Designing Memory Subsystems Resilient to Process Variations

Mahmoud Bennaser
Department of ECE
University of Massachusetts
Amherst, MA 01003
mbennase@ecs.umass.edu

Yao Guo
Department of ECE
University of Massachusetts
Amherst, MA 01003
yaoguo@ecs.umass.edu

Csaba Andras Mortiz
Department of ECE
University of Massachusetts
Amherst, MA 01003
andras@ecs.umass.edu

Abstract

As technology scales, more sophisticated fabrication processes cause variations in many different parameters in the device. These variations could severely affect the performance of processors by making the latency of circuits less predictable and thus requiring conservative design approaches. In this paper, we use Monte-Carlo simulations in addition to worst-case circuit analysis to establish the overall delay due to process variations in a cache sub-system under both typical and worst-case conditions. The distribution of a cache critical-path-delay in the typical scenario was determined by performing Monte-Carlo simulations at different supply voltages, threshold voltages, and transistor lengths on a complete cache design. In addition to establishing the delay variation, we present an adaptive variable-cycle-latency cache architecture that mitigates the impact of process variations on access latency by closely following the typical latency behavior rather than assuming a conservative worst-case design-point. Simulation results show that our adaptive data cache can achieve a 9% to 21% performance improvement in a superscalar processor, on the SPEC2000 applications studied, compared to a conventional design. Additional performance improvement potential exists in processors where the data cache access is on the critical path, by allowing a more aggressive clock rate.

1. Introduction

As technology scales, the feature size reduces thereby requiring a sophisticated fabrication process. The manufacturing process causes variations in many different parameters in the device, such as the effective channel length L_{eff}, the oxide thickness t_{ox}, and the threshold voltage V_{th}. These variations increase as the feature size reduces due to the difficulty of fabricating small structures consistently across a die or a wafer [3]. Controlling the variation in device parameters during fabrication is becoming therefore a great challenge for scaled technologies.

The performance of integrated circuits can be greatly affected by these variations. The process variations are random in nature and are expected to become significant in the smaller geometry transistors commonly used in memories. Question is whether there is a significant enough delay variation overall that will drive a change in memory architecture design.

Our simulation results with HSPICE show that process variations on effective channel length and threshold voltage at 32-nm CMOS technology can affect the performance, after all factors are considered, at around 2-3X under the worst-case operating conditions. To account for the worst-case scenario we might need to increase the cache access time by 2 to 3 cycles or adopt other design approaches. Application performance could be impacted by as much as 30-40% as shown in Figure 1.

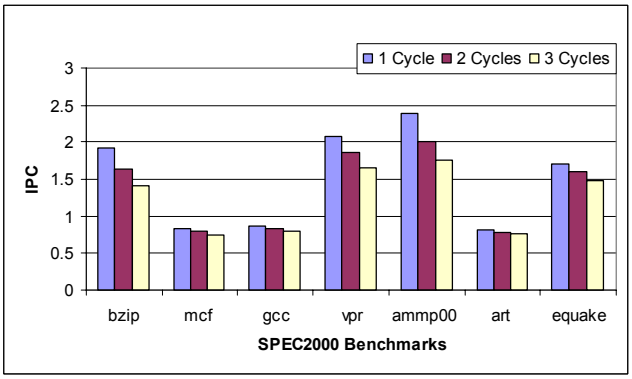

Figure 1. Application performance for different cache access cycles in 4-way superscalar (see Table 4 for parameters).

These results suggest that process variations must be taken into consideration while designing circuits and perhaps even architectures. There are several ideas that could be exploited in a memory system: 1) reduce performance by operating at a lower clock frequency (conservative approach); 2) increase cache access latency assuming worst-case delay (conservative approach); and 3) variable delay cache architecture (adaptive approach proposed in this paper). The first approach would clearly have a large impact on overall performance. The second

IEEE Computer Society Annual Symposium on VLSI(ISVLSI'07)
0-7695-2896-1/07 $20.00 © 2007 IEEE

approach would also have a significant impact as shown above in Figure 1.

The goal of this paper is to estimate both the worst-case and typical delay variation expected in a state-of-the-art cache and to introduce an adaptive cache system that would mitigate the impact of process variations without taking any of the conservative design paths suggested earlier.

The rest of this paper is organized as follows. Section 2 presents a detailed analysis on the impact of process variation on caches under worst-case and expected behavior conditions. To estimate the typical delay in a cache, we determine the distribution of delays by performing Monte-Carlo sampling at different supply voltages, threshold voltages, and transistor lengths. In Section 3, we describe architecture techniques to mitigate the effect of process variations and propose a variable-cycle adaptive cache. We show simulation results in Section 4 by running applications on a superscalar processor with this design. We have implemented the cache at circuit level and extended the SimpleScalar [5] architecture simulator. We use a set of SPEC2000 [2] benchmarks to compare the performance with a conventional approach. We conclude in Section 5.

2. Impact of Process Variation in Caches

In this section, we analyze the impact of different sources of process variations in caches under worst-case-operating and expected-behavior conditions. We use a state-of-the-art low power cache that we have designed in our research group as the starting point for our evaluation.

Table 1 shows the configuration of this cache design.

Table 1. Configuration of our 16 KB Low Power Cache

Cache Component	Power Optimization Technique
Tag Array	10 transistors CAM Cells
Data Array	6T SRAM Cells
Cache Line	Wordline Gating
Tag and Data Array	Cache Subbanking
Column S. A.	Alpha Latch S. A. and Sharing Sense Amplifiers
Bank Decoder	4-input static NOR gates
Line Decoder	Two level decoding: First level 3-input Dynamic NAND gate and Second level 2-input NOR gate

In order to evaluate the impact of the parameter variations on circuit speed we consider variability on the critical path. The critical path of our CAM-tag cache is shown in Figure 2.

The CAM-Tag critical path in our design is composed of the global address decoder to select a bank, tristate I/O to drive the search bitlines, the dynamic match comparators in the CAM cells, wordline gating, the data SRAM array, column multiplexer, sense amplifier and the tristate I/O drivers connecting back to the CPU. The tristate bus that connects one 32-bit subbank column back to the CPU 32-bit load data path has the same fan-in in all configurations.

Process variations in caches affect the performance of circuits like sense amplifiers that require identical device characteristics, and SRAM cells that require near-minimum-sized cell stability for large arrays in embedded, low-power applications. In addition, the delay of the address decoders suffer from the process variations that can result in shorter time left for accessing the SRAM cells.

In order to examine delay tradeoffs under process variations, we have evaluated the impact of process variations under both worst-case operating and typical behavioral conditions. The goal here is to establish a worst-case baseline that might be used in conventional conservative designs and also a typical behavior that could be used to estimate the benefits of migrating to an adaptive design.

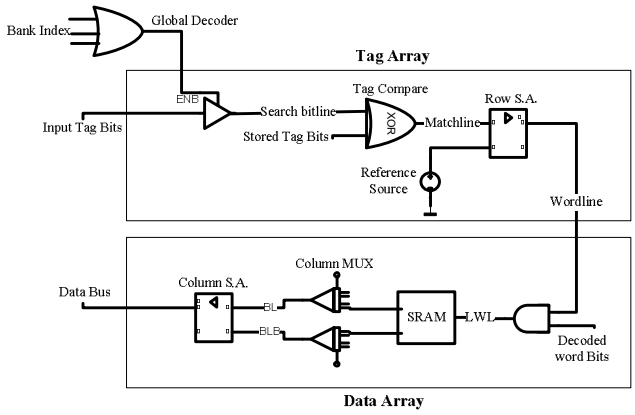

Figure 2. Critical path of a CAM-tag cache

2.1 Worst-Case Conditions

Under worst-case operating conditions, we assume that parameter variations happen at each transistor in the cache critical path. We have used the HSPICE circuit simulator at 32-nm PTM device model [1]. The nominal value used for V_{th} is 0.2V and the nominal value for L_{eff} is 25.3nm, given by the PTM technology [1].

2.1.1 Channel Length Variation

Channel length variation L_{eff} is due to limitation in the lithographic process. These variations result in changes in device performance characteristics. A total of 40% variation in effective channel length L_{eff} is expected within a die [3]. We have found that the use of longer effective channel lengths tends to increase the wordline and bitline capacitances in caches, thus increasing access time as shown in Figure 3. The access time can vary by as much as 2.13X.

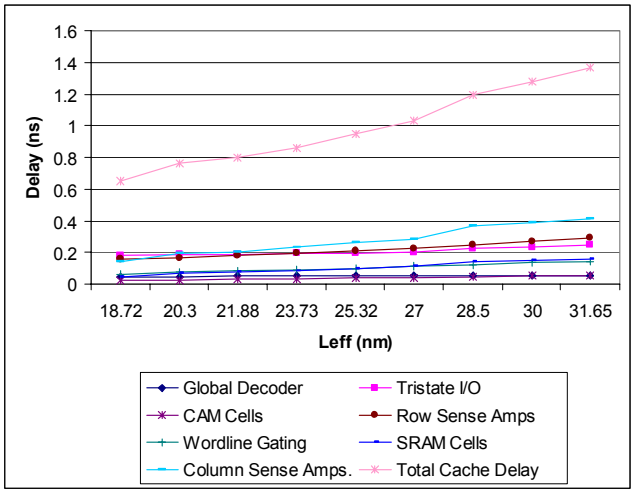

Figure 3. Effect of Leff variation on cache delay

2.1.2 Threshold Voltage Variation

Threshold voltage can vary due to (1) changes in oxide thickness, (2) changes in the dopant levels in the substrate, polysilicon and implants, and (3) surface charge. Accurate control of V_{th} is very important for many performance and power optimizations and for correct execution [6]. Higher transistor threshold voltage V_{th}, due to process variations, impacts the access time due to the lower read current as shown in Figure 4. The impact on the access time could be as much as 2.7X.

2.1.3 Supply Voltage Variation

One of the most important environmental factors that cause variations in operating condition is supply voltage (V_{dd}). In deep submicron technology the supply voltage is typically scaled down to reduce power consumption; effects such as the *IR* voltage drop and *L di/dt* noise can affect the voltage level at the power supply thereby modifying the characteristics of the transistors in the circuits.

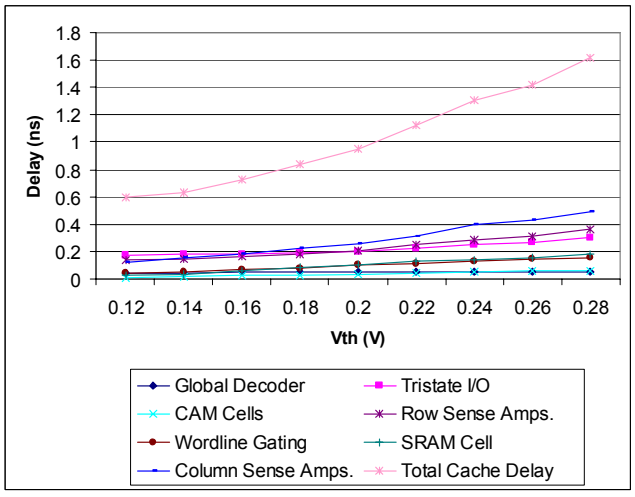

Figure 4. Effect of Vth variation on cache delay

A total variation of 15% in V_{dd} was considered [3] with a nominal value of 0.9V. Table 2 summarizes our results showing delay for our cache design. From Table 2, a reduction in supply voltage causes an increase in the access time of the cache by up to 12% of the nominal value.

Table 2. Effect of power supply voltage variations

V_{dd} (V)	Delay (ns)
0.83	0.746
0.86	0.717
0.90	0.667
0.93	0.634
0.97	0.601

The deviations in effective channel length and threshold voltage are shown to have a more significant contribution to the delay than variations in power supply voltage. The impact on cache access time due to process variations and longer wordline/bitline could become very significant. The access delay could be impacted by around 2-3X (compared to the nominal value) when counting all the possible process parameters.

2.2 Expected Conditions

The simple use of worst-case values for all parameters that have been shown in Section 2.2 can result in larger path delay estimates than typical. These will certainly be pessimistic but would need to be considered in conventional designs. Now, the question we try to answer next is what the delay distribution is in a cache due process variation?

To accurately predict critical path delay distribution at the circuit level, cache delay variability can be studied through Monte-Carlo in HSPICE circuit simulations. Process variations are typically represented by continuous probability distributions, and are often assumed as normal distribution [4].

The distribution of delay of a cache critical path was determined by performing Monte-Carlo sampling at different supply voltages, threshold voltages, and transistor lengths. Under the assumption of separated normal distributions of L_{eff}, V_{th} and V_{dd} variations, Monte-Carlo simulations verify model predictions over a wide range of process and design conditions. We have used the Monte-Carlo simulation with 5,000 trials where the variation sources all vary simultaneously. We simulate the critical path and measure delay with all the parameters varying with 3σ and mean values as specified in Table 3.

Table 3. Parameter values and 3σ variations

Technology	32nm	
Device	NMOS	PMOS
L_{eff}	25.32nm (+-20%)	
V_{th}	0.2V (+-7.5%)	-0.21V (+-7.5%)
V_{dd}	0.9V (+-7.5%)	
Temp.	75°C	

The probability density function (PDF) of the cache delay was measured (see Figure 5) for each process parameter. Also, we have combined all the parameters in another experiment. We have found most the cache accesses under the impact of supply voltage or threshold voltage parameters would be relatively close to the nominal delay. The deviations in L_{eff} are shown to have a significant contribution to the delay distribution (wider curve). It is also very close to the case with all the parameter combined.

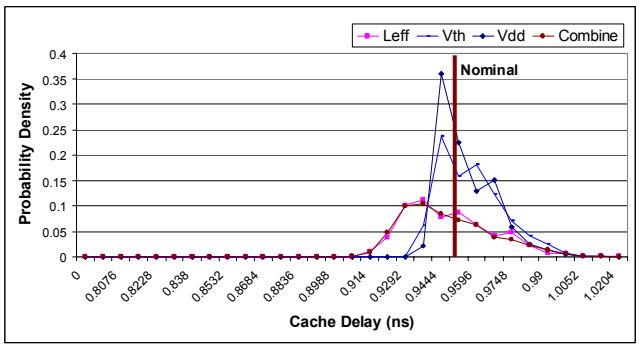

Figure 5. Distribution of the cache access latency.

Out of 5,000 random samples, assuming a 1 cycle cache at 1 GHz, 2,000 samples of the cache accesses are expected to be faulty, resulting in a probability of failure of 40%. However, with an increased cache delay of 2 cycles allowed and after adjusting the path across the components to accommodate a larger variation in the SRAM access, the probability that this cache will have to take 2 cycles has been found to be only 25%. This means that in an adaptive scenario only 25% of the accesses would require 2 cycles. We have also found that even in this case a small fraction of accesses would fail suggesting that there are cases that would need 3 cycles for correctness.

3. Architectural Techniques

At the architectural level, we might be able to help to mitigate the negative impact of process variation such that the low-power circuits and designs can still be applied. There are several ideas that could be exploited to cope with this problem while not giving up performance. These could range from utilizing smaller first level caches (that would meet the preferred access time even under worst case variation) to more adaptive cache architectures that we will present next.

3.1 Conservative Cache

As we have shown in the previous section, process variations affect the latency significantly for each cache access. The cache access latency difference could be as much as 3X if we consider all the possible variations in process parameters. The conservative cache would have to be based on the worst-case process variation analysis (as shown in Section 2.1). Alternatively, one could make the cache access time slightly more aggressive (than the conservative one) but then the yield would be likely affected.

In a conservative cache design, to ensure the correct execution in the pipeline architecture, the cache access delay cycles must be decided based on the longest delay possible within the process variation range. For example, even a small latency increase due to variation may require the whole cache access latency to be increased by one or more processor cycles. This can severely degrade the overall application performance.

For example, even if we could access 90% of the cache lines within 1 cycle, for the remaining 10%, we might need 2 or 3 cycles to finish the access due to the delay increase resulting from process variations. In this situation, we might need to assume the worst-case scenario, which is three cycles for all the cache accesses. This will, as a result, severely affect the total performance and is clearly not a choice that we can live with even if some architectural tricks could be applied to hide the cost.

For example, clever scheduling techniques might try to increase the distance between memory reads and consumer instructions to hide a longer latency. As we have seen, however, there is a limit to how much that can help as very often basic blocks are short and the scheduler cannot move dependent instructions several cycles away.

3.2 Proposed Adaptive Cache

In the proposed adaptive cache design, instead of accessing the cache with a long fixed latency assuming worst-case conditions, the adaptive architecture can have different access latency for different cache lines. The typical case analysis encourages efforts towards developing adaptive design methodologies that suppress the impact of process fluctuations on performance. The expectation is that most of the cache lines will have much lower latency compared to the worst-case scenario.

Figure 6 shows a possible adaptive architecture. One of the important blocks in the proposed architecture is the delay storage unit. This unit stores the speed information and is read along with the data array on every cache access. The operation on the delay storage has two phases: classification and execution.

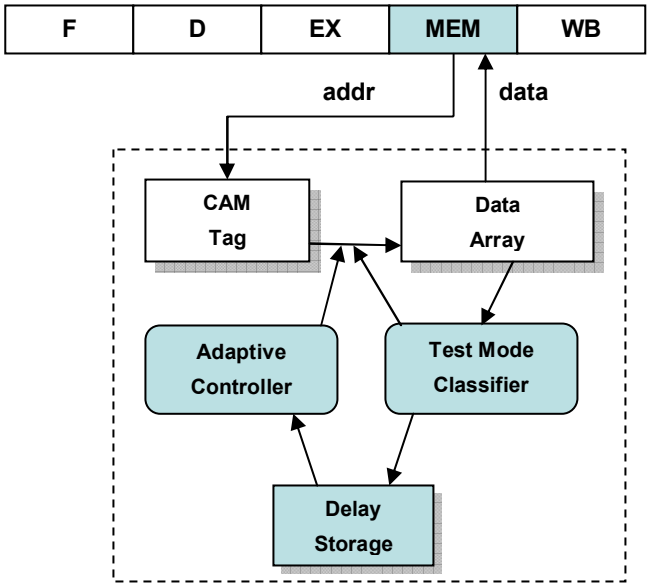

Figure 6. The proposed adaptive cache architecture (shown in a single 5-stage pipeline).

Delay information for each cache line is first achieved during a classification process where each cache line is probed individually and its delay information is written into the delay storage unit. Then, during the execution phase, the delay information is fetched from the delay storage and each cache line can be accessed based on its estimated speed.

With the addition of the delay storage, we are able to access the cache with an adaptive speed. The adaptive architecture will enable us to maximize the performance compared to the traditional fixed latency cache architecture.

The area penalty for this cache is really minimal as we only need to use 2 bits (our example with 1-3 cycles latencies) for each cache line (or 256 bits) to encode the speed. The sense amplifiers would need to be triggered at different time points depending on the speed access. In our analysis, we have evaluated the area overhead associated with extra BIST, delay storage, and control circuitry by using the Synopsys Design Compiler CAD tool. We have found the overall area overhead to be less than 1% of the total cache area. Because the delay storage is a small structure, its own delay variation due to process variation is relatively small compared to the cache.

4. Results and Analysis

The initial adaptive cache architecture is implemented in SimpleScalar with the simulation parameters summarized in Table 4. We have conducted simulations of SPEC2000 benchmarks using the adaptive approach. We vary the cache access latency from 1 to 3 cycles. The adaptive cache based on the delay distribution is determined by the Monte-Carlo simulation. Based on our analysis, the adaptive cache is expected to have 75% of 1 cycle, 25% of 2 cycles and negligible 3 cycles cache line accesses.

Preliminary results on application performance are shown in Figure 7. The comparison is made between a conservative cache that requires 3 cycles per access and an adaptive cache that has variable cache access latencies.

Our results show that the adaptive cache design can achieve a 9% to 21% performance improvement on the applications studied compared to a conservative design assuming worst-case latency, while providing resilience against failures due to process variations. Using the adaptive cache architecture can also mean that one can set the clock rate slightly more aggressively: the increase in clock rate would likely compensate for a larger fraction of memory accesses falling into higher-latency memory access categories in a processor. Furthermore, when low power is important, a slightly slower cache (e.g., due to asymmetric cell designs with some high V_{th} transistors to reduce cell power) would mean a redistribution between 1, 2, and 3 cycle accesses.

Table 4. SimpleScalar parameters for CPU

Instruction Window	RUU=16; LSQ=8
Fetch, dispatch, commit width	4
Integer ALU/mult-div	4/1
FP ALU/mult-div	4/1
Number of Banks	16 banks
L1 D-cache Size	16KB, 32-way, 32B blocks, 2 cycles
L1 I-cache Size	16KB, 32-way, 32B blocks, 2 cycles
L2 Unified Cache Size	128KB, 64-way, 4B blocks, 8 cycles
Memory Latency	100 cycles
Memory Ports	2
TLB Size	128-entry, fully assoc., 30-cycle miss penalty
Branch Predictor	Comb. Of bimodal and 2-level gshare; bimodal size 2048; level1 1024 entries, history 10; level2 4096 entries (global)
Branch Target Buffer	512-entry, 4-way
Return-address-stack	8-entry

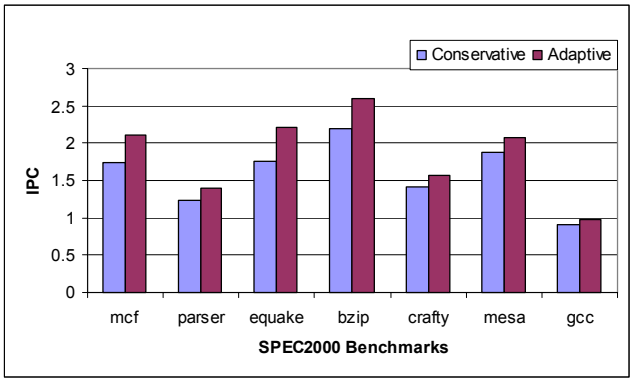

Figure 7. Performance improvement between the adaptive cache vs. a conservative cache using 3-cycle access time.

5. Conclusion

Process variations will become worse with technology scaling; techniques are necessary at the architecture and circuit levels to reduce the impact of these variations while providing the highest performance for the given power constraints. In this paper, we have found significant delay variation between worst-case and expected behavioral analysis, motivating us to design adaptive cache architecture. We have shown that process variation can have a significant impact on delay (2-3X) under worst-case operating conditions, while under the expected condition a large fraction of accesses would be still close to the nominal value. The adaptive cache architecture proposed can improve the application performance in a superscalar design by as much as 21% depending on the application and configurations used, compared to a conservative design. The adaptive cache architecture also allows a designer to choose the main cache access latency more aggressively and possibly increase the clock rate in a processor design where cache access is the main critical path. Furthermore, it could help strike a better balance between power and delay optimizations in a design.

6. REFERENCES

[1] Predictive technology model. Nanoscale Integration and Modeling Group at ASU. [Online]. Available: http:// www.eas.asu.edu/~ptm

[2] The standard performance evaluation corporation, 2000. http://www.spec.org

[3] D. Boning and S. Nassif. Models of process variations in device and interconnect. In *Design of High-Performance Microprocessor Circuits, A.Chandrakasan, Chapter 6*, pp. 98–115, IEEE Press 2001.

[4] K. Bowman, X. Tang, J. Eble, and J. Menldl. Impact of extrinsic and intrinsic parameter fluctuations on cmos circuit performance. In *IEEE Journal of Solid- State Circuits*, volume 35, pp. 1186–1193, Aug 2000.

[5] D. C. Burger and T. M. Austin. The SimpleScalar tool set, version 2.0. Technical Report CS-TR-1997-1342, University of Wisconsin, Madison, June 1997.

[6] X. Tang, V. K. De, and J. D. Meindl. “Intrinsic mosfet parameter fluctuations due to random dopant placement”. *IEEE Trans. Very Large Scale Integr. Syst.*, Vol. 5 (No.4): pp. 369–376, 1997.