A Floorplan-Aware Dynamic Inductive Noise Controller for Reliable Processor Design

Fayez Mohamood Michael B. Healy Sung Kyu Lim Hsien-Hsin S. Lee

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332
{fayez, mbhealy, limsk, leehs}@ece.gatech.edu

ABSTRACT

Power delivery is a growing reliability concern in microprocessors as the industry moves toward feature-rich, power-hungrier designs. To battle the ever-aggravating power consumption, modern microprocessor designers or researchers propose and apply aggressive power-saving techniques in the form of clock-gating and/or power-gating in order to operate the processor within a given power envelope. These techniques, however, often lead to high-frequency current variations, which can stress the power delivery system and jeopardize reliability due to inductive noise ($L\frac{di}{dt}$) in the power supply network. To counteract these issues, modern microprocessors are designed to operate under the worst-case current assumption by deploying adequate decoupling capacitance. With the trend of lower supply voltage and increased leakage power and current consumption, designing a processor for the worst case is becoming less appealing.

In this paper, we propose a new dynamic inductive-noise controlling mechanism at the microarchitectural level that will limit 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 controller 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%.

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, thereby posing a serious issue in circuit reliability. This is especially a concern in high-frequency processors where the supply-voltage needs to respond and stabilize to varying current demands without violating stringent timing constraints. In the worst case, overshoot or undershoot in the power supply network can adversely flip data values in the data path, resulting in incorrect computation. To address this reliability issue, processors are often over-designed, typically with the use of an excessive amount of decoupling capacitors (decap) that can warrant reliable operations under the worst case current consumption scenario. For increasingly complex processors, inserting an excessive amount of decaps, however, enlarges the chip area and at the same time exacerbates the leakage power. Moreover, significant design effort and cost for a worst-case design is inevitable to manage the infrequent cases where programs exhibit the maximum level of varying current demands during the course of execution.

Traditional technology scaling for CMOS is one reason that causes a high variability in current flow within a processor. As the dimension of devices keeps shrinking, the supply voltage is reduced as well in order to meet the gate-oxide reliability requirement. Such lowered supply voltage imposes a smaller absolute noise margin, exacerbating the inductive noise issue. On the other hand, the increasing number of available transistors on chip as well as the pursuit of ever-higher operating frequencies result in more power consumption. To mitigate power consumption and its ensuing thermal management problems, aggressive power-saving techniques such as clocking gating and/or power gating were widely studied and applied. Processors such as the Intel Pentium 4, Pentium M and IBM Power5 [3, 13, 17] use different levels of clock gating to dynamically disable portions of the circuits that do not change states. Meanwhile, 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 [13]. This is also one reason why ideal clock-gating, limiting power dissipation to only active modules, is difficult to attain in practical designs.

Conventionally, the worse-case current consumption can be profiled and gauged by exercising power virus programs [9]. These programs were written with a goal in mind — varying the execution behavior from extremely high activity to almost none for inducing drastic fluctuation of current demands to stress the power-delivery network. Such exercises provide an approximation of the maximal supply-voltage overshoot or undershoot conceivable in a design. Designers then allocate the appropriate amount of decaps to manage this worst-case voltage ringing in a repetitive process until a given module is within the noise margin. The drawback in such designs is that a significant amount of chip area (in the form of decaps) is devoted to the coverage of those infrequent corner cases. For example, the Alpha 21264 reported that roughly 15 to 20% of the die area is occupied by decaps [8]

The 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)
0-7695-2732-9/06 $20.00 © 2006 IEEE

and the trend is going up. Also note that, these decaps will contribute a considerable amount of leakage power in future deep submicron processors. Clearly, the trade-off in future processor designs can be rather complicated in determining the degree of gating sustainable by a chip, the area overheard, and additional leakage due to decaps.

To address these shortcomings in the worst-case design methodology, we propose a low-cost di/dt controlling mechanism embedded into the microarchitecture that will dynamically limit high-frequency di/dt below a predefined threshold. Our design can be integrated into the early microarchitecture planning stages to facilitate the design of a processor for the average-case current consumption scenario. The main contributions of this research include the following.

- We present the use of decay counters as a simple mechanism to monitor the access pattern of each microarchitectural module to prevent unsteady self-switching activity.
- We propose a novel microarchitectural technique using a *queue-based dynamic di/dt controller* to avoid overly prescribed simultaneous (or correlated) gating of modules that share the same local power-pins on the power delivery network.
- To simultaneously avoid performance loss as well as reduce high-frequency di/dt effect, we present the integration of *Preemptive ALU Gating* into our queue-based dynamic di/dt controller.
- Finally, to achieve fine-grained di/dt control for large modules, we present an enhancement to perform progressive clock-gating without violating the current demand threshold.

Unlike prior techniques [9, 14, 15, 20, 21, 22, 23, 24] that largely aim at providing chip-level di/dt control, our technique monitors and controls di/dt by leveraging spatial information of modules obtained from a given floorplan and its power-pin distribution. Inductive noise is highly dependent on the chip floorplan which determines the relative location of functional modules and their distance from the power-pins. Since power-pins will be stressed non-uniformly across the power supply network, certain modules will have a higher susceptibility to inductive noise. 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.

The rest of the paper is organized as follows. We begin with an outline of prior art and their limitations in Section 2. Section 3 describes power delivery issues in processor designs and the inefficacy of worst-case design. Section 4 describes the design of our dynamic di/dt controller. Section 5 discusses our experimental methodology followed by the evaluation in Section 6. Finally, Section 7 concludes.

2. RELATED WORK

The microarchitecture community has recently paid notable attention to di/dt issues due largely to the use of power-saving techniques such as clock-gating, and have proposed solutions to characterize and address them. Grochowski et al. [9] was one of the first to illustrate the criticality of di/dt and propose solutions from the perspective of architects. Their work showed that applications exhibit varying current characteristics in a large range and proposed a microarchitectural solution to improve current demands with a feedback control mechanism. Their proposed mechanism can dynamically estimate supply noise violations by performing current-to-voltage calculations on the chip and throttle instruction fetch or issue upon the detection of a reliability emergency. This work, however, addressed the mid-frequency di/dt problem at the chip-level and the architecture presented is incapable of altering current demands over a smaller clock-cycle interval (e.g. less than 25 cycles). In contrast, our work is targeted at mitigating the high-frequency di/dt issue to enable average-case microprocessor design and reduce the on-die decap required.

In [14, 15], Joseph et al. analyzed power supply response and control voltage emergencies in a processor via microarchitectural techniques. They concentrated on the worst case di/dt occurring at the *resonance frequency* of the power supply in the 50-100MHz range. Similarly, Powell and Vijaykumar proposed techniques in [24] to mitigate the reliability issues caused by resonant frequency. Another technique called *Pipeline Damping* in [22] by the same authors throttles microarchitectural activity at the front-end and the back-end to alter the current surges at the resonant frequency.

In contrast to pure hardware-based techniques, Hazelwood and Brooks proposed a hybrid hardware/software approach to address the mid-frequency di/dt issue [10]. Their work performed dynamic optimization to alter the program codes that induce large di/dt oscillation via a compiler's assistance. They showed that software pipelining, code motion and instruction padding can modify program behavior that causes di/dt at the resonance frequency, while avoiding performance overhead.

Different from the above mentioned work tackling the *mid-frequency di/dt* issue (50-100MHz), the focus of this paper is to mitigate *high-frequency di/dt* that requires immediate response, for which the above solutions are inappropriate. Toward this effort, Powell and Vijaykumar proposed *Pipeline Muffling* [23] which controls instruction issue and limits the use of resources for the high-frequency di/dt concerns. Tang et al. [26] proposed controlled ramping of FPUs via scanning the IFQ for upcoming instructions. Note that the cause of high-frequency di/dt is highly dependent on the spatial distribution of modules across the floorplan and their distances from the power-pins. In addition, the 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 works accounted for this fact, which could result in violated current demand guarantees or false alarms.

Outside the microarchitecture domain, several solutions were proposed at the circuits level to address high frequency di/dt [5, 19, 27]. Most of these techniques tried to reduce the impedance path to individual modules in a processor, minimizing the voltage surges and dips. Floorplanning algorithms with the objective of minimizing inductive noise were studied recently [5, 6, 18]. Unlike this work, they are completely static solutions. Despite they can improve the average-case noise problem, the worst-case events still needs to be guaranteed due to the fact that static solutions cannot exploit or react to dynamic program behavior. Note that our technique is complementary to such circuit solutions that improve the average-case voltage swing. An ideal design, also our advocate, is to involve optimizing a floorplan and its power supply network for the average-case inductive noise and integrating our dynamic controller to prevent the worst-case current demand at a given power pin domain. Since worst-case program behavior is infrequent, our low-overhead technique can trade off nominal performance to meet the current demand threshold.

3. HIGH-FREQUENCY INDUCTIVE NOISE ISSUES

To address inductive noise issues that result from abrupt changes in the dynamic current demands, designers typically target a low-impedance power delivery network by deploying adequate decoupling capacitors.[1] In order to meet the impedance target across a wide range of frequencies, multi-stage decoupling capacitors are necessary. High-frequency noise is handled by decaps that are distributed throughout the die. Medium-frequency decaps are typically placed on the land side of the package, as close as possible to the motherboard, to facilitate the lowest possible impedance. Finally, bulk capacitors on the motherboard address the low-frequency current fluctuations. Since our work targets the the high-frequency di/dt issue, this section will describe some key issues responsible for exacerbating high-frequency di/dt in deep submicron designs.

3.1 Sources of High-Frequency Inductive Noise

General purpose processors run a wide variety of applications; the current profile for each application varies depending on many factors. Generally, applications with high ILP typically exhibit constant use of most of the modules in the processor, resulting in less current variability. In contrast, applications that oscillate between high and low level activities will display a more irregular current profile. With dynamic clock-gating for idle functional units, the abrupt current variation is even more prominent. The current profile also correlates closely to program phases. For instance, a program might consistently performs simple arithmetic operations in a certain phase, leaving little activity in the caches. From a fine-grained perspective, however, even consecutive instructions can vary the current demands substantially if the functional units they exercise are completely different. Although a program appears to be in a consistent phase with a regular instruction profile for thousands of cycles, minute irregularity in-between consecutive instructions can still cause an unexpected current surge or dip, resulting in detrimental voltage spikes. Furthermore, the exact same instruction can generate a different current profile due to dynamic effects like cache misses. For instance a LOAD instruction that hits in the cache versus the same instruction that misses the next time will create different module access patterns and clock-gating activity. Therefore, understanding and exploiting current profiles of applications requires a much finer grained control.

Microarchitectural modules have non-uniform current demands and it is critical to create a low impedance path to modules demanding high current. Similarly, modules that induce high current fluctuation can create a greater burden on the power supply, if they are placed close to each other and have a high probability of switching simultaneously. Note that simultaneous switching events along the same direction raises a major issue to power delivery. A floorplan that is resistant to inductive noise tries to generate a well balanced layout to distribute the current demand in a more regular manner across the power-supply grid [5, 6, 18]. Nonetheless, floorplanning is a static solution. While a noise-aware floorplan can mitigate di/dt effects to a certain extent, it is still unable to completely eliminate the dynamic reliability emergency due to high frequency inductive noise.

3.2 Quantifying Module Activity

To effectively manage and prevent the high-frequency voltage ringing at the microarchitectural level, it is imperative to understand the simultaneous switching behavior among different microarchitectural blocks and their relative locations to each other (i.e. the sharing of power-pins) on a given floorplan. To understand switching behavior of microprocessor modules, we describe two metrics: the self-switching activity and the correlated or simultaneous-switching activity of modules. The self-switching measurement is used to quantify the number of gating occurrences of a given module during the profiling period. Both gating on and off are considered likely events to cause current fluctuation. The objective of this metric is to single out the microarchitectural modules of high switching activity. In addition, the intensity of the gating activity also depends on the current consumption of each module. In other words, even if a module switches less frequently than the others, it still can induce intolerable noise if it draws a significant amount of current. The relative number of switching events and the current consumption per cycle called *intensity of switch* are combined into a single weight. If sw_i represents the relative number of switching events for module i and I_i is the intensity of the switch, then the self-switching factor α_i is represented by the following relationship.

$$\text{Self-switching factor, } \alpha_i = sw_i \times I_i \tag{1}$$

$$\begin{bmatrix} \alpha_1 & \frac{1}{2}\left(\frac{X_{12}}{sw_1}+\frac{X_{21}}{sw_2}\right) & \dots & \frac{1}{2}\left(\frac{X_{1n}}{sw_1}+\frac{X_{n1}}{sw_n}\right) \\ 0 & \alpha_2 & \dots & \frac{1}{2}\left(\frac{X_{2n}}{sw_2}+\frac{X_{n2}}{sw_{n-1}}\right) \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \alpha_n \end{bmatrix}$$

Figure 1: Correlated Switching Matrix

Correlated switching events are the gating events in the same direction i.e. both modules are clock-gated ON or OFF simultaneously. To measure correlation, we capture the inter-cycle gating direction of each module in the profiling process. Then each module is paired with every other module in the processor, and checked for simultaneous gating in the same direction. The result is an upper triangular correlation matrix with each location representing the number of simultaneous gating events encountered. An illustration of the calculation process of correlated switching events is given in Figure 1. In the matrix, X_{ij} is the number of raw correlated switches that occurred over the profiling duration and sw_i is the number of self-switching events for module i. Note that the correlation metric X_{ij} isolates only the modules in consideration. The upper bound of 100 indicates a perfect correlation, i.e. modules i and j switched simultaneously every single time[2] (100% of the switching events). The forward diagonal in the same matrix represents the self-switching factor, α_i, for each module.

Using eight SPEC2000 INT benchmark programs,[3] Table 1 shows the switching correlation as a matrix for 23 microarchitectural modules considered in our processor model. The diagonal (gray) in the matrix represents the amount of *self-switching* factor. As observed from Table 1, certain modules switch far more frequent than others. On the other hand, the weights of the modules that are likely

[1] Current designs target at sub-milliohm impedance.

[2] Please note that the correlated switching factor was scaled as a percentage, hence the upper bound is 100.

[3] Since correlation profiling was compute intensive, we used the following subset of benchmarks for this motivational data: 256.bzip, 186.crafty, 252.eon, 254.gap, 164.gzip, 181.mcf, 253.perl and 300.twolf.

	LSQ	RUU	BTB	L2$	IRF	L1D$	ALU1	ALU2	ALU3	ALU4	ALU5	ALU6	L1I$	Bpred	DTLB	ITLB	FALU1	FALU2	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*

Table 1: Self and Correlated Switching Weights of Modules

to be accessed every cycle (turned on mostly) such as the L1 I-Cache and the I-TLB are lower. Some modules with smaller weights are dormant, e.g. floating-point register file (Freg),[4] only accessed once in a long while. In addition, as expected, the branch predictor and the BTB, the I-Cache and the I-TLB (and the D-Cache and the D-TLB) are all highly correlated modules. In addition, it is also observed that the first six ALU modules are also highly correlated as concurrency exists in integer instructions. The design of our high-frequency dynamic di/dt controller is mainly based around the intrinsic switching behavior of microarchitectural modules.

Our technique addresses the high-frequency inductive noise issues directly caused by clock-gating. Clock-gating is a well established method in dealing with the increasing power concern and thermal pressure. The main issue in reliability associated with clock-gating is that there is no deterministic or predictive way for determining whether it is reliable to gate off modules without inducing hazardous current surges. In addition, conventional clock gating techniques do not have any knowledge of adjacent modules and the extent of correlated clock-gating activity. Our microarchitectural level high-frequency di/dt controller is based on such fundamental observations on the clock-gating activity of modules, their correlation with adjacent modules, the module locations in a floorplan, and the power-pin distribution of the chip.

4. A FLOORPLAN-AWARE, QUEUE-BASED DYNAMIC di/dt CONTROLLER

In order to address inductive noise issues due to high switching activity in the processor, we now present the design of our dynamic di/dt controller that aims to improve the current profile of a processor regardless of program behavior. Our design is easily customizable, in order to enable a given design achieve the right balance among dynamic di/dt control, power consumption, and performance overhead. The primary components of the di/dt controller include 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.
- Preemptive activation of ALUs through pre-decoding for simultaneous di/dt and performance enhancement.
- An enhancement to queue controller in order to enable progressive clock-gating on large modules like L2 cache banks.

[4]Because Table 1, used a demonstration, only profiled results of integer benchmark programs.

4.1 Decay Counter based Clock-Gating

The key to avoiding clock-gating induced noise lies in identifying program phases to see whether it is reliable, at a particular moment, to gate off an entire microarchitecture module. Although certain elaborate techniques can accurately predict module requirement patterns, clock-gating requires low-overhead mechanisms to justify the extra hardware cost [17]. To enable a low-overhead, dynamic clock-gating scheme while providing a tunable form of di/dt control, we propose the use of decay counters. By using low-resolution decay counters to monitor module access patterns, we can choose to save power only during long-stretches of inactivity. To illustrate this, we use an example that quantifies fine-grained module access patterns of certain processor modules over a small simulation period in Figure 2. The figure shows an example of access pattern profile for the branch predictor, the L1 I-Cache, an Integer ALU and the Integer Register File for bzip. The 200-cycle interval is shown here to illustrate the potential high-frequency di/dt effects from a fine-grained perspective. It is to be noted that the decay counter does not require a specific access patten to eliminate unnecessary switching activity, such as the ones presented in the figure. The 200-cycle access pattern for different modules with varying access patterns is merely used to illustrate the significance of employing decay counters in our design. Typically, it is observed that a module that is inactive for more than 10-12 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 gated-off reliably with the least likelihood of encountering high frequency inductive noise. On the other hand, it can be seen that when a module is not accessed for less than 5-10 cycles, it is highly likely to be accessed soon in subsequent few cycles. A decay counter is employed to exploit this behavior by enabling clock-gating activity only when a minimum turn-off threshold has exceeded. We use a 4-bit decay counter for each microarchitecture module inside the processor that only permits clock-gating of a module if it has not been accessed during the last 16 cycles. For any given module, the counter decays 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 trade-off 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 A Floorplan-aware Queue Based di/dt Controller

Even though the decay counter can provide a smoother current profile for each module by eliminating unwanted switching activity, it is 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* which is aware of the pro-

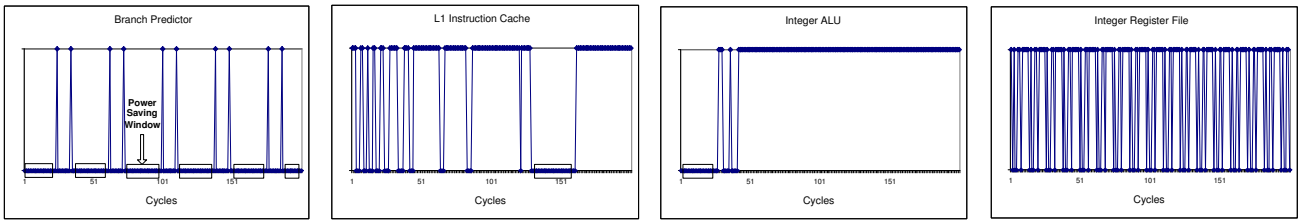


Figure 2: Module Access Patterns

cessor'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 a 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 described 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.[5]
- Every queue-entry has the corresponding *state* of the module that 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→OFF and OFF→ON transitions. The *state* is used to drive the pre-wired 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 use 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.

Our high frequency di/dt controller architecture is depicted in Figure 3. The "+" signs on the chip floorplan (left-hand side) indicate the power-pins locations on the power delivery network. For simplicity, we illustrate only four power-pins. The queue based controller works in the following manner. The decay counter will signal a transition event, i.e. ON→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 the weights of the consecutive modules that are in the transition states (ON→OFF or OFF→ON), such that $\gamma \leq \delta$. Since integer weights can be negative as well,[6] the sliding window will attempt to permit the maximally 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, $\delta = 3$. In the figure shown, ALU-2 and ALU-3 are gated off (indicated by the bigger, bold arrows that are the output of the queue controller). Both Bpred and ALU-1 have an activation request indicated by the OFF→ON state. Therefore, the combined weight of the sliding window, $\gamma = 3$.[7] 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 has a higher weight that results 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 the ALU-1 requires an ON→OFF transition which represents a negative weight. In this case, γ=1, thus still permitting both Bpred and ALU-1 to perform their transitions. In this case, however, the sliding window threshold is still 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 show the modules in the descending order of weights. It is to be noted that the di/dt controller will enforce the current demand threshold regardless of the order in which they are in the queue. Nevertheless, 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 simultaneously 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 the architects for a given floorplan.

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 pre-wired for each module at design time to simplify the logic for driving clock-gating signals directly to the modules.[8] However,

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

[6]OFF→ON is a positive switch while ON→OFF represents a negative switch.

[7]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.

[8]Since the queue entries are pre-wired 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 transition at a given time.

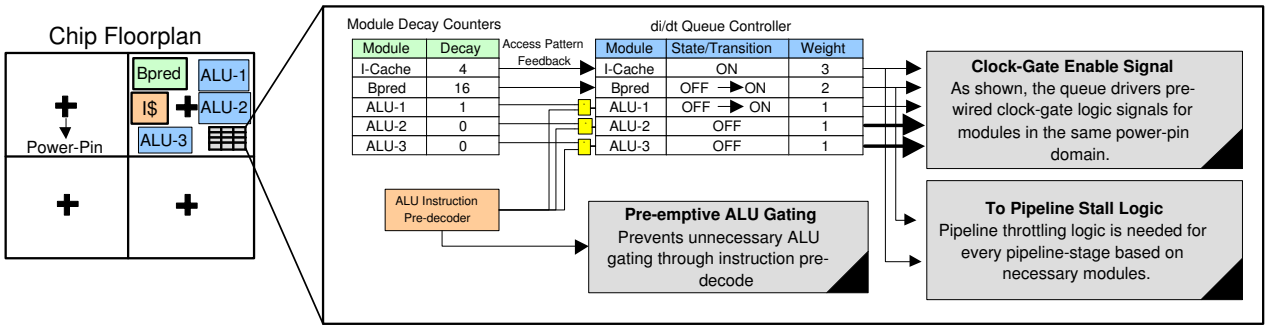


Figure 3: di/dt Controller Architecture

functional-wise the controller works like a circular queue that traverses as many modules as indicated by the sliding window threshold. It is to be noted that the maximum hardware overhead of each microarchitectural module is merely 11 bits (including the decay counter). Also, the additional power dissipated and the extra current drawn by the controller itself are rather negligible for these small sizes.

4.3 Preemptive ALU Gating

Preemptive ALU clock-gating through pre-decoding instructions is another technique we propose to prevent unnecessary gating activity. Note that decay counter based clock-gating allows gating events to occur based on the history of module accesses. However, decay counters by itself will be unable to predict the requirement of a module if it is required in the immediate future for a recently fetched instruction. 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 pre-decoding instructions. In a typical RISC ISA, the opcode can be determined by observing the first few bits of the instruction,[9] allowing the processor to pre-decode 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. While in a CISC ISA, it might not be easily possible to perform a simple pre-decode due to variable length instructions, but even so, other techniques such as storing pre-decode information in the L1 Instruction Cache [2] can be used to achieve this effect.

4.4 Enhanced Progressive Gating of Large Modules

Even though simultaneous gating of multiple modules can be prevented completely by selective gating for a given set of power-pins in a power-delivery network, some monolithic modules like the L2 cache can still consume large current resulting in unreliable voltage swing. For this reason, certain processors employ progressive gating of large modules like the L2 cache to mitigate di/dt effects [13]. 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.[10] 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 noise-tolerant current demand threshold through mitigation of simultaneous switching effects.

4.5 Pipeline Design Implications

The employment 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 the operations. The integration of a di/dt controller into a conventional architecture will require the pipeline logic to be accurately aware of the clock-gating state of the module as well, in order to issue operations without affecting correctness. For this reason, it is essential that the di/dt controller not impose impractical design implications 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 events of resource hazards such as over-subscription of ports in register file, caches, or load-store queue, the selection logic will appropriately delay certain operations from issuing. As indicated in Figure 3, our queue has static entries and pre-wired logic that indicates the availability of any given module. This makes it efficient to integrate the additional resource availability constraint into 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. EXPERIMENTAL METHODOLOGY

Due to the fact that our design leverages spatial information of modules and power-pins from a given chip-floorplan, we will now briefly describe the floorplanning algorithms we employed to create our floorplan. The specific floorplan we used is independent of running applications, i.e. no profile-guided optimizations were employed in the floorplanning al-

Such optimizations however are out of the scope of this work.

[9]For example, Alpha and PowerPC ISA uses the prefix 6 bits for opcode.

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

0-7695-2732-9/06 $20.00

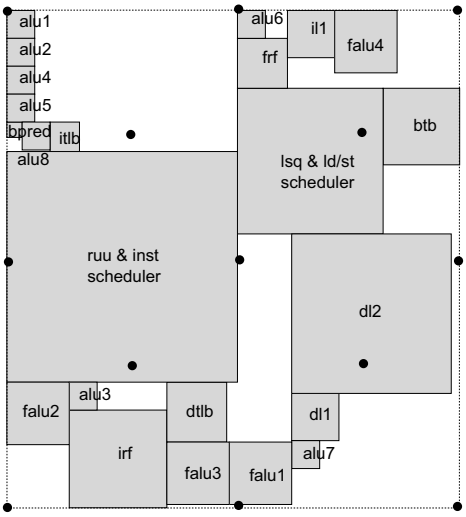


Figure 4: Floorplan. Black dots denote power pin locations.

gorithms. The obtained floorplans along with the predefined power-pin distribution determined the configuration of the queue entries in the dynamic di/dt controller.

5.1 Floorplanning Methodology

Our dynamic di/dt controller is general enough to work for any floorplan since the queue configuration is determined by the given module and power-pin location. Thus, the architecture of the queue-based controller is universally applicable to any given floorplan to achieve reliable di/dt fluctuation. To gain more insight into our floorplanning process, we briefly describe the basics of the floorplans and provides details on how they were obtained. Our floorplan contains 23 modules whose areas are determined by the machine configuration presented in Table 2.

The goal of floorplanning is to determine the width, height, and x/y location of the microarchitectural modules. The objective in our case is to minimize a weighted sum of the overall footprint area and the total weighted length of interconnects for IPC optimization [7]. We use a two-step approach: Linear Programming (LP) based floorplan construction followed by Simulated Annealing (SA) [16] based floorplan refinement. Our final floorplan along with their power pin locations are shown in Figure 4. The black dots "•" in alternating columns represent the power-pin locations on the power grid.[11]

The basic objective of our dynamic di/dt controller is to minimize the burden on power-pin(s) caused by adjacent modules to a reliable level. 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 will 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, because of the fact that the worst-case delay in transition is higher. A queue was assigned to each quadrant for all the modules placed in it. Since the floorplan determines the queue configuration, different floorplans will have different performance impact as well as distinctive di/dt characteristics.

[11]This is a type of power-pin configuration that certain flip-chip IC designs use.

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)

Table 2: Microarchitecture Parameters

5.2 Simulation Framework

Our simulation framework is based on SimpleScalar 3.0 and Wattch [4] running 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 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 2. The power and current consumption metrics were based on a 5GHz processor developed using a 70nm process technology [1]. Each simulation was fast-forwarded by 4 billion instructions and simulated for 1 billion instructions. The current signature that was chosen to evaluate the dynamic di/dt controller was obtained by profiling for the worst-case overall module activity over the entire simulation period. To study the thermal impact of our di/dt controller, we integrated Hotspot 3.0 [25] into our simulators. Hotspot assumed the same process technology parametric as mentioned earlier. The heat-sink and heat spreader modules were obtained from the default model and the initial temperatures were set to 300 kelvins.

6. QUANTITATIVE ANALYSIS

In order to evaluate the effectiveness and overhead of our dynamic di/dt controller under different scenarios, we applied our technique to the previously described floorplan. The results presented include current profiles on a baseline machine without a di/dt controller versus our technique and the average current variability across all benchmarks. Since di/dt is a reliability issue, we also quantify any potential reliability impact due to our technique in the form of thermals. Finally and most importantly, we present the performance overhead incurred due to our dynamic di/dt controller.

6.1 Current Profile of Applications

To demonstrate the effectiveness of our controller in improving high-frequency di/dt effect, we now present the current profile of the whole chip as well as for each queue clus-

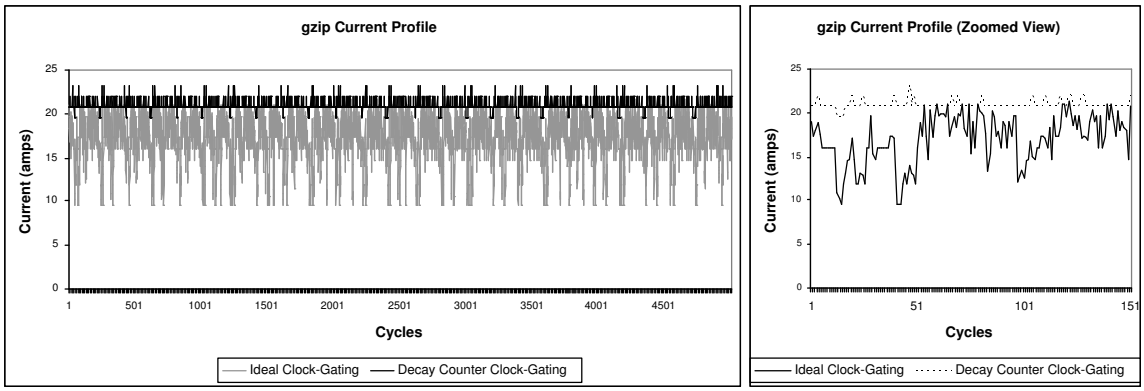


Figure 5: High ILP Benchmark Current Profile (164.gzip)

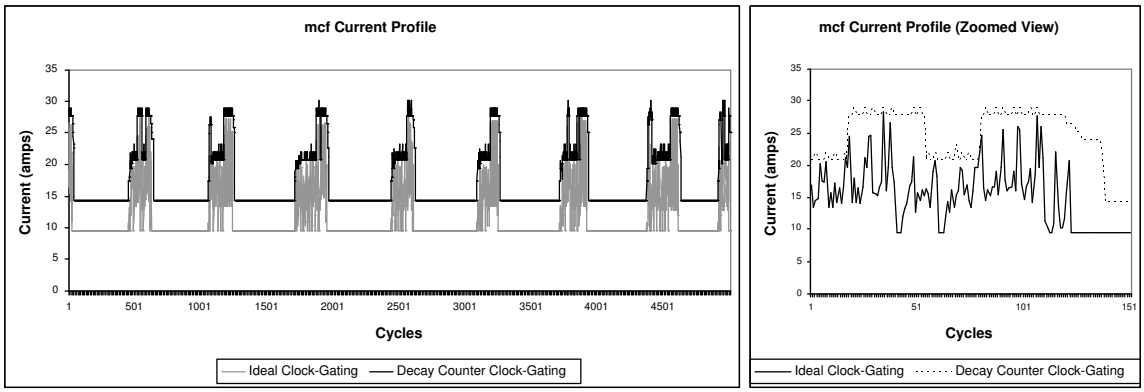


Figure 6: Low ILP Benchmark Current Profile (181.mcf)

ter for the 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 maximum switching activity of modules. Due to the staggeringly huge amount of current profiles of all benchmark programs, we epitomize their representative characteristics using two types of benchmark programs for this specific study as a demonstration of our analysis. Note that the crucial information conveyed in this section is to show the effectiveness of our proposed mechanism.

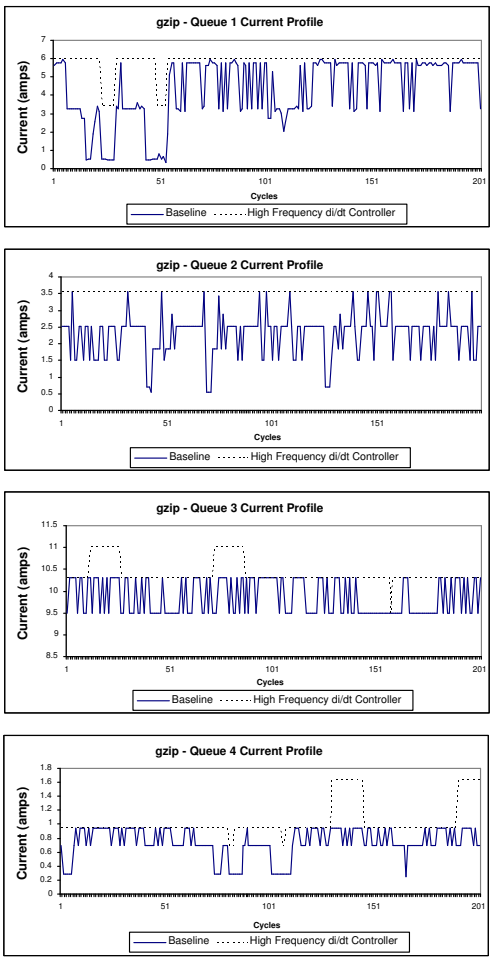
To explain the current profiles, we profiled one high-ILP benchmark (164.gzip) and another low-ILP (181.mcf, memory-bound) benchmark. The current profiles shown in Figure 5 and Figure 6 were obtained by profiling for the worst-case switching activity during the course of execution. A 4-bit decay counter was used for each module in all experiments.[12] 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 their close-up versions (right-hand side) of the representative portions, 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 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 mcf. During which 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 swing in the current profile and produces a much smoother down-ramp. For gzip in Figure 5, we observe large current variation in the ideal-clock gating scheme due to high activity across all modules. Since there is no significant duration of time where reasonable power savings are possible Because that modules are never inactive for extended periods of time, the decay counters rarely clock-gate off most 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 significant likelihood that the given modules will unlikely be accessed again soon.[13]

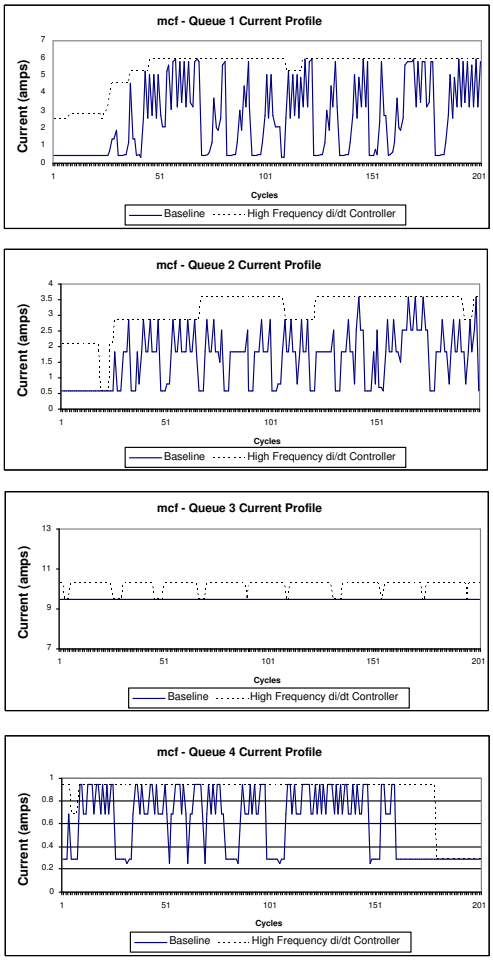
Next, we present the current profile with the integration of the complete queue-based controller. Note that this is the complete controller that incorporates prevention of simultaneous switching, decay counter based feedback for clock-gating, preemptive ALU gating and progressive gating of L2 cache banks. Figure 7 shows the current profile for all four queues for gzip and mcf. In all cases it can be observed that the current profile is significantly improved by eliminating excessive switching activity. In addition, it can be observed that both the upward ramp and downward ramp effects due

[12] The resolution of the decay counter was based on the motivational data discussed in Section 3.

[13] Note that the chip level current is with the decay counter based technique alone, which alone does not prevent simultaneous switching. Large upward ramps are resolved by the queue-based controller.



(a) High ILP Benchmark Queue Controller Current Profile (164.gzip)



(b) Low ILP Benchmark Queue Controller Current Profile (181.mcf)

Figure 7: Queue Controller Current Profile

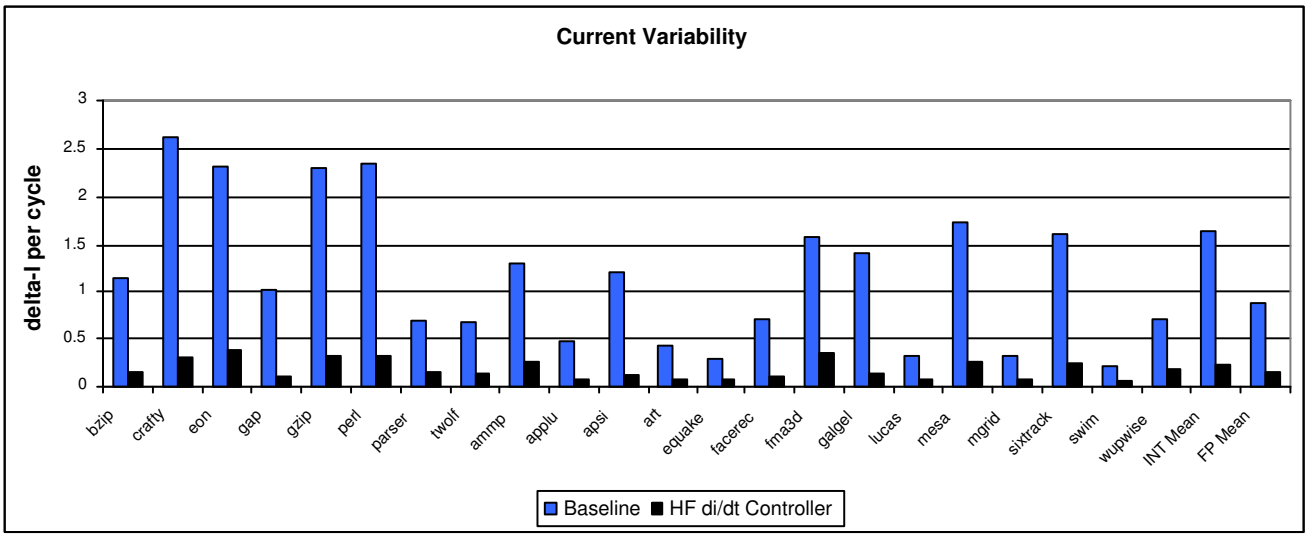


Figure 8: Current Variability

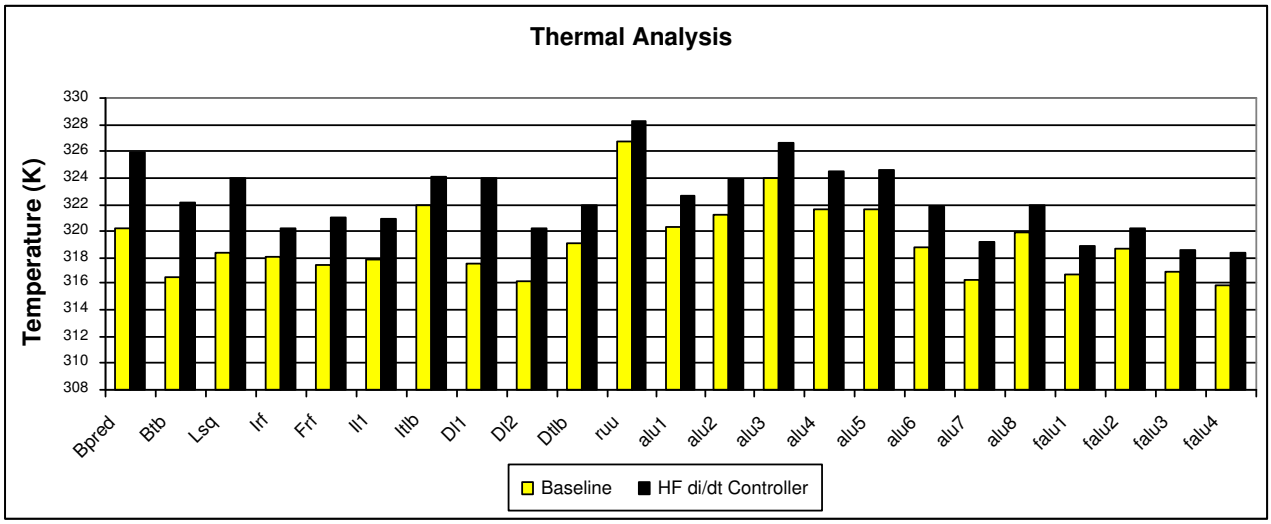


Figure 9: Thermal Impact of Dynamic di/dt Controller

to multiple modules in the same power pin domain (i.e. using the same module queue) are spread out 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 mcf. For *Queue 3* in 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 20-25 clock cycles. However, these ramps are still spread out over many cycles and do not violate the current demand threshold. In the case of *Queue 4*, although there is a significant current decay towards the end, it is to be noted that the simultaneous gating is prevented even in this case (the slope of the drop is less steep, which is not obvious in the graph due to the scale). For gzip, where there is high ILP/switching activity, we notice that the queue-based controller ramps up to the required current levels and do 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 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.

Since presenting detailed current profile is infeasible for all benchmarks, we now present the current variability per cycle for the complete duration of the benchmark execution. Unlike the worst case profile that was presented earlier, this metric presents the *average variability of current per cycle* for both the baseline 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 inter-cycle current fluctuations (in 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 show that 186.crafty exhibits the highest variability whereas 171.swim has the lowest variability. In any case, regardless of the native current variability, our dynamic di/dt controlling mechanism can significantly mitigate the dynamic oscillating behavior of current profile of running applications. The di/dt controller pushes the current variability below 0.5 amps/cycle for all the benchmark programs we studied. Note that, a traditional power-virus will no longer be able to stress the power delivery network in the presence of our di/dt controller.

6.2 Thermal Impact

In typical high-performance processor design, high-frequency inductive noise issue is handled through the worst-case design method (e.g. using on-die decaps). In contrast, the goal of our technique is to guarantee this reliability by enabling an average-case design, while meeting the stringent reliability requirements via dynamic control mechanisms. Therefore, it is critical that our di/dt controller must not induce other forms of reliability vulnerability. 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 due to our technique. Thermal issues are particularly critical in newer processors for their higher power density as well as the greater difficulty in dissipating heat across multiple die layers.

We used Hotspot 3.0 [25] to evaluate the thermal impact of our high-frequency di/dt controller on the given floorplan. We compared our architecture against the baseline designs that uses ideal-clock gating, which represents the scenario of the least power and current consumption. Figure 9 presents the thermal analysis for all 23 modules in our processor model for SPEC2000 benchmark suite.

Overall, we observe nominal thermal impact across all modules. We observe an average temperature increase of 3.15 kelvins over the baseline counterparts for the floorplan. The highest temperature rise (over 5 kelvins) is observed in the L1 Data Cache, Branch Predictor, BTB and LSQ modules. Majority of the remaining modules exhibit an average temperature increase below 3 kelvins. Note that it is possible to further mitigate these worst-case thermal effects by using a thermal-aware floorplanner described in [11, 12], however this is outside the scope of this paper. (Our floorplans were

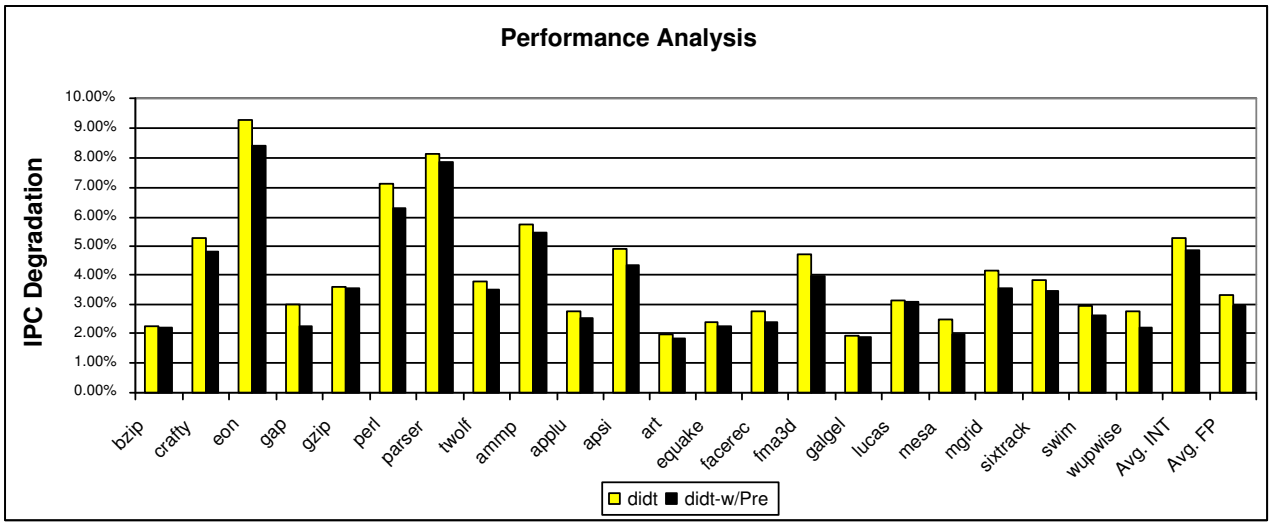


Figure 10: Performance Degradation of dynamic di/dt controller

generated with a goal of minimum total wirelength and die area.) Our thermal analysis results indicate that the integration of the di/dt controller does not pose any large adverse thermal effect on our overall design.

6.3 Performance Impact

We now present the performance analysis of our di/dt controlling mechanism. Figure 10 shows the IPC degradation for SPEC2000 INT and FP benchmark suite with the di/dt controller over the baseline machines without any di/dt control. The *di/dt-w/Pre* configuration shows the queue controller with preemptive ALU gating turned on 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 benchmark. Note that the performance overhead is also dependent on the floorplan because it affects the queue configuration. A more optimized floorplan will result in a better 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 due to the fact that the worst-case module activation time is longer. We observe an an average performance overhead of 4.0% for the floorplan we simulated. The worst performance degradation is shown in 252.eon, at 9.2% for the controller without preemptive ALU gating. One explanation for the increased performance impact is due to the fact that the ray-tracing algorithm in eon is ALU intensive. Since modules in the floorplan are asymmetric, the floorplan results in locally clustered ALUs that are not symmetrically distributed in all queues. Highly ALU intensive applications will suffer a performance loss in such cases, since a quick ramp-up of modules will take longer if most of the ALUs are clustered in the same queues. A strong indicator of this fact is evident from the higher sensitivity eon shows to preemptive ALU gating, compared with the other benchmarks. Most of the other benchmarks only exhibit little performance loss that is below 5%.

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 (resulting in turning 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 solely based on the past activity profile. In contrast, preemptive gating can "look-ahead" and override unnecessary gating that the decay counters themselves cannot prevent, thereby inhibiting unnecessary performance loss. The minimal IPC overhead illustrates the practical potential of employing a low-overhead technique to control high-frequency di/dt.

7. CONCLUSION

The exponential increase in current consumption by newer generations of processors coupled with aggressive power saving techniques have exacerbated the high-frequency di/dt issue that forces designers to elongate the design time in the analysis and implementation of the power delivery network. As long as the current trends in process and performance scaling continue, ad-hoc solutions to mitigate di/dt effects using an adequate decoupling capacitance will not suffice eventually. Decaps not only occupy considerable chip area but also but also contribute the already problematic leakage power issue. Current microarchitecture based solutions are inadequate for deep submicron designs where high-frequency di/dt is intricately intertwined with the chip floorplan as well as the power-pin distribution. In addition, the high module density facilitated by deep submicron technologies will stress the power delivery network even further, worsening reliability due to di/dt.

To address the high-frequency di/dt issues and maintain high reliability while alleviating the design effort of creating a low impedance power delivery network, we propose a dynamic queue-based di/dt controller for reliable processor design. By using decay counters to limit clock-gating activity based on module access patterns and by using this feedback in a queue-based di/dt controller, we show how current demands can be guaranteed for modules in the same power-pin domain. In addition, we also present a preemptive ALU gating mechanism as a performance enhancement technique and integrate an enhanced progressive gating technique for large modules (e.g. L2 cache) into our queue-based control mechanism, without violating current demand thresh-

olds due to simultaneous switching. In addition, we also explain how the di/dt architecture can be integrated into a conventional out-of-order pipeline in a complexity-effective manner. The 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 for the simulated 2D floorplan.

The high-frequency di/dt noise will keep deteriorating due to the continuing CMOS scaling that drives down the operating voltage while simultaneously increasing peak power consumption. In overall, our design provides a realistic microarchitectural approach that can be used to alleviate the effort of design afterthoughts and reduce the use of extensive decoupling capacitors that consume larger chip area. Our technique also incurs little performance overhead and does not have any adverse thermal impact.

8. ACKNOWLEDGMENT

This research was supported by the MARCO C2S2 and GSRC Centers.

9. REFERENCES

[1] International Technology Roadmap for Semiconductors. 2004.

[2] T. M. Austin and G. S. Sohi. Zero-cycle Loads: Microarchitecture Support for Reducing Load Latency. In *Proceedings of the 28th annual International Symposium on Microarchitecture*, pages 82–92, 1995.

[3] B. Bentley. Validating the Intel Pentium4 Microprocessor. In *Proceedings of the 2001 International Conference on Dependable Systems and Networks*, pages 493–500, 2001.

[4] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In *Proceedings of the 27th annual International Symposium on Computer Architecture*, 2000.

[5] H.-M. Chen, L.-D. Huang, I.-M. Liu, and M. D. F. Wong. Simultaneous Power Supply Planning and Noise Avoidance in Floorplan Design. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 24(4):578–587, 2005.

[6] Y. Chen, K. Roy, and C.-K. Koh. Current Demand Balancing: a Technique for Minimization of Current Surge in High Performance Clock-gated Microprocessors. *IEEE Transactions on Very Large Scale Integration Systems*, 13(1):75–85, 2005.

[7] M. Ekpanyapong, J. R. Minz, T. Watewai, H.-H. S. Lee, and S. K. Lim. Profile-Guided Microarchitectural Floorplanning for Deep Submicron Processor Design. In *Proceedings of the 41st Design Automation Conference*, pages 634–639, 2004.

[8] M. K. Gowan, L. L. Biro, and D. B. Jackson. Power Considerations in the Design of the Alpha 21264 Microprocessor. In *Proceedings of the 35th Design Automation Conference*, pages 726–731, 1998.

[9] E. Grochowski, D. Ayers, and V. Tiwari. Microarchitectural Simulation and Control of di/dt-induced Power Supply Voltage Variation. In *Proceedings of the 8th International Symposium on High-Performance Computer Architecture*, 2002.

[10] K. Hazelwood and D. Brooks. Eliminating Voltage Emergencies via Microarchitectural Voltage Control Feedback and Dynamic Optimization. In *Proceedings of the 2004 International Symposium on Low power Electronics and Design*, pages 326–331, 2004.

[11] M. Healy, M. Vittes, M. Ekpanyapong, C. Ballapuram, S. K. Lim, H.-H. S. Lee, and G. H. Loh. Microarchitectual Floorplanning Under Performance and Temperature Tradeoff. In *Proceedings of the Design, Automation and Test in Europe*, pages 1288–1293, 2006.

[12] M. Healy, M. Vittes, M. Ekpanyapong, C. Ballapuram, S. K. Lim, H.-H. S. Lee, and G. H. Loh. Multi-Objective Microarchitectural Floorplanning For 2D and 3D ICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2006.

[13] H. Jacobson, P. Bose, Z. Hu, A. Buyuktosunoglu, V. Zyuban, R. Eickemeyer, L. Eisen, J. Griswell, D. Logan, B. Sinharoy, and J. Tendler. Stretching the Limits of Clock-Gating Efficiency in Server-Class Processors. In *Proceedings of the IEEE Symposium on High-Performance Computer Architecture*, pages 238–242, 2005.

[14] R. Joseph, D. Brooks, and M. Martonosi. Control Techniques to Eliminate Voltage Emergencies in High Performance Processors. In *Proceedings of the 9th International Symposium on High-Performance Computer Architecture*, 2003.

[15] R. Joseph, Z. Hu, and M. Martonosi. Wavelet Analysis for Microprocessor Design: Experiences with Wavelet-Based dI/dt Characterization. In *Proceedings of the 10th International Symposium on High Performance Computer Architecture*, 2004.

[16] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, pages 671–680, 1983.

[17] H. Li, S. Bhunia, Y. Chen, K. Roy, and T. N. Vijaykumar. DCG: Deterministic Clock-gating for Low-power Microprocessor Design. *IEEE Transactions on VLSI Systems*, 12(3):245–254, 2004.

[18] F. Mohamood, M. B. Healy, S. K. Lim, and H.-H. S. Lee. Noise-Direct: A Technique for Power Supply Noise Aware Floorplanning Using Microarchitecture Profiling. In *Proceedings of the 12th Asia and South Pacific Design Automation Conference*, 2007.

[19] 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.

[20] M. D. Pant, P. Pant, D. S. Wills, 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*, 1999.

[21] M. D. Pant, P. Pant, D. S. Wills, and V. Tiwari. Inductive Noise Reduction at the Architectural Level. In *Proceedings of the International Conference on VLSI Design*, 2000.

[22] M. D. Powell and T. N. Vijaykumar. Pipeline Damping: a Microarchitectural Technique to Reduce Inductive Noise in Supply Voltage. In *Proceedings of the 30th International Symposium on Computer Architecture*, pages 72–83, 2003.

[23] M. D. Powell and T. N. Vijaykumar. Pipeline muffling and a priori current ramping: architectural techniques to reduce high-frequency inductive noise. In *Proceedings of the 2003 International Symposium on Low Power Electronics and Design*, pages 223–228, 2003.

[24] M. D. Powell and T. N. Vijaykumar. Exploiting resonant behavior to reduce inductive noise. In *Proceedings of the 31st annual International Symposium on Computer Architecture*, 2004.

[25] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan. Temperature-aware Microarchitecture: Modeling and Implementation. *ACM Transactions on Architecture and Code Optimization*, 1(1):94–125, 2004.

[26] Z. Tang, N. Chang, S. Lin, W. Xie, S. Nakagawa, and L. He. Ramp up/down Floating Point Unit to Reduce Inductive Noise. In *Workshop on Power-Aware Computer Systems*, 2000.

[27] S. Zhao, C. Koh, and K. Roy. Decoupling Capacitance Allocation and Its Application to Power Supply Noise Aware Floorplanning. *IEEE Transactions on Computer-Aided Design*, pages 81–92, 2002.