Minimizing Total Area of Low-Voltage SRAM Arrays through Joint Optimization of Cell Size, Redundancy, and ECC

Shi-Ting Zhou, Sumeet Katariya, Hamid Ghasemi, Stark Draper, and Nam Sung Kim
The University of Wisconsin-Madison, WI 53706, U.S.A.

Abstract—The increasing power consumption of processors has made power reduction a first-order priority in their design. Voltage scaling is one of the most successful power-reduction techniques introduced to date, but it is limited to some minimum voltage, VDDMIN, below which all components cannot operate reliably. In particular, ever-increasing process variability due to shrinking feature size further degrades the low-voltage reliability of, e.g., SRAM cells. Larger SRAM cells are less sensitive to process variability and their use would allow a reduction in VDDMIN. However, large-scale memory structures, e.g., last-level caches (LLCs) that often determine the VDDMIN of processors, cannot afford to use such large SRAM cells due to the resulting increase in die area. In this paper we propose a joint optimization of LLC cell size, number of redundant cells, and ECC (error-correction coding) strength to minimize total SRAM area while meeting target yields and VDDMIN. The use of redundant cells and ECC enable the use of smaller cell sizes while maintaining target yields and VDDMIN. Smaller cell sizes more than make up for the extra cells required by redundancy and ECC. We first assess each approach individually, i.e., only redundancy or ECC for various cell sizes. We then consider a combined approach and observe significant improvements. For example, in 32nm technology our combined approach yields a 27% reduction in total SRAM area (including redundant cells) when targeting a VDDMIN of 600mV.

I. INTRODUCTION

As technology scaling allows us to integrate more and more devices the power consumption of both high-performance and low-power processors has become a most critical design priority. For example, the performance of high-performance processors is often limited by their maximum power consumption, which is determined by their power supply and cooling capacity. As a second example, the operating time of mobile, battery-powered processors is determined by their power efficiency.

Voltage scaling has been one of the most effective techniques used to minimize power consumption of such processors. However, ever-increasing process variability in nanoscale technology such as *random dopant fluctuation* (RDF) limits voltage scaling of on-chip memory to some minimum operating voltage, VDDMIN. At lower voltages a processor cannot operate reliably. As a result, the failure probability of individual memory cells in large-scale structures such as on-chip caches often determines the VDDMIN for the whole processor.

Many circuit-level techniques have been proposed to lower VDDMIN for on-chip caches based on conventional six-transistor (6T) SRAM cells [1]-[4]. These techniques either apply different supply voltages [1] or use special assisting circuits [2]-[4] for read and write operations to reduce the failure probability of SRAM cells. These techniques reduced the failure probability by orders of magnitude, but at the cost of additional specialized circuitry and additional design complexity. In a separate approach, micro-architecture techniques have been proposed to cope with the VDDMIN challenge [5]-[7]. In effect, to support lower VDDMIN, they either identify defective cells and disable the entire cache line that contains those cells [5][7], or implement incrementally stronger ECC at the expense of reduced cache capacity as voltage is lowered [6][7]. A final approach to lower VDDMIN is to adopt a SRAM cell design that differs from the traditional 6T design. For example, design variants such as the 8T, 10T, and ST SRAM cells have been used [8]. However, the VDDMIN reduction resulting from the use of such alternate SRAM designs comes at the cost of significant increases in SRAM cell area (e.g., 100% increases for the ST cell).

One standard technique we leverage is the use of redundant columns of SRAM cells. Originally this technique was introduced to improve manufacturing yields of SRAM arrays. A small number of extra (redundant) columns of SRAM cells are available to replace a column of cells that contains a cell (or cells) with manufacturing faults (e.g., stuck-at-faults) [9]. Following manufacturing, a test for bad cells is performed. The SRAM array is then reconfigured to access a redundant column instead of the column containing the defective cell(s). In addition, *error-correction coding* (ECC) was adopted to protect SRAM arrays from particle-induced soft errors. Any single cell error induced by an impacting particle can be corrected using the redundant information provided by the ECC applied to each cache line. A new failure mechanism (in addition to stuck-at and particle-induced errors) arises at low operating voltages, where SRAMs can fail because process variability for small-scale SRAM structures can be quite significant. Thus, redundancy and ECC are now also used to correct low-voltage SRAM cells failures, thereby allowing a lower VDDMIN.

Our approach in this paper is to combine the proven design techniques discussed above in a novel hybrid manner that delivers quantitatively different effectiveness from those achieved by any single technique. First, for 6T SRAM cells we explore the change in single-cell failure probability as we vary the size of the constituent transistors. Increased transistor size reduces the amount of RDF-induced *threshold voltage* (VTH) variations that are the main source of SRAM failures at low VDD [14]. On its own, this leads to lower Vddmin, but (as with ST cells) also a significant increase in total SRAM area. Thus, we apply both redundancy and ECC techniques to fix failing cells at low VDD. Naturally, more redundancy and stronger ECC are often required as we lower the target VDDMIN. Substantial area penalties are thereby incurred due to extra redundant columns and check bits. In one way or another we must pay for a lower VDDMIN through increased area. The focus of this paper is the minimization of the total area of SRAM arrays used to implement large on-chip caches, while providing a low enough VDDMIN for ultra-low-power operations.

The followings are the specific contributions of this paper. First, we investigate the impact of each technique, i.e., transistor sizing, redundancy, and ECC on both the overall SRAM area and VDDMIN through analytical and experimental models and results. Second, we minimize the overall SRAM area by jointly optimizing the size of transistors, the degree of redundancy, and the strength of ECC while meeting target VDDMIN and manufacturing yields. Stronger ECC requires not only more check bits (and thus area) per cache line, but also more encoding/decoding time, which increases cache access latency and power consumption. To validate our designs we evaluate the performance of our designs on multi-core processors using a micro-architecture simulator running commercial server workloads that can aggressively exercises large on-chip caches.

To the best of our knowledge, this is the first study that (a) analyzes the individual and combined impact of cell size, redundancy, and ECC on VDDMIN and total area through analytical and experimental models, and (b) minimizes the total die area of large on-chip caches by applying the three techniques jointly. Prior

978-1-4244-8935-0/10/$26.00 ©2010 IEEE

studies of V_{DDMIN} reduction through the use of redundancy, ECC, or other fault-tolerance techniques, did not take advantage of the trade-off between SRAM cell size and failure probability. As we show, a joint optimization including transistor sizing can impact the overall failure probability of large on-chip caches quite substantially, enough to change the degree of redundancy and/or the strength of ECC required to satisfy target manufacturing yields and V_{DDMIN}.

The remainder of this paper is organized as follows. In Section 2, we discuss the SRAM failure probability modeling and cell optimization methodology used. In Section III and Section IV, we analyze the area and performance impact of cell size combined with either redundancy or ECC. In Section V, we jointly explore the size of SRAM cells, the degree of redundancy, and the strength of the ECCs used to minimize the total SRAM area. Section VI concludes the study.

II. SRAM Cell Failure Probability Model and Optimization

To achieve viable manufacturing yields, individual SRAM cells must have extremely small failure rates. For example, to achieve a 99.9% yield rate, the failure probability of a single SRAM cell must be below 3×10^{-11} for a 4MB cache. Estimating such small probabilities through standard Monte Carlo methods would require roughly 10^{12} simulations. Such large-scale simulations make Monte Carlo methods impractical for estimating SRAM failure probability in our study as we require failure probabilities for memory cells of six different sizes running at a number of different supply voltages.

Thus, to estimate RDF-induced failure probability of SRAM cells in our study we adopt a *most probable failure point* (MPFP) method, similar to [10]. Note that the *static noise margin* (SNM) based method to check a failure of a cell cannot accurately account for various other failure mechanisms during SRAM read and write operations. Thus, we adopt the dynamic simulation method using a 128 by 256 SRAM array, checking the failures for each variation sample [10]. The standard deviation of each transistor's V_{TH} is given by [14]:

$$\sigma V_{TH} = \sqrt{\frac{q}{3\varepsilon_{ox}}} \cdot \sqrt{T_{oxe}(V_{TH0} - V_{FB} - 2\phi_B)}\sqrt{WL} \qquad (1)$$

The σV_{TH} for a NMOS (PMOS) transistor with W equal to the minimum L_{EFF} in the high-performance 32nm *predictive technology model* (PTM) is 24mV (29.2mV) [12]. Our base-line cell has the minimum width (W=3λ) for all 6 transistors. We created cell layouts using a TSMC 0.18μm technology design rule and Cadence® Virtuoso® layout editor to analyze the relative area of 6 different SRAM cells. This is shown in Figure 1 where an SRAM cell consists of a pair of *pull-down* (PD), *pass-gate* (PG), and *pull-up* (PU) transistors; the widths of PD, PG, and PU transistors are represented by W_{PD}, W_{PG}, and W_{PU}, respectively. Since W_{PG} is often equal to or smaller than W_{PD}, it does not contribute to the cell size increase. Thus, the sum of W_{PD}, W_{PG}, and a fixed width cost (e.g., the minimum spacing between the active to N-well, etc.) determines the overall area of SRAM cells. Finally, the cell height is usually fixed while $W_{PD}+W_{PU}$ determines the width of the cell; the

Fig. 1. A 6T SRAM cell layout with W_{PD}=15λ, W_{PG} =8λ, and W_{PU} =4λ in 0.18μm technology.

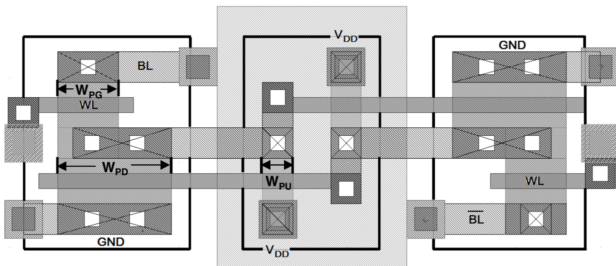


Table 1. W_{PD}, W_{PU}, and W_{PG} for C1~C6 cells and their area relative to C1 size.

	C1	C2	C3	C4	C5	C6
WPD	3λ	5λ	8λ	11λ	14λ	16λ
WPU	3λ	4λ	4λ	4λ	4λ	5λ
WPG	3λ	5λ	8λ	11λ	14λ	16λ
Relative Area	1.00	1.12	1.23	1.35	1.46	1.58

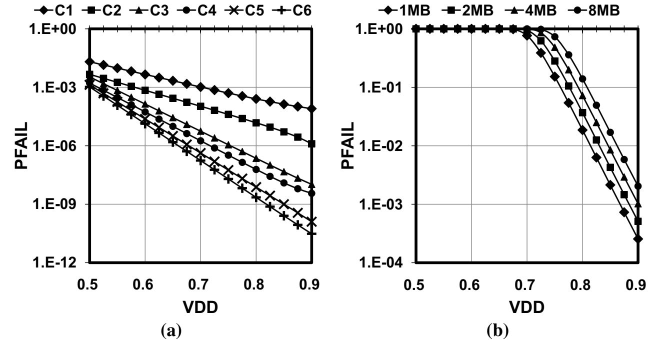


Fig. 2. (a) Failure probabilities of 6 different SRAM cells. (b) Failure probabilities of 1~8MB caches implemented with C6.

minimum size cell has $W_{PD}+W_{PU}$ equal to 6λ plus the fixed width cost. For each cell, we sweep the size of W_{PD}, W_{PG}, and W_{PU} to find the minimum failure probability of each cell as follows:

Objective:

$$Minimize(P_{FAIL}(W_{PU}, W_{PD}, W_{PG})) \qquad (2)$$

Constraints:

$$W_{PG} \le W_{PD}\,,\; W_{PU} + W_{PD} \le W_{CONST} \qquad (3)$$

where P_{FAIL} is the maximum of the read and write failure probabilities for given W_{PD}, W_{PG}, and W_{PU}. In Table 1 we tabulate the optimized W_{PD}, W_{PG}, and W_{PU} for each cell and its cell area relative to the area of C1. Each cell size can have W_{CONST} equal to 9, 12, 15, 18, and 21λ. All the possible combinations of W_{PD}, W_{PG}, and W_{PU} are exhaustively searched to minimize the failure probability at 600mV with the step value equal to 0.5λ for a given cell size, i.e., W_{CONST}.

Figure 2-(a) plots the failure probabilities of 6 different SRAM cells for C1, C2, C3, C4, C5, and C6. In the 32nm technology with the given σV_{TH} for NMOS and PMOS transistors, the failure probabilities are fairly high for very small cells, e.g., C1 and C2. This is because transistors with smaller W_{PD}, W_{PG}, and W_{PU} exhibit larger σV_{TH}, cf. Eq. (1). Their (on-current) characteristics are thus more sensitive to V_{TH} variability, leading to higher failure probability than larger cells. Note that the cell failure probabilities for the largest two cells—C4 and C5 are very close to the Intel's data based on 45nm technology [6]. Figure 2-(b) shows the calculated failure probabilities of 1, 2, 4, and 8MB caches implemented with the C6 SRAM cells. As the number of cells (or cache size) increases, the overall failure probability increases as well. This limits the scaling of SRAM cell size for large on-chip last-level caches (LLCs) where the typical cache size is often larger than 4MB in commercial multi-core processors. For a 4MB cache with 90% yield (or failure probability equal to 0.1), the V_{DDMIN} will be around 0.775V without using any special techniques (e.g., read and write assisting, redundancy, or ECC).

III. SRAM Total Area Minimization Using Redundancy

Various redundancy techniques have been widely adopted in DRAM and SRAM to repair defective cells caused by stuck-at-faults [9][16]. More recently such techniques have also been used to

improve VDDMIN for SRAM thereby enabling lower operating voltages. Figure 3 shows an example of a column redundancy implementation in an SRAM array using a "shift redundancy" scheme [16]. Basically, if any cell in a column fails, the next column replaces the failed column. In turn, the next column is replaced with the column one next to it, and so on. This is repeated until a redundant column replaces the last non-redundant column in a segment as shown in Figure 3.

The redundancy MUX illustrated in Figure 3 is often programmed by *one-time-programmable* (OTP) fuses and a redundancy decoder comprised of 4-bit scan flip-flops. Combined with the column address, it steers each bit from each column, including the redundant one, to the correct corresponding bit location of the I/O bus. The bold lines in Figure 3 illustrate the data flow after the third column in the second bit is fixed by a redundant column. Note that only one redundant column per every m columns can be used due to the increasing complexity of the steering logic circuit [18]. For example, consider the case when the data I/O width is 32 and there are two redundant columns. The first redundant column can only fix a single column in the first 16-bit segment and the second one in the second segment. Thus, we cannot repair two failing columns in the first 16-bit segment even with two redundant columns unless a much more complex and expensive data steering mechanism is adopted. This would increase area and delay overhead in the SRAM array. The area and delay impact of redundancy will be addressed further, later in this section.

Let the probability of failure of a cell be P_{cell}. The column-failure probability, P_{col}, is

$$P_{col} = 1 - (1 - P_{cell})^{i} \quad (4)$$

when a column in an array has i cells. Next, let the number of redundant columns be equal to k in an array with j columns, i.e., one redundant column per $m = j/k$ columns. Finally, let a segment be a set of m consecutive columns. Then, the failure probabilities of a block, an array, and a cache are

$$P_{segment} = 1 - (1 - P_{col})^{m} - m \cdot P_{col} \cdot (1 - P_{col})^{m-1} \quad (5)$$

$$P_{array} = 1 - (1 - P_{segment})^{k} \quad (6)$$

$$P_{cache} = 1 - (1 - P_{array})^{l} \quad (7)$$

where l is the number of arrays in an L-MB cache — $(L \times 8 \times 2^{20})$ / $(i \times j)$. Figure 4 shows (a) the number of required redundant columns per array and (b) the relative total SRAM array area including the redundant columns to satisfy 95% yield for a 4MB cache and different target VDDMIN values. The total area of an SRAM array is normalized to that of an array implemented with the largest C6 cells where no redundant columns are used. Each bank with a capacity of 4KB has 128 columns and 256 rows, i.e., $i = 256$ and $j = 128$. When, for a given cell size, the target yield and VDDMIN cannot be met even if 32 redundant columns are provided, the results are not plotted in Figure 4. We assume that we have already applied various circuit techniques, e.g., [1]-[4] to reduce the failure probabilities of the cells shown in Figure 2-(a) by a factor of ten. First note that results for C1 and C2 are not included as they could not satisfy the target yield for the 4MB cache even with 32 redundant columns since they have very high failure probabilities for the given target VDDMIN range (600~700mV). In addition, C3, C4, and C5 could not satisfy the target yield below 675, 650, and 625mV, respectively, even with 32 redundant columns. Second, note that Figure 4 shows that, as expected, SRAMs with smaller cells require more redundant columns to meet the target yield. However, even after including the extra area penalty due to more redundant columns, C3, C4, and C5 lead to the smallest total area at 700, 675, and 650mV target VDDMIN (22%, 14%, and 7% smaller than C6). Third, as the total cache size increases, the cost of redundancy increases and larger cells are preferred to achieve the smallest total area. This is shown in Table 2 which summarizes the relative total SRAM array area and the number of required bit corrections for 8 and 16MB caches. Green-colored cells show the optimal cells for each target VDDMIN value. For example, for VDDMIN equal to 700mV C4 leads to the smallest total area in 16MB caches while C3 does in 4 or 8MB caches. Note that other ECC schemes can result in different choices for optimal cell size.

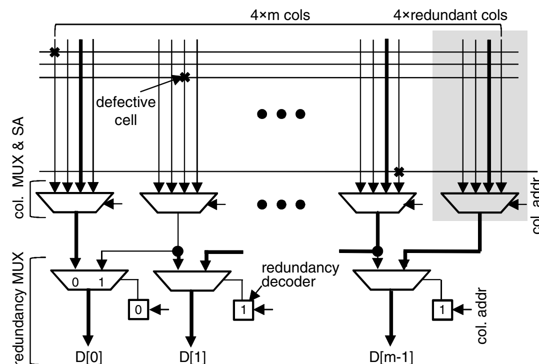


Fig. 3. An example of column redundancy implementation in an SRAM array employing 4-to-1 column MUXs.

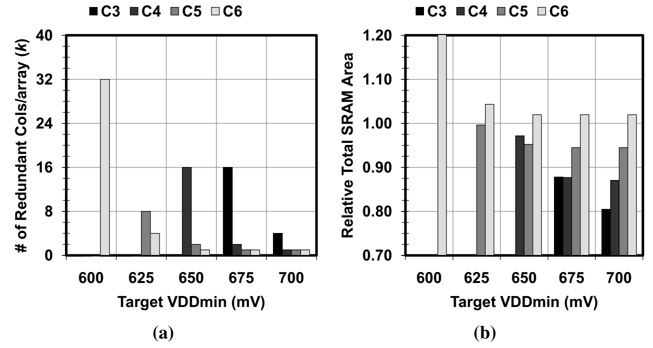


Fig. 4. (a) k per array and (b) the relative total SRAM array area including the redundant columns to satisfy 95% yield for a 4MB cache and different target VDDMIN values.

Area impact of column redundancy logic: The OTP fuses to program redundancy MUXs and decoders can incur a significant area penalty. This can increase the total SRAM area when the area for the fuses is also accounted for. However, recent *package-on-package* (PoP) technology allows us to mount a Flash memory package, often containing BIOS and processor specific information, on a processor package inexpensively and it begins to replace expensive on-chip OTP fuses [17]. The area overhead due to the extra redundancy MUXs and decoders was estimated by performing a layout in the 0.18μm technology, and scaled accordingly for the

Table 2. The relative total SRAM array area versus k to satisfy 95% yield for 8 and 16MB caches and different target VDDMIN. "—" implies a cell fails to achieve the target yield at the given VDDMIN values.

Cache Size	VDDMIN	C1	C2	C3	C4	C5	C6
8MB	600mV	—	—	—	—	—	1.25/32
	625mV	—	—	—	—	1.04/16	1.06/8
	650mV	—	—	—	1.07/32	0.96/4	1.01/1
	675mV	—	—	0.98/32	0.88/4	0.93/1	1.01/1
	700mV	—	—	0.83/8	0.86/1	0.93/1	1.01/1
16MB	600mV	—	—	—	—	—	—
	625mV	—	—	—	—	1.16/32	1.13/16
	650mV	—	—	—	1.07/32	0.98/8	1.01/1
	675mV	—	—	—	0.91/8	0.93/1	1.01/1
	700mV	—	—	0.88/16	0.87/2	0.93/1	1.01/1

32nm technology. The conservatively estimated area overhead for the MUXs and redundancy decoders is roughly equal to 1.2% of the total SRAM array area.

Performance impact: As mentioned in Section II, the height of cells is most fixed and only the width varies depending on the size of the transistors (WPD, WPG, and WPD). Thus, the size of arrays with larger cells or more redundant columns grows in the word-line direction, leading to slower word-line delay. In other words, an array with less area must have a shorter word-line length, i.e., less word-line delay. However, smaller cells often exhibit more bit-line delay. Overall, when the increased bit-line delay of smaller SRAM arrays is countered by the reduced word-line delay, an SRAM array access time impact is very small. According to our simulation, an array with C4 cells shows only less than 5% increase in word-line plus bit-line delay, relative to one with C6. Note that smaller area with lower VDDMIN is the most critical design priority for architects in large LLCs. Further, a large LLC has a long latency (more than 10 processor cycles) which is mainly needed to distribute address and data signals across the structure. Therefore, the impact of smaller cell size on overall latency (and thus performance) is negligible. Finally, the delay impact due to the redundant column MUX is a fixed cost independent of the number of redundant columns.

IV. SRAM Total Area Minimization Using ECC

ECCs add redundant *parity* or *check bits* to detect and repair errors in stored data. *Single-error-correction, double-error-detection* (SECDED) codes such as the Hamming codes are well known and have been widely used in both DRAM and SRAM, primarily to protect them from particle-induced soft errors. However, due to the increasing probability of SRAM cell failure as technology is scaled, multi-bit error correction codes are now being used in large on-chip LLCs to satisfy yield targets at desired VDDMIN values. For example, a *double-error-correction*, *triple-error-detection* (DECTED) BCH codes has been deployed in Intel's Core™ i7—new Intel®'s 45nm quad-core processor family to cope with increased scaling-induced SRAM failures.

Figure 5 illustrates an ECC implementation in a large 4-way 8-MB LLC. With a q-bit error correction capability requiring r check bits, the failure probability of a k-bit data block is given by

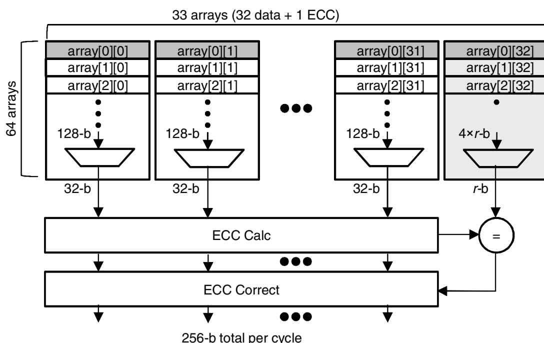
$$P_{block} = 1 - \sum_{i=0}^{q} \binom{n}{i} (1 - P_{cell})^{n-i} \times P_{cell}^{i} \qquad (8)$$

where n is the length of the code equal to k+r. The failure probability of a cache with a total of j data block is

$$P_{cache} = 1 - (1 - P_{block})^{j} \qquad (9)$$

Due to the substantially higher failure rates of SRAM cells at very low voltages, we need to implement multiple-bit error correction capability per data block to meet yield targets. Conventional multiple-bit error-correcting codes such as BCH codes do not have an encoding/decoding complexity that scales in a simple way with correction capability. For this reason in this study we use *Orthogonal Latin Square Codes* (OLSC) to implement ECCs for $q \geq 3$ [19]. While OLSC requires more check bits than Hamming or BCH codes, they have modular correction hardware and lower coding complexity, often allowing for fast encoding/decoding for multiple-bit error corrections. In OLSCs, when k is equal to p^2, they require $2 \times p \times q$ check bits. For example, n becomes 256+32q for each 256-bit data block and q-bit error correction capability. Further discussion of the area and delay impacts of the use of ECCs will be addressed later in this section.

Fig. 5. ECC implementation in a 4-way 8-MB LLC.



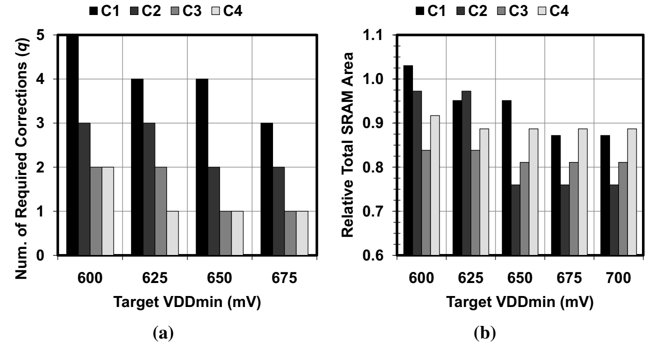


Fig. 6. (a) q per 256-bit data and (b) the relative total SRAM array area including the overhead of extra check bits to satisfy 95% yield for a 4MB cache and different target VDDMIN values.

Figure 6 shows (a) the number of required bit corrections (q) per 256-bit data and (b) the relative total SRAM array area *including check bits* to satisfy 95% yield for a 4MB cache and different target VDDMIN values. The total SRAM array area is normalized to the SRAM array implemented with C6 where no ECCs are used. The configuration of SRAM array and the data used to generate the raw cell failure probabilities are the same as in the redundant-columns case discussed in Section III. In comparison with those results on redundant columns, the use of ECCs allows us to use even smaller cells (e.g., C1 and C2) while still meeting yield targets in the specified VDDMIN range. In contrast, in Section III (cf. Figure 4-(b)) only the largest C6 cell, paired with the largest number (32) of redundant columns could meet the yield targets at 600mV, and required more than a 25% increase in area. However, ECC allows cell sizes C1-C5 to meet the target yield at 600mV with less total area than the C6 solution of Figure 4-(b). In addition, SRAM arrays with cell sizes C2, C3, or C4 will lead to smaller total area than ones with C5 or C6 plus ECC; C5 and C6 still require at least 1-bit correction. Thus, the results for cell sizes C5 and C6 are not plotted in Figure 6. In terms of selecting a cell that minimizes total area, we argue that C3 (alternately, C2) is the best choice for 600mV-625mV (650mV-700mV), leading to 20% (28%) smaller total area than the baseline C6 as can be seen by examining Figure 6-(b). Finally, note that the optimal cell size increases as target VDDMIN becomes lower, i.e., C2 is optimal at 650mV while C3 is optimal at 600mV. Table 3 summarizes the relative total SRAM array area versus the number of required bit corrections for 8 and 16MB caches; the green-colored cells show the optimal cells for each target VDDMIN value.

Area impact of ECC encoder and decoder: As either the ECC correction capability q increases, or as the size k of the data block to which the ECC is applied increases, the ECC hardware complexity (and thus the area required) also increases. Due to the nature of most ECCs, such as Hamming and BCH, the complexity or encoder/decoder increases logarithmically. However, to conservatively estimate the overall area impact of the ECC use, we assume that the area required to implement encoding and decoding increases linearly with q and k. As a baseline, we adopted the area numbers reported in [21] and scale them appropriately to the 32nm technology. Our assumptions are first that the ECC encoder and decoder is located at the front-end of a LLC, i.e., they are shared by

Table 3. The relative total SRAM array area versus q to satisfy 95% yield for 8 and 16MB caches.

Cache Size	VDDMIN	C1	C2	C3	C4	C5	C6
8MB	600mV	1.03/5	0.97/3	0.84/2	0.92/2	1.00/2	1.04/1
	625mV	0.95/4	0.97/3	0.84/2	0.92/2	0.96/1	1.04/1
	650mV	0.95/4	0.76/2	0.84/2	0.89/1	0.96/1	1.04/1
	675mV	0.95/4	0.76/2	0.81/1	0.89/1	0.96/1	1.04/1
	700mV	0.87/3	0.76/2	0.81/1	0.89/1	0.96/1	1.04/1
16MB	600mV	1.03/5	0.97/3	0.84/2	0.92/2	1.00/2	1.04/1
	625mV	1.03/5	0.97/3	0.84/2	0.92/2	0.96/1	1.04/1
	650mV	0.95/4	0.97/3	0.84/2	0.89/1	0.96/1	1.04/1
	675mV	0.95/4	0.76/2	0.81/1	0.89/1	0.96/1	1.04/1
	700mV	0.87/3	0.76/2	0.81/1	0.89/1	0.96/1	1.04/1

Table 5. Multi-core processor simulation parameters.

Parameter	Value
Cores	4 cores on a fully connected on-chip network running at 3GHz with 1 thread per core
Pipeline	4-wide with fetch, decode, dispatch, execute, and retire stages.
ROB/LSQ/SB	32
Private L1/D1 caches	32KB, 8-way, 32B block, 3-cycle latency
Shared L2 caches	Banked per core, 8MB, 12-cycle latency
Coherence Protocol	Directory-based MESI
Main memory	DDR3-1600MHz 7-7-7-20 latency, 4GB, 64B block, 4096 page size

all arrays, and second that the area efficiency of SRAM[1] is 0.75. Then, for example implementing single error correction requires an additional overhead of only 0.02% for a 4MB LLC.

Impact of ECC on total execution time: In an Itanium processor, a 3MB LLC using SECDEC codes for 256-bit data block consumes a half-cycle latency for a total LLC latency of 9 cycles [20]. In [21], it is reported that DECTED requires slightly less than twice the ECC latency of SECDED. As mentioned before, due to the logarithmic structure of most ECC encoding/decoding logic, ECC latency grows roughly logarithmically in q. Being conservative in our assumptions, we model ECC latency for SECDED as 1 cycle, resulting in a total LLC latency of 12 cycles, and growing linearly in proportion to q, i.e., LLC latency for q-bit error corrections is $11+q$ cycles. Table 4 summarizes the impact on total execution time of adopting stronger ECCs when running commercial server applications (Apache, OLTP, JBB, Zeus) on the GEMS simulator [22]. Table 5 provides details on multi-core processor simulation parameters. Since commercial server workloads have a large cache footprint, incurring many L1 cache misses, it is very important to assess the effect on total execution time due increased L2 cache access latency. Note that every cell

Table 4. The performance degradation (%) due to ECC encoding /decoding time versus q. The first and second numbers are the degradations normalized to 1-bit ECC and no ECC, respectively.

q	1	2	3	4	5
Latency	12	13	14	15	16
Apache	—/0.5	1.0/1.5	2.5/3.0	3.8/4.4	5.5/6.1
OLTP	—/2.0	1.3/3.3	1.1/3.1	2.5/4.6	3.5/5.6
Zeus	—/0.1	1.9/1.9	2.0/2.1	2.2/2.3	1.7/5.0
JBB	—/0.5	0.6/1.1	1.3/1.9	2.1/2.6	2.8/3.3
Geo Mean	—/0.4	1.1/1.8	1.6/2.5	2.6/3.3	3.1/4.9

[1] The area efficiency is defined by the total SRAM cell area in an LLC divided by the total LLC area including all the peripheral circuits, i.e., decoders, column MUXs, sense-amplifiers, and address/data buses.

requires at least 1-bit correction to meet the target yield for 4MB and larger caches. All runtimes are normalized to those of 12-cycle LLC latency. We also show the increase in execution time relative to no ECCs (and hence encoding/decoding latency overhead). Even under our conservative assumptions about latency, the average increase in total execution time resulting from an increase from single-bit error correction to double-bit error corrections negligible, roughly 1%. However, as the strength of the error-correction used increases, so does the negative impact on total execution time. This, for example makes C1 (which requires 5-bit error-correction capability) impractical for achieving the target yield at 600mV due to considerable increase of total execution time.

V. SRAM Total Area Minimization Using Both Redundancy and ECC

Since column redundancy and ECC are distinct techniques, we can jointly apply both to reduce further the total SRAM area. The optimum mixture of the two techniques depends on cell size. Furthermore, combining column redundancy with ECC can also allow a finer adjustment of overall cache failure probability to meet target yields. For example, C6 has very low failure probability but, to satisfy 95% yield for 4MB caches at 600mV, it requires either at least one redundant column per array or single-bit error correction capability per 256-bit data block. Single-bit error correction capability per 256-bit data block required 10 check bits, i.e., 4% overhead. On the other hand, a single redundant column per array incurs only 0.8% overhead. In both cases overhead is calculated in terms of the total number of extra bits per array required. However, as the failure rate of cells increases (as individual transistors scale down), ECC is a more efficient method, i.e., less area overhead in protecting caches against multiple-bit failures.

The problem of analyzing the exact failure probabilities for a scheme where column redundancy and ECC are combined is mathematically complex. In fact, error calculations turn into integer programs. In this paper, we suggest a heuristic approach to calculating such failure probabilities. The approach yields good estimates of failure probabilities, with the advantage of being mathematically tractable. Our approach is as follows. Say that the number of redundant columns is k, the error correction capability is q, and the failure probability of a cell is P_{cell},

1. We compute the failure probability of a cache, $P_{cache\text{-}red}$ which only uses k redundant columns per array using Eq. (4)~(7).
2. We calculate the "equivalent failure probability" of a cell $P_{cell\text{-}eq}$. This is the cell failure probability that will make the failure probability of the cache be equal to $P_{cache\text{-}red}$ if there were no redundant columns.
3. We estimate the failure probability P_{cache} for a cache with an error correction capability of q bits using Eq. (9) where the equivalent cell failure probability $P_{cell\text{-}eq}$ is assigned to Ex. (8) instead of P_{cell}.

Figure 7-(a) plots the optimal number of extra columns required per array to implement column redundancy and ECC. Figure 7-(b) plots the relative total SRAM array area, including the extra columns, to satisfy a 95% yield for a 4MB cache and different target VDDMIN values. We explore all the ECC parameter pairs (q, k) that satisfy 95% yield with each cell at each target VDDMIN and chose the combination that results in the minimum extra column overhead. For each of the four cell sizes, C1, C2, C3, and C4, and at each VDDMIN (600, 625, 650, 675, and 700mV), respectively the best choice of design parameters is {(3, 16), (2, 1), (1, 1), (1, 1)}, {(2, 32), (2, 1), (1, 1), (1, 1)}, {(2, 16), (1, 1), (1, 1), (1, 1)}, {(2, 8), (1, 2), (1, 1), (0, 2)}, and {(2, 2), (1, 1), (0, 4), (0, 1)}. While the column redundancy is implemented per array, the ECC is applied to the cells at the same row across multiple arrays. Thus, we assume that we distribute the ECC check-bits evenly across the multiple arrays to calculate "the number of extra columns per array" in

Fig. 7. (a) The optimal number of extra columns per array to implement both redundancy and ECC. (b) The relative total SRAM array area including the overhead of extra check bits to satisfy 95% yield for a 4MB cache and different VDDMIN values.

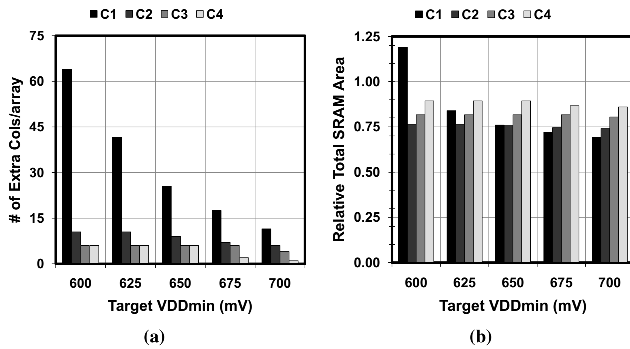


Table 6. The relative total SRAM array area versus the number of required extra columns per array to satisfy 95% yield for 8 and 16MB caches for different target VDDMIN values.

Cache Size	VDDMIN	C1	C2	C3	C4	C5	C6
8MB	600mV	1.19/64	0.77/11	0.82/7	0.89/6	0.97/6	1.04/5
	625mV	0.84/42	0.77/11	0.82/6	0.89/6	0.96/5	1.04/5
	650mV	0.76/26	0.76/10	0.82/6	0.89/5	0.96/5	1.04/5
	675mV	0.72/18	0.75/7	0.81/5	0.89/5	0.96/5	1.04/5
	700mV	0.70/14	0.74/6	0.81/5	0.89/5	0.96/5	1.04/5
16MB	600mV	1.27/80	0.77/12	0.82/7	0.89/6	0.97/6	1.04/5
	625mV	1.15/56	0.77/11	0.82/6	0.89/6	0.96/5	1.04/5
	650mV	0.84/42	0.77/11	0.82/6	0.89/5	0.96/5	1.04/5
	675mV	0.76/26	0.76/9	0.81/5	0.89/5	0.96/5	1.04/5
	700mV	0.70/14	0.74/6	0.81/5	0.89/5	0.96/5	1.04/5

Figure 7-(a). Note that in many layouts, separate arrays are used to store the ECC check bits. For the purposes of this study, however, we calculated the overhead on per array basis.

As the cell size or target VDDMIN decreases, examining Figure 7 we see that the required extra columns to implement both column redundancy and ECC increases. Although C1 is the smallest cell size, the overall overhead for redundancy and ECC to satisfy the target yield at 600mV when using cell size C1 is quite considerable. Meanwhile, C2 is larger than C1 but smaller than C3 and C4. In fact, the choice of C2 minimizes total SRAM area because its failure probability is not too high and both redundancy and ECC are effective. Comparing the ECC only results shown in Figure 6-(b) to the ones presented in Figure 7-(b), we reduce the total SRAM area by 8% for a 4MB cache (while targeting 95% yield and a VDDMIN of 600mV*)* ECC-only reduces the total SRAM area by 20% while ECC plus redundancy by 27% relative to the SRAM implemented with C6. Table 6 summarizes the relative total SRAM array area and the number of extra required columns per array for 8 and 16MB caches; the green-colored cells show the optimal cells for each target VDDMIN value.

VI. CONCLUSION

In this paper, we discuss how the sizing of SRAM cells can be determined jointly with the amount of available redundancy and the use of ECC to minimize total SRAM array area. Although the failure probability of SRAMs decreases with its increasing cell size, using a large SRAM cell to implement large on-chip caches is not often economical due to the associated die cost. Instead redundancy and ECC can be used to repair failing cells to meet target yields while enabling reliable lower-voltage operation. In this paper we combine redundancy, ECC, and appropriate cell sizing, to yield a jointly optimized SRAM design with minimum area for yield and VDDMIN targets. According to our experiments using a 32nm technology, our final SRAM design (cache sizes from 4 to 16 MB and targeting a VDDMIN of 600mV) reduces by 27% the total required SRAM area with minimum impact on the performance of multi-core processors. Finally, we argue that the total SRAM area can be minimized further with a more efficient ECC that exploits SRAM architecture and its failure characteristics.

ACKNOWLEDGEMENT

This work is supported by an NSF grant (CCF-1016262), a Fall Competition Multidisciplinary Research Award from the University of Wisconsin-Madison Graduate School, and NSF CAREER Awards (CCF-0953603 and CCF-0844539).

REFERENCES

[1] K. Zhang et al. A 3-GHz 70-Mb SRAM in 65-nm CMOS technology with integrated column-based dynamic power supply. *IEEE JSSC*, 41(1): 146-151, 2006.

[2] F. Hamzaoglu, et al. A 153Mb-SRAM design with dynamic stability enhancement and leakage reduction in 45nm high-k metal-gate CMOS technology. *IEEE ISSCC*, pp: 376, 2008.

[3] M. Khellah et al. PVT-variations and supply-noise tolerant 45nm dense cache arrays with diffusion-notch-free (DNF) 6T SRAM cells and dynamic multi-Vcc circuits. *IEEE VLSI Circuit Symposium*, pp: 48-49, 2008.

[4] S. Ohbayashi. A 65-nm SoC embedded 6T-SRAM designed for manufacturability with read and write operation stabilizing circuits. *IEEE JSSC*, 42(4): 820-829, 2007.

[5] A. Agarwal et al. A process-tolerant cache architecture for improved yield in nanoscale technologies. *IEEE TVLSI*, 13(1): 27-38, 2005.

[6] C. Wilkerson et al. Trading off cache capacity for reliability to enable low-voltage operation. *IEEE ISCA*, pp: 203~214- 2008.

[7] D. Roberts et al. On-chip cache device scaling limits and effective fault repair techniques in future nanoscale technology. *IEEE DSD*, pp: 570-578, 2007.

[8] J. Kulkarni et al. A 160mV robust Schmitt trigger based subthreshold SRAM. *IEEE JSSC*, 42(10): 2303-2313, 2007.

[9] S. Schuster. Multiple word/bit line redundancy for semiconductor memories. *IEEE JSSC*, 13(5): 276-287, 1978.

[10] D. Khalil et al. Accurate estimation of SRAM dynamic stability. *IEEE Trans. on VLSI*, 16(12): 1639-1647, 2008.

[11] http://www.eas.asu.edu/~ptm.

[12] ITRS 2009.

[13] Z. Chishti et al. Improving cache lifetime reliability at ultra-low voltages. *IEEE MICRO*, pp:88-99, 2009.

[14] J. Croon et al. Physical modeling and prediction of the matching properties of MOSFETs. *IEEE ESSDERC*, pp: 193-196, 2004.

[15] S. Mukhopadhyay et al. Modeling and estimation of failure probability due to parameter variations in nanoscale SRAMs for yield enhancement. *IEEE VLSI Circuit Symp.*, pp. 64-67, 2004.

[16] A. Ohba et al. A 7-ns 1-Mb biCMOS ECL SRAM with shift redundancy. *IEEE JSSC*, 26(4): 507-512, 1990.

[17] http://www.toshiba.com/taec/news/media_resources/docs/FlashFactSheet.pdf

[18] T. Kirihata et al. Fault-tolerant Designs for 256 Mb DRAM. *IEEE JSSC*, 31(4): 558-566, 1996.

[19] H. Hsiao et al. Orthogonal Latin Square Codes. *IBM Journal of Research and Development*, 14(4): 390-394, 1970.

[20] D. Weiss et al. The on-chip 3-MB subarray-based third-level cache on an Itanium microprocessor. *IEEE JSSC*, 37(11): 1523-1529, 2002.

[21] R. Naseer et al. Parallel double error correcting code design to mitigate multi-bit upsets in SRAMs. *IEEE ESSCIRC*, pp: 222-225, 2008.

[22] M. Martin et al. Multifacet's general execution-driven multiprocessor simulators (GEMS) toolset. *ACM Computer Architecture News*, 33(4): 92~99, 2005.