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 [1, 7, 8, 11, 16, 21, 36]. This is especially a problem for cyber-physical systems (CPS) and Internet-of-Things (IoT) systems, which are often placed in publicly accessible locations and yet contain sensitive data, such as sensor data, login information for over-the-network management of the system and/or accessing back-end cloud infrastructure. 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 [19, 35, 39, 82]. 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, recent analog side channel attacks demonstrated by PIs Prvulovic and Zajic have shown that both attacks and mitigation are becoming increasingly dependent on microarchitectural behavior, making mitigation potentially fragile to future microarchitectural changes [1,2]. Unfortunately, early-design tools, such as cycle-accurate (microarchitecture-level) simulators [5, 12, 20, 46, 47, 58, 62, 74], do not model analog side channel signals, so these side channels can only be considered when they can be physically measured (post-fabrivation). 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 qualitative or entirely absent during software design, giving first-mover advantage to attackers, and resulting in mitigation via localized patches, which themselves are becoming increasingly microarchitecture-dependent.

Ideally,information leakage through analog side channels, and potential for "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 prototypes are fabricated [5, 12, 20, 46, 47, 58, 62, 74] If such efficient-yet-highly-accurate simulation would exist for analog side channels, computer architects could include analog side-channel leakage among their design considerations [3, 4, 54, 60, 70, 81], compilers could use simulation models to optimize for reduced leakage [38,48,59], software designers could detect and mitigate information leakage problems for security-sensitive applications [22, 73, 75], etc.

While there are some tools and metrics to quantify analog side-channel leakage [9, 18, 28, 51, 78], they are limited in several ways: *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. However, these metrics have inherent assumptions about the aspects of the signal the attacker may exploit, so they may not reveal *all* of the information the signal may contain. *Second*, most existing methods only model the system at the *architecture* (ISA) level, associating a (leakage) value to individual instructions based on the ISA, while ignoring how the microarchitecture (pipeline stages, stall cycles, etc.) affects the signal. As we have demonstrated in our seminal paper on analog side channel modeling [67] (these findings will be briefly be summarized in Section 4), this can lead to significant inaccuracy. *Third*, by neglecting the impact of micro-architecture, these methods implicitly assume that the entire hardware design is a *single source* of the combined signal. Such an assumption can lead to large inaccuracies for some side channels (e.g. EM), because contributions of different microarchitectural blocks (e.g., cache, register-file, etc.) may experience different propagation (e.g. attenuation, low-pass filtering, etc.) to where the overall signal is being measured.

The PIs first attempt to address these challenges and develop microarchitecture-level analog side-channel model, was called EMSim [67]. That proof-of-concept study has shown that efficient, yet highly accurate, estimation of analog side channel signals is feasible, allowing modeling of numerous entire-program runs, e.g. to assess and improve leakage from a particular piece of code on a set of hardware platforms. EMSim was also the first model 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 gate-level or physics-based models. However, while EMSim was good enough to demonstrate feasibility (and to serve as an excellent starting point for our proposed work, as will be discussed in Section 4), it used a number of simplifications that limit its accuracy and its ability to model more complex processors.

1.1. Summary of Proposed Research

To address these issues and allow analog side channels to be treated as first-class design considerations, we propose to develop methods that would allow microarchitecture-level simulation to generate highly accurate analog side channel signals, even for sophisticated processors (deep pipelines, superscalar, out-of-order). Specifically, the proposed research has four main thrusts:

1. New methods that allow microarchitecture-level simulation to produce highly accurate analog side channel signals, along with per-block and per-code-region breakdowns/attribution to help computer architects and software developers assess how microarchitecture and software changes would affect the tradeoff between performance, power, and side-channel leakage. Our overall approach to this will be to combine per-event signal snippets into the overall side channel signal using event times from cycle-accurate simulation.
2. New methods for efficiently performing circuit-level simulation of caches, functional units, and other building blocks of the processor, to provide per-event timelines for voltage, current, and possibly other metrics that are needed to construct signal snippets for various side channels, along with other useful information such as how much the values of the metrics depend on data values.
3. New methods that produce per-event signal snippets for various side channels, by 1) transforming the outputs of circuit-level simulation, 2) decomposing measured signals, or 3) synergistically combining both methods. This thrust will also explore new methods that estimate per-block scaling factors for contributions of these snippets to the overall analog side channel signals.
4. New methods that validate simulated signals against measurements, and to adjust scaling factors, signal snippets, and the simulator's timing parameters to calibrate snippets, weights, and timing parameters to a real processor.

2. Broader Impacts

We expect that our results will help include analog side channels among early-design considerations, which 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 that are otherwise not possible. The proposal also includes 1) developing an interactive demonstrator for the general public, to educate and raise awareness about several key cybersecurity concepts and issues, 2) visits and activities in local schools to improve K-12 education, and to encourage participation of women and minorities in STEM, 3) course and curriculum development activities at the undergraduate and graduate level, and 4) activities to broaden participation in computing (as discussed in our BPC plan).

3. Related Work

There is a large body of work focused on preventing specific side-channel attacks, e.g., [1, 2, 7, 28, 40, 42, 49, 55, 69], either by removing the tie between sensitive information and the side-channel signal, or by trying to make the signal more difficult to measure, with less focus on the fundamental relationships between hardware, software, and side-channel signals.

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) [28] 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.

Signal Available to Attacker (SAVAT) [18] 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 the ISA level, ignoring the underlying relation of each instruction to the hardware or other instructions in the sequence. Extensions of SAVAT that include modeling instruction in a simple pipeline have also been proposed [77, 80].

McCann *et al.* [51] proposed a modeling technique capable of producing an instruction-level leakage metric for power and EM side-channel signals on ARM M0/M4 cores. To estimate the leakage for individual instructions, that method only requires knowledge about different ISA-level characteristics of the system, such as data-dependent effects of neighboring instructions in a sequence, register effects, bit-flips, etc. Similarly to SAVAT, the effects of micro-architecture events such as (cache miss, branch miss-prediction, etc.) are largely ignored.

Barenghi and Pelosi [9] proposed a method that calculates leakage for individual instructions by measuring power consumption between two consecutive cycles and employs the Pearson correlation coefficient between the two measurements. To calculate the leakage, in addition to leveraging ISA-level information, a pipeline model was also used. However, the framework did not consider pipeline stalls or any other micro-architecture events, and 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 [52], and to calculate 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); 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 activity often inserts activity into the timeline of the spy program's execution. While [76, 77, 79] derive bounds for information leakage from side-channels with insertions, deletions, and errors, but fall short of considering how executed instructions affect each other through their interaction with the microarchitecture, and that can significantly impact the signal. Side-channel information capacity created by execution of a sequence of instructions was also derived by PIs Zajic and Prvulovic and their students [80], using a Markov Source model to account for dependencies among instructions, treating a processor as a transmitter with multiple "antennas", where each antenna corresponds to a pipeline stage. While these capacity-based approaches can represent the worst-case severity of side channel leakage, none of them provides enough detail to identify specific vulnerabilities and/or attribute leakage to specific hardware blocks or program code.

Another body of work related to this proposal are the cycle-accurate models/tools to simulate power and/or microarchitecture [5, 12, 20, 46, 47, 58, 62, 74]. 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 [67], 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.

The relationship between executed instructions, microarhitecture, and the resulting side-channel signals is also explored by work on leveraging EM signals for program profiling, tracking, and analysis [17,30,31,38, 64,65,71]. Spectral Profiling methods [17,64,65,71] track program execution at loop [65], basic-block [17, 64], or individual-instruction [71] granularity. EMPROF [30] profiles cache misses and PRIMER [31] profiles interrupts, 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 identify LLC miss events and to 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). All these methods are complementary to this proposal and combined with proposed modeling tool can enhance leakage estimation, compiler development, etc.

4. Proposed Research

Figure 1 shows our overall proposed approach. Most parts of this approach are within the orange-background "Training" oval in Figure 1, which indicates that they are only needed for training, before the cycle-accurate simulation that constructs the overall simulated side channel signal begins. We propose to implement this overall signal construction efficiently by first preparing signal snippets $S_b(t)$ for each type of event in each of the processor's microarchitectural building blocks, along with per-building-block weights α_b that specify how to scale the contribution of each block to the overall side channel signal. Given these snippets and weights, we can obtain timestamps of individual events through cycle-accurate simulation, and then construct the overall signal by adding the appropriate snippet to the signal at each point in the signal's timeline where the corresponding event occurs.

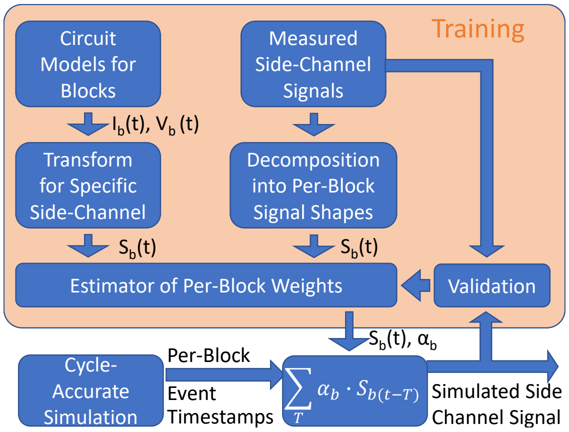

Training
Circuit Models for Blocks
Measured Side-Channel Signals
$I_b(t)$, $V_b(t)$
Transform for Specific Side-Channel
Decomposition into Per-Block Signal Shapes
$S_b(t)$
$S_b(t)$
Estimator of Per-Block Weights
Validation
$S_b(t)$, α_b
Cycle-Accurate Simulation
Per-Block Event Timestamps
$\sum_T \alpha_b \cdot S_{b(t-T)}$
Simulated Side Channel Signal

Figure 1: Overview of the proposed approach.

Our training constructs the signal snippets and their estimated weights through 1) circuit-level simulation of individual events in individual building blocks, 2) through decomposition of measured signals from an actual processor, or 3) through a synergistic combination of these methods. Finally, simulation-generated and measured signals can be compared for validation and, if calibration against a real processor is desired, discrepancies found during validation can be used to improve circuit-level models, signal snippets, and per-block weights such that future simulated results would match measured signals better.

To implement this approach, our proposed research consists of four primary research thrusts:

1. Discovering how to efficiently generate detailed event timestaps and use them to combine signal snippets into the overall side channel signal, along with per-block and per-code-region breakdowns and attribution that can help computer architects and software developers assess how microarchitectural and software changes affect the tradeoff between performance, power, and side channel leakage.
2. Discovering how to efficiently perform circuit-level simulation of caches, functional units, and other building blocks of the processor, to provide per-event timelines for voltage, current, and possibly other metrics that are needed to construct signal snippets for various side channels. along with other useful information such as how much the values of the metrics depend on data values.
3. Discovering new methods that produce per-event signal snippets for varous side channels, by 1) transforming the outputs of circuit-level simulation, 2) decomposing measured signals, or 3) synergistically combining both methods, and also design of new methods for estimating per-block weights.
4. Discovering new methods to validate simulated signals against measurements, and to adjust weights, signal snippets, and the simulator's timing parameters to calibrate snippets, weights, and timing parameters to a real processor.

In the rest of this section, we provide more detail on each of these research thrusts, along with preliminary results that demonstrate the feasibility of the proposed approaches and provide an excellent starting point for the proposed research in each thrust.

4.1. Thrust 1: Construction of Side-Channel Signals Through Cycle-Accurate Simulation

With the scaled per-event signal snippets from training, we can construct the side channel signal that corresponds to longer-term program execution. This signal construction uses a microarchitecture-level cycle-accurate simulation to determine the times at which events occur in each of the processor's block-level parts (ALU, decoder, reservation stations, cache, etc.), and these per-block event timestamps are then used to combine per-event signal snippets into an overall simulated signal.

One of the key research problems for this thrust will be to explore the tradeoff between the accuracy of the generated signal and the level of detail in the cycle-accurate simulation. For example, in an L1 data cache we may lump all events (hit, lookup with miss, fill following a miss, etc.) into a single "cache action" category that is represented by a single snippet. This reduction in the level of detail would still allow the generated signal to capture the impact of the overall timeline of cache activity, e.g. having more prominent signals when executing code code regions with a lot of cache accesses, having low-signal-activity periods when the cache is not used (due to a processor stall between a miss and the corresponding fill), etc. However, these events would need to be differentiated to account for differences between reads and writes, to account for a lookup-miss having to initiate an access in the next-level cache while hits do not, etc.

A similar tradeoff will be explored with respect to the impact of data values. We will start with data-value-agnostic simulation, which can still model some attacks that exploit which code is executed, such as secret-dependent control flow. We will then add ability to obtain data-value statistics in cycle-accurate simulation, and leverage the signal envelopes produced by our symbolic circuit-level simulation, to identify events (and blocks) whose signal snippets are affected the most by data values. For such blocks/events, we will identify the nature of data-value dependency and construct data-value-dependent snippets, transformations, and weights to account for data-value dependencies in the signal. We will do this selectively (not for all events) because there is a tradeoff between accuracy and training/simulation time, so a given amoung of accuracy improvement will come at a lower cost (training/simulation time) when only the most data-value-dependent signal snippets are refined in this way.

4.1.1. Preliminary Results for Thrust 1

Our preliminary results for this thrust are based on EMSim [67], a first-of-its-kind cycle-accurate simulator that generates practically relevant EM side channel signals for program execution on a canonical 5-stage pipelined processor. This simulator, implemented by PI Prvulovic, Co-PI Zajic, and several PhD students they have co-advised, demonstrates the feasibility of using event timings from microarchitectural simulation, together with signals snippets, to generate overall side channel signals.

In EMSim, each pipeline stage of the processor is modeled as a single block, and each instruction is treated as a separate event for each stage. The signal snippets all use the same shape, which is a mathematical function (rather than a measured or circuit-simulation-derived shape that will be used in the proposed approach). The results we show here use an exponentially-decaying oscillation as the snippet shape, which tends to match measurements well. Differences among types of events events are represented in EMSim only by having each event type have a separate scaling factor for this common snippet shape, and these scaling factors are estimated using linear regression from signals that are measured on an FPGA-based imlpementation of this processor. Note that, although a more detailed RTL model of this processor was available, EMSim only uses the microarchitectural model in its simulation; the RTL was only used to implement the processor that was used to measure signals for finding scaling factors (training) and for evaluation of EMSim's accuracy.

EMSim combines scaled snippets into the overall simulated side channel signal (for program code that was not used to estimate the scaling factors). For this, it uses a microarchitecture-level simulator of the processor's pipeline. In each cycle, for each stage of the processor, this simulator identifies which instruction is in that stage in that cycle, applies the corresponding scaling factor to the decaying-oscillation shape, and adds the scaled "snippet" to the overall signal at that point in time. The resulting signal for one such simulation is shown, along with the measured signal for the same program run, in Figure 2. We observe that, although the signals have some differences, most of the prominent features from the measured signal are also present in the simulator-generated signal.

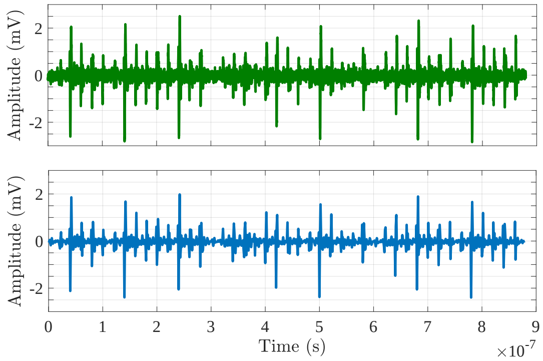


Figure 2: Measured (top) and EMSim-generated (bottom) EM signal.

Our sensitivity analysis for EMSim [67] has demonstrated the importance of modeling timing and microarchitectural behaviors, i.e. the need for cycle-accurate simulation of the microarchitecture rather than a simpler architecture-level emulation of the instruction stream.

To illustrate the need to account for microarchitectural events, Figure3 shows how a pipeline stall affects the signal. For illustration purposes, we increased the latency of a MUL instruction to eight cycles, and show a situation where the next instruction uses the result of a MUL. In the top part in Figure 3, where the stall is correctly modeled in EMSim, the simulated signal has good agreement with the measured signal. However, in the bottom part in Figure 3 EMSim was modified to ignore this read-after-write dependence, resulting in not modeling the stall and, consequently, not having a good match to the measured signal. Similar effects are observed when the simulation ignores other microarchitectural events and their timing, e.g. cache misses, branch mispredictions, etc.

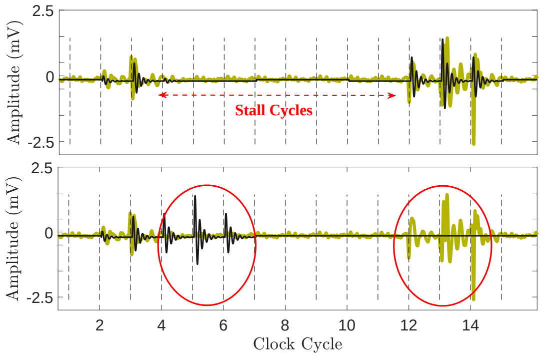
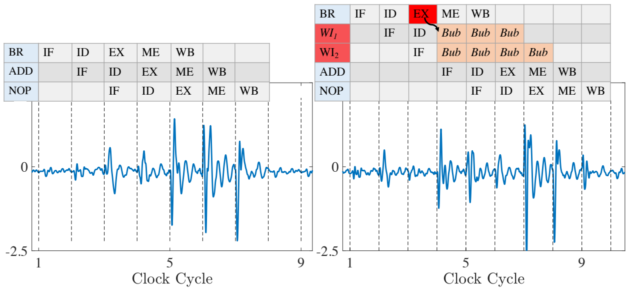


Figure 3: Comparison of measured (green) and simulated (black) EM signals with (top) and without (bottom) modeling pipeline stalls.

In addition to its impact on *timing*, we observed that a branch misprediction also has noticeable impact on the *shape* of the side-channel signal. In the 5-stage processor that was used in EMSim, a branch misprediction causes a flush of the first two pipeline stages, which adds two cycles to the execution and *also* changes which instructions are in which stage of the pipeline for the next few cycles. This is shown in Figure 4, along with how the change impacts the measured EM signal. This effect would be even more pronounced for a more sophisticated processor, with more instructions flushed and more change in overlap among activity in different blocks.



BR	IF	ID	EX	ME	WB		
ADD		IF	ID	EX	ME	WB	
NOP			IF	ID	EX	ME	WB

BR	IF	ID	EX	ME	WB					
WI_1		IF	ID	*Bub*	*Bub*	*Bub*				
WI_2			IF	*Bub*	*Bub*	*Bub*	*Bub*			
ADD				IF	ID	EX	ME	WB		
NOP					IF	ID	EX	ME	WB	

Figure 4: Signal without (left) and with (right) a branch misprediction.

Overall, even though EMSim had used a simple mathematical function as the snippet shape for all events, the EM side channel signal it constructs matches the measured EM side channel signal relatively well. EMSim was evaluated using synthetic benchmarks that were specifically designed to create various combinations of instructions in different pipeline stages. Using normalized cross-correlation as the accuracy metric, we found that EMSim achieves 94.1% accuracy. While this is sufficient for many purposes, e.g. assessing vulnerability to simple power attacks (SPA) and simple EM attacks (SEMA) that extract secrets by differentiating between executing different parts of the program code (e.g. a secret-dependent if-then-else) and/or differential attacks (DPA, DEMA) that depend on large differences in values that are operated on, more accuracy is likely to be needed for simulated signals to be used to assess vulnerability to more sophisticated attacks that differentiate between data values that are similar to each other. We expect that the proposed use of per-event signal snippets from circuit-level simulation, with additional data-value-dependent considerations, will enable such uses of our simulation-generated signals. Additionally, our proposed work will enable signals to be generated not only for the (simplest) pipelined processors, but also to the much mode complex processors (deep pipelines, superscalar, out-of-order, etc.), by extending cycle-accurate simulators such as gem5 [12] and ESESC [6].

4.2. Thrust 2: Circuit-Level Simulation to get Per-Event Current/Voltage Timelines

Even when a full circuit-level model for an entire processor is available, circuit-level simulation is not efficient enough to go through numerous program runs, with billions of processor cycles in each run, that would be needed to provide computer architects and software developers with actionable insights about the impact of microarchitectural and/or software changes on the tradeoff between performance, energy, and side-channel vulnerability. For example, PI Kumar's symbolic execution framework [25] currently uses detailed gate-level models of the microprocessor hardware, resulting in high-fidelity reproduction of metrics that are needed to predict the microprocessor's EM and power side-channel signals. However, this comes at a high computational cost when modeling entire-program execution – an FFT implementation for a 16-bit MSP430 DSP takes over 3 hours, and this time would become several times longer for more complex processors and orders of magnitude longer for realistic runs of larger applications.

Instead, our proposed approach will raise the abstraction level of the whole-program simulation to the microarchitecture level (i.e. cycle-accurate simulation); it will use circuit-level simulation only during training, separately for each block of the processor, e.g. decoder, adder, multiplier, instruction scheduler, cache, etc. to produce, for each kind of activity in a block, how currents and voltages in that block change over time (i.e. $I_b(t)$ and $V_b(t)$ in Figure 1) during that activity. Each of these circuit-level simulations will only model la few cycles, as it only needs to account for activity that corresponds to a single event. For example, for a data cache the events would be a lookup with a read-hit, lookup for a read-miss, fill on a read-miss, and read completion after a miss and fill, and the corresponding events for a write.

To improve accuracy of our overall signals, we will leverage the ability of the PI Kumar's existing circuit-level simulator to evaluate circuit activity using symbolic data. For a given time period, the current (and other properties) can be modeled not as a single curve, but rather as a time-changing distribution or as an "envelope" that shows the possible range of values at each point in in time. This presents us with a number of research challenges: how to express these distributions in the way that aids accuracy of the overall signals produced by cycle-accurate simulation, how to prevent path explosion when modeling blocks that may have significant changes in activity depending on the data (e.g. a symbolic address causing a hit or a miss in a cache), how to express aggregate data-dependent quantities as simplified symbolic expressions, to allow microarchitecture-level simulation to account for significant changes in aggregate activity within a block without resorting to modeling the block at a lower lever of abstraction, etc.

To help designers of highly secure microprocessors, i.e. processors which should have minimal information leakage via side-channels for anyprogram, we will also investigate a complementary approach, Property Driven Automatic Transformation (PDAT) [15], which analyzes net switching activity within a circuit for any inputs and performs model checking that allows processor designers to choose (or redesign) specific building blocks in a way that minimizes data-dependent changes in activity that would leak information. In essence, rather than accounting for data-dependent activity and expressing it in the signals provided to cycle-accurate simulation, this approach changes the hardware of the block to minimize such data-dependent activity, likely eliminating the need to account for data-dependent activity for that block.

Finally, while we expect that overall current and voltage will allow our side-channel-specific transformations to produce highly accurate signal snippets, more detailed outputs, e.g. separate $I_b(t)$ and $V_b(t)$ for different kinds of circuit elements, e.g. transistors, interconnect wires, power delivery wires, etc. may be needed to further improve accuracy, and we will investigate when such refinements are needed and how to implement them efficiently. For example, transistors that have the same geometry may have similar impact on side-channel-relevant properties (current, voltage, etc.), and thus may be represented in aggregate, but activity of larger transistors may need to be aggregated separately from smaller ones, to allow different scaling factors to be used for their activity when generating side channel snippets and/or accounting for data-dependent changes in the side channel signal.

4.2.1. Preliminary Results for Thrust 2

Co-PI Kumar's prior efforts to enable software developer to analyze their software for analog side channels have resulted in a framework which performs symbolic execution of a program on a detailed (gate-level) model of a microprocessor [25]. This framework was originally developed to automate design of 'bespoke microprocessors' — microprocessors designed to execute a single application binary. The flowchart for this use case is in shown in Figure 5, and the symbolic execution of the program on the hardware occurs in the 'Gate Activity Analysis' stage.

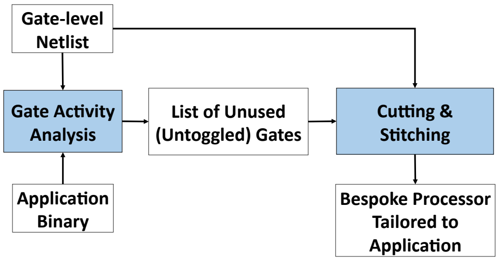


Figure 5: Hardware-software co-analysis used to generate bespoke microprocessors.

This framework accounts for how activity within the processor is affected by program inputs (e.g. sensor inputs in an IoT system), both for data-dependent (data values) and instruction-dependent (which instructions end up being executed) activity within the circuitry of the processor. This symbolic execution framework can identify whether any possible execution of a program leads to significant changes in gate-level activity and, consequently, side-channel leakage, allowing programmers to to add or remove often-expensive software mitigation techniques depending on which mitigations are actually needed. This framework has shown applicability for security applications [26], including taint-tracking for information flow security to locate information policy violations, even violations which result from covert timing side-channels.

This prior work provides us with an excellent starting point for the proposed symbolic circuit-level simulation. However, as pointed out earlier, circuit-level simulation of an entire processor throughout the execution of a program is exceedingly slow compared to state-of-the-art microarchitecture-level (cycle-accurate) simulation. Our proposed approach will synergistically combine the advantages of symbolic circuit-level simulation and microarchitecture-level simulation, using circuit-level simulation only during training to generate accurate per-event signal snippets, and using the (orders of magnitude faster) microarchitecture-level simulation to model the entire microprocessor for an entire program execution and generate a side channel signal by combining pre-made snippets.

Additionally, while gate-level switching activity can be a good proxy for side channel information leakage, PI Prvulovic and Co-PI Zajic's past work [67] has shown that the mapping from gate-level activity to side channel signals tends to be proportional only for gates within a relatively homogeneous block (e.g. a cache, a multiplier, etc.), but the scaling factors for different blocks can differ significantly. Thus our proposed approach is to account for these unequal transformations on a block-by-block basis, accounting for the specifics of how each side channel (EM, power, etc.) is affected by gate-level events, to produce *side channel signals* (rather than proxy metrics) that can be compared to and calibrated against measured signals for various side channels.

4.3. Thrust 3: Per-Event Side Channel Snippet Construction and Scaling

This research thrust will explore 1) how to transform the results of the circuit-level simulation into a signal snippet that represents the shape of the side channel signal that is contributed to the overall side channel signal by an occurrence of that event, 2) how to estimate the magnitude and (at least to some extent) the shape of these side channel signal snippets by decomposing measured signals, and 3) how to synergistically combine the side channel snippets obtained from circuit-level simulation and from signal decomposition.

Specifically, after the circuit-level simulator produces $I_b(t)$ and $V_b(t)$ signals for a specific event that happens in a specific block, these need to be transformed into a side-channel signal snippet $S_b(t)$. This step accounts for how current and voltage (and their changes over time) impact that side-channel. For example, for the power side channel the transformed signal would be the product of the current and voltage signals, for the EM side channel the transformed signals would be using Maxewell' equations to produce transformed signals that are proportional to the first derivative of the current and voltage signals, etc.

Snippets $S_b(t)$ can also be obtained from measurements of side channel signals on a real processor. This can be used when a circuit-level model of some part of the processor is not available. Measurements can also be used to correct (or calibrate) the scale of snippets from circuit-level simulations that have systematic biases, e.g. due to imprecisions in transistor models or lack of accounting for some electrical, electromagnetic, or quantum effects during circuit-level simulations. Our initial approach to get per-event snippets from measured data will be to collect side channel signals as the processor executes various microbenchmarks that are designed to create specific events in specific parts of the processor, allowing the measured signals to be decomposed (e.g. using SVD) into signal snippets that correspond to contributions of individual events .

Finally, we note that the two methods of obtaining per-event signal snippets are highly complementary. Specifically, because of measurements noise and bandwidth limitations, measurement-derived snippets tend to be very accurate in terms of the overall magnitude of the event's contribution to the signal [18], and less accurate in terms of the shape of that contribution (snippet) over the event's "life span". Conversely, circuit-level models are likely to produce very accurate shapes (if N identical-in-size transistors toggle, the current draw is proportional to N), with a lot of detail and metadata (e.g. the distribution or the "envelope" that shows how the signal snippet can change due to different data values). However, the magnitude of these shapes may not be scaled correctly due to imperfect modeling of device physics, device parameters, or the circuit. This means that, when both circuit-level models and empirical measurements are available, the model-generated signal snippets can be scaled to match the average magnitude of measurement-derived signals. Our initial approach for this will be to estimate the per-block scaling factors (α_b in Figure 1) using linear regression, and we will refine this as needed to improve accuracy further (e.g. using functions other than simple multiplication with a constant scaling factor).

The second reason for using scaling factors (or other "scaling" functions) is to account for side-channel specific differences in attenuation of signals from different blocks. Specifically, for the EM side channel, signal contributions from different blocks may experience different propagation loss, depending on the materials each signal propagates through and/or the position of the antenna or probe. These differences in propagation can also be accounted for using the per-block scaling factors, which can be determined using either measured signals or simulation-based propagation models.

Finally, we will account for data-dependent activity by considering how the signal is affected by values both within each microarchitectural block and between the blocks. Fundamentally, signals in most side channels (e.g. EM and power) are created during *bit-flips* at the transistor-level [72, 82], i.e. by changes in values at outputs of individual gates, flip-flops, and entire building blocks. Values that are outputs of individual blocks (e.g. the result produced by an ALU or the value read from a data cache) are typically already tracked during cycle-accurate simulation, so we can account for how changes in these values affect the side channel by adjusting the scaling factors according to data-dependent metrics such as the number of bit-flips. Values within a building block (e.g. values of carry bits within an adder or values produced by individual stages within a multiplier), however, are typically not tracked by a cycle-accurate simulator, and doing so precisely would be very costly in terms of simulation time. However, we note that values at outputs of building blocks tend to drive long interconnect wiring, whereas the wiring that carries values within a block tends to be local. This implies that for most analog side channels, e.g. power, EM, etc., values at outputs of building blocks will tend to have a significantly stronger impact of the overall side channel signal, and this is also supported by our preliminary results. Therefore, bit-flips within a building blocks are likely to have an aggregate effect, i.e. the side channel signal is affected mostly when data values at inputs of the block produce large differences in the amount of internal bit-toggling. Rather than add simulation of such intra-block activity to the cycle-accurate simulator, we plan to account for this in the aggregate, by leveraging the ability of our circuit-level simulators to consider symbolic outputs and produce the envelope that describes how much the signal can vary depending on input values and, if such signal variation is significant enough, by generating symbolic expressions for how the aggregate amounts of internal switching depends on the input values of the block.

4.3.1. Preliminary Results for Thrust 3

Our preliminary results for this thrust are also based on our work on EMSim [67], where we considered two main contributors to the EM side-channel signal emanated by each stage (in EMSim, an entire pipeline stage of the simple 5-stage processor is treated as a building block). The first group of contributors, which we call *instruction-dependent* contributors, consists of per-use signal contribution of each micro-architectural block, e.g. whether instruction fetch is happening or not (due to a stall) in a particular cycle, whether or not the register-file is being written in the last stage of the pipeline in a particular cycle, etc.

The second group of contributors is *data-dependent* activity, which is created by bit-flips on the long wires, such as address and data lines going to/from the cache, and in the processor's registers. In reality, operand and data values also change the signal in other parts of the processor, e.g. on wiring between gates within an ALU, but EMSim only accounted for values that are already available during cycle-accurate simulation, whereas accounting for other bit-flips would have required RTL-level or circuit-level modeling.

For both types of contributors, we assumed that activity creates a signal that has the same overall shape (an exponentially-decaying sinusoid), because we did not use circuit-level models to get a better estimate of the shape. However, the magnitude of that shape is estimated separately, using measured data, for each contributor. For instruction-dependent contributors, i.e. effect on the signal of a specific type of instruction utilizing a specific pipeline stage, this estimation relies on executing microbenchmarks in which a single instruction, surrounded by NOP instructions, is executed, with all of its operands and data values set to zero, to minimize the effect of data-dependent contributors and create a *baseline* level of activity for each type of instruction in each stage of the processor.

Figure 6 shows the measured and EMSim-generated EM signals when an ADD instruction progresses through the pipeline while all the other instructions are NOPs. The top part of Figure 6 corresponds to EMSim simulation where the contribution of each stage is scaled separately, using the per-stage scaling factors obtained from measured data. The bottom part of Figure 6 shows the simulated signal when all stages are treated as one source, i.e. only one (average) scaling factor is estimated from measurements and then used for activity in any stage. As we can see in Figure 6, failing to model each stage individually (as used in previous work [51]) can lead to significant inaccuracy.

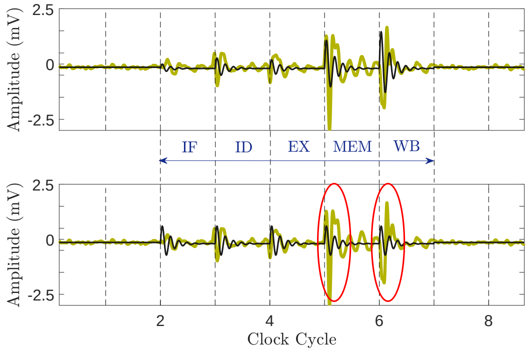


Figure 6: Measured (green) and simulated (black) signal when each pipeline stage is treated a separate source (top), and when considering the entire processor as a single source (bottom). Red ellipses point out the largest differences between simulated and measured signals.

For data-dependent contributors, the pipeline simulator used in EMSim identifies the number of bit-flips on address and data lines and in the register being written. Intuitively, more bit-flips cause more current flow and thus larger amplitude of EM emanations. To combine the effects of such data-dependent activity with the baseline (instruction-dependent) activity, we define the *activity-factor*, *beta*, as a ***scaling factor*** for each baseline activity A. To find this β, we assume all bit-flips within the stage contribute equally, i.e. that each bit-flip contributes equally to the signal amplitude. Because the baseline signal amplitude already includes some bit-flips (instruction addresses and instructions themselves are non-zero when accessing the instruction cache, etc.), we calculate β as $\beta = 1 + \frac{(flips_{new} - flips_{base})}{flips_{total}}$, where $flips_{new}$ is the total number of flips for the current instruction, $flips_{base}$ is the total number of flips when the previous instruction is a NOP, and $flips_{total}$ is the maximum possible number of flips for the current instruction. The overall scaling factor for the (decaying sinusoid) snippet is then $A' = \beta \times A$.

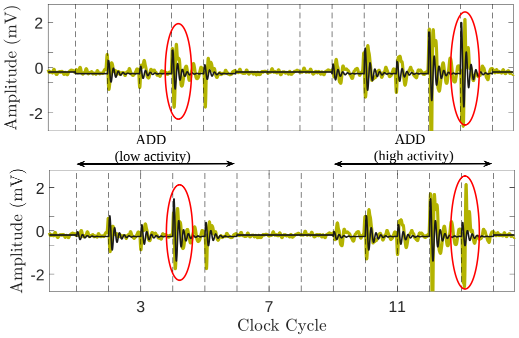


Figure 7: Measured (green) and simulated (black) signal when activity factor is modeled using a linear regression model (top) and when an *average* activity factor is used (bottom).

To illustrate why these data-dependent considerations also need to be considered separately for each block, Figure 7 shows the measured and EMSim-generated signals, with the top part showing simulated signals that use per-stage activity factors in each cycle as described above. In the bottom part, the simulation always uses one (average) activity factor. We observe that accounting for data-dependent activity in at lest some parts of the processor is needed to achieve good accuracy of the simulated signal. Further analysis of measured data has also shown that not all bit-flips have the same effect on the signal. Specifically, we found that 1) bit-flips in the output of the ALU and the cache have the most significant impacts on the signal, which can be expected in this FPGA-based design because these outputs were the driving the longest total amount of interconnect to reach all of their fanout targets, and 2) bit-flips in about 64% of block output bits had a negligible contribution to overall accuracy (as determined through stepwise regression [41]). This supports the approach described in this proposal: using measured data to estimate the contribution of each block (and how much of it is data-dependent), and then selectively add accounting for data-dependent activity only for blocks where such accounting can result in a meaningful improvement in accuracy. Also, our preliminary findings are encouraging because the largest data-dependent contributions were outputs of the largest units (ALU and cache), which implies that the most impactful bit-flips in larger processors are likely to occur at outputs of microarchitectural blocks – values that are already modeled by cycle-accurate simulators – while the values on wires within an individual block (connecting much smaller sub-blocks) are likely to have less of an effect. However, in our proposed work we plan to use aggregate accounting for toggling within blocks through symbolic circuit-level simulation to obtain not only the "baseline" signal snippets for each block, but also the "envelopes" and input-dependent expressions for how the shape of the signal snippet would be affected by the values at the block's inputs and, for register files, caches, etc., by changes to values stored within those blocks.

4.4. Validation/Calibration Against Measured Signals

In this thrust we will use measured side channel signals collected from real processors to evaluate the accuracy of snippet generation, transformation, and overall simulated signals produced by the methods that will be devised. The results of this evaluation will help us identify likely sources of error and refine our approach to create efficient simulators that produce highly accurate side channel signals that are calibrated against specific real processors.

The insights from this evaluation, refinement, and calibration efforts will then be used to design systematic mechanisms that can automate this iterative refinement/calibration, resulting in mechanisms that selectively refine the circuit-level models, accounting for data-dependent behaviors, scaling factors, and other aspects of the training process. The goal here will be to achieve a desired level of accuracy while minimizing the impact of these refinements on simulation time, and to avoid costly refinements (e.g. development of more detailed circuit-level models) that do not lead to commensurate increases in signal accuracy.

4.4.1. Preliminary Results for Thrust 4

PIs Prvulovic and Zajic have extensive experience in side-channel measurements for EM, power, and several other analog side channels (acoustic, optical, temperature, ...), comparing these signals to each other, and, more recently, comparing synthesized or modeled signals to measured ones [67] and using the results of these measurements to refine the models.

PI Kumar in experienced in digital design, chip design, tape-out, and testing, and runs an undergraduate course in which students design processors that are then are manufactured by TSMC and tested on the UIUC campus. His group has also participated in Intel's Chip Design Challenge, manufacturing RISC-V microprocessors in Intel's 16 nm technology. This gives us a sizeable collection of microprocessor artifacts, with both large and small microarchitectural differences, which can be subjected to side-channel measurement and also modeled at various levels of abstraction – high-level microarchitecture, RTL, gate-level, GDS/geometry level, and physical chips. This will allow us to evaluate the predictive accuracy of the simulation-generated signals with respect to microarchitectural changes, and also to calibrate based on numerous real-world designs and implementations.

5. Proposed Outreach and Education Activities

The proposed research 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, as well as the activities described in our Broadening Participation in Computing (BPC) plan.

- We will continue to 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 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. PIs Zajic and Prvulovic have long history of advising undergraduate students (26 undergrads) through Opportunity Research Scholar (ORS) program at Georgia Tech and individual mentorship, while PI Kumar has advised several undergraduate researchers who have gone on to join graduate studies at UIUC and elsewhere, and has advised undergraduate students who have recently published in premiere computer architecture conferences, including ISCA, HPCA (best paper nomination), and DAC. The interdisciplinary nature of the project will help us engage a large number of undergraduate students in research, and will provide both graduate and undergraduate student with a unique interdiciplinary skill-set.
- 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 side-channel signals. This initiative will be pursued through Georgia Tech's established outreach programs for high schools in the greater metro Atlanta area [27] and at events at UIUC. 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. Similarly, Dr. Kumar has assisted in the organization of events such as HackIllinois and Engineering Open House (EOH) that see enthusiastic participation from the local community. He will have a booth on “processors that leak information” in the next three EOHs to get high school students excited about intersection of computer architecture and security.

6. Integrated Research, Education, and Outreach Plan

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

Year	**Research**	**Education and Outreach**
1	Develop methods that allow microarchitecture-level simulation to produce highly accurate analog side channel signals; Develop methods for efficiently performing circuit-level simulation of caches, functional units, and other building blocks of the processor, to provide per-event timelines for voltage and current; Develop methods that produce per-event signal snippets for various side channels by transforming the outputs of circuit-level simulation.	Develop initial modules for graduate courses, visits to local schools, undergraduate teams work on lab measurement setups
2	Refine methods to allow for per-block and per-code-region breakdowns and attribution to help computer architects and software developers assess the impacts of microarchitectural and software changes; Refine circuit-models to include other metrics that are needed to construct signal snippets for various side channels, along with other useful information such as how much the values of the metrics depend on data values; Develop methods that decompose measured signals.	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	Develop tools to evaluate the tradeoff between performance, power, and side channel leakage; Integrate circuit models into overall architecture side channel simulator; Develop methods that estimate per-block scaling factors for adding these snippets to the overall analog side channel signals; Validate overall side channel simulations against measured side channel signals.	Refine course modules; develop K-12 and public demonstrators for sensory experiences of side-channel concerns (e.g., use software defined radio setups and simulation to demonstrate attacks and defenses); fully integrate undergraduates in research

The PIs plan to conduct the proposed research, education, and outreach activities in a highly integrated manner, as the project 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. To keep the project integrated, we will regularly communicate among the team members, hold bi-weekly project-wide virtual meetings, and meet in person at least once per year (more details are in our Collaboration Plan).

7. Results from Prior NSF Support

Dr. Prvulovic and Dr. Zajic were PIs/Co-PIs on the following NSF-funded projects in the last five years:

- Grant CCF-1563991 SHF *Spectral Profiling: Understanding Software Performance without Code Instrumentation* (\$850*K*, July 2016- June 2022).
  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 [17,64,66,71].
  Broader Impacts: 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* (\$200*K*, August 2017–July 2019).
  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 [43,56,57].
  Broader Impacts: 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 the following NSF-funded project:

- Grant ECCS-1651273 NSF CAREER *Modeling and Measurements for THz Wireless Chip-to-Chip Communications Propagation* (2017-2023, \$500*K*)
  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 [24, 33, 44] and develops channel models [34, 45] 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 Impacts: 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 [61].

Dr. Kumar has served as PI on the following NSF-funded project in the last five years:

- Grant CCF-2006763 SHF Printed Computer Systems (\$500K, October 2020- September 2023).
  Intellectual Merit: The project has already resulted in first sub-cent microprocessor citebleier22, first plastic encryption engines [13], first work on printed microprocessors [14], and first work on printed classifiers [53].
  Broader Impacts: This project has opened up new possibilities for ultra-low cost and conformal active electronics, enabling applications such as wearable patches [32], garments [63], packaging [37] for fast moving consumer goods (FMCG), low-end healthcare products such as smart bandages [29], and disposable sensors for food [23], pharmaceuticals [10], agriculture and forestry [68], and environment [50] — that have not seen much penetration of computing.