An Architectural Solution for the Inductive Noise Problem due to Clock-Gating

Mondira Deb Pant Pankaj Pant D. Scott Wills
{mandy,pant,scotty}@ece.gatech.edu
Georgia Institute of Technology, Atlanta , GA

Vivek Tiwari
vtiwari@ichips.intel.com
Intel Corporation, Santa Clara, CA

Abstract

As we approach Gigascale Integration, chip power consumption is becoming a critical system parameter. Clock-gating idle units provides needed reductions in power consumption. However, it introduces inductive noise that can limit voltage scaling. This paper introduces an architectural approach for reducing inductive noise due to clock-gating through gradual activation/deactivation of units. This technique provides a 2x reduction in ground bounce on a 16 bit ALU simulated in SPICE, while reducing simulated SPEC95 performance by less than 5% on a typical superscalar architecture.

1 Introduction

The quest to increase computer performance has led to large complex microprocessors operating at high clock speeds. In order to provide reasonable chip execution, it has become necessary to carefully consider power density, power delivery and chip packaging. There is growing concern that power consumption may set a limit on how much can be integrated on a chip and how fast it can be clocked.

Power breakdown in a high performance CPU reveals the clock distribution network to be the largest power consuming component [6]. Reducing the switched capacitance on the clock would help reduce the total power. A popular and easily incorporated technique to achieve this is clock-gating [6], which allows only the required portions of the clocked network to toggle. However, it cannot be used indiscriminately since it can introduce signal integrity problems.

Ground bounce [4] [1] [5] [3], 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 its magnitude has been very small compared to the noise margin of CMOS circuits. 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 also decreases. 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 clock-gating. SPICE simulations of a test ALU suggest a 2x reduction in ground bounce with a cost of less than 5% in performance.

2 Related Work

Clock-gating methods to reduce power consumption of circuitry, essentially involve ANDing the clock feeding a unit with a control signal which shuts down the unit when not in use. However, rapid powering down implies rapid switching and hence enhanced ground bounce. Moreover, the powered down unit may not become ready on time which could cause synchronization problems. This calls for very careful design that ensures such race conditions cannot exist.

In [6] 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. Figure 1 shows real power vs. time curves for a Pentium® Processor (from [6]). 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. However, 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.

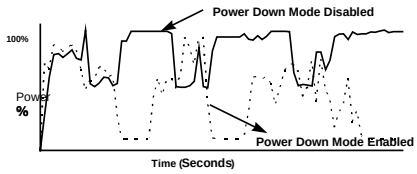

Figure 1: Power vs. time plot for a Pentium ® Processor [6]

3 Proposed Clock-Gating Scheme

Currently, in modern processors clock-gating is implemented such that the processor pipeline itself is unaware of the "state" of a resource (ALU, multipliers/dividers etc.). It is assumed that the unit will always be available for execution provided that it is not busy with a previous instruction. In the case that the unit is busy, the pipeline might have to be stalled till it becomes available again. 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 increase the time for the units to switch on (off). This causes them to draw a lesser instantaneous current and hence reduces the glitch in the power/ground lines. However, since it now takes a finite amount of time for a 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 small for a typical modern day microprocessor. The reliability level attained by the proposed scheme justifies this small drop in performance.

To measure the performance change of a processor due to the proposed clock-gating 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. 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 studies we modified the SimpleScalar simulator to incorporate the concept of clock-gating. Originally the simulator assigned one of three states *sleep/dormant, busy* or *ready* to every resource depending of whether it is dormant, in use or ready for use. A resource being dormant can be interpreted as a way of denoting that the clock to it is turned off. If there is a request for a dormant resource from the scheduler, then it is assumed to be made available immediately by turning on its clock. Figure 2(a) shows a typical timing diagram of a resource.

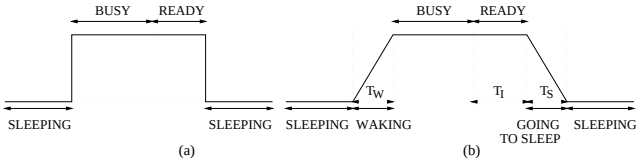

Figure 2: Timing diagram for operator states

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 Figure 2(b). If a resource is requested for when it is dormant, instead of going from sleep to busy directly, it goes through an intermediate waking state during which time it cannot be used. T_W and T_S denote the number of clock cycles taken by a resource to wake up and go to sleep respectively. A resource goes to *sleep* after it has been *ready* for a pre-determined number of cycles referred to as the idle time, T_I. 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.

The proposed clock-gating scheme need not be restricted to basic functional units (like ALU's etc.). The idea of a clock-gated resource can be extended to complete systems and processors as in SIMD arrays where entire sets of processors are simultaneously turned on and off, causing large instantaneous currents to be drawn from the power lines.

In any pipeline architecture, the finite time required to access a dormant resource may be interpreted as a "bubble" or a stall being introduced into the pipeline. The value of T_W determines the performance cost introduced. After compiling the SpecInt95 and SpecFP95 benchmarks for the SimpleScalar architecture, we ran the benchmarks on a prototype modern day processor [Table 1] to study the difference in performance due to the proposed scheme.

The instructions-per-cycle (IPC) for the SPEC95 benchmarks 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 T_W of the resources. (T_S was assumed to be the same as T_W.) T_I was set to 10 cycles for all the experiments. Figure 3 shows the variation of the IPC for various SpecInt95 and SpecFP95 benchmarks plotted against T_W.

Increasing the value of T_W/T_S obviously reduces the IPC. However, we observe that the drop is less than 5% of the original IPC when T_W is less than or equal to 3 cycles, clearly indicating that the loss of performance is minimal for small values of the clock-gating delay. Our experiments revealed that the impact on IPC decreased if a larger number of resources were available.

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
Memory bus width = 16
Main memory seek time = 18 cycles

Table 1: Superscaler architecture used for study

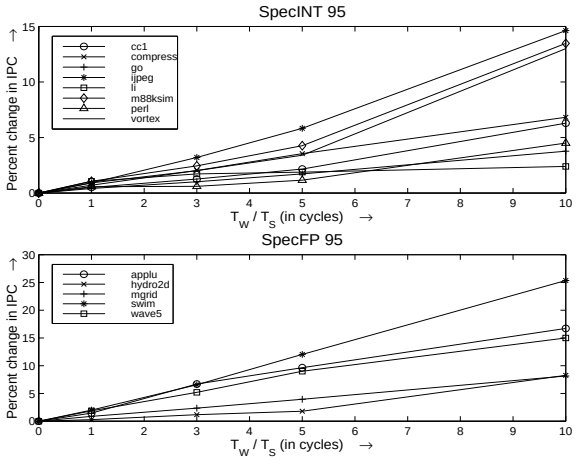

Figure 3: IPC variation for SPEC95 benchmarks

The small loss in performance may be eliminated almost totally if it is possible to predict 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 Implementation and Verification

Until now, we have not made any assumptions as to how the resources are actually woken up and put to sleep over a finite period. There are a number of possible circuit realizations to accomplish this task. As such our proposed scheme is independent of the circuit implementation.

Figure 4 shows one possible circuit implementation of the proposed scheme. The clocked unit is divided into a variable number of slices, n (n = 4 in Figure 4). During normal operation of the functional unit, each slice gets the same clock signal. When the unit needs to be clock-gated, the enable signal goes low. During successive cycles each of the n slices gets clock-gated, one per cycle, as the disable signal travels down the chain of flip-flops. This leads to a staggered powering down of the unit. Therefore n cycles after the initial clock-gating signal is issued, the entire unit finally

shuts down, instead of an instantaneous power down. Similarly, the 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 datapath, this implementation has a potential of drastically reducing the $\partial I/\partial t$ problem encountered when large units are powered up or down. Notice that we make no assumptions as to the division of the resource. The split might be done 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 unaffected by the split. In order to reduce the delay to less than n cycles, one may clock the delay flip-flops faster than the circuit clock. Hence, n phases after the initial clock-gating signal is generated, the entire unit gets powered down.

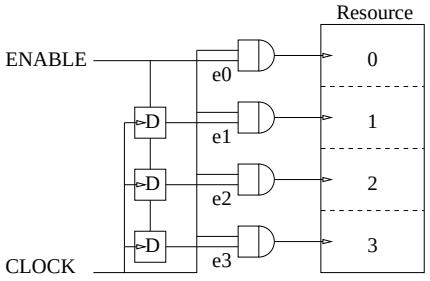

Figure 4: Circuit Implementation

One of the advantages of the implementation in Figure 4 is that state information can be easily extracted for each resource. The "dormant" state, for instance, is indicated by all the four disable signals (e_0, e_1, e_2 and e_3 in Figure 4) being low. The "busy" and "idle" states, on the other hand, are detected when all the disable lines are high. The resource is "going to sleep" when e_0 is low but at least one of e_1, e_2 and e_3 are high and, finally, the resource is said to be "waking up" when e_0 has gone high and one or more of the other three disable signals (e_1, e_2 and e_3) are still low. Figure 5 illustrates simple circuits to realize the state detection logic.

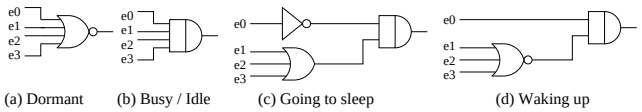

Figure 5: State detection logic

Reducing power supply voltage (V_{dd}) to the unit, or even increasing the threshold voltage (V_{th}) of the devices in the unit during the "waking-up" and "going-to-sleep" process of a resource may be other ways to slow it down and hence reduce the instantaneous current drawn during these times.These techniques would fit in well with schemes which use multiple-supply voltages for low power operation scheduling and dynamic threshold voltages for standby leakage power reduction.

To see the extent of improvement in $\partial I/\partial t$, the 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 Figure 6(a) and (b) respectively. Figure 6(c) shows a plot of current drawn across the V_{dd} line versus time for the original single clock circuit (n = 1). Figures 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 spread out in the proposed scheme, the current spikes observed drop as the number of slices, n, increases. The peak current indicates the rate of change of the current and hence the extent of the voltage drop across the V_{dd} line. We 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.

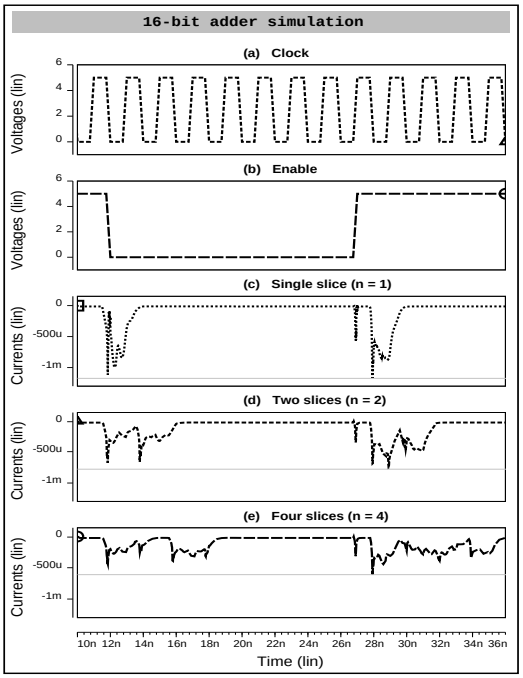

Figure 6: HSPICE simulation of an 16-bit ALU

5 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.

References

[1] BAKOGLU, H. B. *Circuits,Interconnections and Packaging for VLSI*. Addison-Wesley, 1990.

[2] BURGER, D., AND AUSTIN, T. M. The SimpleScalar tool set, version 2.0. Tech. Rep. 1342, Univ. of Wisconsin-Madison Computer Science Department, June 1997.

[3] CHANG, Y.-S., GUPTA, S. K., AND BREUER, M. A. Anaylsis of ground bounce in deep sub-micron circuits. In *VLSI Test Symposium* (1997), IEEE, IEEE Computer Society Press, pp. 110–116.

[4] JOHNSON, H. W., AND GRAHAM, M. *High speed Digital Design: A Handbook of Black Magic*. Prentice Hall, January 1993.

[5] SENTHINATHAN, R., AND PRINCE, J. L. Simultaneous switching ground noise calculation for packaged CMOS devices. *IEEE Journal of Solid-State Circuits 26*, 11 (Nov 1991), 1724–1728.

[6] TIWARI, V., SINGH, D., RAJAGOPAL, S., MEHTA, G., PATEL, R., AND BAEZ, F. Reduing power in high-performance microprocessors. In *Design Automation Conference* (June 1998), pp. 732–737.