Pipeline Muffling and A Priori Current Ramping: Architectural Techniques to Reduce High-Frequency Inductive Noise

Michael D. Powell and T. N. Vijaykumar
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907
{mdpowell,vijay}@purdue.edu

ABSTRACT

While circuit and package designers have addressed microprocessor inductive noise issues in the past, multi-gigahertz clock frequencies and billion-transistor-level integration are exacerbating the problem, necessitating microarchitectural solutions. The large net on-die decoupling capacitance used to address this noise throughout the chip consumes substantial area and can cause a large leakage current. This paper proposes microarchitectural techniques to reduce high-frequency current variability, reducing the need for decoupling capacitors. We observe that we can control inductive noise by reducing current variability either in *space* (i.e., variability in usage of circuit blocks) or in *time* (i.e., variability within a circuit block across clock cycles).We propose pipeline muffling, a novel technique to reduce changes in the number of resources being utilized by controlling instruction issue, trading off some energy and performance to control di/dt in space. We also extend a previous technique, which incurs performance and energy degradation, and propose a priori current ramping to allow the current of a resource to ramp up ahead of usage, with virtually no performance loss, and ramp down immediately after usage, with little energy loss. Our techniques *guarantee* a worst-case bound on the di/dt, which is required to reduce the demand for decoupling capacitors, saving area and reducing leakage.

Categories and Subject Descriptors

C.1.0 [**Processor Architectures**]: General

General Terms

Reliability.

Keywords

Inductive Noise, Pipeline Muffling, A Priori Current Ramping, Decoupling Capacitors, Leakage.

ISLPED'03, August 25-27, 2003, Seoul, Korea.

1 INTRODUCTION

Inductive noise (Ldi/dt) is a voltage glitch at power/ground connections due to a large current spike in a span of time (di/dt) flowing through the wire/substrate inductance (L) of the power and ground rails [5]. Coupled with narrow noise margin, the inductive noise induced by current spikes in the processor circuitry degrades data integrity causing reliability problems [2]. As key structures such as the issue queue and caches get taller and wider, more processor signals switch simultaneously causing larger current spikes. Low-power techniques such as clock gating exacerbate current spikes by gating components on and off [12, 8, 6, 10]. While circuit and package designers have addressed microprocessor inductive noise issues in the past, multi-gigahertz clock frequencies and billion-transistor-level integration are exacerbating the problem, necessitating microarchitectural solutions.

To prevent current spikes from becoming voltage spikes (inductive noise), designers want processor pipelines to see a low impedance over a wide range of frequencies. They place capacitors off-package, on-package, and on-die to provide a low impedance at low-frequencies (in the MHz range), medium frequencies (in the tens to hundreds of MHz), and high-frequencies (near the processor clock frequency) [9, 8]. This paper focuses on high-frequency inductive noise, which largely occurs due to variation in resource usage and is exacerbated by clock gating.

To address the high-frequency inductive noise, large on-die decoupling capacitors must be placed close to resources with high current variability, such as caches and functional units. If the decoupling capacitance near a high-variability resource is insufficient, the parasitic wire resistance and inductance between the resource and some farther-away decoupling capacitance delay compensating for current spikes, causing noise. The demand for large, distributed on-die decoupling capacitance results in a substantial total capacitance. The Pentium II included 180 nF of on-die decoupling capacitance while the Alpha 21264 included 320 nF [13]. The total requirement for decoupling capacitances increases with I/V_{dd} [13], indicating an increase in decoupling capacitance with future high-power (i.e., high-current), low-V_{dd} microprocessors.

Large on-die decoupling capacitance is a concern for two reasons. First, the capacitance can consume up to 10% of die area [13]. Second and more important, the capacitance can cause a large leakage current. A recent Design & Test roundtable discussed the technology-scaling issue of increasing leakage through the capacitors, and

Table 1: Instruction schedule for 10 ALU operations on a previously idle processor.

Cycle #	Conventional	Pipeline Muffling, $\delta = 4$	A priori current ramping
1	Issue 8	Issue 4	Issue 8
2	Issue 2	Issue 6	Issue 2, Ramp up current: 8 * (1/2)
3	First 8 execute	First 4 execute, issue 2 fake operations	Execute 8
4	Second 2 execute	Second 6 execute	Execute 2, Ramp down current: 6 * (1/2)
5		downward muffling: 2 fake executions	Ramp down current: 2 * (1/2)

Preston, a participant from Compaq, claimed that amperes of current flow quiescently through the capacitors, complicating I_{DDQ} testing and increasing chip power consumption [7].

This paper proposes two microarchitectural techniques to reduce high-frequency current variability. We observe that we can control inductive noise by reducing current variability either in space (i.e., variability in usage of circuit blocks) or in time (i.e., variability within a circuit block across clock cycles). We propose *pipeline muffling* to reduce changes in space by controlling instruction issue and limiting changes in the number of resources utilized. [12] proposed ramping resource current over time to reduce the rate of current change in individual resources. [12] degrades performance due to delaying of instructions by a few cycles to ramp up clock-gated circuits, and degrades energy due to waiting (i.e., not clock-gating for) a few cycles before ramping down to avoid future ramp-up delays. We propose *a priori current ramping*, an extension of [12], to allow the current of a resource to ramp up *ahead* of usage, with virtually no performance loss, and ramp down *immediately* after usage, with little energy loss.

While circuit techniques like on-die capacitors attempt to cure current variations by preventing them from becoming voltage variations (noise), we prevent the current variations at the source. Our techniques *guarantee* a worst-case bound on the di/dt (as opposed to reducing the average), which is required to reduce the demand for decoupling capacitors, saving area and reducing leakage.

The main contributions of this paper are:

- We propose pipeline muffling which controls instruction issue and limits the number of used resources to increase (or decrease) only within a pre-specified delta, trading performance (and energy) for reduction in worst-case variability. Muffling guarantees a 50% reduction in worst-case inductive noise in execution units with an processor energy-delay penalty of 3%.
- We propose a priori current ramping which allows time for resource current to ramp up a few cycles ahead of utilization, and to ramp down immediately after utilization. For the same reduction in noise as above, [12] incurs a processor energy-delay penalty of 44%, while a priori ramping incurs a 2% penalty.

2 RELATED WORK

2.1 Contrast to previous architectural proposals

Previous architectural proposals have spread out current by ramping resources up and down over time to avoid large steps in current [12, 15]. We discussed [12] in Section 1. [15] applies ramping to only floating-point units.

Other architectural proposals have also focused on medium frequency inductive noise, which occurs over tens to hundreds of cycles (10-100 MHz) [1, 8]. Because ramping current over that many cycles would be impractical, these techniques focus on controlling noise over space rather than time. [16] uses the compiler to control resource utilization in a VLIW processor; this technique will not be effective in an out-of-order system. [8] uses current convolution to determine which resources may be utilized and control changes in current. [10] uses voltage sensors to react to noise and change resource utilization accordingly. [8] and [10] may have delayed response due to computation time and sensor delay, respectively, as mentioned by their authors. These techniques are not applicable to high-frequency noise for which immediate response is critical. [14] also proposes architecturally controlling medium-frequency noise, but our paper focuses solely on high-frequency inductive noise.

2.2 Di/dt modeling and circuit solutions

A number of recent papers model power distribution and inductive noise. [9] models a processor power distribution network and evaluates effectiveness of on-die and on-package capacitors. [6] uses a microarchitectural simulator to show step power profiles for SPEC 2000 benchmarks for a clock-gated microprocessor. [13] proposes a design technique for distributing on-die decoupling capacitance. They note that decoupling capacitance for a circuit block should be allocated in proportion to the power of that block. They evaluate inductive noise in caches and integer units and show their technique reduces transmission of noise between these circuit blocks for the Pentium II and Alpha 21264.

Capacitance allocation based on the maximum power of a circuit block, as in [13], occurs because designers assume a step from zero to maximum current. This assumption makes capacitance (C) directly proportional to the worst-case rate of current change (di). Architecturally *guaranteeing* a reduced di within a functional block reduces the maximum step and the inductive noise, correspondingly reducing the need for decoupling capacitance. In the next section, we introduce two techniques for providing these guarantees.

3 REDUCING HIGH-FREQUENCY NOISE

Because high-frequency inductive noise is distributed throughout the chip, circuit designers use distributed capacitors to address inductive noise in individual circuit blocks and to avoid one block affecting others [13]. Accordingly, our techniques address inductive noise in a distributed fashion by addressing noise within indi-

vidual circuit blocks, rather than in the whole chip. In this section, we discuss pipeline muffling and a priori current ramping.

3.1 In Space: Pipeline Muffling

Pipeline muffling controls resource-current variability in space by employing two components, *upward* and *downward*, which limit increases and decreases in current respectively between adjacent cycles to a pre-specified delta (δ). Though we discuss them separately, it is important to note that both upward and downward pipeline muffling must be implemented together to provide a bound on inductive noise.

Upward pipeline muffling smooths increases in resource utilization by regulating large increases, trading performance for inductive noise reduction. Utilization increases are restricted to those causing current increases less than δ. If operations that would cause current to increase faster than δ are available, some of the operations must wait until the next cycle. Upward muffling, however, does not restrict peak utilization. Once resource current has increased to within δ of the maximum, maximum utilization (and current) are allowed.

Downward pipeline muffling smooths decreases in resource utilization by padding decreases with "fake" operations. These fake operations prevent current from decreasing faster than δ. As the fake operations exist solely to prevent a large drop in current and perform no useful work, they trade extra energy for reduction in inductive noise.

We illustrate the concepts of pipeline muffling in integer ALUs for the out-of-order pipeline shown in Figure 1 with an example shown in the second and third columns of Table 1. Assume that a previously idle, 8-issue, out-of-order processor is ready to issue 10 integer ALU operations, and we wish to reduce worst-case di/dt by 50%. A conventional processor issues those instructions without regard to inductive noise as in the second column of Table 1. If we wish to reduce noise by 50%, we set δ equal to the current of 4 ALU operations, meaning the number of ALU operations can neither increase nor decrease by more than 4 between adjacent cycles. Because the processor was previously idle, it issues only 4 instructions on the first cycle (for an increase of 4). On the second cycle, it issues the remaining 6 instructions (for an increase of 2). On the third cycle, two "fake" ALU operations are issued to prevent the current from dropping by more than 4 units.

Upward and downward pipeline muffling can be implemented in the issue stage of an out-of-order processor. Conventional processors use select logic to issue ready instructions to a finite number of available resources. Pipeline muffling augments the select logic to include the δ conditions for the controlled resources, possibly altering the number of available resources. With pipeline muffling, the number of available resources for the next cycle changes depending on the number of resources in use the current cycle. The information about the number of resources in use can be computed in the previous cycle; therefore this additional condition can be folded into select without substantial delay in the logic. Small integers (e.g. 2-4 bits) are used for the δ conditions to simplify the necessary logic in the pipeline.

FIGURE 1: Out-of-order pipeline stages.

Finally, because pipeline muffling controls variability based on predetermined, constant estimates of resource current, inaccuracies in the estimation are a concern. For example, an estimator may assume that because high-performance circuits are implemented in dynamic logic and dynamic logic power is dominated by the clock, this assumption is reasonable. Though clock power is dominant, some variability will still occur due to different input bits.

Even in the presence of estimation inaccuracies, it is possible to use pipeline muffling to establish current variability bounds. If the current consumed by a utilized resource is estimated at δ but actually may be x% higher or lower, then the actual maximum variability for that resources is an increase from the minimum current, (1 - x/100)δ, to the maximum current (1 + x/100)δ. The total worst case variability is then (1 + 2x/100)δ. For example, if the actual current change of a resource could be 20% higher or lower than the estimated bound, δ, then the actual current bound would be 1.4δ instead of δ. By knowing in advance the maximum error in the current estimate, a δ that will lead to a suitable actual current bound may be chosen.

3.2 In Time: A Priori Current Ramping

A priori current ramping exploits the observation that in modern processor pipelines, certain resource usage information is known a few cycles of ahead of the usage. For instance, the number of ALUs that will be in use is known at the end of instruction issue. Because there is *at least* one cycle of register read between issue and execute, as shown in Figure 1, a priori ramping uses that one-cycle gap to ramp up previously-idle clock-gated functional units, *without incurring any performance penalty as occurs in [12]*. If the resource becomes idle after use, it's current may then be ramped down immediately because there is no performance penalty associated with ramping it up again. While ramping does introduce a small energy penalty, it is less than that of letting idle functional units consume full current to avoid future ramp-up penalty, as is done in [12]. A similar observation was made by Deterministic Clock Gating (DCG) [11]. DCG, however, uses the ahead-of-time knowledge to perform clock gating and not reduce inductive noise.

We illustrate a priori current ramping in the fourth column of Table 1 for a previously idle system with 10 ready ALU operations. On cycle 1, 8 operations are issued. On cycle 2, the 8 operations are in register read and 8 ALUs are ramped up, consuming half of their maximum current. The final 2 operations are also issued this cycle. On cycle 3, the first 8 operations are executed while the last 2 are in register read. On cycle 4, 2 ALUs execute the last 2 operations, while 6 ALUs ramp down their current. At this point execution has completed with the same schedule as the base processor (the second column of Table 1), meaning there is no performance loss. Finally, on cycle 5, the last 2 ALUs ramp down their current.

Table 2: System parameters.

instruction issue	8, out-of-order, 128-entry RUU
Issue queue/ROB	128 entries, 1-cycle reg. file
L1 caches	64K 2-way, 2 cycle, 2 ports
L2 cache	2M 8-way, 12 cycles
Memory latency	80 cycles
Fetch	up to 8 instructions/cycle with 2 branch predictions per cycle
Int ALU & mult/div	8 & 2
FP ALU & mult/div	4 & 2

There are some limitations to the application of a priori current ramping. To avoid a performance penalty, a priori current ramping is restricted to using intervening cycles between issue and execute (for ALUs), and issue and the d-cache (for loads/stores). Assuming an aggressive 1-cycle register file means only one cycle is available for current ramping for ALUs. If resource current is brought to half of the active current during the register read-cycle, a 50% reduction in maximum di/dt is achieved for that resource. A two-cycle register file would allow for current to ramp over 2 cycles, corresponding to a 67% reduction in noise.

A priori current ramping can also be applied only to those resources whose schedule is known at least one cycle in advance of utilization, such as ALUs and the d-cache. A priori current ramping cannot be applied to register read, for example, because there is no time between issue and register read to ramp-up register read, as shown in Figure 1. A priori current ramping can be expected to be applicable to back-end resources, however, because of the presence of register read after issue. While it is possible to move the register read stage before the issue stage, doing so requires that operands be stored in the issue queue, adding substantial complications.

Pipeline muffling and a priori current ramping are orthogonal and therefore may be applied at the same time for additional inductive noise savings. For example, if a priori current ramping is limited to a 50% noise reduction (because there is only one intervening cycle between issue and execute), but a 75% reduction is desired for the integer ALUs, pipeline muffling may also be applied with a δ that reduces variability by 50%. The maximum variabilities multiply, resulting in a final maximum variability that is 25% of that in the base case.

Table 3: Latencies and relative integral current estimates within circuit blocks.

Component	latency (cycles)	current per cycle
Integer ALUs		
Add/Multiply/Divide	1/3/12	14/7/1
Floating-Point ALUs		
Add/Multiply/Divide	2/4/12	6/3/1
I- and D-caches		
I-cache/D-cache	2/2	1/1

4 METHODOLOGY AND RESULTS

In this section, we present our methodology and results. Our first results compare pipeline muffling, a priori current ramping, and the current ramping scheme in [12]. Second, we show combinations of pipeline muffling and a priori current ramping applied to ALUs, the d-cache, and the i-cache.

4.1 Methodology

Table 2 shows the configuration for the simulated system which uses the out-of-order pipeline shown in Figure 1. We modify SimpleScalar 3.0b [4] and Wattch [3] and simulate a high-performance, out-of-order microprocessor executing the Alpha ISA. We use 23 of the 26 applications in the SPEC 2K benchmark suite (*ammp*, *mcf*, and *sixtrack* are excluded due to simulation time), fastforwarding 2 billion instructions to skip initialization code, and then running 500 million instructions. The base IPC for each application is shown under the graph in Figure 2. To estimate di/dt, we extend Wattch to compute current for each cycle in addition to energy based on component level activity similar to the procedure in [6]. To enable calculation of per-cycle current, we spread the execution energy of multi-cycle functional units and pipeline events (e.g., cache accesses and multiplies) over each of the relevant cycles. We use our simulator to determine performance degradation, energy penalty, and current variability for the various techniques.

We estimate high-frequency current variability within several microprocessor circuit blocks: the i-cache, the d-cache, and integer and floating-point functional units. We convert current to small integral estimates to be used in computing δ for pipeline muffling, as using floating-point values would make the computation cumbersome in a real implementation. Current is based on the number of active units within the circuit block, and the estimates are shown in Table 3. For the i-cache and d-cache, the current estimate is simply the number of ports in use. For functional units, we determine the relative per-cycle current of each integer (or floating point) operation relative to the other integer (floating point) operations based on Wattch's current models. While Wattch's estimates may have some error, muffling is tolerant of such inaccuracies, as discussed in Section 3.1.

4.2 Comparing techniques

In this section, we compare pipeline muffling, a priori current ramping, and the conventional ramping technique of [12] for configurations which achieve the same reduction in inductive noise. We configure the three techniques to achieve a 50% reduction in noise within three circuit blocks: the integer ALUs, floating-point ALUs, and the d-cache.

In Figure 2 we show performance degradation (black sub-bars, scale on left) and relative energy-delay *(*full-height bars, scale on right) for the three techniques over 23 SPEC 2000 applications. All values are relative to a processor with no inductive noise reduction. Because the individual benchmarks behave similarly, our discussion is based on the averages as shown in Figure 3. Figure 3 shows average performance degradation (x-axis), total processor energy penalty (y-axis), and relative energy-delay (text aside) for the three techniques over the 23 SPEC 2000 applications.

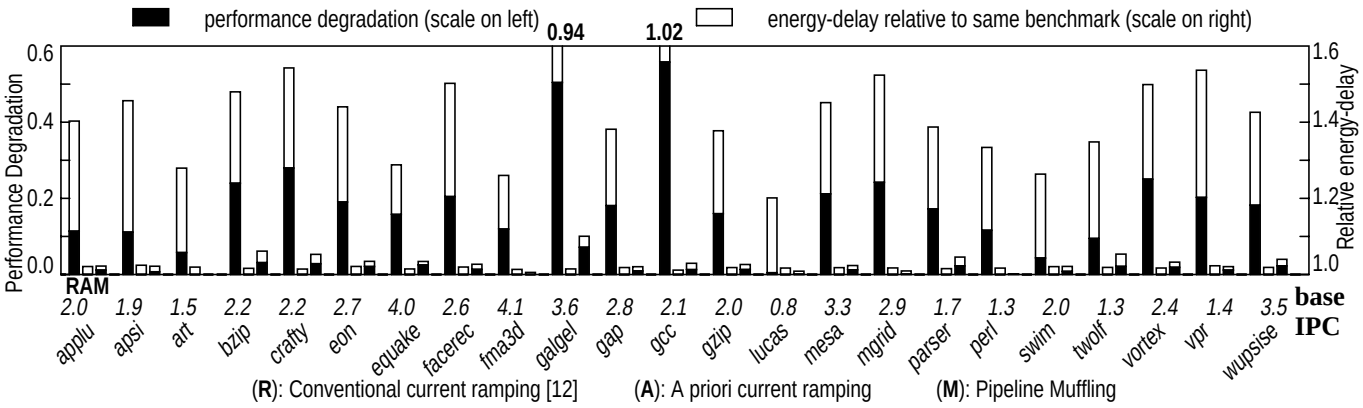

FIGURE 2: Base IPC and comparison of techniques for all applications.

We see that conventional current ramping from [12] incurs a 19% performance degradation compared to 1.5% for pipeline muffling and 0% for a priori current ramping. Unlike the ramping scheme in [12], pipeline muffling incurs performance degradation only for resource-utilization changes that would cause a current change greater than δ (e.g., an increase from 0 to 6 integer adds); therefore the performance loss is small. Because a priori current ramping ramps current before the resources are needed, it incurs no performance loss.

The performance degradation for conventional current ramping is substantially larger than reported by [12] because we are applying the technique to *all* integer, floating-point, and d-cache resources. [12] does not report to which specific resources they apply their technique.

The figure indicates similar energy penalties of 1.4% and 1.8% for muffling and a priori ramping. Though the difference is small, it occurs because a priori ramping uses extra energy to ramp down the current of *any* unit that becomes idle, while muffling uses extra energy only for fake operations to prevent large decreases that would violate δ. Both techniques have substantially lower energy penalties than the 21% of the [12] technique. The energy penalty for [12] occurs almost entirely due to leaving resources active for ten cycles after they are last used to avoid delay due to future ramp up. This procedure is unnecessary in a priori ramping, which ramps up resource current without inserting delay.

4.3 Combination of techniques

In this section, we show the effects of combining pipeline muffling and a priori current ramping. Recall from Section 3.2 that noise reduction from a priori current ramping is limited by the number of cycles between the issue stage and resource utilization and that a priori ramping can be applied to only those resources whose usage is known in advance. We show that pipeline muffling can be combined with a priori current ramping to increase inductive noise savings without incurring the performance penalty of using muffling alone. Because a priori current ramping avoids performance degradation, a design combining the techniques will reduce noise more than a priori ramping and have smaller performance degradation than a design using muffling alone. We also apply pipeline muffling to pipeline resources that cannot use a priori ramping.

Table 4: Combination of techniques

Configuration	Perf. loss %	Energy loss %
75% noise reduction: Int ALUs, FP ALUs		
muffling alone	1.1	0.8
combined	0.1	1.3
50% noise reduction: i-cache		
muffling	1.0	0.3
50%: i-cache. 75%: Int ALUs, FP ALUs, d-cache		
combined	1.9	3.1

Table 4 compares results of using muffling *alone* to achieve a 75% inductive noise reduction (top section, first row) against the combination (second row) for the integer and floating-point ALUs. The combination uses pipeline muffling to reduce inductive noise in the ALUs by 50% and a priori current ramping to reduce noise by a further 50%, for a total of 75%. From the table, we see that there is virtually no performance degradation for the combined technique, in contrast to 1.1% for muffling alone. However the combined technique has a total energy penalty of 1.3%, 0.5% higher than that of muffling alone. This extra energy is expected for the same reason a priori ramping had higher energy than muffling in the previous section.

Previous results have shown inductive noise reduction only in the pipeline back-end. It is possible to apply pipeline muffling to the front-end as well, specifically to the i-cache. Our two-ported i-cache, which is normally accessed every cycle, experiences current variability during a cache miss. This current variability can be reduced by 50% using pipeline muffling by accessing only one port during the cycle after a miss returns and by performing a fake i-cache access to one port in the cycle after a miss (i.e., setting δ =

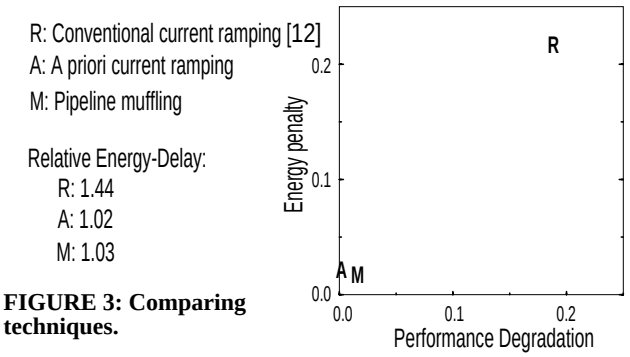

FIGURE 3: Comparing techniques.

1). A priori ramping is not applicable because we cannot determine in advance when a cache miss will return, as either an L2 miss or main memory access may occur. The middle section of Table 4 shows results for pipeline muffling applied to provide a 50% reduction in inductive noise in the i-cache only. I-cache misses are rare in the simulated benchmarks, and both performance and energy degradation are small.

Finally, we show results for pipeline muffling and a priori current ramping combined through several pipeline stages. The last section of Table 4 shows results when pipeline muffling is applied for a 50% inductive noise savings in the i-cache, integer ALUs, floating-point ALUs, and d-cache with a priori current ramping added in the integer ALUs, floating-point ALUs, and d-cache for a total savings of 75% in those three current blocks. The total performance degradation is 1.9% with an energy degradation of 3.1%, combining for an energy-delay penalty of 5.1%.

4.4 Reducing Decoupling Capacitance

As discussed in Section 2.2, [13] estimates the amount of on-die decoupling capacitance to be proportional to the worst-case current variability which is equal to maximum current in a conventional design. We reduce current variability by up to 75% in circuit blocks (i.e., L1 caches and ALUs) whose maximum currents aggregate to 30% to 45% of total processor current. Therefore we estimate the total on-die decoupling capacitance reductions achieved by our techniques to be 22% to 34%. Because decoupling capacitance area and leakage are proportional to the amount of capacitance, reductions in area and leakage would be similar.

5 CONCLUSIONS

We propose two techniques, pipeline muffling and a priori current ramping, to reduce high-frequency inductive noise within circuit blocks of a microprocessor. Pipeline muffling controls instruction issue and limits changes in the number of used resources to a prespecified delta, trading performance and energy for reduction in worst-case inductive noise. Muffling guarantees a 50% reduction in worst-case inductive noise in execution units with an processor energy-delay penalty of 3%. A priori current ramping allows time for resource current to ramp up a few cycles ahead of utilization and to ramp down immediately after utilization. A priori current ramping guarantees a 50% reduction in worst-case inductive noise with a 2% energy penalty and virtually no performance loss. We estimate total on-die decoupling capacitance reductions achieved by our techniques to be up to 22% to 34%. Our techniques will grow in importance in future technologies as decoupling-capacitance leakage worsens.

ACKNOWLEDGMENTS

This research is supported in part by NSF under CAREER award 9875960-CCR, NSF Instrumentation grant CCR-9986020, SRC contract 2000-HJ-768, DARPA contract F33615-02-1-4003, an Intel Ph.D. fellowship, and an NSF Graduate Research Fellowship.

REFERENCES

[1] W. Becker, H. Smith, et al. Mid-frequency simultaneous switching noise in computer systems. In *1997 Electronic Components and Technology Conference*, pp. 676–681, 1997.

[2] D. Boggs. Breathing life into a paper tiger. In *Keynote speech: 33rd International Symposium on Microarchitecture*, Dec. 2000.

[3] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *27th International Symposium on Computer Architecture*, pp. 83–94, June 2000.

[4] D. Burger and T. M. Austin. The SimpleScalar tool set, version 2.0. Technical Report 1342, Computer Sciences Department, University of Wisconsin–Madison, June 1997.

[5] Y. S. Chang, S. K. Gupta, and M. A. Breuer. Analysis of ground-bounce in deep sub-micron circuits. In *Proceedings of the VLSI test symposium*, pages 110–116, 1997.

[6] W. El-Essawy, D. H. Albonesi, and B. Sinharoy. A microarchitectural-level step-power analysis tool. In *International Symposium on Low Power Electronics and Design*, pages 263–266, Aug. 2002.

[7] T. Gabara, C. Nicol, T. Ning, R. Preston, N. Shanbhag, and C. Svensson. Power delivery and distribution roundtable. *IEEE Design and Test of Computers*, pages 98–103, Oct–Dec 2000.

[8] E. Grochowski, D. Ayers, and V. Tiwari. Microarchitectural simulation and control of di/dt-induced power supply voltage variation. In *Eighth International Symposium on High Performance Computer Architecture (HPCA)*, pages 7–16, Feb. 2001.

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

[10] R. Joseph, D. Brooks, and M. Martonosi. Control techniques to eliminate voltage emergencies in high-performance processors. In *Ninth International Symposium on High Performance Computer Architecture (HPCA)*, pages 79–90, Feb. 2003.

[11] H. Li, S. Bhunia, Y. Chen, T. N. Vijaykumar, and K. Roy. Deterministic clock gating to reduce microprocessor power. In *Ninth International Symposium on High Performance Computer Architecture (HPCA)*, pp. 113–122, Feb. 2003.

[12] M. D. Pant, P. Pant, D. Willis, and V. Tiwari. An architectural solution for the inductive noise problem due to clock-gating. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 255–257, 1999.

[13] M. D. Pant, P. Pant, and D. S. Wills. On-chip decoupling capacitor optimization using architectural level prediction. *IEEE Transactions on VLSI Systems*, 10(3):319–326, 2002.

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

[15] A. Tang, N. Chang, S. Lin, W. Xie, S. Nakagawa, and L. He. Ramp up/down floating point unit to reduce inductive noise. In *Lecture Notes in Computer Science*, volume 2008, pages 291–321, 2001.

[16] H.-S. Yun and J. Kim. Power-aware modulo scheduling for high-performance vliw processors. In *Proceedings of the International Symposium on Low Power Electronics and Design*, Aug. 2001.