Cache Design for Yield-per-Area Maximization: Switchable Spare Columns with Disabling (SSC-Disable)

Soowang Park and Sandeep K. Gupta
Ming Hsieh Department of Electrical Engineering,
University of Southern California, Los Angeles, CA 90089-2560
{soowangp, sandeep}@usc.edu

***Abstract*—Modern SRAMs have high failure rates due to defects and variations, especially during low-voltage operation in low power modes. At high failure rates, a direct application of spare rows and columns approaches incurs high overheads. Since a recent paper [1] says that error correcting codes (ECCs) are the only cost-effective approach under high failure rates, we analyzed the strengths of ECC. This showed that, at high overheads, ECC provides one key advantage: the ability to correct failures in cells in different locations in different rows.**

To find a low overhead solution, we turn to the Divided Wordline/Bitline (DWL+DBL) [2] approach, a spares-based approach designed to provide exactly this capability. Since DWL+DBL also has high overheads, we develop our ideas for simplifying ECC as well as DWL+DBL and derive our new approach called Switchable Spare Columns (SSC), where a spare column can replace failing cells in different locations in different rows at lower overheads. To further reduce overheads, we propose SSC-Disable, which combines SSC with cache block disabling [3].

We then develop a method to find globally efficient cache designs with SSC or SSC-Disable. This method maximizes yield-per-area (YPA) under user-specified constraints on delay and power overheads. We show that the proposed SSC-Disable approach significantly improves yield compared to state-of-the-art ECC-based approaches.

I. Introduction

As technology continues to scale, concerns continue to grow about variations and defects during chip fabrication, as well as about soft errors during a chip's operational lifetime. Table I, summarizes the terms we use for these phenomena as well as their effects and key characteristics. *Variations* refer to the deviations in the geometries of layout features and/or compositions of the materials that implement these features, due to the limitations of the fabrication processes. *Defects* refer to severe physical deviations, such as opens and shorts within or between features. These affect circuit operation in many ways, including making circuits slower hence causing logic errors when the circuit is operated at desired speed or even at slow speed. For each individual fabricated chip, these effects are typically permanent, and the corresponding deviations in circuit operation are referred to as *fabrication failures.*

In addition, during the operational lifetime of a chip, circuit operation may be affected by radiation or alpha-particles. Such deviations are referred to as *soft errors* which typically manifest as a deviation in logic values, e.g., values stored in an arbitrary SRAM cell or latch, and are mostly transient.

The focus of this paper is on high rates of fabrication failures due to defects and variations, especially during low-voltage operations in low power modes.

To understand the level of challenge, we turn to the variation-induced failure rates of SRAM cells presented in [3][4]. (Even for the 32nm technology, variations were a dominant cause for SRAM yield loss [1].) As SRAM cell failure rate due to defects ($P_{fr,d}$) was identified as being close to 1×10^{-6} [5], we update the analysis in [3][4] to obtain the failure rates shown in Table II. This table clearly shows that we face extremely high fabrication failure rates, especially in low-power modes.

In terms of a solution, a direct application of spare rows and columns approach provides designs with very high overheads (e.g., see [6]). Since a recent paper [1] says that ECCs are the only cost-effective means for tackling high failure rates in memories, we study the benefits and costs of ECCs vs. the traditional spare columns/rows approaches. We learn that, while ECC has high area and delay overheads, it has two key strengths: (1) the ability to correct failures in cells in different locations in different rows, and (2) the ability to correct errors even as the locations of cells with failures change over time. We then note that while we do face high rates of fabrication failures, the locations of cells with such failures are known during post-fabrication testing and fixed thereafter. Hence, for fabrication failures, only the first key strength of ECCs, namely fix failures in different cells in different rows, is of interest. This observation leads us to a search for ways of designing spare columns/rows that can provide this benefit of ECC at lower area and performance overheads.

After a summary of previous approaches in Section II.A, in Section II.B we analyze ECC approaches and identify the strength of ECC that is important to us. In the remainder of Section II we present the new ideas we use to dramatically reduce overheads to derive our new approaches: switchable spare columns (SSC) and SSC-Disable. In subsequent sections, we present our methods for optimal design of SSC and SSC-Disable, followed by experimental results which show that SCC-Disable provides much higher yield-per-area than state-of-the-art ECC approaches.

II. Switchable Spares

A. Previous Approaches

Classical Spares: The traditional approach to combat failures in SRAMs is to use spare rows/columns in memory to repair cell failure due to defects identified during post-fabrication test and repair [7], and to use error correcting codes (ECC) to correct soft errors. However, for our target failure rates, a spare row or column with thousands of cells typically only repairs a small number of failures, e.g., a spare column with 2048 cells will typically repair <10 failures. Fig. 1(b) provides a simplified illustration of this. Furthermore, increasing the number of spare rows/columns is difficult beyond some level, since beyond that the area overhead of the multiplexing circuitry becomes extremely high [6]. The alternative approach of partitioning the

This research is supported by National Science Foundation.

978-1-7281-1170-4/19/$31.00 ©2019 IEEE

TABLE I. CLASSIFICATION OF THE DIFFERENT TYPES OF PHYSICAL PHENOMENA IN AN SRAM CELL. THE LAST TWO COLUMNS SHOW OUR APPROACH.

Phase of chip's life	Causes of failure	Characteristics of failures	Approaches for mitigation							
			No OS-level disabling				With OS-level disabling			
			Classical	Spares-ECC [9][10]	DBL+DWL [2]	ECC-ONLY [1][3][8]	ECC-DIS [3][11]	Spares-PCD [6]	SSC-ECC-DIS	SSC-DIS
Fabrication	Defects Variations	Location of failing cells known at post-fabrication test	Spare rows/cols	Spare words + ECC	Switchable spare rows/cols	ECC	ECC + block disabling	Spare rows/cols + page cover disabling	Switchable spare columns + ECC + block disabling	Switchable spare columns + block disabling
			-	-						
Operational life	Soft errors: Radiation, α-particles, ...	Failure can occur at any location	ECC	ECC	ECC	ECC	ECC	ECC	ECC	ECC

memory into a larger number of smaller subarrays to effectively add higher levels of spares per cell, is also limited by the area overhead of inter-subarray interconnects and circuitry. In summary, the classical spare rows and columns approach provides insufficient yield-per-area for the extremely high failure rates we face during low-voltage operation.

ECC: As the concerns regarding variations emerged, cache design approaches that use ECC to correct errors caused by fabrication variations as well as soft errors [3][8] were proposed. Previous techniques that combined ECC and spares [9][10] repaired failures due to defects (Spares-ECC in Table I). Furthermore, a recent approach [1] suggested that at high failure rates memories achieve resilience primarily by using ECCs. However, ECC is expensive to implement. In addition to the overheads of storing check bits, ECC approaches incur high area and delay overheads of decoders.

Highly efficient approaches [3][6][8][11][12] have been proposed based on the observation that some of the largest SRAMs are viewed as caches by operating systems. Hence, it is not necessary to repair every failing cell by adding hardware redundancy. Instead, if we use a lower level of redundancy which leaves some failing cells unrepaired, we can use software level approaches to avoid using unrepaired cache blocks. In this fashion, we can still achieve a high yield at a much lower hardware overhead. Since such ECC-based approaches are indeed the most efficient for our failure rates, we compare our approach with the best ECC-based approaches (in Section IV).

B. Analysis of ECC: What makes it effective?

We analyze the use of ECC for our high failure rates, given the guidance in [1] that for such failure rates ECC is the only effective approach. The yield is improved significantly by using ECC with a relatively small number of check-bit columns. Specifically, for k-bit error correction across an n-bit word, we need roughly $k \times log_2(n)$ additional check-bits. However, the area and delay overheads of the check-bits and the error correcting circuitry is high and yield-per-area is unsatisfactory.

A quantitative analysis of multiple ECC approaches for SRAM sub-arrays of different sizes showed that for fabrication failures, ECC has two strengths compared to spare columns.

1) ECC can correct failures in cells in different locations in different rows, and
2) ECC can correct errors even as the locations of cells with failures change over time.

However, for fabrication failures due to defects and variations, for each fabricated chip, locations of failing cells are fixed and identified during post-manufacturing testing. In the light of this, while the first strength is useful for our problem, the second one is not of interest. (In contrast, both strengths are useful for soft-errors, which is why ECC has always been the approach for protection against soft-errors.)

This brings us to the following key question: Are there spares based approaches that provide the first benefit above at lower overheads?

For our problem, while each row may have failing cells at different locations, all these locations are fixed and known during post-fabrication testing. This leads to the following.

Key Idea 1: *For our problem, instead of storing check-/parity-bits for ECC and using a high-overhead decoder to determine the locations of failing cells, we can simply store the locations of failing cells in each row.*

C. Analysis of DBL+DWL: How to improve?

Based on above exploration and analysis, our goal is to design area-efficient switchable spare columns and rows which can be used flexibly to replace failing cells in arbitrary columns for each row, as conceptually illustrated in Fig. 1(c). Our challenge is to find a low-overhead approach for implementing this scheme.

Previously, Divided Bitline/Wordline (DBL+DWL) [2] approach was proposed to serve precisely this purpose. It divides every bit-line/word-line and assigns divided spare rows/columns

TABLE II. SRAM FAILURE RATE; (A) FAILURE RATE DUE TO DEFECTS, $P_{fr,d}$ AND (B) FAILURE RATE DUE TO VARIATIONS, $P_{fr,v}$ [4][5]

VDD (mV)	700	650	600	550	510	480	450
Pfr,v	10^{-7}	10^{-6}	10^{-5}	10^{-4}	$3x10^{-4}$	10^{-3}	$3x10^{-3}$
Pfr,d	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}

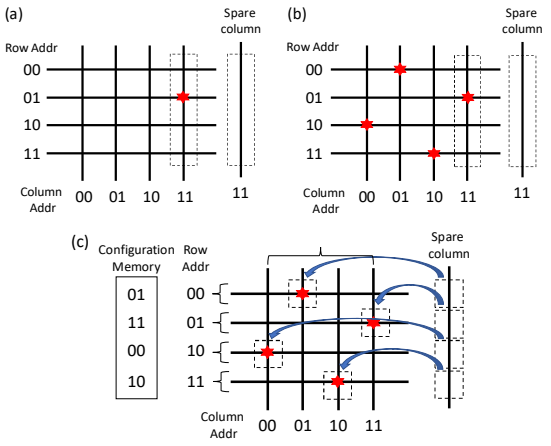

Fig. 1. Traditional spare column: (a) can repair a failure at low-failure rate, (b) cannot repair all failures at high-failure rate, and (c) Switchable Spare Column (SSC) can repair all failures at high-failure rate.

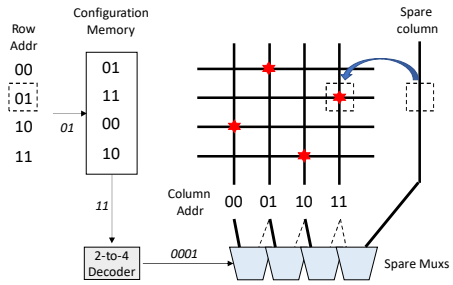

Fig. 2. Switchable Spare Columns (SSC) architecture

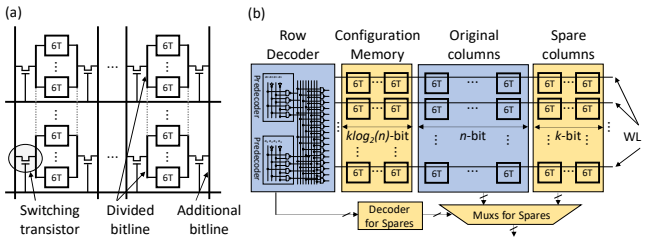

Fig. 3. (a) DBL [2] (redrawn from [13]), and (b) SSC

selectively to divided rows/columns including failing cells (DBL+DWL in Table I). DBL needs switching transistors as shown in Fig. 3(a) (redrawn from [13]) and every column needs two additional bit-lines. Since [2] does not quantify hardware overhead, we enumerated many different layouts and estimated that these two requirements increase layout width of the entire SRAM width by at least 30%, even when we use minimum metal pitch [14]. DWL also has high hardware overhead.

Our detailed analysis of DBL+DWL showed that insertion of additional transistors and lines within the SRAM array significantly decreases layout density thereby causes high overheads. Hence, we must find ways of achieving our goals without modifying the SRAM array. More concretely, this led us to the following.

Key idea 2: *We need to move the logic required to enable row-by-row selection of the location of the failing cell repaired by a spare column into some control logic outside the SRAM array.*

D. Proposed Scheme: Switchable Spare Columns (SSC)

Building on the above analysis, especially the two key ideas, we propose an approach to implement Switchable Spare Columns (SSC) at low overheads. Conceptually, SSC would be efficient since, instead of one spare column replacing an entire original column, parts of the spare column can replace parts of multiple original columns without modifying the SRAM cell array.

For the example SRAM in Fig. 1(c), we need 2 bits to store the location of one failing cell in each row. Hence we need a total of 12 bits of storage, including the four bits of the spare column itself. In addition, we need multiplexers as shown in Fig. 2. As we have five columns, including one spare column, four 2-to-1 multiplexers are required to select four out of five columns.

When the second row (i.e., row 01) is selected, the second row (also 01) of the configuration memory is also selected and the ID of the failing cell in the selected row, 11 in this case, is accessed. This ID is used to configure the multiplexors to select the the spare column instead of the failing bit at the column address 11. (To reduce the number of configuration bits, if a row is failure-free, a spare replaces the cell at column 00.) Since each row can store a different configuration for multiplexers, one spare column can repair a failure in a different column location in each row.

Fig. 3(b) shows how SSC is implemented. As this figure shows, SSC does not modify the logic in the SRAM array and avoids severe reduction in layout density. Instead, it uses extra SRAM cells in the configuration memory to store the IDs of failing cells in each row, and SRAM cells on to the right to serve as spare columns. Also, we move the circuitry required to select spare columns, namely the decoder and the multiplexors, outside of the SRAM subarray.

Overheads and Repair Capacity of SSC: Let n be the number of columns in the original SRAM array (the middle part of Fig. 3(b)). For each spare column, SSC requires one spare column (on the right in Fig. 3(b)) plus $log_2\, n$ bits to capture the information about the location of a failing cell in that row (on the left in Fig. 3(b)). Hence, if we want to repair up to k failing cells in a row, SSC requires a total of $k(log_2\, n + 1)$ bits of additional storage per row. In addition, SSC requires the circuitry to select bits from spare columns, namely the decoder and multiplexors (below the array in Fig. 3(b)).

Compared to this, ECC requires $k\ log_2\, n$ additional bits per row, which is approximately the same as our SSC. The main advantage of SSC over ECC is that SCC avoids the need for ECC decoder, which has very high area and delay overheads, especially as k, the number of cells per row that we would like to repair, increases. Hence, as shown in detailed comparison (Section IV), SSC has lower area overheads and higher yield-per-area compared to ECC. In addition, SSC has significantly lower delay overheads.

The repair capacity of SSC is identical to that of the corresponding ECC.

Generalizations of SSC: We can generalize SSC in two ways.

1) *Column grouping*: In the baseline SSC, each spare column cell (on the right of Fig. 3(b)) can replace any original SRAM cell in the corresponding row. The overheads of SSC can be decreased via column grouping, where the n columns in the original SRAM array can be partitioned into p column groups, each with n/p columns. k spare SSC columns can also be partitioned into p groups each with k/p spare SSC columns, and SSC columns in each group can be used only to repair failures in the corresponding group of original columns.

For example, Fig. 4 shows a practical size subarray with 2048 rows and 1024 columns, with SSC configuration (p, q, k) of (2, 1, 2), where each group of 512 columns can have one bit repair capacity.

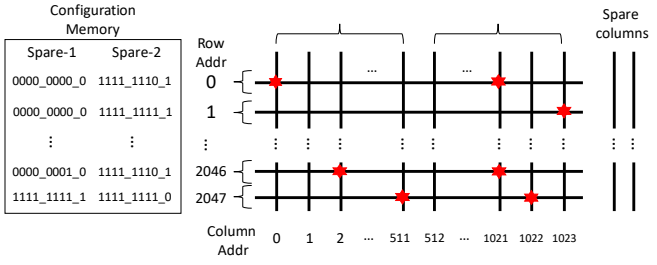

Fig. 4. A 2048x1024 subarray with SSC configured as (p, q, k) = (2, 1, 2)

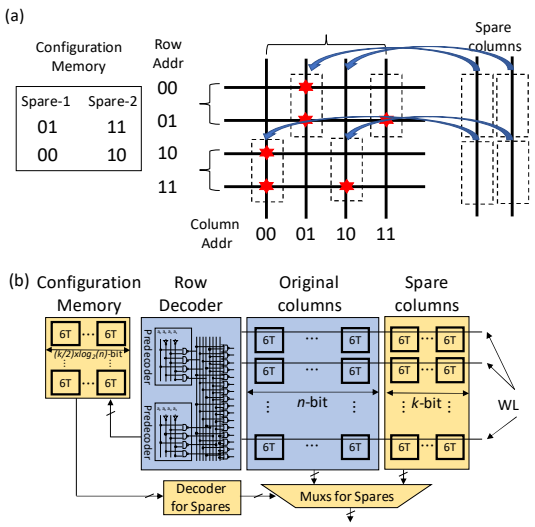

Fig. 5. SSC (a) can repair all failures where each spare column is switchable between two rows, and (b) floorplan with row grouping

Note that column grouping slightly reduces the repair capacity of SSC. At the same time, the overheads of configuration memory and multiplexors decrease. There is an analogous transformation in ECC which also reduces its overheads at slight reduction in repair capacity.

2) *Row Grouping:* We group consecutive q rows of the original array into one row group. A spare column can be used to repair cell failures in different locations in different row groups, but within the rows in one row group, the cells from one column are configured to repair failures in lock-step as shown in Fig. 5(a).

Note that row grouping slightly reduces the repair capacity of SSC. However, the overhead reduction is dramatic. For example, as shown in Fig. 5(b), the configuration memory needs only $1/q$ as many rows, hence its area overhead is much lower. Importantly, in ECC there is no analogous transformation. In other words, row grouping provides a unique advantage that makes SSC considerably more efficient compared to ECC.

SSC-Disable: We can enhance SSC by using near-zero overhead OS-level approach, namely cache block disabling [3], to disable an SRAM block with unrepaired failures. This allows SSC-Disable to use fewer spares and still maintain yield by disabling a limited number of blocks in some SRAMs (SSC-DIS in Table I).

The number of blocks disabled is kept within a limit that ensures that cache performance does not suffer. Yet, we adjust yield-per-area to account for any reduction in cache capacity by using a new metric, expected capacity per area (ECPA) [6].

Comparison with DBL+DWL: SSC-Disable has lower repair ability than DBL+DWL. However, it also has much lower overheads, since it is designed with much fewer spare columns and zero spare rows. Hence, at much lower area overheads, SSC-Disable is able to provide high YPA and ECPA, since we do not try to repair every failing cell using spares and avoid unrepaired failures by taking advantage of OS-level disabling approach.

E. Yield Analysis

1) Subarrays of SRAM

An SRAM is physically divided into subarrays and each subarray with n columns and m rows has spares to deal with fabrication failures due to variations and defects. The need to overcome higher rates of fabrication failures increases the numbers of subarrays (N_{sub}) and spare columns, and diminishes yield per area (YPA) and expected capacity per area (ECPA).

2) Yield Analysis for a Single Subarray

We calculate yield of a group, $Y_{groupcells}$, which has the number of bits in a group, G_{bit} ($= n/p \times q$). We use group size such that the probability that multiple fabrication failures occur in any row or column of a group is small. Hence, we ignore this case to calculate the following *lower bound* on $Y_{groupcells}$:

$$Y_{groupcells} = \sum_{i=0}^{k/p} \binom{G_{bit}}{i} (1 - P_{fr})^{G_{bit}-i} (P_{fr})^{i} . \quad (1)$$

Further, the yield of cells in a subarray, $Y_{subcells}$, is the product of yield of the cells of each group ($Y_{groupcells}$):

$$Y_{subcells} = Y_{groupcells}^{\,p \times m/q} . \quad (2)$$

3) Yield Analysis of SRAM

Within the defined range of N_{sub}, we select yield and area. To analyze yield and YPA of caches, we use the enhanced version of CACTI to evaluate the overall area (A_{all}) which is divided into (a) area of all SRAM cells in the cache (A_{cells}), and (b) the area of the remaining circuitry and interconnects in the cache (A_{rest}). We estimate yield and area by enumerating all possible values of N_{sub} and aspect ratios. Each SRAM cell has fabrication failure rates due to defects ($P_{fr,d}$) and variations ($P_{fr,v}$) and such cell failures are independent and identically distributed (*i.i.d.*). Further, the yield of all the SRAM cells in the cache, Y_{cells}, is the product of yield of the cells part of each subarray ($Y_{subcells}$) for all subarrays:

$$Y_{cells} = Y_{subcells}^{\,N_{sub}} . \quad (3)$$

The yield of rest of the SRAM (Y_{rest}) is calculated by assuming that the defects follow Poisson distribution:

$$Y_{rest} = e^{-A_{rest}D} , \quad (4)$$

where D is the defect density pessimistically speculated as 0.2/cm^2 [15]. Thus, yield of overall SRAM, Y_{all}, is calculated as:

$$Y_{all} = Y_{cells} \times Y_{rest} . \quad (5)$$

III. Design Process

Our goal is to find optimal cache designs in terms of yield-per-area under user-specified constraints on delay and power.

First, to meet user constraints, we use CACTI to estimate delay and power for different numbers of subarrays. Second, in the selected range of numbers of subarrays, to maximize yield-per-area, we search to find the most efficient cache designs using SSC-Disable.

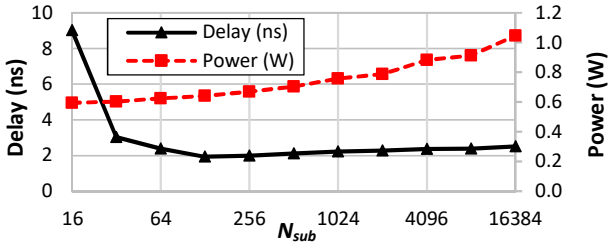

Fig. 6. Delay and leakage power per bank of 16MB cache with 64-byte cache blocks, for different numbers of subarrays (N_{sub})

Throughout this paper, we study a cache which has 16MB capacity, has 8 banks, is 16-way set associative, and has 64-byte cache blocks, for a 32mn technology. Due to its large size, to obtain acceptable delay and power consumption, the cache must be divided into subarrays.

A. Minimize delay and power

Fig. 6 shows average delay and power for different numbers of subarrays, N_{sub}. (Since leakage power is dominant for LLCs [16], in Fig. 6 we present the leakage power per bank.) The average delay and power consumption are estimated using CACTI by considering that, for each number of subarrays, aspect ratios of subarray below 1/16 and over than 16 are not used to avoid significantly high delay or power consumption.

Cache delay is considerably higher when N_{sub} is less than 64. The power consumption increases as N_{sub} increases. Based on this graph we set a power constraint that the leakage power per bank is not over 700mW.

To minimize delay and power, we can design caches with N_{sub} in the range from 32 to 1024. However, to focus on caches which achieve high yield-per-area which is highly dependent on the cache area, we use N_{sub} as 64, the number of rows (m) as 2048, and the number of columns (n) as 1024.

B. SSC and SSC-Disable

When SSC is used, we can use YPA for the primary metric to compare different approaches. However, our SSC-Disable and some previous ECC approaches [3][11] avoid some fabrication failures by disabling a part of the cache.

To effectively evaluate these caches with different ECC codes, we use expected capacity (EC) and expected capacity per area (ECPA) proposed in [6] as objective functions:

$$EC = \sum_i (1 - d_i) \times u_i / t\,, \tag{6}$$

$$ECPA = \frac{EC}{A_{all}}, \tag{7}$$

where d_i is the percentage of disabled cache bits, u_i is the number of d_i-disabled caches, and t is the total number of caches manufactured. For approaches that only deliver chips with full capacity caches, ECPA is directly proportional to YPA, while for approaches which also deliver some chips where some cache locations are disabled, ECPA adjusts YPA by a fraction which captures the average proportion of delivered caches that are disabled. Note that in all cases, our approach guarantees correct execution of every program.

Due to the complexity of yield estimation when cache block disabling is used, we estimate yield (or EC) by running Monte-Carlo simulations where 1,000 caches are generated and analyzed for each configuration and each scheme.

C. SSC/SSC-Disable Overheads for a Single Subarray

Next we estimate the overheads for SSC/SSC-Disable for each cache subarray with n columns and m rows. SSC and SSC-Disable are configured using parameters (p, q, k), where p is the number column groups ($p = 1, 2, 4, \ldots$); q is the number of rows per row group ($q = 1, 2, 4, \ldots$); and k is the the number of spare columns ($k = p \times i$, where $i = 1, 2, 3, \ldots$).

The number of columns in a column group is n/p, and each spare column for the group needs $log_2(n/p)$ bits of configuration ID. Hence we need $k/p \times log_2(n/p)$ bits for each column group, and across the p column groups, the total number of failing cell ID bits is $k \times log_2(n/p)$. We need the above number of bits for each row group. Since the subarray has m/q row groups the total number of bits in the configuration memory and spare columns is:

$$k \times log_2(n/p) \times m/q + k \times m, \tag{8}$$

where $k \times m$ is the total number of bits in the spare columns.

To account for the spare columns and configuration memory for SSC (Fig. 3) and SSC-Disable, bus width of the H-tree between subarrays increases. Also, each cache subarray has 512 multiplexers and a multiplexer signal decoder (Fig. 3). We take into account their area overheads, which increase as multiplexer sizes increase (2-to-1, 3-to-1, etc.). Since our cache is also supplemented with cache block disabling, a disabling bit is required for each cache tag to indicate whether the cache block is disbled or not. As we use SECDED only for soft errors, SECDED encoder/decoder as well as 11 check bits for each cache block are required and included in our area estimates.

TABLE III. SSC-DISABLE DESIGNS THAT MAXIMIZE ECPA AT EACH VOLTAGE

VDD (mV)	Optimal ECPA (*p, q, r*) Config.	Area	EC	ECPA
700	(1, 256, 1)	1.020	98.8%	96.9%
650	(1, 128, 1)	1.020	98.8%	96.9%
600	(1, 16, 1)	1.021	98.8%	96.7%
550	(1, 4, 2)	1.027	98.7%	96.1%
510	(1, 4, 3)	1.030	98.2%	95.3%
480	(1, 2, 5)	1.050	98.2%	93.5%
450	(2, 2, 14)	1.086	97.9%	90.2%

D. ECPA Maximization

Table III shows ECPA maximization (p, q, k) configuration using SSC-Disable among the design space for each VDD value. These results show that it is very effective for a spare column to be switched between rows, which results in low area overhead (8.6% for VDD = 450mV) and high ECPA (90.2% for VDD = 450mV).

In the next section, we compare our SSC-Disable with recently proposed ECC approaches in terms of area, delay, and ECPA.

IV. EFFECTIVENESS OF OUR APPROACH

We show that our new switchable spare columns are extremely efficient when combined with near-zero overhead OS-level approach, cache block disabling. We now compare our approach with previous approaches. We show that, for ECPA maximization as well as delay and power constraints imposed by designers, for high failure rates we face during low-voltage operation, our new SSC-Disable approach is more efficient than the most effective prior approaches, namely the ECC-based approaches. Further, we show that SSC is helpful on previous approaches (SSC-ECC-DIS in Table I).

A. Comparision with ECC-based approaches

Previous ECC-based approaches suggested the use of a limited level of the ECC's error correcting capability to correct errors due to variations, leaving the remaining ECC capability to correct soft errors (ECC-ONLY in Table I). Most effective approaches disable cache blocks where ECC cannot correct all fabrication failures (ECC-DIS in Table I).

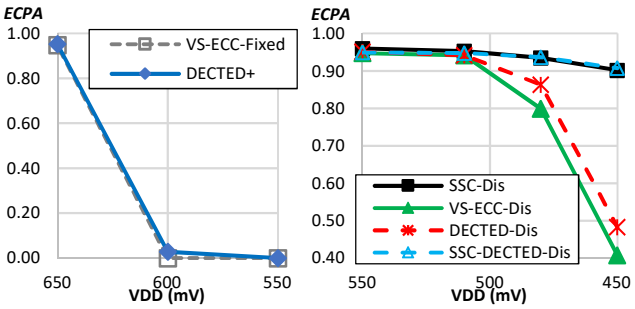

Fig. 7. ECPA of different schemes

Fig. 7 shows ECPA of caches with (1) ECC-ONLY approaches, namely DECTED+ [11] and VS-ECC-Fixed [3]; and (2) ECC-DIS approaches, namely DECTED-Disable [11] and VS-ECC-Disable [3], and (3) SSC equipped approaches, namely SSC-Disable and SSC-DECTED-Disable. The area of the cache configured as $N_{sub} = 64$ is normalized as 1 and used for measuring YPA and ECPA.

Our above method identifies as optimal a cache with SSC-Disable scheme with configuration (p, q, k) = (2, 2, 14) and a cache with SSC-DECTED-Disable with configuration (1, 2, 7). SSC-Disable and SSC-DECTED-Disable achieve close to 90% ECPA at 450mV. (SSC-Disable has slightly lower ECPA compared to SSC-DECTED-Disable but has much lower delay since it uses SECDED is used only to repair soft errors.) In contrast, both DECTED+ and VS-ECC-Fixed approaches fail when VDD is 600mV or lower (i.e., ECPA becomes 0), and other previous ECC approaches with disabling provide much lower ECPA, namely 41-50%, at 450mV. Table IV provides more information, including area, EC, ECPA, and delay, for each approach at the desired low-voltage, 450mV.

Simply, SSC-Disable provides much higher EC and ECPA of 90.2%, compared to 48.2% for the best of the previous ECC-DIS schemes, namely DECTED-Disable.

B. *Delay Benefits of SSC-Disable*

Besides high ECPA of 90.2%, proposed cache with SSC-Disable is significantly faster than other three schemes. Encoding/decoding ECCs causes high delay for each read/write operation. In a cache with VS-ECC-Disable at 450mV, 23% cache blocks use 4EC5ED which requires 15 cycles [3]. In a cache with DECTED-Disable and SSC-DECTED-Disable, every block must encode/decode DECTED codes where delay is more than twice of the delay of SECDED codes [17]. While the proposed SSC-Disable scheme has additional delay due to spare multiplexers, the access time is still within one cycle. This shows that the proposed switchable spare columns are dramatically beneficial, especially when low-VDD operation is necessary to meet power constraints. Finally, the degradation in terms of CPI (clocks per instruction) for the user program running on the processor, for the level of capacity loss due to cache block disabling, is negligible [3][6].

V. Conclusion

We propose SSC-Disable, a completely new approach to dramatically improve effectiveness of spares in conjunction with OS-level cache block disabling. We also propose a systematic method to optimally harness SSC-Disable for cache designs to increase YPA and ECPA considerably. We demonstrate that, compared to the best previous ECC-disable approaches, our SSC-Disable approach significantly improves EC, ECPA, and performance, especially in low power modes, e.g., at 450mV supply voltage. Overall, we demonstrate a very efficient and useful approach for implementing switchable spare columns.

TABLE IV. Comparison of LLC Cache with Different Schemes in Terms of Area, EC, and ECPA at VDD = 450mV

Scheme	Norm. Area	EC	ECPA	Delay (cycle)
DECTED+[§]	1.037	0.0%	0.0%	N/A
DECTED-Disable[§]	1.038	50.0%	48.2%	4
VS-ECC-Fixed	1.042	0.0%	0.0%	N/A
VS-ECC-Disable	1.043	42.5%	40.8%	2 for 20% 15 for 23% N/A for 57%
SSC-DECTED-Disable[§] (p, q, k) = (1, 2, 7)	1.078	97.8%	90.7%	4
SSC-Disable (p, q, k) = (2, 2, 14)	1.086	97.9%	90.2%	2

[§]DECTED+, [§]DECTED-Disable, and [§]SSC-DECTED-Disable uses up to one bit error correcting capability of DECTED to repair fabrication failures.

References

[1] P. Papavramidou, "Memory repair for high fault rates," in *2016 IEEE International Test Conference (ITC)*, 2016, pp. 1–10.

[2] S.-K. Lu, Y.-C. Tsai, C. Hsu, K.-H. Wang, and C.-W. Wu, "Efficient built-in redundancy analysis for embedded memories with 2-D redundancy," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 1, pp. 34–42.

[3] A. R. Alameldeen, I. Wagner, Z. Chishti, W. Wu, C. Wilkerson, and S. L. Lu, "Energy-efficient cache design using variable-strength error-correcting codes," in *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, 2011, pp. 461–471.

[4] J. P. Kulkarni, K. Kim, and K. Roy, "A 160 mV Robust Schmitt Trigger Based Subthreshold SRAM," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 10, pp. 2303–2313, Oct. 2007.

[5] H. Pilo, C. Barwin, G. Braceras, C. Browning, S. Lamphier, and F. Towler, "An SRAM Design in 65-nm Technology Node Featuring Read and Write-Assist Circuits to Expand Operating Voltage," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 4, pp. 813–819, Apr. 2007.

[6] H. Hsuing, B. Cha, and S. K. Gupta, "Salvaging chips with caches beyond repair," in *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2012, pp. 1263–1268.

[7] S. E. Schuster, "Multiple word/bit line redundancy for semiconductor memories," *IEEE Journal of Solid-State Circuits*, vol. 13, no. 5, pp. 698–703, Oct. 1978.

[8] 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," in *35th International Symposium on Computer Architecture, 2008. ISCA '08*, 2008, pp. 203–214.

[9] H. L. Kalter *et al.*, "A 50-ns 16-Mb DRAM with a 10-ns data rate and on-chip ECC," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 5, pp. 1118–1128, Oct. 1990.

[10] M. Nicolaidis, N. Achouri, and L. Anghel, "A diversified memory built-in self-repair approach for nanotechnologies," in *22nd IEEE VLSI Test Symposium, 2004. Proceedings.*, 2004, pp. 313–318.

[11] M. Zhang, V. M. Stojanovic, and P. Ampadu, "Reliable Ultra-Low-Voltage Cache Design for Many-Core Systems," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 12, pp. 858–862, Dec. 2012.

[12] D. Roberts, N. S. Kim, and T. Mudge, "On-chip cache device scaling limits and effective fault repair techniques in future nanoscale technology," *Microprocessors and Microsystems*, vol. 32, no. 5, pp. 244–253, Aug. 2008.

[13] C.-H. Hsu, S.-K. Lu, and S.-Y. Kuo, "Novel fault-tolerant techniques for high capacity RAMs," in *Proceedings 2001 Pacific Rim International Symposium on Dependable Computing*, 2001, pp. 11–18.

[14] N. H. E. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. Pearson, 2010.

[15] M. Neisser and S. Wurm, "ITRS lithography roadmap: 2015 challenges," *Advanced Optical Technologies*, vol. 4, no. 4, pp. 235–240, 2015.

[16] M. T. Chang, P. Rosenfeld, S. L. Lu, and B. Jacob, "Technology comparison for large last-level caches (L3Cs): Low-leakage SRAM, low write-energy STT-RAM, and refresh-optimized eDRAM," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, 2013, pp. 143–154.

[17] R. Naseer and J. Draper, "DEC ECC design to improve memory reliability in Sub-100nm technologies," in *2008 15th IEEE International Conference on Electronics, Circuits and Systems*, 2008, pp. 586–589.