. In our processor, using this method we managed to reduce the size of $\mathcal{T}$ by more than 65%.

Figure 2 (top) shows the simulated signals when the linear regression (LR) model is used for activity factors. Compared to the averaging method (bottom), using LR has significantly improved the simulation accuracy.

3.2. Multi-Input Modeling

Once the signal amplitude for individual sources are calculated, the next step is to combine the signals generated by these individual sources to create the simulated EM signal. In principle, the generated EM signal is the *superposition* of individual waves thus depending on each source's phase, the superposition of each pair can be either *constructive* or *destructive*. Using this fact, the overall signal can be approximated as a *linear* combination of these individual sources where the coefficients may vary between ± 1, depending on the phases.

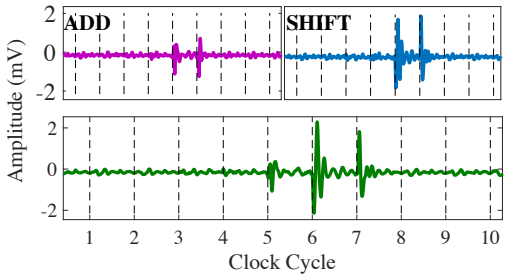

Figure 3: An example of how individual sources (pipeline stages) are combined to form the final signal. ***Top***: how the actual EM signal looks like when the instructions are executed in isolation (`NOP, inst, NOP`). ***Bottom:*** The actual EM signal when the instruction sequence is `NOP, ADD, SHIFT, NOP` (i.e., a combination of multiple instruction in the pipeline).

Due to the complex nature of the generated EM signals, accurately modeling each and every source mathematically is significantly time-consuming and often infeasible in practice. To tackle this problem and find coefficients for each source, a *model-fitting* approach can be used. We use a *linear-regression* model to find (predict) the overall EM signal. Specifically, we use:

$$X = \delta_s + (\alpha A) \times M + \epsilon_s, \tag{4}$$

where αA is the vector of individual sources amplitudes (α is the activity factor and A is the baseline amplitude), δ and ϵ are the intercept and error vectors, M is the predicted coefficients, and X is the final amplitude which will be used in Equ. 1 to simulate the signal.

Figure 3 shows an example of how two individual sources are combined in each cycle to form the final signal. Figure 3 (top) shows `ADD` and `SHIFT` instructions when they are executed in isolation (i.e., `NOP, inst, NOP`), and Figure 3 (bottom) shows how the final signal looks like when the executed sequence is `NOP, ADD, SHIFT, NOP`. Specifically, cycle 6 is when the `ADD` instruction is in `WB` stage and `SHIFT` is in `MEM`, and the resulting signal is a linear combination of these two sources. Note that to find M, we need to measure all the possible combinations of the entire instructions in the ISA, however, as we will show in Section 5, the number of required measurements can be significantly reduced using standard *clustering* algorithms.

3.3. Modeling of Micro-Architectural Events

The last step of simulating an EM side-channel signal is adding the signatures of different micro-architectural events to the signal. We particularly add the signatures of the following three events to the signal:

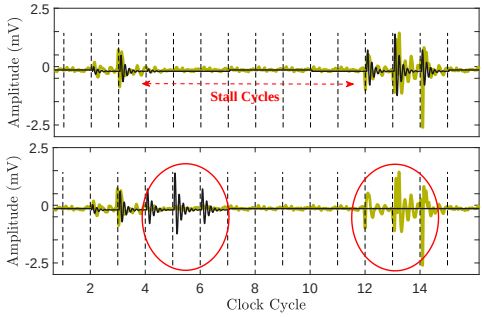

Figure 4: Effect of stalls on the signal. The actual signal shown in green, while simulated signals are shown in black when modeling pipeline stalls (top) and not modeling it (bottom).

Pipeline Stall. Stalling is a common event in a processor which prevents successor instructions from advancing in the pipeline and preserves the instruction and operands in the stalled stages. Due to this preservation no bit-flips occur in the stalled stages. In addition, to save power, a control signal is typically used to disable (e.g., through power-gating) hardware components in the stalled stages. As a result, stalling typically has a dramatic impact on the switching activities of the stalled stages and, consequently, results in a significant reduction in the amplitude of the generated side-channel signals. Figure 4 shows the effect of stalling on the signal where a `MUL` instruction has stalled the pipeline for eight cycles (we intentionally increased the stall cycles in `MUL` for clarity). As can be seen from the figure, not properly simulating stalls (bottom) results in a significant deviation from the original signal (shown in light green).

Note that stalling does not have any impact on prior instructions thus they still generate side-channel signals as they advance through the pipeline. As a result, during stall the received side-channel signal is only generated by the instructions in the non-stalled stages (if any).

To properly model stalling, the simulator should be able to detect when stalls are happening (using the micro-architecture model), and ignores the signals generated by the stalled stages during the stall phase. In our model, this is done by setting the amplitudes of stalled stages to zero in Equ. 4.

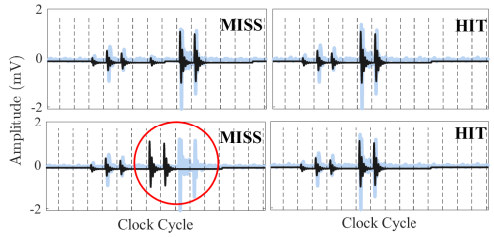

Figure 5: Effect of cache-miss (left) and cache-hit (right) on the signal. Miss causes two extra stall cycles. The actual signal (light blue) and simulated signals (black) with (top) and without (bottom) modeling cache misses are shown.

Cache miss. Similar to pipeline stalls, due to a data-dependency, cache miss can also cause stalls. In our design, accessing the cache stalls the pipeline for one cycle. Further, cache miss and access to the memory causes extra two stall cycles. These two signals and their differences are shown in Figure 5.

As can be seen in the figure, two extra stall cycles (total of three) can be seen in `LD` instruction. Similar to stalls, the cache activity should be properly simulated using the micro-architecture model. Figure 5 illustrates how without properly modeling the cache misses the simulated signal will be deviated from the original signal (bottom left).

Misprediction. In addition to stalls, we observed that branch misprediction also has noticeable impact on the side-channel signals. Depending on the pipeline design, the correct outcome of a branch instruction can be resolved after a few cycles (2 cycles in our design), and if a *misprediction* is detected, the processor has to *flush* the incorrectly fetched instruc-

tions, and begin executing the correct ones after that. In order to do that, processors typically substitute the incorrect instructions with `NOP` instructions. It is expected that executing these *bubble* instructions temporarily changes the side-channel signals since they change the switching activities of each stage. Figure 6 shows the received EM signals with and without a misprediction along with instructions present at each cycle in each pipeline stage.

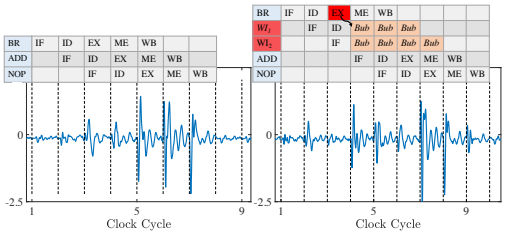

Figure 6: Effect of misprediction (right) on the signal. It causes two instructions being flushed from the pipeline and hence affect the signal in those cycles.

Similar to pipeline-stall, using the micro-architecture model, mispredictions can be detected and simulated in our simulator. It is important to mention that we also studied the impact of using different branch-predictors on the side-channel signals (e.g., always not-taken, 2-level, g-share, etc.) and we did not observe any statistically significant difference between these predictors mainly because they have relatively small switching activities (especially for low-end processors).

It is also important to mention that *we tested the effect of other micro-architectural events* such as data-forwarding on the signal and did not observe any significant difference in the presence and/or absence of them. Also note that, as we mentioned before, in this paper, we limited the modeling to bare-metal, and left system-level activities modeling such as interrupts, exceptions, context-switch, etc., and advanced power-saving methods (e.g., DVFS, power gating, etc.) to future work.

3.4. Evaluating Model Accuracy

Table 1: RISC-V (R32IM) instruction-set and their cluster used in this paper.

Cluster	Type	Inst.	No. Inst.
1	ALU	`ADD,XOR,JAL, ...`	13
2	Shift	`SLLI,SRT, SRA, ...`	10
3	MUL/DIV	`MUL, DIV, REM, ...`	8
4	Load	`LB, LW, LH, ...`	5
5	Store	`SB, SH, SW`	3
6	Cache	`LB, LW, LH, ...`	5
7	Branch	`BEQ, BLT, BGE, ...`	6

Setup. We implemented a RISC-V based processor on a Terrasic DE0-CV board with an Altera Cyclone-V FPGA [66] with 50 MHz clock-rate. To record side-channel signals, we used a Keysight digital oscilloscope (DSOS804A), with 1 GHz bandwidth and 10 GSa/s rate. We further studied the effect of changing the sampling-rate on the accuracy and found that similar accuracy can be achieved with much lower sampling-rate (about 200 MSa/s in our measurements). As a result, similar results can be achieved using a less expensive device (e.g., TBS1032B Tektronix Digital Oscilloscope [65] costs around $300) and/or a high sampling-rate device can be used for modeling devices with faster clock-rates.

To receive EM signals, we used a magnetic probe [1], placed 5 cm above the FPGA. Signal processing is done in Matlab2017-b and the simulator is implemented in standard C++ programming language.

Model Building. In order to fit a model, *ALL* possible combinations of instructions should be measured (i.e., about three hundred million combinations in RISC-V ISA). Clearly such a

requirement makes the model building extremely time-consuming in practice. However, intuitively, we expect instructions with similar behaviors (e.g., ALU-type, memory-type, etc.) have similar side-channel signals since they share identical hardware activities. Using this intuition, we used the *hierarchical agglomerative algorithm* [27] with the *cross-correlation* as the distance metric to cluster instructions with similar EM pattern into a same cluster. We found that RISC-V ISA can be *clustered* into 7 categories (when the operands are similar) where a single instruction in each category can be a representative of all instructions in that category.

These categories are shown in Table 1. Using this table, we then used only a *representative* instruction of the cluster for model building which, in turn, reduce the model building complexity significantly. In our setup, the number of measurements was reduced from 300 million to only 16 thousands. Note that while the clustering algorithm did not use the micro-architecture model as a prior knowledge, the clusters confirmed that instructions with similar micro-architecture activities should be clustered in a same group.

Metric. To measure how well the simulated signals "match" with the real signals, we leverage *normalized cross-correlation* as our metric. To compute that, we first normalize both signals, real and simulated, to have similar average. We then divide each signal to individual clock cycles, and then compare each cycle (between the simulated and the real signals) using cross-correlation as the distance metric. We then define *accuracy* as the average of this cross-correlation across all cycles for all measurements (i.e., we were able to match the waveform in this degree across all possible instruction sequences). Note that we specifically used this approach to show how well the time-domain signal *matches* with the original signal instead of relying on a specific *leakage metric* such as Hamming weight. However, to show the usefulness and versatility of our tool, those results will be shown in the next section.

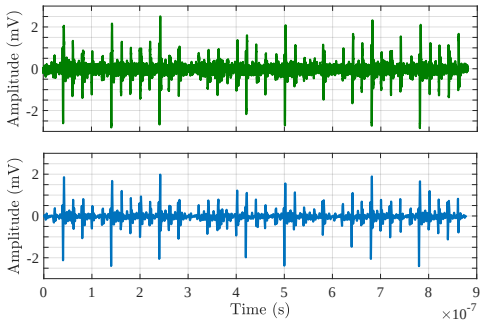

Figure 7: A comparison between the signal generated by a real hardware (top) and the simulated signal (bottom) in EMSim.

Benchmark. To prove that our approach provides accurate simulated signals for *ALL* possible instruction combinations thus can be applicable to *ANY* complex program that uses the mixture of the implemented ISA (R32IM), we created a microbenchmark using all possible combinations of the representative instructions shown in Table 1. Particularly, for a 5-stage pipeline and 7 distinct clusters, there are $7^5 = 16807$ possible combinations that can appear together (in the pipeline) in a cycle. We created a program to generate all these combinations with random operands. We then manually modified branch instructions and assigned the target address and branch condition to create loops with random instruction and iteration sizes. To limit the execution time, we then randomly put these instructions into groups of 1024 combinations (i.e., 5120 instructions in each group which were executed one after another similar to a real program). To cover all the combinations, 17 of such groups were needed (no two groups were similar). We then executed these randomly-generated groups on the processor normally, and recorded the real and simulated signals. To further prove the validity and correctness of our simulator, we also randomly created another 17 groups, this

time from all instructions in the ISA and not just the representatives.

Results. Using these 34 groups/applications described above, we then compared the simulated signals with the actual ones using the our metric defined earlier. Each group/application takes about 9000 cycles to finish on average. The execution-time varied depending on the instructions used and microarchitectural events.

Figure 7 shows the simulated and actual EM-side-channel signals for one of the groups tested in our evaluation (for clarity, only the first 50 cycles are shown in the figure). As can be seen from the figure, the simulated signal matches the real signal with high accuracy. We found that, on average, ***EMSim has about 94.1% accuracy in simulating side-channel signals across all possible instruction combinations***.

4. Proposed Research

5. Integrated Research, Education, and Outreach Plan- EDIT

The research, education, and outreach milestones for each year are outlined in the following table.

Year	Research	Education and Outreach
1		Develop initial modules for graduate courses, visits to local schools, undergraduate teams work on lab measurement setups
2		Develop course modules for graduate and undergraduate courses; visit school for hands-on demonstrations, undergraduate teams experiment with refinements to measurement, training, and matching methodology
3		Refine course modules; develop K-12 and public demonstrators for sensory experiences of HT detection (e.g., use software defined radio setups to demonstrate HT detection); fully integrate undergraduates in research

The PIs plan to conduct the proposed research in a highly integrated manner. Three PhD students will work on this project, and we will seek out students whose primary expertise is in one of the areas relevant to this project, but who are willing to learn and contribute to the other areas. We also anticipate that most (or all) of the students will be co-advised. Finally, we will hold weekly meetings to both keep the project well-coordinated and to foster exchange of ideas across the disciplines involved in this project.

Our interdisciplinary approach requires technical and research expertise from several Computer Science (CS) and Electrical Engineering (EE) disciplines, and each PI brings skills essential for the success of this work.

5.1. Proposed Outreach and Education Activities

In addition to research proposed here, this proposed work will help foster interaction among researchers from a very diverse set of research areas and will involve training students with a considerable multidisciplinary expertise. To further increase the impact of this work, we are also planning to perform the following outreach and education activities.

- We will introduce relevant cross-disciplinary material into several courses at the graduate and undergraduate level. For example, we will add to VLSI and computer architecture courses an introduction to physical side-channel (EM) signals. Similarly, in electromagnetics, telecommunications, and signal processing classes, we will add a discussion of how computer hardware and software interact to create side-channel signals. This will help students understand the broader perspectives relevant to each class, and also help them appreciate the increasingly multi-disciplinary nature of science and technology.
- We will continue to include undergraduate students in our research. The PIs Zajic and Prvulovic have long history of advising undergraduate students (26 undergrads) through Opportunity Research Scholar (ORS) program at Georgia Tech and individual motorship and plan to continue that activity.
- Another outreach activity will be the development of displays and tools for educating the general public, especially high-school students and teachers, on issues related to both HTs and side-channel signals. This initiative will be pursued through Georgia Tech's established outreach programs for high schools in the greater metro Atlanta area [21]. The PIs Zajic and Prvulovic have done this in the past with hands-on demonstration for the Family Science Night at the Mimosa Elementary School in Roswell, GA and hands-on STEM activities at Pace Academy in Atlanta, GA.

6. Results from Prior NSF Support

Dr. Prvulovic and Dr. Zajic have served as PIs and Co-PIs on the following NSF-funded projects in the last five years CCF-1563991 SHF (from 2016-2022 $\$850K$) and CNS 1740962 (from 2017-2019, $\$199,866$).

Grant CCF-1563991 SHF *Spectral Profiling: Understanding Software Performance without Code Instrumentation* ($\$850K$, October 2013- September 2016). *Intellectual Merit:*Our seminal work on EM emission based program profiling provided the basis for future work in the more general area of program analysis techniques that leverage the physical side effects of computation [14,59,61,67]. *Broader Impact:*This work has opened up new possibilities in a number of additional areas, including program testing and debugging and software security. Inherently interdisciplinary, research has improved the state of the art in many areas including electromagnetic, signal processing, computer architecture, and software engineering.

Grant CNS 1740962*EAGER: Exploration of THz Backscattering as a Side-channel in Computer Systems* ($\$107K$, September 2010–August 2012). *Intellectual Merit:*Our seminal work on backscattered signals and how they can be used for HT detection has opened up new possibilities for non-destructive testing of electronics and for more precise detection of dormant HTs [38, 51, 52]. *Broader Impact:*THz back-scattering side-channel is an entirely new side channel that significantly differs from existing ones for both defensive and offensive uses. We have built several testbeds and demos and demonstrated it to various communities to educate public about this new technology. We have received several best paper and patent awards.

Dr. Zajic is currently also the PI on grant ECCS-1651273 NSF CAREER(from 2017-2023 $\$500K$) *Modeling and Measurements for THz Wireless Chip-to-Chip Communications Propagation Intellectual Merit:*To enable future THz wireless communication between chips in a system and between blades and racks in large-scale data center systems, this project is investigating propagation mechanisms [20], [39], [28] and develops channel models [40], [29] to enable communication between chips on a motherboard inside a computer system, between the motherboard and an add-on card (e.g., a graphics card), between boards/blades in a rack-mounted system typical for base stations, and

between racks in a data center environment (with raised floors, rows of racks, cooling ducts, etc.). *Broader Impact:* In addition to the broader impact within the research community and training graduate students funded by this grant, the PI is leading mm-wave indoor channel modeling and measurements efforts in the 5G Millimeter Wave Channel Model Alliance, led by National institute of Standards and Technology, which aims to produce more accurate, consistent and predictive channel models for frequencies above 6 GHz and lead the standardization process [57].

References Cited

[1] AARONIA. Datasheet: Rf near field probe set dc to 9ghz. `http://www.aaronia.com/Datasheets/Antennas/RF-Near-Field-Probe-Set.pdf`, 2019 (accessed April. 1, 2019).

[2] M. Alam, H. A. Khan, M. Dey, N. Sinha, R. Callan, A. Zajic, and M. Prvulovic. One&done: A single-decryption em-based attack on openssl's constant-time blinded RSA. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 585–602, Baltimore, MD, 2018. USENIX Association.

[3] M. Alam, B. Yilmaz, F. Werner, N. Samwel, A. Zajic, D. Genkin, Y. Yarom, and M. Prvulovic. Nonce@once: A single-trace em side channel attack on several constant-time elliptic curve implementations in mobile platforms. In *2021 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 507–522, 2021.

[4] A. Althoff, J. McMahan, L. Vega, S. Davidson, T. Sherwood, M. B. Taylor, and R. Kastner. Hiding intermittent information leakage with architectural support for blinking. In *Proceedings of the 45th Annual International Symposium on Computer Architecture*, ISCA '18, pages 638–649, Piscataway, NJ, USA, 2018. IEEE Press.

[5] R. J. Anderson and F. A. Petitcolas. On the limits of steganography. *IEEE Journal on selected areas in communications*, 16(4):474–481, 1998.

[6] M. Andrysco, A. Nötzli, F. Brown, R. Jhala, and D. Stefan. Towards verified, constant-time floating point operations. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 1369–1382, New York, NY, USA, 2018. ACM.

[7] E. K. Ardestani and J. Renau. Esesc: A fast multicore simulator using time-based sampling. In *Proceedings of the 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, HPCA '13, pages 448–459, Washington, DC, USA, 2013. IEEE Computer Society.

[8] M. Backes, M. Dürmuth, S. Gerling, M. Pinkal, and C. Sporleder. Acoustic side-channel attacks on printers. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, pages 20–20, Berkeley, CA, USA, 2010. USENIX Association.

[9] J. Balasch, B. Gierlichs, O. Reparaz, and I. Verbauwhede. Dpa, bitslicing and masking at 1 ghz. *IACR Cryptology ePrint Archive*, 2015:727, 2015.

[10] A. Barenghi and G. Pelosi. Side-channel security of superscalar cpus: Evaluating the impact of micro-architectural features. In *Proceedings of the 55th Annual Design Automation Conference*, DAC '18, pages 120:1–120:6, New York, NY, USA, 2018. ACM.

[11] D. J. Bernstein, J. Breitner, D. Genkin, L. Groot Bruinderink, N. Heninger, T. Lange, C. van Vredendaal, and Y. Yarom. Sliding right into disaster: Left-to-right sliding windows leak. In

W. Fischer and N. Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, pages 555–576, Cham, 2017. Springer International Publishing.

[12] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, Aug. 2011.

[13] J. Brouchier, T. Kean, C. Marsh, and D. Naccache. Temperature attacks. *IEEE Security Privacy*, 7(2):79–82, March 2009.

[14] R. Callan, F. Behrang, A. Zajic, M. Prvulovic, and A. Orso. Zero-overhead profiling via EM emanations. In *Proceedings of the 25th International Symposium on Software Testing and Analysis, ISSTA 2016, Saarbrücken, Germany, July 18-20, 2016*, pages 401–412, 2016.

[15] R. Callan, A. Zajić, and M. Prvulovic. A practical methodology for measuring the side-channel signal available to the attacker for instruction-level events. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-47, pages 242–254, Washington, DC, USA, 2014. IEEE Computer Society.

[16] G. Camurati, S. Poeplau, M. Muench, T. Hayes, and A. Francillon. Screaming channels: When electromagnetic side channels meet radio transceivers. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 163–177, New York, NY, USA, 2018. ACM.

[17] T. E. Carlson, W. Heirman, S. Eyerman, I. Hur, and L. Eeckhout. An evaluation of high-level mechanistic core models. *ACM Transactions on Architecture and Code Optimization (TACO)*, 2014.

[18] S. Chari, J. R. Rao, and P. Rohatgi. Template attacks. In *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '02, pages 13–28, London, UK, UK, 2003. Springer-Verlag.

[19] J. Chen, Y. Feng, and I. Dillig. Precise detection of side-channel vulnerabilities using quantitative cartesian hoare logic. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 875–890, New York, NY, USA, 2017. ACM.

[20] C.-L. Cheng and A. Zajic. Characterization of propagation phenomena relevant for 300 ghz wireless data center links. *IEEE Transactions on Antennas and Propagation*, 68(2):1074–1087, 2020.

[21] L. Conrad. Ece outreach:@ georgia tech. `https://www.www-new.ece.gatech.edu/outreach`.

[22] V. Crespi, G. Cybenko, and A. Giani. Engineering statistical behaviors for attacking and defending covert channels. *IEEE Journal of Selected Topics in Signal Processing*, 7(1):124–136, 2013.

[23] M. Davey and D. MacKay. Reliable communication over channels with insertions, deletions, and substitutions. *Information Theory, IEEE Transactions on*, 47(2):687–698, Feb 2001.

[24] J. Demme, M. Maycock, J. Schmitz, A. Tang, A. Waksman, S. Sethumadhavan, and S. Stolfo. On the feasibility of online malware detection with performance counters. In *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ISCA '13, pages 559–570, New York, NY, USA, 2013. ACM.

[25] M. Dey, A. Nazari, A. Zajic, and M. Prvulovic. Emprof: Memory profiling via em-emanation in iot and hand-held devices. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 881–893, 2018.

[26] M. Dey, B. B. Yilmaz, M. Prvulovic, and A. Zajic. Primer: Profiling interrupts using electromagnetic side-channel for embedded devices. *IEEE Transactions on Computers*, pages 1–1, 2021.

[27] W. B. Frakes and R. Baeza-Yates. *Information retrieval: Data structures & algorithms*, volume 331. Prentice Hall Englewood Cliffs, NJ, 1992.

[28] J. Fu, P. Juyal, and A. Zajic. Thz channel characterization of chip-to-chip communication in desktop size metal enclosure. *IEEE Transactions on Antennas and Propagation*, 67(12):7550–7560, 2019.

[29] J. Fu, P. Juyal, and A. Zajic. Modeling of 300 ghz chip-to-chip wireless channels in metal enclosures. *IEEE Transactions on Wireless Communications*, 19(5):3214–3227, 2020.

[30] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer. Stealing keys from pcs using a radio: Cheap electromagnetic attacks on windowed exponentiation. In T. Güneysu and H. Handschuh, editors, *Cryptographic Hardware and Embedded Systems – CHES 2015*, pages 207–228, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[31] D. Genkin, L. Pachmanov, I. Pipman, E. Tromer, and Y. Yarom. Ecdsa key extraction from mobile devices via nonintrusive physical side channels. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1626–1638, New York, NY, USA, 2016. ACM.

[32] D. I. Gorman, M. R. Guthaus, and J. Renau. Architectural opportunities for novel dynamic emi shifting (demis). In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, pages 774–785, New York, NY, USA, 2017. ACM.

[33] Z. Hadjilambrou, S. Das, M. A. Antoniades, and Y. Sazeides. Leveraging cpu electromagnetic emanations for voltage noise characterization. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 573–585, Oct 2018.

[34] Y. Han, S. Etigowni, H. Liu, S. Zonouz, and A. Petropulu. Watch me, but don't touch me! contactless control flow monitoring via electromagnetic emanations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1095–1108, New York, NY, USA, 2017. ACM.

[35] D. J. Hand. Statistical concepts: A second course, fourth edition by richard g. lomax, debbie l. hahs-vaughn. *International Statistical Review*, 80(3):491–491, 2012.

[36] Z. He and R. B. Lee. How secure is your cache against side-channel attacks? In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, pages 341–353, New York, NY, USA, 2017. ACM.

[37] J. Hu, T. Duman, M. Erden, and A. Kavcic. Achievable information rates for channels with insertions, deletions, and intersymbol interference with i.i.d. inputs. *Communications, IEEE Transactions on*, 58(4):1102–1111, April 2010.

[38] E. Jorgensen, A. Kacmarcik, M. Prvulovic, and A. Zajic. Novel feature selection for non-destructive detection of hardware trojans using hyperspectral scanning. *J Hardware and Systems Security*, 6:32–46, 2022.

[39] S. Kim and A. Zajic. Characterization of 300-ghz wireless channel on a computer motherboard. *IEEE Transactions on Antennas and Propagation*, 64(12):5411–5423, 2016.

[40] S. Kim and A. Zajic. Statistical modeling and simulation of short-range device-to-device communication channels at sub-thz frequencies. *IEEE Transactions on Wireless Communications*, 15(9):6423–6433, 2016.

[41] A. Kirsch and E. Drinea. Directly lower bounding the information capacity for channels with i.i.d. deletions and duplications. *Information Theory, IEEE Transactions on*, 56(1):86–102, Jan 2010.

[42] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42Nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 469–480, New York, NY, USA, 2009. ACM.

[43] S. Li, K. Chen, J. H. Ahn, J. B. Brockman, and N. P. Jouppi. Cacti-p: Architecture-level modeling for sram-based structures with advanced leakage reduction techniques. In *Proceedings of the International Conference on Computer-Aided Design*, ICCAD '11, pages 694–701, Piscataway, NJ, USA, 2011. IEEE Press.

[44] C. Liu, A. Harris, M. Maas, M. Hicks, M. Tiwari, and E. Shi. Ghostrider: A hardware-software system for memory trace oblivious computation. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '15, pages 87–101, New York, NY, USA, 2015. ACM.

[45] Y. Liu, L. Wei, Z. Zhou, K. Zhang, W. Xu, and Q. Xu. On code execution tracking via power side-channel. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1019–1031, New York, NY, USA, 2016. ACM.

[46] D. McCann, E. Oswald, and C. Whitnall. Towards practical tools for side channel aware software engineering: Grey box' modelling for instruction leakages. In *Proceedings of the 26th USENIX Conference on Security Symposium*, SEC'17, pages 199–216, Berkeley, CA, USA, 2017. USENIX Association.

[47] H. Mercier, V. Tarokh, and F. Labeau. Bounds on the capacity of discrete memoryless channels corrupted by synchronization and substitution errors. *Information Theory, IEEE Transactions on*, 58(7):4306–4330, July 2012.

[48] J. K. Millen. Covert channel capacity. In *Security and Privacy, 1987 IEEE Symposium on*, pages 60–60, April 1987.

[49] K. Nayak, C. W. Fletcher, L. Ren, N. Chandran, S. V. Lokam, E. Shi, and V. Goyal. Hop: Hardware makes obfuscation practical. In *Proceedings of the Annual Network and Distributed System Security Symposium (NDSS)*, NDSS '17, 2017.

[50] A. Nazari, N. Sehatbakhsh, M. Alam, A. Zajic, and M. Prvulovic. Eddie: Em-based detection of deviations in program execution. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, pages 333–346, New York, NY, USA, 2017. ACM.

[51] L. Nguyen, C.-L. Cheng, M. Prvulovic, and A. Zajic. Creating a backscattering side channel to enable detection of dormant hardware trojans. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2019.

[52] L. N. Nguyen, B. B. Yilmaz, M. Prvulovic, and A. Zajic. A novel golden-chip-free clustering technique using backscattering side channel for hardware trojan detection. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 1–12, 2020.

[53] A. Patel, F. Afram, S. Chen, and K. Ghose. Marss: A full system simulator for multicore x86 cpus. In *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1050–1055, June 2011.

[54] M. Rahmati and T. Duman. Bounds on the capacity of random insertion and deletion-additive noise channels. *Information Theory, IEEE Transactions on*, 59(9):5534–5546, Sept 2013.

[55] A. Rane, C. Lin, and M. Tiwari. Raccoon: Closing digital side-channels through obfuscated execution. In *Proceedings of the 24th USENIX Conference on Security Symposium*, SEC'15, pages 431–446, Berkeley, CA, USA, 2015. USENIX Association.

[56] A. Rane, C. Lin, and M. Tiwari. Secure, precise, and fast floating-point operations on x86 processors. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, pages 71–86, Berkeley, CA, USA, 2016. USENIX Association.

[57] T. Rappaport, K. Remley, C. Gentle, A. Molisch, and A. Zajic. *Radio propagation measurements and channel modeling: best practices for millimeter-wave and sub-terahertz frequencies*. Cambridge University Press, 2022.

[58] J. Renau, B. Fraguela, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, S. Sarangi, P. Sack, K. Strauss, and P. Montesinos. SESC simulator, January 2005. http://sesc.sourceforge.net.

[59] R. Rutledge, S. Park, H. Khan, A. Orso, M. Prvulovic, and A. Zajic. Zero-overhead path prediction with progressive symbolic execution. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 234–245, 2019.

[60] N. Sehatbakhsh, A. Nazari, A. Zajic, and M. Prvulovic. Spectral profiling: Observer-effect-free profiling by monitoring em emanations. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–11, Oct 2016.

[61] N. Sehatbakhsh, A. Nazari, A. Zajic, and M. Prvulovic. Spectral profiling: Observer-effect-free profiling by monitoring em emanations. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–11, 2016.

[62] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[63] M. Taha and P. Schaumont. Key updating for leakage resiliency with application to aes modes of operation. *IEEE Transactions on Information Forensics and Security*, 10(3):519–528, March 2015.

[64] M. Taram, A. Venkat, and D. Tullsen. Mobilizing the micro-ops: Exploiting context sensitive decoding for security and energy efficiency. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 624–637, June 2018.

[65] Tektronix. tbs1000-digital-storage-oscilloscope. https://www.tek.com/oscilloscope/tbs1000-digital-storage-oscilloscope, 2019 (accessed Nov. 6, 2019).

[66] Terassic. De0-cv. http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=364, 2019 (accessed Nov. 6, 2019).

[67] E. M. Ugurlu, B. B. Yilmaz, A. Zajic, and M. Prvulovic. Pitem: Permutations-based instruction tracking via electromagnetic side-channel signal analysis. *IEEE Transactions on Computers*, pages 1–1, 2021.

[68] W. van Eck. Electromagnetic radiation from video display units: An eavesdropping risk? *Computers and Security*, 4(4):269 – 286, 1985.

[69] R. Venkataramanan, S. Tatikonda, and K. Ramchandran. Achievable rates for channels with deletions and insertions. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pages 346–350, July 2011.

[70] S. Verdú and S. Shamai. Variable-rate channel capacity. *IEEE Transactions on Information Theory*, 56(6):2651–2667, 2010.

[71] Z. Wang and R. Lee. Capacity estimation of non-synchronous covert channels. In *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*, pages 170–176, June 2005.

[72] J. Wichelmann, A. Moghimi, T. Eisenbarth, and B. Sunar. Microwalk: A framework for finding side channels in binaries. In *Proceedings of the 34th Annual Computer Security Applications Conference*, ACSAC '18, pages 161–173, New York, NY, USA, 2018. ACM.

[73] S. J. E. Wilton and N. P. Jouppi. Cacti: an enhanced cache access and cycle time model. *IEEE Journal of Solid-State Circuits*, 31(5):677–688, May 1996.

[74] M. Wu, S. Guo, P. Schaumont, and C. Wang. Eliminating timing side-channel leaks using program repair. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2018, pages 15–26, New York, NY, USA, 2018. ACM.

[75] B. Yilmaz, A. Zajic, and M. Prvulovic. Modelling jitter in wireless channel created by processor-memory activity. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018*, pages 2037–2041, 04 2018.

[76] B. B. Yilmaz, R. Callan, A. Zajic, and M. Prvulovic. Capacity of the em covert/side-channel created by the execution of instructions in a processor. *IEEE Transactions on Information Forensics and Security*, 13(3):605–620, 2018.

[77] B. B. Yilmaz, R. L. Callan, M. Prvulovic, and A. Zajić. Capacity of the em covert/side-channel created by the execution of instructions in a processor. *IEEE Transactions on Information Forensics and Security*, 13(3):605–620, 2017.

[78] B. B. Yilmaz, M. Prvulovic, and A. Zajic. Capacity of deliberate side channels created by software activities. In *Military Communications Conference (MILCOM), MILCOM 2018-2018 IEEE*. IEEE, 2018.

[79] B. B. Yilmaz, M. Prvulovic, and A. Zajic. Electromagnetic side channel information leakage created by execution of series of instructions in a computer processor. *IEEE Transactions on Information Forensics and Security*, 15:776–789, 2020.

[80] J. Yu, L. Hsiung, M. E. Hajj, and C. W. Fletcher. Data oblivious isa extensions for side channel-resistant and high performance computing. In *Proceedings of the Annual Network and Distributed System Security Symposium (NDSS)*, NDSS '19, 2019. `https://eprint.iacr.org/2018/808`.

[81] A. Zajić and M. Prvulovic. Experimental demonstration of electromagnetic information leakage from modern processor-memory systems. *IEEE Transactions on Electromagnetic Compatibility*, 56(4):885–893, Aug 2014.