2018 51st Annual IEEE/ACM International Symposium on Microarchitecture

EMPROF: Memory Profiling via EM-Emanation in IoT and Hand-Held Devices

Moumita Dey*, Alireza Nazari†, Alenka Zajic, Milos Prvulovic
Georgia Institute of Technology, Atlanta, USA
Email: mdey@gatech.edu, anazari@gatech.edu, alenka.zajic@ece.gatech.edu, milos@cc.gatech.edu

***Abstract*— This paper presents EMPROF, a new method for profiling the performance impact of the memory subsystem without any support on, or interference with, the profiled system. Rather than rely on hardware support and/or software instrumentation on the profiled system, EMPROF analyzes the system's EM emanations to identify processor stalls that are associated with last-level cache (*LLC*) misses. This enables EMPROF to accurately pinpoint LLC misses in the execution timeline and to measure the cost (stall time) of each miss. Since EMPROF has zero "observer effect", so it can be used to profile applications that adjust their activity to their performance. It has no overhead on target machine, so it can be used for profiling embedded, hand-held, and IoT devices which usually have limited support for collecting, and limited resources for storing, the profiling data. Finally, since EMPROF can profile the system as-is, its profiling of boot code and other hard-to-profile software components is as accurate as its profiling of application code. To illustrate the effectiveness of EMPROF, we first validate its results using micro-benchmarks with known memory behavior, and also on SPEC benchmarks running a cycle-accurate simulator that can provide detailed ground-truth data about LLC misses and processor stalls. We then demonstrate the effectiveness of EMPROF on real systems, including profiling of boot activity, show how its results can be attributed to the specific parts of the application code when that code is available, and provide additional insight on the statistics reported by EMPROF and how they are affected by the EM signal bandwidth provided to EMPROF.**

***Index Terms*—EM side-channel, last-level cache, main memory, memory profiling, EM emanation, system profiling, cycle-accurate simulation, code attribution**

I. INTRODUCTION

The disparity between processor and memory speed, known as *"memory gap"* would be a major performance detractor without multiple levels of caches. Of particular importance is the last level cache (*LLC*) which is the largest and slowest of the cache levels but still has an order of magnitude faster access time than a main memory access. Consequently, optimization approaches such as prefetching try to increase the percentage of memory references that result in cache hits, while methods that improve Instruction Level Parallelism (*ILP*) and Memory Level Parallelism (*MLP*) try to hide the latency of cache misses [1]. However, in many applications, such as databases, web servers, scientific algorithms, and modern data analytics 50% to 80% of CPU cycles are still spent on stalling caused by LLC misses [2], [3].

Memory profiling approaches are designed to help programmers and system developers characterize the memory behavior of a program, and this information can then be used by the compiler and/or programmer to improve performance [4], [5]. In general, simulation, hardware support, and program instrumentation are three main methods of profiling cache behavior. Cache simulation can help identify general locality problems that would cause performance degradation on cache-based systems in general [6], but is typically much slower than native execution and fails to model the complex architectural details of modern LLCs and their interaction with memory. Hardware support typically takes the form of hardware performance counters, which are counting actual microarchitectural events as they occur without significant performance overheads due to counting itself [7]. However, an interrupt is needed to attribute a counted event to a specific part of the code, which would cause major performance degradation if every event is attributed. Instead, a sampling method is used, typically by interrupting each time the count reaches some pre-programmed threshold T, to provide statistical attribution of the events [7]–[10]. This creates a trade-off between 1) the granularity at which events are attributed, and 2) the overhead introduced by profiling and the distortion of results by profiling activity itself [11]–[13]. Program instrumentation methods can also be a powerful memory profiling approach [14], [15], but they have a similar trade-off between precision/granularity and overhead/disruption.

Problems associated with hardware counters and/or program instrumentation are exacerbated in real-time and cyber-physical systems, where programs have to meet real-time deadlines, and thus often change behavior depending on their own performance. In such systems, the overheads and memory access pattern interference caused by profiling activity lead the profiled program execution to no longer be representative of the profiling-free execution. Moreover, while high-end processors provide counters and have excess performance that can be used for profiling activity, in many embedded and internet-of-things (IoT) systems, cost and form-factor limitations mandate use of simpler processors that lack the needed performance counter support and/or cannot keep up with their regular workload if profiling activity is added to it.

This work has been supported, in part, by NSF grant 1563991 and DARPA LADS contract FA8650-16-C-7620. The views and findings in this paper are those of the authors and do not necessarily reflect the views of NSF and DARPA.

*†These two authors contributed to the paper similarly (author order does not reflect the extent of contribution).

978-1-5386-6240-3/18/$31.00 ©2018 IEEE
DOI 10.1109/MICRO.2018.00076

Finally, memory profiling based on counting events, such as LLC misses, collect information that is only a proxy for the actual performance impact of these events, and in many cases it would be more useful to account for the actual stall activity these events cause rather than the number of the events. To overcome this challenge, some high-end processors provide `stalled-cycles-frontend` and `stalled-cycles-backend` performance counters that allow measurement of the overall number of stall cycles in a period of time, but to limit performance impact and interference these are not attributed to individual LLC miss events. If such stall-time accounting could be attributed to individual LLC misses, it would provide better understanding of which misses are the most costly in terms of performance, and it would also provide tail latencies among LLC misses (misses whose latency is much higher than average), which would help when trying to debug real-time behavior of many embedded and IoT systems.

In this paper we introduce EMPROF, a new approach to profile memory and its impact on performance. Unlike previous profiling approaches, EMPROF monitors the electromagnetic (EM) emanations produced by the processor as it executes instructions. 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. Finally, we show how EMPROF and Spectral Profiling [16] can be applied to the same EM signal to attribute each event/stall identified by EMPROF to the loop-level regions of code (identified by Spectral Profiling) in which that event occurs.

The key contributions of this paper are:

- A new memory performance profiling method that can be applied without any impact (or even contact with) the profiled system,
- A proof-of-concept implementation of the new method for memory performance profiling,
- An experimental validation and evaluation of this profiler on cycle-accurate simulation and on three real-world systems.

The remainder of this paper is organized as follows: Section II discusses some background information on memory stalls and EM side-channel, Section III provides an overview of how memory accesses are reflected in the side-channel signal, Section IV describes the proposed proof-of-concept implementation of EMPROF, Section V describes the methodology used for validating profiling results from EMPROF, Section VI presents the results of our experimental evaluation, Section VII reviews the related state-of-the-art, and Section VIII presents some concluding remarks.

II. BACKGROUND

A. Side-Channel Signals from Processor Activity

Current flow caused by switching activity results in unintended electromagnetic (EM) emanations [17]. Traditionally, these emanations were used as a side-channel for recovering sensitive data, such as stealing the secret keys during execution of cryptographic algorithms [18]–[22].

More recently, EM emanations and other physical side channels (e.g. variation in power consumption over time) have also been used to detect anomalous (e.g. malware-affected) execution [23]–[26] and to attribute execution to a specific part of the application [16], [27]. Specifically, Spectral Profiling [16] compares short-term spectra of EM emanations to those obtained during training to recognize which loop-granularity region of code the signal corresponds to. One of the main reasons for using spectra rather than the time-domain signal itself was to make signal matching resilient against the effects of microarchitectural events, but unfortunately this also makes such spectral approaches unsuitable for EMPROF, whose goal is to identify specific microarchitectural events (stalls caused by LLC misses). In ZOP [27], the time-domain signal is used to reconstruct the execution path with a finer granularity by testing against multiple (many) hypotheses about which path in the program was taken and using signal matching to determine which of those paths is the most likely. This ZOP-style matching based on hypothetical path exploration has a very high computational cost, and can only be applied to short bursts of execution, so it is not suitable for memory profiling of real systems.

Unlike prior methods that attribute execution to a specific part of the code based on signal similarity to previously recorded examples of signals that correspond to each part of the code, EMPROF relies on an observation that the processor's circuitry exhibits much less switching activity when a processor has been stalled for a while.

B. Processor Stalls Caused by Cache Misses

In general, a processor has two types of stalls. Front-end stalls occur when the front-end of the processor cannot fetch instructions, e.g. because of a miss in the instruction cache (I$), and the processor is fully stalled when it completes instructions it already fetched but still cannot fetch new ones. This typically occurs for hundreds of cycles when the I$ miss is also a miss in the LLC (which is typically unified for instructions and data) and thus experiences the main memory latency. Back-end stalls occur when the processor cannot retire an instruction for a while, e.g. because of a data cache (D$) miss, and the processor is fully stalled when it runs out of one or more of its resources (such as ROB entries, load-store queue entries, etc.) and can no longer fetch new instructions. This typically occurs for hundreds of cycles when the D$ miss is also an LLC miss and thus experiences the main memory latency.

In a sophisticated out-of-order processor, the fully-stalled condition is averted for tens of cycles because the processor already has many tens of instructions in various stages of completion when an I$ miss occurs, and because it has plentiful resources when the D$ miss occurs. The exact number of cycles during which the processor can keep busy depends on the program and the processor's microarchitectural state at the point of the miss, but an LLC miss has latencies in the hundreds of cycles and thus typically still results in numerous fully-stalled cycles. Most resource-constrained devices, such as IoT and hand-held devices, use simple in-order cores that need less power and produce less heat [28], [29], but can avert a full stall for fewer cycles during an LLC miss. These simple cores are superscalar and still can benefit from ILP and MLP by dispatching and executing multiple instructions in-order and send more than one memory request on multiple read channels to multi-banked LLC. Because non-stalled cycles during a cache miss (by exploiting ILP) have far less impact on performance than stalled cycles, and because multiple cache misses that are in progress concurrently (by exploiting MLP) result in fewer overall number of stalled cycles than if these misses were handled with no overlap, accounting for the processor's stalled time rather than counting its LLC cache misses provides better insight into how (and why) LLC misses affect a program's performance. The role of this variable memory latency in performance prediction is studied in [30], [31]. Hardware counters indicating stall cycles due to off-chip memory accesses are used to estimate this latency [32]. Our focus in this paper will be primarily on accounting for and attributing stalled cycles, but for clarity of presentation we will often use the term MISS to refer to a sequence of stalled cycles that are all caused by one LLC miss or even by several highly-overlapped LLC misses. Also, we will often use the term MISS LATENCY to refer to the number of stalled cycles that are all caused by one LLC miss or even by several highly-overlapped LLC misses.

III. Overview of EMProf

A. *Side-Channel Signal During a Processor Stall*

Since changes in the EM signal variation are mostly caused by changes in circuit activity, this means that long stalls will have signals that are relatively stable from cycle to cycle, and that the signals during different long stalls will be similar to each other and very different to signals coming from a busy processor. Our actual measurements confirm this, with the dashed blue line in Fig. 1 showing the magnitude of a time-domain signal acquired with a 40 MHz bandwidth around the clock frequency (1.008 GHz) of the ARM Cortex-A8 processor on the A13-OLinuXino-MICRO IoT single-board computer [33]. Also shown (solid red line) in Fig. 1 is the moving average of the signal. The part of the signal that corresponds to the processor stall is between the two dotted vertical lines in Fig. 1, and the duration of this part of the signal is Δt. This duration can be determined from the signal by multiplying the number of samples in that interval with the duration of a sample period, and the number of cycles this stall corresponds to can be computed by multiplying Δt with the processor's clock frequency.

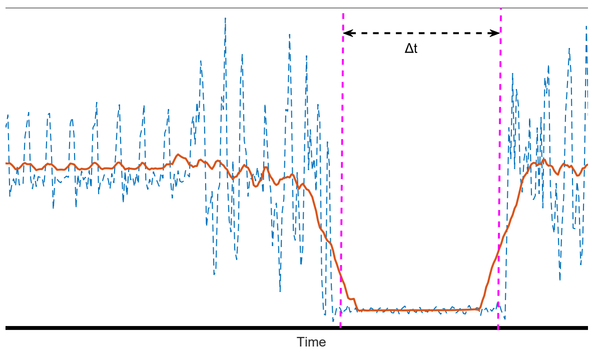


Fig. 1. Change in EM emanation level caused by processor stall, with signal magnitude shown in dashed blue and its moving average shown as solid red.

B. *Memory Access in Simulated Side-Channel Signal*

This section reviews the types of stalls that we have observed in the side-channel signals. We begin with signals obtained by modeling power consumption in a cycle-accurate architectural simulator SESC [34], because the simulator can provide us with the ground-truth information about the exact cycle in which the miss is detected, in which the resulting stall begins, and in which the stall ends. We model a 4-wide in-order processor, with two levels of caches with random replacement policies, which mimics the behavior of the processors encountered in many IoT and hand-held devices. We collect the average power consumption for each 20-cycle interval, which corresponds to a 50 MHz sampling rate for a 1 GHz processor.

To observe how the side-channel signal is affected by different types of cache misses, a small application was created that performs loads from different cache lines in an array. The size of the array can be changed in order to produce cache misses in different levels of the cache hierarchy. Two signals that correspond to the same part of the code are shown in Fig. 2. Fig. 2a corresponds to an L1 D$ miss that is an LLC hit

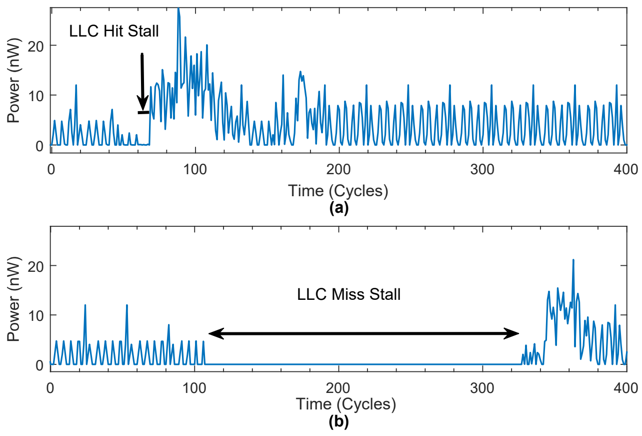
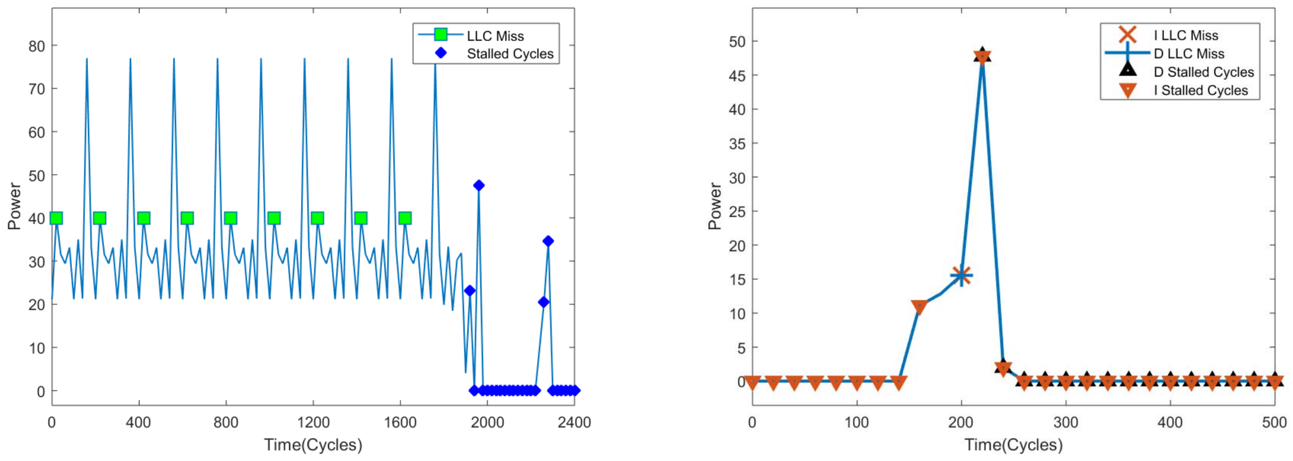


Fig. 2. (a) LLC hit stalls and (b) LLC miss stalls in SESC simulator.



(a) Stalls can be avoided for some LLC misses.

(b) Stalls for coinciding misses overlap.

Fig. 3. Some LLC miss events produce no invididually attributable stalls.

containing a very brief stall, during which the processor core consumes very little power because none of its major units are active. In contrast, Fig. 2b corresponds to an LLC miss having order-of-magnitude longer low-power-consumption period that corresponds to a much longer stall.

When the sampling rate is reduced, each data point (sample) in the signal corresponds to an increased number of cycles, and thus shorter stalls become increasingly difficult to find in the signal. The long stalls that correspond to LLC misses, however, still contain multiple signal samples that allow the stall to be identified. The reduction in the signal's sampling rate does, however, reduce the signal measurement resolution of the duration of those stalls - e.g. with one signal sample per 20 processor cycles, the duration of the stall can be "read" from the signal only in 20-cycle increments.

We have confirmed that the power signal produced by SESC drops to its full-stall level only a number of cycles after the miss occurs, after the processor runs out of *useful* work. As explained before, this occurs either when the processor suffers an I$ miss and eventually completes the instructions it did fetch prior to that, or when the processor has a D$ miss, finishes all the work that could be done without freeing any pipeline resources, and eventually cannot fetch any more instructions until the miss "drains" out of the pipeline.

Fig. 3a shows a signal that corresponds to several consecutive LLC misses. When the first miss occurs, the processor still has a lot of resources and its activity continues largely unaffected by the miss. During this activity, several other LLC misses occur and eventually the first miss ends while the processor is still busy and has not stalled yet. This allows the processor to fetch more instructions, staying busy until the second LLC miss ends, etc. until eventually the processor encounters a miss during which it does run out of resources and is forced to stall. Since the first several LLC misses in this example do not cause any stall activity, a detector of LLC misses based on identifying the resulting stalls in the signal will fail to detect those misses, which will cause under-counting of LLC miss events. However, since those misses do not introduce stalls, their impact on performance is much lower compared to stall-inducing misses, so the signal-based reporting of miss-induced stalls in this example would still be close to the actual performance impact of these misses.

Another situation in which the LLC miss count would be under-reported involves overlapping LLC misses. One example of such a situation is shown in Fig. 3b, where an I$ access and a D$ access are both LLC misses that overlap and the processor is stalled during that overlapped time. When analyzing the side-channel signal to identify stalls caused by LLC misses, this usually results in reporting only one stall, which would be counted as one LLC miss. If the goal of the profiling is to actually count LLC misses, rather than account for their impact on performance, this will cause under-counting of LLC misses. However, this situation is a typical example where the MLP allows the performance penalties of multiple LLC misses to overlap with each other, and the resulting overall performance impact is not much worse than if there was only one LLC miss, so the signal-based reporting of how many long stalls occur and the performance impact of each such stall still very accurately tracks the actual performance impact of the LLC misses.

C. Memory Access in Physical EM Side-Channel Signal

We now run the same small application on Olimex A13-OLinuXino-MICRO [33], an IoT prototyping board. We use a near-field magnetic probe to measure the board's EM emanations centered around the processor's clock frequency. Even though we are using EM emanations from a real system, rather than power consumption from a simulated system, stalls produced by cache misses in this system still produce signal patterns (Fig. 4) similar to that observed for the simulator - stalls cause a significant decline in signal magnitude. The stalls produced by most LLC misses lasts around 300 ns, with small

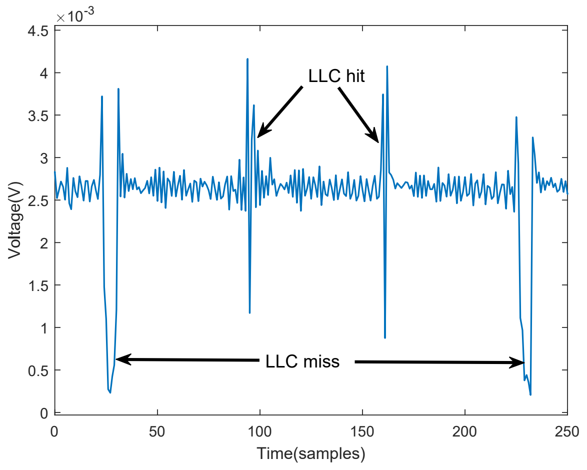


Fig. 4. LLC hit and miss from physical side-channel signal of Olimex board.

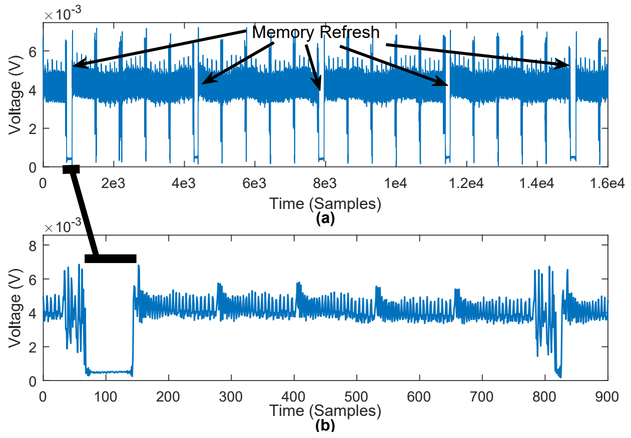


Fig. 5. Memory refresh in Olimex IoT device, with (a) memory refresh replacing LLC miss, (b) zoomed-in memory refresh.

variations that are likely due to variations in how much work the processor can do until it stalls after encountering an LLC miss.

However, the stalls caused by LLC misses exhibit several behaviors that were not encountered in the simulator because it used a simplified model of the main memory. One such behavior, shown in Fig. 5, occurs when an LLC miss occurs while the memory is performing its periodic refresh activity.

We observed that a stall for an LLC miss that coincides with a memory refresh lasts approximately 2-3 μs, and this situation occurs approximately at least every 70 μs for the on-board H5TQ2G63BFR SDRAM chip [35]. Since these stalls do affect program performance and (especially) the tail latency of memory accesses, we count them (and account for their performance impact) separately when reporting our experimental results.

IV. Prototype Implementation of EMProf

Because cache misses occur at a time when the processor is busy, the signal shape at the point of the miss changes significantly depending on the surrounding instructions and the processor's schedule for executing them. Therefore, rather than attempting to recognize the signal that corresponds to the misses themselves, which would require extensive training for each point in the code where the misses can occur, EMProf detects the signal activity that corresponds to a stalled processor. The first step in this detection is to normalize the signal to compensate for the variation among different devices, measurement setup, etc. For example, we have found that even small changes in probe/antenna position can dramatically change the overall magnitude of the received signal. However, this change largely consists of a constant multiplicative factor that is applied to the entire signal. Similarly, the voltage provided by the profiled system's power supply vary over time. The impact of power supply variations on the EM emanations is largely that signal strength changes in magnitude over time. EMProf compensates for these effects by tracking a moving minimum and maximum of the signal's magnitude and using them to normalize the signal's magnitude to a range between 0 (which corresponds to the moving minimum) and 1 (which corresponds to the moving maximum). EMProf then identifies each significant *dip* in the signal whose duration exceeds a threshold. The threshold is selected to be significantly shorter than the LLC latency but significantly longer than typical on-chip latencies.

One of the key advantages of EMProf is that it can efficiently identify LLC-miss-induced processor stalls without any a-priori knowledge about or training on the specific program code that is being profiled. This enables EMProf to profile execution equally well regardless of the program analysis tools and infrastructure available for the target system and software. One specific example of this versatility is that EMProf can be used for memory profiling of the system's boot from its very beginning, even before the processor's performance monitoring features are initialized and before the software infrastructure for saving the profiling information is loaded.

V. Validation of EMProf

EMProf's goal is to identify in the side-channel's signal each LLC miss that stalls processor and to measure the duration of that stall. Since existing profiling methods do not operate at such level of detail, validation of EMProf is a non-trivial problem. One problem is that hardware performance counters, such as `LLC-load-misses`, count all LLC miss events, including those that cause no stalls and those that overlap with other misses. Another problem is that their accuracy is reduced by either high interference with the profiled execution (when sampling the counter value often) or significant sampling error (when sampling the counter value rarely), so they are not very effective for relatively brief periods of execution. One illustration of this is that, as will be reported in more detail in Section VI, when using `perf` [10]

TABLE I
SPECIFICATIONS OF EXPERIMENTAL DEVICES

	Alcatel	Samsung	Olimex
Processor	QS[a] MSM8909	QS[a] MSM7625A	AW[b] A13 SoC
Frequency	1.1 GHz	800 MHz	1.008 GHz
#Cores	4	1	1
ARM Core	Cortex-A7	Cortex-A5	Cortex-A8

[a]Qualcomm Snapdragon
[b]Allwinner

```
// perform page touch
for(#pages_to_be_used)
    load(page(cache_line_0))

exec_blank_loop() // empty for loop

// perform memory loads
// TM: Total Misses requested
while(num_accesses != TM)
    page = rand()
    cache_line = rand()
    addr = page*PAGE_SIZE + cache_line*CACHE_LINE_SIZE
    load(addr)
    // CM: Consecutive Misses occurring in groups
    if(num_accesses % CM == 0)
        micro_function_call()
    num_accesses++

exec_blank_loop() // empty for loop
```

Fig. 6. Pseudocode of the microbenchmark.

on Olimex A13-OLinuXino-MICRO to count LLC misses for a small application that was designed to generate only 1024 cache misses, the number of misses reported by `perf` had an average of 32,768 and a standard deviation of 14,543.

Thus, we validate EMPROF's results in two ways. First, we engineer a microbenchmark that generates a desired number of LLC misses and compare EMPROF's reported LLC miss count to an a-priori-known number of misses. Second, we apply EMPROF to the power side-channel signal generated through cycle-accurate simulation, and then compare EMPROF's results to the simulator-reported ground truth information about where the misses occur in the signal's timeline and where the resulting stall begins and ends.

A. Experimental Setup

To demonstrate EMPROF's ability to profile LLC misses, we ran the engineered microbenchmark on Android-based cell phones - Samsung Galaxy Centura SCH-S738C [36] and Alcatel Ideal [37], and on the Olimex A13-OLinuXino-MICRO [33] IoT prototype board. More detail on these three devices is provided in Table I.

In our experimental setup, the signals received using a small magnetic probe are recorded using a Keysight N9020A MXA spectrum analyzer [38]. The spectrum analyzer was used for initial studies as it has built-in support to visualize signals in real-time, offers a range of measurement bandwidth and perform basic time-domain analysis.

B. Validation by Microbenchmarking

We implement a microbenchmark (Fig. 6) which generates a known pattern of memory references leading to LLC misses. The access pattern of the microbenchmark can be adjusted to produce LLC misses in groups, where the number of LLC misses in a group and the amount of non-miss activity between groups can be controlled. We refer to the total number of expected LLC misses as *TM* and the number of LLC misses consecutively in groups as *CM*. For example, if *TM*=100 and *CM*=10, the microbenchmark creates 10 groups of 10 consecutive LLC misses, each group separated by a micro-function call.

First, every page is accessed once to avoid encountering page faults later. Then the microbenchmark executes a tight `for` loop with no memory accesses. The signal that corresponds to this loop has a very stable signal pattern that can be easily recognized, which allows us to identify the point in the signal where this loop ends and the part of the application with LLC miss activity begins.

Next the microbenchmark executes a section of code that contains the LLC misses. The access pattern accesses cache-block-aligned array elements (so that each access is to a different cache block), with randomization designed to defeat any stride-based pre-fetching that may be present in the processor. Finally, another tight `for` loop allows us to easily identify the point in the signal at which the carefully engineered LLC miss activity has ended.

Fig. 7a shows the overall EM signal from one execution of this microbenchmark on the Olimex A13-OLinuXino-MICRO board. The "memory accesses" portion of the signal includes many LLC misses, which occur in groups of 10, i.e. *CM*=10. A zoom-in that shows the signal for one such group of 10 LLC misses is also shown in Fig. 7b.

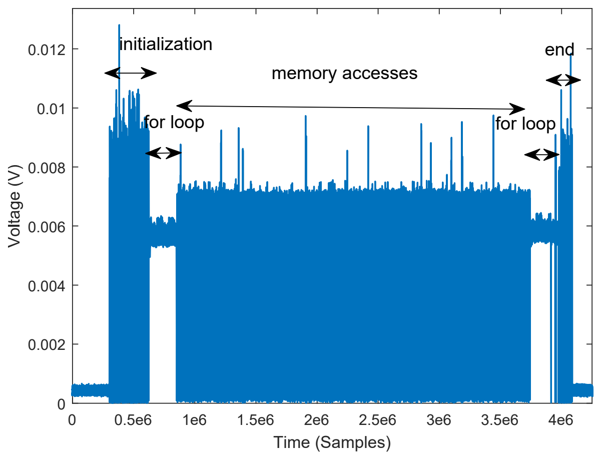
TABLE II
ACCURACY OF EMPROF FOR MICROBENCHMARKS ON ALCATEL CELL PHONE, SAMSUNG CELL PHONE AND OLIMEX IOT DEVICE

Benchmark		Devices		
#TM	**#CM**	**Alcatel**	**Samsung**	**Olimex**
256	**1**	99.61%	99.22%	99.61%
256	**5**	100%	99.61%	99.22%
1024	**10**	99.41%	99.61%	99.51%
4096	**50**	99.83%	99.71%	98.98%

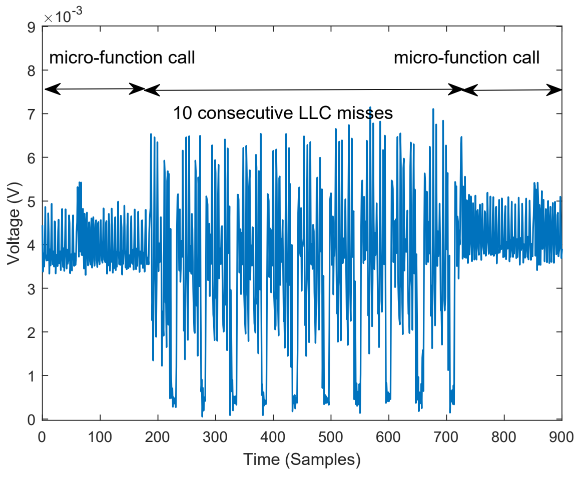
Because the number of misses in the "memory accesses" section of the microbenchmark is known, and because this section can easily be isolated in the signal, we can apply EMPROF to this section and compare the LLC miss count it reports to the actual LLC miss count the section is known to produce. The results of this comparison were used to compute EMPROF's accuracy shown in Table II. On all microbenchmarks the accuracy of EMPROF's LLC miss counting is above 99%, with an average of 99.52%.

C. Validation by Cycle-Accurate Architectural Simulation

To evaluate EMPROF's ability to measure the duration of each LLC-miss-induced stall, and to assess the impact of



(a) Entire microbenchmark run.



(b) Zoomed-in LLC miss section with *CM*=10.

Fig. 7. EM signal from single microbenchmark run on Olimex device.

overlapping LLC misses, we use a simulator configuration that mimics the processor and cache architecture of the Olimex A13 OLinuXino-MICRO board. The simulator is enhanced to produce a power consumption trace that will be used as a side-channel signal in EMPROF, and also to produce a trace of when (in which cycle) each LLC miss is detected and when the resulting stall (if there is a stall) begins and ends.

We first run the same microbenchmarks we ran on the Olimex board and compare the two signals (Fig. 8). We observe that, although one signal is produced by the simulator's (unit-level) accounting of energy consumption and the other is produced by actually receiving EM emanations from a real processor (Fig. 8b), the relevant aspects of the two signals are similar in nature - the loops used to mark the beginning and ending of the "memory accesses" section in the microbenchmark are clearly visible and can be easily identified in the signal, and the LLC misses themselves exhibit similar behavior in the signal. The most prominent difference between the two overall signals is that the start-up and tear-down involves much more activity on the real system than on the simulator (Fig. 8a), primarily because the simulation begins at the entry point and ends at the exit point of the microbenchmark's executable, whereas in the real system the start-up and tear-down also involves system activity, e.g. to load and prepare the executable for execution before any of the executable's instructions are executed.

Having determined that the simulator's power signal is a reasonable proxy of the real system's EM signal for the purposes of EMPROF validation, we proceed to run the microbenchmarks and several SPEC CPU2000 benchmarks in the simulator, analyze the resulting power signals in EMPROF, and compare EMPROF's results to the ground-truth results recorded by the simulator. The results of this comparison are shown in Table III in terms of accuracy of the reported LLC miss count and the reported LLC-miss-induced stall cycles.

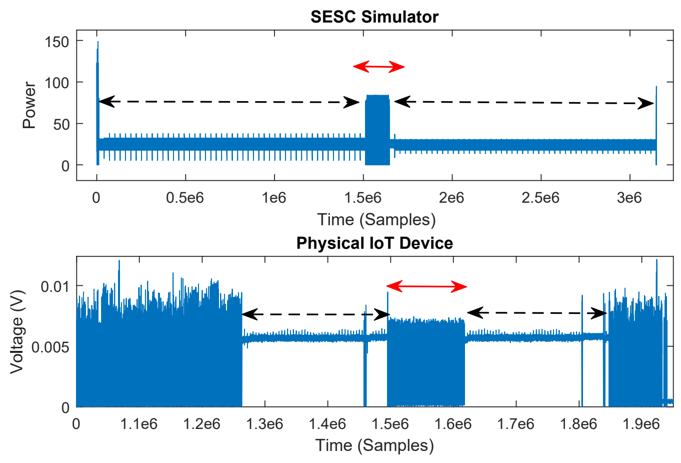
D. Validation by Monitoring Main Memory's Signals

Finally, we wanted to systematically verify that the stalls reported by EMPROF coincide with actual memory activity. Intuitively, when a memory request is getting fulfilled by the main memory, the activity level in main memory should increase. Similar to how we receive the EM emanations from the processor, we can also receive the EM emanations from main memory. By receiving the two signals simultaneously and checking whether the dips in the processor's signal coincide with memory activity, we can gain additional confidence that the dips detected by EMPROF are indeed a consequence of LLC misses.

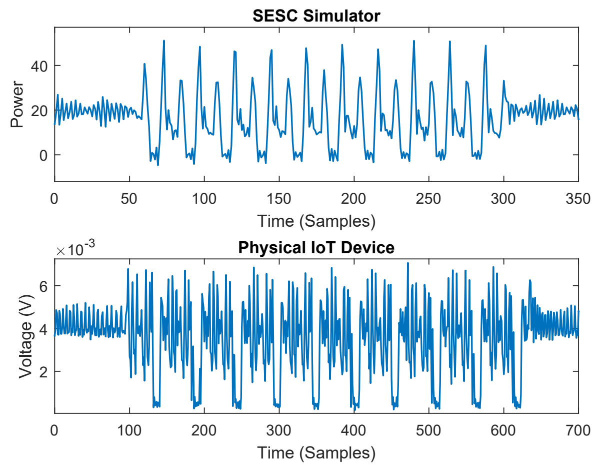
Unfortunately, not every device is emenable to such dual-probing - the processor and memory tend to be physically close to each other, so the physical placement of both probes is a challenge. However, we found that on the Olimex board, the processor and memory are relatively spaced apart, allowing

TABLE III
ACCURACY OF EMPROF ON SIMULATOR DATA FOR BENCHMARKS

Benchmark		Miss Accuracy(%)	Stall Accuracy(%)
#TM	**#CM**	**Microbenchmark**	
256	**1**	97.7	99.3
256	**5**	97.8	99.3
1024	**10**	99.4	99.9
4096	**50**	99.8	99.8
SPEC CPU2000			
ammp		99.67	98.4
bzip2		95.2	99.96
crafty		99.31	100
equake		93.2	98.5
gzip		99.96	99.8
mcf		99.9	99.5
parser		99.9	99.7
twolf		99.16	100
vortex		99.04	99.9
vpr		100	99.24



(a) An entire microbenchmark run. Solid red arrows indicate the LLC-miss-intensive part, while dashed black arrows indicate the identifier loops.



(b) Zoom-in to the LLC-miss-intensive part with *CM*=10.

Fig. 8. Comparison of signals from the SESC simulator and actual Olimex A13-OLinuXino-MICRO IoT device.

simultaneous probing with no interference. We additionally use a passive probe to measure the CAS pin activity off a resistor, and this experimental setup is shown in Fig. 9.

Fig. 10a shows the magnitude of processor's and memory's side-channel signal for *CM*=10 to the main memory that are separated by non-memory activity. We observe that an LLC miss causes the processor's signal magnitude to drop significantly (Fig. 10b), while the memory request caused by the LLC miss result in a sudden burst of memory activity.

Finally, one may intuitively expect that the memory's EM signal might be a better indicator of LLC misses than the processor's signal. Unfortunately, this is not true - while it is true that memory activity occurs on each LLC miss, memory activity also occurs due to DMA transfers, DRAM refreshes, and many other reasons that are completely unrelated to LLC misses. Furthermore, by measuring stalls in the processor's EM signal, we obtain information that is more relevant for performance optimizations - how often the processor stalls due to LLC misses and how long these stalls last.

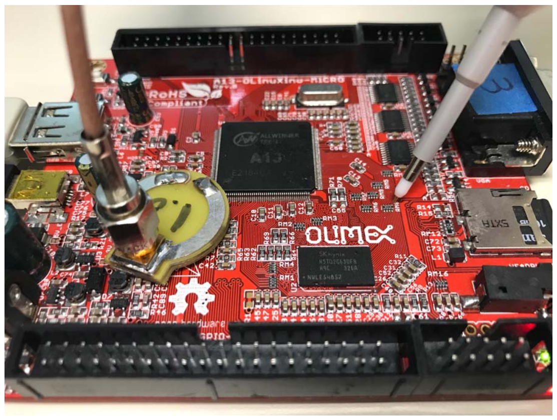


Fig. 9. Measurement setup for simultaneous monitoring of the processor's emanations and memory signals.

VI. Experimental Results

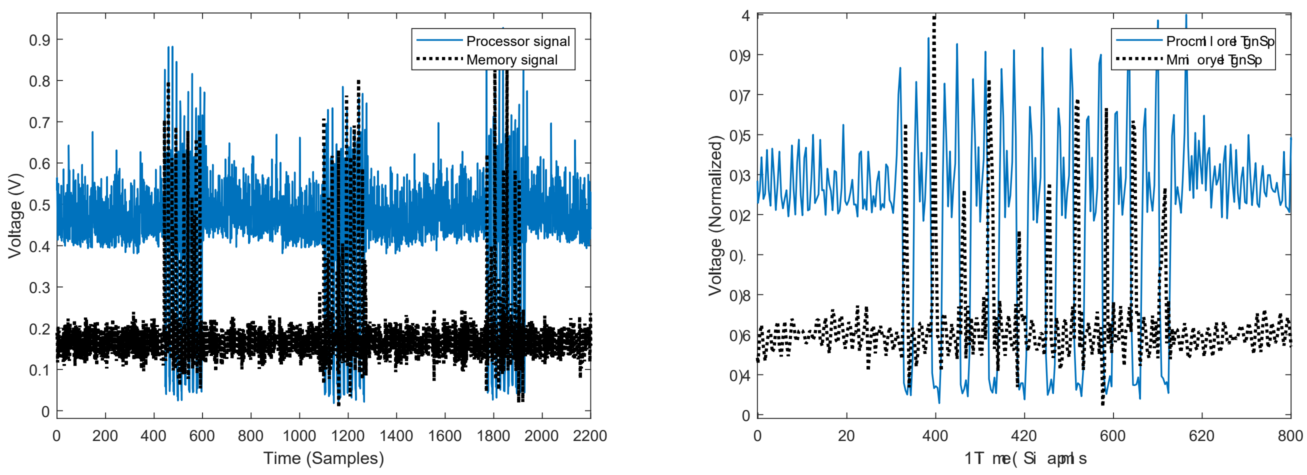
To evaluate the effectiveness of EMProf on real-world applications, it was additionally applied to ten SPEC CPU2000 benchmarks [39] on the two Android cellphones and one IoT device. The integer SPEC CPU2000 benchmarks were chosen as they display a range of realistic memory behaviors [4].

The final results of SPEC CPU2000 benchmarks were recorded using near-field magnetic probe, by first down-converting the signal using a ThinkRF Real Time Spectrum Analyzer WSA5000 [40] and then digitizing the signal using a computer fitted with two PX14400 high speed digitizer boards from Signatec [41]. This was needed because the N9020A MXA has a limit on how long it can continuously record a signal, and most SPEC benchmarks execution times significantly exceed this limit.

A. Profiling Results

Table IV shows the results for the total number of LLC cache misses and the number of stall cycles (as a percentage of the benchmark's execution time), as reported by EMProf for each benchmark on each target device.

The number of LLC misses is much lower for the Alcatel cellphone than for the other two devices, mainly because the LLC in Alcatel is 1 MB while Olimex and Samsung device both have a 256 KB LLC, so the LLC miss rate for Alcatel can be expected to be much lower. Samsung device's processor has a hardware prefetcher, so it is able to avoid some of the LLC misses that occur in the Olimex device, and therefore the number of misses Olimex can be expected to be higher



(a) Three groups of LLC misess with periods of non-LLC-miss activity in between.

(b) Zoom-in showing a single group of LLC misses*CM*=10.

Fig. 10. EM side-channel emanations from the processor (solid blue) and memory (dotted black) for a microbenchmark with *CM*=10.

TABLE IV
STATISTICS OF TOTAL LLC MISSES AND TOTAL PERCENTAGE LATENCY IN EXECUTION TIME OBTAINED FROM EMPROF FOR ALCATEL CELL PHONE, SAMSUNG CELL PHONE AND OLIMEX IOT DEVICE

Benchmark		**Total LLC Misses**			**Miss Latency (%Total Time)**		
		Devices			**Devices**		
		Alcatel	**Samsung**	**Olimex**	**Alcatel**	**Samsung**	**Olimex**
#TM	**#CM**	**Microbenchmark**					
256	**1**	257	254	255	0.92	3.57	9.44
256	**5**	256	255	258	1.15	4.06	10.10
1024	**10**	1030	1020	1029	1.00	4.19	9.88
4096	**50**	4103	4084	4138	2.05	4.55	10.25
		SPEC CPU2000					
ammp		20511	280141	174142	2.42	7.59	9.06
bzip2		451103	3353123	5932148	2.15	1.04	6.59
crafty		314385	901153	675781	2.44	0.38	1.52
equake		627734	2138132	5925404	2.68	0.61	7.49
gzip		152264	1001392	513256	1.11	0.39	1.21
mcf		303365	895295	546714	5.22	7.18	3.28
parser		365412	1934334	2318384	2.19	3.39	8.63
twolf		35086	1014888	228465	1.03	4.84	3.00
vortex		235667	897317	533784	3.63	2.03	2.9
vpr		4970	100750	203130	0.09	0.23	0.6
Average		251049.7	1251652.5	1705120.8	2.3	2.77	4.43

than in the Samsung phone even though their LLCs have the same size. Additionally, the processor in the Olimex board has a higher clock frequency than the processor in the Samsung phone, while their main memory latencies (in nanoseconds) are very similar, creating more stall time per miss in the Olimex board and also allowing fewer LLC misses to be completely hidden by overlapping them with useful work in the processor.

While the total stall cycles provides an indication of how performance is affected by LLC misses overall, a key benefit of EMPROF is that it also provides information about the stall time of each LLC miss. Fig. 11 shows the histogram of LLC miss latencies observed on the three devices for the SPEC CPU2000 benchmark *mcf*. Most stalls are brief in duration, mostly due to the processor's ability to keep busy as the miss is being serviced, as explained before in Section III. However, a significant number of stalls last hundreds of cycles, and we observe that, compared to the IoT board, the two phones have a thicker "tail" in the stall time histogram.

B. Effect of Varying Measurement Bandwidth

During our initial study, we tested the effect of a varying measurement bandwidth from 20 MHz, 40 MHz, 60 MHz, 80 MHz and 160 MHz. Low measurement bandwidth may not introduce enough samples to exemplify the cache miss feature, and with increasing measurement bandwidth, an increase in sampling rate correlates to a better and more accurate detection of LLC miss and its associated latencies. Fig. 12 illustrates this effect for Alcatel cell phone and A13-OLinuXino-MICRO running SPEC CPU2000 *mcf* benchmark. In the IoT device, a lower sampling rate mostly results in less accurate stall latency determination. However, for the Alcatel cell phone the lowest sampling rates also prevent detection of many LLC-induced stalls, such that at 20 MHz EMPROF detects only

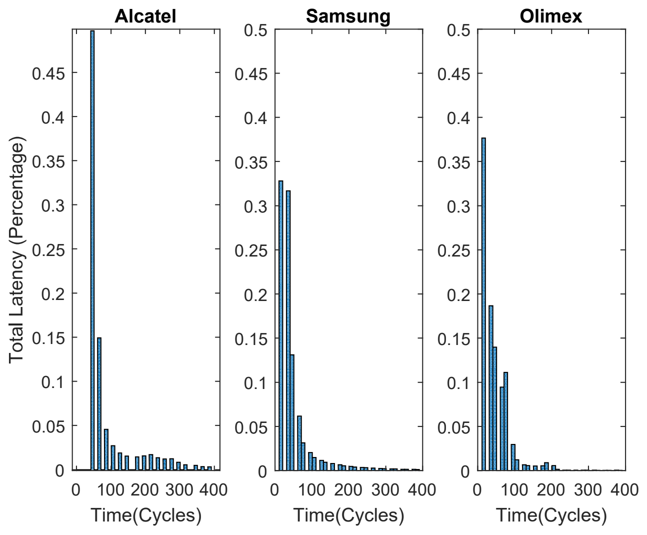


Fig. 11. Histogram of stall latencies obtained for SPEC CPU2000 *mcf* benchmark for Olimex IoT device, Alcatel cell phone and Samsung cell phone.

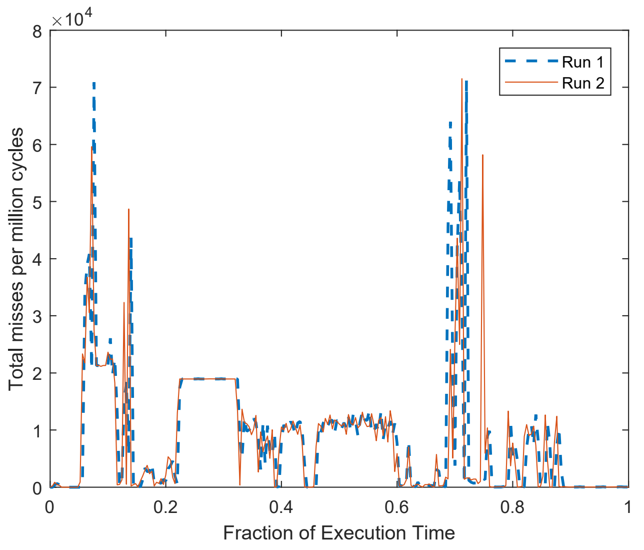


Fig. 13. Boot sequence EMPROF profiling for two distinct runs on IoT device.

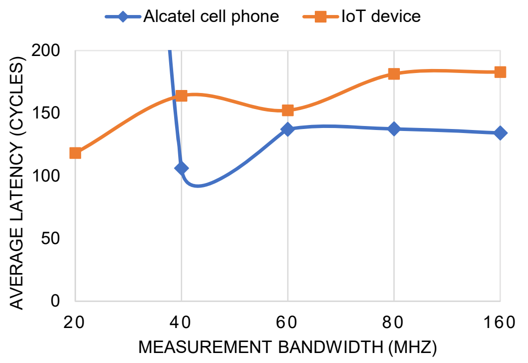


Fig. 12. Effect of varying measurement bandwidth for SPEC CPU2000 *mcf* benchmark across Alcatel cell phone and IoT device.

the very few stalls that have extremely long durations (their average duration is 1100 clock cycles). For both devices, the average stall time stabilizes at 60 MHz or more of measurement bandwidth, indicating that bandwidth equivalent to only 6% of the processor's clock frequency is sufficient to allow identification of LLC-miss-induced stalls in the signal.

C. Profiling the Boot Sequence

One of the most promising aspects of EMPROF is its ability to profile hard-to-profile runs, such as the boot sequence of the device. Fig. 13 shows the rate of total LLC misses as time progresses for two boot-ups of the IoT device. These results can be used to, for example, decide whether memory locality optimization should be considered as a way to speed up the boot of the device.

D. Code Attribution

While accurately profiling stalls due to LLC misses in execution time provides very valuable insight into performance, it would be even more helpful for developers to be able to identify in which parts of the code these LLC misses and stalls are happening. Ideally, such attribution of misses to code would also be based on the EM signal, to retain the zero-interference advantages of EMPROF. Several methods that attribute parts of the signal to application code have been reported in the literature, including Spectral Profiling [16], EDDIE [26], which can attribute parts of the signal to the code at the granularity of loops, and even ZOP [27], which can achieve fine-grain attribution of signal time to code albeit that requires much more computation so it may not be feasible for long stretches of execution. By applying one such scheme and EMPROF to the same signal, the LLC miss stall discovered in the signal by EMPROF's could be attributed to the application code in which they occur. To illustrate this, Fig. 14 shows how the spectrum (horizontal axis) for the SPEC CPU2000 benchmark *parser* changes over time (vertical axis). There are three distinct regions that can be observed in this spectrogram, which correspond to three functions in *parser*. To illustrate how signal-based attribution would be used in conjunction with EMPROF, we (manually) mark the transitions between these function in the signal as shown by horizontal dashed lines in Fig. 14 and attribute misses in each part of the signal to the corresponding function. Table V shows EMPROF's results with this attribution. As we see, the batch_process function should be the main target for optimizations that target LLC misses – it occupies the largest fraction of execution time, it suffers the highest LLC miss rate, and it has the highest fraction of its execution time spent on stalls caused by these LLC misses. We note that this only serves to illustrate how this attribution would work - finer-grain analysis (à la Spectral Profiling) of the signal would identify multiple loops in some of these functions and thus likely provide more precise (and informative) results than those shown in Table V.

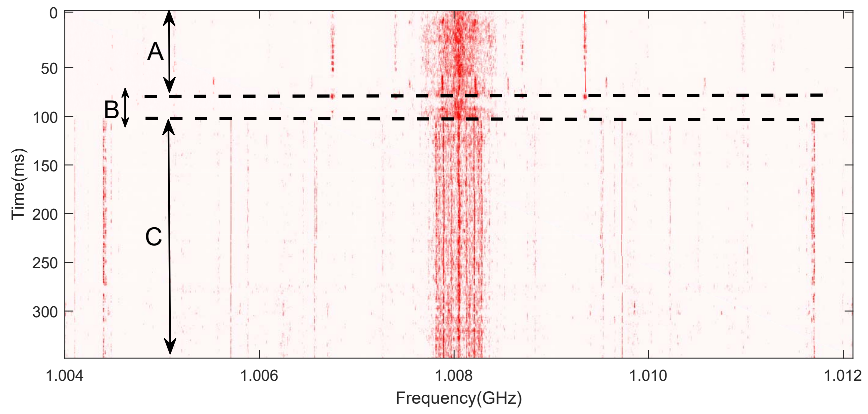


Fig. 14. Spectrogram of SPEC CPU2000 *parser* benchmark.

TABLE V
OBSERVABLE LOOPS IN SPEC CPU2000 *parser* BENCHMARK

Region	Function	Total Miss	LLC Miss Rate(per Million Cycles)	Mem Stall Cycles (%)	Avg. Miss Latency (Cycles)
A	read_dictionary	292296	2667.71	1.63	218.71
B	init_randtable	34805	317.66	0.18	211.85
C	batch_process	1840246	16795.47	10.17	217.72

VII. Related Work

Most modern processors have hardware counters available for performance measurement [7], [42]–[44]. Operating systems provide an interface to these counters [10], [45], [46] and profiling tools access CPU registry directly for faster measurement [47]–[50]. To collect hardware counters a range of methods from simple aggregation, statistical sampling to self-monitoring (instrumentation) are available [13]. While aggregation of counters is collected by operating system on timer or overflow interrupts and saved on context switch, to gather information with finer granularity such as per-function, statistical sampling or instrumentation methods should be used [13]. To collect desired fine-grained information, high sampling rate or software calls by binary instrumentation should be used. Increased interrupt rate as well as binary software calls introduce overhead and may distort the measurement, creating an "observer effect" [11]–[13], [51], [52].

Increase in popularity of embedded systems, hand-held and IoT devices and insufficient debug infrastructure has increased the demand for such special profilers for them [5]. Profiling for the purpose of software optimization is even more important in resource-constrained devices where performance is vital. While there are more options in embedded devices for direct measurement profiling processors such as Logic analyzers (e.g. [53]), Trace/Debug interfaces, and JTAG interfaces, performance counters are still the preferred profiling method in these devices. Some architectures including various ARM processors (e.g Cortex-A9 and Raspberry Pi) which are commonly used in IoT devices, have no or inaccurate overflow interrupt handling which prevents them from supporting sampling [13]. Also adding efficient profiling capabilities to processor can be costly in terms of die area and power consumption which is impractical in such devices [54].

VIII. Conclusions

This paper presents EMProf, a new approach to profile memory behavior in resource-constrained systems such as IoT and hand-held devices. EMProf does not require any hardware or software support including hardware counters or instrumentation in target machine. EMProf measures the EM emanated signal from the processor without any support or assistance of the device-under-test. Hence, it is totally *observer-effect free* and has no interference with the running application such as memory pollution or frequent interrupts. It utilizes this fact that signal level drops when processor gets stalled. By dynamically detecting this signal level, EMProf can accurately find main memory accesses which cause stall in processor and matter for the purpose of performance profiling very accurately in execution time. It is also able to measure effective latency associated with each of these misses.

To validate EMProf, microbenchmarks with known memory behavior are used. Also, EMProf has been tested on simulated side-channel signal where EMProf is able to pin-point expected cache misses with 99.1% accuracy in microbenchmarks and 98.1% in for real applications. Two Android cellphones and an IoT device have been used to evaluate EMProf. It shows a cumulative 99.5% accuracy on acquired signal from these devices. We illustrated how EMProf can profile execution even where no other profiler can be used, such as the boot sequence in these devices. Moreover EMProf can be used with other profilers which are able to profile execution and attribute EM emanation to code sections.

References

[1] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Elsevier, 2011.

[2] A. Ailamaki, D. J. DeWitt, M. D. Hill, and D. A. Wood, "DBMSs on a modern processor: Where does time go?" in *VLDB" 99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, no. DIAS-CONF-1999-001, 1999, pp. 266–277.

[3] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware," *ACM SIGPLAN Notices*, vol. 47, no. 4, pp. 37–48, Mar. 2012. [Online]. Available: http://doi.acm.org/10.1145/2248487.2150982

[4] Q. Wu, A. Pyatakov, A. Spiridonov, E. Raman, D. W. Clark, and D. I. August, "Exposing Memory Access Regularities Using Object-Relative Memory Profiling," in *Proceedings of the International Symposium on Code Generation and Optimization: Feedback-directed and Runtime Optimization*, ser. CGO '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 315–. [Online]. Available: http://dl.acm.org/citation.cfm?id=977395.977677

[5] P. Nagpurkar, H. Mousa, C. Krintz, and T. Sherwood, "Efficient remote profiling for resource-constrained devices," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 3, no. 1, pp. 35–66, 2006.

[6] J. Weidendorfer, M. Kowarschik, and C. Trinitis, "A Tool Suite for Simulation Based Analysis of Memory Access Behavior," in *Computational Science - ICCS 2004*, M. Bubak, G. D. van Albada, P. M. A. Sloot, and J. Dongarra, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 440–447.

[7] M. Zagha, B. Larson, S. Turner, and M. Itzkowitz, "Performance Analysis Using the MIPS R10000 Performance Counters," in *Supercomputing '96:Proceedings of the 1996 ACM/IEEE Conference on Supercomputing*, Jan 1996, pp. 16–16.

[8] P. Agrawal, A. J. Goldberg, and J. A. Trotter, "Interrupt-based hardware support for profiling memory system performance," Jun. 16 1998, uS Patent 5,768,500.

[9] S. Browne, J. Dongarra, N. Garner, K. London, and P. Mucci, "A scalable cross-platform infrastructure for application performance tuning using hardware counters," in *Supercomputing, ACM/IEEE 2000 Conference*. IEEE, 2000, pp. 42–42.

[10] Linux, "Linux kernel profiling with perf," https://perf.wiki.kernel.org/index.php?title=Tutorial&oldid=3520, accessed March 2018.

[11] A. D. Malony, D. A. Reed, and H. A. G. Wijshoff, "Performance measurement intrusion and perturbation analysis," *IEEE Transactions on parallel and distributed systems*, vol. 3, no. 4, pp. 433–450, 1992.

[12] A. D. Malony and S. S. Shende, "Overhead compensation in performance profiling," in *European Conference on Parallel Processing*. Springer, 2004, pp. 119–132.

[13] V. M. Weaver, "Self-monitoring overhead of the Linux perf_ event performance counter interface," in *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, March 2015, pp. 102–111.

[14] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: Building customized program analysis tools with dynamic instrumentation," in *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '05. New York, NY, USA: ACM, 2005, pp. 190–200. [Online]. Available: http://doi.acm.org/10.1145/1065010.1065034

[15] M. Martonosi, A. Gupta, and T. Anderson, "Memspy: Analyzing memory system bottlenecks in programs," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 20, no. 1. ACM, 1992, pp. 1–12.

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

[17] P. Rohatgi, "Electromagnetic attacks and countermeasures," in *Cryptographic Engineering*. Springer, 2009, pp. 407–430.

[18] M. G. Kuhn, "Compromising emanations: eavesdropping risks of computer displays," Ph.D. dissertation, University of Cambridge, 2002.

[19] W. van Eck, "Electromagnetic radiation from video display units: an eavesdropping risk?" *Computers and Security*, vol. 4, no. 4, pp. 269–286, Dec. 1985.

[20] T. Sugawara, D. Suzuki, M. Saeki, M. Shiozaki, and T. Fujino, "On measurable side-channel leaks inside asic design primitives," in *Proceedings of the 15th international conference on Cryptographic Hardware and Embedded Systems - CHES*, ser. CHES'13, 2013, pp. 159–178.

[21] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM side-channel(s)," in *Proceedings of Cryptographic Hardware and Embedded Systems - CHES 2002*, 2002, pp. 29–45.

[22] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer, "Stealing Keys from PCs Using a Radio: Cheap Electromagnetic Attacks on Windowed Exponentiation," in *Cryptographic Hardware and Embedded Systems – CHES 2015*, ser. Lecture Notes in Computer Science, T. Gneysu and H. Handschuh, Eds. Springer Berlin Heidelberg, 2015, vol. 9293, pp. 207–228. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-48324-4_11

[23] S. S. Clark, H. Mustafa, B. Ransford, J. Sorber, K. Fu, and W. Xu, "Current events: Identifying webpages by tapping the electrical outlet," in *European Symposium on Research in Computer Security*, 2013, pp. 700–717.

[24] C. Aguayo Gonzlez and J. Reed, "Power fingerprinting in sdr integrity assessment for security and regulatory compliance," *Analog Integrated Circuits and Signal Processing*, vol. 69, no. 2-3, pp. 307–327, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10470-011-9777-4

[25] S. S. Clark, B. Ransford, A. Rahmati, S. Guineau, J. Sorber, W. Xu, K. Fu, A. Rahmati, M. Salajegheh, D. Holcomb *et al.*, "WattsUpDoc: Power Side Channels to Nonintrusively Discover Untargeted Malware on Embedded Medical Devices." in *HealthTech*, 2013.

[26] 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. ACM*, 2017, pp. 333–346.

[27] R. Callan, F. Behrang, A. G. 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*, 2016, pp. 401–412.

[28] T. Mudge, "Power: a first-class architectural design constraint," *IEEE Transactions on Computer*, vol. 34, no. 4, pp. 52–58, April 2001.

[29] M. K. Gowan, L. L. Biro, and D. B. Jackson, "Power considerations in the design of the alpha 21264 microprocessor," in *Proceedings of the 35th annual Design Automation Conference*. ACM, 1998, pp. 726–731.

[30] R. Miftakhutdinov, E. Ebrahimi, and Y. N. Patt, "Predicting performance impact of DVFS for realistic memory systems," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2012, pp. 155–165.

[31] T. S. Karkhanis and J. E. Smith, "A First-Order Superscalar Processor Model," in *Proceedings of the 31st Annual International Symposium on Computer Architecture*, ser. ISCA '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 338–. [Online]. Available: http://dl.acm.org/citation.cfm?id=998680.1006729

[32] S. Eyerman and L. Eeckhout, "A counter architecture for online DVFS profitability estimation," *IEEE Transactions on Computers*, vol. 59, no. 11, pp. 1576–1583, 2010.

[33] Olimex, "A13-olinuxino-micro user manual," https://www.olimex.com/Products/OLinuXino/A13/A13-OLinuXino-MICRO/open-source-hardware, accessed March 2018.

[34] 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," 2005.

[35] Hynix, "2Gb DDR3 SDRAM Lead-Free and Halogen-Free (RoHS Compliant) H5TQ2G63BFR," http://www.farnell.com/datasheets/1382727.pdf?_ga=2.38941163.99566001.1522095243-708330394.1522095243, accessed March 20, 2018, "Rev 0.5".

[36] Samsung, "Samsung galaxy centura sch-s738c user manual with specs," https://www.cnet.com/products/samsung-galaxy-centura-sch-s738c-3g-4-gb-cdma-smartphone/specs/, accessed March 26, 2018.

[37] Alcatel, "Alcatel ideal / streak specifications," https://www.devicespecifications.com/en/model/e2ef3d31, accessed March 26, 2018.

[38] Keysight, "N9020a mxa spectrum analyzer," https://www.keysight.com/en/pdxx202266-pn-N9020A/mxa-signal-analyzer-10-hz-to-265-ghz?cc=US&lc=eng, accessed March 26, 2018.

[39] J. L. Henning, "SPEC CPU2000: Measuring CPU performance in the new millennium," *IEEE Transactions on Computer*, vol. 33, no. 7, pp. 28–35, 2000.

[40] ThinkRF, "Rtsa v3 real-time spectrum analyzer," https://s3.amazonaws.com/ThinkRF/Documents/ThinkRF+RTSAv3+TDS+74-0034.pdf, accessed March 26, 2018.

[41] Signatec, "Px14400a 400 ms/s, 14 bit, ac coupled, 2 channel, xilinx virtex-5 fpga, pcie x8, high speed digitizer board," http://www.signatec.com/products/daq/high-speed-fpga-pcie-digitizer-board-px14400.html, accessed March 26, 2018.

[42] J. Dean, J. E. Hicks, C. A. Waldspurger, W. E. Weihl, and G. Chrysos, "Profileme: Hardware support for instruction-level profiling on out-of-order processors," in *Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture*. IEEE Computer Society, 1997, pp. 292–302.

[43] J. M. Anderson, L. M. Berc, J. Dean, S. Ghemawat, M. R. Henzinger, S.-T. A. Leung, R. L. Sites, M. T. Vandevoorde, C. A. Waldspurger, and W. E. Weihl, "Continuous profiling: Where have all the cycles gone?" in *ACM SIGOPS Operating Systems Review*, vol. 31, no. 5. ACM, 1997, pp. 1–14.

[44] ARM, "Arm performance monitor unit," http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0388f/Bcgddibf.html, accessed April 6, 2016.

[45] S. Eranian, "Perfmon2: a flexible performance monitoring interface for linux," in *Proc. of the 2006 Ottawa Linux Symposium*, 2006, pp. 269–288.

[46] V. M. Weaver, "Linux perf_event features and overhead," in *The 2nd International Workshop on Performance Analysis of Workload Optimized Systems, FastPath*, vol. 13, 2013.

[47] Intel-Corporation, "Intel VTUNE Amplifier," https://software.intel.com/en-us/intel-vtune-amplifier-xe/details, accessed March, 2018.

[48] P. J. Drongowski, A. C. Team, and B. D. Center, "An introduction to analysis and optimization with amd codeanalyst performance analyzer," *Advanced Micro Devices, Inc*, pp. 1–20, 2008.

[49] Intel-Corporation, "Precise-event-based-sampling on intel processors," http://www.intel.com/content/dam/doc/manual/64-ia-32-architectures-optimization-manual.pdf, accessed March, 2018.

[50] T. Willhalm, R. Dementiev, and P. Fay, "Intel performance counter monitor-a better way to measure cpu utilization," 2012. [Online]. Available: https://software.intel.com/en-us/articles/intel-performance-counter-monitor

[51] T. Mytkowicz, A. Diwan, M. Hauswirth, and P. F. Sweeney, "Understanding measurement perturbation in trace-based data," in *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*. IEEE, 2007, pp. 1–6.

[52] J.-P. Lehr, "Counting performance: hardware performance counter and compiler instrumentation," *Informatik 2016*, 2016.

[53] Keysight Technologies, "Data-sheet: Probing Solutions for Logic Analyzers," http://literature.cdn.keysight.com/litweb/pdf/5968-4632E.pdf, accessed March, 2018.

[54] C. B. Zilles and G. S. Sohi, "A programmable co-processor for profiling," in *High-Performance Computer Architecture, 2001. HPCA. The Seventh International Symposium on*. IEEE, 2001, pp. 241–252.