Row/Column Redundancy to Reduce SRAM Leakage in Presence of Random Within-Die Delay Variation

Goudarzi, Maziar
System LSI Research Center, Kyushu University

Ishihara, Tohru
System LSI Research Center, Kyushu University

Row/Column Redundancy to Reduce SRAM Leakage in Presence of Random Within-Die Delay Variation

Maziar Goudarzi, Tohru Ishihara
System LSI Research Center, Kyushu University
3-8-33 Momochihama, Sawara-ku, Fukuoka, Japan
{goudarzi, ishihara}@slrc.kyushu-u.ac.jp

ABSTRACT

Traditionally, spare rows/columns have been used in two ways: either to replace too leaky cells to reduce leakage, or to substitute faulty cells to improve yield. In contrast, we first choose a higher threshold voltage (V_{th}) and/or gate-oxide thickness (T_{ox}) for SRAM transistors at design time to reduce leakage, and then substitute the resulting *too slow* cells by spare rows/columns. We show that due to within-die delay variation of SRAM cells only a few cells violate target timing at higher V_{th} or T_{ox}; we carefully choose the V_{th} and T_{ox} values such that the original memory timing-yield remains intact for a negligible extra delay. On a commercial 90nm process assuming 3% variation in SRAM cell delay, we obtained 47% leakage reduction by adding only 5 redundant columns at negligible area, dynamic power and delay costs.

Categories and Subject Descriptors

B.3.1 **[Semiconductor Memories]**: *static memory (SRAM)*, C.5.4. [**Computer system implementation**]**:** *VLSI systems.*

General Terms

Algorithms, Design, Experimentation, Performance.

Keywords

Power reduction, leakage power, SRAM, process variation, delay variation, random variation, redundancy, spare row, spare column.

1. INTRODUCTION

With every new technology node, the share of leakage in total power consumption of cache and other SRAM-based memories increases considerably since dynamic power decreases and leakage increases with technology scaling. Among major leakage components, subthreshold leakage is dominant in cache and other SRAM-based memories in nanometer technologies [19] but gate leakage also effectively increases at very thin gate-oxides. The naïve solution for reducing SRAM leakage is to increase transistor threshold voltage (V_{th}) and/or gate-oxide thickness (T_{ox}) to exponentially reduce respectively subthreshold and gate leakage powers, but this negatively affects SRAM access delay. Traditionally, access delay of all SRAM cells of a memory module on a chip have been the same, and hence, choosing a higher V_{th} or T_{ox} would have resulted in almost all of the cells violating the original timing, and hence, row/column redundancy could not afford replacing them. In sub-90nm technologies, however, same-sized SRAM cells may have different delays even within the same die (i.e. *within-die delay variation*); we show that in this case, only a few cells violate target timing at higher V_{th}, T_{ox}, and hence we propose to replace these timing-violating SRAM cells with redundant rows or columns of SRAM.

Process variation results in changes in circuit parameters (such as transistor gate-length, threshold voltage, circuit delay and the like) during manufacturing such that the manufactured device may differ from the designed device in some features. Within-die variation means variations in circuit parameters within a single die, which means similar circuits in different parts of the die may have different features. Die-to-die variations, on the other hand, are the variations that happen from one die to the other, from wafer to wafer, and from wafer lot to wafer lot. Our focus is *uncorrelated random within-die variation of SRAM cell delay*. While die-to-die variations have been observed for several years, within-die variations have more recently started to intensify [9]. Empirical studies in [20] show that 3.54% random within-die delay variation is observed for a single logic-element (roughly equivalent to a single SRAM cell) of 90nm FPGAs. Furthermore, this within-die variation is predicted to rise when further approaching atomic sizes with every new technology node [9][20]. Within-die variations are commonly modeled by Gaussian distribution [1] which also well matches empirical results in [20].

Leakage sensors and fuses have been used in [11] to detect abnormal leakages in SRAM cells and to cut-off power lines of the corresponding rows and columns to reduce leakage. Row/column redundancy has long been used to repair faults in RAM memories [13] and has more recently become inevitable in high-density SRAM memories to obtain acceptable manufacturing yield [13]. In cache memories, several previous works address improving timing-yield in presence of process variation by proposing process-tolerant cache architectures [1], [21] and code-placement compiler techniques [10], but they actually reduce the useful capacity of the cache by marking and avoiding to use too-slow cache lines. Although [10] provides a solution to mitigate the performance impact, it demands a per-chip different binary executable. Other highly-cited works exist to reduce cache static power [2][12][6], but they do not consider process-variation.

In this paper, we propose an optimization technique for SRAM-based memory design that is applied at design and manufacturing time of the memory-containing chip and reduces leakage power, by choosing a higher V_{th} and/or T_{ox}, at the cost of extra area for redundant rows/columns of SRAM cells. We *(i)* keep V_{DD} untouched (to avoid its quadratic impact on dynamic power), and *(ii)* based on the given number of redundant rows/columns, we optimally choose a higher V_{th} and T_{ox} at memory design time (so as to exponentially reduce subthreshold leakage as well as gate leakage) such that the original memory capacity and timing-yield are maintained. A negligible delay overhead is imposed due to the needed programmability of row/column decoders to remap slow rows/columns to spare ones. Also, longer word-lines or bit-lines (respectively corresponding to column or row redundancies) result in a marginal higher dynamic power per access. Results of our SRAM design on a commercial 90nm process shows that 2% redundancy (5 columns added to a 256×256 memory) reduces leakage by 47% while dynamic power, delay, and area increase by less than 3%; timing-yield and memory capacity are kept intact.

In the rest of this paper, Section 2 reviews related works, Section 3 presents our approach, Section 4 analytically models timing-yield, Section 5 formulates the optimization problem and presents algorithm. Section 6 provides experimental results and finally Section 7 summarizes and concludes the paper.

2. RELATED WORKS

Dual V_{th} and dual T_{ox} values are used in [3] to reduce leakage by setting appropriate V_{th} and T_{ox} for each individual SRAM cell based on its closeness to the sense amplifier. We use the same V_{th} and T_{ox} for all cells and we use spare rows/columns.

Cache memories are among the most widely used SRAM-based memories. Turning off *unused* parts of the cache [2][12][18] or putting them in a low-energy "drowsy" mode using two different supply voltages [6] are well known techniques to reduce cache leakage, but they cannot handle the cache parts *being accessed*. Source biasing by a virtual ground [7] improves leakage in sleep mode, but cannot be used in the active mode. Reverse body biasing [5] or choosing a higher V_{th} at manufacturing time can effectively reduce leakage, but this increases cells delay and results in lower performance and/or reduced timing-yield. Forward body biasing during active-mode [16] and dynamic V_{th} control [14] can improve delay, but naturally they increase leakage in the active mode. The above-mentioned delay impact can be compensated by increasing V_{DD} in line with the increase in V_{th}, but this results in a quadratic increase in dynamic power [23]. We keep the original V_{DD} although we use a higher V_{th}; instead, we add extra rows/columns to compensate for the rows/columns that violate target delay due to the increased V_{th}. Using spares to repair manufacturing defects [8][11][13] is not new, but the way we use it (i.e. replacing *slow, less-leaky,* cells) is new: [11] eliminates *too leaky* cells by cutting their power lines while we replace *too slow* cells; [8],[13] use spares to repair faulty memories and improve yield; they do not concern power consumption.

Shirvani et al. [21] describe a programmable address decoder that redirects accesses to slow/faulty cache lines to other lines in the same cache-set. Another simpler technique to bypass slow cache lines [10] uses an unused combination of flag bits to mark the cache-line as faulty and to eliminate it from normal operation of the cache. These are not applicable to general SRAM memories other than caches.

Ozdemir et. al [17] propose to turn off delay-violating and too leaky cache ways or lines. They also propose to allow different parts of the cache be accessed at different latencies. Although these techniques improve chip yield, they affect cache capacity and/or speed while our technique keeps the original yield with no capacity impact and negligible speed overhead. We reduce leakage by choosing higher T_{ox} and V_{th}, not by turning off leaky parts. Thus, memory capacity and cell array speed remain intact.

Meng and Joseph [15] consider within-die leakage variation and extend *selective cache ways* [2] by starting from the leakiest cache way when disabling the ways that are not used by the application. As above, this reduces cache capacity, and furthermore, we do not reduce leakage by disabling parts of the cache.

3. OUR APPROACH

Motivational Example. The left-hand side of Figure 1 shows a 4-column SRAM memory implemented in the high-performance (HP) manufacturing-process option of a commercial 90nm process with V_{th}=270mv. The last row of SRAM cells demonstrates within-die delay variation: SRAM cells have different latencies. At the right-hand side, middle-performance (MP) process option, which employs 50mv higher V_{th}, is adopted to implement the memory.

Figure 2 gives SPICE simulation results of leakage vs. delay of standard 6T SRAM cells using the same 90nm process and shows that delay increases and leakage decreases by raising V_{th},T_{ox}. Thus, in the right-hand side of Figure 1 cell latency increases (compare the two last rows) and one of the cells violates the target delay of 540ps. To compensate for the delay-violating SRAM cell, one spare column of SRAM cells is added to substitute the slow column (the red-bordered column that contains the delay-violating cell in Figure 1). Note that the *slowest* cell (which is also the least leaky one) is replaced, not the *leakiest* cell as in [11]. (To further reduce leakage, one can also replace leaky cells in addition to slow ones, but we do not do it in this paper.) Power supply of the slow column is cut using power-gating [18] or fuse-cutting [11]. Consequently, the memory latency is kept at the original 540ps and the memory capacity is also the same as before. The choice of higher V_{th} reduces the leakage by 36% in this example. It is noteworthy that additional circuitry is required to remap slow rows/columns to spare ones. This adds negligible overheads to delay and dynamic power per memory access.

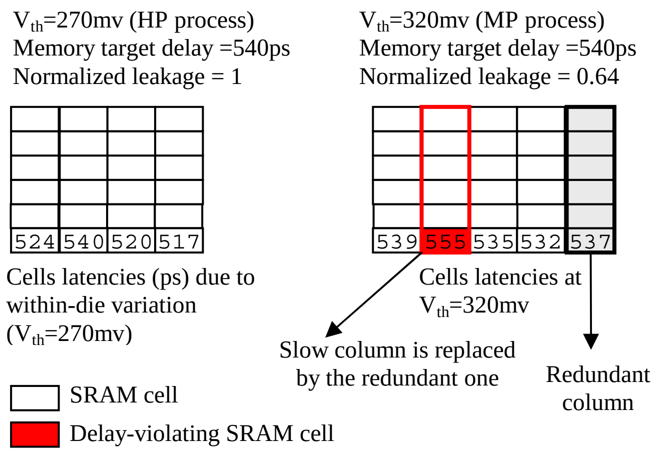
V_{th}=270mv (HP process)
Memory target delay =540ps
Normalized leakage = 1
V_{th}=320mv (MP process)
Memory target delay =540ps
Normalized leakage = 0.64
524 540 520 517
539 555 535 532 537
Cells latencies (ps) due to within-die variation (V_{th}=270mv)
Cells latencies at V_{th}=320mv
Slow column is replaced by the redundant one
Redundant column
SRAM cell
Delay-violating SRAM cell

Figure 1. One spare column is added to a 4-column memory to let choose 50mv higher V_{th}, and hence reduce leakage. The only resulting delay-violating cell is replaced by the redundant column. The delay-violating column is disconnected from power supply by power-gating [18] or fuse-cutting [11].

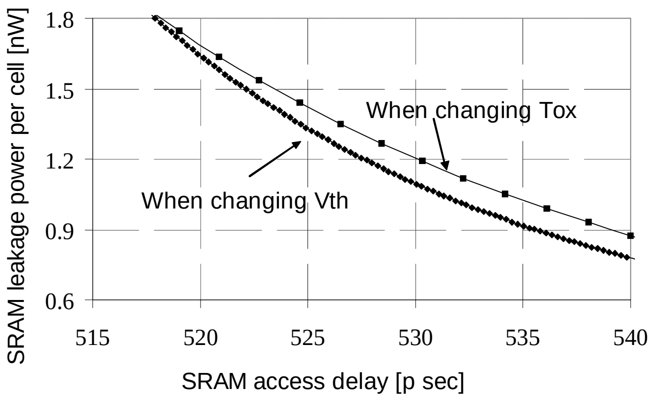

Figure 2. Leakage power vs. access-delay of a single SRAM cell when raising V_{th} and T_{ox} (from left to right) in a commercial 90nm process technology.

Optimization Flow. Figure 3 shows the outline of the operations in our proposed approach. We propose a design- and manufacturing-time optimization that determines the best values of T_{ox} and V_{th} for the transistors of the memory SRAM cells. Semiconductor manufacturers typically provide choice of a few V_{th} and T_{ox} values to designers. In addition, body biasing [14] can be used to further tune V_{th}. We use these choices for optimization. The original memory organization (i.e. number of rows and columns), the acceptable number of redundant rows/column (based on acceptable area overhead), and finally the process-technology characteristics (i.e., delay variability and initial mean delay of SRAM cells caused by within-die variation as well as leakage-delay curves of the cells at various V_{th} and T_{ox} values) are the inputs of the optimization program. The manufacturing options of T_{ox} and V_{th} are chosen by our optimization technique such that original memory capacity and timing-yield remain intact. The chosen manufacturing options of T_{ox} and V_{th} are handed over to the manufacturer for chip fabrication. The produced chips are then tested offline to detect and mark slow SRAM cells. If the number of such delay-violating cells exceeds the number of redundant rows/columns, the chip is considered faulty and contributes to loss of yield. Finally at the repair phase, rows/columns containing delay-violating SRAM cells are replaced by redundant ones and are disconnected from power supply.

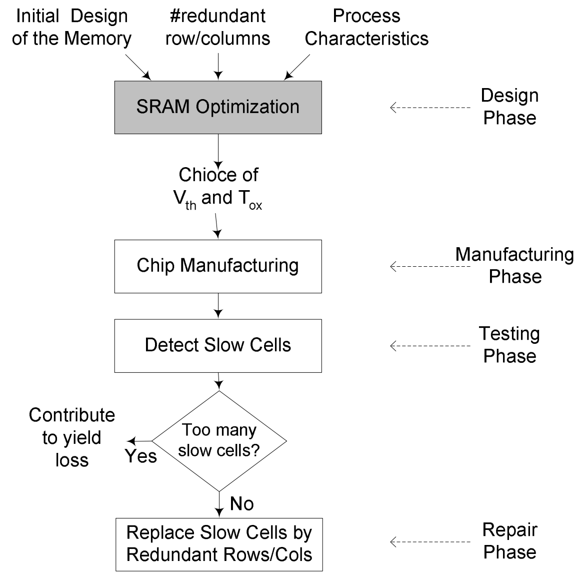

Figure 3. Big picture of our proposed approach.

4. ANALYTICAL YIELD MODEL

The Gaussian distribution models within-die variations [20][1]. We define the following notations:

- μ_d: the original mean delay of SRAM cells.
- σ_d: the original standard deviation of delay of SRAM cells.
- ***D:*** the target delay of the SRAM array.
- ***R:*** the number of rows of SRAM in the memory module.
- ***C:*** the number of columns of SRAM in the memory module (excluding redundant columns).
- ***N:*** the number of redundant columns (we suffice to column redundancy. Row redundancy can be similarly analyzed).
- N_{slow}: the average number of delay-violating SRAM cells in a memory module.
- P_{cell}: the probability for a single SRAM cell to meet target delay; i.e. the probability that its delay is less than or equal to D.
- Y_0: timing-yield of cell array without redundancy.
- Y_N: timing-yield of cell array with *N* redundant columns.

Assuming Gaussian distribution for delay of SRAM cells within a single die, probability of a SRAM cell meeting target delay of the cell array is given by:

$$P_{cell} = \Pr[x \le D] = \int_{-\infty}^{D} f(x)dx$$
$$f(x) = \frac{1}{\sigma_d\sqrt{2\pi}} e^{\frac{-(x-\mu_d)^2}{2\sigma_d^2}} \qquad (1)$$

where *f(x)* is the Probability Density Function (PDF) of Gaussian distribution.

The average number of delay-violating cells in the entire cell array when including *N* redundant columns is:

$$N_{slow} = (1 - P_{cell}) \times R \times (C + N) \qquad (2)$$

because (1-P_{cell}) is the probability of a cell violating the delay, and *R×(C+N)* is the total number of cells in the memory.

The memory module without redundancies contributes to yield only if all its comprising cells satisfy the target delay. Thus, the timing-yield of the original memory module is:

$$Y_0 = P_{cell}{}^{(R\times C)} \qquad (3)$$

In presence of *N* redundant columns, the memory module still contributes to yield as long as at most *N* columns out of the *C+N* available columns violate target delay.

$$Y_N = \sum_{i=0}^{N} \binom{C+N}{i} \times \left(1 - P_{cell}{}^{R}\right)^{i} \times \left(P_{cell}{}^{R}\right)^{C+N-i} \qquad (4)$$

(As a hint to understanding the formula, note that $P_{cell}{}^{R}$ gives the probability that all cells in a column satisfy the target delay.)

5. PROBLEM FORMULATION

We define the following additional notations:

- ***var***: the amount of delay variability (i.e. *var* = $3\sigma_d/\mu_d$)
- P_L: the average leakage power of the memory (*average* power, due to variability).
- V_{th}: the optimal value for the V_{th} of SRAM transistors.

***T_{ox}*:** the optimal value for the T_{ox} of SRAM transistors.

The problem can be formally defined as follows:

"For a given memory (i.e. R and C) with a given delay distribution (i.e. μ_d, var), given target timing-yield (Y_0), and given number of redundant columns (N), minimize the average leakage power of the memory (i.e. P_L) by choosing V_{th} and T_{ox} such that the target delay, D, is kept unchanged."

It is noteworthy that Eq. 1 and 3 show that for a given memory organization and process technology, the target delay (*D*) and target timing-yield (Y_0) correspond to each other and either of them can be equally chosen as given while the other one is kept unchanged. Here we have chosen Y_0 as given.

Algorithm. The following straightforward procedure is used:

```
Procedure: OptimizeMemoryDesign()
Inputs: (var, μd: characteristics of the original
                  delay distribution of SRAM cells),
        (R, C: original memory dimensions),
        (Y0: Target timing-yield of memory),
        (N: number of spare columns)
Output: (chosen Vth, Tox).
1 calculate Pcell of original memory using Eq. 3 and
  given R, C, and Y0.
2 σd = var × μd / 3
3 calculate D using Eq. 1 and given original μd and
  above-calculated Pcell and σd.
4 set YN=Y0 and then calculate new Pcell of memory
  with redundancy using Eq. 4 and given R, C, N.
5 modify Eq. 1 by setting (new σd) = var × (new μd)/3
6 calculate new μd using just-modified Eq. 1, the
  above-calculated D, and new Pcell.
7 return Vth and Tox corresponding to the new μd from
  data tables illustrated in Figure 5.
```

The procedure can be understood by the illustration in Figure 4 which shows delay distribution of SRAM cells before (left-hand side) and after (right-hand side) increasing V_{th} and/or T_{ox}. Each diagram shows the probability density function (PDF) of cells delay distribution; the horizontal axis is delay of SRAM cells and the vertical axis is the probability density (i.e. *f(x)* in Eq. 1 where *x* is the cell delay). The gray area gives the probability that cell delay is less than *D*; i.e. the gray area is P_{cell}.

By adding spare rows/columns, a smaller P_{cell} is enough to obtain the same timing-yield $Y=Y_0=Y_N$ for the entire memory; thus ($new\ P_{cell}$)<P_{cell}, and hence, the distribution can be shifted to the right if *D* is kept constant. This means that all SRAM cells can be delayed by ($new\ \mu_d$)-μ_d time units, which means less total leakage since slower cells have lower leakage (see Figure 2).

The above procedure computes P_{cell} before (line 1) and after (line 4) adding spares; lines 2 and 3 are the needed immediate steps. Then at line 6, the new mean delay of cells (i.e. *new* μ_d) is computed; line 5 is needed to calculate new delay standard deviation (σ_d) since at higher V_{th}/T_{ox}, the σ_d increases but delay variability *var* remains constant.

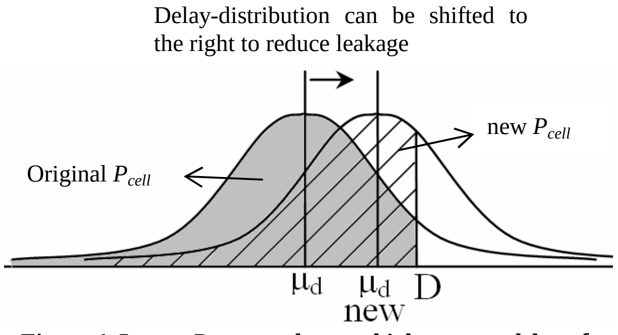
Delay-distribution can be shifted to the right to reduce leakage
Original P_{cell}
new P_{cell}
μ_d
μ_d new
D

Figure 4. Lower P_{cell} translates to higher mean-delay of SRAM cells resulting in lower leakage.

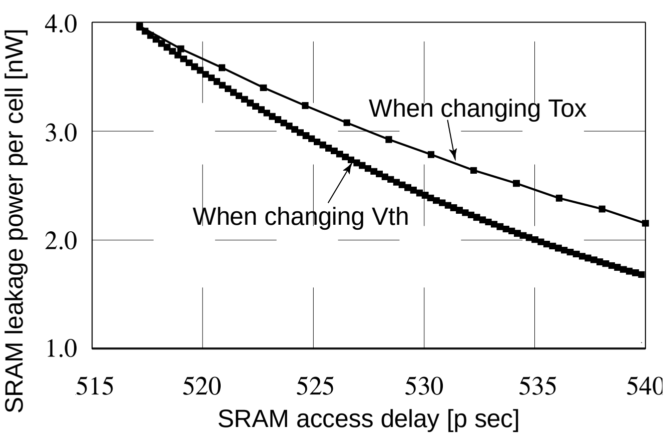

Figure 5. Average leakage power of a single SRAM cell vs. its average access-delay in a commercial 90nm process in presence of 5% delay variation.

Finally line 7 obtains the new V_{th} and T_{ox} corresponding to the now-shifted delay distribution. To avoid having to run SPICE Monte Carlo simulations at every execution of the algorithm, we generated tables that give average leakage vs. average delay of a single SRAM cell at various delay variabilities (e.g. Figure 5 depicts such table at 5% delay variability). To do this, we first ran several SPICE simulations in a commercial 90nm process using various values for V_{th} and T_{ox} of the transistors and obtained leakage per SRAM cell (this results in Figure 2), then generated a large number of Gaussian random values as SRAM cell delay, and finally averaged their corresponding leakages; this results in Figure 5 for 5% delay variation.

Thus instead of running a new Monte Carlo simulation at every execution of the algorithm, we consult the above mentioned leakage-delay curves: given the new mean delay (new μ_d, line 6), the corresponding points in the curves are found and their associated V_{th} and T_{ox} are obtained and returned (line 7).

6. EXPERIMENTAL RESULTS

Methodology. We designed a 256×256 SRAM memory in a commercial 90nm technology and implemented the corresponding row decoder and column multiplexer at gate-level using a 90nm standard cell library (names undisclosed due to NDA). Power and delay of SRAM cells are obtained from SPICE simulation and that of decoder and multiplexer are reported by Synopsys Power Compiler tool. The dynamic and static power and delay results are given in Table 1. Leakage power and delay of SRAM cells are the SPICE simulation results before considering delay variations.

Table 1. Power and delay of our 90nm 256×256 memory.

	Dynamic energy per access (pJ)	Leakage power (μW)	Delay (ps)
8×256 row decoder	0.12	0.367	200
256×32 col. mux.	1.08	0.523	170
SRAM cells	11.07	7.10*	642*

* Not considering delay variations.

We focus on the leakage of the cell array since leakage of other parts is an order of magnitude smaller. To obtain leakage power of the cell array in presence of delay variation, first new μ_d is calculated (see Section 5) and then its corresponding average leakage per cell is obtained from leakage-delay curves (Figure 5) and is multiplied by the number of SRAM cells in the memory module. Since delay-violating and unused spare columns are disconnected from power supply, total number of powered-ON SRAM cells is the same as the original memory.

Experiment results. All experiments basically correspond to a 256×256 cell array in 90nm process with 3% delay variability, 99% target timing-yield, and 642ps target delay. Effect of changing each factor is evaluated separately.

At 3% variation and for a target timing-yield of 99%, Figure 6 shows the obtained leakage savings vs. the number of redundant columns. Table 2 gives P_{cell} and μ_d in each case. Over 38% leakage can be saved by adding only a single spare column.

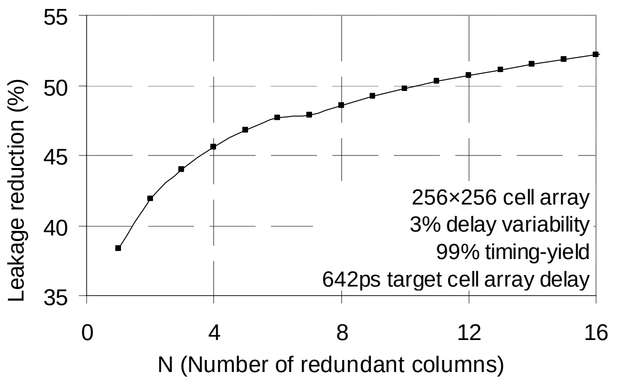

Figure 6. Leakage reduction vs. number of redundant columns at 3% variation.

More redundant columns allow to further increase V_{th}/T_{ox} and hence to improve leakage reduction, but the slope of the diagram reduces at higher *N* and saturates around 55% leakage reduction using 32 spares. This can be explained by looking at Figure 4: given the same amount of decrease in P_{cell}, only a smaller shift to the right is possible with adding each spare column; in other words, a bigger shift is possible when *D* is closer to the tail end of the distribution, and hence, since larger delay-shift means larger leakage reduction, leakage is more reduced in the original case (no redundancy) than after adding first spare row/column, and so forth.

Table 2. Reductions in required P_{cell} and the resulting increase in μ_d due to the redundancies (corresponding to Figure 6).

# Spare columns (N)	P_{cell}	μ_d *(mean delay) (ps)*	Leakage reduction (%)
0 (original)	0.999999962000000	552.82	-
1	0.999984762583658	570.61	38.39
2	0.999969680992248	573.05	41.90
4	0.999939865846154	575.59	45.61
8	0.999881590787879	578.22	48.58
16	0.999770182588235	580.93	52.18
32	0.999565934222222	583.67	54.79

The algorithm execution time is just a fraction of a second on a Xeon 3.80GHz processor with 2MB of cache and 3.5GB of memory, but it took us a week to run all SPICE simulations on the same machine so as to obtain the tables used in step 7 of the algorithm (i.e. Figure 5); however, noting that this is done only once for entire manufacturing of all the memory chips, this execution-time would not be a limiting factor.

Overheads. The area overhead can be roughly estimated by the ratio of spare columns to the original columns since additional area in the multiplexer and decoder are small compared to the area occupied by spare SRAM cells. Since timing-violating rows/columns are still connected to bit/word lines (although disconnected from power), more time and dynamic energy is required for driving bit/word lines. The row/column multiplexer also slows down and consumes larger dynamic power. Since row/column redundancy is not new and is a well-known technique in SRAM-based memory design, we did not evaluate its delay, dynamic power, and area overhead by experiments; instead, based on estimations by CACTI ver. 4.2 [4] the overheads for adding 8 spare columns to a 256×256 SRAM memory are estimated as 0.17% delay, 1.30% dynamic read energy, 1.51% dynamic write energy, and 2.23% area overhead.

Effect of higher variation. At higher variations, the technique becomes more effective. Table 3 shows the results at 5% delay variation. Since delay variation is expected to rise with technology scaling [9], leakage saving results improve in future technologies.

Table 3. Leakage savings at 5% variation (99% target timing-yield, 642ps target SRAM cell delay)

#spare columns (N)	1	5	10	15	20	25
Leakage reduction (%)	59.36	70.18	73.70	75.34	76.42	77.12

Effect of target timing-yield. Table 4 shows leakage savings at various target timing-yields. More leakage can be saved when targeting higher timing-yields. This can again be explained by the position of *D*: the closer is *D* to the tail end of the distribution, the larger is the possible delay shift, and hence, the leakage reduction. Higher target timing-yield requires a higher P_{cell} which translates to pushing *D* farther toward tail end of the delay distribution.

Table 4. Leakage savings at various timing-yields (3% variation, 642ps target SRAM cell delay, 5 spare column)

Target timing-yield	75	80	85	90	95	99
Leakage reduction (%)	30.88	32.46	34.39	35.88	39.80	46.84

Effect of target delay. Table 5 shows leakage savings when changing target delay. Reducing target delay while maintaining other factors results in shifting the delay distribution toward smaller delays. Since the leakage-delay curve has a sharper slope at lower delays (see Figure 5), this results in a bigger saving given the same amount of delay shift.

Table 5. Leakage savings at various target delays (3% variation, 99% timing-yield, 5 spare columns)

Target delay (ps)	650	642	630	620	610	600
Leakage reduction (%)	44.61	46.84	48.12	51.31	55.23	59.37

Effect of memory size. Table 6 shows the obtained leakage savings by adding 5 redundant columns to memories with different sizes. Benefits of the technique slightly reduce with memory size. For bigger memories, the number of delay-violating cells increases (see Eq. 2), and hence, the same amount of redundancy is less effective.

Table 6. Leakage savings for various memory sizes (3% variation, 99% target timing-yield, 642ps target SRAM cell delay, 5 redundant columns)

Memory size	2KB	8KB	32KB	128KB
Leakage reduction (%)	47.85	46.84	45.60	44.75

7. SUMMARY AND CONCLUSION

We presented a design- and manufacturing-time optimization technique to reduce leakage power of SRAM-based memories (such as cache and scratchpad memory) in presence of within-die delay variation. We choose a higher V_{th} and/or T_{ox} for SRAM transistors to reduce subthreshold and gate leakage, and substitute the resulting timing-violating SRAM cells with redundant rows/columns; moreover, higher V_{th} and T_{ox} improve static noise margin (SNM) and memory reliability [22]. Within-die delay variation is the main enabler behind this technique since due to it, delays of most cells are far below the target delay of the memory, and hence, only a few cells violate the target timing when choosing a higher V_{th}/T_{ox}, and they can be compensated by a few redundant rows/columns. Experimental results on a commercial 90nm process (assuming 3% within-die delay variation [20]) show that by adding only 5 spare columns of SRAM to a 256×256 memory (i.e. <3% area, delay, and dynamic power penalty) it is possible to reduce leakage by 47%. With higher variation expected to occur with technology scaling [9], even more leakage can be saved with the same amount of redundancy (e.g. 70% leakage reduction at 5% delay variation).

Our assumption of Gaussian delay distribution can be changed to more complex ones considering *systematic* and *correlated random* variations as well; as long as D can be computed for a given P_{cell} and μ_d (step 3 of our algorithm), and μ_d can be computed for a given P_{cell} and D (step 6 of algorithm), our same algorithm can be used with any delay distribution. We leave evaluating more complex delay distribution models to future work.

8. ACKNOWLEDGMENTS

This work is supported by VDEC, The University of Tokyo with collaboration of STARC, Panasonic, NEC Electronics, Renesas Technology, and Toshiba. This work is also supported by CREST project of JST. We are grateful for their support.

9. REFERENCES

[1] Agarwal, A., Paul, B.C., Mahmoodi, H., Datta, A., and Roy, K., 2005. A process-tolerant cache architecture for improved yield in nanoscale technologies. *IEEE T. VLSI Syst.*, 13, 1, 27-38.

[2] Albonesi, D., 1999. Selective cache ways: on-demand cache resource allocation. In Proc. International Symposium on Microarchitecture (MICRO).

[3] Amelifard, B., Fallah, F., Pedram, M., 2006. Reducing the sub-threshold and gate-tunneling leakage of SRAM cells using dual-V_t and dual-T_{ox} assignment. In Proc. of Design Automation and Test in Europe (DATE).

[4] CACTI, an integrated cache access time, cycle time, area, leakage, and dynamic power model. HP Labs. http://hpl.hpl.hp.com/personal/Norman_Jouppi/cacti4.html

[5] Fallah, F., Pedram, M., 2002. Circuit and system level power management. In Power Aware Design Methodologies, M. Pedram and J. Rabaey Eds. Kluwer Academic Pub., 373-412.

[6] Flautner, K., et al., 2002. Drowsy caches: simple techniques for reducing leakage power. In Proc. Int'l Symposium on Computer Architecture (ISCA), 148-150.

[7] Hase, M., Akie, K., Nobori, M., and Matsumoto, K., 2007. The development of low-power and real-time VC-1/H.264/MPEG-4 video processing hardware. In Designers' Forum, Asia South-Pacific Design Automation Conference (ASP-DAC), Yokohama, Japan.

[8] Hemmady, V.G., Reddy, S.M., 1989. On the repair of redundant RAMS. In Proc. Design Automation Conference (DAC), 710-712.

[9] International Technology Roadmap for Semiconductors—Design, 2006 Update. http://www.itrs.net

[10] Ishihara, T., Fallah, F., 2005. A cache-defect-aware code placement algorithm for improving the performance of processors. In Proc. Int'l Conference on Computer-Aided Design (ICCAD), 995-1001.

[11] Kanda, K., Duc Minh, N., Kawaguchi, H., and Sakurai, T., 2001. Abnormal leakage suppression (ALS) scheme for low standby current SRAMs. In Proc. IEEE International Solid-State Circuits Conference (ISCAS), 174-176.

[12] Kaxiras, S., Hu, Z., Martonosi, M., 2001. Cache decay: exploiting generational behavior to reduce cache leakage power. In Int'l Symp. on Computer Architecture, 240-251.

[13] Kim, I., Zorian, Y., Komoriya, G., Pham, H., Higgins, F.P., Lewandowski, J.L., 1998. Built in self repair for embedded high density SRAM. In Proc. Int'l Test Conference (ITC).

[14] Kuroda, T., Fujita, T., Hatori, F., and Sakurai, T., 2000. Variable threshold-voltage CMOS technology. IEICE T. Fund. Electr., E83-C, 1705-1715.

[15] Meng, K., Joseph, R., 2006. Process variation aware cache leakage management. In Proc. Int'l Symposium on Low Power Electronics and Design (ISLPED), 262-267.

[16] Narendra, S., et. al, 2002. 1.1 V 1 GHz communications router with on-chip body bias in 150nm CMOS. In Proc. IEEE ISSCC Dig. Tech. Papers, 270-271.

[17] Ozdemir, S., Sinha, D., Memik, G., Adams, J., Zhou, H., 2006. Yield-aware cache architectures. In Proc. Int'l Symp. on Microarchitecture (MICRO), 15-25.

[18] Powell, M.D., et al., 2000. Gated-V_{dd}: a circuit technique to reduce leakage in cache memories. In Proc. Int'l Symposium Low Power Electronics and Design (ISLPED).

[19] Rodriguez, S., Jacob, B., 2006. Energy/power breakdown of pipelined nanometer caches (90nm/65nm/45nm/32nm). In Proc. Int'l Symposium on Low Power Electronics and Design (ISLPED), 25-30.

[20] Sedcole, P., Cheung, P.Y.K., 2006. Within-die delay variability in 90nm FPGAs and beyond. In Proc. IEEE Field-Programmable Technology (FPT).

[21] Shirvani, P.P., and McCluskey, E.J., 1999. PADded cache: a new fault-tolerance technique for cache memories. In Proc. IEEE VLSI Test Symposium, 440-445.

[22] Tsukamoto, Y., et al., 2005. Worst-case analysis to obtain stable read/write DC margin of high density 6T-SRAM-array with local V_{th} variability. In Proc. Int'l Conference on Computer-Aided Design (ICCAD).

[23] Weste, N.H.E., Harris, D., 2004. CMOS VLSI Design: A Circuits and Systems Perspective, Addison Wesley.