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 timing-yield remains intact for a negligible extra 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

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
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 256x256 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 \( V_{th}=270\text{mv} \). 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 \( V_{th} \). 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.

\begin{align*}
V_{th}=270\text{mv} & \quad \text{(HP process)} \\
V_{th}=320\text{mv} & \quad \text{(MP process)} \\
\text{Memory target delay} = 540\text{ps} & \\
\text{Normalized leakage} = 1 & \\
\text{Normalized leakage} = 0.64
\end{align*}

Cells latencies (ps) due to within-die variation (\( V_{th}=270\text{mv} \))

<table>
<thead>
<tr>
<th>524</th>
<th>540</th>
<th>520</th>
<th>517</th>
</tr>
</thead>
<tbody>
<tr>
<td>539</td>
<td>535</td>
<td>532</td>
<td>537</td>
</tr>
</tbody>
</table>

Slow column is replaced by the redundant one

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].
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.

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

- $\mu_d$: the original mean delay of SRAM cells.
- $\sigma_d$: 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:

$$P_{cell} = \Pr\{x \leq D\} = \int_{-\infty}^{D} f(x)dx$$

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:

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

because $(1 - P_{cell})$ is the probability of a cell violating the delay, and $R \times (C + N)$ 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:

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

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.

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

(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:

- $\gamma$: the amount of delay variability (i.e. $\gamma = 3\sigma_d/\mu_d$)
- $P_L$: 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. $\mu_d$, 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: \text{OptimizeMemoryDesign()}

\textbf{Inputs:} ($\text{var}$, $\mu_d$: characteristics of the original delay distribution of SRAM cells),
\quad ($R$, $C$: original memory dimensions),
\quad ($Y_0$: Target timing-yield of memory),
\quad ($N$: number of spare columns)

\textbf{Output:} (chosen $V_{th}$, $T_{ox}$).

1. calculate $P_{cell}$ of original memory using Eq. 3 and given $R$, $C$, and $Y_0$.
2. $\sigma_d = \text{var} \times \mu_d / 3$
3. calculate $\mu_d$ using Eq. 1 and given original $\mu_d$ and above-calculated $P_{cell}$ and $\sigma_d$.
4. set $Y_0$, and then calculate new $P_{cell}$ of memory with redundancy using Eq. 4 and given $R$, $C$, $N$.
5. modify Eq. 1 by setting (new $\sigma_d$) = var $\times$ (new $\mu_d$) / 3
6. calculate new $\mu_d$ using just-modified Eq. 1, the above-calculated $D$, and new $P_{cell}$.
7. return $V_{th}$ and $T_{ox}$ corresponding to the new $\mu_d$ 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 $Y_0=V_{th}=X_{th}$ for the entire memory; thus (new $P_{cell}$) $\leq$ $P_{cell}$, 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 $\mu_d$) = $\mu_d$ 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 $\mu_d$) is computed; line 5 is needed to calculate new delay standard deviation ($\sigma_d$) since at higher $V_{th}$/$T_{ox}$, the $\sigma_d$ increases but delay variability $\text{var}$ remains constant.

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 $\mu_d$, 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 256x256 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>
<thead>
<tr>
<th>8x256 row decoder</th>
<th>Dynamic energy per access (pJ)</th>
<th>Leakage power (µW)</th>
<th>Delay (ps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>256x32 col. mux.</td>
<td>1.08</td>
<td>0.523</td>
<td>170</td>
</tr>
<tr>
<td>SRAM cells</td>
<td>11.07</td>
<td>7.10</td>
<td>642</td>
</tr>
</tbody>
</table>

Table 1. Power and delay of our 90nm 256x256 memory.

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 $\mu_d$ is
More redundant columns allow to further increase $V_{th}/V_{Tox}$ 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 $\mu_d$ due to the redundancies (corresponding to Figure 6).

<table>
<thead>
<tr>
<th># Spare columns (N)</th>
<th>$P_{cell}$</th>
<th>$\mu_d$ (mean delay) (ps)</th>
<th>Leakage reduction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (original)</td>
<td>0.99999962000000</td>
<td>552.82</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>0.999984762583658</td>
<td>570.61</td>
<td>38.39</td>
</tr>
<tr>
<td>2</td>
<td>0.999969680992248</td>
<td>573.05</td>
<td>41.90</td>
</tr>
<tr>
<td>4</td>
<td>0.999939865846154</td>
<td>575.59</td>
<td>45.61</td>
</tr>
<tr>
<td>8</td>
<td>0.999881590787879</td>
<td>578.22</td>
<td>48.58</td>
</tr>
<tr>
<td>16</td>
<td>0.999770182588235</td>
<td>580.93</td>
<td>52.18</td>
</tr>
<tr>
<td>32</td>
<td>0.999565934222222</td>
<td>583.67</td>
<td>54.79</td>
</tr>
</tbody>
</table>

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.

Efficiency. 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 256x256 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)

<table>
<thead>
<tr>
<th>#spare columns (N)</th>
<th>Leakage reduction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>59.36</td>
</tr>
<tr>
<td>5</td>
<td>70.18</td>
</tr>
<tr>
<td>10</td>
<td>73.70</td>
</tr>
<tr>
<td>15</td>
<td>75.34</td>
</tr>
<tr>
<td>20</td>
<td>76.42</td>
</tr>
<tr>
<td>25</td>
<td>77.12</td>
</tr>
</tbody>
</table>

Effect of target timing-yield. Table 4 shows leakage savings at various target timing-yield. More leakage can be saved when targeting higher timing-yield. 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-yield (3% variation, 642ps target SRAM cell delay, 5 spare column)

<table>
<thead>
<tr>
<th>Target timing-yield</th>
<th>Leakage reduction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>75</td>
<td>30.88</td>
</tr>
<tr>
<td>80</td>
<td>32.46</td>
</tr>
<tr>
<td>85</td>
<td>34.39</td>
</tr>
<tr>
<td>90</td>
<td>35.88</td>
</tr>
<tr>
<td>95</td>
<td>39.80</td>
</tr>
<tr>
<td>99</td>
<td>46.84</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Target delay (ps)</th>
<th>Leakage reduction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>650</td>
<td>44.61</td>
</tr>
<tr>
<td>642</td>
<td>46.84</td>
</tr>
<tr>
<td>630</td>
<td>48.12</td>
</tr>
<tr>
<td>620</td>
<td>51.31</td>
</tr>
<tr>
<td>610</td>
<td>55.23</td>
</tr>
<tr>
<td>600</td>
<td>59.37</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Memory size</th>
<th>Leakage reduction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2KB</td>
<td>47.85</td>
</tr>
<tr>
<td>8KB</td>
<td>46.84</td>
</tr>
<tr>
<td>32KB</td>
<td>45.60</td>
</tr>
<tr>
<td>128KB</td>
<td>44.75</td>
</tr>
</tbody>
</table>

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 256x256 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 \( \mu_D \) (step 3 of our algorithm), and \( \mu_D \) 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