

Inductive Noise Reduction at the Architectural Level

Mondira Deb Pant[†], Pankaj Pant[†], Donald Scott Wills[†], Vivek Tiwari[‡]
[†]Georgia Institute of Technology, [‡]Intel Corporation

Abstract

A methodology for reducing ground bounce in typical microprocessors and image processing architectures has been described. As we approach Gigascale Integration, chip power consumption is becoming a critical system parameter. Deactivating idle units provides needed reductions in power consumption. However, it introduces inductive noise that can limit voltage scaling. The paper introduces an architectural approach for reducing this inductive noise by providing gradual activation and deactivation of functional blocks. This technique provides a 2x reduction in ground bounce current on a 16 bit ALU simulated in SPICE, while reducing simulated SPEC95 performance by less than 5% on a typical superscalar architecture. It has also been demonstrated to be effective for image processing SIMD architectures.

1. Introduction

Advances in silicon CMOS technologies have resulted in increases in both circuit speed and wiring density, along with a reduction in power supply voltages. These advances have intrinsically produced fast transition rates, increased current density, and reduced noise margins. Such trends lead to data integrity and performance degradation issues, primarily due to simultaneous switching noise generated by core logic switching, I/O circuitry switching, or both. The need to control these induced noises has become a major design effort for both chip and package designers. It is as becoming increasingly necessary to develop methodologies to attenuate the problem in the early phases of a design cycle itself.

Power breakdown in a high performance CPU usually reveals the clock to be the largest power consuming component [12]. This includes the clock generator, the clock drivers, the clock distribution tree, the latches and the clock loading due to all clocked elements. Reducing the switched capacitance on the clocks would as such help to reduce the total power. An increasingly popular technique to achieve this is clock-gating [10], which allows only those portions

of the clocked network to toggle that are needed on each cycle. While it can be integrated rather easily into existing clock networks, it cannot be used indiscriminately. This is because it can introduce signal integrity problems.

Ground bounce [7] [1] [11] [5], also known as simultaneous switching noise or delta-I noise, is a voltage glitch induced at power/ground distribution connections due to switching currents passing through the wire/substrate inductance (L) associated with power or ground rails. These voltage glitches which are proportional to $L\partial I/\partial t$, have not caused any serious problems in the past, since the magnitude of ground bounce was very small compared to the noise margin of CMOS circuits operating with a relatively small number of gates, at low frequencies and with narrow data buses. However, as the frequency and number of gates increase, and data and address buses become wider, more signals switch simultaneously, resulting in large currents that charge/discharge the power/ground buses in a short time leading to ground bounce. Also, with a drop in supply voltages, the noise margin of circuits is also dropping. If the magnitude of the voltage surge/droop due to ground bounce is greater than the noise margin of a circuit, the circuit may erroneously latch the wrong value or switch at the wrong time.

In this paper we will look at the noise integrity issues involved with the use of resource activation/deactivation. We introduce a design methodology that addresses design techniques at the architectural level to control and minimize sources of induced $\partial I/\partial t$ noise for not only a modern day complex micro-processor but also for a simple image-processing SIMD architecture, since this problem is gaining importance in the entire spectrum of computer architectures. SPICE simulation of a test ALU suggest a 2x reduction in ground bounce with a cost of less than 5% in performance. Preliminary results have been presented in [8].

2. Related Work

Clock-gating methods to reduce power consumption of circuitry, essentially involve ANDing the clock to a resource with a control signal which shuts down the resource when not in use. Clock-gating causes huge amount of data

transitions to occur at the powering up/down stages and hence enhanced ground bounce. Both data dependent circuit switching and the clock switching itself contribute to ground bounce within a chip. However, in well designed circuits, the $\partial I/\partial t$ impact due to data dependent circuit switching is found to be much more significant than that of the clock itself.

In [12] the authors refer to the need to have a power delivery system that can withstand greater inductive noise as voltages drop and frequencies increase, thereby increasing the CPU and system cost. Fig. 1 shows real power vs. time curves for a Pentium® Processor (from [12]). The plots indicate a greater difference in the peak and idle power drawn by the processor when the power down mode is enabled. Note that this figure shows current over a large time window. The variation is due to the processor going to a low power state during which most of the internal clocks are turned off (i.e. system power management), and not just due to on-chip clock-gating. This is true for highly parallel SIMD architectures where a large number of processors could be switched on or off in a cycle, depending on the data being processed. Moreover, we can be sure that as the systems become more complex and clock frequencies cross the 1GHz mark, turning various units on and off will have an equivalent effect on the currents drawn from the power lines.

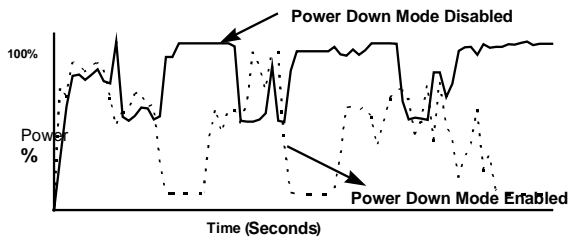


Figure 1. Power vs. time plot for a Pentium Processor [12]

3. Proposed Scheme

In the following, we provide a high-level description of an improved resource activation/deactivation scheme that reduces the instantaneous current drawn when the resources are switched on and off. Here a “resource” could imply either a functional unit within a processor (eg. ALU) or the entire processor itself (eg. a node in a SIMD array), depending on the level of abstraction that we are dealing with.

Currently, in modern micro-processors, clock-gating is implemented such that the processor pipeline is unaware of the “state” of a resource. Whether or not the clock feeding

a particular unit is turned off, it is assumed that the unit will always be available for execution provided that it is not busy with a previous instruction. In this paper, we propose a variation of the clock-gating scheme that is in essence visible to the resource scheduling unit of the processor.

The basic idea is to divide the resource into smaller parts and switch each part on (off) at separate times. This causes them to draw a lesser amount of instantaneous current and hence reduces the glitch introduced in the power and ground lines. However, since it may now take a finite amount of time for a clock-gated resource to become available, additional stalls might occur in the execution pipeline. This will have an impact on the performance of the processor, specifically the instructions-per-cycle (IPC) executed. We will show that this performance loss is not significant and the reliability levels attained by the proposed scheme will justify this small drop in performance. Next we show the validity of this scheme by applying it on a typical superscalar processor and an extension of it to a typical SIMD architecture.

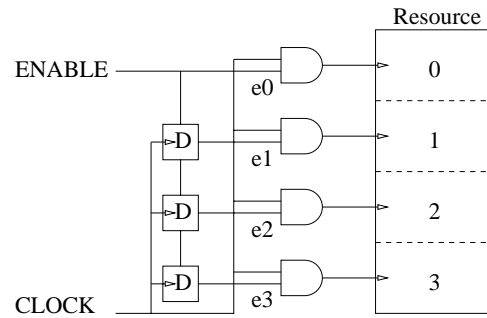


Figure 2. Circuit Implementation of the modified clock-gating scheme

4. Application I: Superscalar processor

Fig. 2 shows a possible circuit implementation of the modified clock-gating scheme for a functional unit. The clocked unit is divided into a variable number of slices, n ($n = 4$ in Fig. 2). During the normal operation mode of the functional unit each slice gets the same clock signal. When the unit needs to be turned off, the enable signal goes low. During successive cycles each of the slices 1, 2, 3, ..., n , gets clock-gated, one per cycle, as the change in the enable signal travels down the chain of flip-flops. Therefore, n cycles after the initial clock-gating signal is issued, the entire unit finally shuts down, instead of an instantaneous power down. Similarly, when the unit needs to be powered up again, the enable signal reaches the i^{th} slice after i cycles. Hence, the complete unit is powered-up in n cycles with each slice receiving its power-up signal in a staggered

manner. While introducing a small amount of delay into the data-path, this implementation has a potential of drastically reducing the $\partial I/\partial t$ problem encountered when large units are powered up or down. We have made no assumptions about the division of the resource into slices. The split might be based on functional considerations (eg. a 16-bit ALU might be split into 4 slices with 4 bits in each slice) or physical layout considerations. This flexibility is possible since the functionality of the resource is not affected by splitting the clock signal lines.

Next we look at the impact of such a scheme on the performance of a processor. Note that if it is necessary to reduce the clock-gating delay to less than n cycles, one may clock the delay flip-flops at a rate faster than the circuit clock. As such, n phases after the initial clock-gating signal is generated, the entire unit gets powered down.

4.1. Performance measurement

To measure the impact on performance of a superscalar processor due to the proposed scheme, we used the SimpleScalar tool set [2] developed at the University of Wisconsin-Madison. This tool set provides a fast, flexible and accurate simulation of a processor that implements the SimpleScalar architecture (a close derivative of the MIPS architecture [9]). The advantage of using this tool is that the standard benchmark binaries (SPEC95, etc.) can be compiled for the SimpleScalar instruction set and evaluated against any specified architecture.

To conduct our experiments, modifications had to be made to the SimpleScalar simulator, since the original version did not provide for explicit clock-gating of the resources. In fact the simulator did not need to understand the concept of clock-gating, since clock-gating does not effect the high-level processor operation in any way.

Under the original mode of operation, the simulator assigns one of three states *sleep/dormant*, *busy* or *ready* to every resource depending of whether the resource is dormant, being used or ready for use. The notion of a resource being dormant can be interpreted as a way of denoting that the clock to the resource is turned off. If there is a request for a dormant resource from the resource scheduler, then it is assumed to be made available immediately by turning on its clock. Fig. 3(a) shows a typical timing diagram of a resource. Here the resource is assumed to stay in the ready state for a prespecified number of cycles, after which it goes to sleep.

For our proposed clock-gating scheme, two additional resource states, ‘waking up’ and ‘going to sleep’ have been incorporated into the simulator. This is illustrated in Fig. 3(b). If a resource is requested when it is dormant, instead of going from sleep to busy directly, the resource goes through an intermediate waking state during which time it

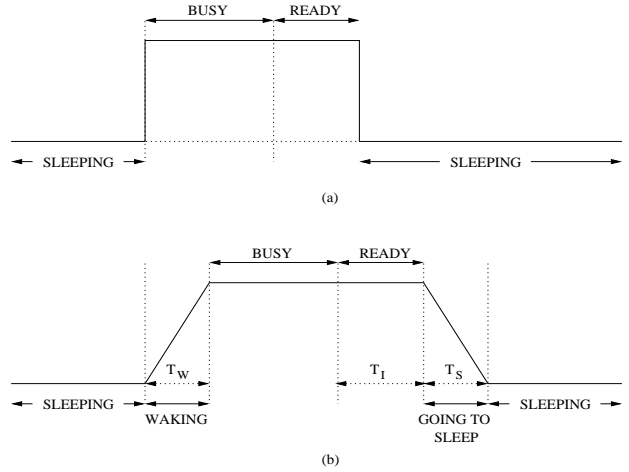


Figure 3. Timing diagram for operator states

cannot be used. The waking time, T_w , denotes the number of clock cycles taken by a resource to wake up. Similarly, T_s denotes the time (in cycles) a resource takes to go to sleep. ($T_w = T_s = 4$ in Fig. 2) A resource goes to *sleep* after it has been *ready* for a predetermined number of cycles referred to as the idle time, T_l . The idle time of the resource plays an important role in the modified clock-gating operation. Setting the idle period to zero causes the resource to get clock-gated as soon as its operation is over, while setting it to infinity ensures that it never goes to sleep i.e. there is no clock-gating. Varying the values of T_w and T_s enables us to measure the performance trade-off the proposed scheme presents.

In the event that a resource is requested when it is ‘going to sleep’, to reduce the implementation complexity, we decided to let it go to the *sleep* state before waking it up again as illustrated in Fig. 4(a). This need not be a limitation for actual implementations and the resource could be woken up immediately on request even when it is ‘going to sleep’ as in Fig. 4(b). In the second case, the impact on the IPC will be definitely lesser. The drop in IPC reported by our experiments might therefore be pessimistic because of this assumption.

In a pipeline architecture, the finite time required to access a dormant resource may be interpreted as a “bubble” or a stall [6] being introduced into the pipeline. The resulting performance cost depends upon the number of cycles we allow for a dormant resource to become available again upon request. After compiling the SpecInt95 and SpecFP95 benchmarks for the SimpleScalar architecture, we ran the benchmarks on a prototype modern day processor to study to see the impact of our proposed scheme on performance. The architecture selected for the study is that of a typical superscalar microprocessor with out-of-order execution and is specified in Table 1. The instructions-per-cycle (IPC) for

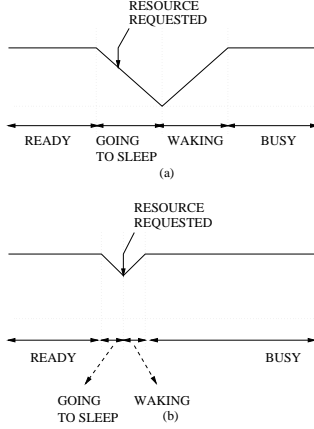


Figure 4. Request for a resource that is “going to sleep”

the SPEC95 benchmark was used as a measure of the performance of the processor for the studies. For each benchmark, the IPC was measured for varying values of the waking time (T_W) of the resources. (The time for a resource to go to sleep T_S was assumed to be the same as T_W .) The idle time (T_I) was set to 10 cycles for all the experiments. Fig. 5 shows the variation of the IPC for various SpecInt95 and SpecFP95 benchmarks plotted against T_W .

Cache				
Type	Size	Assoc.	Hit lat.	Policy
L1 data	16KB	4 way	1 cycle	LRU
L1 instr	16KB	1 way	1 cycle	LRU
L2 unified	256KB	4 way	6 cycles	LRU
Translation Lookaside Buffer				
Type	Entries	Assoc.	Page size	Miss lat.
data TLB	128	4 way	4KB	30 cycles
instr TLB	64	4 way	4KB	30 cycles
Resources				
Type	Number	Operation lat.		
i-alu	4	1 cycle		
i-mult/div	1	3/20 cycles		
fp-alu	4	2 cycles		
fp-mult/div	1	4/24 cycles		
Fetch/decode/issue width = 4				
RUU size = 16; Load/Store Queue size = 8				
Mem bus width = 16; Main mem seek time = 18 cycles				

Table 1. Typical microprocessor architecture used for performance study

Increasing the value of T_W and T_S obviously reduces the IPC. However, we observe that the drop is not more than 5% of the original IPC when the value of T_W is less than or equal to 3 cycles, clearly indicating that the loss of performance is very minimal for small values of the clock-gating delay. Increasing the number of available resources caused the drop in IPC observed by the proposed scheme to de-

crease indicating that greater number of available resources further reduces the impact on performance.

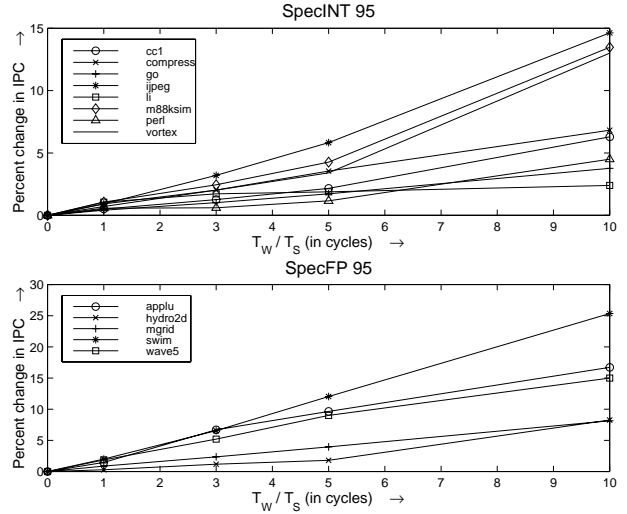


Figure 5. Plots showing varying IPC for SPECINT95 Benchmarks

The small loss in performance may be eliminated almost totally if it becomes possible to know a few cycles in advance whether a resource is required or not, for example in the decode stage. This would enable a request for the resource to be placed earlier itself, firing up a sleeping resource and making it available for use well in time. The drawback, however is that the decode stage of the pipeline would increase in complexity.

4.2. SPICE Verification

To get an estimate of the extent of improvement in $\partial I / \partial t$, the above discussed scheme was implemented in HSPICE on a 16-bit ALU split up into 2 slices of 8 bits each, and 4 slices of 4 bits each for the purpose of clock-gating. The system clock and the enable signal for the circuit are shown in Fig. 6(a) and (b) respectively. The enable signal indicates that the system goes through a power down phase followed by a subsequent power up stage. Fig. 6(c) shows a plot of current drawn across the V_{dd} line versus time for the original single clock circuit ($n = 1$). Fig. 6(d) and (e) give the current drawn across the V_{dd} line for the circuit configured to be clock-gated in a period of two ($n = 2$) and four ($n = 4$) clock cycles respectively, as discussed in Section 4. From the plots it becomes clear that while the current consumption gets progressively more spread out in the proposed scheme, the current spikes observed drop as the number of slices, n , increases. The peak current provides an indication of the rate of change of the current and hence

the extent of the voltage drop across the V_{dd} line. We can see an improvement by a factor of about 2x when 4 slices of 4 bits are used. The absolute change in the peak currents drawn will definitely improve when this approach is applied to larger units. This clearly indicates the feasibility of the proposed scheme, in a world where noise is growing to be a major concern.

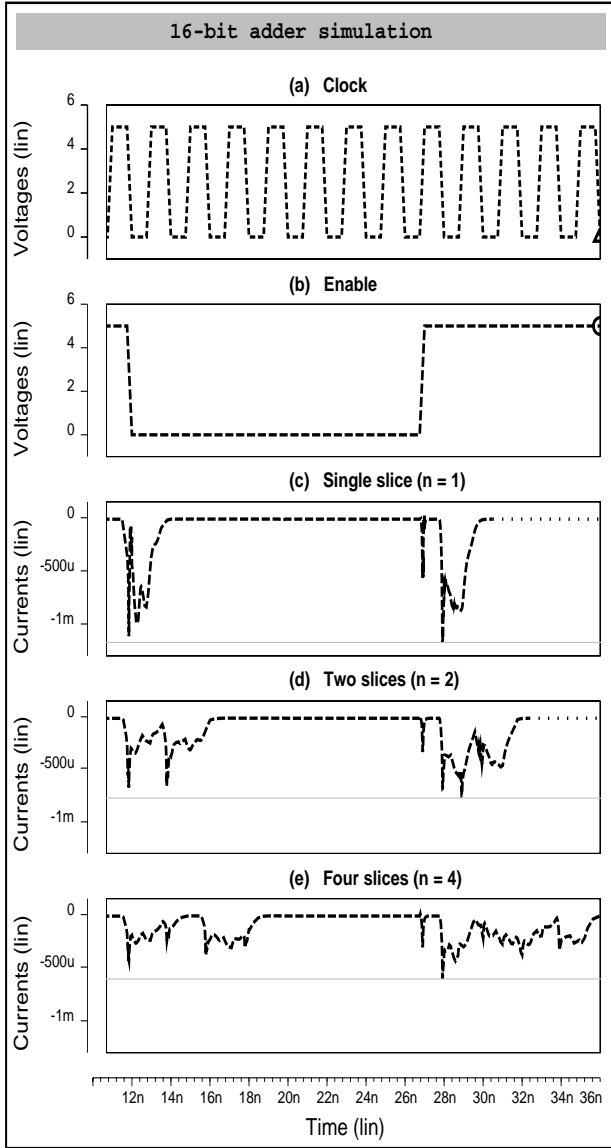


Figure 6. HSPICE simulation of an 16-bit ALU

5. Application II: SIMD Processor

In this section we study the application of the proposed scheme to SIMD architectures. As mentioned earlier, in

such arrays, hundreds of processors, (also called Processing Elements or PEs) may be simultaneously switched on and off. This may lead to huge amounts of instantaneous currents being drawn from the power supply. Note that the switching on/off of PEs in the array need not be for the purpose of power reduction. In most cases such arrays are used for specialized DSP operations (such as image processing) and preventing certain PEs from executing the broadcast instructions may be imperative to the correctness of the algorithm. For our experiments we will use the SIMD Pixel Processor, (SIMPi16) [4] which is an image processing focal plane architecture consisting of a mesh of 4096 PEs arranged in a 64x64 square grid.

In the SIMPi16 architecture, a single instruction is broadcast to all the PEs. Only the “active” processors actually execute the instruction. In order to activate/deactivate PEs, the instruction set includes the commands SLEEP[condition] and WAKEUP[condition]. As with other instructions, these are also broadcast to all the PEs and those which meet the condition respond by turning themselves on or off. The conditions usually involve the data that is residing in the PE’s local registers.

In an active PE, various functional units transition between idle and active states depending upon instructions being executed. The resulting $\partial I/\partial t$ surges are localized within the PE itself and can be attenuated by the method described in the previous section. What is of greater concern here is global surges within the mesh of PEs which will occur when large number of PEs are turned off/on. Since these architectures target portable high speed image computational devices wherein the noise margins tend to be smaller and speeds higher, these surges degrade signal integrity considerably.

What follows is a description of how this problem may be resolved in image processing architectures.

5.1. Implementation

Similar to the situation involving clock-gating a unit, we propose to activate/deactivate the PEs in a staggered manner. Already incorporated in SIMPi16 are dedicated activation/deactivation cycles for the PEs. This feature may be exploited to totally nullify performance impact that may occur due to our proposed scheme. In this vein, it may be noted that for superscalar processors we had to introduce exclusive time intervals for activation/deactivation (Section 4), thereby incurring a likely reduction in performance.

We have no prior knowledge of the spatial distribution of the PEs that may get activated at a given time. Hence, we cannot easily divide them into equal lots for activation in steps. However, we can take advantage of the regularity of the processor grid and ensure that two adjacent PEs are never activated/deactivated together. This helps to accom-

plish the following two goals: (a) the peak instantaneous current is reduced and (b) the activated PEs are separated spatially which implies that voltage glitches that arise are isolated and don't affect the other parts of the array.

Fig. 7 shows two possible separations of the processor array. Each node in the grid represents a PE and all identically marked PEs are activated/deactivated together. In the first case, the PEs are divided into two types and it takes two cycles to activate the entire array, whereas the second case shows a four way division of the grid. While the second case takes four cycles to complete the activation step, it offers additional separation between the PEs. By using separate clock phases, we could accomplish the entire process within a single clock cycle, thereby eliminating completely any impact of the staggered activation on the throughput of the SIMD processing array.

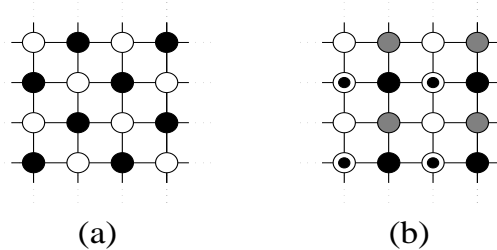


Figure 7. SIMD implementation

To run our experiments we used SIMPIL16 [3], an architectural level SIMD simulator developed at Georgia Institute of Technology. Running typical image-processing applications like median filtering, wavelet decomposition, spatial filtering, discrete Fourier transform, morphological filtering, image rotation etc. revealed that activation/deactivation instructions constitute about 10% to 15% of the total workload. Further, hundreds or even thousands of PEs on an average get activated/deactivated on each of these instructions. Measuring the extent of $\partial I/\partial t$ that this may introduce is a daunting task, since it is not easy to conduct SPICE like simulations for a single PE, let alone an array of thousands of processors. However, since this scheme is similar to that proposed for superscalar architectures which revealed a 2x improvement for a 16 bit ALU, we can say that it is definitely a feasible solution for reduction of ground bounce in SIMD systems. Also significant in this scheme is the zero performance impact that it will cause, if the staggered activation is done using separate clock phases in a single cycle.

Using SIMPIL16 chips fabricated in CMOS 0.8 micron technology, work is in progress to obtain measurement data that will not only reveal the extent of $\partial I/\partial t$ in SIMD architectures, but will also show how it can be alleviated when our proposed scheme is applied.

6. Conclusion

This paper has presented a technique to reduce $\partial I/\partial t$ noise at the architectural level at a modest cost in hardware and performance. SPEC95 benchmarks were simulated to examine the latency hiding capacity of a typical superscalar processor employing this approach. SPICE simulation of a typical ALU shows a decrease of about 2x in instantaneous current, with a performance cost of less than 5% and minimal additional hardware. The scheme has been extended to the realm of SIMD architectures wherein high amounts of instantaneous current can be reduced at no performance loss and minimal hardware cost.

References

- [1] H. B. Bakoglu. *Circuits, Interconnections and Packaging for VLSI*. Addison-Wesley, 1990.
- [2] D. Burger and T. M. Austin. The SimpleScalar tool set, version 2.0. Technical Report 1342, Univ. of Wisconsin-Madison Computer Science Department, June 1997.
- [3] H. H. Cat and et. al. Simpil: An oe integrated simd architecture for focal plane processing applications. In *Massively Parallel Processing using Optical Interconnection*, pages 44–52, 1996.
- [4] S. M. Chai, A. Gentile, and D. S. Wills. Impact of power density limitations in gigascale integration for the simd pixel processor. In *Conference on Advanced Research on VLSI*, pages 57–71, Mar 1999.
- [5] Y.-S. Chang, S. K. Gupta, and M. A. Breuer. Analysis of ground bounce in deep sub-micron circuits. In *VLSI Test Symposium*, pages 110–116. IEEE, IEEE Computer Society Press, 1997.
- [6] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, Inc., 1990.
- [7] H. W. Johnson and M. Graham. *High speed Digital Design: A Handbook of Black Magic*. Prentice Hall, January 1993.
- [8] M. D. Pant, P. Pant, D. S. Wills, and V. Tiwari. An architectural solution for the inductive noise problem due to clock-gating. In *To be presented at International Symposium on Lower Power Electronics and Design*, Aug 1999.
- [9] C. Price. *MIPS IV Instruction Set, revision 3.1*. MIPS Technologies, Inc., Mountain View, CA, Jan 1995.
- [10] J. M. Rabaey and M. Pedram. *Low Power Design Methodologies*. Kluwer Academic Publishers, 1996.
- [11] R. Senthinathan and J. L. Prince. Simultaneous switching ground noise calculation for packaged CMOS devices. *IEEE Journal of Solid-State Circuits*, 26(11):1724–1728, Nov 1991.
- [12] V. Tiwari, D. Singh, S. Rajagopal, G. Mehta, R. Patel, and F. Baez. Reducing power in high-performance microprocessors. In *Design Automation Conference*, pages 732–737, June 1998.