Integrated Microarchitectural Floorplanning and Run-time Controller for Inductive Noise Mitigation

MICHAEL B. HEALY, IBM Corporation
FAYEZ MOHAMOOD, The MathWorks
HSIEN-HSIN S. LEE and SUNG KYU LIM, Georgia Institute of Technology

In this article, we propose a design methodology using two complementary techniques to address high-frequency inductive noise in the early design phase of a microprocessor. First, we propose a noise-aware floorplanning technique that uses microarchitectural profile information to create noise-aware floorplans. Second, we present the design of a dynamic inductive-noise controlling mechanism at the microarchitectural level, which limits the on-die current demand within predefined bounds, regardless of the native power and current characteristics of running applications. By dynamically monitoring the access patterns of microarchitectural modules, our mechanism can effectively limit simultaneous switching activity of close-by modules, thereby leveling voltage ringing at local power-pins. Compared to prior art, our di/dt alleviation technique is the first that takes the processor's floorplan, as well as its power-pin distribution, into account to provide a finer-grained control with minimal performance degradation. Based on the evaluation results using 2D floorplans, we show that our techniques can significantly improve inductive noise induced by current demand variation and reduce the average current variability by up to 7 times, with an average performance overhead of 4.0%. In addition, our floorplan reduces the noise margin violations using our noise-aware floorplan by an average of 56.3% while reducing the decap budget by 28%.

Categories and Subject Descriptors: B.7.2 [**Integrated Circuits**]: Design Aids—*Placement and routing*

General Terms: Design, Reliability

Additional Key Words and Phrases: Floorplanning, microarchitecture, power supply noise

ACM Reference Format:
Healy, M. B., Mohamood, F., Lee, H.-H. S., and Lim, S. K. 2011. Integrated microarchitectural floorplanning and runtime controller for inductive noise mitigation. ACM Trans. Des. Autom. Electron. Syst. 16, 4, Article 46 (October 2011), 25 pages.
DOI = 10.1145/2003695.2003706 http://doi.acm.org/10.1145/2003695.2003706

1. INTRODUCTION

High-performance, power-conscious microprocessors exhibit varying current demands, depending on the execution characteristics of a given program. For a high-frequency microprocessor, any abrupt change in current demand (referred to as dI/dt) will result in high-frequency inductive noise that leads to voltage ringing in the power-supply network. In the worst case, unreliable supply voltage can flip data values, resulting in incorrect computation. Processors are often over-designed, with the use of excessive

This work is supported by the National Science Foundation under CAREER grant CCF-0546382, the Center for Circuit and System Solutions (C2S2), and the Interconnect Focus Center (IFC).
Author's addresses: M. B. Healy, IBM Research, Yorktown Heights, NY; email: mbhealy@gatech.edu; F. Mohamood; email: fayez.mohamood@gmail.com; H.-H. S. Lee, School of Electrical and Computer Engineering, Georgia Institute of Technology; email: leehs@ece.gatech.edu; S. K. Lim, School of Electrical and Computer Engineering, Georgia Institute of Technology; email; limsk@ece.gatech.edu.

amounts of decoupling capacitors (decap) to address this reliability issue. For increasingly complex processors, inserting an excessive amount of decaps enlarges the chip area and, at the same time, exacerbates the leakage power problem. Moreover, significant design effort and cost is inevitably expended to manage the infrequent cases where programs exhibit the maximum level of varying current demands during the course of execution.

Traditional technology scaling has exacerbated the problem as well. Decreasing supply voltages reduce the absolute noise margin, and increasing transistor counts and higher frequencies result in more power dissipation. Aggressive power-saving techniques like clock-gating and/or power-gating are widely studied and applied. Processors such as the Intel Pentium 4, Pentium M, and IBM Power5 [Jacobson et al. 2005] use different levels of clock-gating to dynamically disable portions of the circuit. The industry has acknowledged the dI/dt issue due to the extensive application of clock-gating and responded with architectural solutions. For instance, the L2 cache in the Power5 processor uses progressive clock-gating in different cache banks to mitigate the dI/dt effect [Jacobson et al. 2005].

Conventionally, the worst-case current consumption can be profiled and gauged by exercising power virus programs [Grochowski et al. 2002]. These programs are written expressly to vary the execution behavior of the microprocessor to induce drastic current-demand fluctuations. Designers then allocate an appropriate amount of decap to manage these worst-case fluctuations in a repetitive process until the reliability targets are met. The main drawback of this design technique is that a significant amount of chip area is devoted to these decaps, which only cover those infrequent corner cases. For example, the designers of the Alpha 21264 reported that roughly 15 to 20% of the die area is occupied by decaps.

To address these shortcomings in the worst-case design methodology, we advocate a design procedure that builds inductive noise awareness into the complete processor design cycle. Our proposed methodology involves two components. The first component involves using microarchitectural feedback to perform noise-aware floorplanning. With this floorplanner, we introduce the following innovations.

—Two metrics called *self-switching weight* and *correlated switching weight* for identifying modules that are highly likely to cause large dI/dt problems.
—A simulated-annealing-based floorplanning algorithm that incorporates microarchitectural feedback for module placement. Combined with dynamic control at the microarchitectural level to eliminate the worst-case scenario, it enables designers to focus on the average-case current variability.

Our floorplanner allows the design of a floorplan that is inherently noise-tolerant. However, it still cannot guarantee reliable operation during the worst-case noise scenario. To prevent the worst-case scenario, we also introduce a low-cost dI/dt controlling mechanism embedded in the microarchitecture that will dynamically limit high-frequency dI/dt. This design can be integrated into the microarchitecture during the early planning stages to facilitate the design of a processor for the average-case current consumption scenario. This dynamic dI/dt controller is the second component of our design methodology, which features the following:

—Decay counters used as a simple mechanism to monitor the access pattern of each microarchitectural module to prevent unsteady self-switching activity.
—A novel microarchitectural technique using a *queue-based dynamic dI/dt controller* to prevent simultaneous (or correlated) Gating of modules that share the same local power-pins on the power delivery network.

—*Preemptive ALU gating* that is integrated into our queue-based dynamic dI/dt controller to avoid performance loss and further reduce high-frequency dI/dt noise.
—An enhancement that performs progressive clock-gating to achieve fine-grained dI/dt control for large modules without violating the current demand threshold.

Unlike prior techniques [Grochowski et al. 2002; Joseph et al. 2004; Pant et al. 2000; Powell and Vijaykumar 2003, 2004], which largely aim to provide chip-level dI/dt control, our technique monitors and controls dI/dt by leveraging the spatial information of modules obtained from a given floorplan and its power-pin distribution. This is the primary reason why our floorplanning algorithm and dynamic controller compliment each other perfectly. Inductive noise is highly dependent on the chip floorplan, which determines the relative location of functional modules and their distance from the power-pins. Hence, a solution at the chip-level is too coarse-grained and cannot account for the fact that certain power-pins are unaffected by a distant module. For the same reason, such designs are also likely to generate many false alarms, resulting in undesired performance degradation. In contrast, by guaranteeing the prevention of simultaneous gating of modules that share the same power-pins, our proposed technique can accurately limit the current demands to be within designated bounds.

2. RELATED WORK

Power supply noise-aware floorplanning has been studied in the past [Zhao et al. 2002; Chen et al. 2005; Minz et al. 2006] by the design automation community. The central idea of these works involves two concepts: the first one focuses on creating a low-impedance path to the power supply, and the second involves optimizing on-chip decap placement and allocation to suppress inductive noise effects. Additionally, minimizing on-chip decap requirements is important for several reasons. The current leaked by the MOS transistors used to implement decaps increases power consumption and causes additional thermal problem for modern designs. Decaps can also cause wiring congestion and take up silicon area. Lu et al. [2008] proposes a floorplanning algorithm that inserts decaps during optimization in order to minimize the required amount of decap. The floorplanning algorithm used in this work, and presented in Section 5, optimizes decap requirements after the floorplanning optimization algorithm has run. Additionally, the main focus of this work is to develop a floorplanning algorithm that expressly works with the dynamic noise controller, explained in Section 4.

In contrast to most prior art, we advocate a methodology that takes inductive noise issues, into account early in the architecture planning phase of design. By analyzing the microarchitectural behavior of real workloads, we exploit module placement in the floorplanning process to create a design that is inherently more tolerant to inductive noise than a conventional wirelength-driven floorplan. There are a few works that consider power-supply-noise optimization at both the architectural and physical design levels. For example, [Chen et al. 2005] focuses on functional units. The authors propose an architectural-level functional unit selection scheme and find an optimal functional unit floorplan ordering that minimizes power-supply noise and decap demand. In contrast, our work demonstrates minimization of power-supply noise for the entire architecture and chip, and could be used in concert with their technique. The only other prior effort that includes early-design-phase consideration of power-supply noise [Mohamood et al. 2007] does not directly consider the use of a dynamic controller. A comparison to this work is included in Section 6.

The microarchitecture community has also recently paid notable attention to dI/dt issues, due largely to the use of power-saving techniques like clock-gating. Several publications [Grochowski et al. 2002; Joseph et al. 2004; Powell and Vijaykumar 2004] have proposed hardware-based microarchitectural solutions as well as hybrid

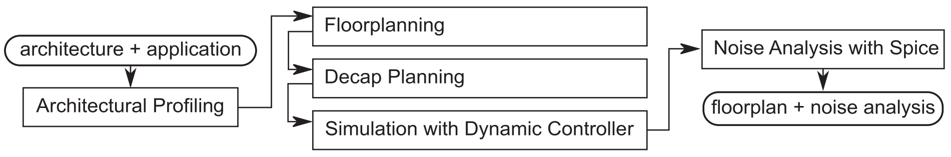

architecture + application
Architectural Profiling
Floorplanning
Decap Planning
Simulation with Dynamic Controller
Noise Analysis with Spice
floorplan + noise analysis

Fig. 1. The design flow used in this work.

hardware/software techniques [Hazelwood and Brooks 2004]. Different from the preceding, mentioned work tackling the *mid-frequency dI/dt* issue (50-100HMz), the focus of this article is to mitigate *high-frequency dI/dt* that requires immediate response, for which the solutions just mentioned are inappropriate. Toward this effort, Powell and Vijaykumar proposed *pipeline muffling* [Powell and Vijaykumar 2003], which controls instruction issues and limits the use of resources for the high-frequency dI/dt concerns.

Note that the cause of high-frequency dI/dt is highly dependent on the spatial distribution of modules across the floorplan and their distance from the power-pins because of on-chip inductance. We make this conclusion based on the following two results. First, it has been demonstrated [Zhao et al. 2002] that the current is mainly supplied by the power-pins nearest to a switching circuit. Second, the ultra-high frequency noise caused by individual gate transitions is confined to extremely small areas [Pant et al. 2007], and should not affect the chip-level supply noise. Together, these observations imply that the supply noise resulting from both the timescale and size of current demand changes related to clock-gating are of utmost importance to distribution network reliability. Finally, high-frequency dI/dt is not only dependent on a given module's self-activity, but also correlated to gating events that stress nearby power-pins. None of the existing work accounts for this fact, which could result in current demand violations or false alarms.

3. UNIFIED DESIGN METHODOLOGY

3.1. Design Flow

An overview of this work's design flow is shown in Figure 1. The input to the flow is an architectural description and a set of benchmark programs. The size of each module in the floorplan is estimated using GENESYS [Eble et al. 1996] and eCACTI [eCACTI]. The flow begins with cycle-level microarchitectural simulation using SimpleScalar [Austin et al. 2002] and integrated power consumption estimation using Wattch [Brooks et al. 2000]. During this simulation, the power consumption of each microarchitectural block, and its switching activity, are collected on a per-cycle basis. This collection is done with the dynamic noise controller inactivated. The switching activity factors are then used to optimize the floorplan. A large number of candidate floorplans are generated and stored during the floorplanning phase. Finally, decap planning is used to select the best among the candidate floorplans. This best floorplan is used to report our results.

The next steps in power-supply planning include final determination of the number and location of the power-supply pins. The most important considerations at this stage are to maintain supply currents below the limits of the C4s and to minimize IR-drop. In high-power designs that are near the limit of the technology may require several iterations. Existing optimization algorithms [Sato et al. 2005; Zhao et al. 2006] may be used. In extreme cases, the floorplan may need to be adjusted. In this work we assume that the supply-pin locations are given before the floorplan is optimized, and are located in a regularly alternating pattern between power and ground.

3.2. Architectural Profiling

Architectural profiling is done using SimpleScalar, a cycle-level microarchitectural simulator. This work assumes aggressive and coarse-grained (module-by-module) clock gating on a cycle-by-cycle basis. Two statistics are collected about each module, namely, the self-switching weight and the correlated switching weight. The self-switching weight is a normalized measure of how often a block changes power state and the correlated switching weight is a normalized measure of how often a pair of blocks switch power states in the same direction during the same cycle. Highly correlated blocks are likely to cause power noise problems, and so should be placed far from each other in a noise optimized floorplan.

3.3. Floorplanning

This work uses simulated annealing based on the sequence pair [Murata et al. 1998] floorplan representation. Floorplanning can impact noise problems in a power distribution grid by moving noisier blocks both away from each other and closer to the power-pins. A longer current-delivering path between a power-pin and a noisy block will lead to more noise for that block's neighbors and for itself, due to the increased inductance in the longer path to the power-pin. Additionally, if the architecture utilizes a dynamic controller that is floorplan-aware, as in this work, then it is possible to optimize the operation of that controller by providing it with a well-formed floorplan. This work uses a new annealing cost function that specifically targets two sources of noise, as well as the physical basis of the dynamic noise-control algorithm. More details are provided in Section 5.1.

3.4. Decoupling Capacitor Planning

Large amounts of decoupling capacitors (decaps) are used to reduce inductive noise in modern designs. This work uses a network-flow-based approach [Wong et al. 2006] to perform decap planning. The planning algorithm is used to select the best among a large group of low-cost (according to the cost function) floorplans found during annealing. The network-flow algorithm is used to analyze how much decap each of the floorplans requires. The floorplan with the smallest requirement is chosen as the overall best. All white space in the floorplan is occupied by decaps.

3.5. Run-time Noise Controller

Clock-gating at the microarchitectural level is used extensively to control the total dynamic power dissipation of modern processors. In this work, a dynamic noise controller based on a floorplan-aware set of queues is implemented to address the noise problems created by clock-gating. Each queue in the controller is based on modules close to each other in the floorplan, and thus likely to cause noise problems for the other modules in the queue. The controller prevents modules within the same queue from switching in the same cycle, and each module's request to power-off is limited by a decay counter. The decay counter prevents modules that are used regularly, but not continuously, from rapidly switching on and off in a short time-frame. The worst-case noise scenario is when all modules attempt to switch simultaneously from one power state to another. The dynamic controller is designed to eliminate occurrences of this, and other less severe, noise behavior. The run-time noise controller is detailed in the next section.

4. QUEUE-BASED DYNAMIC DI/DT CONTROLLER

The design of our dynamic dI/dt controller is presented in this section. The controller is intended to address inductive noise issues caused by excessive clock-gating by

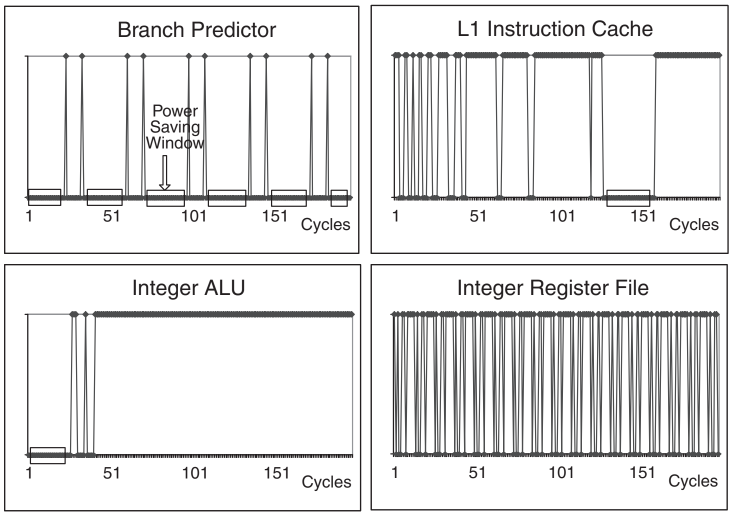


Fig. 2. Module access patterns.

improving the current-demand profile regardless of program behavior. Our design is easily customizable to enable a given design to achieve the right balance among dynamic dI/dt control, power consumption, and performance overhead. The primary components of the dI/dt controller are the following:

—A low-overhead modular decay-counter-based clock-gating mechanism. The objective of the decay counters is to throttle excessive self-gating activity of modules.
—A floorplan-aware clock-gating queue that selectively disables simultaneous switching of modules in the same direction. The queue-based controller is designed to limit the maximum current surge or dip for a given set of power-pins shared by several modules on the power supply grid.
—Pre-emptive activation of ALUs through predecoding for simultaneous dI/dt and performance enhancement.
—An enhancement to the queue controller that enables progressive clock-gating on large modules, such as L2 cache banks.

4.1. Decay-Counter-Based Clock-Gating

The key to avoiding clock-gating induced noise lies in identifying program phases to determine whether clock-gating activity will impact power-supply reliability. Although certain elaborate techniques can accurately predict module requirement patterns, clock-gating requires low-overhead mechanisms to justify the extra hardware cost [Li et al. 2004]. We propose the use of decay counters to enable a low-overhead, dynamic clock-gating scheme that provides a tunable form of dI/dt control. By using low-resolution decay counters to monitor module access patterns, we can choose to save power only during longer stretches of inactivity.

To illustrate this, we provide an example that quantifies fine-grained module access patterns for select modules over a small simulation period in Figure 2. The figure shows an example of an access pattern profile for the branch predictor, the L1 I-cache, an integer ALU, and the integer register file for the 256.bzip benchmark. The 200-cycle interval is shown in the figure to illustrate the potential high-frequency dI/dt effects from a fine-grained perspective. It is observed that typically a module that is inactive for more than 10 to 12 cycles is likely to remain dormant for an extended period of time. Clearly, there is a threshold cycle count beyond which a module can be reliably gated-off with the least likelihood of encountering high-frequency inductive noise. On the other hand, it can be seen that when a module remains unaccessed for less than 5 to 10 cycles, it is highly likely to be accessed again soon.

To exploit this behavior, a decay counter is employed to enable clock-gating activity only when a minimum turn-off threshold has been exceeded. We use a 4-bit decay counter for each microarchitectural module inside the processor. The decay counters only permit clock-gating of a module if it has not been accessed during the last 16 cycles. For any given module, the counter decays every cycle unless there is an access made to that particular module, in which case the decay counter is reset back to the maximum.

The resolution of the decay counter provides the tradeoff between high-frequency inductive noise control and power dissipation. A large decay counter will further smooth out current spikes over time, but at a cost of higher average power consumption due to the fact that modules will be gated-off only after a long interval of inactivity. The opportunity for power saving is also dependent on the module access pattern. As shown in Figure 2, certain modules, such as the branch predictor or I-ALU, exhibit larger potential for power savings than others that display high activity, like the integer register file.

4.2. Dynamic dI/dt Controller Architecture

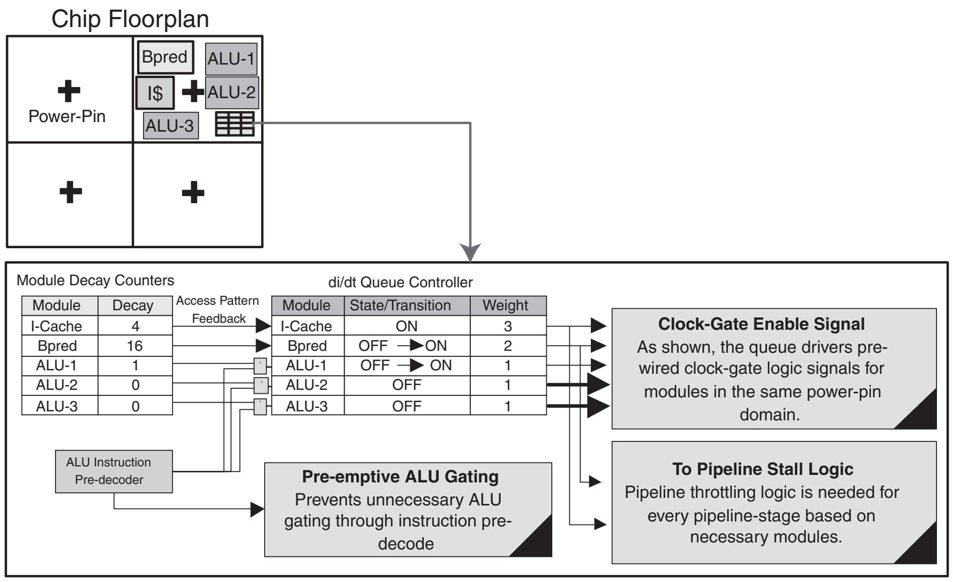
Even though the decay counters can provide a smoother current profile for each module by eliminating unwanted switching activity, they are inherently incapable of avoiding dI/dt issues caused by simultaneous gating of modules that share common power-pins. To address these shortcomings, we propose a *queue-based controller* that is aware of the processor's floorplan and power-pin distribution. In the processor's power-delivery network, a module usually draws more current from spatially nearby power-pins, in other words, following the path(s) with the lowest impedance. Consequently, adjacent modules, if they switch simultaneously in the same direction, will unreliably stress local power-pins. Therefore, to guarantee the maximum current ramp at any given time, it is necessary to be able to dynamically alter simultaneous gating of modules that are likely to stress the same power-pin(s). The proposed queue-based controller is designed to overcome unreliable simultaneous switching of adjacent modules. The salient features of the controller are as follows.

—A static queue with an entry for each module sharing the same power-pin domain. Ideally, there will be no more than eight entries in a queue, resulting in a 3-bit module identification number that is local to each queue.[1]

—Every queue entry contains the corresponding *state* of the matching module, which indicates either the current state or any requested clock-gating transition event. This will require 2 bits for the ON/OFF states as well as the ON$\rightarrow$OFF and OFF$\rightarrow$ON transitions. The *state* is used to drive the prewired clock-gating signals to the corresponding modules.

—Every queue entry that represents a module also has an associated *integer weight* that is proportional to the current consumed by the corresponding module. We use a two bit integer to represent one of the four different current consumption levels. Since weights are used to compute and check for current demand violations, integer weights are appropriate for faster current demand calculations. Fast calculations are essential for quick response to high-frequency dI/dt events.

[1]The number of entries are limited to minimize the performance loss, as explained in Section 4.2.


Chip Floorplan
Bpred
ALU-1
I$
ALU-2
ALU-3
Power-Pin
Module Decay Counters
Access Pattern Feedback
di/dt Queue Controller
ALU Instruction Pre-decoder
Pre-emptive ALU Gating
Prevents unnecessary ALU gating through instruction pre-decode
Clock-Gate Enable Signal
As shown, the queue drivers pre-wired clock-gate logic signals for modules in the same power-pin domain.
To Pipeline Stall Logic
Pipeline throttling logic is needed for every pipeline-stage based on necessary modules.

Module	Decay
I-Cache	4
Bpred	16
ALU-1	1
ALU-2	0
ALU-3	0

Module	State/Transition	Weight
I-Cache	ON	3
Bpred	OFF → ON	2
ALU-1	OFF → ON	1
ALU-2	OFF	1
ALU-3	OFF	1

Fig. 3. Noise controller architecture.

A simplified version of our high-frequency dI/dt controller architecture is depicted in Figure 3. The "+" signs on the chip floorplan (left side) indicate the power-pin locations. For simplicity, we illustrate only four power-pins. The queue-based controller works in the following manner. The decay counters will signal a transition event, that is, ON$\rightarrow$OFF for a given module in the queue. Let Δ be the current demand threshold that is permitted for a given power-pin domain. At any given time, a *head* pointer is always pointed to one single module in the queue. Every cycle, the queue is traversed by a window size which has a total weight of Γ. The value of Γ is the largest sum of weights of consecutive modules that are in the transition states (ON$\rightarrow$OFF or OFF$\rightarrow$ON), such that $\Gamma \leq \Delta$. Since integer weights can be negative as well,[2] the sliding window will attempt to permit the maximum allowed transitions without violating the maximum current demand constraint.

To better understand the dI/dt queue-controller mechanism, we use an example based on the instantaneous state of the controller, as shown in Figure 3. Let us assume that the value of the current demand threshold is, $\Delta = 3$. In the figure, ALU-2 and ALU-3 are gated-off (indicated by the bold arrow that is the output of the queue controller). Both Bpred and ALU-1 have an activation request indicated by the OFF$\rightarrow$ON state. Therefore, the combined weight of the sliding windows, $\Gamma = 3$.[3] The queue-controller will therefore permit both module-gating events to occur, since the threshold constraint is not violated in this case. After servicing the transition, the *head* pointer will traverse two entries and point to the ALU-2 entry in the queue. In contrast, consider an alternate case where ALU-1 had a higher weight that resulted in the weight of the sliding window to exceed the current threshold budget. In this case, only the Bpred transition will be serviced by the queue-controller. Also, the *head* pointer will traverse only one module entry to ALU-1, so that it can be serviced in the next cycle. Furthermore, consider yet another example where ALU-1 requires an ON$\rightarrow$OFF transition, which represents a

[2]OFF$\rightarrow$ON is a positive switch, while ON$\rightarrow$OFF represents a negative switch.
[3]Please note that in a real implementation, the sliding window will have an upper limit in terms of how many modules weights can be computed in a given cycle.

negative weight. In this case, $\Gamma = 1$, thus still permitting both Bpred and ALU-1 to perform their transitions. However, the sliding-window value will still be below the threshold, Δ—and the queue-controller can potentially gate the next ALU-2 module if it requires a transition. These examples are provided to illustrate how the sliding window adjusts dynamically, based on the worst-case current demand that can be sustained in a given power-pin domain.

The example dI/dt queue in Figure 3 shows the modules in descending order of weight. Note that the dI/dt controller will enforce the current demand threshold regardless of the order of the modules in the queue. However, the ordering of modules does affect the performance, overhead imposed by the design. For instance, clustering modules in the queue that have high weights will create a larger performance overhead, since multiple modules will not be permitted to transition because they consistently violate the current demand threshold. The ordering of modules in the queue is static and presents a design choice that needs to be made by an architect for a given floorplan.

Also, note that the queue in our dI/dt controller is different from a typical queue structure like the instruction fetch queue, a memory structure allocated at run-time. In contrast, the entries in the dI/dt controller queue are prewired for each module at design time in order to simplify the logic for driving clock-gating signals directly to the modules.[4] Functionality-wise, the controller behaves like a circular queue that traverses as many modules as determined by the sliding-window threshold. Note that the maximum hardware overhead of each microarchitectural module is merely 11 bits (including the decay counter). This is rather negligible in terms of additional power dissipated and extra current drawn by the controller itself.

4.3. Preemptive ALU Gating

Preemptive ALU clock-gating through predecoding instructions is another technique we propose to prevent unnecessary gating activity. Decay-counter-based clock-gating allows gating events to occur based on the history of module accesses. However, decay counters by themselves will be unable to predict the future switching activity of modules. For instance, it will be detrimental to performance if an ALU is going to be gated-off due to a saturated decay-counter, when in fact an incoming ALU instruction has just been fetched. Furthermore, if an ALU instruction is on its way, it makes sense to leave the unit "on," even from a dI/dt perspective. To achieve this goal, we include preemptive *turn-on* gating of ALU modules by predecoding instructions. In a typical RISC ISA, the opcode can be determined by observing the first few bits of the instruction,[5] allowing us to predecode this information simultaneously with the instruction fetch. In the case that an ALU instruction has been detected early on, it is used to override the decay-counter *turn-off* request. In CISC ISAs, it might not be possible to easily perform a simple predecode due to variable-length instructions, but even in this case, other techniques, such as storing predecode information in the L1 instruction cache [Austin and Sohi 1995], can be used to achieve this effect.

This technique is similar to deterministic clock-gating (DCG) proposed by Li et al. [2004], in that we predict the necessity of ALU units in the near future based on the most recently fetched instructions. However, the application is different in our case. The goal of DCG is to reduce power dissipation in units by predicting the necessity for those units. In our scheme, we use the preemptive ALU-gating knowledge to *prevent* unnecessary clock-gating of units that could result in dI/dt violations.

[4]Since the queue entries are prewired to the clock-gating output, it is possible to apply certain heuristics to the order of modules in the queue with asymmetric weights, in order to permit the maximum possible number of transitions at any given time. Such optimizations, however, are out of the scope of this work.
[5]For example, Alpha and PowerPC ISA uses the prefix 6 bits for the opcode.

4.4. Enhanced Progressive Gating of Large Modules

Simultaneous power-state-switching of multiple modules can be prevented completely by selective gating. However, some monolithic modules like the L2 cache can still consume large amounts of current, resulting in unreliable voltage swings. For this reason, certain processors employ progressive gating of large modules, like the L2 cache, in order to mitigate dI/dt effects [Jacobson et al. 2005]. However, ad-hoc progressive gating does not prevent other adjacent modules from switching simultaneously, and can still result in unreliable dI/dt surges. To counteract this issue, our queue-based controller can be used to generate multiple clock-gating domains for even a single monolithic module by merely replicating multiple entries for a module with smaller weights. For instance, for a banked L2 cache, there can be as many entries as the number of banks within the queue with proportionally lower weights.[6] Since the queue inherently throttles simultaneous switching activity, it presents a much more effective progressive gating mechanism than current solutions. Thus, the queue-based controller can enable efficient progressive gating of such modules while maintaining the current demand below the noise-tolerant threshold, which is set to mitigate negative simultaneous switching effects.

4.5. Pipeline Design Implications

The use of any dynamic dI/dt controller requires an appropriate performance-throttling mechanism to guarantee program correctness, even if certain necessary processor components are unavailable when needed. For instance, the instruction scheduler needs to be accurately aware of the ALU availability before issuing operations. The integration of a dI/dt controller into a conventional architecture will require the pipeline logic to be fully aware of the clock-gating state of the modules in order to issue operations without affecting correctness. For this reason, it is essential that the dI/dt controller not impose impractical design requirements on the processor pipeline.

Our queue-based high-frequency dI/dt controller can be easily built into a conventional out-of-order pipeline without significant additional complexity. Conventional processor modules are already capable of correctly operating under resource contention. In the event of resource hazards such as over-subscription of ports in the register file, caches, or load-store queue, the selection logic will properly delay the issue of certain operations. As indicated in Figure 3, our queue has static entries and prewired logic that indicates the availability of any given module. This makes it efficient to integrate the additional resource availability constraint into an existing selection logic in the pipeline. Since resource availability can be directly interpreted from the output of the queue-based controller, an enhanced pipeline with the dI/dt controller merely needs to ensure that the resource availability constraint overrides all conventional hazards for correct functionality.

5. NOISE-CONTROLLER-AWARE FLOORPLANNING ALGORITHM

Previous work has addressed the IR drop problem by including decap considerations during the floorplanning process. Other work has independently addressed the coupled dynamic inductance noise problem by separating blocks that switch during the same cycle. This is the first work to combine both direct IR drop considerations and LdI/dt dynamic noise considerations with dynamic-controller awareness during floorplanning. This is accomplished through the combination of a novel design flow and a new cost function. It should be noted that the term “switching activity” refers to changes in a clock-gated power state, and not transistor-based switching.

[6]Typically, L2 cache banks are in separate clock-gating domains.

5.1. Annealing Cost Function

A new annealing cost function is used that specifically targets two factors affecting power supply noise, as well as the physical basis of the dynamic noise control algorithm. There are five terms in the cost function. The first two target traditional physical design objectives, area (A) and wire-length (W). The third term of the cost function, I, addresses self-induced inductance and IR drop. Correlated switching factors are considered in the fourth term, C. The final term, Q, includes consideration of the dynamic noise control algorithm. The total cost function is given by

$$Cost = \alpha \cdot A + \beta \cdot W + \gamma \cdot I + \delta \cdot C + \epsilon \cdot Q$$

where, α, β, γ, δ, and ϵ are weighting constants. In this work the values for the weighting constants were empirically determined to be 1, 0.2, 0.5, 0.5, and 0.025 for α, β, γ, δ, and ϵ, respectively. The first two terms are defined in the usual way, based on Manhattan distance and the bounding box of the floorplan. The three final terms, and how they are used to control the three sources of noise and optimize the performance of the dynamic noise control algorithm, are described in the following sections.

It should be noted that this work assumes the power-supply C4s are capable of delivering a sufficient amount of current without suffering from electromigration effects. Modern high-performance processor designs are running into this current density limit. It is possible to combine the algorithms presented in this work with other methods or additional cost-function terms to ensure that these electromigration current limits are met. However, the microarchitectural simulator used in this work limits the granularity of the powermap that can be simulated, which constrains our ability to provide a meaningful analysis of this issue.

5.2. Self-Switching Current

The self-switching current term, I, is defined as follows:

$$I = \sum_{\forall i \in B, \forall j \in P} curr_i \cdot sw_i \cdot dist_{i,j} \cdot reg_{i,j}$$

where B is the set of all blocks; P is the set of all pins; $curr_i$ is the current requirement of block i; sw_i is the self-switching factor of block i; $dist_{i,j}$ is the Euclidean distance between block i and pin j; and $reg_{i,j}$ is one if and only if block i is in the current drawing region of pin j, and zero otherwise. The current drawing region of a pin is defined to be half the distance to the next nearest pin. Figure 4 shows an illustration of the self-switching term. Previous work that considered the LdI/dt problem [Mohamood et al. 2007] did not directly consider the IR drop problem. The pin capacity force described in that work focused on satisfying the current drawing requirements of each pin, pushing blocks away from pins that were overloaded and pulling them towards pins that were underloaded. Their work did not weight blocks that needed more current than others. The self-switching term used in this work, I, considers both the current requirements of each block and the amount of switching that the block exhibits. It therefore considers both the IR drop seen by each block as well as the self-induced inductance noise seen by each block. When blocks are farther away from the pins, the resistance term, R, of the IR drop is increased. In a complementary fashion, the inductance term, L, of LdI/dt inductance noise is increased when blocks are farther away from pins. The distance between pins and blocks that have high current demand and high switching activity is minimized by minimizing I. Therefore, the IR drop and LdI/dt noise seen by the chip as a whole is also minimized.

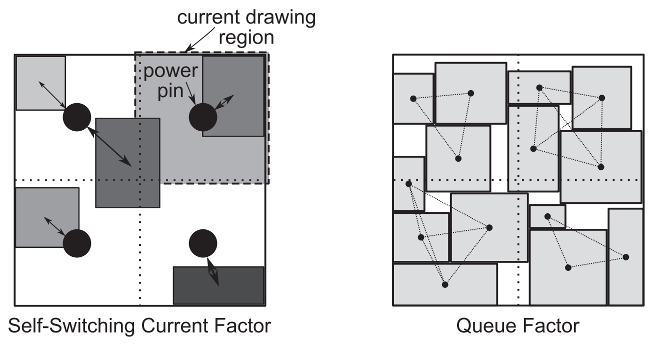


Fig. 4. Illustration of the self-switching-current factor and queue factor cost function terms. For self-switching current, the higher-weighted (darker) blocks, based on current demand and switching activity, are drawn to the power-pins more strongly. For the queue factor, each quadrant of the chip has a different queue. Only modules within the same queue are given a weighted cost function bonus based on their correlated switching activity and current demand. Queues are defined spatially, and blocks have no movement restrictions during annealing.

Table I. Self and Correlated Switching Weights of Modules for a Sample Benchmark

	LSQ	RUU	BTB	L2$	IRF	L1D$	ALU0	ALU1	ALU2	ALU3	ALU4	ALU5	L1I$	Bpred	DTLB	ITLB	FALU0	FALU1	Freg
LSQ	*28*	0	20	13	20	2	10	10	10	10	10	10	11	20	0	11	10	10	12
RUU		*26*	8	4	13	2	0	0	0	0	0	0	5	8	2	5	0	0	5
BTB			*18*	7	29	17	13	13	13	13	13	13	37	100	17	37	13	13	13
L2$				*16*	14	28	12	12	12	12	12	12	21	7	26	21	4	4	7
IRF					*10*	17	7	7	7	7	7	7	23	29	17	23	8	8	24
L1D$						*7*	6	6	6	6	6	6	11	17	93	11	5	5	6
ALU0							*3*	100	100	100	100	100	15	13	6	15	66	66	4
ALU1								*3*	100	100	100	100	15	13	6	15	66	66	4
ALU2									*3*	100	100	100	15	13	6	15	66	66	4
ALU3										*3*	100	100	15	13	6	15	66	66	4
ALU4											*3*	100	15	13	6	15	66	66	4
ALU5												*3*	15	13	6	15	66	66	4
L1I$													*3*	37	12	100	11	11	5
Bpred														*3*	17	37	13	13	13
DTLB															*2*	12	5	5	6
ITLB																*1*	11	11	5
FALU0																	*1*	100	5
FALU1																		*1*	5
Freg																			*0*

5.3. Correlated Switching Factor

The correlated switching factor term, C, is defined as follows:

$$C = \sum_{\forall i,j \in B} \frac{curr_i \cdot curr_j \cdot corr_{i,j}}{dist_{i,j}},$$

where $corr_{i,j}$ is the correlated switching activity, described above, between blocks i and j. The rest of the terms are as defined in Section 5.2: B is the set of all blocks; $curr_i$ is the current drawn by block i; and $dist_{i,j}$ is the Euclidean distance between blocks i and j. The minimization of this term maximizes the distance between blocks that have both high current and often switch simultaneously. If two blocks are positioned near one another and draw current from the same power-pin, then simultaneous switching would exacerbate the LdI/dt noise seen by both blocks. An example table of the per-module correlated switching weights for a sample benchmark is shown in Table I.

5.4. Dynamic Controller Queue Factor

The dynamic controller queue factor, Q, is defined as follows:

$$Q = \sum_{\forall i,j \in B} -(curr_i \cdot curr_j \cdot corr_{i,j} \cdot q_{i,j}),$$

where $q_{i,j}$ is one if and only if blocks i and j reside within the same dynamic noise controller queue, and is zero otherwise. The rest of the terms are as defined above: B is the set of all blocks; $curr_i$ is the current drawn by block i; and $corr_{i,j}$ is the correlated switching activity between blocks i and j. Figure 4 shows an illustration of the queue factor. The dynamic noise controller is floorplan-aware through the use of spatially organized queues. Therefore, by specifically optimizing the blocks that occupy each queue it is possible to optimize the operation of the dynamic noise controller through physical design. Including this type of optimization into a force-directed approach would be extremely difficult, which is why this work uses the more flexible simulated-annealing-based floorplanning approach. The queue factor takes its form from the correlated switching factor. It adds consideration of current-weighted correlated switching activity to the cost function, based on whether two blocks share the same queue or not.

The initial motivation behind the form of the queue factor was that we believed it would be more problematic if highly-correlated blocks resided within the same queue. Highly correlated blocks should be separated from one another with a large distance. If those blocks are within the same queue, it is bad, because the queues are spatially designated, and blocks in the same queue are necessarily close to one another. Experimentation proved this assumption to be incorrect. Floorplans produced using a cost function with a positive queue factor (+Q) had uniformly worse power supply noise than floorplans produced using no queue factor (No Q) at all. However, if the queue factor is included as a bonus (−Q) instead of as a penalty, the noise characteristics are improved over a noise-aware-only floorplan. These results are discussed in more depth in Section 6.4. This behavior can be explained by the fact that no matter how far away highly correlated blocks are from one another, they are still connected to the same power distribution grid, and can cause coupled inductive noise by switching simultaneously. It is also possible that highly correlated blocks are near one another, but just slightly over the queue boundary. This scenario would cause noise problems, but be given a lower cost with a positive queue factor. By adding a negative bonus (−Q) to the cost function when blocks that are highly correlated reside within the same queue, the dynamic controller is allowed to deal with the noise problem more effectively.

5.5. Decoupling Capacitor Planning

The floorplanning algorithm described above can produce a floorplan with low cost, according to the cost function. However, the whitespace of the floorplan will be used for decaps to reduce the dI/dt-induced voltage swing. The network-flow-based approach from Wong et al. [2006] is used to perform decap planning in this work. The decap planning algorithm is used to choose a floorplan that has minimal decap requirements. The planning algorithm is used to select the best among a large group (in this case, 200) of low-cost (according to the cost function) floorplans found during annealing. The network flow algorithm is used to analyze how much decap each of the floorplans requires. The floorplan with the smallest requirement is chosen as the overall best. In this way, the amount of decoupling capacitance is minimized.

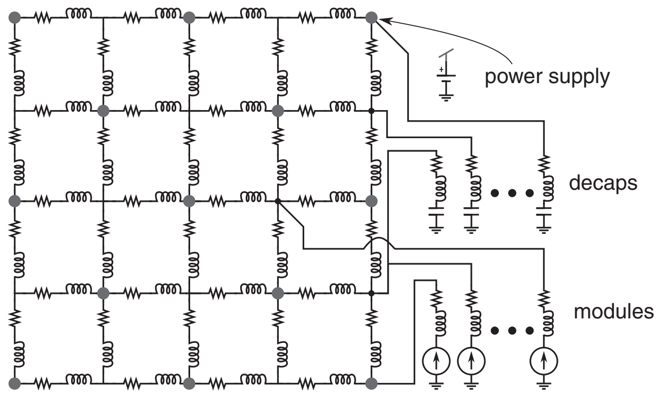


Fig. 5. The SPICE circuit used for simulating LdI/dt noise. Voltage supply bumps are positioned at every other grid point. Decoupling capacitors and modules (current sources) are connected to the nearest grid point in the floorplan.

5.6. Power Network Analysis

To evaluate the effectiveness of the cost function that is used to guide the dynamic-noise-controller-aware floorplanning, we use a SPICE model of the on-chip power delivery network [Chen et al. 2005]. A depiction of the SPICE model is shown in Figure 5. The noise-mitigation technique is evaluated using the worst-case current consumption scenario. The worst-case switching activity of an application is determined by sampling microarchitectural activity of all modules over the duration of the simulation. By comparing module activity during different program phases, the period where the highest module switching occurs can be determined. Once the worst-case phase is identified, the current profile of each module is generated from the microarchitectural simulator. This complex current waveform is used as piecewise linear (PWL) source input to the SPICE module. Induced noise effects can then be observed as a direct function of the application's behavior.

6. EXPERIMENTAL RESULTS

This section describes the experimental techniques and simulation framework used to implement and evaluate both the dynamic dI/dt control mechanism and the controller-aware floorplanning algorithm.

6.1. Simulation Framework

Our simulation framework is based on SimpleScalar 3.0 and Wattch [Brooks et al. 2000] running the SPEC2000 INT and FP benchmark suite. To understand the access patterns of individual modules that motivated the solution of this work, we include various profiling and instrumentation facilities in our simulator. For the implementation of the dynamic dI/dt controller, we extended SimpleScalar/Wattch to incorporate a floorplan-aware queue configuration. We also implemented a detailed, floorplan-dependent performance-throttling model and queue configuration for studying the performance impact of our technique. The primary simulation parameters used in our simulations are shown in Table II. The power and current consumption metrics were based on a 5GHz processor developed using a 70nm process technology. Each simulation was fast-forwarded by 4 billion instructions and simulated for 1 billion instructions. The current signatures that were chosen to evaluate the dynamic dI/dt controller represent the worst-case overall module switching activity over the entire simulation period.

Table II. The Microarchitecture's Parameters

Parameters	Values
Fetch/Decode width	8-wide
Issue/Commit width	8-wide
Branch predictor	Combining: 16K entry Metatable Bimodal: 16K entries 2-Level: 14 bit BHR, 16K entry PHT
BTB	4-way, 4096 sets
L1 I- and D-Cache	16KB 4-Way 64B line
I- and D-TLB	128 Entries
L2 Cache	256KB, 8-way, Unified, 64B line
L1/L2 Latency	1 cycle / 6 cycles
Main Memory Latency	500 cycles
LSQ Size	64 entries
RUU Size	256 entries
Functional Units	8 IntAlu (only 2 can be used for IntMult) 4 FPAlu (only 2 can be used for FPMult)

6.2. Baseline Floorplanning Methodology

Our baseline-specific floorplan is purely wirelength and area-driven. The baseline floorplan is independent of running applications, that is, no profile-guided optimizations were employed in the floorplanning algorithms. This baseline is compared against the noise-aware floorplan with the dynamic dI/dt control mechanism in place. The noise-aware floorplan is always used in conjunction with the dynamic controller because it is specifically designed to work with the controller. The floorplan obtained, along with the predefined power-pin distribution, determined the configuration of the queue entries used in the dynamic dI/dt controller.

6.3. Dynamic dI/dt Controller Results

The basic objective of our dynamic dI/dt controller is to minimize the burden on power-pin(s) caused by nearby modules. Therefore, for any given floorplan and power-pin configuration, the design objective of the dI/dt controller is to place queues for effective dI/dt control in a distinct section of the floorplan. For this work, we divided the floorplan into four quadrants, with each quadrant representing a distinct power-pin domain. Note that certain power-pins can be in multiple domains. For instance, quadrant-based module separation could result in 5 power-pins per quadrant, because the power-pins on the borders of the quadrants exist in multiple domains. The number of distinct power-pin domains is a design choice influenced by the degree of dI/dt control that is required. A high number of power-pin domains results in a larger number of queues and finer grained control. On the other hand, too few power domains will result in larger queues impacting performance, due to the fact that the worst-case delay in transition is higher. For our experiments, queues were assigned to each quadrant and all the modules placed in that quadrant. Since the floorplan determines the queue configuration, different floorplans will have different performance impacts as well as distinctive dI/dt characteristics.

We applied the technique to both the baseline floorplan and the controller-aware floorplan to evaluate the effectiveness and overhead of our dynamic dI/dt controller under different scenarios. The results include current profiles on a baseline machine without a dI/dt controller versus our technique, as well as the average current variability across all benchmarks. We also present the performance overhead incurred due to our dynamic dI/dt controller. Finally, we show the thermal impact of the controller, which is caused by the additional power dissipated by modules that would have been clock-gated in a design without a dynamic noise controller.

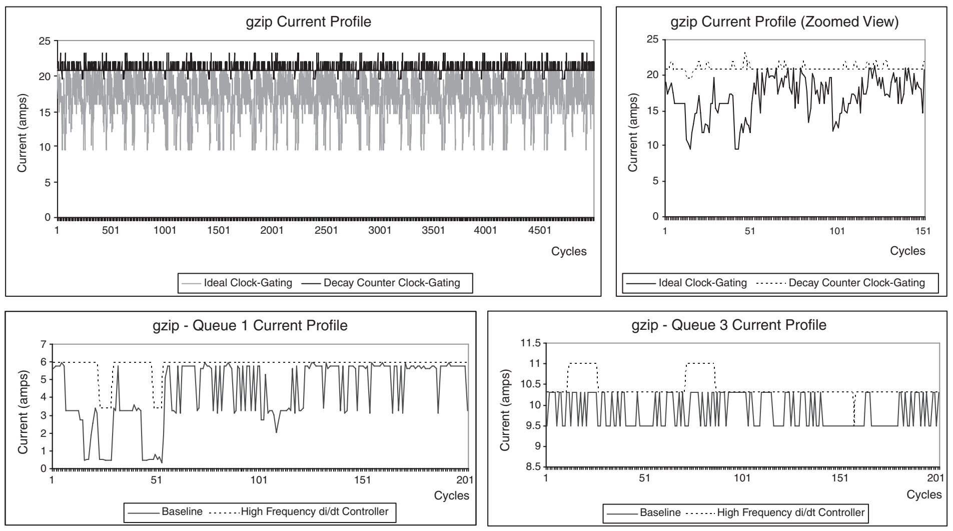


Fig. 6. High ILP benchmark (164.gzip) chip current and queue current profiles.

6.3.1. Current Profile of Applications. To demonstrate the effectiveness of our controller in improving high-frequency current-demand changes (dI/dt), we now present the current profile of the whole chip as well as the current profile for each queue cluster for the wirelength-driven floorplan. Note that the effectiveness of a dI/dt controller is evaluated by observing its effect on the worst-case current profile of a given application, which represents the computation phase with maximum module-switching activity. Due to the staggeringly large number of current profiles for all of the benchmark programs, we demonstrate representative characteristics using two types of benchmark programs. Note that the crucial information conveyed in this section is to show the effectiveness of our proposed mechanism.

We profiled one high-ILP benchmark (164.gzip) and another low-ILP (181.mcf, memory-bound) benchmark. The current profiles, shown in Figures 6 and 7, were obtained by determining the worst-case switching activity during the course of execution. A 4-bit decay counter was used for each module in all experiments.[7] Each graph shows the current profile for both the processor with ideal clock-gating as well as the decay-counter-based clock-gating mechanism. We also provide close-up versions of the representative, highly active, region of the graph for better visibility.

It can be seen that both 164.gzip and 181.mcf exhibit a repetitive current profile during the worst-case switching period. This is especially prominent in the current profile of 181.mcf where there is a period of high activity for a few hundred cycles, followed by a stable current profile for approximately 500 cycles. This is due to the long-familiar cache misses to main memory that occur in 181.mcf. During this period most modules are inactive and can be clock-gated off to save dynamic power. The effectiveness of the dI/dt controller in improving the current ramp is obvious in the zoomed versions of the graphs. It shows that with the decay counter, our system (shown in dashed lines) successfully prevents unnecessary oscillating swings in the current profile and produces a much smoother down-ramp. For 164.gzip in Figure 6, we observe large current variation in the ideal-clock gating scheme due to high activity across all modules. This results in there being no significant duration of time where reasonable

[7]The resolution of the decay counter was based on the motivational data discussed in Section 4.2.

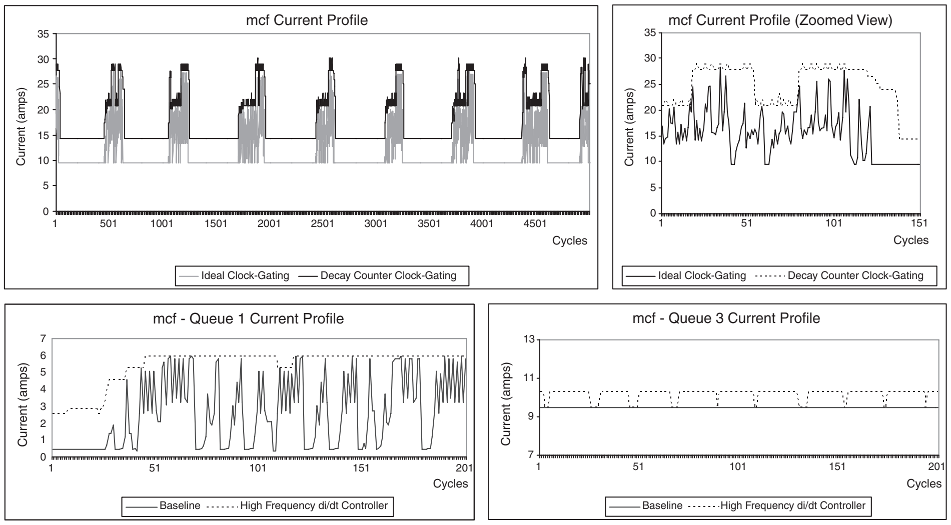


Fig. 7. Low ILP benchmark (181.mcf) chip current and queue current profiles.

power savings are possible. Because of this, modules are never inactive for extended periods of time, and the decay counters rarely clock-gate-off the modules. The current profile is extremely stable for this reason. In short, the decay-counter-based technique finds the optimal power envelope right above the ideal clock-gating mechanism and allows clock-gating only when there is a significant chance that the given modules will not be accessed again soon.

Next, we present the current profile with the integration of the complete queue-based controller. Note that this is the complete controller that incorporates prevention of simultaneous switching, decay-counter-based feedback for clock-gating, preemptive ALU gating, and progressive gating of L2 cache banks. The lower boxes of Figures 6 and 7 also show the current profile for Queues 1 and 3 for 164.gzip and 181.mcf. In all cases, it can be observed that the current profile is significantly improved by eliminating excessive switching activity. In addition, both the upward ramp and downward ramp effects due to multiple modules in the same power-pin domain (i.e., using the same module queue) are spread across multiple cycles. This is more prominent in the upward ramp of the current with the dI/dt controller between cycle 20 and cycle 50 for Queue 1 in 181.mcf. For Queue 3 in 181.mcf, we observe a different trend, whereby the dI/dt controller ramps up current repeatedly compared to the baseline, which is stable. This is due to the preemptive ALU gating effect that ramps up additional ALUs which are otherwise unused in the baseline clock-gating scheme due to low ILP. We observe a repetitive pattern where ALUs are gated preemptively only to later decay after approximately 20 to 25 clock cycles. However, these ramps are still spread over many cycles and do not violate the current demand threshold.

For 164.gzip, where there is high ILP/switching activity, the queue-based controller ramps up to the required current levels and does not saturate the decay counters for long enough. For this reason, the queue-current profile is almost always stable, except for the few cases where the decay counters decay long enough to enable clock-gating. It is important to note that this does not mean that there is no opportunity for power-savings in such a design without dI/dt control. The presented phase of 164.gzip is the highest ILP portion in our simulation, and it is simply not worth it to clock-gate elements during this phase because of the dI/dt as well as the performance penalty.

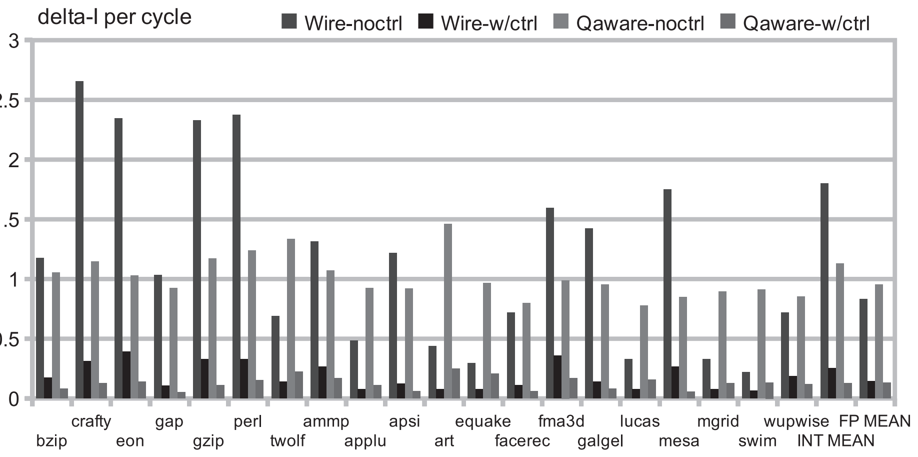


Fig. 8. Average current variability for the benchmark suite with the wirelength-aware floorplan.

Since presenting detailed current profiles is infeasible for all benchmarks, we now show the current variability per cycle for the complete duration of the benchmark execution. Unlike the worst-case profile presented earlier, this metric gives the *average variability of current per cycle* for both the baseline processor and the processor with our dynamic dI/dt controller. Figure 8 shows the comparison for various SPEC2000 INT and FP benchmark programs. The current variability is calculated by measuring intercycle current fluctuations (in the absolute value of the swing) over the entire simulation period, as a fraction of the total number of simulation cycles. It can be observed that the baseline architecture shows a higher degree of current variability across the board. The data shows that 186.crafty exhibits the highest variability, whereas 171.swim has the lowest variability. Regardless of the native current variability, our dynamic dI/dt controlling mechanism can significantly mitigate the dynamic oscillating behavior of the current profiles of the running applications. The dI/dt controller pushes the current variability below 0.5 amps/cycle for all the benchmark programs studied.

6.3.2. Performance Impact. We now present the performance analysis of our dI/dt controlling mechanism. Figure 9 shows the IPC degradation for the SPEC2000 INT and FP benchmark suites with the dI/dt controller over the baseline machines without any dI/dt control. The Wire-w/Pre configuration shows the queue controller with preemptive ALU gating turned on in order to differentiate the type of applications that can benefit from predecoding ALU instructions. Progressive gating in the L2 cache was applied to all cases.

In general, we observe minimal performance degradation for most of the benchmarks. Note that the performance overhead is dependent on the floorplan because the floorplan affects the queue configuration. A more optimized floorplan will result in a more balanced queue configuration. However, if the floorplan results in a configuration where one queue carries a significantly larger number of modules than the others, IPC will be adversely affected because the worst-case module activation time is longer. We observe an average performance overhead around 4% for the floorplans that were simulated.

We also observe that preemptive gating of ALUs improves the performance for certain benchmark programs such as 252.eon, 254.gap, 253.perl, and 168.wupwise. This is due in part to the fact that the 4-bit decay counter saturates consistently for ALUs

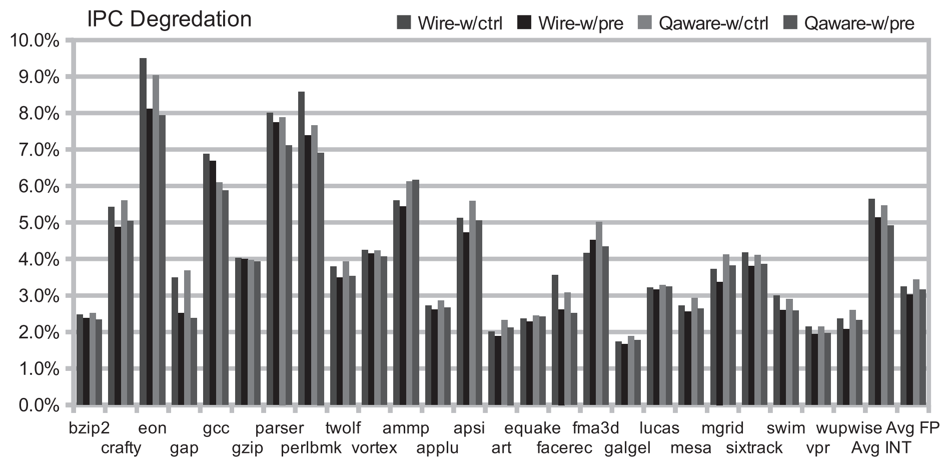


Fig. 9. The performance degradation of the dynamic dI/dt controller across the benchmark suite for both the wire-length-aware (wire-) and queue-aware (qaware-) floorplans.

(resulting in powering-off the module) right before ALU instructions are issued. It is in these scenarios that the preemptive gating provides simultaneous performance and dI/dt benefits. The decay counters predict future likelihood of module access based solely on the past activity profile. In contrast, preemptive gating can "look-ahead" and override unnecessary gating that the decay counters themselves cannot prevent, thereby reducing unnecessary performance loss. The minimal IPC overhead illustrates the practical potential of employing a low-overhead technique to control high-frequency dI/dt.

6.3.3. Power Consumption Impact. The dynamic controller presented in Section 4 includes mechanisms that decrease the frequency of clock-gating events. Clock-gating is a technique designed to limit dynamic power consumption. When the dynamic controller prevents a module from turning off to limit dI/dt noise, the processor will dissipate some extra energy. When the dynamic controller prevents a module from turning on, the processor will dissipate less power, however the performance impact will cause the program being executed to consume more total energy over an extended period of time. Figure 10 shows the impact on power dissipation for both the wirelength- and queue-aware floorplans using the dynamic controller compared to the same floorplans without the dynamic controller. The average increase in power consumption is near 2% for both the floating-point and integer suites. The maximum increase in power consumption is 6.3% for the queue-aware floorplan.

6.3.4. Thermal Impact. The goal of our technique is to improve power-supply reliability and reduce design effort. Therefore, it is critical that our dI/dt controller must not cause other reliability problems. Since our technique provides fine-grained dI/dt control at the expense of increased power consumption, it is necessary to quantify any potential adverse thermal effect caused by the dynamic controller. Thermal issues are particularly critical in modern processors due to increases in power density caused by process technology shrinks.

We used Hotspot 3.0 [Skadron et al. 2004] to evaluate the thermal impact of our high-frequency dI/dt controller on the given floorplan. We compare our architecture against the baseline design that uses ideal-clock-gating, which represents the scenario with the least power consumption. Figure 11 shows the thermal analysis for all

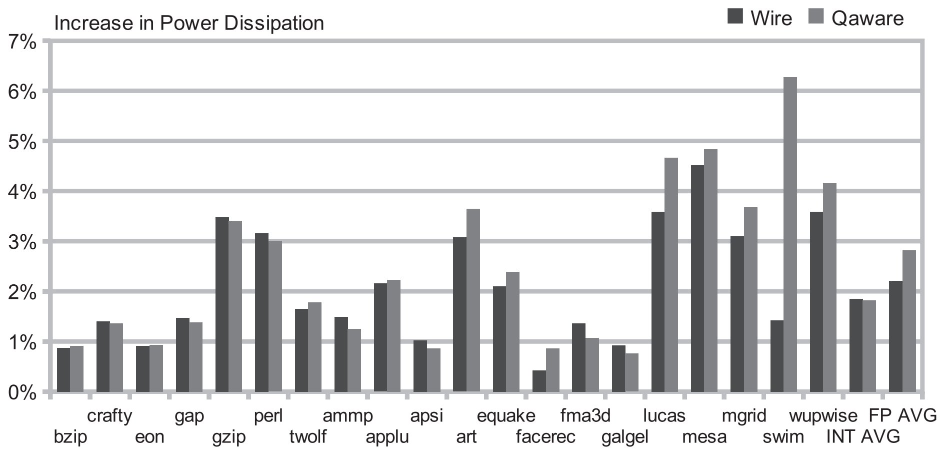


Fig. 10. The power consumption impact of the dynamic dI/dt controller on the wirelength- and queue-aware floorplans.

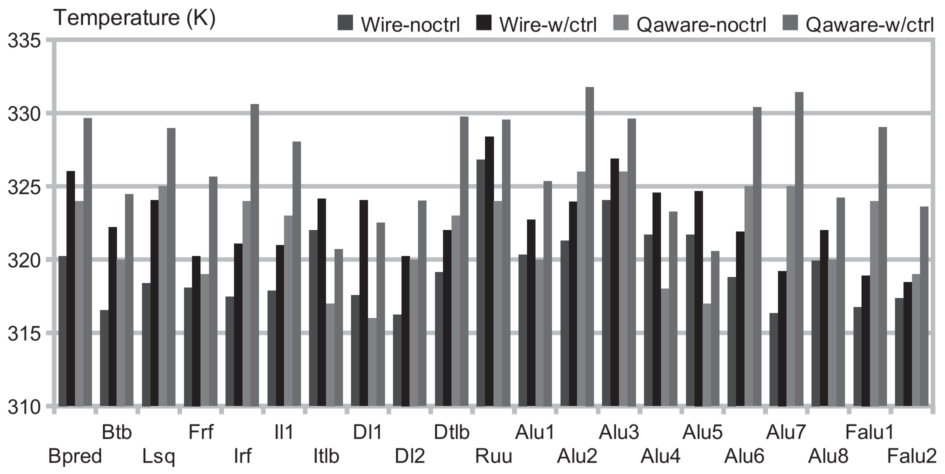


Fig. 11. The per-module thermal impact of the dynamic dI/dt controller on the wirelength- and queue-aware floorplans.

23 modules in our processor model, evaluated using the SPEC2000 benchmark suite for both the wirelength- and queue-aware floorplans.

First, we consider only the impact of using the dynamic noise controller. In Figure 11, comparing the wirelength-aware floorplan (blue) versus the wirelength-aware floorplan with the dynamic noise controller (black), the increase in average unit temperature is 3.2 kelvin. The maximum increase in unit temperature is 6.5 kelvin for the L1 data cache. The maximum temperature of the floorplan increases by 1.5 kelvin. The queue-aware floorplans are more effective at reducing power-supply noise with the dynamic noise controller, which causes the power consumption to increase by a small amount, as shown in Figure 10. For the queue-aware floorplans, the average unit temperature increases by 5.2 kelvin with the dynamic noise controller. The maximum increase in unit temperature is 6.8 kelvin for the data TLB. The maximum temperature of the floorplan increases by 5.8 kelvin. These thermal analysis results indicate that the use of the dI/dt controller does not create any large adverse thermal effects on our design.

Table III. Average and Maximum per-unit Temperature (for the wirelength-aware and queue-aware floorplans both with and without the dynamic noise controller)

	Wire-noctrl	Wire-w/ctrl	Qaware-noctrl	Qaware-w/ctrl
Average	319.5	322.7	321.7	326.8
Maximum	326.8	328.4	326.0	331.8

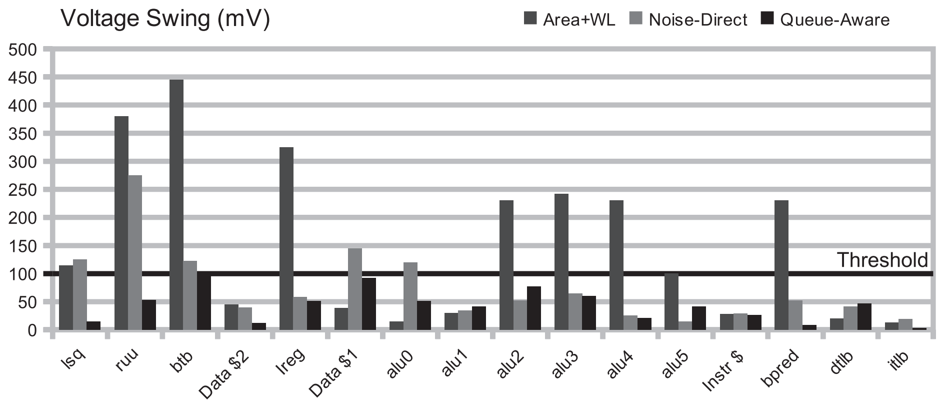


Fig. 12. Comparison with noise-direct. The voltage violation threshold is 0.1V. This comparison does not include the use of decaps.

The increase in temperature when comparing the wirelength-aware floorplan versus the queue-aware floorplan is more significant on a per-unit basis. However, this is not a reasonable comparison because the floorplans are completely different. Comparing floorplans from a thermal perspective makes sense only when using the average and maximum temperatures. Table III shows the average and maximum per-unit temperatures for the wirelength-aware and queue-aware floorplans, both with and without the dynamic noise controller activated. The results show a range of maximum temperature of less than 6 kelvin, which again demonstrates the nominal thermal impact of our techniques.

6.4. Noise-Aware Floorplan Results

To compare our floorplanning algorithm with previous work, we replicate the simulation and parameter infrastructure of Noise-Direct [Mohamood et al. 2007]. A comparison of the voltage swing of the controller-aware floorplan and the previously published numbers for Noise-Direct are shown in Figure 12. Queue-aware floorplanning results in an overall smaller voltage swing as compared to Noise-Direct. However, most significantly, it reduces the voltage swing to below the 10% voltage violation threshold of $0.1V$, and therefore there are zero noise constraint violations compared to the approximately 10% noise violations per cycle reported by Noise-Direct. For the purposes of comparison, there were no decaps included in the SPICE netlist for these voltage swing numbers. Therefore, no direct comparison between these values and those of the other experiments is logical. Given that Noise-Direct had no decap consideration at all, there is very little that could be comparable to the remaining experiments.

Next, a comparison between the traditional area-and-wirelength objective (A+W), floorplanning with *positive Q factor* (+Q), floorplanning *without the queue weights* (No Q), and the new controller-aware floorplanning with *negative Q factor* (−Q), all with decoupling capacitors added, is shown for voltage swing in Figure 13 and noise violations in Figure 14. A comparison between the +Q and −Q bars indicates a change in the cost function, switching the queue factor from positive to negative and shows that

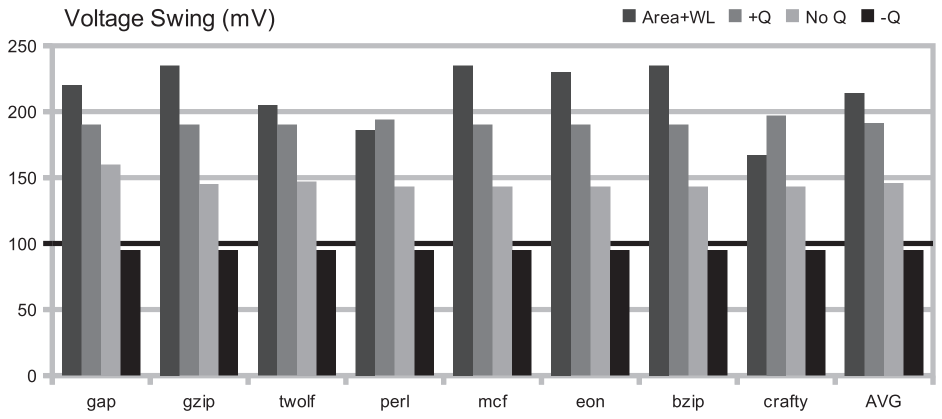


Fig. 13. Voltage swing comparison between area and wirelength, positive queue factor (+Q), noise-only (No Q), and negative queue factor (−Q). Decoupling capacitors and the decap allocation network flow are used for these results.

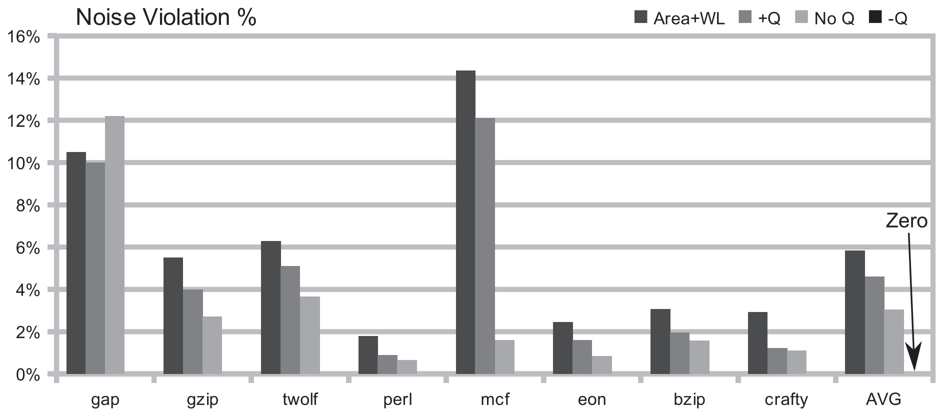


Fig. 14. Noise violation comparison between area and wirelength, positive queue factor (+Q), noise-only (No Q), and negative queue factor (−Q). (−Q) has zero violations. As in Figure 13, this data is generated with the decap allocation flow.

our initial intuition about the form of the queue factor was incorrect. As a reminder, the queue factor provides a bonus (in negative form) to the cost function whenever blocks with high correlation and current demand reside within the same dynamic controller queue. The No Q bars show a floorplan that has 0 for the ϵ weight, and thus is most similar to the work of Noise-Direct. However, as stated previously, due to the inclusion of decoupling capacitors here, no direct comparison of values between the two is logical. As shown in Figure 13, we can observe that the negative Q controller-aware floorplan has better noise characteristics than those of the traditional A+W objective, the positive Q objective, and the Noise only objective. The queue-aware floorplan has approximately 30% smaller voltage swing than the Noise only objective. This demonstrates that adding queue awareness to the floorplanner has a substantial impact for the simplicity of the change. The negative Q factor floorplan also reduces the voltage swing to below the violation threshold, and therefore there are no voltage violations for this floorplan, as shown in Figure 14. Additionally, the voltage swing graph reveals that the swing is independent of the benchmark for several experiments. This is the result of the dynamic controller fully controlling the coupled voltage swing of the processor. In those

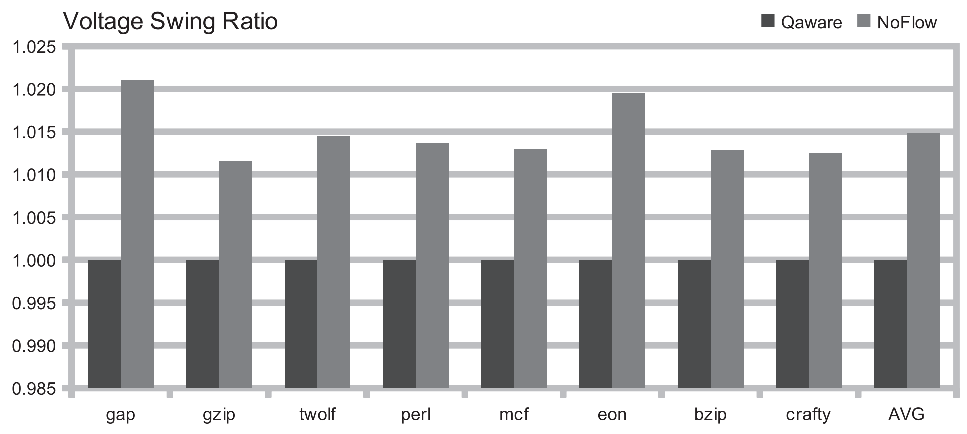


Fig. 15. Voltage ratio comparison between using the decap allocation flow (Queue-Aware) and the best according to the cost function (NoFlow). In the NoFlow case the decap allocation flow is not used to choose the best floorplan.

cases, individual module swings are fully responsible for the magnitude of the chip-level voltage swing. This indicates that the negative Q-aware floorplanner is the most effective method to use with the dynamic controller.

Finally, we show that the use of the network-flow-based decap allocation algorithm improves the dynamic noise results. A comparison of the voltage swing between the queue-aware floorplan and the top floorplan according to the cost function (NoFlow) is shown in Figure 15. In the NoFlow case, decaps are added in all the white space of the floorplan with the lowest cost function value. We can observe that for every benchmark the floorplan that utilizes the decap allocation flow has improved voltage swing. And, in fact, without the use of the decap allocation flow, the floorplan does violate the noise threshold by a small amount.

7. CONCLUSION

The exponential increase in the current consumption of newer generations of processors coupled with aggressive power-saving techniques have exacerbated the high-frequency dI/dt issue. If current trends continue, ad-hoc solutions that mitigate dI/dt effects using excessive decoupling capacitance will eventually become insufficient. Decaps not only occupy considerable chip area, but also contribute to the already problematic leakage power issue. Current microarchitecture-based solutions are inadequate for deep submicron designs where high-frequency dI/dt is intricately entwined with both the chip floorplan and power-pin distribution.

We have presented a unified design methodology that addresses the high-frequency dI/dt issues and maintains high reliability while alleviating the design cost of creating a low impedance power delivery network using a dynamic queue-based dI/dt controller and a controller-aware floorplanning algorithm. By leveraging microarchitectural profile information in the floorplanning stage, and by monitoring application-based module activity at run-time with our dynamic controller, we show that current demands can be guaranteed for modules residing within the same power-pin domain. In addition, we integrate a pre-emptive ALU gating mechanism as a performance enhancement technique as well as an enhanced progressive gating technique for large modules (L2 cache) into our queue-based control mechanism. We have also explained how the dI/dt architecture can be implemented in a conventional out-of-order pipeline in a complexity-effective manner. Experimental results show that our dI/dt controller can improve the current variability of applications by an average of 7x with a mere 4.0% IPC degradation. SPICE simulations show that the combination of our dynamic controller and controller-aware floorplanner can completely eliminate power-supply noise-margin violations.

Overall, our design provides practical microarchitectural and physical design approaches that can be used in concert to alleviate the effort of design afterthoughts and reduce the use of leaky decoupling capacitors that consume large chip area. Our technique incurs little performance overhead and has very little thermal impact.

REFERENCES

Austin, T., Larson, E., and Ernst, D. 2002. SimpleScalar: An infrastructure for computer system modeling. *IEEE Micro Mag.* 59–67.

Austin, T. M. and Sohi, G. S. 1995. Zero-cycle loads: Microarchitecture support for reducing load latency. In *Proceedings Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, New York, 82–92.

Brooks, D., Tiwari, V., and Martonosi, M. 2000. Wattch: A framework for architectural-level power analysis and optimizations. In *Proceedings of the IEEE International Symposium on Computer Architecture*. IEEE, Los Alamitos, CA, 83–94.

Chen, H.-M., Huang, L.-D., Liu, I.-M., and Wong, M. D. 2005. Simultaneous power supply planning and noise avoidance in floorplan design. *IEEE Trans. Comput.-Aid. Des. Integrat. Circuits Syst.* 578–587.

Chen, Y., Roy, K., and Koh, C.-K. 2005. Current demand balancing: A technique for minimization of current surge in high performance clock-gated microprocessors. *IEEE Trans. VLSI Syst.* 75–85.

Eble, J. C., De, V. K., Wills, D. S., and Meindl, J. D. 1996. A generic system simulator (GENESYS) for ASIC technology and architecture beyond 2001. In *Proceedings of the IEEE International ASIC Conference and Exhibit*. IEEE, Los Alamitos, CA, 193–196.

ECACTI. http://www.ics.uci.edu/ maheshmn/eCACTI/main.htm.

Grochowski, E., Ayers, D., and Tiwari, V. 2002. Microarchitectural simulation and control of di/dt induced power supply voltage variation. In *Proceedings of the IEEE International Symposium on High- Performance Computer Architecture*. IEEE, Los Alamitos, CA, 7–16.

Hazelwood, K. and Brooks, D. 2004. Eliminating voltage emergencies via microarchitectural voltage control feedback and dynamic optimization. In *Proceedings of the International Symposium on Low Power Electronics and Design*. 326–331.

Healy, M. B., Mohamood, F., Lim, S. K., and Lee, H.-H. S. 2008. A unified methodology for power supply noise reduction in modern microarchitecture design. In *Proceedings of the Asia and South Pacific Design Automation Conference*. 611–616.

Jacobson, H., Bose, P., Hu, Z., Buyuktosunoglu, A., Zyuban, V., Eickemeyer, R., Eisen, L., Griswell, J., Logan, D., Sinharoy, B., and Tendler, J. 2005. Stretching the limits of clock-gating efficiency in server-class processors. In *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture*. IEEE, Los Alamitos, CA, 238–242.

Joseph, R., Hu, Z., and Martonosi, M. 2004. Wavelet analysis for microprocessor design: Experiences with wavelet-based dI/dt characterization. In *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture*. IEEE, Los Alamitos, CA, 36–46.

Li, H., Bhunia, S., Chen, Y., Roy, K., and Vijaykumar, T. N. 2004. DCG: Deterministic clock-gating for low-power microprocessor design. *IEEE Trans. VLSI Syst.* 245–254.

Lu, C.-H., Chen, H.-M., and Liu, C.-N. J. 2008. Effective decap insertion in area-array SoC floorplan design. *ACM Trans. Des. Autom. Electron. Syst.*

Minz, J., Wong, E., Pathak, M., and Lim, S. K. 2006. Placement and routing for 3-d system-on-package designs. *IEEE Trans. Components Packaging Technol.*, 644–657.

Mohamood, F., Healy, M. B., Lim, S. K., and Lee, H.-H. S. 2006. A floorplan-aware dynamic inductive noise controller for reliable processor design. In *Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, New York, 3–14.

Mohamood, F., Healy, M. B., Lim, S. K., and Lee, H.-H. S. 2007. Noise-direct: A technique for power supply noise aware floorplanning using microarchitecture profiling. In *Proceedings of the Asia and South Pacific Design Automation Conference*. 786–791.

Murata, H., Fujiyoshi, K., and Kaneko, M. 1998. VLSI/PCB placement with obstacles based on sequence pair. *IEEE Trans. Comput.-Aid. Des. Integrat. Circuits Syst.* 60–68.

PANT, M. D., PANT, P., WILLS, D. S., AND TIWARI, V. 2000. Inductive noise reduction at the architectural level. In *Proceedings of the International Conference on VLSI Design*. 162–167.

PANT, S., BLAAUW, D., AND CHIPROUT, E. 2007. Power grid physics and implications for CAD. *IEEE Design Test of Computers*. 246–254.

POWELL, M. D. AND VIJAYKUMAR, T. N. 2003. Pipeline muffling and a priori current ramping: Architectural techniques to reduce high-frequency inductive noise. In *Proceedings of the International Symposium on Low Power Electronics and Design*. 223–228.

POWELL, M. D. AND VIJAYKUMAR, T. N. 2004. Exploiting resonant behavior to reduce inductive noise. In *Proceedings of the IEEE International Symposium on Computer Architecture*. IEEE, Los Alamitos, CA, 288–299.

SATO, T., ONODERA, H., AND HASHIMOTO, M. 2005. Successive pad assignment algorithm to optimize number and location of power supply pad using incremental matrix inversion. In *Proceedings of the Asia and South Pacific Design Automation Conference*. 723–728.

SKADRON, K., STAN, M. R., SANKARANARAYANAN, K., HUANG, W., VELUSAMY, S., AND TARJAN, D. 2004. Temperature-aware microarchitecture: Modeling and implementation. *ACM Trans. Architect. Code Optim.* 94–125.

WONG, E., MINZ, J., AND LIM, S. K. 2006. Decoupling capacitor planning and sizing for noise and leakage reduction. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. ACM, New York, 395–400.

ZHAO, M., FU, Y., ZOLOTOV, V., SUNDARESWARAN, S., AND PANDA, R. 2006. Optimal placement of power supply pads and pins. *IEEE Trans. Comput.- Aid. Des. Integrat. Circuits Syst.* 144–154.

ZHAO, S., KOH, C., AND ROY, K. 2002. Decoupling capacitance allocation and its application to power supply noise aware floorplanning. *IEEE Trans. Comput.-Aid. Des. Integrat. Circuits Syst.* 81–92.

Received July 2010; revised February 2011; accepted April 2011