RC22935 (W0310-099) October 16, 2003
Computer Science

IBM Research Report

Spectral Analysis for Characterizing Program Power and Performance

Russ Joseph, Margaret Martonosi
Department of Electrical Engineering
Princeton University

Zhigang Hu
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Spectral Analysis for Characterizing Program Power and Performance

Russ Joseph and Maraget Martonosi
Dept. of Electrical Engineering
Princeton University
{rjoseph,mrm}@ee.princeton.edu

Zhigang Hu
T.J. Watson Research Center
IBM Corporation
zhigangh@us.ibm.com

Abstract

Performance and power analysis in modern processors requires managing a large amount of complex information across many time-scales. For example, thermal control issues are a power sub-problem with relevant time constants of millions of cycles or more, while the so-called dI/dt problem is also a power sub-problem but occurs because of current variability on a much finer granularity: tens to hundreds of cycles. Likewise, for performance issues, program phase analysis for selecting simulation regions requires looking for periodicity on the order of millions of cycles or more, while some aspects of cache performance optimization require understanding repetitive patterns on much finer granularities.

Fourier analysis allows one to transform a waveform into a sum of component (usually sinusoidal) waveforms in the frequency domain; in this way, the waveform's fundamental frequencies (periodicities of repetition) can be clearly identified. This paper shows how one can use Fourier analysis to produce frequency spectra for some of the time waveforms seen in processor execution. By working in the frequency domain, one can easily identify key application tendencies. For example, we demonstrate how to use spectral analysis to characterize the power behavior of real programs. As we show, this is useful for understanding both the temperature profile of a program and its voltage stability. These are particularly relevant issues for architects since thermal concerns and the dI/dt problem have significant influence on processor design. Frequency analysis can also be used to examine program performance. In particular, it can predict the degree of latency tolerance in a program. It can also identify periodic occurrences of important microarchitectural events like cache misses. Overall, the paper demonstrates the value of using frequency analysis in different research efforts related to characterizing and optimizing application performance and power.

1 Introduction

As modern processors become increasingly complex, understanding and predicting program behavior becomes increasingly difficult. Simulation and performance modeling techniques continue to have their limits tested with each new processor generation [18], making it difficult to characterize program behavior. Statistical simulations, performance modeling, and path analysis have all recently been studied [6, 11, 20], but emerging problems such as dI/dt noise and thermal hotspots have clouded the matter, adding a new class of benchmark classification problems that require insight on power and performance.

Fourier analysis can simplify the analysis of complex application behavior. First, Fourier techniques present a clear and orthogonal view of a program's behavior by presenting a power/performance metric with respect to frequency rather than time. As a result, key periodicities can be easily identified. Second, some important physical phenomena such as inductive power supply noise and thermal flow can be understood most naturally using frequency and filter analysis. This allows for powerful program classification and characterization using established Fourier analysis techniques.

Despite these benefits, Fourier analysis has only been applied sparsely in computer architecture studies. A few papers have examined periodicity in applications [9, 21] through use of the Fourier transform, but these approaches are not widely used. This paper argues that frequency-based assessments of processor behavior have distinct value to computer architects because they are useful in classifying program behavior with respect to important power and performance metrics.

In some cases, processor behavior is directly linked to frequency ranges in ways that make Fourier analysis germane. For example, the so-called "dI/dt" problem occurs when strong variations in processor current drain (I) occur at the resonant frequencies of an inadequately-damped power supply network [8]. Thus, it is natural to use a Fourier transform of processor current, to identify whether a particular workload may have frequency components that lead to resonance in the power supply network which creates voltage regulation problems. Another power-related problem, thermal regulation, can also be aided by frequency domain analysis. In particular, temperature models of heat-flow within a processor act as low-pass filters. Frequency analysis thus offers a direct way to reason about thermal behavior. While power related problems like dI/dt and thermal management have natural links to frequency analysis, other processor insights can be gained from Fourier transforms as well. The paper also discusses how one can compare frequency spectra, for example between in-order and out-of-order processors, to learn about the ability of a processor/workload to tolerate different performance disruptions such as cache misses.

The contributions of this paper lie at several levels:

- First, we present a methodology for computing frequency spectra of benchmark programs for important power and performance metrics.
- We demonstrate how frequency analysis and filter techniques can be used to diagnose the severity of dI/dt power supply noise for different benchmarks.
- We also show how filter based analysis techniques can be used to classify programs with reselect to thermal hotspots.

- We discuss how frequency analysis can be extended to understand performance issues such as the ability of a workload and execution core to tolerate traumatic events (e.g. cache misses).

The remainder of the paper is structured as follows: Section 2 gives a review of Fourier transforms and some basic filter theory. Section 3 gives details on our Fourier implementation and our experimental methodology. Sections 4 and 5 show how Fourier analysis can be used to analyze and characterize benchmarks with respect to dI/dt inductive supply noise and thermal hotspots. These are important problems in current microprocessor design and the insight achieved through frequency and filter analysis of benchmarks is valuable. Section 6 discusses how the frequency analysis techniques in this paper may be extended to also examine performance issues such as tolerance to cache misses, and we describe how our contribution relates to existing work. Finally, Section 7 offers a summary and conclusion.

2 Fourier Analysis for Computer Architecture Studies

The well-known Fourier Transform is useful for expressing a time-domain series (such as a waveform of power consumption versus time) into frequency spectra. Important filter operations and identities such as Parseval's Theorem are also part of Fourier theory and are helpful in describing the interactions between complex systems. In this section we provide some background for the Fourier analysis used in this paper. In the following sections we use this basic theory to characterize and reason about program power and performance.

2.1 A Review of Fourier Transforms

The Fourier Transform is a mathematical operation that can represent a single complicated waveform as the sum of many simple analytic functions, namely sinusoids. Each of these sinusoidal functions corresponds to a distinct frequency. The magnitude of an individual sinusoid represents the strength of its corresponding frequency component in the original waveform. The Fourier transform of a function $f(t)$ is defined as:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt \tag{1}$$

In the above formula, values of $F(\omega)$ describe the frequency content of the original function $f(t)$ at different frequencies given by ω. For example, $F(0)$ is the value of the Fourier transform at frequency zero, and it corresponds to the "DC component" of the function – the average value of f(t). On the other hand, $F(2\pi/s)$ would be the frequency component at an angular frequency of $2\pi/second$, more commonly known as 1Hz. Equation 1 is the standard continuous time description of the Fourier transform, but we plan to operate on data collected from cycle-level architecture simulations. This changes the integral from the previous representation into a summation of a discrete set of sinusoids as shown below:

$$F(k) = \sum_{t=0}^{N-1} f(t)e^{-2\pi i \frac{t}{N} k} \tag{2}$$

If the waveform being transformed is itself a sinusoid of a particular frequency, then the frequency domain representation will only require a single sinusoid at that frequency. On the other hand, other waveforms, such as a square wave, require an infinite number of sinusoids of increasing frequencies to represent them accurately. This transform is illustrated in Figure 1. For a square wave of frequency ω, the lowest frequency sinusoid in the Fourier transform will be of frequency ω, and the remainder of the sinusoids will be increasing multiples of ω. In this case, the lowest-frequency sinusoid is called the *fundamental mode* and the higher frequencies are referred to as *harmonics*. The relationship between frequency components including fundamental modes and harmonics can be visually represented in a spectral plot. As an additional example, Figure 2 shows both the time domain and frequency domain representations of a signal.

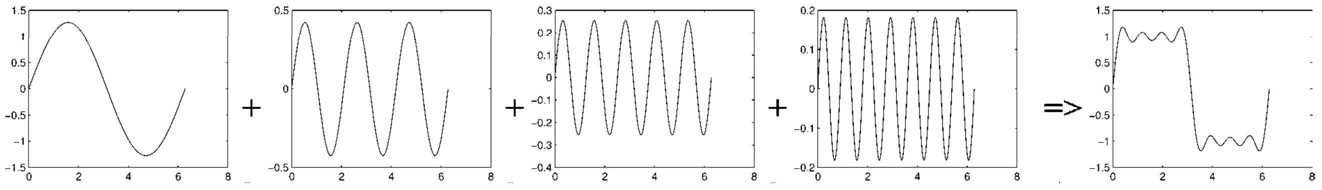


Figure 1. A series of sinusiodal signals can be added successively to build a near-square wave. The fundamental mode (far left) and harmonics at higher frequencies all contribute.

2.2 Relating Complex Behavior to Frequency Analysis

In addition to basic signal decomposition, Fourier analysis is a powerful tool for analyzing complex systems because it allows one to apply filter theory to explain how individual components interact. Consider the diagram in Figure 3a which shows the connections between two cascaded component boxes f and g. These two elements work together to transform a single input waveform to a single output waveform. To understand how the two items work together with time-domain analysis, one would analyze the relationships between the input $i(t)$, the intermediate signal $f(t)$ and the output signal $g(t)$. Fourier analysis can be used to reach the same conclusions, often with greater ease and added intuition. Figure 3b, shows the frequency domain visualization of the cascaded filters. Instead of identifying the relationships between $f(t)$ and $g(t)$, one can compare the relationships between the frequency spectra $F(w)$ and $G(\omega)$. This is useful because the key interactions between components are sometimes more visible in the frequency domain. For example, if component g acts as a *low-pass filter*, then most of the higher frequency content in $f(t)$ will be suppressed in the output $g(t)$, and most of the low frequency aspects of $f(t)$ will appear.

On the surface basic filter classifications seem rather simplistic yet they explain how some subsystems in a microprocessor interact. Specifically, the power delivery system can be thought of as a bandpass filter since parasitic inductances in the electronic packaging can amplify mid-frequency fluctuations in the amount of current consumed. Typically, the most critical frequencies are within the range of 50MHz-200MHz, which is an intermediate point between the high frequency cycle-to-

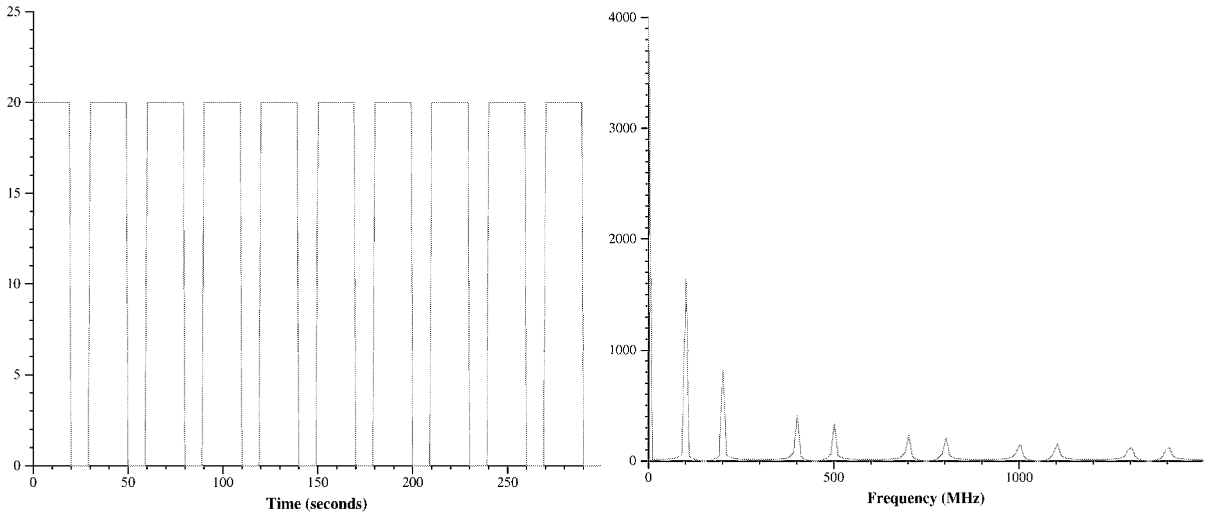


Figure 2. Depiction of signal in the time domain (left) and frequency domain (right) for a simple pulse. The frequency domain depiction identifies the fundamental mode at 100MHz and harmonics at higher frequencies.

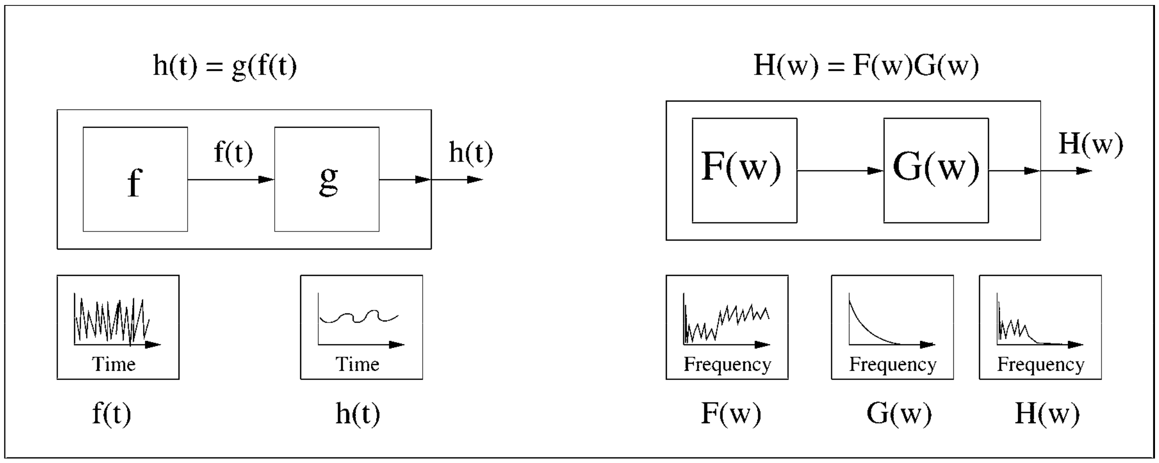


Figure 3. Output of cascaded components in time-domain (left) and frequency-domain (right). Mathematically the time-domain components act as a composition of two functions. The frequency-domain composition is a multiplicative product.

cycle fluctuations and low frequency effects such as program phases. In contrast, the thermal interface of the CPU can be thought of as a low-pass filter. Since the time constants involved in thermal flow are very large compared to the clock rate of the processor, high frequency operations are filtered out, but long-term (low frequency) program phases can have an impact on temperature.

In addition to explaining important subsystem interactions, Fourier and filter analysis can be used to classify program behavior based on frequency. For instance, rather than directly reasoning about temperature or power supply noise, one can classify and characterize with respect to power or current consumption, metrics that are more intuitive to most architects. Frequency and filter techniques form a bridge between these intuitive metrics and the physical phenomena that they influence. At the same time, fidelity is not sacrificed because Fourier analysis gives us a clear method for relating these intermediate metrics with the metrics of study, namely temperature or supply voltage variation. Specifically, for a linear system such as the power supply, the voltage spectra is the multiplicative product of current spectra with the frequency response of the power supply network. Mathematically, the Fourier transform of current consumption, $F(w)$ is point-wise multiplied with $H(w)$, the frequency response of the filter, i.e. $G(w) = F(w)H(w)$. The thermal interface is also a linear system, and temperature spectra can be computed in a similar fashion. Basic filter theory allows more intuitive metrics such as current and power consumption to be related to voltage noise and temperature. Furthermore, the ability to separately characterize these program metrics provides considerable modularity. For instance an experimenter could collect a single current consumption spectrum and use these filter techniques to project power supply noise under many different types of power supplies, each of which has its own frequency response.

Filter theory also gives mechanisms for translating back and forth between frequency-domain and time-domain behavior. In particular, basic identities allow one to directly compute mean and variance for a time domain signal solely from frequency spectra. The mean value of metric f is equal to its DC component, $F(0)$. The variance of the same metric is computed by squaring its spectral function, $F(w)$ and taking the area under the curve. Mathematically: $Variance(f) = \int_{w=-\infty}^{\infty} F(w)^2$. These identities enable architects to reason about issues like voltage variation and temperature in the frequency domain where filter theory lends intuition. Statistical metrics like mean and variance explain how these findings translate in a time-based sense. These basic principles lend themselves to efficient and accurate methodologies for characterizing program behavior with respect to temperature and power supply noise.

3 Fourier Methodology and Implementation

In this paper, we present a methodology for calculating frequency spectra of important power and performance metrics. Frequency spectra can be used directly to characterize program behavior. In addition, because microprocessor power delivery and thermal exchange mechanisms are sensitive to current and power consumption at specific frequency ranges, filter analysis techniques can be combined with the frequency spectra to offer insights in both dI/dt and thermal behavior.

Our frequency analysis is a two-part process which is detailed in Figure 4. We use this sequence to produce data and characterizations for two different power/performance metrics: (i) a primary metric and (ii) a secondary metric which is related to the primary. For the studies in this paper, a primary waveform is an input into a lineary system whose output is the secondary waveform. For example, the current consumed by the processor is an input to the power deliery system, and the output is an operating voltage level. In the first half of our analysis, the program to be classified is run under a detailed microarchitectural simulation, and we calculate a frequency spectrum for an important primary metric (e.g. power or current consumed). In the second half, the spectrum for the primary metric can be analyzed using filter techniques to estimate a second metric (e.g. temperature or inductive noise).

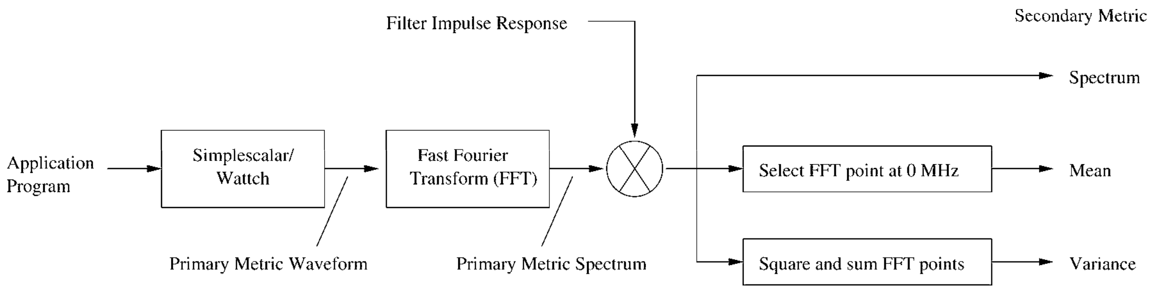


Figure 4. Flowchart for the Fourier and filter analysis in this paper. FFTs and filter operations are used to produce frequency spectra and statisitical information for important power/performance metrics.

To produce spectra for the primary metric, we need both a traditional cycle-level simulation framework as well as the ability to take Fourier transforms of the time-series data that the simulator produces. Because we are interested in evaluating patterns in both power and performance data, the starting point for our simulation framework is a modified version of Wattch [4], a power/performance simulator based on SimpleScalar [5]. We made several modifications to Wattch to improve the accuracy of power and performance simulations at the cycle-by-cycle granularity. First, we used scaling factors from [14] to tune our Wattch model for a 3GHz processor with a nominal supply voltage of 1.0V. Since Wattch and SimpleScalar do not accurately model the impact of pipeline refill costs following branch misprediction (and since we feared that this effect could change the per-cycle bursts we observed) we added additional pipeline stages to account for the super-pipelined fetch and decode stages. Furthermore, we made modifications to improve the per-cycle power computations, spreading the energy of multiple cycle operations, such as floating point execution, over several cycles. The processor we simulate has the parameters shown in Table 1.

The modified Wattch simulator tracks an important primary metric (e.g. power or current consumed) at a fine-grained cycle-by-cycle basis and at a coarse-grained multi-cycle basis. This allows us to produce both mid-frequency spectra which capture program behavior over tens or hundreds of cycles and low-frequency spectra which capture program behavior over the full course of simulation. Figure 5 shows pseudocode that describes how frequency spectra for a primary metric (in this case power) can be computed. In addition to standard performance and power simulation, the simulator stores the power

Execution Core	
Clock Rate	3.0 GHz
Instruction Window	80-RUU, 40-LSQ
Functional Units	4 IntALU, 1 IntMult/IntDiv
	2 FPALU, 1 FPMult/FPDiv
	2 Memory Ports
Front End	
Fetch/Decode Width	4 inst,4 inst
Branch Penalty	12 cycles
Branch Predictor	Combined: 4K Bimod Chooser
	4K Bimod w/ 4K 12-bit Gshare
BTB	1K Entry, 2-way
RAS	32 Entry
Memory Hierarchy	
L1 I-Cache	64KB, 2-way, 3 cycle latency
L1 D-Cache	64KB, 2-way, 3 cycle latency
L2 I/D-Cache	2MB, 4-way, 16 cycle latency
Main Memory	300 cycle latency

Table 1. Processor Parameters

```
while(!done_simulation)
{
  performance_simulation();
  power_simulation();

  mid_power_frame[mid_power_index] = instantaneous_cycle_power;
  mid_power_frame_index++;

  if (mid_power_index == mid_power_length)
  {
    /* compute mid frequency spectra */
    frame_spectra = fft_magnitude(mid_power_frame);
    mid_power_frame_index = 0;
    mid_power_frame_count++;
    for(i = 0; i < mid_power_length; i++)
    {
      mid_power_sum[i] += frame_spectra[i];
    }

    /* compute low frequency data point */
    low_power_frame[low_power_index] = average(mid_power_frame);
    low_power_index++;
  }
}

low_power_spectra = fft_magnitude(low_power_frame);
for(i = 0; i < mid_power_frame_count; i++)
{
  mid_power_spectra[i] = mid_power_sum[i] / mid_power_frame_count;
}
```

Figure 5. Pseudocode for low frequency and mid frequency spectra computation.

consumed each cycle in an array. This array is used to compute the mid-frequency spectra and represents the power waveform with respect to time over a small window of `mid_power_length` cycles. In addition, the average value of this array is used as a datapoint in the low-frequency spectra. The `mid_power_length` variable controls not only the number of points in the waveform, but the frequency range represented in the frequency spectra. For our analysis in Sections 4 and 5 we discuss our choice of values for those experiments. When the waveform buffer is full, a FFT is taken, and we aggregate the result of this FFT by averaging individual spectral windows in the program. This windowing method that is similar to that done by Voldman and Hoevel in their early work [21] and indeed also similar to the windowing methods used by many audio systems like mp3 players.

As is true for the limited other uses of Fourier analysis in computer architecture, we only use the frequency magnitudes and not the phase information from the transform. The magnitude totals are stored as SimpleScalar output statistics. We note that the overhead to compute the FFT is relatively small compared to the existing power/performance simulation activity in Wattch/Simplescalar. For example, with the 4096 sample window sizes used in our experiments, we observed less than 8% slowdown over baseline Wattch simulation.

At the end of simulation, the frequency spectrum for the primary metric is produced in addition to standard performance summary statistics. The spectra contains a useful frequency domain characterization of the benchmark that can be used directly. On the other hand, this primary metric may also be used to analyze the benchmark's behavior with respect to a secondary metric like temperature or dI/dt noise. This step is accomplished in the second analysis phase where a filter is applied to the primary spectra to produce frequency spectra for the second metric.

The filter analysis used in this paper uses some of the basic theory presented in Section 2 to relate primary metrics like power or current consumed to voltage and temperature. As discussed previously, the spectra for the secondary metric is the point-wise multiplicative product of the primary spectra and the frequency response for the secondary system (e.g. power supply network or thermal flow mechanism). In either case, the frequency response is the Fourier transform of the secondary system's response to a narrow pulse (or impulse). For the power supply system, the impulse response is already used commonly in simulation [7], and is easily available. For the thermal studies, the impulse response is the decaying exponential $e^{-t/\tau}$, where τ is the thermal time constant. The frequency response was computed by taking the Fourier transform of the impulse responses. The primary spectra was multiplied by the frequency response to yield the secondary spectra.

4 Fourier Analysis to Gauge Voltage Variations

As a first example of the benefit of Fourier analysis in benchmark classification, we demonstrate that Fourier analysis can be used to characterize applications that stress the current delivery and power supply network of a processor. Frequency domain analysis seems a natural tool for understanding the so called "dI/dt problem" because parasitic electrical components amplify the impact of current variations at specific frequency ranges. By applying Fourier analysis, one can isolate these power fluctuations by frequency range and then determine how these variations are likely to impact voltage stability.

4.1 Power Supply Networks Overview

The dI/dt problem is a growing concern for high performance microprocessors that complicates the design process and adversely impacts cost. In essence, parasitic inductances in the materials used to build the power supply system can produce large ripples on the V_{DD} and GND lines informally known as "ground bounce". Large voltage ripples are unacceptable since they may cause circuits to malfunction. The term "dI/dt" refers to change of current with respect to time and the severity of the voltage ripples are directly proportional to this quantity. This is problematic because modern processors can create very large current swings and correspondingly large values of dI/dt when they switch to and from energy-saving execution modes. Packaging techniques that attempt to lower the inductance or electrically dampen the voltage swings are potentially expensive.

Recently proposed microarchitectural dI/dt solutions have the potential to offer dI/dt protection at lower cost, but they necessitate complicated analysis. Most of the recently proposed dI/dt control mechanisms monitor processor behavior and increase or decrease the degree of computation to limit the value of dI/dt [7, 8, 12]. This momentarily degrades performance by stalling in-flight instructions to limit an upward surge in current consumed, or temporarily hurts energy efficiency by issuing no-ops to functional units. The authors of these proposals stress that very large swings in current are rare and consequently the microarchitectural interventions are also rare. As such, they are unlikely to seriously impact performance and power over a full application run. Nevertheless, the advent of microarchitectural dI/dt mechanisms demands that we have analysis techniques to characterize and reason about benchmarks that pose varying degrees of challenge to dI/dt controllers.

Frequency analysis is ideally suited to classifications of this nature. Current fluctuations at different frequency levels have dissimilar effects on the inductive noise. In particular, the most important frequencies are typically within the range of 50MHz-200MHz for state-of-the art processors. In essence, the power supply system acts as a bandpass filter (as shown in Figure 6), amplifying current fluctuations at those time scales, and converting them into possibly large voltage ripples. Fourier analysis of program power and current consumption can first identify benchmarks which have large frequency components in this range. Secondly, it can quantitatively analyze their likelihood for requiring dI/dt control and hence incurring performance loss and increased energy consumption.

4.2 Results: Benchmark Classification for dI/dt

To demonstrate the benefits of Fourier techniques for dI/dt characterization, we used the Simplescalar/Wattch simulation infrastructure described in Section 3 to obtain the frequency spectra of processor current. Since the principle frequencies involved for dI/dt are within the range of 50MHz-200MHz, we used the mid-frequency spectra and an analysis window size of $mid\_frequency\_length$ = 4096 cycles. For evaluation, we used the full set of SPEC2000 benchmarks with reference inputs. The benchmarks are compiled under full optimization, and we simulated for 500 million committed instructions after

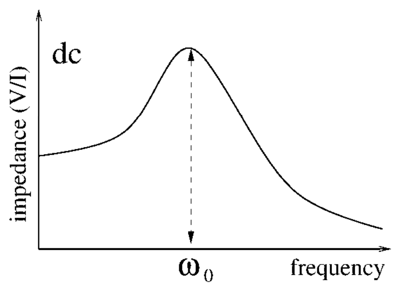


Figure 6. Frequency response of a second-order linear system (log scale). It acts as a bandpass filter and amplifies any components near the critial frequency ω_0.

fast-forwarding 2 billion instructions to avoid unrepresentative behavior.

As a power supply system model, we assume that the nominal supply voltage is set at 1.0 Volts, and that the processor can tolerate a +/-5% voltage swing. The critical frequency for this supply network is 50MHz, which means that current fluctuations around this frequency will have the most impact on the supply voltage level. This frequency agrees with empirical data from a commercial microprocessor [1].

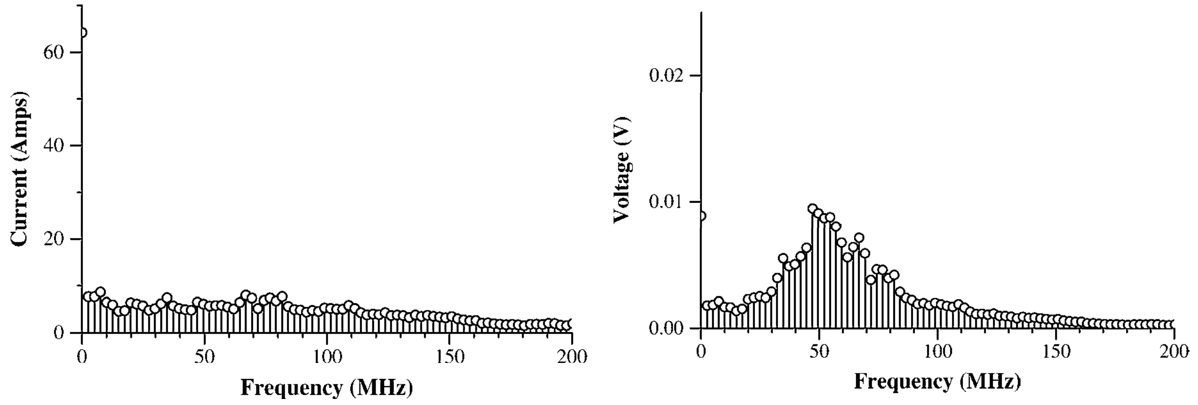
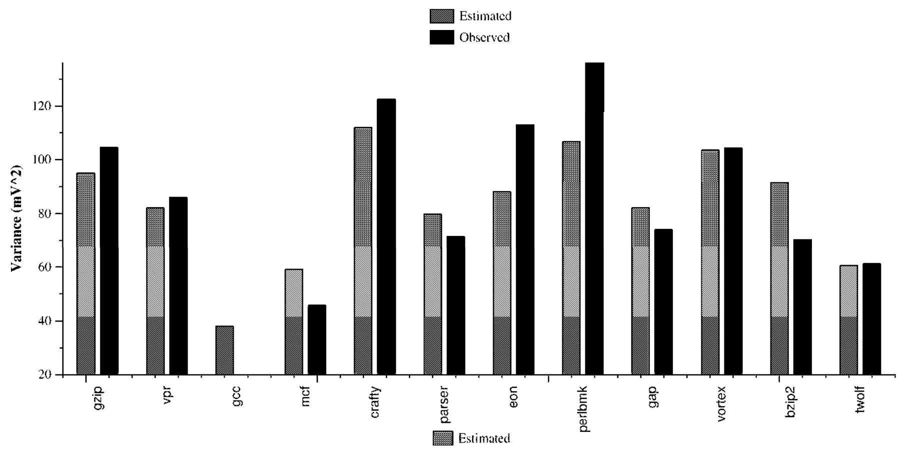


Figure 7. Current (left) and voltage spectra (right) for a 500M instruction runs of 252.eon. The power supply's bandpass filter tendencies amplify a somewhat flat current spectrum into voltage spectrum with a pronounced peak peak near 50MHz.

Figure 7 shows the processor current and voltage spectra for the SPEC benchmark eon. The current spectrum on the left was computed using the mid-frequency Fourier analysis described in Section 3 where the instantaneous current consumed per cycle was the primary metric. The voltage spectrum on the right was computed as a secondary metric via the filter operation also described in Section 3. We see clearly in this figure that eon has a rather flat current spectrum. However, the voltage spectrum shows the effect that the power supply has on the region near 50MHz. It functions as a bandpass filter, magnifying spectral points near the critical frequency. Together the two spectra make it easy to identify how current fluctuations at various frequencies will affect dI/dt inductive noise.



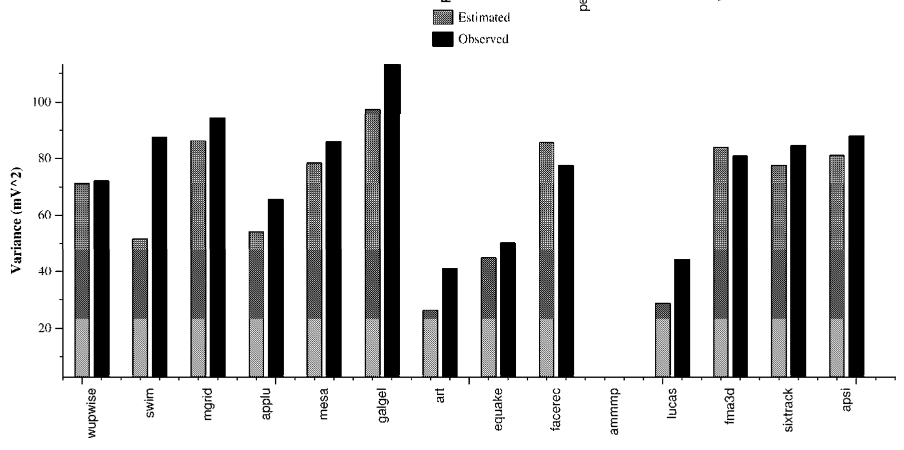


Figure 8. Comparison of observed and estimated voltage variances for SpecINT (top) and SpecFP (bottom).

As Section 2 discussed, filter analysis identities make it easy to relate spectra plots to statistical quantities like mean and variance. For dI/dt analysis, large voltage variances are indicative of problematic execution sequences. They would likely stress a dI/dt control mechanism, resulting in lost performance and energy-efficiency. To show the value of these statistical calculations, we use the approach described in Section 3 to compute voltage variance. While Parseval's Theorem (the identity used in this computation) provides an exact variance number, a small amount of error is introduced in our methodology because we are effectively aggregating the contribution of the individual window segments. This makes the spectra variance an estimate rather than an absolute variance figure. Figure 8 shows that our approximated variance performs very well compared to traditionally computed statistical variance. For both the SPEC integer and floating point benchmarks, the estimates show high fidelity to the observed variance. In benchmarks that exhibit high variance such as crafty, perlbmk, mgrid, and galgel, the estimated variance is also large. At the same time, benchmarks like gcc, mcf, art, ammp, and lucas which have low observed variances also have low estimated variances. The largest discrepancy occurs for swim, where a

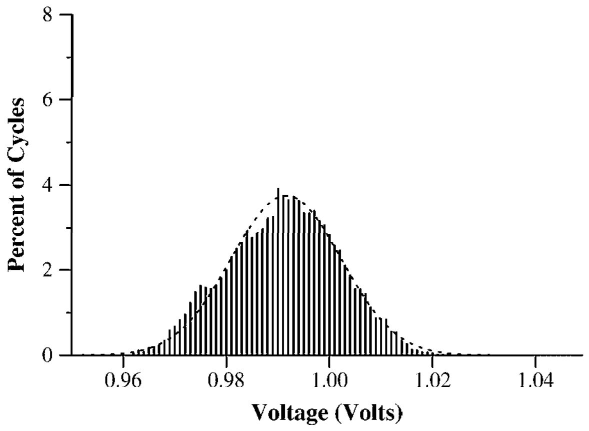


Figure 9. A voltage histogram for 252.eon (bars), compared to a Gaussian distribution (dotted lines) with the same mean and variance.

rather large variance is reported, despite only a moderate variance estimate. In this case the aggregation used in reducing the mid-frequency analysis frames into a single spectrum is largely responsible. One solution might be to group analysis frames which are have common characteristics to avoid constructing a composite depiction of multiple dissimilar regions. This is an important area for future work. However, the combination of Fourier and filter analysis can be useful in characterizing problematic dI/dt applications and visualizing the important spectral contributions.

Since dI/dt control is enabled whenever the voltage approaches an extremely low or high value, it would be useful to have a method for quantifying how likely the voltage is to approach an extreme. The statistical variance of the voltage level can be combined with the mean to produce a powerful characterization tool. For our data, we have found that a Gaussian distribution is a good fit to the statistical voltage distributions. For example, Figure 9 shows how a Gaussian distribution closely matches the voltage histogram of eon. Since a Gaussian distribution is uniquely identified by a $(mean, variance)$ pair, the filter analysis procedures we describe can provide all the needed information to produce a representative distribution.

To determine whether a benchmark is stressful, we first compute the mean and variance as described above. As we noted, many of voltage histograms we observed followed normal distributions. We applied a Gaussian model to our calculated mean and variances and used probabilistic analysis to determine whether or not a benchmark spent an appreciable portion of its execution below 0.97V. This is within 20mV of the likely minimum voltage level for this processor. In this study we chose a threshold of 1% of the program's execution time to signify a noticeable portion of execution time. This technique proved effective at characterizing 22 out the 26 SPEC benchmarks. In particular, the classifier was able to correctly identify many voltage stressful benchmarks like gzip and crafty. It also identified which benchmarks would not be stressful (e.g. mcf, ammp, and lucas).

In all, spectral analysis can be useful in characterizing the dI/dt behavior of programs. Properties of the power supply system make it ideally suited for frequency analysis techniques which can visually and quantitatively identify problematic applications.

5 Fourier Techniques for Thermal Analysis

In this section, we demonstrate the strength of Fourier analysis in thermal characterization of benchmarks. Microarchitectural temperature regulation offers promise in mitigating thermal issues in high-performance processor design, but architects currently do not have many tools for analyzing temperature profiles. Fourier analysis can be helpful because heat transfer processes in a chip mimic low-pass filters. This makes a large body of existing work on frequency analysis immediately applicable to studying microprocessor temperature. We show that frequency domain analysis can help to explain important behavior in real applications, and aid architects in evaluating the efficacy and impact of temperature-aware micro-architectures.

5.1 Thermal Behavior and the Frequency Domain

Trends in processor design have steadily increased the importance of thermal concerns [2, 10]. In response to the challenges of cooling system design, several proposals have examined the benefits of temperature-aware computing [3, 13, 17, 19]. These proposals consider thermal monitors that detect when the operating temperature approaches extreme levels and use dynamic mechanisms either in software [13] or in hardware [3, 17, 19] to respond by delaying or migrating computation.

If dynamic thermal management is to be effective at reducing cooling system demands while not harming performance, architects need powerful tools to reason about thermal behavior in real programs. In this work, we show how frequency domain analysis can be used to study localized heating and architectural hotspots.

Well-established dualities between current and heat-flow are useful in thermal models. In particular, architectural level temperature simulators [17] use temperature RC equations that mirror electrical RC equations. Within these models, temperature corresponds to voltage, and heat-flow corresponds to current. There are thermal capacitances and resistances that govern the flow of heat between architectural structures in a similar way that their electrical counterparts affect the flow of current.

The RC networks that represent heat flow for processor components are linear time-invariant systems, just as the electrical networks that model the power supplies. As a direct consequence, they can be represented by impulse responses. This means that a large body of useful analysis techniques can be used to understand the thermodynamics. Specifically, the Fourier domain depiction of the impulse response, also known as the frequency response, highlights important thermal properties.

The frequency response of the thermal RC network depicts how power variations affect temperature at various frequencies. Figure 10 shows the frequency response of the thermal behavior of a load store queue. Since the low frequency components are significantly larger than the higher frequency components, it is clearly a low-pass filter. Consequently, high frequency power variations have little impact on the temperature profile, but low frequency power bursts do. As we show, this knowledge aids in thermal benchmark classification. Although frequency analysis was not directly applied in [19], the authors also pointed out that thermal systems act as low-pass filters.

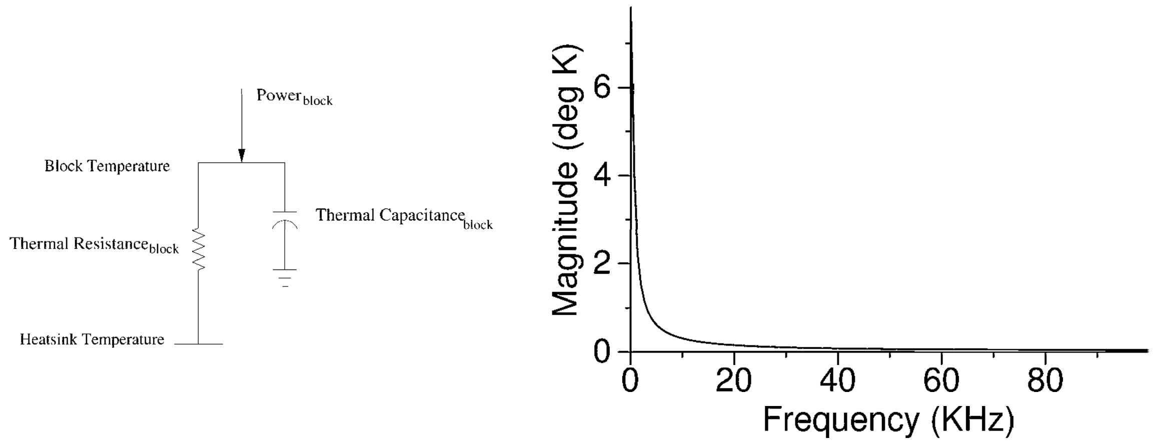


Figure 10. RC temperature model circuit (left) and frequency response for thermal model of LSQ (right). The thermal model exhibits low-pass filter behavior in the frequency domain.

While it may be tempting to take power averages over very large windows to serve as a direct proxy for temperature, [19] demonstrated that this approach is far less accurate than methods that also consider physical properties of heat flow. By explicitly modeling thermal properties in the frequency domain, we avoid these limitations.

For the temperature analysis described in this paper, we apply thermal models similar to [17]. Specifically, we model per unit temperature using single lumped RC circuits and assume a heatsink/spreader temperature of 100 °C. We also chose a thermal stress point of 107 °C and a thermal emergency level of 108 °C as in [17]. These correspond to points when temperature is at the edge of the allowable operating range and when thermal failure can be expected. While the models used in this paper are somewhat simplified, they are still useful in evaluating thermal behavior, and the approach outlined here could be modified to work with more detailed models. In Section 7, we discus ways to extend our analysis to more detailed temperature models.

5.2 Temperature Analysis: Results

In this section we show how to use frequency analysis to characterize temperature behavior. For illustrative purposes, we focus on hotspot behavior in the load store queue. The same methodology can be applied to other processor structures since the same basic equations govern their heat-flow.

In our experiments, we used SPEC2000 benchmarks, with the power and performance model described in Section 3. The thermal models are based on [17]. We first skipped unrepresentative program phases, and then simulated for 200 million cycles. In our analysis, we used $mid\_power\_length = 100,000$. This allowed us to focus on the low frequency components which are important for temperature.

Table 2 presents power profiles for six SPEC benchmarks. It shows performance as measured by IPC. It also conveys both

Benchmark	IPC	Total Power		LSQ Power			
		Mean	Var	Mean	Total Var	Low Freq	High Freq
gzip	1.784	63.6	320.5	1.073	0.411	0.007	0.404
mesa	1.988	62.4	509.7	1.122	0.464	0.017	0.447
wupwise	1.099	46.3	704.1	0.846	0.270	0.000	0.270
facerec	0.944	44.6	571.5	0.829	0.236	0.035	0.201
twolf	0.552	40.0	301.5	0.754	0.155	0.000	0.155
mgrid	0.436	35.4	373.5	0.724	0.125	0.000	0.125

Table 2. Power mean and variance for whole processor and load store queue. Low Freq and High Freq show spectral density below and above 30KHz.

the mean and variance of total processor power and more specific information about power usage of the load store queue. We see that two benchmarks (gzip and mesa) have relatively high power means for both total power and the LSQ. Wupwise and facerec have smaller means, but twolf and mgrid have the smallest power means. For the LSQ, a steady state power level of 0.875 W would place the unit at a thermal stress level of 107 °C and 1.00 W at steady state would place the system at a thermal emergency of 108 °C. Based on these numbers, one might suspect that gzip and mesa might be thermally intensive because of their high average power levels. Similar reasoning suggests that lower power benchmarks like twolf and mgrid would not be thermal stressors. Simulations confirm these initial classifications. However, it is not clear initially how much time wupwise and facerec spend in thermal emergency. The mean power for both of these benchmarks is just below the steady state thermal stress point of 0.875 W, and their variances suggest that they might spend some portion of their execution above the thermal stress level, and possibly the thermal emergency level.

Spectral density measures how much a particular frequency range contributes to the statistical variance of a metric, in this case power of the LSQ. Table 2 also shows how the spectral density is distributed over low and high frequency ranges. The Low Freq column shows which portion of the LSQ variance is present at frequencies below 30KHz, and the High Freq column shows how much of the remaining variance is distributed above 30KHz. For all of the benchmarks, the higher frequencies contribute to a dominant portion of the execution. Facerec has a significant low frequency component, compared to the other benchmarks. Wupwise on the other hand, has no significant low frequency components – all of its power variation comes from high frequencies. Figure 11 shows the LSQ power spectra for wupwise and facerec. Note the presence of low frequency components in facerec, and the absence of them in wupwise. Since the thermal hotspot model acts as a low pass filter, one might expect facerec to be more thermally stressful than wupwise. The observed temperature profile presented in Table 3 confirms this notion. Wupwise spends no time in either a stress or emergency mode, yet facerec spends almost 20% of its execution in these modes. To further demonstrate the utility of Fourier analysis for thermal analysis, Figure 12 also shows how temperature variances computed with Parseval's Theorem and the power spectra compare to those gathered in the time domain. Figure 12 shows that the estimated variance closely matched the observed temperature variance. This is true for both high variance benchmarks like facerec and low variance benchmarks like wupwise, twolf and mgrid.

Frequency domain analysis can be extremely useful in characterizing thermal behavior. It first helps to explain properties

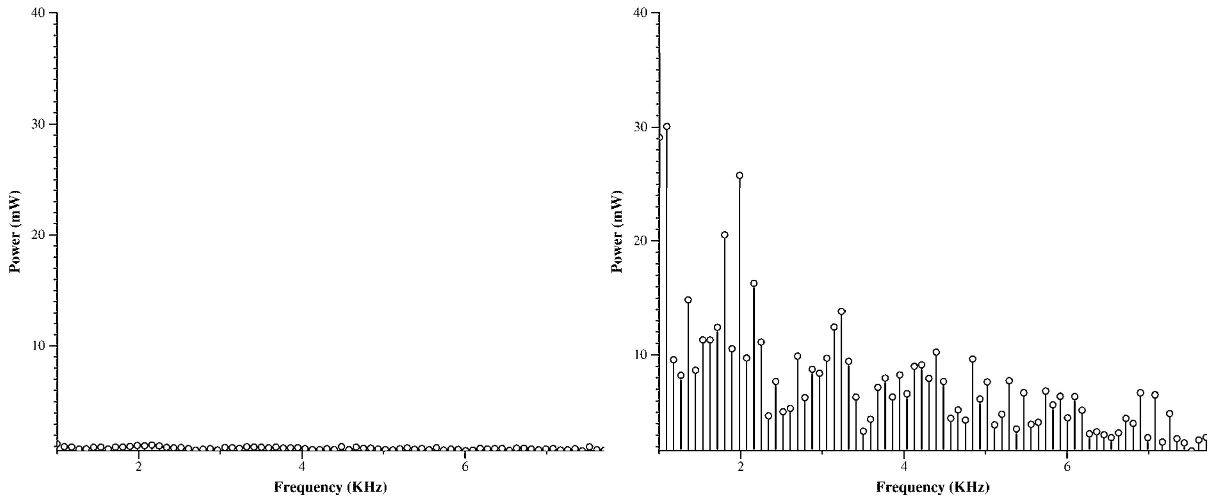


Figure 11. LSQ power spectra for wupwise (left) and facerec (right). Wupwise shows little frequency-wise variation, while facerec shows frequency components at with both high and low magnitudes.

Benchmark	Temperature		Above 107 °C	Above 108 °C
	Mean	Var		
gzip	108.5	0.663	98.0	85.6
mesa	108.9	2.193	84.2	83.4
wupwise	106.7	0.113	0.0	0.0
facerec	106.6	4.145	21.3	19.8
twolf	106.0	0.116	0.0	0.0
mgrid	105.8	0.082	0.0	0.0

Table 3. Observed temperature mean and variance of load store queue and percent of cycles spent in thermal stress (above $107\,°\mathrm{C}$**) and emergency (above** $108\,°\mathrm{C}$**) .**

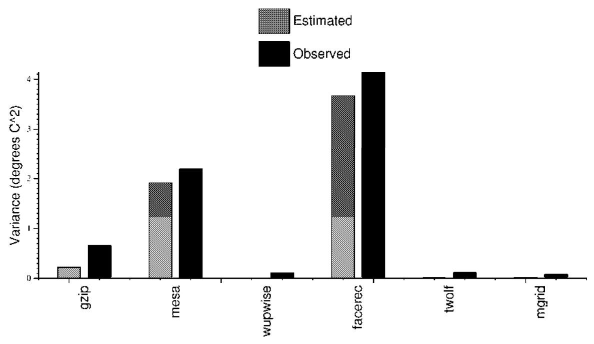


Figure 12. Comparison of observed and estimated temperature variances for the LSQ.

of the system, especially its function as a low-pass filter. Secondly, it can be used to gauge the thermal profile of applications. Power variations at high frequencies have virtually no impact on temperature, but at lower frequencies, these power variations can be indicative of bursty thermal behavior. These types of analysis are particularly useful since aggregate measures like average power are not as useful for identifying some of these patterns.

6 Discussion

While the focus of this paper was primarily on showing how frequency analysis can be used to classify program behavior with respect to power related phenomena such as dI/dt inductive noise and thermal hotspots, some of these techniques can be extended to performance analysis as well. For example, one could take Fourier transforms of performance related quantities such as IPC or branch prediction accuracy. This type of analysis can provide additional insight in processor/workload characterizations.

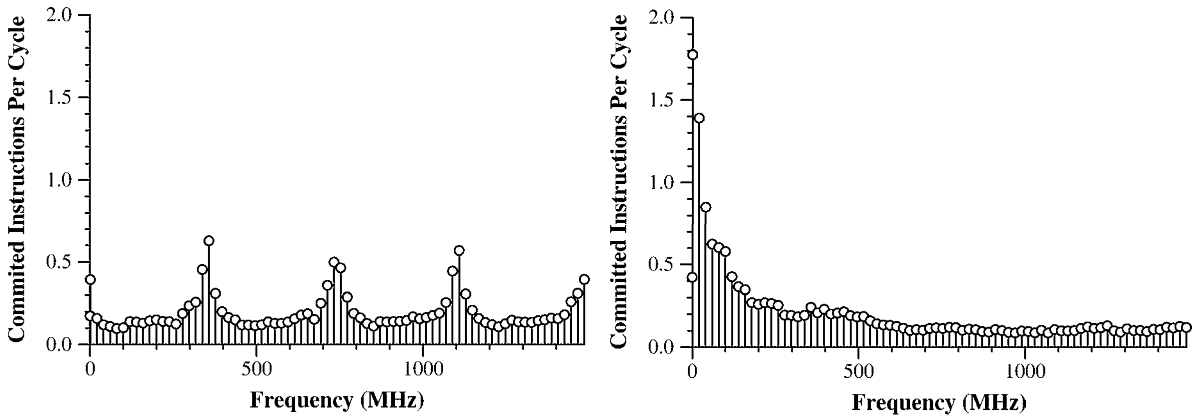


Figure 13. IPC spectra for 172.mgrid with in-order execution (left) and out-of-order execution (right). The in-order spectra shows a significant spike at 360MHz and a sequence of harmonics.

To illustrate this point, we present a small example that examines the latency tolerance of a processor and workload to overcome cache misses. In this experiment the metric of interest is committed instructions per cycle. We use the same methodology outlined in Section 3 to create IPC spectra. Figure 13 shows the IPC spectra for the SPEC2000 benchmark mgrid under two different processor issue policies: in-order execution and out-of-order execution. There are several key differences in these two plots. First the out-of-order spectrum has a significantly higher DC (0 Hz) component. This is not surprising since out-of-order execution is known for performance advantages over in-order execution. The second observation is that the in-order spectrum has a number of sharp spikes occurring at a fundamental frequency of 360MHz and its harmonics. The source of these spikes are L1 data cache misses that occur roughly every 8 cycles in a dominant region of code. Due to its inability to tolerate L1 misses, the IPC spectrum for in-order execution echos this periodicity and produces the spikes. On the other hand, the same number of cache misses occurs under out-of-order execution, but there are no spikes because out-of-order execution is better suited to tolerate L1 misses. We believe that this kind of performance analysis can be expanded upon, and we are actively pursuing these types of studies because they offer a unique perspective on processor performance.

While Fourier techniques have been applied widely in many engineering and scientific fields, their application in computer architecture has been quite limited. One example, a 1981 paper by Voldman and Hoevel, used frequency spectra to analyze some rudimentary aspects of cache behavior [21]. This paper showed that program behavior, such as tight loops, could be identified from Fourier transforms of a memory reference trace. In this Voldman/Hoevel study, the waveform being studied is a signal based on per-memory-reference, rather than per-cycle data: each hit in the reference stream is a zero in the signal, and a miss in the reference stream is a one in the signal. Their approach precludes tying behaviors to true time durations and frequencies, nor does it look at the highly-parallel superscalar and out-of-order pipelines that are the norm today.

As noted in the previous section, the temperature models uses in our experiments are somewhat simplified, and further considering tangential heatflow between neighboring blocks and models for the heatsink/heatspreader would improve the accuracy of our temperature model [19]. While this complicates the RC network used to describe the thermodynamics of the processor, it still remains a linear time invariant system, and as such the frequency analysis described here would still be feasible. Instead of a single impulse response for individual nodes, one would generate multiple impulse responses which capture the effect of power dissipated at a particular site on all the other nodes in the system. Expanding our frequency domain analysis to more advanced thermal models is an area for future work.

The value of Fourier analysis is that it helps capture the relationship between repetitive events even if they are widely-spaced in the time domain. As such, the value of such analysis increases with the increasing subtleties and complexities inherent in modern processors. Sherwood *et al.* have used Bayesian techniques to identify repetitive phase behavior in programs [16, 15]. Although we have not shown examples here, we feel that such far-flung phase identification could also be achieved via a natural extension of the Fourier analysis we do here.

7 Conclusions

This paper proposes and evaluates spectral analysis as a means of characterizing and understanding processor power and performance. By analyzing from the frequency domain instead of the time domain, key periodic patterns embedded in complex processor behavior can be recognized and the underlying program characteristics can be identified.

We start from a review of Fourier transforms, the main spectral analysis method used in this paper. We then discuss how filter theory can explain the interactions between different components in a processor. In particular, two important power related phenomena, namely dI/dt power supply noise and thermal hotspots, can be analyzed using frequency and filter analysis. We provide two detailed case studies that demonstrate that frequency analysis techniques can be used to classify and characterize the inductive noise and thermal behavior or applications.

The dI/dt problem, an important power-related issue, is naturally linked to frequency analysis, because the power supply network can be characterized as a band-pass filter. Applying spectral analysis to the dI/dt problem, we find that with the packaging techniques used by modern microprocessors, current swings in mid-range frequencies (e.g. 50-200MHz) tend to have the most impact on supply voltage fluctuations. Moreover, with frequency-based analysis, we are able to develop a useful gauge of a benchmark's likelihood of posing dI/dt problems.

As a second case study, we demonstrate the strength of Fourier analysis in the thermal characterization of benchmarks. We model the thermal system as RC networks, and deduce that the frequency response of a thermal system is in fact a low-pass filter. This explains why power variations at high frequencies have virtually no impact on temperature, but at low frequencies, they can lead to bursty thermal behavior. Like in the dI/dt case, we can conveniently gauge the thermal profile of applications in the frequency domain.

In addition to the power based benchmark analysis, we discuss how one can use Fourier techniques for performance studies. We examine the ability of a processor to tolerate cache misses. An important strength of spectral analysis is the ability to concisely represent millions of cycles of complex time-domain behavior in the frequency domain, where key periodic patterns can be easily identified. This feature proves useful in performance as well as power studies.

In summary, this paper represents a first step in applying and evaluating spectral analysis as a way of characterizing program power and performance in modern superscalar processors. With processor design and application programs becoming more and more complex, we believe spectral analysis will prove to be a powerful way for analyzing and evaluating program behavior.

References

[1] P. J. Bannon. Personal communication, 2002.

[2] S. Borkar. Design challenges of technology scaling. *IEEE Micro*, 19(4):23–29, July 1999.

[3] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proceedings of the Seventh International Symposium on High-Performance Computer Architecture (HPCA-7)*, January 2001.

[4] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th International Symposium on Computer Architecture*, June 2000.

[5] D. Burger, T. M. Austin, and S. Bennett. Evaluating future microprocessors: the SimpleScalar tool set. Tech. Report TR-1308, Univ. of Wisconsin-Madison Computer Sciences Dept., July 1996.

[6] B. Fields, S. Rubin, and R. Bodik. Focusing processor policies via critical-path prediction. In *Proc. of the 28th International Symposium on Computer Architecture*, July 2001.

[7] E. Grochowski, D. Ayers, and V. Tiwari. Microarchitectural simulation and control of di/dt-induced power supply voltage variation. In *Proceedings of the Eighth International Symposium on High-Performance Computer Architecture (HPCA-8)*, February 2002.

[8] R. Joseph, D. Brooks, and M. Martonosi. Control techniques to eliminate voltage emergencies in high performance processors. In *Proc. of the 9th International Symposium on High Performance Computer Architecture (HPCA-9)*, February 2003.

[9] M. Kampe, P. Stenstrom, and M. Dubois. The FAB Predictor: Using Fourier Analysis to Predict the Outcome of Conditional Branches. In *Proc. Eighth International Symposium on High-Performance Computer Architecture (HPCA'02)*, Feb 2002.

[10] T. Mudge. Power: A first class design constraint. *Computer*, 34:52–57, April 2001.

[11] M. Oskin, F. T. Chong, and M. Farrens. Hls: Combining statistical and symbolic simulation to guide microprocessor design. In *Proc. of the 27th International Symposium on Computer Architecture*, June 2000.

[12] M. D. Powell and T. N. Vijaykumar. Pipeline damping: A microarchitectural technique to reduce inductive noise in supply noise, June 2003.

[13] E. Rohou and M. D. Smith. Dynamically managing processor temperature and power. In *2nd Workshop on Feedback-Directed Optimization*, Nov 1999.

[14] Semiconductor Industry Association. International Technology Roadmap for Semiconductors, 2001. http://public.itrs.net/Files/2001ITRS/Home.htm.

[15] T. Sherwood, E. Perelman, and B. Calder. Basic block distribution analysis to find periodic behavior and simulation points in applications. In *Proc. Intl. Conference on Parallel Architectures and Compilation Techniques (PACT 2001)*, Sep 2001.

[16] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *Proc. Tenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Oct 2002.

[17] K. Skadron, T. Abdelzaher, and M. R. Stan. Control-theoretic techniques and thermal rc-modeling for accurate and localized dynamic thermal management. In *Proceedings of the Eighth International Symposium on High-Performance Computer Architecture (HPCA-8)*, February 2002.

[18] K. Skadron and M. Martonosi. Final Report from NSF Workshop on Simulation and Evaluation Techniques for Modern Processors, 2001. http://www.ee.princeton.edu/ mrm/CPUperf.html.

[19] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecutre, June 2003.

[20] D. J. Sorin, V. S. Pai, S. V. Adve, M. K. Vernon, and D. A. Wood. Analytic evaluation of shared-memory systems with ilp processors. In *Proceedings of the 25th International Symposium on Computer Architecture*, June 1998.

[21] J. Voldman and L. W. Hoevel. The Software-Cache Connection. *IBM Journal of Research and Development*, 25(6):877–893, 1981.