Wavelet Analysis for Microprocessor Design: Experiences with Wavelet-Based dI/dt Characterization

Russ Joseph
Dept. of Electrical Eng.
Princeton University
rjoseph@ee.princeton.edu

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

Margaret Martonosi
Dept. of Electrical Eng.
Princeton University
mrm@ee.princeton.edu

Abstract

As microprocessors become increasingly complex, the techniques used to analyze and predict their behavior must become increasingly rigorous. This paper applies wavelet analysis techniques to the problem of dI/dt estimation and control in modern microprocessors. While prior work has considered Bayesian phase analysis, Markov analysis, and other techniques to characterize hardware and software behavior, we know of no prior work using wavelets for characterizing computer systems.

The dI/dt problem has been increasingly vexing in recent years, because of aggressive drops in supply voltage and increasingly large relative fluctuations in CPU current dissipation. Because the dI/dt problem has a natural frequency dependence (it is worst in the mid-frequency range of roughly 50-200MHz) it is natural to apply frequency-oriented techniques like wavelets to understand it. Our work proposes (i) an off-line wavelet-based estimation technique that can accurately predict a benchmark's likelihood of causing voltage emergencies, and (ii) an on-line *wavelet-based control technique that uses key wavelet coefficients to predict and avert impending voltage emergencies. The off-line estimation technique works with roughly 0.94% error. The on-line control technique reduces false positives in dI/dt prediction, allowing voltage control to occur with less than 2.5% performance overhead on the SPEC benchmark suite.*

1 Introduction

The dI/dt problem—exacerbated both by ongoing increases in CPU current fluctuations and by decreasing CPU supply voltages—has seen increasing attention from computer architects over the past two to three years. This attention is largely due to the increasing difficulties projected for producing cost-effective power supply and voltage regulation systems in upcoming generations of high-performance microprocessors. With an inadequate power delivery system, large swings in current cause ripples on the supply voltage lines that may cause circuits to fail [2].

The severity of these voltage ripples are sensitive to the frequency at which the current changes as well as the absolute magnitude of the swings. Parasitic inductances in the power supply network amplify certain types of current variations, making their impact particularly harmful [19]. For current electronic packaging materials, current variations within the range of 50-200MHz have the most impact [10]. At the same time variations at much lower and higher frequencies have less impact.

Due to the frequency sensitive nature of the dI/dt problem, it is tempting to consider frequency based analysis techniques to characterize its effects on high performance processors. One such frequency based approach, *wavelet analysis* has become an indispensable tool in many scientific disciplines because of its ability to identify how frequency components of a waveform change over time. In this paper, we demonstrate that wavelet analysis can be used in computer architecture studies; more specifically we demonstrate that these techniques are ideally suited for analyzing the dI/dt problem. The many strengths of wavelet analysis include: a capability for unified time-frequency analysis, rigorous mathematical definition, and computational efficiency (algorithmic complexity $O(n)$ for basic wavelet operations). These qualities are key to the two distinct applications we explore.

First, we present wavelet based analysis techniques that provide both efficient ways at characterizing the current variations of real programs and expressive means of quantifying the effect that these variations can have on a processor's power supply. Due to the growing importance of the dI/dt problem, architects need to understand what types of current variations appear in typical applications, what microarchitectural and program characteristics are responsible for the variations, and how these variations translate into voltage ripples.

Second, we propose an accurate and hardware efficient mechanism for monitoring voltage levels at runtime. This mechanism uses wavelet convolution methods to efficiently track the instantaneous processor voltage level based on the utilization of execution resources. Because the wavelet based methods are attuned to representing variation over different time scales, the numerous computations that would ordinarily be necessary to compute voltage at run-time dovetail into a reduced number of operations with wavelets. Our approach makes it easier to implement microarchitectural voltage monitoring in hardware.

Overall, the contributions of this work are as follows:

- To our knowledge, we are the first to present an application of wavelet transforms for microarchitectural analysis and design.

- We introduce wavelet analysis in the context of the dI/dt problem, and we show how wavelet representations can be used to automatically classify a program's susceptibility to dI/dt-induced supply voltage fluctuations.

- We show how wavelet-based characterizations illustrate the interplay of architectural events and power dissipation on different time scales. The presence of cache misses and other events are germane not just to performance issues, but also to the dI/dt problem. This work represents some of the first findings on these phenomena.

- We present a wavelet-based approach for identifying voltage levels at run-time. The wavelet factorization that we propose allows for effective voltage computation with modest hardware cost during execution. Wavelet-based control reduces complexity over previous full convolution methods, while offering superior performance compared to existing pipeline control schemes.

The remainder of this paper is structured as follows. Section 2 gives an overview of wavelet analysis and transforms, and Section 3 gives the needed background on our models for power supply networks and the processor being studied. Section 4 then presents our method for *off-line* estimation of voltage emergencies using wavelet analysis. Section 5 follows this with an *online* method for estimating voltage levels using a streamlined version of wavelet convolution. In Section 6, we discuss our work and relate it to other prior work, and in Section 7, we offer conclusions.

2 Wavelet Background

Wavelet analysis is a powerful method for decomposing and representing signals that has proven useful in a broad range of fields. Wavelet based techniques have been shown to asymptotically approach the optimal solutions for important types of problems including signal de-noising and compression [5]. Despite their widespread use in science and engineering, no one has used wavelet analysis for computer architecture studies. We show that many key properties of wavelet analysis are beneficial to dI/dt analysis.

Wavelet transforms are somewhat similar to Fourier transforms, in that they expose a function's frequency content. Fourier analysis begins with a waveform, a sequence of values indexed by time, and transforms this waveform into a sequence of coefficients which are indexed by frequency. In a similar manner, wavelet techniques can also be used to analyze a time indexed function and represent it as a group of frequency components. In addition, wavelet analysis also includes benefits that make it suitable for analysis of the dI/dt problem. A thorough comparison of wavelet and Fourier analysis techniques is beyond the scope of this paper, but in this brief overview we identify two key differences between wavelet analysis and Fourier analysis:

- Analysis Functions - Fourier analysis uses a single type of analysis function, the sinusoid, to represent a waveform and identify all of its frequency content, ranging from low-frequency oscillations to high-frequency noise. In contrast, wavelet analysis uses a pair of analysis functions: $\phi(t)$, the *scaling function*, interprets low-frequency information, and $\psi(t)$, the *wavelet function*, identifies high frequency information. Wavelet analysis allows one to chose the pair of analysis functions that best represent the signal rather than constraining analysis to a single type of function (e.g. sinusoid).

- Time-Frequency Localization - While Fourier analysis provides a single frequency decomposition for an entire signal, wavelet analysis also shows how frequency decomposition changes over time. While a Fourier representation of a signal $F(\omega)$ is indexed solely by frequency, ω, the wavelet representation of a signal is indexed by time-scale (inverse frequency) and by time-interval. With these two indices one can explore how frequency content evolves over time.

Some of the key characteristics of wavelet analysis make it ideally suited to dI/dt analysis. In particular, the choice of analysis functions and the time-frequency localization are instrumental to effectively representing dI/dt variation and determining what impact it will have on supply voltage noise. We briefly discuss how these qualities affect our characterizations.

Rather than determining a fixed function for analysis (e.g. sinusoid), wavelet analysis allows one to choose the pair of scaling $\phi(t)$ and wavelet $\psi(t)$ functions, collectively called a *wavelet basis* from an infinite set of functions. For dI/dt analysis, this allows us to choose functions that best capture the variations seen in processor current consumption. We found that the Haar analysis functions pictured in Figure 1 were attuned to the sharp discontinuities that appear in processor current consumption waveforms, so we use them exclusively in this paper.

Time-frequency localization is also important in accurately gauging the dI/dt severity of real programs. In general, programs exhibit *non-stationary* behavior, where program metrics including IPC, cache miss rates, branch prediction accuracy, and power consumption vary over time as the program progresses through long-lived phases [18]. For the dI/dt problem much smaller time scales are important (tens to hundreds of cycles), and it is important to identify how variations

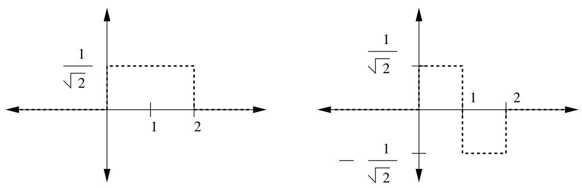


Figure 1. Haar scaling function $\phi(t)$ (left) and Haar wavelet function $\psi(t)$ (right).

<table>
<tr><td>D[0,0]</td><td>D[0,1]</td><td>D[0,2]</td><td>D[0,3]</td><td>D[0,4]</td><td>D[0,5]</td><td>D[0,6]</td><td>D[0,7]</td></tr>
<tr><td colspan="2">D[-1,0]</td><td colspan="2">D[-1,1]</td><td colspan="2">D[-1,2]</td><td colspan="2">D[-1,3]</td></tr>
<tr><td colspan="4">D[-2,0]</td><td colspan="4">D[-2,1]</td></tr>
<tr><td colspan="4">A[0]</td><td colspan="4">A[1]</td></tr>
</table>

Figure 2. The wavelet coefficient matrix. A[k]'s are approximation coefficients and D[j,k]'s are detail coefficients.

change during execution. Wavelet analysis makes this possible by introducing time-frequency localization. In addition to quantifying how large current variations are at different frequencies, wavelet analysis also tells how these variations change with respect to time. For example, this would allow one to distinguish intervals that are stressful for dI/dt from execution regions which are not.

$$A[k] = \int_{-\infty}^{\infty} 2^{\frac{j_0}{2}} \phi(2^{j_0} t - k) x(t) \qquad (1)$$

$$D[j,k] = \int_{-\infty}^{\infty} 2^{\frac{j}{2}} \psi(2^{j} t - k) x(t) \qquad (2)$$

The wavelet transform operations presented in Equations 1 and 2 produce two types of coefficients. Approximation coefficients, A[k] as computed by Equation (1), capture coarse-grained features. Detail coefficients, D[j,k], as computed by Equation (2), isolate fine-grained characteristics.

The scaling function, $\phi(t)$, is used to compute the approximation coefficients. The approximation coefficients are indexed by a single variable, k, which corresponds to time regions in the original signal $x(t)$. Each approximation coefficient can be thought of as a weighted average of $x(t)$ over a window of size 2^{j_0}. The *resolution level* j_0 allows one to chose a maximum granularity for wavelet analysis. For example, setting $j_0 = 2$ would mean that approximation coefficients would capture activities spanning no more than 4 time units, while $j_0 = 3$ would extend the analysis to windows of 8 time units.

While the scaling function and approximation coefficients isolate low frequency behavior, the wavelet function, $\psi(t)$ and the detail coefficients isolate higher frequency components. The calculation described in Equation 2 shows that the detail coefficients are indexed by two variables: j, which corresponds to frequency, and k, which corresponds to time. The j index isolates *time scales*. Increasing values of j identify fine granularity changes in $x(t)$, due to the 2^j time scaling factor. In addition, the k index isolates these frequency effects with respect to time windows which correspond to 2^j.

Together the detail and approximation coefficients capture localized frequency information about the original signal $x(t)$. Figure 2 shows one way to think about the relation between wavelet detail and approximation coefficients. First, the approximation coefficients cover large windows. Second, as the scale index j increases, more coefficients are needed because the granularity becomes finer. Analysis can be focused on a specific instant in time, an important property when dealing with bursty signals. Together, the wavelet detail and approximation coefficients can represent localized time and frequency effects.

The discrete wavelet transform has an extremely efficient implementation: the *fast wavelet transform* (FWT) has an algorithmic complexity of $O(n)$ [5]. Furthermore, wavelet representations are quite sparse. In other words, the majority of the terms in the coefficient matrices (e.g. Figure 2) are either zero or nearly zero. This is a useful property for many applications such as compression and de-noising [5].

The brief overview presented in this section is intended to familiarize the reader with general concepts involved in wavelet analysis, and it provides background for the wavelet applications we discuss in Sections 4 and 5. For an even more thorough description of the mathematics that underscore wavelet analysis, we refer readers to other resources [5, 1, 6].

3 Power Supply and Processor Models

Power supply design for high-performance processors is an extremely difficult task, and looming technology trends dictate that it will only become more taxing in terms of cost, design time, and overall complexity [17]. However, recent research has shown that microarchitectural voltage control can reduce the burden of traditional power supply design [8, 11, 14, 16]. Our research here extends on this prior work by using wavelet analysis to estimate voltages and predict voltage emergencies. In this section, we first describe models for the power supply network and microprocessor that help us to explore how program characteristics and hardware design impact current dissipation and voltage oscillations.

3.1 Power Supply Model

If not adequately addressed, parasitic inductance found in power supply networks can produce large voltage ripples that have an adverse effect on reliability and performance [2]. Typically, a CPU's supply voltage must be maintained within a +/-5% V_{DD} operating range to prevent circuits from encountering timing or noise-induced error. We term instances where the voltage stays beyond the allowed operating levels *voltage faults*. While the impact of inductive noise can be reduced somewhat by decoupling capacitors [10, 19], trends [17] suggest that it may be difficult and costly to protect against voltage faults with traditional mitigation approaches.

In this paper, we model the power supply system and the inductive noise it generates with a linear systems model [9]. The second order linear system model that we use captures the mid-frequency effects that are most pressing for dI/dt [10]. Figure 3 shows the frequency profile for a second order linear system. Our supply model has a resonant frequency of 50MHz, which matches empirically measured data for a recent high-performance microprocessor [20]. As a starting point for an impedance value, we created a worst-case dI/dt execution sequence as in [11] and chose the largest value of peak impedance that would maintain the processor voltage within +/-5% V_{DD} under that worst case scenario. For such a system no runtime dI/dt control is necessary, but the cost and complexity of designing the system could be problematic. Microarchitectural dI/dt control mechanisms, such as [8, 11], are meant to work with less complicated power supply networks which may have larger peak impedances. For the studies in this paper, we chose an impedance which is 50% larger than the maximum allowed impedance without dI/dt control. This is a reasonable challenge for the dI/dt control mechanism because this less sophisticate power supply could allow voltage ripples as large as +/-7.5% V_{DD}.

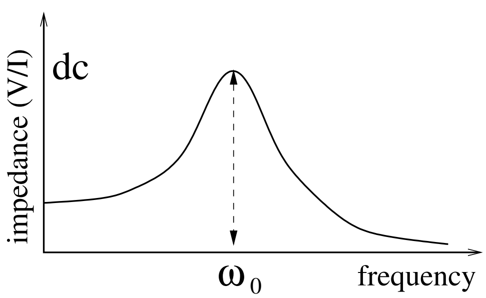


Figure 3. Frequency response of a second-order linear system, which models a typical power supply system.

3.2 Processor Model and Benchmarks

For our processor model, we used Wattch [3], a widely used architectural power simulator based on Simplescalar [4]. We modified Wattch to simulate a 3.0GHz processor with a nominal V_{DD} of 1.0V executing the Alpha 21264 architecture. Table 1 presents the parameters we used. We modified Wattch/Simplescalar to model the performance/energy impact of deep pipelines including multiple fetch and decode stages. We also updated Wattch to spread the power usage of pipelined structures over multiple stages. To compute per-cycle current, we divided the per-cycle power from Wattch by the supply voltage. For our choice of V_{DD} = 1.0V, one watt of power consumed corresponds to one ampere of current drained. When the supply voltage drops, the current consumed by devices on chip decreases, so the active elements on chip may actually dampen the voltage ripples somewhat. However, the same assumptions are used by power supply designers in early stage planning [2], and are considered good, conservative estimates.

For evaluations, we use all 26 SPEC integer and floating-point benchmarks. To ensure that we observed representative behavior, we used simulation points presented in [18]. These simulation points were automatically chosen to capture as much of the true program behavior as possible while reducing simulation time.

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	250 cycle latency

Table 1. Processor Parameters

4 Wavelet Variance Characterization

We describe a wavelet based approach to quantifying the severity of current variations and a statistical model that determines their likely impact on voltage variations. Our approach uses wavelet transforms and properties of the wavelet coefficients to characterize a current consumption pattern and assess the impact of dI/dt noise on the power supply as a function of variance. Variance is a common statistical metric that measures the “spread” of data points around a mean value [13], and it is useful in expressing how much of a dI/dt stressor an individual benchmark is. Furthermore, we show how probability analysis can use these variance estimates to quantify how likely the voltage is to approach an extreme level.

4.1 Representing dI/dt Severity with Wavelets

Our wavelet characterization and variation model is based on two key observations. First, patterns in the wavelet coefficients describe current variation with respect to both time and frequency, and can be used to determine the impact on voltage fluctuations. Secondly, most of the localized current consumption patterns in programs follow a Gaussian probability distribution and as a result the majority of localized voltage variation patterns also follow a Gaussian distribution. Both of these observations are important to effectively characterize the dI/dt behavior of a program, so we discuss their individual impact before describing our full approach to dI/dt characterization.

Because the power supply network is a linear system, the magnitude of the voltage variation (the output of the system) is proportional to the magnitude of current variation (the input of the system) [7]. With this insight, we developed an empirical model that related qualities of current consumption to voltage variance via wavelets. The wavelet representation allows us to isolate the different frequency components of the current consumption waveform. This is an important step since each wavelet scale corresponds to a distinct frequency range and has its own impact on voltage variations.

Our empirical model determines how much each wavelet scale level contributes to the total voltage variance and adds those individual contributions to assess the severity of the dI/dt noise. We found the correlation between adjacent wavelet coefficients to be helpful at identifying the oscillatory behavior which is most harmful for dI/dt [11]. For each wavelet level, we constructed a table that related the correlation of adjacent wavelet coefficients to total voltage variance. This table was constructed by calculating the voltage variance on the power supply under several input current waveforms whose wavelet coefficients had different correlation factors. Figure 4(a) shows the effect that adjacent coefficient correlation can have on voltage variance. A strong correlation between wavelet coefficients appears and is indicative of a pulse pattern which can be more severe. Little to no correlation suggests lack of a pattern (i.e. white noise).

While the correlation analysis identifies the effect that patterns within a scale level have on dI/dt noise, wavelet variance (the sum of squared wavelet coefficients) quantifies the magnitude of the current swings. Because the power supply network is a linear system, variance in the amount of current consumed is proportional to the variance on voltage level. Figure 4(b) shows how wavelet variance is related to voltage variance in our model. The sum of squared wavelet coefficients on a scale level, j, is the wavelet variance of current, σ_C, for that scale level. We use the correlation factor described above and the scale level, j, to index a lookup table that determines how wavelet variance scales to voltage variance, σ_V. The lookup table places heavier weightings on scale levels close to the resonant frequency of the supply network and to correlation factors that indicate resonance is occurring.

To relate the magnitude of the voltage variation to the likelihood that the voltage drops or rises below of a given value, we fit voltage profiles to statistical distributions. Specifically, we observed that within localized regions (lasting tens to hundreds of cycles), the current consumed by the processor and the supply voltage level seen by the CPU were approximately Gaussian.

Our wavelet characterization scheme uses the key observations we described to identify the severity of dI/dt current variation and project its impact on voltage levels. Figure 5 shows pseudocode that outlines the steps used to perform dI/dt analysis. First the discrete wavelet transform is applied to a cycle-

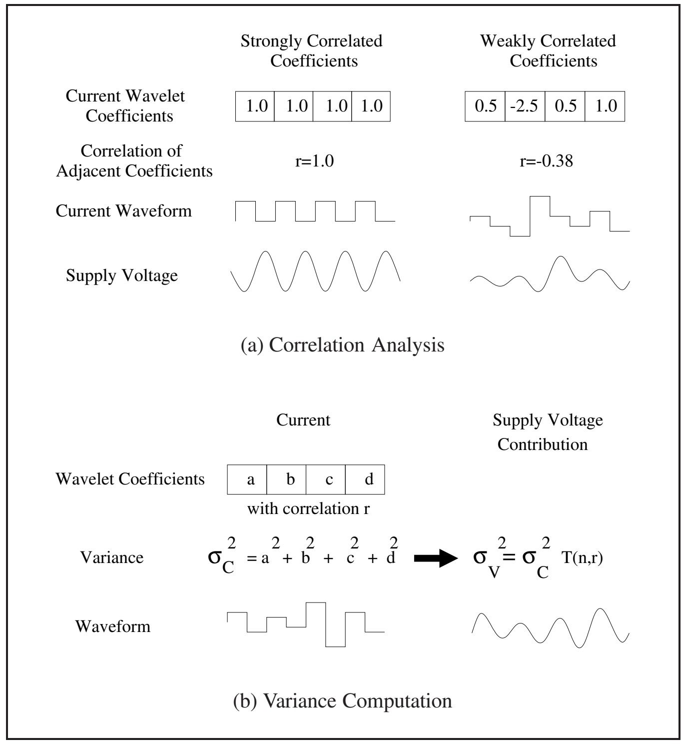


Figure 4. Computations for modeling the voltage variation contributed by wavelet scale level n. The top (a) shows correlation analysis and how two different adjacent coefficient correlation factors can influence voltage. The bottom (b) shows how one can compute the contribution to voltage variation, σ_V^2, using wavelet variance, σ_C^2 and the adjacency correlation, r.

by-cycle current waveform which is read from a microarchitectural power simulator (e.g. Wattch). In our studies we used a window size of 256 cycles because it was long enough to capture important dI/dt variations which typically span tens of cycles. However, this window size was also short enough to capture localized current consumption patterns. We used eight scale levels in our analysis, the maximum number allowed under this window sizing. After performing the wavelet transform to collect approximation and detail coefficients, the procedure then determines how much each scale contributes to voltage variance using the wavelet variance, adjacency correlation, and lookup table. Then the individual scale contributions are added to produce a total voltage variance estimate. To calculate the mean voltage level, we average the approximation coefficients, and multiply result by the DC resistance of the power supply network. In a linear system, the mean output value (voltage) is the product of the mean input value (current) and the DC value of the system's frequency response (resistance) [7]. Finally, the procedure projects the estimated voltage mean and variance onto a Gaussian probability model to assess how often the voltage will achieve an extreme level.

```
characterize_window(current_window[]) {
  /* perform wavelet transform */
  [approx[], detail[][]] =
    discrete_wavelet_transform( current_window[] )

  /* compute contribution of all scales */
  for(i=1 to max_scale) {
    /* compute adjacent coefficient correlations */
    r[i] =
       adjacent_coefficient_correlation(
         detail[i][] )
    /* compute wavelet variance */
    wave_var[i] = 0
    for(j=1 to max_components( detail[i][] ) {
      wave_var[i] += detail[i][j] * detail[i][j]
    }
    /* find scale's voltage variance */
    volt_var_contrib[i] =
      correlation_lookup( r[i] ) * wave_var[i]
  }

  /* add all variances */
  volt_var_total = 0
  for(i=1 to max_scale) {
    volt_var_total += volt_var_contrib[i]
  }

  /* fit to Gaussian distribution */
  volt_mean = average( approx ) *
    power_supply_dc_resistance
  volt_prob_distrib = gaussian_distrib( volt_mean,
     volt_var_total)
}
```

Figure 5. Pseudocode outlining the dI/dt characterization process.

4.2 Results: Voltage Characterization

Using the voltage estimation scheme outlined in the previous section, we profiled SPEC 2000 benchmarks to estimate the impact of dI/dt induced voltage variation. One of the uses of voltage profiling is to gauge the severity of voltage oscillations so that we can estimate how often a given program will require dI/dt control. Through experimentation we found that 0.97V was a reasonable threshold setting for a dI/dt controller, and hence voltage levels below 0.97V would need to be controlled to prevent voltage low faults. In Figure 6, we compare the percentage of execution cycles actually spent below 0.97V to the estimated percentage of cycles spent below that point using the scheme described in Section 4.1. Overall the root mean square for error is 0.94%. Figure 6 shows that while our estimates do not exactly determine the number of cycles spent below 0.97V, they do a good job at determining whether or not a benchmark might be problematic for dI/dt. For example it identifies mgrid, gcc, galgel, and apsi as benchmarks that spend at least 3% percent of their execution below 0.97V. It also identifies benchmarks such as vpr, mcf, equake, and gap which spend less than 0.5% of their execution time below this control point. Overall, wavelet voltage estimates are useful for identifying the severity of voltage variations.

4.3 Results: Relating Voltage Variation to Architectural Events

One of the interesting aspects of off-line, wavelet-based estimates is that they can be used to offer insights as to the impact of different microarchitectural events on voltage levels and voltage variability. As an example of this, we characterized 26 SPEC benchmarks regarding their voltage variance and we compared it to several microarchitectural events. The clearest relationship was between L2 cache misses and voltage variance. Our variance analysis of wavelet window segments shows that low L2 cache misses correlates strongly with Gaussian voltage distributions.

In particular, Figure 7 shows voltage histograms two benchmarks: crafty and mcf. Crafty has few L2 cache misses. Visually, one sees that the voltage profile for this benchmark is approximately Gaussian. In contrast, the histograms for mcf, which has a high L2 miss rate shows a prominent spike at the nominal supply voltage level 1.0V, and does not exhibit a Gaussian shape.

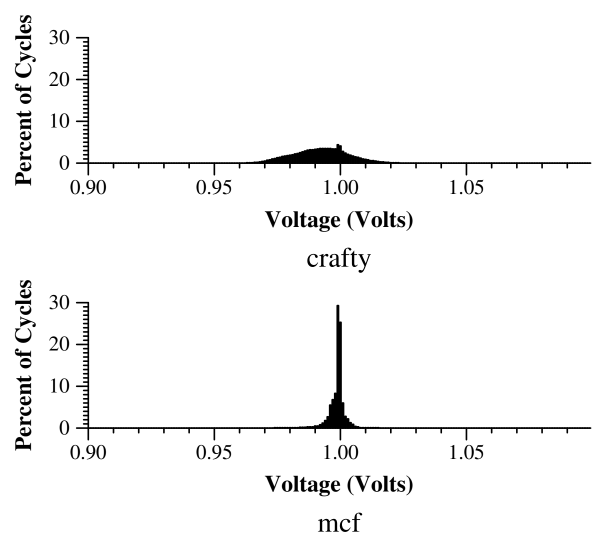


Figure 7. Histogram of cycles spent at different voltage levels for two SPEC benchmarks: crafty (top) and mcf (bottom).

Moving from a visual level to a statistical level, Figure 8 shows a statistical test of "Gaussian-ness" applied to execution windows from the 26 SPEC benchmarks. In particular, we used a Chi-Square test at 95% significance to check for Gaussian behavior in execution windows of 64-cycles in each benchmark. The benchmarks with high L2 cache misses are the least likely to show Gaussian behavior in voltage. This is intuitive because these benchmarks tend to spend long periods of time waiting for L2 misses being serviced, followed by spikes of activity when the data returns. In contrast, programs with fewer cache misses have smoother execution profiles and thus are closer to Gaussian in their current and voltage profiles.

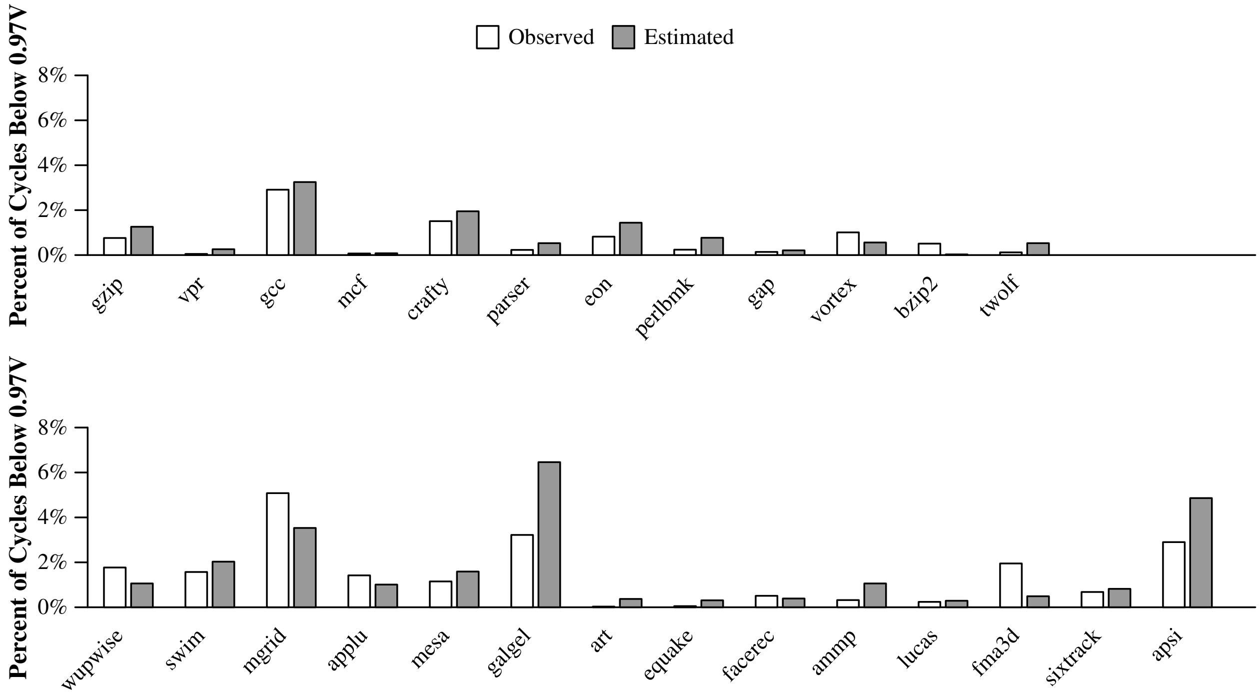


Figure 6. Estimated percent of cycles below control point compared to observed number of cycles.

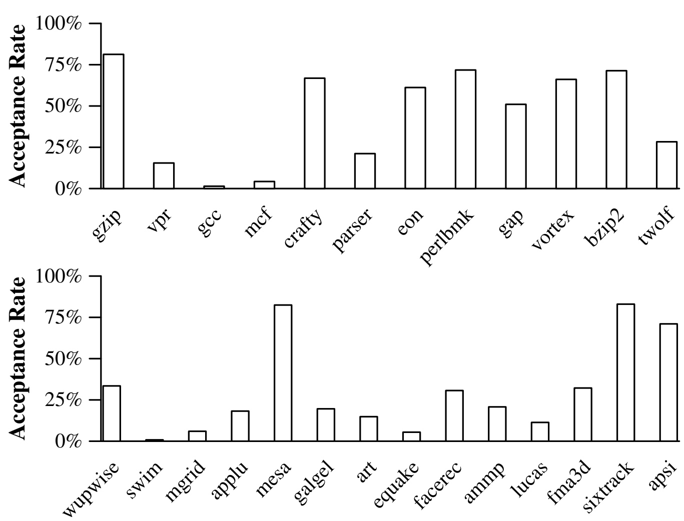


Figure 8. Percentage of 64 cycle execution windows in which the cycle-by-cycle current consumption was determined to be Gaussian under the Chi-Square 95% significant test. We present SPEC integer (top) and floating-point (bottom).

5 Wavelet Based dI/dt Control

In the previous section, we demonstrated that off-line wavelet based statistical models can help to characterize dI/dt behavior and determine when problematic current fluctuations will influence voltage levels inside the processor. In this section, we focus our attention on on-line dI/dt control. In particular, we demonstrate a wavelet-based voltage monitor that can determine how close the processor is to a voltage fault by tracking current variations.

Previous work on architectural control techniques to limit inductive noise have had one of two fundamental strategies: (1) directly or indirectly monitor the *voltage level* and use the voltage level to trigger a reactive microarchitectural control mechanism [8, 11] or (2) estimate the *current* consumed by the processor by tracking microarchitectural events and maintaining invariants on the allowable change in current over a relevant time window [15, 16]. Under both of these approaches, normal execution operations must be suspended to avoid a voltage fault, but this may have an adverse effect on performance and energy-efficiency. For example, if a control point is reached, both types of control mechanisms stall instruction issue to prevent the voltage from dropping below the minimum value. This decreases the current draw, so that the voltage will not sink further, but it may reduce performance since ready instructions are not being issued. Conversely, rising voltages are the result of very low current draws. In this case, both control strategies issue no-ops to increase the current consumption.

Control techniques based on voltage monitors can have relatively small performance and energy impacts, but accurate voltage monitoring can be difficult to implement. Since voltage-based monitors directly track the quantity that ultimately determines whether or not an error can occur, voltage monitoring schemes are unlikely to induce a false positive, e.g. stall instruction issue when a voltage emergency is not imminent. Due to this, control is only likely to be initiated when it is necessary, minimizing performance and energy impact. On the other hand, the complexity of previous voltage sensing proposals is high. In [11] the authors suggest using an analog circuit to sense voltage levels. While today's chips have increasing amounts of analog circuits, the added complexity of integrating a mixed analog/digital design on die might be problematic. Another recent proposal using a convolution based voltage monitor, suffers from implementation difficulties as well. The problem with this approach is that a large number of convolution terms are needed to accurately track voltage level and such hardware is difficult to build with 1-2 cycle delays.

Control schemes that monitor current consumption are easier to build. For example,in [16], the authors propose a mechanism called *pipeline damping*, where the hardware maintains the current consumed over a sufficiently long history. They impose a restriction on the difference in current between cycles of a specified window length. By choosing a sufficiently small delta, they can bound the maximum dI/dt swing. The hardware complexity to implement this is small, but this scheme may produce a significant number of false positives.

The wavelet-based control scheme that we present here is designed to have few false positives and to have an efficient implementation. It allows the microarchitecture to efficiently track voltage levels at run-time, allowing for a dI/dt controller that avoids voltage emergencies without compromising performance or energy-efficiency. The wavelet representation significantly decreases hardware complexity so that we can achieve the higher accuracy of a voltage monitor, but with a more feasible implementation than previously proposed convolution voltage monitors [8].

5.1 Wavelet-Based Voltage Monitors

The convolution operation is a standard signal processing technique that is fundamental to the microarchitectural voltage monitor presented in [8]. Equation 3 shows how instantaneous voltage can be computed as a function of the amperage consumed over previous cycles. The time shifted values of the current, $i(t)$ are weighted by the impulse response $h(t)$, which captures the complete behavior of a linear system [12]. The proposed voltage monitor presented in [8] tracks the current consumption $i(t)$ and performs the multiplication and addition in hardware.

$$v(t) = \sum_{k=0}^{N} i(t-k) * h(k) \tag{3}$$

In principle the wavelet voltage monitors that we propose are similar to standard convolution voltage monitors, but they can drastically reduce the amount of on-line computation, which simplifies the hardware implementation. The voltage monitor we propose is based on *wavelet convolution* which can operate on a reduced number of wavelet coefficients [21]. For our studies, we found that approximation coefficients were most helpful for estimating voltage levels, so we focus on wavelet convolution which works solely with approximation coefficients.

Rather than multiplying cycle-by-cycle current consumption against the full impulse response, wavelet voltage monitors track wavelet coefficients of current consumption and multiply against wavelet coefficients of the impulse response. The wavelet representation allows neighboring terms to be factored together, effectively reducing the number of terms needed in the full convolution.

The reduced number of convolution terms could also decrease the accuracy of the voltage monitor, which means that a real implementation would have to balance hardware complexity with error rate. Large amounts of error are intolerable because they dictate that control thresholds be moved closer to V_{DD} to ensure that the control mechanisms can respond to an impending voltage emergency. This hurts performance and energy-efficiency because false positives are introduced.

To help assess the relationship between the number of coefficient terms and error, Figure 9 plots the maximum possible error versus the number of coefficients under different resolution levels. We found that resolution levels 2 to 5 offered the best trade-offs between the number of coefficients and error, so we focus on this range. The resolution levels determine the number of cycle by cycle current terms represented in each coefficient. For example, under resolution level 5, a total of 2^5 or 32 cycle terms are represented in a single coefficient. Figure 9 shows that in general increasing the number coefficient terms can reduce the error. This is intuitive because increasing the number of coefficients essentially increases the history of the current being monitored, allowing one to capture more current variation in the past. For resolution levels 4 and 5, additional coefficients do not improve the error level beyond 11 and 7 coefficients, respectively. Essentially, these resolution levels focus on variations which are coarse-grained. As a consequence, additional coefficients do not help much since the existing coefficients already capture a significant amount of history. The coefficients under resolution levels 2 and 3 focus on fine-grained detail, so they achieve better results if given enough coefficients to capture history.

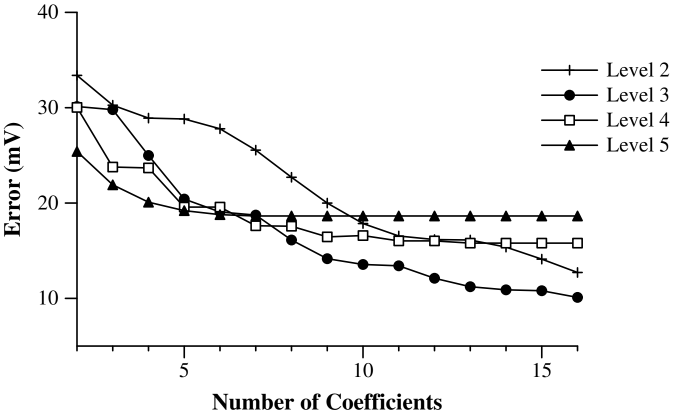


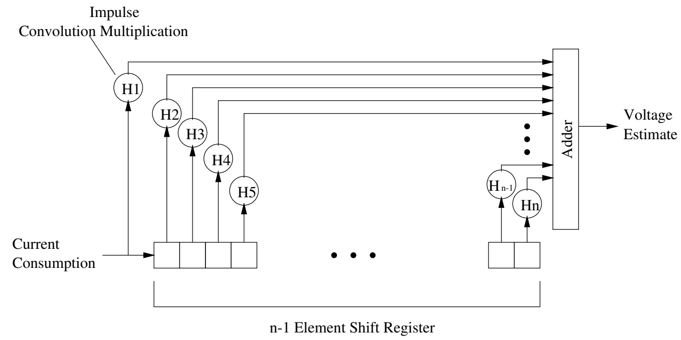
Figure 9. Maximum error under different wavelet convolution for different resolution levels and numbers of coefficients.

5.2 Implementation

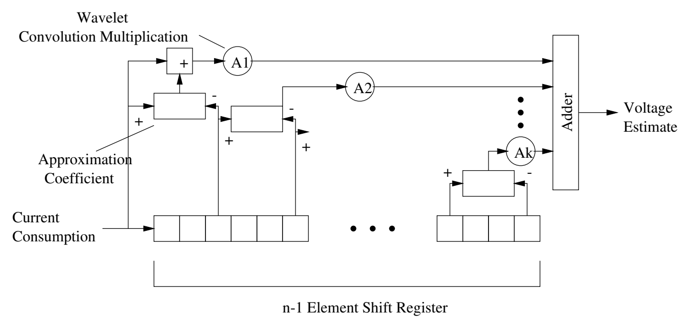
The operations in wavelet convolution are similar to standard convolution since past current consumption levels are multiplied by weighted constants, but the key difference is that approximation coefficients for current consumption and impulse response are used. Since fewer terms are involved in the convolution, the hardware complexity is reduced. Figure 10 shows how wavelet convolution compares to a standard convolution implementation. As in [8], we assume that clock-gate signals from execution units can be used to determine the instantaneous current consumption. For both standard and wavelet convolution implementations, the current consumption values are fed into a shift register. In a standard convolution implementation, cycle-by-cycle current consumption values are multiplied by the impulse response, and the products are added to compute the instantaneous voltage.

The key difference in wavelet convolution is the use of approximation coefficients. As Figure 10 shows, the shift register can be used to update the wavelet approximation coefficients. As the cycle-by-cycle current consumption values move through the shift register, their contributions to coefficients are added and subtracted. This avoids the overhead of directly performing the wavelet transform each cycle. The current consumption approximation terms are then multiplied by approximation coefficients of the impulse response. The products are then added to compute the voltage level.

Wavelet convolution significantly reduces the amount of hardware needed to monitor voltage levels. For example, a full convolution with an impulse response of length N would require a total of N multiplications followed by the addition of N terms. For a resolution level of j, a wavelet voltage monitor would have $k = \lceil \frac{N}{2^j} \rceil$ convolution terms. This reduces the number of operations needed for convolution multiplication and addition steps, in exchange for the $2k$ add/subtract operations needed to update the approximation coefficients. We believe that this can be an effective trade-off.



(a) Full convolution



(b) Wavelet convolution

Figure 10. Voltage monitor implementations which use full convolution (top) and wavelet convolution (bottom).

5.3 Results

To quantify the efficacy of the proposed control scheme, we evaluated wavelet voltage monitors under three different parameter choices which reflect varying amounts of hardware complexity and consequently different error levels. Specifically, we chose wavelet based voltage monitors with 5, 10, and 15 coefficients. Using Figure 9 as a guide to minimize the error for a given number of coefficients, we chose a resolution level of 5 for the 5 coefficient case and a resolution level of 3 for the 10 and 15 coefficient cases.

We found that under a full convolution implementation (no error), threshold levels of 0.96V for the low control point and 1.04V for the high control point were sufficient to guarantee effective dI/dt control under the worst case with the power supply described in Section 3. Under SPEC2000, the mean performance loss compared to an uncontrolled processor was 0.4%.

Using the estimates in Figure 9, we increased the voltage low control point and decreased the voltage high control point to ensure that dI/dt control would be robust under error for the three wavelet monitor configurations that we examined.

Larger error moves that the control points closer to V_{DD} and further away from the real emergency levels. Thus we would expect that larger error values would imply more false positives and hence greater performance loss.

Figure 11 shows the impact that wavelet based control had on our three wavelet control configurations. As intuition suggests, the configuration with the least amount of error, 15 coefficients, suffers the least performance loss. Conversely, the 5 coefficient case which had the largest error, also witnessed the greatest slowdowns. Nevertheless, even the performance impact under 5 coefficients is very small, with a 2.2% mean performance loss. This compares favorably against a recent proposal which tracked differences in current consumption over a window of cycles [16]. This is possible because voltage based control schemes directly track the quantity of interest to the dI/dt problem. Although a full convolution implementation could be expensive in terms of hardware cost, wavelet based monitors are an efficient means to implement microarchitectural voltage calculation.

6 Discussion and Related Work

The topic of dI/dt control began to see attention at the microarchitectural level only in the past two to three years. Essentially there are two main parts to any microarchitectural dI/dt controller. The first part is the sensing mechanism used to determine when trouble is imminent. The second part is the actuation or control mechanism used to take action in order to keep the system's voltage under control.

While the name "dI/dt problem" refers to current fluctuations, it is ultimately the *voltage* fluctuations induced by current changes that are problematic in high-performance microprocessors. Thus, in building a sensing mechanism for dI/dt, one can choose between sensing current, sensing voltage, or sensing some proxy of the two and doing estimation calculations. Current or voltage sensors can be built as analog devices. Current sensors are more readily build-able, while supply voltage sensors are more difficult due to the fact that they are trying to measure V_{DD} itself, though all other on-chip logic typically treats V_{DD} as the bedrock reference value on the chip.

Some prior work has looked at estimation-based proxies for current and voltage. In particular, pipeline damping [16] proposes using current estimation over time windows to determine whether to engage voltage control. While this method is relatively simple to implement, it has the potential for high false-positive rates. High false-positive rates mean that voltage control mechanisms must be engaged more frequently, which leads to potentially large performance and energy impact as well. (Their paper mentions performance slowdowns as large as 22% for SPEC benchmarks, which are not significant dI/dt stressors.) The convolution-based methodology proposed by Grochowski et al. [8] has the potential to be more accurate in its cycle-by-cycle voltage estimates and thus have a lower false positive rate. On the other hand, it is difficult to build a single-cycle implementation of the convolution circuit they propose. Our work offers the low false-positive rate of an accurate sensing circuit, with an easier implementation than full-blown convolution hardware.

7 Summary

Wavelet analysis is a powerful method of decomposing and representing signals in both the frequency and time domains. Compared to traditional Fourier analysis, wavelet analysis has the following advantages:

- Wavelet analysis can analyze signals that contain discontinuities and sharp spikes.
- Wavelet analysis can analyze non-stationary signals whose frequency behavior varies with time.
- Wavelet coefficient matrices are typically sparse. Most coefficients are zero or near zero, so that a small group of coefficients can represent a signal fairly well.
- Wavelet analysis is computationally efficient. A fast wavelet transform can be done in O(N) time.

While wavelet analysis has been widely applied in science and engineering, no published work has shown its application in the computer architecture field. In this paper, we propose the application of wavelet analysis in microprocessor design, and specifically we show how to use wavelets to characterize and estimate voltage variation on chip.

Our first application of wavelet analysis is to characterize the voltage variance of a particular program workload. Voltage variance is a measure of how cycle-by-cycle voltage values spread around the nominal voltage. Large voltage variances are undesirable because they are indicative of dI/dt problems which lead to reliability issues. To calculate voltage variance we developed an empirical model that related wavelet coefficients to dI/dt noise. Wavelet analysis decomposes the original current consumption waveform into time and frequency components, making it easier to identify variations at key frequencies which may be harmful for dI/dt.

Our second application of wavelet analysis is a wavelet-based voltage monitor that is more computationally efficient than a full convolution. This is possible because only a few wavelet coefficients are needed to achieve a reasonable accuracy. Online dI/dt control based on a wavelet-based voltage monitor can eliminate voltage emergencies while limiting performance loss to a few percent. Because of the regularity of the Haar wavelet, the coefficients can be computed efficiently using a shift register implementation.

In summary, this paper represents a first attempt to apply wavelet analysis to the computer architecture field. Because of its power to represent bursty signals and sharp spikes, as well as its computational efficiency, wavelet analysis can be a powerful aid to computer architects in understanding and analyzing complex program and microprocessor behavior.

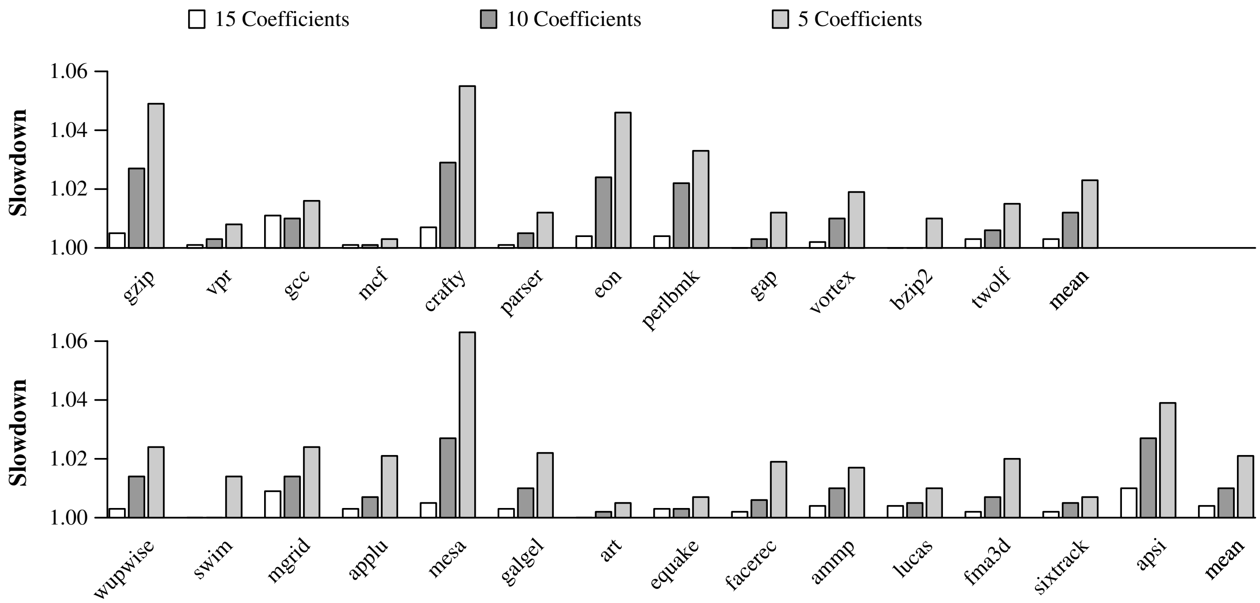


Figure 11. Slowdown for wavelet control with varying number of coefficients for integer (top) and floating point (bottom) benchmarks.

References

[1] An introduction to wavelets. *IEEE Computational Sciences and Engineering*, 2(2):50–61, June 1995. http://www.amara.com/IEEEwave/IEEEwavelet.html.

[2] D. Blaauw, R. Panda, and R. Chaudhry. Design and analysis of power distribution networks. In A. Chandrakasan, W. J. Bowhill, and F. Fox, editors, *Design of High-Performance Microprocessor Circuits*, pages 499–522. IEEE Press, 2001.

[3] 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.

[4] 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.

[5] C. S. Burrus, R. A. Gopinath, and H. Guo. *Introduction to Wavelets and Wavelet Transforms : A Primer*. Prentice-Hall, 1998.

[6] I. Daubechies. The wavelet transform, time-frequency localization and signal analysis. *IEEE Trans. Inf. Theory*, 36:961–1005, 1990.

[7] A. L. Garcia. *Probability and Random Processes for Electrical Engineering*. Addison-Wesley, 1994.

[8] 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.

[9] S. Haykin and B. V. Veen. *Signals and Systems*. John Wiley and Sons, 1999.

[10] D. J. Herrell and B. Beker. Modeling of power distribution systems for high-performance microprocessors. *IEEE Transactions on Advanced Packaging*, 22(3):240–248, August 1999.

[11] 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.

[12] T. Kailath. *Linear Systems*. Prentice-Hall, 1980.

[13] E. Kreyszig. *Advanced Engineering Mathematics*. John Wiley and' Sons, 8th edition, 1999.

[14] M. D. Pant, P. Pant, D. S. Wills, and V. Tiwari. Inductive noise reduction at the architectural level. In *Proceedings of the Thirteenth International Conference on VLSI Design*, January 2000.

[15] M. Powell and T. N. Vijaykumar. Pipeline muffling and a priori current ramping:microarchitecture techniques to reduce high-frequency inductive noise. In *Proceedings of the 2003 International Symposium on Low Power Electronics and Design (ISLPED '03)*, August 2003.

[16] M. D. Powell and T. N. Vijaykumar. Pipeline damping: A microarchitectural technique to reduce inductive noise in supply noise. In *Proceedings of 30th International Symposium on Computer Architecture (ISCA-30)*, June 2003.

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

[18] 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.

[19] L. D. Smith, R. E. Anderson, D. W. Forehand, T. J. Pelc, and T. Roy. Power distribution system design methodology and capacitor selection for modern cmos technology. *IEEE Transactions on Advanced Packaging*, 22(3):284–291, August 1999.

[20] M. Tsuk et al. Modeling and measurement of the Alpha 21364 package. In *Proc. 2001 IEEE Topical Meeting on Electrical Performance of Electronic Packaging (EPEP)*, Oct. 2001.

[21] P. P. Vaidyanathan. Orthonormal and biorthonomal filter banks as convolvers, and convolutional coding gain. *IEEE Transactions on Signal Processing*, 41(6), June 1993.