Enabling Ultra Low Voltage System Operation by Tolerating On-Chip Cache Failures

Amin Ansari Shuguang Feng Shantanu Gupta Scott Mahlke

Advanced Computer Architecture Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI
{ansary, shoe, shangupt, mahlke}@umich.edu

ABSTRACT

Extreme technology integration in the sub-micron regime comes with a rapid rise in heat dissipation and power density for modern processors. Dynamic voltage scaling is a widely used technique to tackle this problem when high performance is not needed. However, the minimum achievable supply voltage is often bounded by SRAM cells since they fail at a faster rate than logic cells. In this work, we propose a novel fault-tolerant cache architecture, that by reconfiguring its internal organization can efficiently tolerate SRAM failures that arise when operating in the ultra low voltage region. Using our approach, the operational voltage of a processor can be reduced to $420mV$, which translates to 80% dynamic and 73% leakage power savings in $90nm$.

Categories and Subject Descriptors

B.3.4 [**Memory Structures**]: Reliability, Testing, and Fault-Tolerance

General Terms

Design, Reliability

Keywords

Dynamic voltage scaling, Fault-tolerant cache, Low voltage operation

1. INTRODUCTION

Power consumption and heat dissipation have become key challenges in the design of high performance processors. Growing power consumption reduces device lifetimes and also affects the cost of thermal packaging, cooling, and electricity [6]. Dynamic voltage scaling (DVS) is widely used to reduce the power consumption of microprocessors, exploiting the fact that the dynamic power quadratically scales with voltage and linearly with frequency. Consequently, lowering the minimum operational voltage of a microprocessor, can dramatically improve the energy consumption and battery life of medical devices, laptops, and handheld products.

The motivation for our work comes from the observation that large SRAM structures are limiting the extent to which operational voltages can be reduced in modern processors. This is because SRAM delay increases at a higher rate than CMOS logic delay as the supply voltage is decreased [13]. With increasing systematic and random process variation in deep sub-micron technologies, the failure rate of SRAM structures rapidly increases in the ultra low voltage regime.

ISLPED'09, August 19–21, 2009, San Francisco, California, USA.

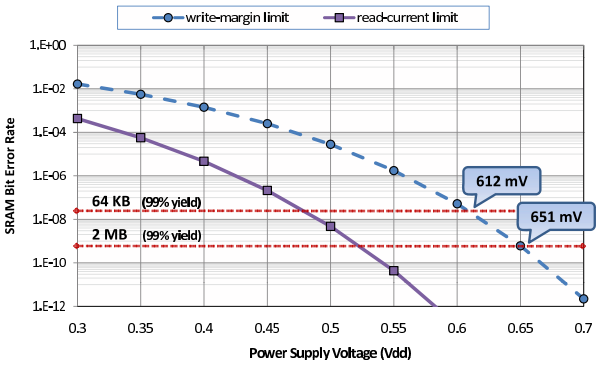

Figure 1: Bit error rate for an SRAM cell with varying V_{dd} values in 90nm. For this technology, the write-margin is the dominant factor and limits the operational voltage of the SRAM structure. Here, the Y-axis is logarithmic, highlighting the extremely fast growth in failure rate with decreasing V_{dd}. The two horizontal dotted-lines mark the failure rates at which the mentioned SRAM structures (64KB and 2MB) can operate with at least **99%** manufacturing yield.

Ultimately, the minimum sustainable V_{dd} of the entire cache structure is determined by the one SRAM bit within the entire system with the highest required operational voltage. This forces designers to utilize a large voltage margin in order to avoid on-chip cache failures. Figure 1 depicts the failure rate of an SRAM cell based on the operational voltage in a 90*nm* technology node [9]. The minimum operational voltage of L1 and L2 caches is selected to ensure a high expected yield, 99% in Figure 1. As can be seen, the write margin mostly dictates the minimum V_{dd} and it is expected to operate with $V_{dd} \geqslant 651mV$ due to the dominating failure rate of the 2MB L2 cache. This number is consistent with predicted and measured values ($\sim 0.7V$) reported in [2].

In the literature, several techniques have been proposed to improve dynamic and/or leakage power of on-chip caches [13]. The usage of V_{dd} gating for leakage power reduction by turning off cache lines is described in [4]. This approach reduces the leakage power of the cache by turning off the cache lines that are not likely to be accessed in the near future. A simultaneous usage of DVS and adaptive body biasing is presented in [7] for reducing power in high-performance processors. They derived a closed-form method for finding the optimal supply voltage and body bias for a given frequency and duration of operation. Meng et.al. [8] proposed a method for minimizing the leakage overhead considering manufacturing variations. In this scheme, they give an artificial priority to the cache ways with smaller leakage and re-size the cache by avoiding sub-arrays that have higher leakage factors. Instead of turning off blocks, drowsy cache [3] is a state preserving approach that has two different supply voltage modes. Recently inactive cache blocks periodically fall into the low power mode in which they cannot be read or written. However, for low V_{dd} values (e.g. $\leqslant 651mV$), the amount of power saving for these methods is restricted due to the failures in SRAM structures.

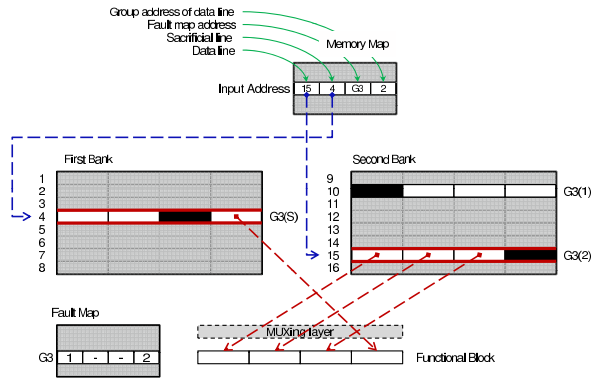

Figure 2: Basic structure of our proposed scheme with two cache banks and eight lines each. Each line consists of 4 equally sized data chunks. Black boxes in each cache line represent chunks that have at least have one faulty bit. The memory and fault maps, which are essential components of the proposed scheme, are also shown.

In contrast, the objective of our work is to enable DVS to push the core/processor operating voltage down to the ultra low voltage region (e.g. low power mode) while preserving correct functionality of on-chip caches. This idea was initially proposed in [11] and later Wilkerson et.al. [14] improved the architecture to enable operation at even lower voltages. On the other hand, many variations of SRAM cells such as 8T, 10T, 11T, and ST [5] have also been proposed which allow the SRAM structures to operate at lower voltages than the conventional 6T cell. Most of these cells have a large area overhead which is significant shortcoming since the extra area does not translate into any performance gains when operating in high power mode. Consequently, we try to minimize the overhead of normal high power mode operation.

In this work, we propose a fault-tolerant cache which intertwines a set of $n + 1$ partially functional cache lines together to give the appearance of n functional lines. The overhead of the approach is a small performance penalty ($< 5\%$) and less than 15% area overhead for the on-chip caches. We apply our scheme to L1-D, L1-I, and L2 caches to evaluate the achievable power reduction for a microprocessor.

2. ARCHITECTURE

In this section, we describe the architecture of our flexible fault-tolerant cache, one which allows our scheme to adaptively reconfigure itself to absorb failing SRAMs. As we discussed earlier, decreasing the operational V_{dd} (i.e., entering low-power mode) causes many cells within a cache to fail. For example, according to Figure 1, for $V_{dd} = 420mV$, the number of faulty bits in just one L2 block ($128B$) is as high as 5. Our scheme provides the appearance of a fully functional cache by tolerating these failures.

To this end, we partition the set of all cache word-lines into large groups, where one word-line (the sacrificial line) from each group is set aside to serve as the redundant word-line for the other word-lines in the same group. In the remainder of this paper, we refer to every cache *word-line*, which may contain multiple blocks as a line. In our approach, each line is divided into multiple data chunks. Each chunk is labeled faulty if it has at least one faulty bit. Two lines have a collision if they have at least one faulty chunk in the same position. For example, if the second data chunk of the 3*rd* and 6*th* lines is faulty, then lines 3 and 6 have a collision. Similarly, in Figure 2, lines 10 and 15 are collision-free. The objective of our scheme is to form groups such that there are no collisions between any two lines within a group. In Figure 2, lines 4, 10, and 15 form the 3*rd* group (*G3*) in the cache. Here, line 4 (labeled G3(S)) is the sacrificial line that furnishes the redundancy needed to accommodate the faulty chunks in lines 10 and 15. In order to minimize the access latency overhead, the sacrificial line (4) and the data lines (10 or 15) should

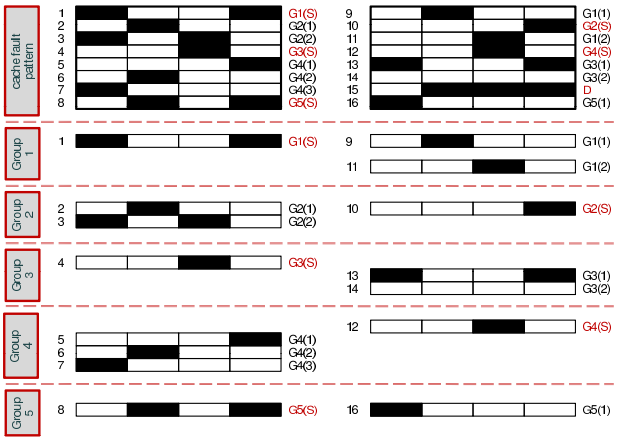

Figure 3: An example of configuration algorithm for a given distribution of faults in the cache banks. Here, each banks has only 8 lines and there are 21 faulty chunks in the cache. The configuration algorithm forms 5 groups in the cache and disables only line 15. Furthermore, these groups and their corresponding sacrificial and normal lines are also shown here.

be in different banks so that the sacrificial line can be accessed in parallel to the original data line.

In our fault-tolerant cache architecture, each cache access first indexes into a memory map, which supplies the location of the data line and its corresponding sacrificial line. After these two lines have been accessed from their respective banks, a MUXing layer is used to compose a fault free block by selecting the appropriate chunks from each line. This MUXing layer receives inputs indirectly from the *fault map*. For a given data line, the fault map determines which chunks are faulty and should be replaced with chunks from the sacrificial line.

To aid in the encoding and decoding of this information a unique address is assigned to all lines within a group (*group address*). For instance, in Figure 2, line 15 is the second line of *G3*. For each data chunk in the sacrificial line, the fault map stores the group address of the line to which that data chunk is assigned. Here, the entry which is assigned to *G3* in the fault map contains (1,-,-,2), indicating that the first chunk of *G3(S)* is devoted to *G3(1)*, the fourth chunk is dedicated to *G3(2)*, and the second and third chunks are not assigned to any line. Finally, the MUXing layer gets its input from a set of comparators that compare the group address of line 15 (e.g., 2, read from memory map) with *G3*'s fault map entries.

Since every group requires a sacrificial line be dedicated solely for redundancy, our scheme strives to minimize the total number of groups that must be formed. Given that the number of lines is fixed within a cache, achieving this objective implies that larger groups are preferred over smaller ones. To maximize the number of functional lines in the cache, we need to minimize the number of sacrificial lines required to enable fault-free operation. As previously discussed, there is a single sacrificial line devoted to every group of lines. This sacrificial line is not addressable as a data line since it does not store any independent data. In other words, sacrificial lines do not contribute to the usable capacity of the cache. Depending on the number of collision-free groups that are formed, the effective capacity of the cache can vary dramatically.

Figure 3 shows the process of forming the groups given a fault pattern for the cache. Group formation is an iterative process and in each iteration of the algorithm, a new group is formed. Here, each group consists of a sacrificial line from one bank and a set of collision-free lines from the other bank which are still not assigned to any other group. By assigning the largest possible set of collision-free lines to each sacrificial line, our algorithm tries to minimize the number of sacrificial lines required for fault-free operation. In order to form the groups, our configuration algorithm starts from the top

line of the first bank and marks it as a sacrificial line for the first group (i.e., *G1(S)*). Next, it switches to the second bank to find the largest set of collision-free lines which can potentially be assigned to the first group. For this purpose, it keeps adding the lines from the top of the second bank to this group and if a line causes a collision, the algorithm simply skips it. As a result, line 9 is added to the first group and marked as *G1(1)*. However, since line 10 has a collision with line 1, it cannot be added to this group and line 11 takes its place. After the first group is formed, we switch to the bank which has the most number of already assigned lines (i.e., second bank here) and mark the first unassigned line of this bank as the sacrificial line for the second group (i.e., line 10 (*G2(S)*)). In addition, lines 2 and 3 are assigned to this group from the first bank. This process continues until all the lines in both banks get either disabled or assigned to groups. A cache line gets disabled if: 1) it contains many faulty chunks which makes its repair unjustified or 2) it cannot be assigned to any of the existing groups with size greater than one due to its particular fault pattern. It should be noted that the back and forth switching between the banks allows our algorithm to minimize the number of lines getting disabled. Figure 3 presents the 5 groups formed by the configuration algorithm.

As depicted in Figure 3, line 15 is disabled (D) since it contains 3 faulty chunks and repairing it is not cost effective. Here, the number of non-functional lines is the summation of the number of sacrificial and disabled lines. The objective of the configuration algorithm is to minimize the number of non-functional lines for a given fault pattern in the cache. In Figure 3, there are 6 non-functional cache lines which consist of 5 sacrificial lines and one disabled line. For each cache instance, the number of lines in the fault map array is equal to the number of sacrificial lines (i.e. 5 here). However, due to the presence of process variation in a large population of the fabricated chips, different fault patterns should be expected. In our evaluation, we employ a Monte Carlo simulation to generate a population of 1000 cache instances and the total number of fault map lines is determined based on the maximum number of sacrificial lines while achieving a 99% yield.

Low Power Mode Operation: The first time a processor switches to low power mode, the built-in self test (BIST) module scans the cache for the potential faulty cells. After determining the faulty chunks of cache lines, the processor switches back to the high power mode and forms the groups as described before. This provides the information that is required to be stored in the memory and the fault maps. This configuration information can be stored on the hard-drive, then is written to the memory map and fault map at system boot-up time. In addition, the memory map, fault map and the tag arrays are protected using the well studied 10T cell [2] which has about 66% area overhead for these relatively small structures. These 10T cells are able to meet the target voltage in this work ($420mV$) without failing. However, these cannot be used for the protection of the large SRAM structures (e.g., L2 data array) since that will impose a much higher overhead [14].

High Power Mode Operation: In high power mode, our scheme is turned off in order to minimize the unwanted overheads: **1)** All the cache lines are functional and there is no sacrifice of the cache capacity. **2)** There is a negligible overhead for the dynamic power due to the switching in the bypass MUXes which consists of the MUXing layer and an additional MUX which can bypass the memory map. **3)** Leakage power overhead remains the same. However, power gating techniques can be used for leakage mitigation.

3. EVALUATION

This section evaluates the effectiveness of our fault-tolerant cache architecture in reducing the power of a processor while keeping the overheads low.

3.1 Methodology

For performance evaluation, we use SimAlpha, a validated micro-architectural simulator based on the SimpleScalar Out-of-Order simulator [1]. The processor is configured as shown in Table 1 and is modeled after the DEC EV-7. CACTI is leveraged to evaluate the delay, power, and area of the SRAM structures [10]. Lastly, the Synopsys standard tool-chain is used to evaluate the overheads of the remaining miscellaneous logic (i.e., bypass MUXes, comparators, etc.).

For a given set of cache parameters (e.g. V_{dd}, chunk size, etc.), Monte Carlo simulations (1000 iterations) are performed using the configuration algorithm described in Section 2 to identify the portion of the cache that should be disabled. Solutions generated by our configuration algorithm target a 99% yield. In other words, only 1% of manufactured and configured on-chip caches are allowed to exhibit failures when operating in low-power mode.

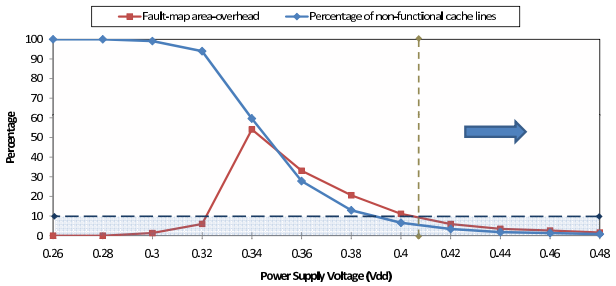

Figure 4: Process of determining the minimum achievable V_{dd} for L2 with 4-*bit* chunk size. The fraction of the non-functional cache lines and also the area overhead of the fault map structure are limited to $\leqslant 10\%$.

3.2 Results

Figure 4 shows the process of determining the minimum achievable V_{dd} for a system. Since protecting the L2 is harder than the L1 (due to its longer lines and larger size [14]), L2 protection cost dictates the minimum operating voltage of a system. In order to evaluate our scheme, we set chunk size to be 4 *bits* for L2 and 8 *bits* for L1 which is easier to protect.

In high-power mode, both fault and memory map arrays remain idle and leak power. It is crucial to minimize the size of these structures. The size of the memory map is essentially fixed by the number of lines in the cache. The fault map size, however, can vary significantly depending on configuration parameters, motivating a closer look at the size of the fault map as an important design factor. Consequently, we limit the area overhead of the fault map to 10% of the total cache area. Furthermore, since cache size has a strong correlation with system performance, we limit our scheme to disable at most 10% of the cache lines.

As evident in Figure 4, decreasing V_{dd} increases the non-functional portion of the cache and also the area of the fault map array. However, beyond a certain point, the area overhead of the fault map starts decreasing. This phenomena is due to the large fraction of the cache lines that get **disabled** as lowering V_{dd} leads to increasing error rates and a precipitous increase in faulty chunks. Here, the vertical line

Table 1: The target system configuration.

Parameters	Value
Technology	90 nm
Clock frequency	1.9 GHz
V_{dd} nominal	1.2 V
L1 Cache	2 banks 64KB data, 2 banks 64KB instruction, split, 2-way set associative, 4 cycles hit latency, 1 port, LRU, 64B block size, write-back
L2 Cache	2 banks 2MB Unified, 8-way set associative, 10 cycles hit latency, 1 port, LRU, 128B block size, write-back
Registers	80 integer, 72 floating point
ROB (re-ordering buffer)	128 entries
LSQ (load/store queue)	64 entries
Instruction fetch buffer	32 instructions
Integer/FP issue queue	32/32 entries
FU (functional unit)	4 int ALU, 4 int mult/div, 2 memory system ports
FPU (floating point unit)	4 FP ALU, 1 FP mult/div
Main memory	225 cycles (high power), 34 cycles (low power)
Branch predictor	combined (bimodal and 2-level)
BHT (branch history table)	4096 entries
RAS (return address stack)	32 entries
BTB (branch target buffer)	2048 entries, 2-way associative

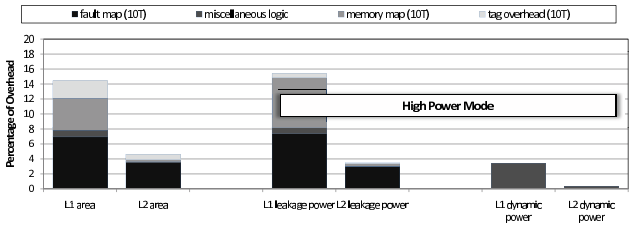

Figure 5: Overheads of our scheme for both L1 and L2 caches. Here, 10T cell is used for protecting fault map, memory map, and tag arrays. Note the area and leakage power overhead of the system are mostly determined based on L2 overheads. This is due to the significantly larger size of L2 for which our scheme has minimal overheads.

highlights the minimum achievable V_{dd} based on the aforementioned 10% limit on disabled lines. As a result, we select $420mV$ as the minimum V_{dd} (i.e., low-power mode operating voltage). All lower voltages violate our 10% limits.

Figure 5 summarizes the overheads of our scheme for both L1 and L2. Leakage overhead in high power mode corresponds to the fault map, memory map, and miscellaneous logic. As mentioned before, we also account for the overheads of using 10T SRAM cells [2] for protecting the tag, fault map, and memory map arrays in low-power mode. Note, the memory map is a far greater contributor to area and leakage power overhead in the L1 than in the L2. The reason behind this is that the L1 has only $\frac{1}{4}$ the lines of L2 while its overall size is $\frac{1}{32}$. For L2, the fault map is the major component of overhead. Due to its large size, and attendant leakage and area overheads, the L2 dominates the processor overheads, warranting the close study of the L2 fault map in Figure 4. Dynamic power overhead in high-power mode can be mainly attributed to bypass MUXes since we assume clock gating for the fault map and memory map arrays. In our proposed scheme, when in low-power mode, the memory map and MUXing layer are in the critical path of the cache access. Based on our timing analysis, this translates to 1 additional cycle latency for L1 and 2 additional cycles for L2 in low-power mode.

In order to evaluate the performance penalty of our scheme in low-power mode, we ran the SPEC2K benchmark suite on SimAlpha after fast-forwarding to an early SimPoint [12]. We assume one extra cycle latency for L1 and 2 extra cycles for L2. Cache size is also reduced based on the fraction of the non-functional lines for L1/L2 caches. On average, a 4.7% performance penalty is seen in low-power mode (Figure 6) from which 1.1% is due to the cache capacity loss. However, one should note that low-power mode performance is usually not a major concern. In high power mode, there is enough slack on the access time of our L1 and L2 caches (CACTI results) to fit the small bypass MUXes (additional $0.07ns$ delay) without adding any extra cycles to the access time. In other words, there is no performance loss in high-power mode. However, one might have a cache design without any slack available. In that scenario, we add an additional cycle for L1 and L2 which translates into a 3.6% performance drop off.

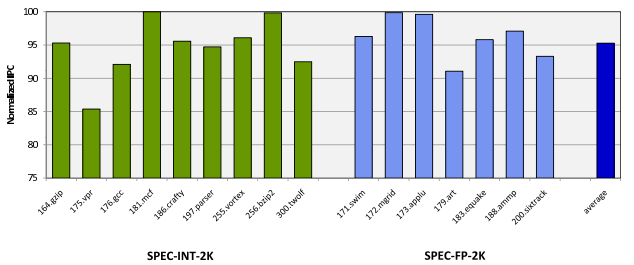

Figure 6: Amount of performance drop-off for our scheme in low power mode using the SPEC-2K benchmarks.

Lastly, we evaluate the power savings that can be achieved using our scheme for the microprocessor. Based on a similar assumption in [14], we assume dynamic power scales quadratically with Vdd and linearly with frequency. Furthermore, leakage power scales with the cube of Vdd. As a result, our scheme allows DVS to potentially save 80% dynamic power and 73% leakage power for the microprocessor.

4. CONCLUSION

With aggressive CMOS scaling, dealing with power dissipation has become a challenging design issue. Consequently, a large amount of effort has been devoted to the development of dynamic voltage scaling methods to tackle this problem. When decreasing the operational voltage of a modern microprocessor, large on-chip cache structures are the first components to fail. Tolerating these SRAM failures, allows DVS to target lower V_{dd} values while preserving the core frequency scaling trend. In this work, we proposed a flexible fault-tolerant cache architecture which allows DVS to achieve $420mV$ in $90nm$. This translates to 80% dynamic and 73% leakage power savings for our target system. This significant amount of saving comes with 4.7% performance overhead for the microprocessor and less than 15% area overhead for the on-chip caches.

5. ACKNOWLEDGMENTS

This research was supported by ARM Ltd., the National Science Foundation grant CCF-0347411, and the Gigascale Systems Research Center, one of five research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation program.

6. REFERENCES

[1] T. Austin, E. Larson, and D. Ernst. Simplescalar: An infrastructure for computer system modeling. *IEEE Transactions on Computers*, 35(2):59–67, Feb. 2002.

[2] B. Calhoun and A. Chandrakasan. A 256kb sub-threshold sram in 65nm cmos. *2008 IEEE International Solid-State Circuits Conference*, pages 2592–2601, Feb. 2006.

[3] K. Flautner, N. Kim, S. Martin, D. Blaauw, and T. Mudge. Drowsy caches: simple techniques for reducing leakage power. *Proc. of the 29th Annual International Symposium on Computer Architecture*, pages 148–157, 2002.

[4] S. Kaxiras, H. Zhigang, and M. Martonosi. Cache decay: exploiting generational behavior to reduce cache leakage power. *Proc. of the 28th Annual International Symposium on Computer Architecture*, pages 240–251, 2001.

[5] J. P. Kulkarni, K. Kim, and K. Roy. A 160 mv, fully differential, robust schmitt trigger based sub-threshold sram. In *Proc. of the 2007 International Symposium on Low Power Electronics and Design*, pages 171–176, New York, NY, USA, 2007. ACM.

[6] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen. Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction. In *Proc. of the 36th Annual International Symposium on Microarchitecture*, pages 81–92, Dec. 2003.

[7] S. Martin, K. Flautner, T. Mudge, and D. Blaauw. Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. In *Proc. of the 2002 International Conference on Computer Aided Design*, pages 721–725, New York, NY, USA, 2002. ACM.

[8] K. Meng and R. Joseph. Process variation aware cache leakage management. *Proc. of the 2006 International Symposium on Low Power Electronics and Design*, pages 262–267, Oct. 2006.

[9] Y. Morita, H. Fujiwara, H. Noguchi, Y. Iguchi, K. Nii, H. Kawaguchi, and M. Yoshimoto. An area-conscious low-voltage-oriented 8t-sram design under dvs environment. *IEEE Symposium on VLSI Circuits*, pages 256–257, June 2007.

[10] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi. Optimizing nuca organizations and wiring alternatives for large caches with cacti 6.0. In *IEEE Micro*, pages 3–14, 2007.

[11] D. Roberts, N. S. Kim, and T. Mudge. On-chip cache device scaling limits and effective fault repair techniques in future nanoscale technology. *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools*, pages 570–578, Aug. 2007.

[12] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *Tenth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 45–57, New York, NY, USA, 2002. ACM.

[13] K. Takeda, Y. Hagihara, Y. Aimoto, M. Nomura, Y. Nakazawa, T. Ishii, and H. Kobatake. A read-static-noise-margin-free sram cell for low-vdd and high-speed applications. *2006 IEEE International Solid-State Circuits Conference*, 41(1):113–121, Jan. 2006.

[14] C. Wilkerson, H. Gao, A. R. Alameldeen, Z. Chishti, M. Khellah, and S.-L. Lu. Trading off cache capacity for reliability to enable low voltage operation. *Proc. of the 35th Annual International Symposium on Computer Architecture*, 0:203–214, 2008.