Noise-Direct: A Technique for Power Supply Noise Aware Floorplanning Using Microarchitecture Profiling

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

This paper proposes *Noise-Direct*, a design methodology for power integrity aware floorplanning, using microarchitectural feedback to guide module placement. Stringent power constraints have led microprocessor designers to incorporate aggressive power saving techniques such as clock-gating, that place a significant burden on the power delivery network. While the application of extensive clock-gating can effectively reduce power consumption, unfortunately, it can also induce large inductive noise (di/dt), resulting in signal integrity and reliability issues. To combat these problems, processors are usually designed for the worst-case current consumption scenario using adequate supply voltage and decoupling capacitances.

To tackle high-frequency inductive noise and potential IR drops, we propose a novel design methodology that integrates microarchitectural profiling feedback into the floorplanning process. We present two microarchitectural metrics to quantify the noise susceptibility of a module:*self weighting* and *correlation weighting*. By using these metrics in a *force-directed* floorplanning algorithm to assign power pin affinity to modules, we can quickly converge to a design for average-case current consumption. By designing for the average-case and employing dynamic di/dt control for the worst-case, we can ensure that a chip is noise-tolerant without exceeding decap budget constraints. Our observations showed that certain functional modules in a processor exhibit consistent and highly correlated switching activity, that can be used to guide module placement distance from power pins. The experimental results demonstrate that the force-directed floorplanning technique can effectively suppress supply noise experienced by modules, reduce the total number of supply-noise margin violations, and achieve a floorplan with considerably lower IR drop, as compared to a wire-length driven floorplan.

1. INTRODUCTION

Power efficiency is the first-order physical constraint in modern day processor design. The excessive power demand has led to the use of aggressive techniques such as dynamic voltage/frequency scaling, clock or power gating, etc. Although techniques like clock-gating can dramatically reduce dynamic power consumption for idle modules, they also exacerbate inductive noise (di/dt) and IR drops on the power delivery network. As a result, processor designers have to account for worst-case inductive noise, typically using an ultra-low impedance power supply network. 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 on-die decaps distributed across the die while low-frequency noise is handled by package level decap.[1] Alternatively, designers also incorporate fine-grained clock-gating domains whereby modules are clock-gated in an incremental fashion in order to minimize abrupt current surges [6, 12]. Note that both techniques are centered around the philosophy of designing the chip based on worst-case switching activity.

[1]Note that this work focuses on the high-frequency di/dt issue.

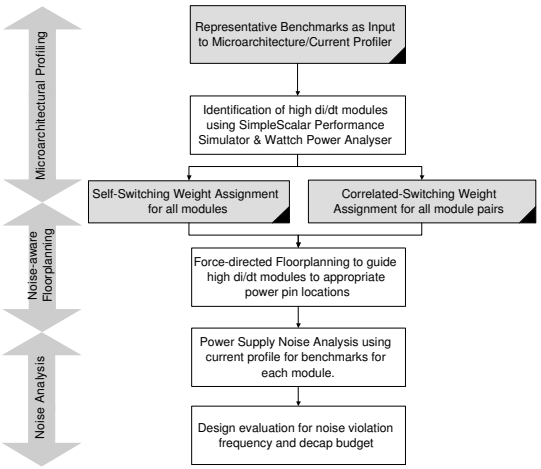

Figure 1: Noise-Direct Design Methodology Overview

With low supply voltages and high power consumption in newer generations of processors, the worst-case design strategy becomes highly inefficient. Increasing amounts of decap will consume chip area and lead to excessive leakage current. Static control for di/dt in the form of fine-grained clock gating will cause performance degradation since modules cannot be gated-on quickly. To overcome these issues and avoid designing for the worst case inductive noise, we propose a design methodology, *Noise-Direct*, that integrates microarchitectural profiling feedback into the floorplanning process. The basic idea involves the identification of correlated modules that are highly likely to cause power supply noise violations and to use such information to guide module placement. An overview of the design flow is illustrated in Figure 1. There are three phases including microarchitectural profiling, noise-aware floorplanning and power supply noise analysis. This paper makes the following contributions:

- We introduce two metrics called *self switching weight* and *correlated switching weight* for identifying modules that are highly likely to cause large di/dt.
- We present a force-directed floorplanning algorithm that incorporates microarchitectural feedback for module placement. It ensures a design for the average-case along with dynamic control at the microarchitectural level to account for the worst-case current scenario.
- To evaluate the effectiveness of our noise-aware floorplan, we apply a SPICE model of an on-chip power delivery network. Based on the model, we present the maximal voltage swing at each module and the overall noise tolerance of the chip.

Current design methodologies consider inductive noise issues in the power supply network as an afterthought. In contrast, we address this issue early in the architectural planning phase, thereby

reducing decap requirements and design complexity. By floorplanning for the average case using the techniques we propose with dynamic di/dt control schemes [15] to account for the worst case, we can ensure a design that is far more resistant to inductive noise than a purely wirelength driven floorplan.

The rest of the paper is organized as follows. Section 2 outlines the motivation. Related works are discussed in Section 3, followed by a design space analysis in Section 4. Section 5 describes Noise-Direct. Section 6 presents the evaluation methodology and Section 7 shows experimental results. Finally, Section 8 concludes.

2. PRELIMINARIES

Power delivery noise is a growing concern and presents a major issue that thwarts processor designers. One reason is due to the increasing amount of current consumption in newer chips. In addition, as devices shrink, the supply voltage is also reduced to meet gate-oxide reliability requirement. Although the lowered voltage offsets the current consumption to some extent, it also results in a lower noise margin. Increasing current consumption and switching activity, coupled with lower noise margins, means that designers have to meet stringent noise constraints by accounting for the worst-case current scenario. This is typically done by using different types of decoupling capacitors and making an extremely low impedance path from the power supply to the chip. This procedure, undoubtedly, is not very effective, in terms of cost or complexity [1].

To mitigate dynamic power, processor vendors employ (aggressive) clock-gating on their chips. Clock-gating not only reduces dynamic power and heat dissipation, but also can save leakage power due to the temperature drop. However, simultaneous gating of large modules in the chip can lead to excessive inductive noise in the power supply. Typically, this issue is dealt with the deployment of both off-chip and on-chip decaps [26, 18], which increases chip area and can result in excessive leakage current. Alternatively, certain commercial processors also employ fine-grained gating domains to prevent large modules from being gated on or off too quickly [6, 12]. Note that fine-grained gating domains increase the design complexity and lead to performance loss due to the fact that modules cannot be gated on immediately. The inefficacy of this technique lies in the fact that this design is aimed at the *infrequent worst-case* current consumption scenarios. This technique also requires complex modeling of the power delivery network for ensuring that all supply noise constraints are met. To minimize the effort of post-design optimization for power supply noise in future processors that have higher functionality and a lower noise margin, alternative design methodologies need to be sought.

3. RELATED WORK

Power supply noise aware floorplanning were studied in the past [4, 5, 14, 26]. The central idea in these arena involves two concepts: the first one involves creating a low impedance path to the chip, and the second involves optimizing on-chip decap placement and allocation to suppress inductive noise effects. At the microarchitectural level, some techniques were proposed to address the worst case inductive noise effects due to applying power saving techniques [23, 21, 22, 19, 20, 10]. These techniques typically employ certain types of dynamic control mechanisms that can estimate the incoming current surges and subsequently throttle processor activity, thereby avoiding noise margin violation.

In contrast to the prior art, we are advocating a methodology that takes inductive noise issues into account early in the architecture planning phase of a design. By analyzing 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 wire-length driven floorplan.

4. DESIGN SPACE ANALYSIS

The main focus of this work is to target a floorplan for the average-case current consumption scenario. Based on actual profiling results of dynamic module switching activity, our floorplan can be inherently more noise tolerant. Nonetheless, every design still has to guarantee the reliability by considering the worst-case scenario even though it might be rather infrequent.

The option with respect to how the worst-case scenario is addressed depends on the designers. A traditional solution could involve deployment of sufficient decoupling capacitance to minimize inductive noise. In order to reduce the increasingly growing size of decaps on a processor, Noise-Direct is aimed to reduce decap requirements by analyzing the switching correlation between microarchitecture modules and placing each module based on the average case di/dt distribution. However, in cases when the current threshold is exceeded, a dynamic di/dt control mechanism at the microarchitecture level is still needed to handle the potential noise emergency in addition to our noise tolerable floorplan. It is achieved by dynamically throttling a processor's activity [10, 15, 23] at the potential cost of performance degradation. By coupling noise-aware floorplanning with dynamic di/dt control, we can guarantee that our floorplan for the average case is more noise tolerant.[2] Compared to prior art, Noise-Direct can both reduce the total decap requirement on an overly conservative chip design and avoid performance throttling by only invoking dynamic di/dt control for the unlikely worst case scenarios.

5. NOISE-DIRECT METHODOLOGY

Our Noise-Direct design methodology consists of two primary phases: (1) microarchitectural profiling and (2) Floorplanning. The following sub-sections detail the entire procedure.

5.1 Microarchitectural Profiling

The dynamic power consumption of a processor is correlated to the characteristics of running programs. To profile current consumption and module activity, cycle-level architecture simulators such as Simplescalar can be used. Microarchitecture level power simulation [2], incorporated inside a cycle-level simulator can be easily extended to quantify current consumption for each module on a per-cycle basis. This method provides a good understanding of current demands (di) during each clock period (dt) and identifies modules that are likely culprits of inducing high di/dt noise.

Recently, researchers [10, 15] advocated incorporating dynamic di/dt control at the microarchitectural level to avoid excessive voltage ringing in the power supply. By including current calculation into microarchitectural simulations, these techniques analyzed benchmark behavior and used it to guide the dynamic di/dt control. Along a similar line, our methodology incorporates a fine-grained current and switching activity profiling by the cycle-level simulator to guide our noise-aware floorplanner. As described earlier, excessive and/or simultaneous gating of microarchitectural modules can lead to reliability issues caused by inductive noise. Our microarchitectural profiling involves quantifying switching activity of modules under ideal clock-gating. By gathering switching correlation and characterizing dynamic current demands for target applications, we can provide essential metrics that can be used in the floorplanning process to generate a noise-aware floorplan aimed for average-case current consumption and switching activity.

To identify problematic (high switching activity) modules and perform di/dt aware power pin assignments to them, we use two metrics. The first metric involves measuring module activity over the duration of a benchmark and assigning weights to modules that is proportional to the relative number of switches and the intensity of the switch. The second metric involves identifying the amount of correlation between each module in the microprocessor, in terms of simultaneous on/off gating. A detailed description of these metrics follow.

5.1.1 Self Switching Weight Assignment

Self switching measurement is used to quantify the number of gating occurrences in the processor for a benchmark during the profiling period. Both gating on and off are considered as likely events to cause di/dt fluctuation. The objective of this metric is to

[2]Note that we are not proposing any new type of dynamic di/dt control, which is outside the scope of this work. Rather, we are advocating a complementary design methodology that inherently tries to achieve a static design that is noise-tolerant for the average case current consumption. The core of Noise-Direct is in noise-tolerant floorplanning. For dynamic di/dt control in processors, interested readers can refer to [10, 15, 23].

isolate the microarchitectural modules with high switching activity. For example, certain modules such as the I-Cache are likely to be needed almost every cycle and hence will not be gated on/off very often unless the instruction fetch is stalled due to misses. Such modules will not be considered as major offenders of inductive noise. This factor is captured and collected in the switching results generated by our extended cycle-level simulator. In addition, the intensity of the gating activity also depends on the current consumption of each module. The normalized switching activity factor and the current consumption per cycle called *intensity of switch* are combined into a single weight α, represented by the following equation. If sw_i represents the raw number of switching events for module i and I_i is the intensity of the switch, then the self-switching factor α_i is denoted by:

$$\alpha_i = sw_i \bullet I_i \tag{1}$$

In essence, modules with larger weights indicate higher susceptibility to functional failure due to higher inductive noise. This heuristic is applied to the force directed floorplanning technique discussed in Section 5.2.

5.1.2 Correlated Switching Weight Assignment

In addition to the absolute self switching magnitude, di/dt issues due to clock-gating also arise from simultaneous gating of neighboring modules. If two modules that switch simultaneously have their least impedance paths to the same power pin, this will cause larger inductive noise effects at the modules. Our second metric, the correlated weight, accounts for the degree of gating correlation between microarchitectural modules. The basic idea is to use this heuristic to either place highly correlated modules away from each other, or at least assign them to different power pins. For instance, the I-Cache and the I-TLB are two units that are likely be highly correlated since they are almost always accessed simultaneously. Simultaneous gating of these modules in the same direction (both on or both off) at the same power pin is likely to induce a high di/dt in the supply voltage.

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 a correlation matrix with each location representing the number of simultaneous gating events encountered.

Since switching characteristics of modules vary from each other, the correlation factors have to be determined in a manner that ensures fairness. For instance, if module A and B switched only twice throughout the execution, and if they happened to switch simultaneously only for one single occasion, this would indicate a correlation of 50%. In contrast, if modules C and D switched 10 times throughout the execution and happened to contain only 3 simultaneous switches, this means that they are correlated only 30% of the time. Clearly, the latter case would be more susceptible to higher inductive noise. We need to consider such occurrences prudently in the correlation factor computation.

To ensure fairness, we begin with a correlation matrix that contains raw numbers of correlated switches. We then normalize each row with respect to a single module that is assigned to the row. In order to ensure fair switching weights for each row, we calculate the average of weight that is normalized to each module (in each row). The result is a symmetric correlation matrix that will contain weights that capture both correlation and ensure that they are relative to the switching of each module. An illustration of the calculation process of correlated switching events that is relative to the modules, is shown in Figure 2. 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*.

The extent of correlation is proportional to the magnitude of the above calculated weight, and the average intensity of the gating event. Equation 2 represents the correlation weight $\gamma_{i,j}$, between two modules i and j.

$$\gamma_{i,j} = \frac{1}{2}\left(\frac{X_{ij}}{sw_i} + \frac{X_{ji}}{sw_j}\right) \bullet \frac{1}{2}(I_i + I_j) \tag{2}$$

During power pin assignment, the modules with a high correlation weight will be placed farther apart from each other for alleviating the inductive noise caused by simultaneous gating. The correlation weights are also factored into the noise-aware floorplanning technique to be described next.

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

Figure 2: Correlated Switching Matrix

5.2 Floorplanning Algorithm

5.2.1 Overview of the Approach

Given a set of microarchitectural modules and a netlist that specifies the connectivity among these modules, our noise-aware microarchitectural floorplanner tries to determine the location of the modules in a chip such that (i) there is no overlap among modules, (ii) the sum of current demand for each power pin does not exceed its capacity, and (iii) power supply noise experienced by each module does not exceed the given bound. Our objective is to provide a floorplan that minimizes the area of the floorplan and total wirelength. Microarchitectural floorplanning has drawn significant interests from both the computer architecture and EDA communities recently [13, 7, 3, 9, 17, 11]. These existing works mainly target performance and thermal issues, but power supply noise issue has not been addressed.

Among several methods known for floorplan optimization, we employ the force-directed floorplanning method [8]. Compared with other methods such as Simulated Annealing [16], slicing method [24], and analytical approach [25], force-directed method does not require tedious parameter tuning and converges quickly while obtaining high quality solutions [8]. We formulate the floorplanning problem as finding a set of forces among and between fixed objects (such as I/O or power pins) and movable modules in order to optimize the objective function. The problem of finding module position then becomes one of finding forces. Our floorplanner consists of the following four steps:

1. Initialization: To begin, all modules are randomly distributed throughout the placement area, without regard to overlap.

2. Iteration: Our objective function is optimized in an iterative manner, where we update a certain set of forces based on the last iteration to guide the optimization process.

3. Stopping Criterion: The iterations are stopped when the utilization of the floorplanning area is above a threshold. This has the effect of an overlap constraint as the floorplan area is related to the sum of the area of the blocks and the utilization cannot go above a certain level without a corresponding drop in the amount of overlap.

4. Legalization: The legalization step removes the overlap among modules while maintaining the quality of the solution.

Our objective function contains the following types forces (see Figure 3 for reference): (1) net force (F_{net}): all pins in the same net are pulled closer together to minimize the wirelength objective. (2) center force (F_{cen}): all modules are pulled to the center of the chip to discourage the modules to escape the chip boundary. (3) correlation force (F_{cor}): modules with high switching activity repel each other so that the noise caused by the modules is reduced. The correlation factors $\gamma_{i,j}$ described in Section 5.1.2 are used to compute the magnitude. (4) density force (F_{den}): modules located in a high density region of the chip are pushed apart to reduce the overlap. (5) pin capacity force (F_{pin}): modules are pulled into or pushed out of each power pin so that the total demand on each power pin is evenly distributed and its capacity is not violated. The first three types are non-iterative, where as the last two are iterative. We fix all the non-iterative forces during the floorplan optimization process, whereas the iterative forces are updated based on the previous

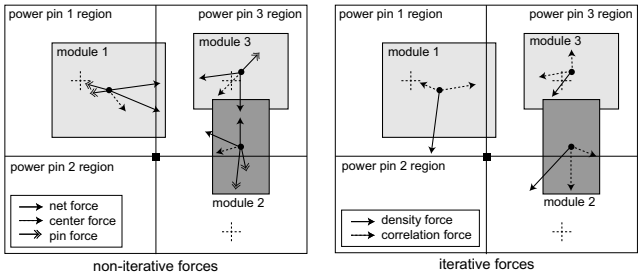

Figure 3: Illustration of various forces optimized in our floorplanner

iterations. In order to balance the impact of the five types of forces, we optimize the following combined force:

$$F_{tot} = \lambda \cdot F_{net} + \theta \cdot F_{cen} + \mu \cdot F_{cor} + K \cdot F_{den} + \rho \cdot F_{pin}$$

where λ, θ, μ, K, and ρ are weighting constants.[3]

5.2.2 Force Equations

Let n be the number of free modules in the floorplan and (x_i, y_i) be the x and y-coordinates of the center of module i, respectively. A placement can be described by the $2n$-dimensional vector $\vec{p} = (x_1, \ldots, x_n, y_1, \ldots, y_n)^T$. The cost of a connection is then formulated such that it is proportional to the squared Euclidean distance between its endpoints. The objective function sums the cost of all connections and therefore can be written in matrix notation as

$$\frac{1}{2}\vec{p}^T C\vec{p} + \vec{d}^T\vec{p} + const \tag{3}$$

where the $2n \times 2n$ symmetric matrix C and the vector $\vec{d}$ are produced from the module connections and their weights and the formula for squared Euclidean distance. For example, the x-part of the connection between two free modules i and j is $(x_i - x_j)^2 = x_i^2 - 2x_ix_j + x_j^2$. The first term adds to $C_{i,i}$, the second term to $C_{i,j}$ and $C_{j,i}$, and the third term to $C_{j,j}$. Similarly for a fixed connection between free module i and fixed location f, $(x_i - x_f)^2 = x_i^2 - 2x_ix_f + x_f^2$ adds the first term to $C_{i,i}$, the second term to $\vec{d}_i$, and the third term to the constant part of Equation (3). This cost function is minimized by solving the linear equation system

$$C\vec{p} + \vec{d} = 0 \tag{4}$$

This formulation is equivalent to modeling connections as springs and calculating the state of equilibrium.

Force-directed floorplanning and placement algorithms are well known for their overlap problems. Spreading or repulsive forces are required to make the final solution feasible, i.e. with zero overlap. These additional forces extend Equation (4) with the force vector $\vec{e}$ to model constant additional forces which are iteratively updated:

$$C\vec{p} + \vec{d} + \vec{e} = 0 \tag{5}$$

The complexity of solving this equation is $O(k \cdot n^2)$, where k is the number of iterations, and n is the number of modules. Our experiments show that k ranges from 1 to 10 and n is around 20. Thus, our algorithm generates optimized solutions quickly.

We compute the pin capacity force as follows: The "current drawing region" of a pin is defined as a rectangle centered on that pin with width and height equal to the distance between pins. Then, the pin capacity force is formulated as follows. Let c_i be the power consumption of module i located within the current drawing region of power pin j, I_j be the capacity of power pin j, (x_i, y_i) be the center of module i, (x_j, y_j) be the location of pin j, and $d_{i,j}$ be the squared Euclidean distance between module i and pin j. Let α_i be the self switching weight of module i defined in Section 5.1.1. The x direction force between free module i and fixed pin j is then

$$F_{pin}^x(i,j) = \left[\left(\frac{I_j}{\sum_i c_i}\right)^2 - 1\right] \cdot \frac{|x_i - x_j|}{d_{i,j}} \cdot \alpha_i \tag{6}$$

A similar definition follows for the force along the y direction. This force is proportional to the distance between the module and the pin, negative if the sum of the current being drawn from the modules in the current drawing region of the pin are greater than the capacity of the pin and positive otherwise, and in the range $(-1, \infty)$. Basically if the demand of block i is higher than the capacity of the pin j, then the force pushes the modules away; otherwise, it pulls the modules towards the pin.

5.2.3 Updating Iterative Forces

As mentioned previously, we update two kinds of forces during each iteration: F_{den} and F_{pin}. Specifically, we first obtain the location of the modules from the previous iteration and use them to recompute the density of each region in the floorplan and attractive or repulsive forces among the modules within a vicinity of each power pin. The main motivation for this force update is to satisfy the non-overlap constraint (via updating F_{den}) and pin capacity constraint (via updating F_{pin}). In case these constraints are not met in the current solution, we try to minimize the amount of violation as much as possible by attempting another iteration. We note that the pin constraint is easily satisfied, but not the overlap constraint. Thus, our post-process explicitly removes the overlap among the modules. Since $\vec{e}$ consists of F_{den} and F_{pin}, $\vec{e}$ gets updated and solved in each iteration.

5.2.4 Legalization

A simple heuristic is used to legalize the floorplan of the modules. Vertical and horizontal constraint graphs similar to those used for the [16] are created based on the floorplan solution. The basic idea is to derive the relative positions among the modules based on the force-directed floorplanning, and use Sequence Pair [16] to encode them to remove overlap. For each pair of modules, the horizontal and vertical distance between their centers is compared. If the horizontal distance is smaller than the vertical distance then the appropriate constraint is added to the vertical constraint graph. Conversely, if the vertical distance is less, the appropriate constraint is added to the horizontal constraint graph. If the modules overlap, then these constraints will push the modules apart in the direction that minimizes overall movement. Thus, the legalized modules remain close to their original locations. The constraint graphs ensure that the final floorplan is non-slicing and non-overlapping.

6. POWER NETWORK ANALYSIS

To evaluate the effectiveness of the two heuristics that were used to guide noise-aware floorplanning, we use a SPICE model of the on-chip power delivery network. We evaluate the benefits of our technique under 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, we can determine the period where the highest module switching occurs. 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 input in the SPICE module as piece-wise linear source (PWL) input. By incorporating per-cycle current consumption profile obtained from our microarchitecture simulation, we are able to observe induced noise effects as a direct function of the application's behavior.

Based on the power supply noise of each module, we also calculate the amount of decap required for the floorplan [26]. If V_{noise} is the noise of a given module, V_{limit} is the noise margin, Q is the amount of charge drawn by the module, then the amount of decap required (C), can be estimated according to the following:

$$\theta = max(1, \frac{V_{noise}}{V_{limit}}) \tag{7}$$

[3]Our empirical choice of these values is to set them all equal. We fix these weights constant during the entire floorplan optimization process. One can tune the weights statically or dynamically to emphasize desired objectives.

$$C = (1 - \theta^{-1})\frac{Q}{V_{limit}} \quad (8)$$

7. QUANTITATIVE ANALYSIS

We used SimpleScalar 3.0 and Wattch for microarchitectural profiling and simulation. We incorporated extensions to generate both self and correlated module switch weights to be used in our floorplanner. The power and current consumptions were based on a 5 GHz processor with 70nm process. Nine integer programs from the SPEC2000 benchmark were used in this study. Each simulation was fast-forwarded by 4 billion instructions and simulated for 100 million instructions.

	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*

Figure 4: Self and Correlated Switching Weights of All Modules

7.1 Self and Correlated Switching Weights

Figure 4 shows the both the average self switching weight and correlated weight of all modules in a symmetric matrix table. The forward diagonal in the matrix represents the self-switching weights of each module and all the remaining locations represent the correlated switching weights.[4] The rows and columns are sorted in the descending order of self-switching weights from left to right. A higher self-switching weight indicates higher susceptibility to di/dt problems. As shown, the load/store queue (LSQ) and register update unit (RUU) carry more weights in comparison to other modules. On the other hand, the weights of the modules that are likely to be accessed every cycle (turned on mostly) such as the L1 I-Cache and the I-TLB are lower. Some modules that are dormant, only accessed once in a long while e.g. FPU register file, also have lower weights.[5]

The correlation weights are used to place modules that switch simultaneously, away from each other in the floorplan. As expected, branch predictor and BTB, I-Cache and I-TLB and D-Cache and D-TLB are all highly correlated modules. In addition, it is also observed that the first six ALU modules are also highly correlated for concurrency exists in integer instructions. These modules will be directed away from each other to lessen the inductive noise by removing clustering of modules.

7.2 Power Supply Noise Analysis

The noise-aware floorplan algorithm used both microarchitectural metrics to guide module placement. In order to demonstrate the noise-tolerance of the force-directed floorplan, we compare our noise-aware floorplan to a baseline floorplan that minimizes total wirelength. In our noise analysis, we assumed a V_{dd} of 1 volt (for 70nm), and a maximum allowed noise margin of 10%. To illustrate the noise analysis in more details, we depict the worse-case noise for each module using gzip, a compute-bound program. They are shown in Figure 5. Note that this graph is sorted from left to right in the decreasing order of module self-switching activity. As shown, the noise-aware floorplan significantly suppresses the noise experienced by modules with high switching activity as well high current consumption. Almost all the ALUs that exhibit a fair amount of switching activity and extremely high correlation with each other, show significant voltage noise reductions. For the integer register file (iregfile), the voltage noise was reduced by 81.7%

[4]Note that this matrix shows both self as well as correlated switches which is why the diagonal is non-zero.

[5]Although we profiled only SPECint2000, there are certain benchmarks (e.g. data compression) that use floats and doubles.

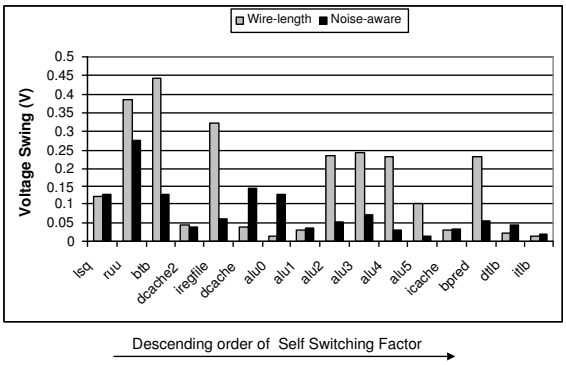

Figure 5: Power Supply Noise at Modules for gzip

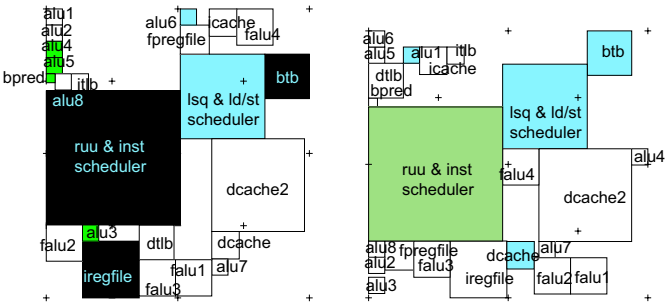

Figure 6: Noise Tolerance for gzip. (Darker module has higher noise)

in the noise-aware floorplan. We do observe that the L1 Data Cache (dcache) and ALU0 have a higher noise violation in the noise-aware floorplan as compared to the baseline, although it has a high self-switching factor. This is due to the fact that other units, especially the remaining ALUs that will have a higher priority when it comes to being directed towards power pins because of their strong correlation. The L1 D-Cache does not exhibit a high correlation with other units and will hence is less important than other modules that have a higher potential of noise margin violations. Nonetheless, it is also be noted that the increased violations in the noise-aware floorplan are only slightly above the allowed 10% margin, making the overall solution much more noise tolerant.

7.3 Floorplan and Decap Requirement

We now present the baseline wire-length driven floorplan and the noise-aware one in Figure 6. The color code in each module represents the degree of noise tolerance. The cross (+) in the figure represents the location of the power pins. The area of the wire-length driven floorplan is 69.35 mm^2 with a total wirelength of 804.86

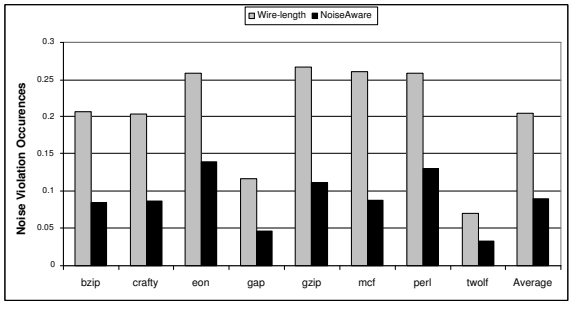

Figure 7: Noise Margin Violation

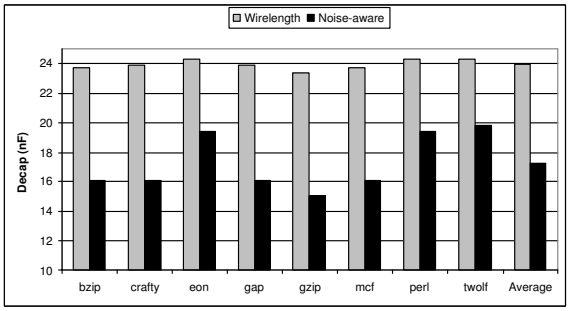

Figure 8: Decap Budget for SPEC2000 Integer benchmarks

mm while the noise-aware floorplan has an area of 67.97 mm^2 with a total wirelength of 825.87 mm. We classify modules in the floorplan based on the worst-case noise they experience, with the darkest modules representing modules with the highest noise. The noise tolerance for the modules are shown for the same benchmarks used in Figure 5. It is observed that the RUU has improved noise tolerance and most of the ALUs are completely below the noise margins for both benchmarks. It is to be noted even though we induce a slightly higher number of violations for certain modules in the noise-aware floorplan, the overall noise tolerance of the chip, in fact, is greatly improved. To demonstrate the noise tolerance over the whole chip using our floorplan, we present the number of noise margin violations in Figure 7. This figure compares the frequency of voltage swings over the 10% noise margin accumulated for all modules. The reduction of noise violations ranges from 46.3% for eon to 66.7% for mcf with an average of 56.3%. The results also indicate the alleviation of applying dynamic di/dt control, which comes at the cost of performance degradation due to throttling. By reducing the total number of violations, the need for dynamic di/dt control will be only needed in the infrequent worst cases.

As described earlier, suppressing power supply noise has two implications. First, it can reduce the amount of decap required, and second, it can bring down design complexity for certain modules that require fine-grain clock gating. The choice to address the worst-case current consumption is dependent on the designer, whereby noise can be addressed by decap alone, or by the incorporation of dynamic di/dt control. To show the advantages in reducing the decap needed on a chip using our noise-aware floorplan, we also present the decap budget requirement based on Eq(7) and Eq(8). Figure 8 compares our noise-aware floorplan against the baseline. It shows that the decap budget is significantly reduced in the noise-aware floorplan for all the benchmarks we evaluated. In the best case, an improvement of 33% reduction in decap budget are observed for bzip, crafty and gap with an average reduction of 28%. Please note that we use this analysis to merely illustrate the potential decap budget improvement that can be obtained using our methodology. Decap allocation or placement is beyond the scope and focus of this work.

Finally, we also examined the impact of performance with our noise-aware floorplan. The inter-module latencies as a result of our floorplan remain unchanged from the baseline. It is also to be noted that pushing apart certain modules that switch together like Caches and TLBs is not unrealistic. Therefore, the IPC will not alter. In fact, as we mentioned earlier, our floorplan will substantially decrease the invocations of architectural throttling for dynamic di/dt control, thereby reducing the likelihood of slowing down the processor. We did not quantify this benefit of our technique in this paper due to the page limitation.

8. CONCLUSIONS

As processor designers aggressively battle between performance vs. power efficiency trade-offs, power delivery and supply noise considerations will play a larger role in the design process from reliability standpoint. In addition, with smaller devices and lower supply voltage, processors will become less tolerant to inductive noise induced by abrupt current fluctuation (di/dt). To address this issue in the early design phase in a cost effective manner, we propose a design methodology called Noise-Direct that incorporates microarchitectural feedback to guide module placement in the floorplanning process.

As noise margins in a processor reduce and the worst-case design becomes inefficient on the cost budget, Noise-Direct provides a systematic approach to significantly reduce supply noise effects by incorporating metrics derived from microarchitectural feedback into the floorplanning process. As the results show, our technique can effectively lower the magnitude of runtime inductive noise as well as alleviate the decap requirement.

9. REFERENCES

[1] K. Aygun, M. J. Hill, K. Eilert, K. Radhakrishnan, and A. Levin. Power Delivery for High-Performance Microprocessors. *Intel Technology Journal*, 9(4), 2005.

[2] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proc. Int'l Symp. on Computer Architecture*, 2000.

[3] M. Casu and L. Macchiarulo. Floorplanning for throughput. In *Proc. Int. Symp. on Physical Design*, 2004.

[4] H. Chen, L. Huang, I. Liu, M. Lai, and D. Wong. Floorplanning with power supply noise avoidance. In *Proc. Asia and South Pacific Design Automation Conf.*, 2003.

[5] 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 Trans. on VLSI Systems*, pages 75–85, 2005.

[6] J. Clabes, J. Friedrich, M. Sweet, J. DiLullo, S. Chu, D. Plass, J. Dawson, P. Muench, L. Powell, M. Floyd, B. Sinharoy, M. Lee, M. Goulet, J. Wagoner, N. Schwartz, S. Runyon, G. Gorman, P. Restle, R. Kalla, J. McGill, and S. Dodson. Design and implementation of the power5 microprocessor. In *Proceedings of the 41st Design Automation Conference*, 2004.

[7] J. Cong, A. Jagannathan, G. Reinman, and M. Romesis. Microarchitecture evaluation with physical planning. In *Proc. ACM Design Automation Conf.*, 2003.

[8] H. Eisenmann and F. M. Johannes. Generic global placement and floorplanning. In *Proc. ACM Design Automation Conf.*, pages 269–274, 1998.

[9] M. Ekpanyapong, J. Minz, T. Watewai, H.-H. S. Lee, and S. K. Lim. Profile-guided microarchitectural floorplanning for deep submicron processor design. In *Proc. ACM Design Automation Conf.*, 2004.

[10] E. Grochowski, D. Ayers, and V. Tiwar. Microarchitectural simulation and control of di/dt-induced power supply voltage variation. In *Proc. IEEE Int. Symp. on High-Performance Computer Architecture*, 2002.

[11] M. Healy, M. Vittes, M. Ekpanyapong, C. Ballapuram, S. K. Lim, H.-H. S. Lee, and G. H. Loh. Microarchitectural Floorplanning Under Performance and Temperature Tradeoff. In *Proc. Design, Automation and Test in Europe*, 2006.

[12] 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 11th International Symposium on High-Performance Computer Architecture*, 2005.

[13] C. Long, L. Simonson, W. Liao, and L. He. Floorplanning optimization with trajectory piecewise-linear model for pipelined interconnects. In *Proc. ACM Design Automation Conf.*, 2004.

[14] J. Minz, S. K. Lim, and C. K. Koh. 3D Module Placement for Congestion and Power Noise Reduction. In *Proc. Great Lakes Symposum on VLSI*, 2005.

[15] F. Mohamood, M. B. Healy, S. K. Lim, and H.-H. S. Lee. A Floorplan-Aware Dynamic Inductive Noise Controller for Reliable Processor Design. In *Proceedings of the 39th International Symposium on Microarchitecture*, 2006.

[16] H. Murata, K. Fujiyoshi, and M. Kaneko. VLSI/PCB placement with obstacles based on sequence pair. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pages 60–68, 1998.

[17] V. Nookala, Y. Chen, D. Lilja, and S. Sapatnekar. Microarchitecture-Aware Floorplanning Using a Statistical Design of Experiments Approach. In *Proc. ACM Design Automation Conf.*, 2005.

[18] M. D. Pant, P. Pant, and D. S. Wills. On-chip decoupling capacitor optimization using architectural level prediction. *IEEE Trans. Very Large Scale Integr. Syst.*, 10(3):319–326, 2002.

[19] M. D. Pant, P. Pant, D. S. Wills, and V. Tiwari. An architectural solution for the inductive noise problem due to clock-gating. In *Proc. Int. Symp. on Low Power Electronics and Design*, 1999.

[20] M. D. Pant, P. Pant, D. S. Wills, and V. Tiwari. Inductive noise reduction at the architectural level. In *Proceedings of the 13th International Conference on VLSI Design*, 2000.

[21] 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*, 2003.

[22] 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 Int'l Symp. on Low Power Electronics and Design*, 2003.

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

[24] L. Stockmeyer. Optimal orientation of cells in slicing floorplan designs. *Information and Control*, pages 91–101, 1983.

[25] S. Sutanthavibul, E. Shragowitz, and J. Rosen. An analytical approach to floorplan design and optimization. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pages 761–769, 1991.

[26] S. Zhao, C. Koh, and K. Roy. Decoupling capacitance allocation and its application to power supply noise aware floorplanning. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pages 81–92, 2002.