Data Memory Subsystem Resilient to Process Variations

Mahmoud Bennaser, *Student Member, IEEE*, Yao Guo, *Member, IEEE*, and Csaba Andras Moritz, *Member, IEEE*

***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 data cache sub-system under both typical and worst-case conditions. The distribution of the 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 31% performance improvement in a superscalar processor, on the SPEC2000 applications studied, compared to a conventional design. The area overhead for the additional circuits of the adaptive technique has less than 1% of the total cache area. Additional performance improvement potential exists in processors where the data cache access is on the critical path, by allowing a more aggressive clock rate.**

***Index Terms*—CMOS memory integrated circuits, memory architecture, process variations.**

I. 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 [11]. The question is whether there is significant delay variation that will drive a change in memory architecture design.

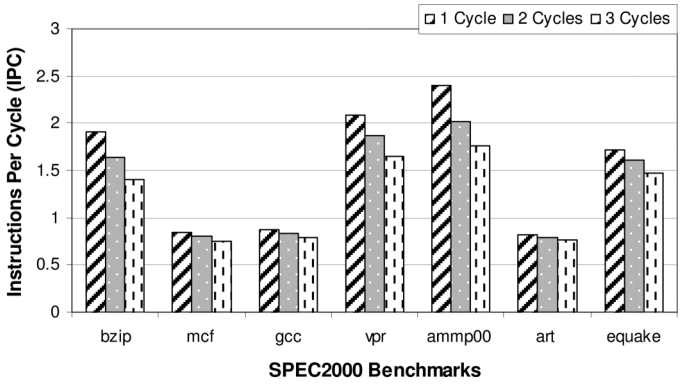

Fig. 1. Application performance for different cache access cycles (see Table IV for the ratio occurrence of these variations).

In state-of-the-art digital design, to ensure the correct execution in the memory circuit, the cache access delay must be decided based on the worst-case latency within the process variation range [12]. 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. The increase of delay due to process variations will be severely impact in low power circuits.

Preliminary simulations results with HSPICE show that process variations on effective channel length, power supply voltage and threshold voltage at 32-nm CMOS technology can affect the performance, after all factors are considered, at around 2–3× under the worst-case operating conditions [7]. To account for the worst-case scenario we might need to increase the cache access time by 2–3 cycles or adopt other design approaches. Application performance could be impacted by as much as 30%–40% as shown in Fig. 1.

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

While there has been a lot of work on statistical methods to modeling and compensating process variation at the circuit

Manuscript received April 11, 2007; revised September 29, 2007. Current version published November 19, 2008.

M. Bennaser is with the Department of Computer Engineering, Kuwait University, Khaldiya, Kuwait (e-mail: bennaser@eng.kuniv.edu.kw).

Y. Guo is with the School of Electrical Engineering and Computer Science, Peking University, Peking 100871, China (e-mail: yaoguo@cs.pku.edu.cn).

C. A. Moritz is with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003 USA (e-mail: andras@ecs.umass.edu).

Digital Object Identifier 10.1109/TVLSI.2008.2001299

level [16], [17], there has been little work on modeling variations at the architecture level. This paper presents a model that allows architects to reason about how process variations affect cache design. In [18], Agarwal *et al.* introduce process-tolerant cache architecture to improve the yield in nanoscale memories. The proposed scheme detects and replaces faulty cells by dynamically resizing the cache. Humenay [12] introduce a pre-register transfer level (RTL), architectural modeling methodology that incorporates the impact of process variations on multi-core chips. Perhaps the most relevant prior work is the "FMAX" model introduced by Bowman [19]. FMAX is a predictive model for capturing the maximum frequency distribution of a microprocessor.

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. This paper is an extension of our previous conference paper [7] with a complete design and evaluation of all cache circuits and architectural components.

The proposed adaptive cache circuit is transparent to other subsystem and has negligible power and area overhead. Instead of accessing the cache with a fixed latency assuming worst-case latency, the proposed architecture can have different latency information stored in the delay storage unit for each cache line. The adaptive technique is enabled by a built-in self-test (BIST) circuitry, which tests the entire cache and detects the speed of the cache access. The speed results are then written into the delay storage. While we access each cache line, we will know the latency for that cache line and the pipeline will act accordingly to save the waiting cycles. The expectation is that most of the cache lines will have lower latency compared the worst-case scenario.

The rest of this paper is organized as follows. In Section II, we briefly describe a state-of-the-art low power cache that we have designed in our research group as the starting point for our evaluation. Section III presents a detailed analysis on the impact of process variation on this cache 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 IV, we describe new architecture techniques to mitigate the effect of process variations and propose a variable-cycle adaptive cache. We show simulation results in Section V 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 VI.

II. Low Power Baseline Cache

Before we start to evaluate the impact of process variations on caches, we would like to establish a baseline cache with state-of-the-art low power techniques. First, we introduce an initial cache with no dedicated power optimization techniques. The initial cache design is a 16 kB, uses nine-transistor NOR-type match line CAM cells [15] and the 6T SRAM cells, for the tag array and the data array circuits, respectively. A single-ended sense amplifier was used for match-line signal sensing in the tag array and a cross-coupled inverter latch was used for read data sensing in the data array. For the bank decoder, we select a four-input static NOR gates-based design; for the cache line decoder, we apply two-level decoding: first level three-input dynamic NAND and second level two-input static NOR gate.

TABLE I
Active Power Reduction Techniques

Cache Component	Power Optimization Technique
Tag Array	10-transistor NOR type match line CAM cells with separate search bitlines from write bitlines [14]
Cache Line	Wordline Gating
Tag and Data Array	Cache Subbanking (16 banks)
Sense Amplifier	Alpha Latch Sense Amplifier 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

TABLE II
Leakage Power Reduction Techniques

Cache Circuit	Leakage Technique
Data Array Sense Amplifier	Stacked transistor [10]
Tag Array Sense Amplifier	Stacked transistor
SRAM Cell	Asymmetric SRAM Cell [8] & Stacked transistor
CAM Cell	Stacked transistor
Wordline Gating	High-V_{th} Transistor [9]

Table I shows the cache design with low power optimization techniques added. Table II shows the leakage reduction techniques. The cache was designed using 32-nm PTM device model 1 [1] and simulated with a supply voltage of 0.9 V and a clock of 1 GHz. The nominal value used for $V_{\rm th}$ is 0.2 V and the nominal value for $L_{\rm eff}$ is 25.3-nm. A 5-pi wire model is used. The wire parameters were obtained from [1].

The simulations were done in HSPICE circuit simulator using the BSIM4.0 model at temperature equal to 75 °C. The dynamic power is measured for a single word cache read.

Fig. 2 summarizes the delay and power consumption in this cache before and after applying all the low power techniques to the initial cache design.

III. Impact of Process Variation

In this section, we analyze the impact of different sources of process variations in caches under worst-case and expected behavior operating conditions. The goal here is to establish a worst-case baseline that might be used in conventional conservative designs and a typical behavior that could be used to estimate the benefits of migrating to an adaptive design.

In order to evaluate the impact of the parameter variations on circuit speed, we consider variability on the critical path. Indeed, the frequency (f), at which a circuit can be operated, is determined by the slowest path delay [13]. The critical path of our CAM-tag cache is shown in Fig. 3. A CAM-based cache stores tags in a CAM array and stores data in SRAM arrays. CAMs

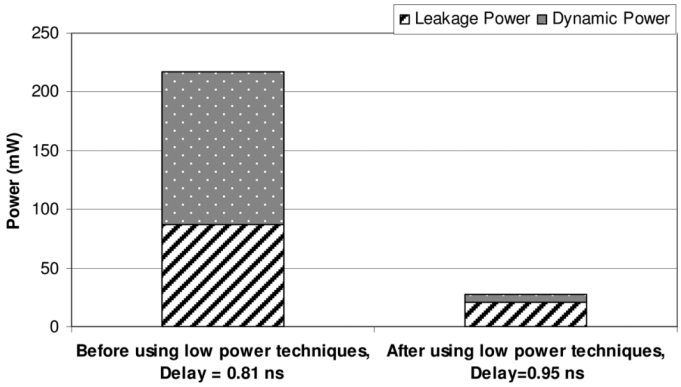

Fig. 2. Total cache power and delay.

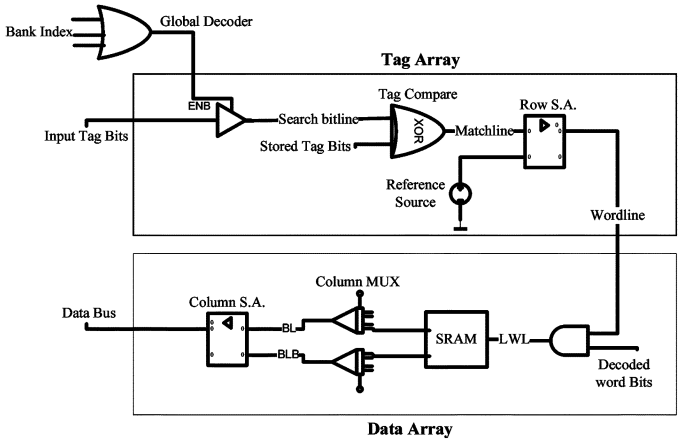

Fig. 3. Critical path of a CAM-tag cache.

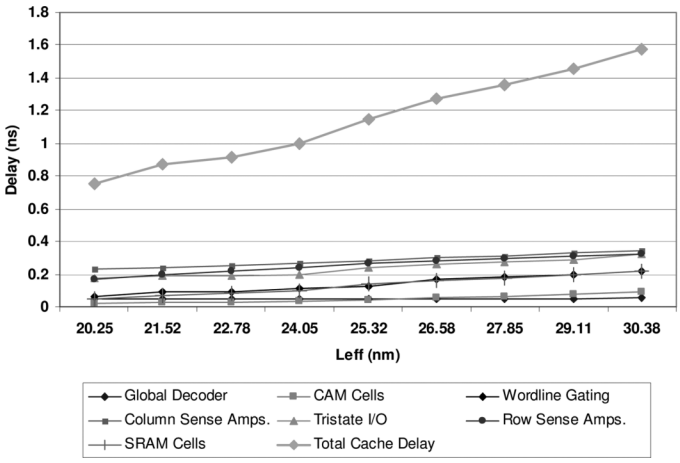

Fig. 4. Effect of $L_{\rm eff}$ variation on cache delay.

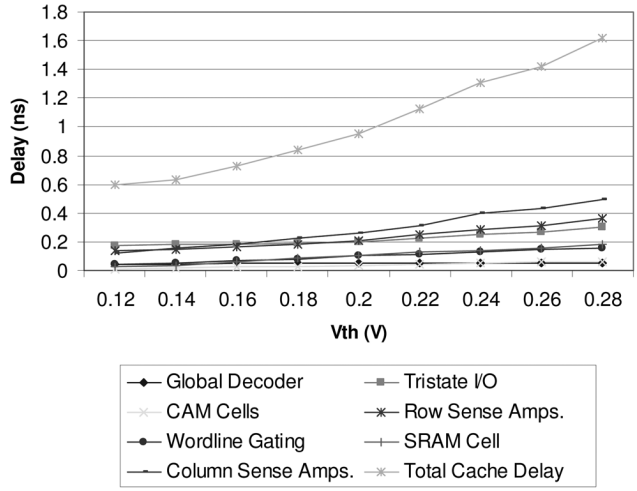

Fig. 5. Effect of $V_{\rm th}$ variation on cache delay.

perform tag checks in parallel. If a match is found, the cache provides the data in the associated cache line to the processor. Otherwise, a cache line replacement takes place.

The CAM-Tag critical path in our design is composed of the global address decoder to select a bank, tristate input/output (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.

A. Worst-Case Conditions

Under worst-case operating conditions, we assume that parameter variations happen at each transistor in the cache critical path.

1) Channel Length Variation: Effective channel length $(L_{\rm eff})$ variation 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 is expected within a die [3]. We have found that the use of longer effective channel lengths tends to increase the input capacitance of the gates associate with the wordline and bitline in caches, thus increasing access time as shown in Fig. 4. The access time can vary by as much as $2.1\times$.

2) Threshold Voltage Variation: Threshold voltage can vary due to: 1) changes in oxide thickness; 2) changes in the dopant levels in the substrate, poly-silicon and implants; and 3) surface charge. Accurate control of $V_{\rm th}$ is very important for many performance and power optimizations and for correct execution [6]. Higher transistor threshold voltage $V_{\rm th}$, due to process variations, affects the access time due to the lower read current as shown in Fig. 5. The impact on the access time could be as much as $2.7\times$.

3) Supply Voltage Variation: One of the most important environmental factors that cause variations in operating condition is supply voltage $(V_{\rm dd})$. In deep submicrometer technology, the supply voltage is typically scaled down to reduce power consumption; effects such as the current–resistance (*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.

TABLE III
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

TABLE IV
PARAMETER VALUES AND $\pm 3\sigma$ 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%)	
Temperature	75°C	

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

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 worst-case access delay could be impacted by around 2–3× (compared to the nominal value) when counting all the possible process parameters.

B. Expected Conditions

The simple use of worst-case values for all parameters that have been shown in Section A can result in larger path delay estimates than typical. These will certainly be pessimistic but would need to be considered in conventional designs. This section presents the delay distribution in a cache due process variation.

To accurately predict cache 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 $L_{\rm eff}$, $V_{\rm th}$, and $V_{\rm dd}$ variations, Monte Carlo simulations verify model predictions over a wide range of process and design conditions. We have used the Monte Carlo simulation with 5000 trials where the variation sources all vary simultaneously. We simulate the critical path and measure delay with all the parameters varying with 3ó and mean values as specified in Table IV.

The probability density function (PDF) of the cache delay was measured (see Fig. 6) for each process parameter. In addition, we have run the simulation with all the parameters vary simultaneously in another experiment. We have found most the

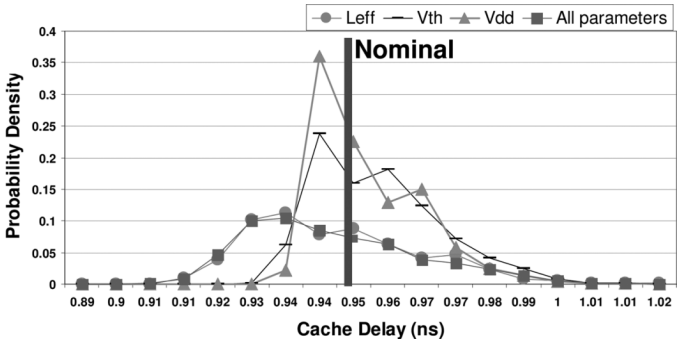

Fig. 6. Distribution of the cache access latency.

cache accesses under the impact of supply voltage or threshold voltage parameters would be relatively close to the nominal delay. The deviations in $L_{\rm eff}$ 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.

Out of 5000 random samples, assuming a 1 cycle cache at 1 GHz, 2000 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 two 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 two cycles has been found to be only 25%. We have also found that even in this case a small fraction of accesses would fail suggesting that there are cases that would need three cycles for correctness.

IV. 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 correct operations 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.

A. Conservative Cache

As we have shown in Section III, process variations affect the latency significantly for each cache access. The cache access latency difference could be as much as 3× 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 III). 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 one cycle, for the remaining 10%, we might need two

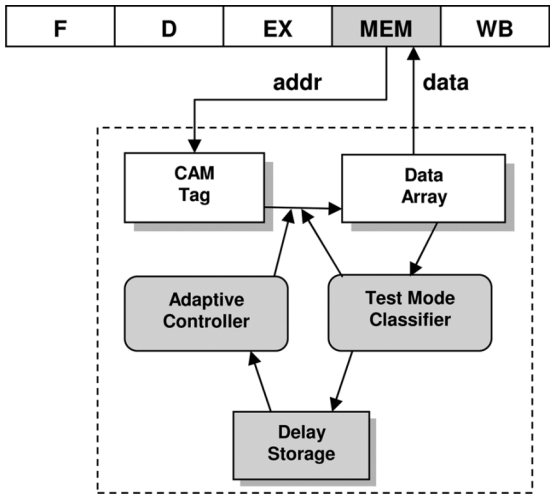

Fig. 7. Proposed adaptive cache architecture (shown in a single five-stage pipeline).

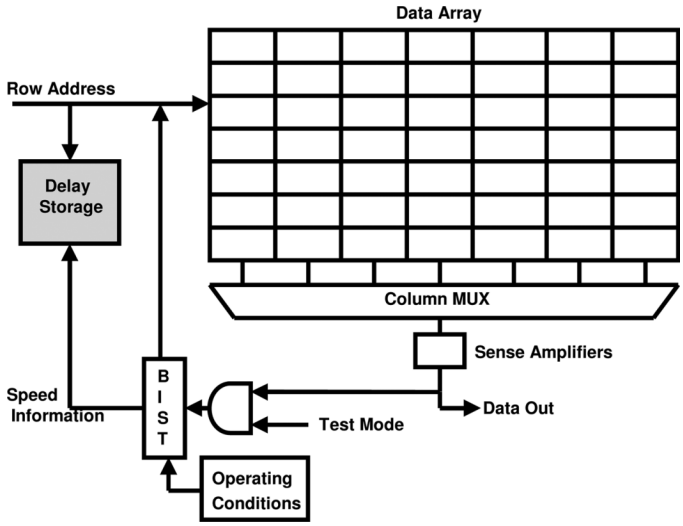

Fig. 8. Adaptive cache architecture during classification phase.

or three 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 use even if some architectural tricks could be applied to hide the cost (e.g., overlapping memory access with other instructions).

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.

B. Proposed Adaptive Cache

In the proposed adaptive cache design, instead of accessing the cache with a 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.

Fig. 7 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.

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.

C. Mechanism and Implementation

Fig. 8 shows the proposed delay-resilient cache architecture during the classification phase. The cache is equipped with a BIST circuitry, which tests the entire cache and detects the speed of each cache line during the classification phase.

The basic idea is to let the sense amplifiers to sample the bitlines during read operation in three stages. In the first stage, the sense amplifier will sense the bitlines in one clock cycle. While in the second stage, the sense amplifier will take two cycles to sense the bitlines, and in the third stage, the sense amplifier has to be fired in three cycles.

During the classification phase, if a cache component is affected by process variations (for example, higher $V_{\rm th}$ in SRAM cell), the SRAM cell might not be able to establish enough voltage differentials between bitlines by the end of the first cycle, thus, the access delay is more than one cycle.

In the second stage, the sense amplifier is fired in two cycles. If the sense amplifier still cannot get the value that has been previously stored in the SRAM cell, it means the delay is more than two cycles. In the third stage, the sense amplifier will be fired in the three cycles. Therefore, comparing the outputs of each sense amplifier with the original value stored, we can identify the actual cache access latency for this cache line. With this technique, we can detect a maximum delay of three cycles (a failure will occur if the actual latency is more than three cycles). This delay information will be stored in the delay storage unit.

Each cache line is tested using BIST when the test mode signal is on. For example, a block is considered fast, medium, or slow. BIST feeds this information into the delay storage.

The delay storage is implemented as a small memory array of two bits per cache line. Therefore, each row in the delay storage stores the speed information of all the SRAM cells in each cache line. For example, the two bits are "11" if the corresponding cache line latency is three cycles; "10" if the corresponding latency is two cycles, or it is "01" for one cycle ("00" can be also used to indicated a failure). These bits are determined at the time of testing and stored by BIST circuit.

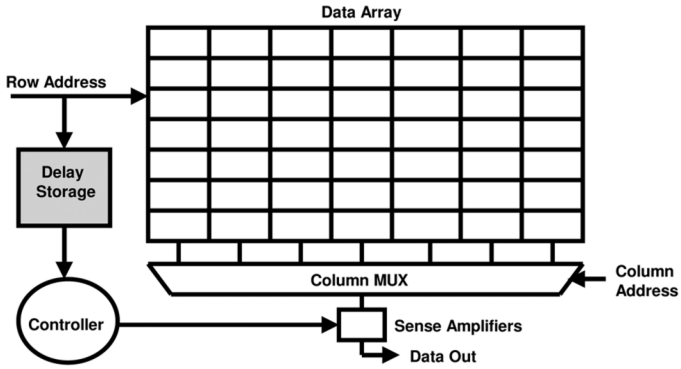

Fig. 9. Adaptive cache architecture during the execution phase.

Delay storage provides the speed information to the controller. The speed information stored in the delay storage is used by the controller to control the speed of the sense amplifiers during regular read operations. We can build the controller using combinational circuits that will provide the enable signals to the sense amplifiers at different speed.

During the execution phase (see Fig. 9), both the data array and delay storage units are accessed in parallel using the row address of the index bits. The delay information is fetched from the delay storage and fed to the controller. If the cache line is slow, the controller will delay the enable signals to the sense amplifiers. The sense amplifiers would need to be triggered at different time points depending on the speed access.

This scheme is transparent to other subsystem and has negligible power and area overhead. The area penalty for this cache is 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.

V. Results and Analysis

In the previous sections, we presented the power and the delay as well as the process variation related implications in our low power cache. This section provides an analysis of the new added components to our adaptive cache. In our analysis, we have evaluated the area overhead associated with the extra BIST, delay storage, and control circuitry by using the Synopsys Design Compiler tool to generate the netlist for the extra hardware needed for the adaptive cache (see Fig. 10).

We have found the overall area overhead to be less than 1% of the total cache area (see Table V). Because the delay storage is a small structure (two bits per cache line), and not on the cache critical path, its own delay variation due to process variation is small compared to the cache. Since the data array and delay storage units are accessed in parallel during the execution phase using the row address of the index bits, there will be no delay overhead by introducing the delay storage. In addition, we have evaluated the power consumption overhead associated with this extra circuitry; we have found the overhead to be 2% of the total cache power for a single word read.

A. Performance Speedup

The adaptive cache architecture is implemented in the SimpleScalar architecture simulator [2]. Simulation parameters are summarized in Table VI. 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

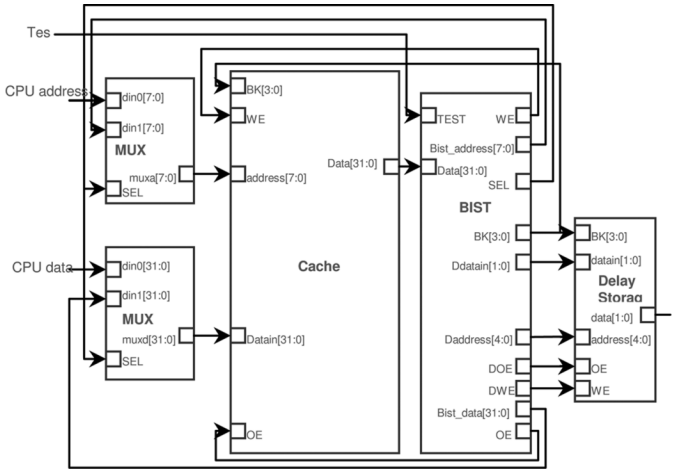

Fig. 10. BIST circuit.

TABLE V
Overhead Associate With the Adaptive Design

Circuit	BIST, delay storage, and control circuitry	Cache
Delay	No delay overhead	0.95 ns
Power	0.55 mW	27.67 mW
Area	0.0048 mm^2	0.540 mm^2

TABLE VI
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 associative, 32-byte per block, 2 cycles
L1 I-cache Size	16KB, 32-way associative, 32-byte per block, 2 cycles
L2 Unified Cache Size	128KB, 64-way, 64-byte per block, 8 cycles
Memory Latency	100 cycles
Memory Ports	2
TLB Size	128-entry, fully associative, 30-cycle miss penalty
Branch Predictor	Combination 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 associative
Return-address-stack	8-entry

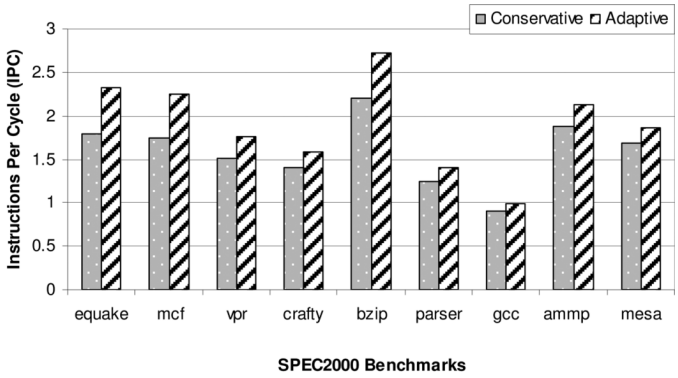

Fig. 11. Performance improvement between the adaptive cache versus a conservative cache using three-cycle access time.

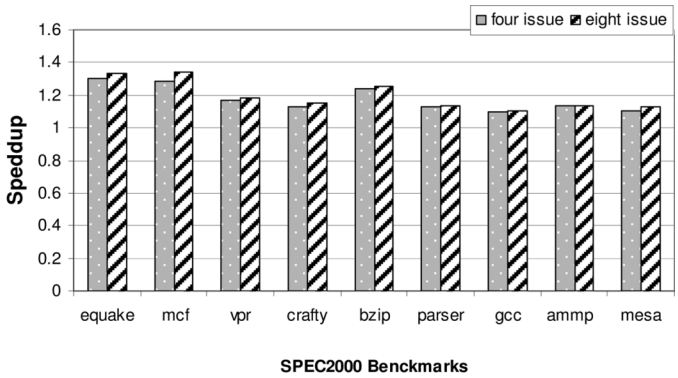

Fig. 12. Comparison of performance speedup for 16-kB cache on four-way issue and eight-way issue SimpleScalar architecture.

simulation. Based on our analysis, the adaptive cache is expected to have 75% of 1 cycle, 25% of 2 cycles, and negligible 3 cycle's cache line accesses.

Results on application performance are shown in Fig. 11. The comparison is made between a conservative cache that requires three cycles per access and an adaptive cache that has variable cache access latencies.

Our results show that the adaptive cache design in a four-way issue processor can achieve a 9% to 31% performance improvement on the applications studied, while providing resilience against failures due to process variations comparison is made to a conservative design assuming worst-case latency.

B. Sensitivity to Issue Width

Fig. 12 shows the performance speedup improvement for different instruction issue widths on several SPEC2000 applications. Speedup values are normalized with respect to the worst-case delay of three cycles. As we can see, the eight-way issues design benefits more than the four-way issues from the adaptive cache architecture.

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 [8] with some high $V_{\rm th}$ transistors to reduce cell power) would mean a redistribution between 1–3 cycle accesses.

VI. 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 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 (2–3×) 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 31% depending on the application and configurations used, compared to a conservative design. This scheme is transparent to other subsystems and has negligible power and area overhead. The adaptive cache architecture also allows a designer to choose the first-level 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.

References

[1] Nanoscale Integration and Modeling Group, ASU, Tempe, AZ, "Predictive technology model," 2008. [Online]. Available: http://www.eas.asu.edu/~ ptm

[2] The Standard Performance Evaluation Corporation, Warrenton, VA, "SPEC homepage," 2000. [Online]. Available: 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*. Piscataway, NJ: IEEE Press, 2001, ch. 6, pp. 98–115.

[4] K. Bowman, X. Tang, J. Eble, and J. Menldl, "Impact of extrinsic and intrinsic parameter fluctuations on CMOS circuit performance," *IEEE J. Solid- State Circuits*, vol. 35, no. 8, pp. 1186–1193, Aug. 2000.

[5] D. C. Burger and T. M. Austin, The SimpleScalar Tool Set, version 2.0 Univ. Wisconsin, Madison, Tech. Rep. CS-TR-1997–1342, 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. (VLSI) Syst.*, vol. 5, no. 4, pp. 369–376, Dec. 1997.

[7] M. Bennaser, Y. Guo, and C. A. Moritz, "Designing memory subsystems resilient to process variations," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Porto Alegre, Brazil, Mar. 2007, pp. 357–363.

[8] N. Azizi, A. Moshovos, and N. Farid, "Low-leakage asymmetric-cell SRAM," in *Proc. Int. Symp. Low Power Electron. Des.*, 2002, pp. 48–51.

[9] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada, "1-V power supply high-speed digital circuit technology with multithreshold-voltage CMOS," *IEEE J. Solid-State Circuits*, vol. 30, no. 8, pp. 847–854, Aug. 1995.

[10] Y. Ye, S. Borkar, and V. De, "A new technique for standby leakage reduction in high-performance circuits," in *Symp. VLSI Circuits Dig. Techn. Papers*, Jun. 1998, pp. 40–41.

[11] D. Burnett, K. Erington, C. Subramanian, and K. Baker, "Implications of fundamental threshold voltage variations for high-density SRAM and logic circuits," in *Proc. Symp. VLSI Technol.*, Jun. 1994, pp. 14–15.

[12] E. Humenay, D. Tarjan, and K. Skadron, "Impact of parameter variations on multi-core chips," presented at the Workshop Arch. Support Gigascale Integr., Boston, MA, Jun. 2006.

[13] S. Hao, K. Kim, and Y. H. Kim, "Critical path analysis considering the signal transition time," in *Proc. Int. Conf. VLSI CAD*, Oct. 1999, pp. 37–40.

[14] M. Zhang and K. Asanovic, "Highly-associative caches for low-power processors," presented at the Koolchips Workshop, 33rd Int. Symp. Microarch., Monterey, CA, Dec. 2000.

[15] K. Schultz, "Content-addressable memory core cells: A survey," *Integr., VLSI J.*, vol. 23, no. 2, pp. 171–188, Nov. 1997.

[16] M. Orshansky and K. Keutzer, "A general probabilistic framework for worst case timing analysis," in *Proc. Des. Automat. Conf.*, Jun. 2002, pp. 556–561.

[17] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical timing analysis for intra-die process variations with spatial correlations," in *Proc. ICCAD*, Nov. 2003, pp. 900–907.
[18] A. Agarwal, B. Paul, H. Mahmoodi, A. Datta, and K. Roy, "A process-tolerant cache architecture for improved yield in nanoscale technologies," *IEEE Trans. Very Large Integr. (VLSI) Syst.*, vol. 13, no. 1, pp. 27–38, Jan. 2005.
[19] K. Bowman, S. Duvall, and J. Meindl, "Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration," *IEEE J. Solid State Electron.*, vol. 37, no. 2, pp. 183–190, Feb. 2002.

Mahmoud Bennaser (S'06) received the B.S. degree in computer engineering from Kuwait University, Khaldiya, Kuwait, in 1999, and the M.S. degree in computer engineering from Brown University, Providence, RI, in 2002. He is currently pursuing the Ph.D. degree in computer engineering from University of Massachusetts, Amherst.

His research interests include computer architecture, and low-power circuit design.

Mr. Bennaser was a recipient of the certificate of academic excellence for the years 1995–1998 and a scholarship from Kuwait University to pursue the M.S. and Ph.D. degrees in computer engineering in 2000.

Yao Guo (M'06) received the Ph.D. degree in computer engineering from the University of Massachusetts, Amherst, in 2007.

He is an Assistant Professor with the School of Electrical Engineering and Computer Science, Peking University, Peking, China. His research interests include operating systems, low-power design, compilers, embedded systems, and software engineering.

Csaba Andras Moritz (M'85) received the Ph.D. degree in computer systems from the Royal Institute of Technology, Stockholm, Sweden, in 1998.

He is an Associate Professor with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst. From 1997 to 2000, he was a Research Scientist with the Laboratory for Computer Science, Massachusetts Institute of Technology, Boston. He was a Consultant for several technology companies in Scandinavia and held industrial positions ranging from CEO, CTO, to founder. He founded and led BlueRISC, a low-power security microprocessor company. His research interests include computer architecture, compilers, low power design, and nanoscale systems.