EMI Architectural Model and Core Hopping

Daphne I. Gorman
gorman@ucsc.edu
Dept. of Computer Science and Engineering
Univeristy of California, Santa Cruz
Santa Cruz, California

Rafael Trapani Possignolo
rpossign@ucsc.edu
Dept. of Computer Science and Engineering
Univeristy of California, Santa Cruz
Santa Cruz, California

Jose Renau
renau@ucsc.edu
Dept. of Computer Science and Engineering
Univeristy of California, Santa Cruz
Santa Cruz, California

ABSTRACT

Processors radiate electromagnetic interference (EMI), which affects wireless communication technologies. However, despite the fact that the EMI generated by a processor is deterministic, architecturally modeling the EMI has proven to be a complex challenge. Moreover, EMI depends on the physical layout of the processor and on the binary being executed (both the application and its compilation options).

This paper proposes Model for EMI on a SoC (MESC), the first architectural framework for modeling electromagnetic emissions from a core. MESC takes into account the layout and the switching activity of a process to model the expected EMI. We validate MESC on a real system to verify its accuracy. We then use MESC to demonstrate that two different core layouts can be leveraged to reduce EMI and propose EMI Core Hopper (EMI CHopper). EMI CHopper uses a multi-core system – where each core has the same RTL but minimally different layouts – and proposes hopping the application between cores to reduce in-band EMI when it interferes with wireless communication.

Our evaluation shows that MESC is able to predict EMI within 95% accuracy across time and across the frequency spectrum, even when using statistical sampling to obtain activity rates. Leveraging MESC, our proposed EMI CHopper reduces in-band EMI by up to 50%, with low impact on performance. MESC will enable a new stream of micro-architectural research the same way architectural level power models have enabled exploration of performance and power simulation.

CCS CONCEPTS

• **Computing methodologies** → **Model development and analysis**; **Simulation evaluation**; • **Computer systems organization** → *Multicore architectures.*

KEYWORDS

Electromagnetic interference, EMI Model, System Simulation, Thread Migration, Dynamic EMI reduction

MICRO-52, October 12–16, 2019, Columbus, OH, USA

ACM ISBN 978-1-4503-6938-1/19/10…$15.00
https://doi.org/10.1145/3352460.3358289

ACM Reference Format:
Daphne I. Gorman, Rafael Trapani Possignolo, and Jose Renau. 2019. EMI Architectural Model and Core Hopping. In *The 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-52), October 12–16, 2019, Columbus, OH, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3352460.3358289

1 INTRODUCTION

The advances in integrated circuits and devices in the last few years have brought to market a growing number of connected devices that are getting smaller each generation. To meet commercial and technological goals, the space being allocated to electronic components is only a fraction of the already small device size, which brings the requirements of tighter integration. However, a processor in a SoC can produce electromagnetic interference (EMI). Since EMI degrades with distance, increasingly integrated devices are more prone to wireless communication disruption or quality degradation, due to EMI emitted by the SoC.

Wireless communication channel capacity, or the upper bound on the bit rate, is governed by Signal to Noise Ratio (SNR). A SNR change of as low as 5 (or a 5 dB difference in noise for a constant signal strength) can be the difference between two "bars" of cell service to none. Moreover, although the EMI produced by a processor is deterministic, it is also "very unpredictable." Thus, the traditional approach in industry to reduce EMI from processors to communication is static, not dynamic such as using shielding that tries to isolate the emissions from the processor in all frequency bands. This approach has proven successful at certain device sizes, but it loses effectiveness as devices shrink, since the shielding consumes space.[1]

Micro-architects are uniquely positioned to solve the problem of EMI in a dynamic fashion, either through higher-level techniques, like code manipulations that will change activity rates on the SoC, or though low-level circuit techniques that change the wires emitting EMI. Recently, a dynamic in-band EMI reduction technique has been proposed [9], however, it requires measuring EMI during runtime and can only address EMI in a reactive way. This is mainly due to a distinct lack of tools or methodology for modeling and ultimately controlling EMI.

Despite the fact that EMI is deterministic, there are no usable tools or methodology for modeling EMI, in particular at the micro-architecture level. This is because, despite its consistency, there are numerous factors that contribute to the EMI a processor produces. EMI can be theoretically modeled as a combination of electromagnetic radiation being produced by alternating current in each wire

[1] From industry sources.

of a processor. In any modern design, the complexity both in number of wires and in the data patterns transmitted across them do not make this prediction an easy task. The use of a full-wave simulation software [2, 10, 21] would provide accurate results, but takes an extremely long time, as the processor's layout would have to be taken into account as well as the activity across each wire. Also, because for this type of simulation, GDSII is necessary, these tools are unusable until just before tapeout, which may be too late for significant design changes. Moreover, the extreme accuracy provided by these full-wave simulators is most likely unnecessary for addressing the problem of processor-generated interference on wireless technologies, since compensation mechanisms, such as redundancy and error-correcting codes, already exist for wireless communication and provide a threshold that allows for some small amounts of noise. Here, we advocate for a lightweight model that can be used early in the design cycle or in hardware to quickly estimate EMI in different frequencies.

Furthermore, the EMI is application/binary dependent, so the same layout will produce different EMI depending on the program executing. This adds additional challenges for designers, as it is difficult to account for the multitudinous, essentially infinite, applications a processor may execute. Then, modeling the EMI will necessarily imply characterization through specific benchmarks that can represent the EMI of a large range of applications. Thus, any model for EMI must be flexible enough to account for different processes. Therefore, previous works suggest simply monitoring the EMI for a runtime solution rather than attempting to create a model [9]. While, simply relying on run-time measurements is applicable in some use-cases, there is need for a design time technique that allows for estimating EMI, even if at lower accuracy.

In this paper, we first propose MESC, Model for EMI on an SoC, the first architectural level EMI modeling framework. MESC takes into account the activity rate of individual wires in a process and the layout to approximate the expected EMI from that process. This is in distinct contrast to previous works that utilize measured EMI to profile programs [5, 19] or to isolate the on-chip location of magnetic field sources [25], as we aim to model the EMI from the layout, not the other way around. MESC uses some basic initial power measurements of a device as well as statistical sampling of switching activity in order to model the expected EMI of a processor. MESC is a simpler model than full-wave simulators, with a more convenient runtime in order to allow design space exploration. Since we only care about the longest wires in the design, we can get estimates during the floorplanning phase, and thus much earlier than GDSII. This would also enable place and route tools to be aware of EMI based on early estimates using MESC, as opposed to needing to fully place and route a design before being able to get EMI results. MESC gets statistical samples of activity rates in the relevant (longest) wires. Activity rates can come from cycle accurate or RTL simulation, FPGA emulation, or it can be estimated on the real system.

To evaluate MESC, we compared MESC estimations with EMI emitted by a soft core running on an FPGA. The EMI was measured with a spectrum analyzer. We then compare the expected EMI profile across different frequencies with the measured EMI. Even by sampling activity rate and only considering the few hundred longest wires, MESC is able to predict with an average of less than 1% error. We note that the frequencies at which we observe peaks of EMI are correctly predicted by MESC and the main source of error is with relation to the magnitude of the EMI. This is particularly important since predicting the frequency bands at which interference occurs is more important than predicting the exact magnitude of the EMI.

Then, we leverage MESC to model the EMI of a core with different layouts, and we propose EMI Core Hopper (EMI CHopper), a technique to reduce EMI by "hopping" between cores with the same RTL but different layouts. EMI CHopper uses the expected EMI provided by MESC in order to determine which of the different core layouts would produce less in-band EMI, and may switch a process to the other core depending on the processing and EMI tradeoffs. Although core hopping, or thread migration, is a known technique, this is the first paper to propose its use to dynamically reduce EMI. We show that EMI CHopper is able to reduce the in-band EMI by 50% power on average, with very low impact on performance. The instantaneous reduction can be of up to 10dB, or about 10x power reduction in EMI from the core.

The contributions of this paper include:

- MESC: the first micro-architectural model for EMI of a process running on a processor,
- Validation of MESC with FPGA measurements of a processor's EMI, and
- EMI CHopper: a technique for reducing in-band EMI via core hopping.

The remainder of this paper is structured as follows: Section 2 provides some background on the electromagnetic principles that we utilized while creating MESC before covering some related work in Section 3. Section 4 describes the setup we used to perform our measurements. From there, Section 5 provides the steps we took to creating MESC as well as the flows for MESC and EMI CHopper before evaluating our flows in Section 6. Finally, we conclude in Section 7.

2 BACKGROUND

This section covers background information about EMI, how it is produced in a core, and other related information that we used to develop MESC. We put special focus on the fact that communication is not using all the frequency spectrum at a given time, but rather uses a single band. Thus, a processor does not need to minimize the EMI over the entire frequency spectrum, but rather just the frequencies being used for communication at that time. Throughout this work, we will rely on this fact.

2.1 Preliminaries

From a Radio Frequency (RF) perspective, any wire with a time dependent current passing through it behaves as an antenna. Traditionally, in integrated circuit design, the main concern regarding antenna effects is to treat wires as antenna receivers and thus potentially being subject to bit-flips within a wire, which can cause data corruption and/or invalid behavior in the circuit. In this work, we do the opposite and look into wires as transmitters because we are interested in the EMI being emitted by each wire and how it negatively interacts with wireless communications.

In general, the power and direction of the radiation depend on the form of an antenna and on the distance from which the

interference is being measured. For instance, for very long wires ($L \gg d$, where L is the wire length and d is the distance of interest) the radiation occurs uniformly throughout the wire axis, varying only with distance. In this particular case, edge effects are usually ignored. Another example is with closed loop antennas, where the radiation is directional and perpendicular to the loop plane. In the specific case of interest in this paper, L is a length within a die and d is a distance within a device, thus usually $d \gg L$. Also, there is a large number of wires that will act as an antenna array, possibly emitting at multiple frequencies with different magnitudes. Therefore, the resulting EMI will be a combination of EMI emitted by each wire.

There are a few electromagnetic theory equations that are relevant to this work, but the most relevant is the basic equation for antenna gain, for a given wire acting as an antenna emitter,

$$G = \frac{4\pi A}{\lambda^2}. \quad (1)$$

This equation states that the gain of the antenna (G), or power multiplier, is directly proportional to the effective area A, which depends both on the actual antenna area and the relative angle between the current direction and the emission direction of interest. In other words, the emission of a CPU wire will be proportional to its length and width.

Another important formula, known as Friis Transmission Formula [18],

$$P_R = \frac{P_T G_T G_R \lambda^2}{(4\pi d)^2} = \frac{P_T G_T G_R c^2}{(4\pi d f)^2}, \quad (2)$$

states that the received power P_R is the product of the transmitted power P_T multiplied by the transmit (TX) and receive (RX) gains G_T and G_R multiplied by the wavelength squared (or multiplied by c^2 and divided by the frequency squared f^2) all divided by 4π times the distance between the two antennas d. This means that the received power is proportional to λ^2 or inversely proportional to f^2. In decibels, this formula is

$$P_R = P_T + G_T + G_R + 20\log_{10}(\frac{\lambda}{4\pi d}). \quad (3)$$

This "shielding" effect is directly observed in our experiments and modeled in MESC.

EMI is typically depicted as the amount of power being radiated at each frequency and thus most of the plots in this paper are represented as such. We also utilize the Fast Fourier Transform (FFT) to convert samples taken over time (in the time domain) into the frequency domain. One important aspect of taking an FFT over a set of time-domain values is that the FFT can only provide values for frequencies up to half of the sampling frequency known as the Nyquist frequency. Therefore, to get the frequency domain up to a frequency f, we need to sample at $2f$. As the primary objective in this work is to model and then reduce in-band EMI, using the frequency domain is a good way to visualize and isolate specific frequencies, as opposed to trying to determine frequencies from periodicity in time-domain plots.

2.2 Layout and Metal

To apply these RF principles to a processor, we treat metal wires in the processor as "antenna elements" or "radiators" that emit EMI.

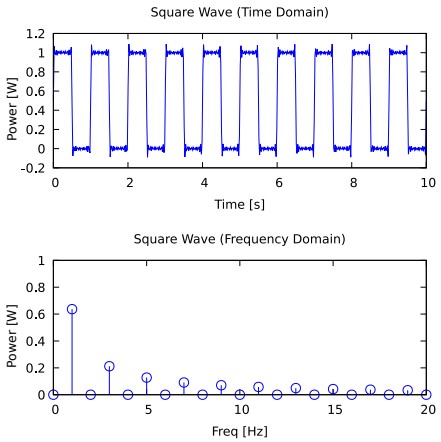

Figure 1: A 1 Hz square wave in the time domain (top) and the frequency domain (bottom). A square wave is constructed from a sine wave and its odd harmonics.

As noted in Section 2.1, a fundamental rule is that the radiated power of an antenna is proportional to the antenna aperture, or the effective area. This means that if an antenna has twice as much metal, we would expect twice as much power to be radiated. Applied to processor design, this means that a net that is twice as long or twice as wide should contribute twice as much EMI. Furthermore, long wires may add inductance, which could increase the EMI.

Additionally, the EMI power is proportional to the integral of the current distribution along the antenna. Basically, this means that areas that have higher current will radiate higher power EMI. If a circuit contains a loop, that loop may couple with an external magnetic field and act as a strong transmitting antenna, but this effect can be minimized by reducing the size of the loop. However, loops are not widespread in chip designs due to inductive effects that are usually undesirable.

2.3 Square Waves

Signals propagating through processor wires are not perfect sinusoids and resemble mostly square waves with slew. Square waves produce strong odd harmonics and weaker even harmonics. That is, for a square wave with a frequency of f we would expect large amounts of power at f, $3f$, $5f$, $7f$, etc.. However, at the even harmonics $2f$, $4f$, $6f$, we would expect some radiated power, but significantly less than at the odd harmonics. Figure 1 shows the wave created by adding the first ten odd harmonics of a square wave as in the equation $0.5 + \frac{2}{\pi}\sum_{k=1}^{10} sin(2\pi(2k-1)t)/(2k-1)$, which produces a square wave with a frequency of 1 Hz and an amplitude of 1. Below that is the same square wave depicted in the frequency domain, which shows the odd harmonics to have significant power, but the even harmonics to be zero.

Another common occurrence are power spikes that occur at subharmonics ($\frac{f}{2}$, $\frac{f}{3}$, $\frac{f}{4}$, etc.). Potentially, we may even observe significant power at the harmonics of the subharmonics, which would could potentially put power spikes at unexpected frequencies.

An important observation is that in most data buses in a processor, the data is not truly periodic, that means that there is not

a single frequency being produced over time. For instance, a bit of data can have a non-periodic pattern of values over time, like "00100011101" (where each bit corresponds to the value at a given clock cycle). This is considered by modeling the data pattern as a temporal wave and applying Fourier Transformation to get the appropriated magnitudes at specific frequencies.

In order to design a realistic model, we must take into account all of these complexities. In our model, we start from a cycle-accurate execution trace of the wires in the processor and assume it is a relatively square wave to perform a Fourier Transformation, which allows us to determine which frequencies are being affected by a specific data pattern. This will be addressed in more detail in Section 5.

3 RELATED WORK

This section provides descriptions of some of the current research on EMI generated by processors. This section briefly covers topics from wireless communication, profiling, and security.

The most closely related work was published by Gorman *et al.* introducing the impact EMI has on in-band wireless communications [9]. The authors show that small changes in architectural and code parameters can significantly impact the EMI emitted by a processor. More importantly, they have shown that two processors with the same RTL can produce significantly different EMI, even when executing the same process. That specific finding is the foundation of this paper.

3.1 Profiling

Recently, there has been a push towards using EMI to profile code. Callan *et al.* [5] proposed ZOP: a zero-overhead approach to obtain profiling information via EMI measurements. ZOP first goes through a training phase in which it builds a model that associates different wave forms with different parts of the code. Using that model, ZOP can then go through its profiling phase, which monitors the EMI and can match the current EMI with what part of the code is being executed at that time.

Sehatbakhsh *et al.* [19] were able to show a correlation between frequency spikes in EMI and the amount of time a loop takes to execute, which allows for zero-overhead spectral profiling. On similar lines, EM has been shown to correlate well with $\frac{di}{dt}$ voltage noise in post-silicon setups [11]. This correlation is used to predict voltage noise stress tests driven by EM amplitude and obtain resonance frequency of the power delivery LC-tank of a chip.

All profiling based on EM radiated from a chip have minimal overheads and thus minimally disrupt the workload being characterized. For this paper, the most relevant fact is that small differences in the code being executed and in the silicon can cause non-trivial changes to EM radiated from the core.

3.2 Security

There has been substantial research on EMI and security, and many tools have been published that address the EMI security side channel. One notable publication proposes FASE, which finds periodic signals dependent on processor or memory activity [7]. Another tool, SAVAT determines the impact an instruction has on the EMI generated by running an instruction [6].

As opposed to the publications that show how vulnerable processors are to side channel attacks, some publications use EMI to discern whether or not a process has been modified. For example, EDDIE detects code injections without introducing any overheads or changing the hardware or software [17].

3.3 Other EMI Models

Wang *et al.* have published a measurement procedure that searches for amplitude-modulated EMI [24]. The algorithm utilizes FASE and SAVAT in order to determine which circuits in a processor are most susceptible to EMI side channel attacks.

Werner *et al.* have proposed a method for determining the instruction-dependent magnetic field sources [25]. Their model determines the locations of the magnetic field sources at a single frequency. In fact, the user is encouraged to choose a "less noisy" frequency for this model. This is different from MESC, which models the EMI over multiple frequencies from the processor, instead of modeling the processor from the EMI.

3.4 VLSI and System-Wide Tools

In general, not many tools for modeling or minimizing the EMI from a processor exist, even for a system-wide or VLSI perspective. Some PCB simulation tools contain rule checking for EMI, such as Hyperlynx [10], but that is only to comply with Federal Communications Commission (FCC) approval. Also, there is considerable difference in the level of complexity of modern PCBs and chips, with chips possessing orders of magnitude more wires and elements than PCBs.

Although not strictly EMI, EmerGPU [22] detects and mitigates resonance voltage noise (which causes EM noise) in GPUs. In order to reduce voltage noise, some techniques include reducing the slope of current changes via hardware or software mechanisms.

3.4.1 Interference from the clock. There has been a lot of work done on minimizing the interference from the clock [12–15]. As clock speeds tend to be lower than the frequencies used by wireless communications, it is the clock harmonics that have an adverse affect on signal. Therefore, it is common for chips to have modulated clock signals, slightly changing the cycle time of the clock every cycle. By modulating the clock signal, the attenuation for the harmonics is significantly increased, and therefore the harmonics at the communication frequencies are much lower than for an unmodulated clock signal.

3.5 Thread Migration/Core Hopping

Thread migration, or "core hopping" has been a topic of research for years, with applications ranging from power and leakage optimizations to performance improvements, but we are the first to propose utilizing this technique for reducing in-band EMI.

Kumar *et al.* [16] propose to use heterogeneous cores that implement the same Instruction Set Architecture (ISA). A thread can be migrated from a core to another to reduce power. The authors report a reduction in power of 39% with a 3% reduction in performance only. To enable this type of migration, a Heterogeneity-Aware scheduler [20] is also proposed. The schedule bases its decision on a signature that estimates the performance of a thread in each core. The same type of schedules could be adapted to be

used by EMI CHopper, however, in this paper we focus purely on reducing the in-band EMI, and not to combine this with performance metrics. Thus, the only metric used to determine whether to migrate a thread is the estimated EMI of the thread in each core.

A similar approach is evaluated in Alpha cores [4], where only two core sizes are used. One important lesson is that, although there may be advantages of migrating the threads, it is necessary to take into account the overhead of switching the context from one core to the other.

PIE [23] uses a more elaborated performance predictor to decide whether and to which cores threads should be migrated. It collects CPI stack, MLP and ILP profile information to estimate performance in each core. The recommended approach is to use dynamic scheduling, where migration overhead needs to be taken into account. The paper also shows that there is an impact on whether or not a shared Last Level Cache (LLC) is present or not. A shared LLC can reduce the performance overhead of migration of cache to less than 2%.

4 SETUP

Before we go into the model and proposals of our paper, we discuss the measurement setup, some challenges associated with taking measurements, and how they were handled throughout the construction of the model.

All EMI measurements were taken using a N9342C handheld spectrum analyzer with a Near-Field probe set from Keysight Technologies. Since our evaluation relied on different designs and layouts, we decided to take measurements on an FPGA to avoid taping-out chips. For the validation of MESC, we used a combination of simple kernels on FPGAs and a soft-core OpenPiton [3] core running Spec06 benchmarks. All our measurements were taken from a Digilent Genesys 2 Kintex-7 FPGA Development Board and synthesis, placement and routing were done using Vivado 2017.4.

Since EMI CHopper requires multiple different layouts, we tweaked some placement optimization parameters on Vivado to generate two slightly different layouts. There is no logic difference between the versions of the core. The run frequency of OpenPiton in the FPGA is 66.6MHz, and for simpler experiments without full cores, the clock frequency is variable and mentioned in the evaluation on a per-experiment basis. In this work we are unable to use real communication frequencies due to the constraints of running processes on an FPGA core instead of an ASIC processor, but this work could be trivially extended without any losses to higher frequencies and to ASICs.

4.1 Setup Verification

Our first task was to ensure the RF principles described in Section 2 were applicable to our setup. We had some concerns that applying those principles to an FPGA may yield some inconsistent results, as "wires" on an FPGA may be broken up with buffers and routing resources such as crossbars and multiplexers. Additionally, FPGAs also contain a considerable number of additional electrical components, such as logic for clock generation, memories, and IO connects that could be active even if they are not used. Thus, we started with a series of smaller tests to see how implementing circuitry on an FPGA compared with our expectations derived from theory.

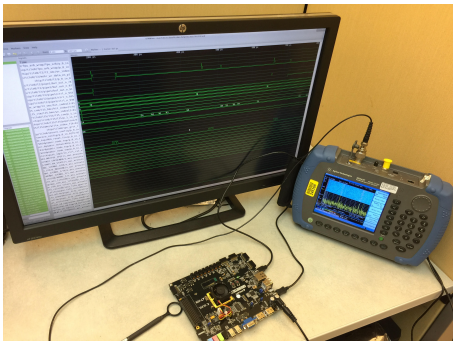

Figure 2: Depicts the measurement setup used in this work.

Our first tests consisted of creating buses that varied in length, number of bits, and shape before transmitting clock signals at multiple different frequencies across them. As expected, buses that were twice as long or twice as wide emitted twice the amount of power at the signal frequency.

Once we determined that the buses radiated EMI as expected, we implemented a debug core in our HDL design suite that monitored the activity of each net on the FPGA. From the switching activity of each net as one might find in a Value Change Dump (VCD) file, we were able to perform a discrete FFT of the values of each net at every clock cycle in order to obtain the signal's distribution in the frequency domain.

Using these simple tests, we were able to monitor the activity of all the nets, but moving forward, we knew that monitoring every net on a processor would not be feasible.

5 MODELS

This section details the design process and building blocks for MESC. From there, we present the MESC algorithm to determine the EMI profile being emitted by a circuit. Then, we will discuss the use of MESC to build EMI CHopper, a dynamic scheduling tool that minimizes EMI with minimal impact on performance and discuss possible other usages of MESC.

5.1 MESC Flow

This section provides details on each step used by MESC to model the expected EMI. The modeling presented here is based on both theoretical understanding of EMI and experimental validation of it, therefore this section is organized as a set of techniques with corresponding validation where needed. The EMI produced by a net depends on the net length and waveform present in the net. In theory, the shape of the net would also have an influence on EMI, but our results show that in chips, shape has a lower impact on the overall EMI produced. Moreover, considering activity in all the nets in a core all the time would generate a lot of overhead, both in terms of hardware needed and in performance impact. However, we were able to determine that not all nets contribute equally to EMI.

5.1.1 Layout Profile. EMI emitted by a wire depends on both its length and width. For each net, we collect information on length and metal layer. Metal layer is important since different layers have different metal pitch, which alters the current (thicker wires have

lower resistance per length unit). Wire width is considered as a multiplying factor to the length. For multilayer wires, each segment must be considered individually.

The total radiated power expected to be proportional to the square of the wire length ($P_{total} \propto L^2$), but in digital design, long wires are typically buffered for the purposes of improving electric propagation [1]. Therefore, each net can be considered to be multiple radiators of equal dimensions instead of one single long wire. Thus, in MESC, each net has a single weight, the product of its length by its width.

We call the combination of wire weights of a design a "layout profile." In the case where the circuit is not fully placed and route, the layout profile of a design can be estimated based on floorplan of major blocks. Estimations of metal layer can be based on whether the wire is local (within a block) or global (between blocks).

5.1.2 Sampling Strategy. Because monitoring every net on a processor would be impractical, we only monitor a subset of the wires in the core. We chose which nets to monitor based on the its weight (length times width): after excluding reset and clock nets, we choose the nets with larger weights until at least 80% of all the metal in the processor was being probed. Resets are excluded since they don't have periodic activity during the execution of a program and clocks are excluded because they are modulated using spread-spectrum techniques, which mean that clock cycles are slightly different from one another, and thus there is no single frequency for clock. This is confirmed by our experiments and observed in prior work [9].

Additionally, in the interest of minimizing the amount of data that is necessary, we decided to take samples of the switching activity as well. Basically, this means that at every period of time, the data in each wire of interest will be collected for a certain number of cycles. As shown in prior work [5, 9], the EMI changes as processors execute different phases of a process. For the purposes of this work, determining which execution phase is being processed is sufficient.

In our evaluation, we show that 1024 samples every 0.1 to 1s per monitored net and $\approx$ 600 nets are enough to get accurate EMI estimate. These figures can be be reduced at the expense of accuracy. The actual logistics of acquiring the samples will depend on how MESC is being used. For pre-silicon modeling, RTL or cycle-accurate simulation can be used, whereas for application in silicon, built-in memories can be used to keep values of interest until use. In our evaluation, we collected samples from a soft-core running on FPGA through FPGA instrumentation. Another important aspect is that for different cores, in particular for high-end cores, the total number of nets could be different than the numbers used for out evaluation. However, we do not expect the number to increase drastically, since even in high-end cores, most of the total metal is dedicated to a relatively small number of nets.

5.1.3 Converting to the Frequency Domain. After retrieving the switching activities, we perform an FFT on the activity of each net, converting these time slices into the frequency domain. When putting together the MESC EMI for the entire processor, we multiply by the net weight before adding the EMI profiles for all the nets. This yields a total power over frequency from 0 to half of the clock frequency.

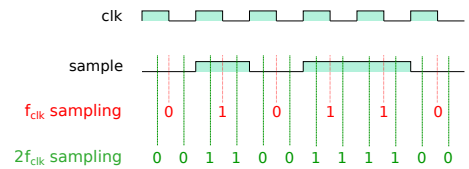

Figure 3: The data in a bus remains constant over a single clock cycle so MESC can artificially increase the sampling frequency, thus increasing the accuracy of the model.

However, only being able to obtain the frequency domain response until up to half of the clock frequency is insufficient for our model. Therefore, to implement MESC, before taking the FFT, we repeat the value of each clock cycle to artificially increase the sampling frequency. Since signals are roughly square, the value should remain unchanged throughout each clock period. For example, a switching activity sample containing the values '0 1 0 1 1 0' with a clock frequency of 1 MHz would only yield a Fourier transform that corresponds to values up to 0.5 MHz. However, if we inject an extra sample per clock cycle, we can use the values '0 0 1 1 0 0 1 1 1 1 0 0' with a sampling frequency of 2 MHz, and we are able to get values in the frequency domain up to 1 MHz. This is shown in Figure 3. To increase precision, slew rate may also be considered and capacitance in the wire may also be considered. However, in our experiments, we did not find that to be necessary to get accuracy.

For this implementation of MESC, we used 8 samples per clock cycle, as after investigation, we found that injecting more samples yielded a small increases in accuracy for a large increase in compute time. For a hardware implementation of MESC, this would also increase the size of the FFT used.

5.1.4 Adding Harmonics. Once the samples have been converted to the frequency domain, MESC must compensate for each signal's harmonics. Remember from Section 2 that square waves are comprised of a signal and its odd harmonics. Thus, MESC adds the odd harmonics of the signal to account up to the highest desired frequency for the model, dividing the power by the value of the harmonic. That is, for a signal at frequency f with strength p, the nth harmonic will be added at the frequency $f \cdot n$ with strength p/n, provided that n is odd.

However, because the FFT takes into account harmonics, this is only necessary for frequencies above half the artificially increased sampling frequency. Adding these higher frequencies increases the model's accuracy without adding a significant compilation time, as this is a linear operation and the FFT is $O(n \log n)$.

5.1.5 Artifacts and Shielding. Packaging, board and other off-chip components affect the observed EMI. Off-chip components may create additional spikes in frequencies that were not expected by simply modeling on-chip wires; we call these artifacts. Also, packaging, cooling and sockets may create an EMI filtering at certain frequencies, and we call this effect shielding.

In our model, we consider both artifacts and shielding. First, to detect artifacts, the chip must be on without processing anything. In the our experiments with an FPGA, we were able to observe that these artifacts existed even when the FPGA had been powered

on, but not yet been programmed. Figure 4 shows an example of artifacts that were measured on the FPGA used in this paper. Then, the FPGA was programmed with a simple bus, that was run at different frequencies (40 MHz and 50 MHz), and as expected, an EMI spike is observed in those frequencies. However, there is a consistent artifact at 60 MHz. We can therefore conclude that this spike does not come from the design implemented, but rather from some other component, either on the FPGA itself or on the board. The same type of artifact is expected on any system, but possibly at different frequency and magnitude.

The process of detecting artifacts is static and can be done once in the lifetime of a platform. The process involves finding EMI peaks that are consistent regardless of the application running.

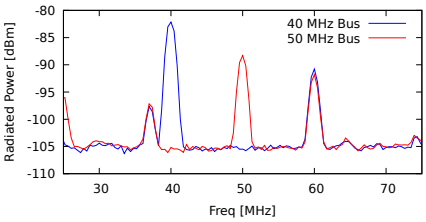

Figure 4: To detect artifacts, we run the FPGA at different frequencies. The plot shows the measurements artifacts at 46 and 60 MHz for our FPGA setup when running a simple 40 or 50 MHz Bus.

Additionally, as we are unable to measure the EMI of the processor directly, we must take into account any shielding created by the housing (packaging, cooling, board, ...). We are able to determine this by measuring the power emitted from a single bus with a consistent width and length carrying varying clock frequencies. With the peak power at each frequency, we are able to construct a linear function over frequency that must be taken into account when measuring the EMI. Figure 5 shows the EMI emitted by a single bus, designed in Vivado for this experiment, with consistent length and width switching at different frequencies. As shown, there is a significant difference in power as the frequency changes. However, we were able to determine an approximate linear function, with the slope of $20\log_{10}(f)$, which we would expect after an inspection of the logarithmic Friis Transmission Formula (Equation 3).

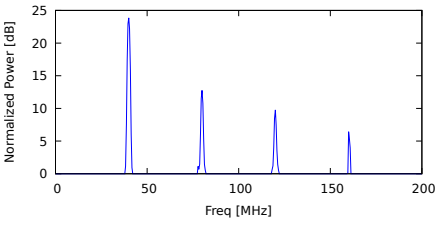

Figure 5: Interference of a single bus toggling at 40, 80, 120, and 160 MHz. From this, we can infer that our measurement setup is not without some shielding as a function of frequency, which must be compensated for in MESC for accuracy.

5.1.6 Other physical factors. There are other factors that affect EMI. In general, any parameter that can affect the current will also affect EMI as a result. In this subsection, we discuss how some parameters are expected to affect EMI. Incorporating these parameters in MESC is left as future work, since they would require an ASIC tapeout.

Another important issue that may arise in particular for ASICs is slew and load. Slew is the rate at which a signal changes in the beginning of a cycle. A signal that has high slew or signals with high load will be "less square." This will mostly affect harmonics of the main signal frequency. Since harmonics have lower amplitude, the error due to non-square waves is expected to be small. In cases where more accuracy is needed, this can be considered by calculating the load in each of the wires of interest.

Dynamic voltage and frequency scaling can be considered by adapting the amplitude (voltage scaling) and/or frequency spectrum (frequency scaling). Although typically considered to not switch (from a digital perspective), the power delivery network (PDN) carries current signals that are frequency-rich. These effects were not observed in our FPGA-based evaluation or in prior investigations with ASICs [9], but could potentially become a source of refinement. Fully incorporating these into MESC is left as future work.

5.1.7 MESC Final Flow. The final MESC flow consists of collecting execution traces for the sampled wires at sampled intervals, calculating the FFT of the traces in each wire, adding harmonics, multiplying by wire length and combining all the wires FFTs into a single FFT. The artifacts and shielding are added to the final FFT. Figure 6 depicts the flow.

5.1.8 Validating MESC in an ASICs setup. Most of the validation work in this paper focused on FPGAs, since they allow us to control wire length, number and shape of wires, and activity factors. An extended validation on ASICs would require a chip tapeout with controlled conditions. In this subsection, we discuss some of the parameters that should be explored.

A few physical factors impact EMI:

- Alternating current through wires and its amplitude
- Wire area and shape
- Electrical resistance in a wire

From circuit design, we know that these will be affected by wire length (same as for FPGAs), wire width (or metal layer) and driver strength. Some additional factors, such as vias and bends, could be considered for more precision, but since they are only known after routing and most likely have a lesser impact on the overall resistance for long wires, they can be ignored at least for initial approximations.

In general, there are two conflicting effects related to wire length: 1) increasing length will increase resistance, thus reducing the current magnitude, which in turn reduces EMI; 2) simply increasing wire length for a constant current magnitude will increase EMI. In ASICs, long wires are fragmented into multiple segments with dedicated drivers, so the increase in resistance due to increased length is less important for EMI magnitude, since the drivers will balance out the extra resistance. Thus, the profile for each wire can be represented by $l \cdot w \cdot s$, where l is the wire length, w is the wire width (fixed per metal layer) and s is the driver strength. For wires composed of multiple segments, segments must be added together (Σlws).

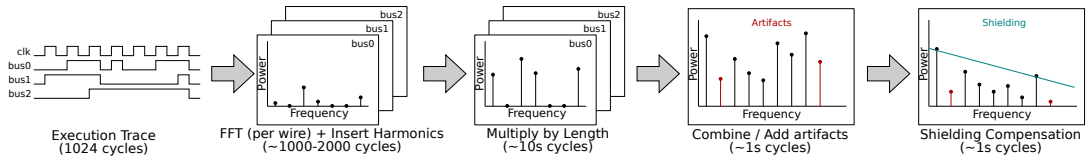

Figure 6: MESC consists of collecting execution traces for each of the sampled wires, calculating the FFT individually, and some further processing before combining the FFTs. The final EMI model is provided after inserting the artifacts and compensating for shielding effects.

5.2 EMI CHopper

Now that we can estimate the EMI emitted by a core, we want to use the estimate to reduce the EMI emitted in a specific communication band. The main objective is to clean bands that are being used for communication of interference. EMI CHopper proposes hopping from core to core in order to reduce in-band EMI. This is usually referred to as thread migration and has been proposed for performance and power [4, 16, 23], but this work is the first to propose it for EMI reduction. In this paper, we propose using two cores that are architecturally identical, *i.e.*, will yield the same performance and power, but that have different layouts. The difference in layout will affect how EMI is produced for a given process running in each core, which is the basis for our proposal.

EMI CHopper works as a dynamic scheduler that estimates EMI for each available core, using MESC, when wireless communication is being used. The layout profiles from MESC for each core are known at design time and are static. If EMI in the band of interest would be reduced by running the application in a different core, EMI CHopper may decide to migrate the thread. To prevent migrating for trivial reductions in EMI, EMI CHopper takes a threshold parameter, and thus the migration only occurs if the reduction in EMI is larger than the threshold. For practical applications, we expect that an OS scheduler will take other factors (besides EMI) into account for thread scheduling, however, this choice will be domain specific and thus is out of the scope of this paper.

5.2.1 Modeling thread migration cost. It is important to note that there is some execution overhead associated with migrating a thread [4, 8]. The thread migration cost may be elevated, in particular when considering rebuilding the state of the cache and branch predictor [4, 23]. There is a clear tradeoff between paying the cost of transferring cache, branch predictor, etc. between cores or going through a warm-up period, where these structures will be organically filled during execution. The most natural solution is to not transfer those states and let the execution take care of it, which is assumed throughout this paper and is the usual strategy already implemented in most multi-core systems.

The migration cost comes from various sources, ordered here in order of magnitude with the less impact first: 1) migrating the architectural state (PC, RF, ...); 2) Warming-up branch predictor tables, prefetcher state, ...; 3) Warming-up the cache state. Overall, a few clock cycles can be assumed to transfer the architectural state between cores [4]. For branch predictors, prefetcher and other predictor tables, it is harder to estimate the impact, but since their state is not transferred in our approach, it will show as a reduction in IPC, due to mispredictions. For caches, the impact can potentially be throughout a large number of cycles, since caches take much longer to warm-up [8]. However, by using a shared last level cache (LLC) even after migration, the penalty of misses will be largely reduced [23]. In fact, when using a shared LLC, even by migrating every 2.5ms, the performance impact is expected to be lower than 1%.

The impact on performance is dependent on the frequency at which the threads are migrated [8]. For wireless communication and EMI reduction, there is no need for fine grained thread migration, thus we do not expect frequent migrations. For EMI CHopper, we limit the migration to at most one thread migration per second, which should reduce any impacts on performance. In our evaluation, we show that even 1s migration is frequent enough for any communication purposes.

5.2.2 Implementing EMI CHopper in real hardware. EMI CHopper requires each core to keep track of the activity in the longest wires and then calculate FFT for each wire. This requires some dedicated hardware. Since keeping 1024 samples for 600 wires would necessitate using too much storage, EMI CHopper proposes taking samples of each wire at a time. Thus, EMI CHopper requires only 1024 bits of storage, pipelined FFT implementation and some arithmetic circuitry per core. For the parameters proposed for MESC, a FFT that takes $\approx 8k$ bits is needed.

Taking one EMI CHopper sample will thus take 600 wires $\times$1024 cycles $\approx 600K$ cycles. At $1GHz$, this means that is possible to calculate over $16k$ samples per second. This is more than enough for 1 hop every second proposed by EMI CHopper. However, more buffering and a larger number of FFT accelerators can be used to speed-up the process.

The overall implementation of EMI CHopper is illustrated in Figure 7. Each core provides EMI CHopper with the execution traces for the monitored wires. The FFT is performed individually per wire trace and then multiplied by wire length from the layout profile information. The aggregated data will guide the EMI CHopper decision to hop applications from one core to the other.

The use of multiple layouts is certainly a challenge in today's systems. Currently, there is a considerable amount of engineering effort put into generating a layout for a core. Multi-core systems typically use a copy of the same layout for all the cores. However, in our experiments, simple changes to placement optimization parameters were sufficient to change the EMI profile and the effort was very low. For real world applications, there may be important trade-offs between EMI management needs and design time effort.

Moreover, since EMI CHopper uses a hardware implementation of MESC, it needs to decide during design time which nets will be

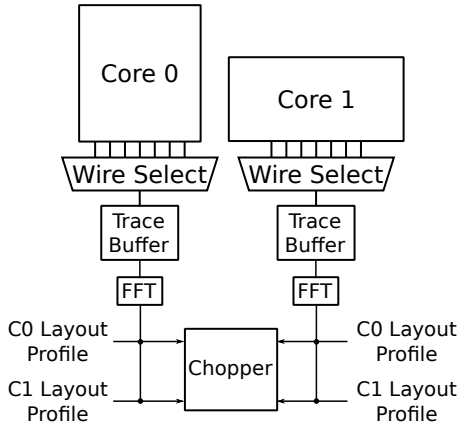

Figure 7: EMI CHopper requires a single FFT and arithmetic operations in addition to a small amount of storage to calculate MESC for each application in each core and decide whether or not to migrate threads from a core to the other.

monitored. This decision may limit the visibility that EMI CHopper has over the EMI produced. Another potential limitation comes from the limited number of layouts used. It is possible that none of the layouts in a system is able to yield low EMI for a specific application and for the band of interest. Even so, our evaluation with a small number of wires and only two layouts is able to largely reduce EMI.

5.3 Alternate MESC Application: Power vs EMI Tradeoff

We presented EMI CHopper as a use-case for MESC in Section 5.2. However, MESC enables architects to model EMI for other applications as well. Thus far, EMI has been only evaluated in architecture papers from measurements in real hardware. The possibility of modeling EMI without the necessity of physical, real hardware may allow EMI estimation to be included in higher level architectural exploration.

For instance, another application for MESC would be to determine the effect of increasing the clock frequency, but only sending the values over the bus every couple of cycles, which should change the EMI produced by the clock without changing the processor speed. One drawback would be that there would be an increase in power consumption as a tradeoff for manipulating the EMI.

MESC can also be adapted to use in a place and route ASIC flow. Since placement is one of the main factors that statically affects EMI, using EMI estimates in placement decision could have a large impact on reducing emissions.

6 EVALUATION

In this section, we start by validating MESC against real hardware measurements, then we evaluate EMI CHopper as a tool for EMI management in a processor.

6.1 MESC Validation

We start our evaluation by validating our MESC implementation. Figure 8 shows a comparison of the output of MESC vs the measured EMI from executing the namd benchmark on OpenPiton running on an FPGA. The model yields an estimated EMI that is quite accurate, with only 5 dB maximum difference between MESC and the measured output. It is reasonable to conclude from those results that MESC is able to accurately estimate the magnitude of the EMI and, most importantly, the frequency at which EMI peaks occur.

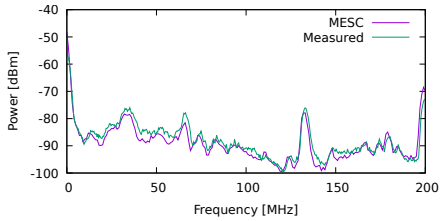

Figure 8: MESC of namd compared with the measured EMI of namd running with the same layout. The maximum difference between two is 10.

To go more into details on the accuracy and limitations of the MESC, we want to analyze other examples. The first point that we want to analyze is the effect of shielding in the model. The next set of results use a simple design with three buses, each running at different toggle rate. One switches at a rate of 200 MHz, one that switches at a rate of 57 MHz, and one that switches in a "random" 7-bit pattern that keeps repeating itself. These activities were chosen in order to ensure that the harmonics would not overlap. Figure 9 shows the measured and estimated EMI for the three buses.

As depicted in Figure 9, we started with small test cases to verify the accuracy of our model. One challenge for comparing measured EMI with our model was some of the inconsistencies in the amount of power being measured as the frequency changed. Figure 9 clearly shows that at the lower frequencies, the peaks tend to be a lower magnitude than the actual measured EMI, whereas the higher frequencies tend to match up more accurately.

Next, we evaluate the need for artifact insertion in the model. Figure 8 also has a peak at 135 MHz that did not originally appear in our model. Unfortunately, this deviation is unable to be detected by MESC as it is being produced consistently by the FPGA, even when the FPGA is not programmed. Thus, this peak is not being produced by the core that MESC is modeling, but rather by the device in which it is being implemented. Figure 10 shows the measured EMI when the FPGA is not programmed with any circuitry at all. As shown, there is a significant spike in EMI at 40, 60, 120, 135, 150, 170, 180, and 200 MHz. Therefore, MESC must add these consistent peaks produced by a device after calculating the EMI from the activity rates. Without inserting those artifacts into the final estimated EMI, there would be an extra source of error in MESC. For other applications, artifacts also need to be measured and considered when estimating EMI.

Once the artifacts and the shielding have both been integrated into MESC, we were able to run the set of SPEC2006 benchmarks

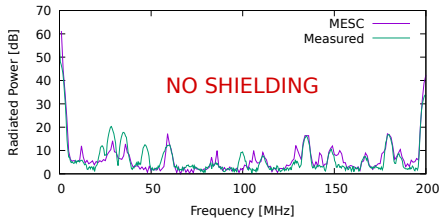

Figure 9: Compares MESC without compensating for shielding, with the measured EMI of a small test case with three buses after adding the artifacts. From this, we conclude that compensating for shielding effects improves MESC accuracy.

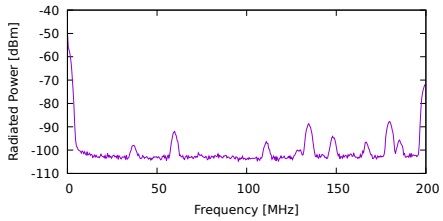

Figure 10: Shows the power emitted by the FPGA before it is programmed. When the device is on, but even before it is programmed, it emits non-uniform EMI, with 9 considerable power spikes in our frequency range.

on the OpenPiton processor. With the data from those runs, we were able to determine that MESC provides accuracy within 5% across all benchmarks, with an average deviation of less than 1%.

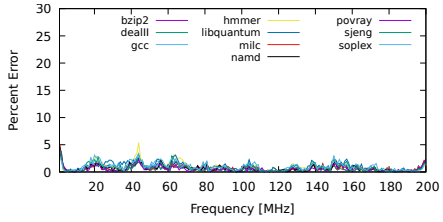

Figure 11: Shows the percent error over frequency for each benchmark.

Finally, we verify that MESC is consistent across a set of SPEC benchmarks running on the OpenPiton core. We measure EMI with the spectrum analyzer and calculate the estimated EMI from MESC. Then we take the percentage difference for each frequency. Figure 11 shows the percent error over frequency of MESC compared with the measured EMI. The average error across all benchmark is about 1% and the maximum observed error was of 5%.

One interesting thing to note in Figure 11 is that around both 20 and 45 MHz, there seems to be a systematic error that is making the modeled EMI smaller than the measured EMI. Even though this error is small, it is consistent across all benchmarks. This type of systematic error could be easily compensated for, even without knowing what the source of the error is. However, for this paper, we decided not to do this compensation, since the magnitude of the error is small and we were not able to find a cause.

6.2 Applying MESC: EMI CHopper

After determining the accuracy and validity of MESC, we are able to apply it to implement and verify EMI CHopper. In this subsection, we evaluate the potential of EMI CHopper to reduce EMI in specific frequency bands.

The two layouts were generated by slightly changing the placement parameters in Vivado. Manual placement or floorplanning could also have been used but would most likely result in a change in the frequency achieved by synthesis. The parameters that we used allowed the core to be run at the same frequency, which was desirable to evaluate EMI CHopper.

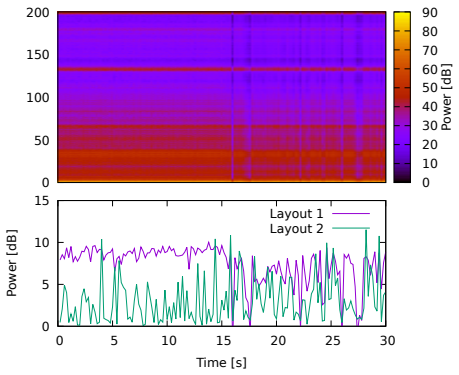

Figure 12: Interference over time for two different layouts at 100 MHz running sjeng. On the top is the EMI for a single layout over time at all frequencies from 0 to 200 MHz. On the bottom is the EMI over time for two different layouts for 100 MHz.

Figure 12 shows the EMI produced by sjeng in each of the layouts. On the top, the EMI is shown as a heat map for frequencies between 0 and 200 MHz over time and on the bottom for 100 MHz only, over time. Our model shows a significant change in EMI as an application goes through different phases. Furthermore, when comparing two different layouts, we can clearly see that in one case, the EMI increases after the phase change, but the opposite occurs for the other layout. In the figure, a phase change occurs right after 15 seconds of execution, and Layout 1 has a decrease in EMI at the 100 MHz frequency, whereas Layout 2 has an increase for 100 MHz. The instantaneous difference in EMI emitted is of up to 10 dB, in particular before 10 seconds of execution, this is equivalent to over 8× power reduction.

However, note that, in Figure 12, there are periods, *e.g.*, ≈ 16 seconds, where there is not a "better" layout. Both the layouts tested result in roughly the same (high) EMI. This may be a limitation of EMI CHopper in practical settings. In particular, the layouts defined at design time may be incapable of producing low EMI for some workloads. This is a trade-off that needs to be considered during design time. Moreover, since EMI CHopper uses a fixed subset of wires to estimate EMI, it may be limited in cases where EMI is being produced mostly by wires not sampled.

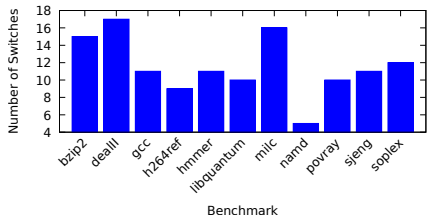

Figure 13: EMI CHopper migrates the thread between cores in order to reduce EMI. We limit the number of switches per benchmark for a maximum of one migration every 1 second. The plot shows the number of times EMI CHopper decided to migrate each application over a period of 30 seconds.

To assess the impact on performance, we look into how frequently EMI CHopper would have a thread migrate from a core to the other. For each benchmark, we obtained switching activity samples for 30 seconds in order to evaluate EMI CHopper. We were able to obtain the MESC model over time for each of these samples on the two layouts described above. From there, we analyzed the number of hops a thread would make to reduce EMI. Figure 13 shows the number of migrations a thread would perform if it were to switch cores every time the core on which it is executing would have higher in-band EMI than the other core within a threshold. Across all benchmarks, dealII had the most hops with 18, and namd had the least, only 5.

However, because there is a migration overhead associated with moving a process from one core to another, we also investigated changing constraints to how frequently a thread can hop between core. Figure 14 shows the correlation between the migration frequency and the EMI reduction at 100 MHz.

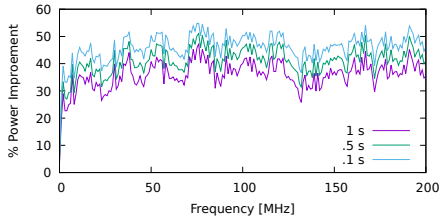

Figure 14: We evaluated how the decision of only allowing one switch every second affected the EMI. Switching every 0.1s would reduce the EMI through time, but the difference is small. When migrating more often, the impact on performance is expected to be higher.

For our default constraint (a maximum of one migration per second), we were able to see an average reduction of 37% of the EMI power across all frequencies, or about 2 dB, across benchmarks. By reducing the constraints to a maximum migration frequency of 10 times a second (once every .1s), there was an average reduction of 47% in EMI power for the 100 MHz band.

A full evaluation of impacts of communication in real world scenarios would require the ability to measure bit-flips in wireless signals, which is not viable in our current setup. However, the reductions in EMI power observed here are compelling for a more rigorous evaluation. Roughly, the noise reductions observed (up to 10 dB instantaneous) are typically equivalent to two "cell phone bars" and could be the difference between the ability to communicate or not.

7 CONCLUSION

The main contribution of this paper is MESC, an architectural level EMI model framework that architects can use to estimate the EMI emitted by a core. MESC can be used during design space exploration with cycle accurate simulators or during run time, as illustrated in this paper. The model needs the layout of the SoC and the ability to get the traces for the longest wires. This information allows us to build an EMI model that can be used in different ways.

Our evaluation against real systems shows measurements for MESC with less than 5% error compared against multiple OpenPiton layouts on Xilinx FPGAs. The model details how to do sampling, select the wires, perform FFTs, and account for artifacts and package shielding. In practice, we expect MESC to be used with higher level models, such as cycle-accurate simulators, since RTL simulation or even FPGA emulation of "real-world" cores can become impractical for a large number of cycles. Exploring the impacts of activity rate estimation through cycle-accurate simulation in EMI accuracy is left as future work.

As an example use for MESC, the paper also proposes EMI CHopper, a method that uses MESC to reduce in-band EMI for a multi-core. EMI CHopper proposes a homogeneous multi-core with two different core layouts, but the same RTL for both cores. It shows that migrating between two cores at low granularity, such as every second, can reduce power across all frequencies by 37%. If a single frequency band is targeted, the reduction is even higher. The main drawback of EMI CHopper is the need for two different layouts, which could increase the design effort. In our evaluations however, minor tweaks to a fully automated flow were sufficient to produce layouts that different enough for EMI CHopper, thus we consider that this cost can be mitigated.

As the first architectural level EMI model, this work provides many opportunities on architectural research. For example, architects can use MESC to introduce activity in wires which would increase power but would change the frequency of the EMI. Similarly, architects can combine MESC with a DVFS and estimate the EMI reduction. EMI CHopper assumes duplicates of same RTL for two differently laid-out cores, so a logical evolution of EMI CHopper would investigate the effects of two entirely different cores with separate RTLs. Again, all these questions can now be answered due to MESC.

ACKNOWLEDGMENTS

We like to thank the reviewers for their feedback on the paper. This work was supported in part by the National Science Foundation under grant CCF-1514284. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

[1] Charles Alpert and Anirudh Devgan. 1997. Wire Segmenting for Improved Buffer Insertion. In *Proceedings of the 34th Annual Design Automation Conference (DAC '97)*. ACM, New York, NY, USA, 588–593. https://doi.org/10.1145/266021.266291

[2] HFSS Ansoft. 2007. ver. 11. *Ansoft Corporation, Pittsburgh, PA* (2007).

[3] Jonathan Balkind, Michael McKeown, Yaosheng Fu, Tri Nguyen, Yanqi Zhou, Alexey Lavrov, Mohammad Shahrad, Adi Fuchs, Samuel Payne, Xiaohua Liang, Matthew Matl, and David Wentzlaff. 2016. OpenPiton: An Open Source Manycore Research Framework. In *Architectural Support for Programming Languages and Operating Systems, Proceedings of the 21st International Conference on (ASPLOS '16)*. ACM, New York, NY, USA, 217–232. https://doi.org/10.1145/2872362.2872414

[4] Michela Becchi and Patrick Crowley. 2006. Dynamic Thread Assignment on Heterogeneous Multiprocessor Architectures. In *Proceedings of the 3rd Conference on Computing Frontiers (CF '06)*. ACM, New York, NY, USA, 29–40. https://doi.org/10.1145/1128022.1128029

[5] Robert Callan, Farnaz Behrang, Alenka Zajic, Milos Prvulovic, and Alessandro Orso. 2016. Zero-overhead Profiling via EM Emanations. In *Proceedings of the 25th International Symposium on Software Testing and Analysis (ISSTA 2016)*. ACM, New York, NY, USA, 401–412. https://doi.org/10.1145/2931037.2931065

[6] Robert Callan, Alenka Zajić, and Milos Prvulovic. 2014. A practical methodology for measuring the side-channel signal available to the attacker for instruction-level events. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 242–254.

[7] Robert Callan, Alenka Zajic, and Milos Prvulovic. 2015. FASE: finding amplitude-modulated side-channel emanations. In *Computer Architecture (ISCA), 2015 ACM/IEEE 42nd Annual International Symposium on*. IEEE, 592–603.

[8] Theofanis Constantinou, Yiannakis Sazeides, Pierre Michaud, Damien Fetis, and Andre Seznec. 2005. Performance Implications of Single Thread Migration on a Chip Multi-core. *SIGARCH Comput. Archit. News* 33, 4 (Nov. 2005), 80–91. https://doi.org/10.1145/1105734.1105745

[9] Daphne I. Gorman, Matthew R. Guthaus, and Jose Renau. 2017. Architectural Opportunities for Novel Dynamic EMI Shifting (DEMIS). In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-50 '17)*. ACM, New York, NY, USA, 774–785. https://doi.org/10.1145/3123939.3123973

[10] Mentor Graphics. 2004. Hyperlynx Signal Integrity Simulation software.

[11] Z. Hadjilambrou, S. Das, M. A. Antoniades, and Y. Sazeides. 2018. Leveraging CPU Electromagnetic Emanations for Voltage Noise Characterization. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 573–585. https://doi.org/10.1109/MICRO.2018.00053

[12] Keith B Hardin, John T Fessler, and Donald R Bush. 1994. Spread spectrum clock generation for the reduction of radiated emissions. In *Electromagnetic Compatibility, 1994. Symposium Record. Compatibility in the Loop., IEEE International Symposium on*. IEEE, 227–231.

[13] Keith B Hardin, John T Fessler, and Donald R Bush. 1995. A study of the interference potential of spread spectrum clock generation techniques. In *Electromagnetic Compatibility, 1995. Symposium Record., 1995 IEEE International Symposium on*. IEEE, 624–629.

[14] Xuchu Hu and Matthew R Guthaus. 2011. Clock tree optimization for electromagnetic compatibility (EMC). In *Proceedings of the 16th Asia and South Pacific Design Automation Conference*. IEEE Press, 184–189.

[15] S I-The, A Chen, and J Keip. 2000. Spread spectrum and PLL technology combine to reduce EMI. *RF DESIGN* 23, 4 (2000), 20–25.

[16] Rakesh Kumar, Keith I. Farkas, Norman P. Jouppi, Parthasarathy Ranganathan, and Dean M. Tullsen. 2003. Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction. *Microarchitecture, IEEE/ACM International Symposium on* 0 (2003), 81. https://doi.org/10.1109/MICRO.2003.1253185

[17] Alireza Nazari, Nader Sehatbakhsh, Monjur Alam, Alenka Zajic, and Milos Prvulovic. 2017. EDDIE: EM-Based Detection of Deviations in Program Execution. *SIGARCH Comput. Archit. News* 45, 2 (Jun. 2017), 333–346. https://doi.org/10.1145/3140659.3080223

[18] Sophocles J Orfanidis. 2004. Electromagnetic waves and antennas, 2008. *Unpublished, available: http://www. ece. rutgers. edu/orfanidi/ewa* (2004).

[19] Nader Sehatbakhsh, Alireza Nazari, Alenka Zajic, and Milos Prvulovic. 2016. Spectral profiling: Observer-effect-free profiling by monitoring EM emanations. In *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on*. IEEE, 1–11.

[20] Daniel Shelepov, Juan Carlos Saez Alcaide, Stacey Jeffery, Alexandra Fedorova, Nestor Perez, Zhi Feng Huang, Sergey Blagodurov, and Viren Kumar. 2009. HASS: A Scheduler for Heterogeneous Multicore Systems. *SIGOPS Oper. Syst. Rev.* 43, 2 (Apr. 2009), 66–75. https://doi.org/10.1145/1531793.1531804

[21] Microwave Studio. 2008. CST-Computer Simulation Technology. *Bad Nuheimer Str* 19 (2008), 64289.

[22] R. Thomas, N. Sedaghati, and R. Teodorescu. 2016. EmerGPU: Understanding and mitigating resonance-induced voltage noise in GPU architectures. In *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 79–89. https://doi.org/10.1109/ISPASS.2016.7482076

[23] Kenzo Van Craeynest, Aamer Jaleel, Lieven Eeckhout, Paolo Narvaez, and Joel Emer. 2012. Scheduling Heterogeneous Multi-cores Through Performance Impact Estimation (PIE). In *Proceedings of the 39th Annual International Symposium on Computer Architecture (ISCA '12)*. IEEE Computer Society, Washington, DC, USA, 213–224. http://dl.acm.org/citation.cfm?id=2337159.2337184

[24] C. Wang, R. Callan, A. Zajic, and M. Prvulovic. 2016. An algorithm for finding carriers of amplitude-modulated electromagnetic emanations in computer systems. In *2016 10th European Conference on Antennas and Propagation (EuCAP)*. 1–5. https://doi.org/10.1109/EuCAP.2016.7481633

[25] F. Werner, D. A. Chu, A. R. Djordjevic, D. I. Olcan, M. Prvulovic, and A. Zajic. 2018. A Method for Efficient Localization of Magnetic Field Sources Excited by Execution of Instructions in a Processor. *IEEE Transactions on Electromagnetic Compatibility* 60, 3 (June 2018), 613–622. https://doi.org/10.1109/TEMC.2017.2742501