An Energy-Efficient Last-Level Cache Architecture for Process Variation-Tolerant 3D Microprocessors

Joonho Kong, *Member, IEEE*, Farinaz Koushanfar, *Member, IEEE*, and Sung Woo Chung, *Senior Member, IEEE*

Abstract—As process technologies evolves, tackling process variation problems is becoming more challenging in 3D (i.e., die-stacked) microprocessors. Process variation adversely affects performance, power, and reliability of the 3D microprocessors, which in turn results in yield losses. In particular, last-level caches (LLCs: L2 or L3 caches) are known as the most vulnerable component to process variation in 3D microprocessors. In this paper, we propose a novel cache architecture that exploits narrow-width values for yield improvement of LLCs (in this paper, L2 caches) in 3D microprocessors. Our proposed architecture disables faulty cache subparts and turns on only the portions that store meaningful data in the cache arrays, which results in high energy-efficiency as well as high cache yield. In an energy-/performance-efficient manner, our proposed architecture significantly recovers not only SRAM cell failure-induced yield losses but also leakage-induced yield losses.

Index Terms—3D microprocessor, last-level cache, leakage energy optimization, narrow-width value, process variation, yield

1 Introduction

Employing advanced process technologies is a natural way to improve performance and power efficiency of microprocessors. However, process variation (PV) is a challenging problem in deep submicron process technologies since it adversely affects performance (mainly clock frequency) as well as power consumption, which in turn results in yield losses. Without any preventive technique, one inevitably faces severe yield losses.

Three-dimensional integration technology is one of the promising technologies (though thermal problems are challenging in 3D chips [1]). It vertically stacks several dies, enabling power and chip footprint (area) reduction, and performance improvement due to wire length reduction. However, 3D microprocessors[1] are not also free from process variations since die manufacturing process is same as 2D chip manufacturing process. Moreover, 3D chips are even more vulnerable than 2D chips since the typical 3D manufacturing process bonds different planar dies. It means that 3D microprocessors may suffer from both die-to-die (D2D) and within-die (WID) variation in one microprocessor. In other words, 3D microprocessors would have a wider range of parameter fluctuations compared to 2D microprocessors.

On the other hand, the components that are composed of SRAM cells are known to be most vulnerable to process variation. By looking into a component-level in 3D microprocessors, last-level caches (LLC: L2 or L3 caches) are known to be the most vulnerable component. As presented in [2], over half of the 3D microprocessors have their critical path in the L2 cache under process variations. Even worse, since LLCs are typically composed of several layers (dies) due to their huge capacity, LLCs are much more vulnerable to process variation compared to the other components that can be implemented within a layer (i.e., D2D+WID variation versus only WID variation).

Another important factor for vulnerability of LLCs to PV (process variation) is that the LLC consumes huge leakage power. Despite of employing high-V_{th} (threshold voltage) devices for SRAM cells residing in LLCs, PV may incur a large fluctuation in leakage power consumption across the SRAM cells. Since a huge number of SRAM cells exist in the LLCs, this leaves a possibility of excessive leakage power consumption in LLCs. As highly leaky chips cannot be used (i.e., the quality of these chips is far less than the quality standard), they also lead to yield losses. Not only for yield improvement, energy-efficiency itself is also crucial since energy is becoming a more scarce computing resource among the available resources. Considering the fact that LLCs are the most dominant leakage source in a microprocessor, leakage energy reduction techniques for LLCs are desirable.

In this paper, we propose an energy-efficient PV-aware 3D LLC architecture. By exploiting an architectural insight referred to as narrow-width values, our novel cache architecture saves many faulty cache lines under severe process variation, which results in significant yield improvement in

1. We refer to the microprocessor implemented by using the 3D die-stacking technology as a '3D microprocessor'.

- *J. Kong is with the School of Electronics Engineering, Kyungpook National University, Daegu 702-701, South Korea. E-mail: joonho.kong@gmail.com.*
- *F. Koushanfar is with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005. E-mail: farinaz@rice.edu.*
- *S.W. Chung is with the Department of Computer and Radio Communication Engineering, Korea University, Seoul 136-713, South Korea. E-mail: swchung@korea.ac.kr.*

Manuscript received 30 July 2013; revised 1 Feb. 2104; accepted 6 Nov. 2014. Date of publication 4 Dec. 2014; date of current version 12 Aug. 2015. Recommended for acceptance by H. Lee.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2014.2378291

a highly energy-efficient manner with only a small performance loss and area overhead. By referring to the data values actually stored in the LLC arrays, our architecture applies Gated-Vdd to the '0'-stored portions in the cache arrays as well as faulty cache portions as in [3]. A significant leakage energy saving is expected, which in turn contributes to the leakage-induced yield loss reduction.

Though many techniques have been proposed for leakage reduction of the cache memories [4], [5], [6], [7], [8], [9], [10], the main contribution of our technique is addressing both SRAM failure- and leakage-induced yield losses under process variation in 3D microprocessors. To the best of our knowledge, this is the first work which considers *both SRAM failures and leakage-induced yield losses in 3D microprocessors*. Moreover, most of the previous studies have been largely ignoring an yield impact of the leakage reduction.

For the sake of generality, our proposed architecture is mainly aiming at yield improvement of the SRAM-based LLC due to its prevalence in commodity microprocessors. In fact, general SRAM-based LLC architectures would gain benefits of wire length reduction by stacking the SRAM array dies, which in turn enables performance improvement and energy reduction [2], [11]. On the other hand, using 3D die-stacked architecture is also beneficial for integrating the dies manufactured from different process technologies (e.g., stacking DRAM arrays on the top of microprocessors [12], [13]). It enables an excessively large LLC which is suitable for high-performance systems. Note that our proposed architecture can also be adopted to DRAM-based LLCs with only a small modification from our SRAM-based architecture.

Our main contributions include:

- We propose a novel leakage-optimized PV-aware LLC architecture with a small area overhead (~10 percent), which enables yield improvement and cache energy saving with a small performance overhead;
- Our new architecture enables near-optimal leakage energy savings in the LLC by considering the data type information and turning on cache portions that store meaningful data;
- Our design-time technique for leakage-induced yield loss recovery further improves microprocessor LLC yield by up to 10 percent compared to our previously proposed architecture [3] and shows considerable elimination of the leakage-induced yield losses;
- We provide energy and performance evaluation results with various fault rates. The results in the case of the high fault rates enable a futuristic projection for our cache architecture (i.e., in the case of using more advanced process technologies).

The rest of this paper is organized as follows. In Section 2, we provide essential background for process variation in 3D microprocessors and narrow-width values. In Section 3, we explain our preliminaries. In Section 4, our novel PV-aware energy-efficient 3D LLC architecture is presented. In Section 5, we explain our evaluation methodology. In Section 6, we provide the evaluation results in terms of yield, energy, performance, and area. In Section 7, we briefly introduce related work relevant to our 3D LLC architecture. Lastly in Section 8, we conclude this paper.

2 Motivation and Background

2.1 Process Variation in 3D Microprocessors

For microprocessor design and manufacturing, process variation is a major challenge in deep-submicron process technologies. In the meantime, many studies have been focusing on mitigating process variation in 2D microprocessors [8], [10], [14], [15], [16], [17], [18]. On the other hand, in 3D microprocessors, process variation is much more severe than in 2D microprocessors. The main reason is that a 3D chip is composed of several different dies. It may cause severe D2D (i.e., inter-die) variation in a microprocessor, which can be another major source of parametric variations in 3D chips.

Among the major components in the microprocessor, caches are known to be most vulnerable to parametric variation. 6T SRAM cells are exposed to many kinds of failures such as delay, read, write, and leakage failures. A recent study already revealed that over 52 percent of the 3D microprocessors have their critical paths in L2 caches (LLC) [2], which imposes a vulnerability to the delay failure of SRAM cells in the LLCs. Moreover, 6T SRAM cells are not free from read and write failures, which may also contribute to the cache yield losses.

Another major problem of LLCs in 3D microprocessors is that LLCs are typically constructed by using several different dies; LLCs are composed of several layers in 3D microprocessors due to their large capacity. It means 3D LLCs face both D2D and WID variation within an LLC while 2D LLCs only encounter WID variation. For instance, assuming several different dies (layers) constitute LLC and only one layer among the LLC layers is severely affected by PV, the entire microprocessor may be accounted for as a yield loss. In addition, it implies 3D LLCs would exhibit a larger parameter fluctuation than 2D LLCs. Thus, employing preventive techniques is necessary for 3D LLCs to avoid severe yield losses.

Apart from the SRAM failure mechanisms explained above, LLCs are also vulnerable to leakage failures (leakage-induced yield losses) since the LLC is a huge source of leakage power in microprocessors, which may also lead to huge leakage fluctuation across chips due to process variation. The leakage failure in the LLC occurs when the leakage power consumption of the LLC is higher than the predefined leakage cutoff boundary. The chips that have a leakage failure are discarded and regarded as yield losses. For the microprocessors used in energy-constraint systems (e.g., battery-powered devices), leakage-induced yield losses are becoming more severe due to more strict standards for leakage failures during the testing procedure.

2.2 Narrow-Width Values

The narrow-width value contains a meaningful data portion in the LSB side while the remaining bit portion in the MSB side is filled with all '0's. Thus, only storing the meaningful portion of data values is enough and the remaining part of the data values can be filled by using the zero-extension logic. The main benefit by leveraging the narrow-width value is that it makes a storage utilization more efficient. In other words, with a same storage capacity, one can store more data by only storing the meaningful bit portion to the

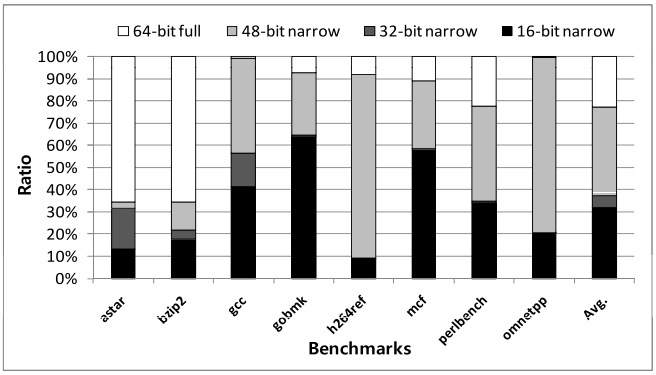
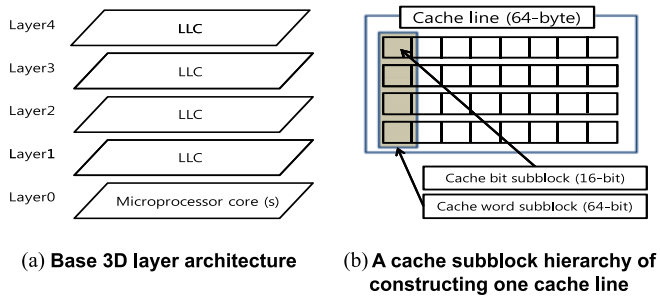


Fig. 1. The ratio of three types of narrow-width values and full-width values that are actually accessed in the L2 caches.



(a) **Base 3D layer architecture** (b) **A cache subblock hierarchy of constructing one cache line**

Fig. 2. Our base 3D layer architecture and cache subblock/line hierarchy.

storage. In fact, an efficient exploitation of the narrow-width values have been widely explored for soft-error protection or power/performance efficiency in register files or L1 caches [19], [20], [21], [22], [23]. However, the narrow-width value feature can also be used in process variation-aware 3D LLCs. The details will be described in Section 4.

In this paper, we classify data words into four types: 16-bit narrow-width, 32-bit narrow-width, 48-bit narrow-width, and 64-bit full-width. For example, 16-bit narrow-width value has its meaningful bit portion in 16-bits of the LSB side and the remaining 48-bits (in the MSB side) of data are all '0's.

We provide a simulation result regarding a ratio of the narrow-width values for a motivational purpose. Note that architectural evaluation framework is presented in detail in Section 5.3. To examine the ratio of the narrow-width values, whenever the LLC (in this case, L2 cache) is accessed, we check the type of the accessed data words and count their appearance over the execution of benchmarks. The most L2 cache sensitive eight benchmarks (*astar*, *bzip2*, *gcc*, *gobmk*, *h264ref*, *mcf*, *perlbench*, and *omnetpp*) [24] from SPEC INT CPU 2006 are used for our evaluation. As shown in Fig. 1, the ratio of the accessed narrow-width value is over 75 percent, on average. Particularly, in four benchmarks (*gcc*, *gobmk*, *h264ref*, and *omnetpp*) the ratio of the accessed narrow-width values is over 90 percent. It means that one can sufficiently gain the storage advantage by using the narrow-width value feature.

3 Preliminaries

3.1 Our Base 3D Integration Model and Cache Configuration

The base 3D integration model is shown in Fig. 2a. In this paper, we assume that four layers (from the layer 1 to 4) constitute the LLCs. In the layer 0 (the lowest layer), there is a microprocessor core (or cores). There are also through silicon vias (TSVs) for interconnection among the layers. It is a widely used 3D architecture that was already introduced in [12], [13]. Though we restrict base 3D integration model shown in Fig. 2a, our architecture can be applied to any other 3D integration models where the LLC is composed of several layers (dies). For example, in case that the processor cores and L2 cache arrays reside in the same layer (as shown in [2], [11], and [25]), our architecture can also be applied with little modification.

In this paper, we assume that the L2 cache (LLC in our base processor model) is eight-way set-associative with 64-byte line size and the total capacity is 1 MB (each layer has 256 KB data array size). Accordingly, the total number of cache sets is 2,048. Since commodity mobile and embedded processors (e.g., ARM Cortex-A [26] and Intel Atom [27] series) have the LLC capacity of 512 KB ~2 MB, our assumption on the cache capacity is reasonable. Note that our cache architecture is scalable to any LLC capacity as we will explain in Section 4. In the rest of this paper, we use the term 'L2 cache' instead of 'LLC' to avoid misunderstanding since the L2 cache is the last-level cache in our base processor model.

3.2 Layer Partition Schemes for 3D L2 Caches

Another important design decision is which layer-partition scheme is used for constructing the 3D L2 caches. There can be several possible ways to divide cache structures into each layer (i.e., which layer-partition scheme is used) in the 3D microprocessor. In this work, we introduce three layer-partition schemes: set-partition, way-partition, and bit-partition. The set-partition scheme divides cache sets into four layers. Thus, each of 512 sets is mapped to each layer. The way-partition scheme divides cache ways into four layers, mapping each of two ways to each layer. The bit-partition scheme divides each 64-bit word into four layers, mapping 16-bit in each word to each layer. The bit-partition scheme is similar to the 3D architecture introduced in [25], where the microprocessor has four layers and each layer has 16-bit data paths.

3.3 A Block Hierarchy for the Inside of a Cache Line

Fig. 2b describes a block hierarchy constructing one cache line. A 'cache line' is 64-byte size and it can be divided into eight 'cache word subblocks' (64-bit size). In each cache word subblock, one word is stored and it can be further divided into four 'cache bit subblocks' (16-bit size). In this paper, we use the terminology introduced in this section to avoid misunderstanding.

3.4 A Naive Way-Reduction Scheme

Due to negative impacts of process variations, there can be a huge amount of faulty SRAM cells in L2 caches. Without any preventive technique (i.e., baseline), only a single faulty SRAM cell in a chip may lead to a yield loss. In order to reduce yield losses, the simplest method is to discard (i.e.,

do not use) the cache line that has faulty SRAM cells, which will be referred to as a 'naive way-reduction scheme' in this paper. In this case, the available number of the cache lines in a cache set decreases.

If there is no available cache line in any cache set (i.e., in at least one cache set), this chip is regarded as a yield loss. One could also discard the faulty cache sets and access the main memory instead of the L2 caches when accessing the specific memory addresses which correspond to the faulty cache sets. However, in this case, one may suffer from severe performance loss due to frequent main memory accesses, which hurts performance of the microprocessor. For example, if an application frequently accesses the specific memory address that corresponds to the faulty cache sets, we may suffer from approximately 10 $\times$ performance loss in the worst-case. Typically, the main memory access latency is much higher than the L2 cache access latency (more than 10$\times$ longer). It can also be more deteriorated if there is a severe bus/interconnect contention. Moreover, to support such kind of the L2 cache bypassing schemes, we need additional logic which may be another burden. Thus, there should be at least one non-faulty cache line in each cache set, so that the chip works well without severe performance loss.

To reduce energy consumption, the unused (faulty) cache lines are permanently power gated. Power gating can be simply implemented by using Gated-Vdd [4] (PMOS power gating has negligible area overhead with leakage energy reduction of 86 percent).

The naive way-reduction scheme needs the fault bitmap (where fault bits reside) to record which cache lines are faulty. Hence, one needs a fault bit for each cache line. Note that '1' in the fault bit means that the corresponding cache line is faulty, and vice versa. Though the naive way-reduction scheme can reduce yield losses compared to the baseline, it hurts performance due to the reduced number of the available cache lines.

4 Our Energy-Efficient 3D L2 Cache Architecture for PV-Tolerance

First, we describe our PV-aware data storing mechanism [3] in Section 4.1. We then explain our leakage optimization technique which can be built on the top of our PV-aware 3D L2 cache architecture [3].

4.1 Storing Data to the L2 Caches for PV-Awareness

We explain how to store the data value to the cache considering PV-awareness in this section. Our PV-aware architecture operates in a finer-grain manner compared to the naive way-reduction scheme. As we explained in Section 3.4, in the naive way-reduction scheme, the discarding decision is made in a cache line (64-byte) granularity. On the other hand, in our architecture, the discarding decision is made in a cache bit subblock (16-bit) granularity to exploit the narrow-width value feature. Obviously, our architecture needs a larger fault bitmap compared to the naive way-reduction scheme. However, overall area and energy overhead is insignificant. As in the naive way-reduction scheme, we also adopt Gated-Vdd [4] to the faulty cache bit subblocks in our architecture for energy-efficiency.

TABLE 1
Possible Types of Data Words that Can Be Allocated According to the # of '0's in the Fault Bits (4-bit)

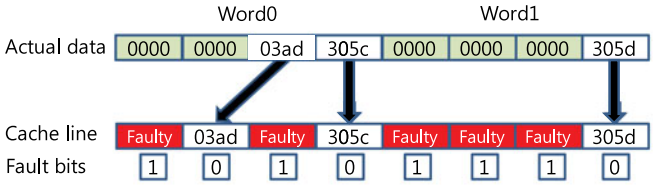
The # of '0's in fault bits of the cache word subblock	Possible types of the data word that can be allocated to the cache word subblock
0	N/A (the entire cache line is not used and turned off)
1	16-bit narrow-width value
2	16-bit, 32-bit narrow-width value
3	16-bit, 32-bit, 48-bit narrow-width value
4	All types of the data word

Among the layer-partition schemes introduced in Section 3.2, our PV-aware architecture adopts the bit-partition scheme to implement the multi-layered (multi-die) L2 cache. Four cache bit subblocks that constitute a cache word subblock are allocated to four separate layers. Due to D2D variations, different layers are likely to have quite different device characteristics. By using the bit-partition scheme, even though some layers suffer from severe process variation, the other layers can save the entire chip with a huge number of the available cache lines by exploiting the narrow-width value feature. In this case, most of the cache word subblocks can still store narrow-width values (16-, 32-, or 48-bit) though they cannot store 64-bit full width values under process variation.

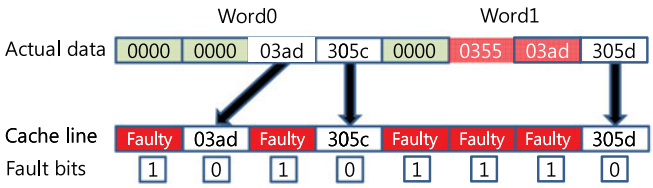
In our PV-aware architecture, when the data is allocated to the cache, the data words are classified into four different types of data (16-, 32-, 48-bit narrow-width, and 64-bit full-width value). After the data word classification is carried out, the fault bitmap is checked if the data words can be fit into the dedicated location of the cache. Our architecture counts the number of '0's in the fault bits (4-bit) of the corresponding cache word subblock to determine which types of the data word can be fit into that cache word subblock. Table 1 shows the possible types of data words that can be allocated according to the number of '0's in the fault bits (4-bit). We then determine whether each word can be fit into the dedicated cache word subblock in the cache line or not. If all of the cache word subblocks in a cache line can contain their responsible data word, the whole data can be allocated in the cache line.

Compared to the naive way-reduction scheme, our PV-aware architecture can save much more cache lines. In the case of the naive way-reduction scheme, only one faulty SRAM cell in the cache line leads to the failure of the entire cache line. However, in the case of our architecture, in spite of faulty SRAM cells, we can still use the cache line unless it is stuck at the case of the first row in Table 1 (i.e., the number of '0's in the fault bits of the cache word subblock == 0). It brings better performance due to the increased number of available cache lines as well as more yield loss recovery. In the case of our proposed architecture, if there is no available cache line in at least one cache set, this chip is regarded as a yield loss, as in the naive way-reduction scheme.

For better understanding of our data allocation mechanism, we provide a simple example of the case when the

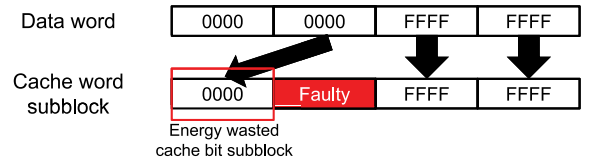


(a) **Case 1: the data can be fit into the dedicated cache line**

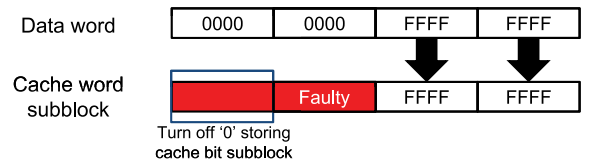


(b) **Case 2: the data cannot be fit into the dedicated cache line**

Fig. 3. Two possible cases when storing data to the cache line: (a) the data can be stored (b) the data cannot be stored.



(a) **When storing data by referring to the fault bit information**



(b) **When storing data by referring to the data type and fault bit information**

Fig. 4. A comparison of the cases between leakage energy is wasted (a) and further reduced (b).

data can be fit into the cache line and the opposite case in Fig. 3. To simplify the example, we assume that one cache line is 16-byte size (originally, 64-byte size in this paper) in this example. Thus, there are two cache word subblocks in one cache line. In Fig. 3a, the word0 is a 32-bit narrow-width value and the number of '0's in the fault bits is '2'. Thus, the word0 can be fit into the cache word subblock. Similarly, the word1 is a 16-bit narrow-width value and the number of '0's in the fault bits is '1', fitting well into the dedicated cache word subblock. Since both of two words fit well in two cache word subblocks, the entire data can be allocated to the cache line. On the other hand, in Fig. 3b, though word0 can be fit into the dedicated cache word subblock, word1 cannot be fit (it is a 48-bit narrow-width value while the number of '0's in the fault bits is '1'). Thus, the 16-byte data cannot be allocated to the dedicated cache line.

4.2 Leakage Energy Optimization

4.2.1 A Motivational Study

In [3], data is stored depending on whether the cache bit subblock is faulty or not. Though the effectiveness of this architecture for PV-tolerance and energy reduction is already shown in [3], there is also a sufficient room for further leakage energy optimization by paying a little more area.

Fig. 4 shows a motivational example of the cases when leakage energy is wasted (a) and optimized (b). Fig. 4a corresponds to the case of [3] in which the data storing relies only on the fault bit information. When one tries to store a 32-bit narrow-width value while there are three available cache bit subblocks, only two cache bit subblocks contain the meaningful data (meaningful data means the data which has non-zero bit). The leftmost cache bit subblock in Fig. 4a stores zero-bit and has to be turned on in the case of [3], even though it could be restored by the existing zero-extension logic.

In contrast, our architecture presented in this paper stores the data by referring to the data types of each word as well as fault bit information. As shown in Fig. 4b, one can store the 32-bit narrow-width value to the dedicated cache word subblock with two turned-off cache bit subblock. In this case, the zero-storing cache bit subblocks as well as faulty cache bit subblocks are turned off by using Gated-Vdd [4].

Fig. 5 illustrates the ratio of meaningful data storing cache bit subblocks when executing the selected benchmarks from SPEC INT CPU 2006. We present the ratio between the meaningful data storing cache bit subblocks and available (i.e., not faulty) cache bit subblocks under the fault rate = 30% (see Section 6.2). On average, only 65.48 percent of the cache bit subblocks stores the meaningful data over the benchmark execution. Under the fault rate = 30%, it implies approximately 35 percent (= 100 − 65) of the cache bit subblocks are storing zero-bits which can be restored by the zero-extension logic. If these energy-wasting cache bit subblocks was turned off, more leakage energy saving could be possible.

4.2.2 A New Data Structure Required for Leakage Optimization

The fault bitmap already introduced in [3] does not store the data type information while maintaining only the faulty cache bit subblock information. Since our architecture needs to know which type of data words is currently stored in the cache word subblocks, a new data structure should be introduced. This new data structure, which we call 'data type bit', stores the type information for each data word stored in the L2 cache. As we explained in Section 2.2, there are four different types of data words. Thus, we need 2-bits for each cache word subblock. Table 2 summarizes the data

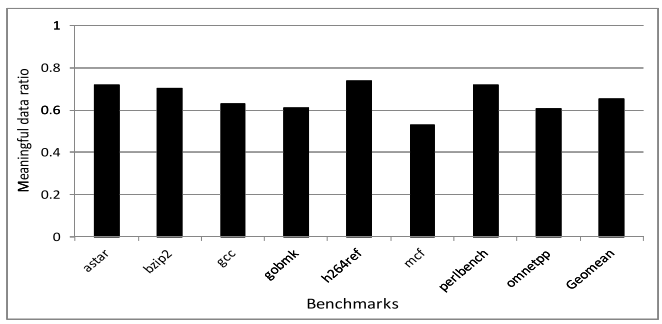


Fig. 5. A ratio between the meaningful data storing cache bit subblocks and available cache bit subblocks under the fault rate = 30%.

TABLE 2
The Data Type Bits According to the Type of Data Stored in the Cache Word Subblock

Data type bits	Actually stored data word
00	16-bit narrow-width value
01	32-bit narrow-width value
10	48-bit narrow-width value
11	64-bit full-width value

type bit values according to the actually stored data type in the cache word subblock.

4.2.3 Meaningful Data (MD)-Positioning Algorithm

To determine the position of the meaningful data in a systematic way, we introduce an algorithm that figures out the layer information of the meaningful data (either already stored or will be stored). This algorithm is referred to as 'MD-positioning algorithm' in this paper.

Algorithm 1 describes the MD-positioning algorithm. The required inputs for each cache word subblock are fault bit (4-bits) and data type bit information (2-bits). The output of this algorithm is a set which contains information of the layer numbers where the meaningful data will be stored or already stored. This set is referred to as 'set MD (meaningful data)' in this paper.

Algorithm 1. The MD-Positioning Algorithm.

```
Input: Integer N = the data type bit + 1;
       Bool B[4] = the fault bits of the cache word subblock;
Output: set MD
1   Integer i ← 1;
2   while (N != 0) // main loop
3       if (B[i-1] == 0)
4           MD ← i;
5           N - -;
6       endif
7       i++;
8   endloop
```

For better understanding of the algorithm, we provide an example that investigates a set MD in Fig. 6. In this example, the data type bit is '01' which also implies the initial N in Algorithm 1 is set to '2'. The fault bit is '1,000' that means layer 1 has a faulty cache bit subblock while the rest of the layers do not. By following Algorithm 1, one can easily find the set MD = {2, 3}, which means the layer positions of the meaningful data is layer 2 and 3. Since there are eight cache word subblocks in a cache line, one can perform eight operations for Algorithm 1 in parallel.

By using Algorithm 1, one can determine:

- which cache bit subblocks must be turned on and off by using Gated-Vdd;
- where to store the meaningful data to the cache arrays (layers) when storing new data to the cache;
- the control signals in zero-extension logic to restore the zero-bit portion in data words.

When storing the data word, the cache bit subblocks located in the 'set MD' layers are turned on to store the meaningful data while the rest of them within a cache word subblock are turned off to reduce leakage energy consumption. The zero-extension procedure is similar to the method presented in [3]. The only difference is how to generate the control signals. In this paper, the control signals are generated by referring to the output of the MD-positioning algorithm, while in [3] those are generated by referring to only the fault bit information.

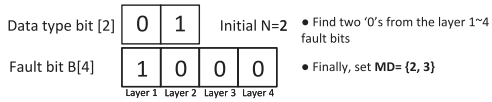


Fig. 6. An example of the MD-positioning algorithm.

4.3 Cache Access Algorithms

4.3.1 Cache Miss

In the case of cache misses (Fig. 7a), new data is fetched from the main memory and this data should be stored to the L2 cache. In this case, we should select a victim cache line that will be evicted from the L2 cache. The conventional way to select the victim cache line is to choose the least recently used (LRU) line in a cache set. However, in our architecture, we should additionally check if the data delivered from the main memory can be fit into the cache line that is selected by the LRU policy.

In the case that the data can be fit into the selected cache line, the data can be stored without any problem. However, there is a case where the data cannot be fit into the selected (LRU) cache line due to the faulty cache bit subblocks in the cache line. In order to deal with this case, we adopt a 'devised LRU policy'. First, in a cache set, our devised LRU policy searches the cache lines that the data value can be fit into. Among the selected cache lines (that the data can be fit into), the least recently used cache line is chosen as a victim cache line. There can be a case where the data is not fit into any cache line in the dedicated cache set. In this case, our architecture does not allocate the data in the L2 cache. Since this case rarely occurs, performance loss is negligible.

To store the new data to the cache line after the victim line is evicted, the data type bitmap is updated according to the data types of the new data words. After that, the MD-positioning algorithm determines where to store the meaningful data. Apart from the cache subblocks that will store the meaningful data, the rest of the cache bit subblocks that were already active (i.e., were turned on) are turned off by using Gated-Vdd.

Note that the latency for all of these operations during the cache misses do not make the processor cores be additionally stalled. This is because the additional operations required for our architecture can be performed in background even after the data is delivered to the processor cores. Thus, our architecture does not incur any additional latency in the case of the cache misses.

4.3.2 Cache Read Hit

In the case of read hits (Fig. 7b), we should check the fault bits and data type bits that correspond to the cache line which will be read from the cache. In our proposed architecture, when a cache line (64-byte) is accessed, 32-bit of the

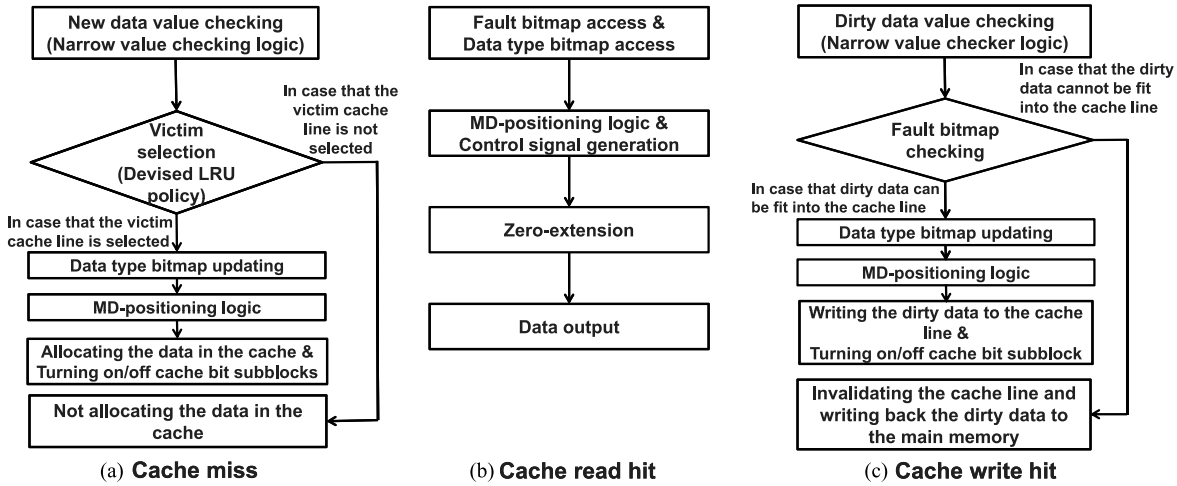


Fig. 7. Modified cache access algorithms for three different cases.

fault bits and 16-bit of the data type bits should be accessed. The MD-positioning algorithm determines which layers must be accessed to fetch the meaningful data from the cache arrays. In the zero-extension logic, by referring to the output from the MD-positioning logic, '0's and meaningful data are filled in the MSB side and the LSB side, respectively. The recovered data words are aligned in order, forming the complete data (a cache line size). Finally, the data is delivered to the microprocessor cores.

Though the zero-extension logic has negligible additional latency, we conservatively assume that accessing the zero-extension logic incurs 1-cycle delay in addition to the cache access latency, since it may be the critical path in a manufactured chip due to process variation. Note that accessing the fault bitmap, data type bitmap, and MD-positioning logic do not incur any additional latency. Since recent L2 cache designs employ a serial (sequential) access of tag and data arrays [28], one can sufficiently overlap the fault bitmap, data type bitmap, and MD-positioning logic access latency with the tag array access latency.

4.3.3 Cache Write Hit

In the case of write hits (Fig. 7c), the dirty (modified) data from the upper-level caches should be written to the L2 cache. In this case, we should check the fault bits of the dedicated cache line to confirm the modified (dirty) data can be fit into that cache line. If the data can be fit into the cache line, the data type bitmap is updated. After that, the cache bit subblocks are either turned on or off according to the output from the MD-positioning algorithm. Finally, the dirty data is written to the cache line. In the opposite case, the dirty data is written to the main memory and the cache line is invalidated to guarantee the operation correctness. In case of using the inclusive cache, the cache line invalidation in the L2 cache may also lead to the invalidation in the L1 cache. However, the cache line invalidation in the L2 cache rarely occurs, resulting in negligible performance losses. In Section 6.3, we give evaluation results for a performance impact of the write invalidations.

Due to the write buffers, the write access to the L2 cache does not cause any stall of the processor cores. Thus, there is no additional latency in the case of the write hit.

4.4 Overall Architecture

Fig. 8 depicts a high-level implementation of the cache controller to support our new cache architecture. Note that we depict only newly added logic components in the cache controller.

There are six newly added logic components to support our proposed architecture: narrow-width value checker logic, fault bitmap, data type bitmap, fit checker logic, MD-positioning logic, and zero-extension logic. The narrow-width value checker logic takes data words as an input and classifies each data word into four types (16-, 32-, 48-bit narrow-width, and 64-bit full-width). The fault bitmap contains information on whether each cache bit subblock is faulty or not. The values in the fault bitmap can be determined during the chip testing procedure and not changed in runtime. In our architecture, we need one fault bit per one cache bit subblock (16-bit). Since we use 1 MB L2 cache, 64 KB fault bitmap is required. The fault bitmap takes the address as an input to access the fault bits which correspond to the accessed cache line.

The data type bitmap is newly introduced data structure for leakage optimization. It stores the data type information of the data word which resides in each cache word subblock. Unlike the fault bitmap, the content in the data type bitmap is changed over the execution time according to the data word actually stored in the corresponding cache word subblock. 32 KB data type bitmap is required since 2-bits are needed for each cache word subblock. The fit checker

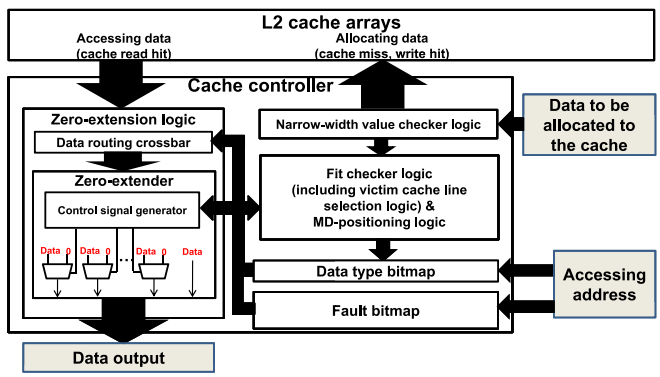


Fig. 8. An overall architecture of the cache controller data path.

logic is needed for the case when either a cache miss or a write hit occurs. The fit checker logic checks if new data or dirty data can be fit into the dedicated cache line. It also includes the victim selection logic for our devised LRU policy. The MD-positioning logic implements Algorithm 1, which figures out the layer information of either the already stored meaningful data (cache read hit case) or meaningful data that will be stored (cache miss or write hit case).

The zero-extension logic restores the narrow-width data to the original data. It includes a data routing crossbar, a control signal generator, and 2-to-1 MUXs. By referring to the output of the MD-positioning logic, the data routing crossbar aligns the meaningful data from the LSB side and the control signal generator provides the proper control signals to the 2-to-1 MUXs. The MUXs select either data values from L2 cache arrays or '0's. Note that we do not need MUXs which correspond to 16 bits in the rightmost side (bit [0-15] in a 64-bit word) of each word since there should be always meaningful data. Thus, we need only three MUXs for each word. The additional logic is simple and easy to implement. In our evaluation, we will also analyze the energy and area overhead of our additional logic in detail.

As we explained in Section 3.1, our cache architecture can easily be applied to any other 3D processor models (e.g., shown in [2], [11], and [25]). This is because all of the additional logic components can be implemented in the L2 cache controller, which results in design flexibility and compatibility.

4.5 A Design-Time Technique for Leakage-Induced Yield Loss Recovery

Since our architecture has a quite high potential for reducing leakage in the L2 caches, a design-time technique which reduces leakage-induced yield losses is also possible. Without the consideration on the impact of leakage savings on cache yield, the chip designer/manufacturer may have a too conservative leakage cutoff boundary (a border line for determining the leakage-induced yield loss). In this case, one may suffer from severe leakage-induced yield losses, even though the actual leakage power consumption in runtime will be much less than that in the chip testing procedure.

To take the impact of leakage savings on cache yield into account, one can set the leakage cutoff boundary more aggressively. In this case, one saves more chips which were supposed to be accounted as leakage-induced yield losses with the conservative cutoff boundary. Our design-time technique aggressively sets the leakage cutoff boundary considering the expected leakage savings from our energy-efficient cache architecture.

The leakage cutoff boundary can be represented as follows:

$$Cutoff_{leak} = Nominal\ leakage\ power \times \alpha, \tag{1}$$

where $Cutoff_{leak}$ is the leakage cutoff boundary set by the chip designer/manufacturer and α is a scaling factor that determines the quality standard of the manufactured chips. If the leakage power consumed from a chip exceeds the $Cutoff_{leak}$, the chip is regarded as a leakage-induced yield loss.

Since our cache architecture saves leakage energy, the chip designer/manufacturer can set the α value higher than the original α. It means more chips can be saved by our aggressive leakage cutoff boundary. If the average leakage energy saving of x percent is expected, then the new α value (α') can be set as follows:

$$\alpha' = \alpha \times \frac{100}{100 - x}. \tag{2}$$

Note that the appropriate α' can be set by considering how much leakage saving one can expect by running the real-world workloads or benchmarks.

Instead of the design-time approach, a dynamic approach may be beneficial. However, a dynamic (runtime) approach needs more hardware overhead for implementation. It requires additional hardware components for sampling (or profiling) and logic components for learning and decision algorithm, which will contribute to more energy consumption and area overhead. In addition, more energy and area overhead may also offset our yield benefit.

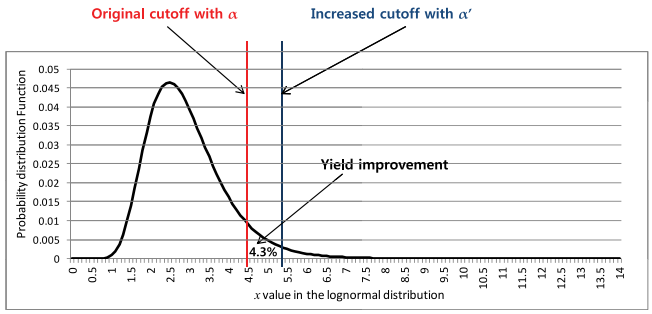


Fig. 9. A statistical yield improvement estimation.

Fig. 9 depicts an example of the calculating the estimated yield improvement. We assume that the leakage power distribution of the chips follows a lognormal distribution [29] and the standard deviation of the leakage power is the mean leakage power $\times$ 0.333333 in this example. We also assume the conventional $Cutoff_{leak}$ is the nominal leakage power consumption $\times$ 1.5 (i.e., $\alpha = 1.5$). As shown in Fig. 9, there might be a quite large number of unusable chips due to the excessive leakage power. However, by using Equation (2), the chip designers can set the leakage cutoff boundary higher than the original cutoff boundary by using α'. In Fig. 9, by setting the α' to $\alpha \times 1.11$ which corresponds to the expected leakage savings of 10 percent, one can further improve 4.3 percent of the cache yield (which also corresponds to the leakage-induced yield loss recovery of over 65 percent). In Section 6.1.2, we provide a comprehensive analysis on the setting of the $Cutoff_{leak}$ and its impact on the cache yield.

5 Evaluation Methodology

5.1 Process Variation Modeling and SRAM Failure Models

In this work, to precisely model process variation, we assign different effective gate length (L_{eff}) and threshold voltage (V_{th}) to each device of SRAM cells in the L2 cache. To model the within-die variations, we build variation maps by using the model described in [30].

In our base processor model, four different dies constitute the L2 cache. Each die has quite different characteristics

TABLE 3
A Classification of Schemes and Their Abbreviations

Categories	Set-partition	Way-partition	Bit-partition
Baseline	base-sp	base-wp	base-bp
Naïve way-reduction	nw-sp	nw-wp	nw-bp
Our proposed scheme w/o energy optimization	X	X	prop-bs
Our proposed scheme w/ energy optimization	X	X	prop-op

due to D2D variations, thus representing both D2D and WID variation in the L2 cache of each processor.

We perform a Monte Carlo simulation to evaluate yield across five variation severities with 240 chips for each variation severity level (denoted as 'PV level'). PV level 0 corresponds to the case of the lowest process variation severity while PV level 4 is the case of the highest. For more details on process variation modeling and parameters, please refer to the Appendix Section A.1.

Note that we use 45 nm technology node and the nominal device parameters for the MOSFETs are based on the device parameters used for ITRS-lstp (low standby power) SRAM cells [31]. This type of SRAM cells is typically used in L2 caches for leakage reduction. We also note here that our yield results do not incorporate the effect of TSV and 3D stacking-induced yield losses.

To model the SRAM failures, we also adopt four different SRAM failure models: delay [32], read [33], write [34], and leakage (BSIM leakage model [35]) failures. The parameters used in this paper are identical to the parameter presented in [3]. Due to the page limit, the details on the SRAM failure models can be found in the Appendix.

5.2 Energy Parameters

For energy evaluation, we derived leakage power and per-access dynamic energy of the L2 cache from CACTI 6.5 [31] and 3DCACTI [36]. For CACTI, we use ITRS-lstp cells for L2 cache arrays and assume that tag and data arrays are sequentially accessed [28]. We derived energy values for three different layer-partition schemes which have different physical layouts of the L2 caches. In the Appendix, we present our energy parameters in detail.

5.3 Architectural Simulation Framework

To obtain the performance results and cache access traces, we use M-Sim 3.0 simulator [37], which is derived from SimpleScalar toolset [38]. We model the processor architecture of our simulator similar to the commercial embedded microprocessors such as ARM Cortex-A9. From the architectural simulator, we collect cache access traces of the L2 cache required for calculating energy consumption and evaluate performance. In our evaluation, the access latency of the L2 cache is assumed to be 10 cycles. We also assume that there is one cycle additional delay for accessing the zero-extension logic in our architecture, as we explained in Section 4.3.2. The clock frequency of the simulated microprocessor is set to be 1.0 GHz. We use eight L2 cache sensitive benchmarks [24] from SPEC CPU 2006 INT benchmark suite (*astar*, *bzip2*, *gcc*, *gobmk*, *h264ref*, *mcf*, *perlbench*, and *omnetpp*). For precise evaluation, we fast-forward 20 billion instructions and actually run 500 million instructions.

TABLE 4
Parametric Yield Results with 1,200 Sample Chips

	base-wp	base-sp	base-bp	nw-wp	nw-sp	nw-bp	prop-*
level 0	240	240	240	240	240	240	240
level 1	240	240	240	240	240	240	240
level 2	15	19	11	240	240	240	240
level 3	0	0	0	232	225	222	239
level 4	0	0	0	2	5	7	111
Total	495	499	491	954	950	949	1,070
Yield	41.3%	41.6%	40.9%	79.5%	79.2%	79.1%	89.2%

6 Evaluation Results

In this section, we show the effectiveness of our proposed architecture. The baseline scheme does not have any countermeasures to mitigate process variation. The naive way-reduction scheme simply discards the faulty cache lines to improve yield, as we explained in Section 3.4. Each of three different layer-partition schemes is applied to the baseline and naive way-reduction schemes. To evaluate the effectiveness of our leakage optimization, we also provide the results of our architecture with and without leakage optimization. Thus, the total number of the schemes we evaluate here is eight. In this section, we use abbreviations for those schemes as shown in Table 3.

6.1 Yield

6.1.1 Parametric Yield Results

In this section, we present yield results of seven different schemes (*prop-** means both *prop-bs* and *prop-op*). Since we used the same leakage cutoff ($Cutoff_{leak}$) for both *prop-bs* and *prop-op*, the yield results of *prop-bs* and *prop-op* are identical. Note that we provide the impact on yield with different $Cutoff_{leak}$ in the next section.

Table 4 shows yield results of the seven different schemes. The numbers in Table 4 represent the number of passed chips or yield. Note that the criteria of determining passed/failed chips according to different schemes were explained in Sections 3.4 and 4.1.

Our proposed scheme shows the best yield (89.2 percent) across the seven schemes. The baseline schemes (*base-sp*, *base-wp*, and *base-bp*) show the lowest yield (40.9 $\sim$ 41.6 percent) since none of the preventive schemes are adopted. Even worse, when PV level is higher than 2, the baseline scheme shows 0 percent yield. Due to severe D2D+WID variations, the baseline yield in 3D microprocessors is significantly low. When adopting the naive way-reduction schemes (*nw-sp*, *nw-wp*, and *nw-bp*), yield is improved compared to the baseline schemes by 37.6 $\sim$ 38.3 percent. However, our proposed architecture (*prop-**) shows still higher yield than the naive way-reduction schemes by 9.7 $\sim$ 10.1 percent. Across the three layer partition schemes, one cannot see noticeable differences among the yield results.

The advantage of our proposed architecture is robustness to severe process variations. In PV level 3 and 4, a

TABLE 5
Expected Cache Yield Improvement When Using the *prop-op* Compared to that When Using *prop-bs* with Regard to α' Across Various σ

	$\sigma = \mu \times 0.25$	$\sigma = \mu \times 0.33$	$\sigma = \mu \times 0.50$
$\alpha' = 1.666$	1.81% (82.71%)	4.27% (65.38%)	6.58% (42.01%)
$\alpha' = 1.744$	2.04% (93.01%)	5.21% (79.82%)	8.74% (55.76%)
$\alpha' = 1.807$	2.13% (97.20%)	5.76% (88.26%)	10.37% (66.18%)

The values within the parenthesis are leakage-induced yield loss recovery ratios.

yield result when using our architecture is 72.9 percent (350/480) while that when using the *nw-wp* (it shows the highest yield among three naive way-reduction schemes) is only 48.8 percent (234/480). As process variation becomes more severe (particularly, due to severe D2D +WID variations), our architecture can save many chips that cannot be saved by the naive way-reduction scheme. Moreover, our architecture achieves high yield without any redundant cells. Along with the redundancy schemes [16], [18], we could achieve much higher yield by adopting our cache architecture.

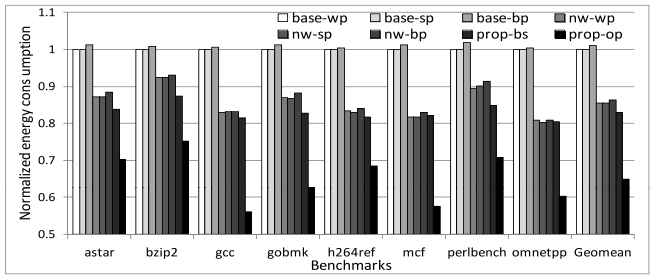
Our huge yield improvement comes from a synergistic effect between the bit-partitioned architecture and narrow-width value feature. The bit-partitioned architecture applied to 3D microprocessors spreads out each of four bit partitions in a data word into different layers. In this case, though some layers suffer from severe process variation, the other layers can save the microprocessor with many available cache lines by utilizing the narrow-width value feature. It also means our proposed architecture is specialized in 3D microprocessors (than 2D microprocessors) where both D2D and WID variations are expected in a microprocessor.

6.1.2 Additional Yield Improvement by Leakage Reduction

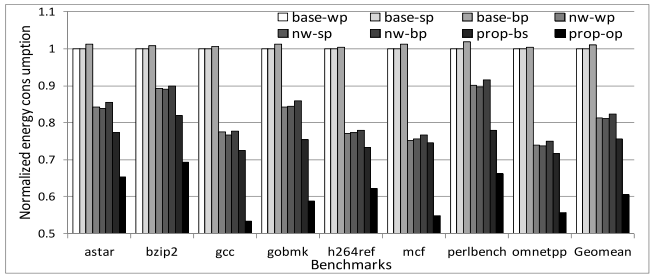
As we explained in Section 4.5, our leakage reduction further improves cache yield by setting higher α' values in Equation (2). We provide a statistical analysis for an expected cache yield improvement with various α' values.

Table 5 shows the expected yield improvement results. μ and σ represent the mean and standard deviation of the L2 cache leakage power consumption, respectively. According to the leakage saving results from our architectural simulation (only the cache leakage savings are considered), we set different α' values. The cases of $\alpha' = 1.666$, $\alpha' = 1.744$, and $\alpha' = 1.807$ correspond to the cases of the fault rate = 50%, the fault rate = 40%, and the fault rate = 30%, respectively. Note that one can expect a different amount of the leakage energy savings across different fault rates (see Section 6.2).

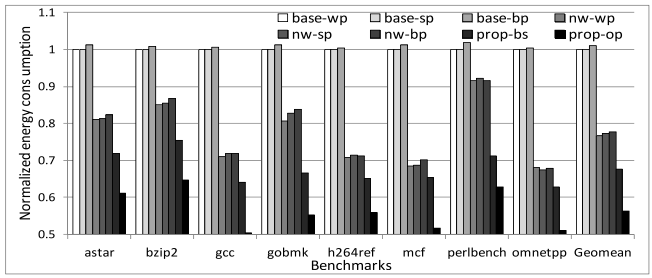
In the case of $\sigma = \mu \times 0.33$ with $\alpha' = 1.666$, one can expect yield improvement of 4.27 percent, which also corresponds to the leakage-induced yield loss recovery of 65.38 percent. When a chip designer or manufacturer increases α' to 1.807, further yield improvement (5.76 percent) is possible. In the cases of $\sigma = \mu \times 0.25$ and $\sigma = \mu \times 0.50$, the overall trends are similar to the case of $\sigma = \mu \times 0.33$. In the case of $\sigma = \mu \times 0.50$, one can expect the yield improvement by up to 10.37 percent.



(a) **Fault rate = 30%**



(b) **Fault rate = 40%**



(c) **Fault rate = 50%**

Fig. 10. Normalized cache energy consumption under different fault rates: (a) fault rate = 30%, (b) fault rate = 40%, and (c) fault rate = 50%.

In the design-time, the α' can be flexibly determined depending on the expected leakage savings. Since the amount of the leakage savings heavily depends on the workload characteristics, more aggressive optimization is also possible if the expected leakage savings from the workload behaviors are likely to be significant. For example, one can profile the leakage saving information of the frequently-used workloads by offline and accordingly set the optimized α' in the design-time.

6.2 Energy

For energy evaluation, we assume that three kinds of failures (delay, read, and write) are randomly distributed across the SRAM cells in the L2 cache. Though the device parameters in the L2 cache have a spatial correlation, the SRAM failures are incurred in a random fashion, rather than in a spatially-correlated fashion [10], [14].

Fig. 10 presents normalized energy consumption of the L2 cache based on three different cache line-level fault rates: 30, 40, and 50 percent. For instance, if the cache line-level fault rate is 30, 30 percent (on average) of the cache lines in the L2 cache are faulty. The fault rate of 30 percent approximately corresponds to the PV level 3 $\sim$ 4 (under severe process variations). All of the results are normalized to the energy results of the *base-wp* scheme.

Our evaluation with higher fault rates (40 and 50 percent) is a projection for more futuristic process technologies. The more advanced process technology one uses, the higher process variation one can expect, which increases a fault rate in the L2 cache.

In the case of the fault rate = 30%, our architecture without energy optimization (*prop-bs*) shows energy reduction of 16.9 and 2.2 percent (on average), compared to the *base-** (averaged across *base-wp*, *base-sp*, and *base-bp*) and the *nw-** (averaged across *nw-wp*, *nw-sp*, and *nw-bp*), respectively. Thanks to the Gated-Vdd, the naive way-reduction schemes and our proposed architecture save more cache energy compared to the baseline schemes. Since the naive way-reduction schemes adopt Gated-Vdd in a cache line-level while our architecture adopts the Gated-Vdd in a cache bit subblock-level, a larger portion of the L2 cache might be turned off in the case of naive way-reduction schemes, under the same fault rate. However, the naive way-reduction schemes suffer from more performance degradation (the detailed performance results are provided in the next section) due to the reduced number of available cache lines. This performance loss brings more leakage energy consumption due to longer execution time. Thus, the performance overhead reduction of our architecture also brings energy reduction.

Our proposed architecture with energy optimization (*prop-op*) achieves further energy reduction by 18.2 percent compared to the *prop-bs*, which is significant. Our energy optimization technique turns off cache bit subblocks by considering the actually stored data, which yields near-optimal leakage energy savings. Though a little more dynamic energy is consumed to support the *prop-op* due to the additional logic, it can be overwhelmed by huge leakage energy savings. Note that the leakage energy consumption accounts for more than 95 percent (on average) of the entire energy consumption in the L2 cache. In the cases of *mcf* and *gcc*, they tend to save more energy compared to the other benchmarks because a ratio of the narrow-width values is high in these benchmarks.

In the cases of the fault rates = 40% and 50% with the *prop-op*, one can save more energy compared to the *base-** by 39.5 and 43.6 percent respectively. Compared to the *prop-bs*, the *prop-op* saves 15.1 and 11.2 percent more energy in the case of the fault rates = 40% and 50%, respectively. As the fault rate increases, relative energy savings of the *prop-op* compared to the *prop-bs* decrease. This is because many cache bit subblocks are already faulty in the case of high fault rates, which lessens a room for further leakage energy reduction by the *prop-op*.

Though our proposed schemes reduce L2 cache energy consumption, the chip-wide energy consumption including core's energy consumption may be different from the L2 cache-only energy results since energy consumption is affected by execution time. We also show the energy results which incorporate energy consumption of the processor core. To figure out chip-wide energy consumption, we adopt an analytical model that calculates chip-wide energy consumption:

$$E_{proc\_X} = P_{core} \times T_{exec\_X} + P_{L2cache} \times T_{exec\_baseline} \times Norm_{energy}, \quad (3)$$

where E_{proc_X} corresponds to energy consumption of the processor when using the scheme 'X'. P_{core} and $P_{L2cache}$ are power consumed by processor core and L2 cache, respectively. T_{exec_X} represents the execution time with the scheme 'X'. $Norm_{energy}$ corresponds to our normalized energy results shown in Fig. 10. T_{exec_X} results come from our performance evaluation results (see Section 6.3). P_{core} and $P_{L2cache}$ are obtained from McPAT [39].

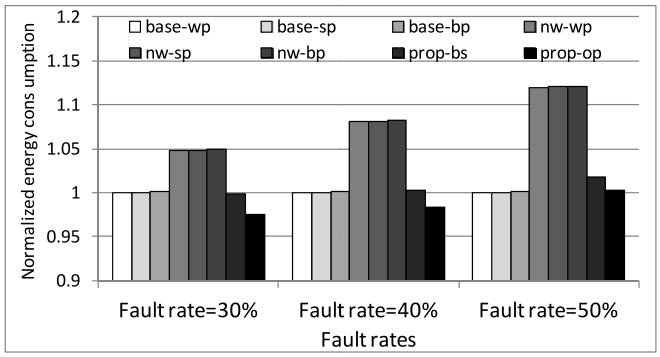


Fig. 11. Normalized energy consumption (average) of the processor under different fault rates (30, 40, and 50 percent).

Fig. 11 shows the chip-wide energy consumption when varying the fault rates. In overall, the energy savings from the L2 cache are offset by increased core energy consumption. This is because of the increased execution time in the case of *nw-** and *prop-**. In the case of the fault rate = 30%, the *prop-op* saves processor energy by 2.5 percent while the *nw-** consumes more energy compared to the *base-** (4.5 percent). In the case of the fault rate = 50%, the *prop-op* consumes a little more energy compared to the *base-** (0.3 percent). However, compared to the *nw-**, the *prop-op* is still far more energy-efficient (10.4 percent more energy saving than the *nw-wp*).

6.3 Performance

Fig. 12 shows the IPC (normalized to the *base-**) for three different cache line-level fault rates: 30, 40, and 50 percent. Since we assume that the cache access cycle is same across the layer-partition scheme, we show performance results across three different schemes: baseline (*base-**), naive way-reduction (*nw-**), and our proposed scheme (*prop-**). As shown in Fig. 12, the *prop-** shows a small performance loss (2.3 percent, on average) in the case of the fault rate = 30%. However, the *nw-** suffers from higher performance degradation compared to the *prop-** (7.1%, on average).

In the case of the fault rate = 40% and 50%, the average performance degradation in the case of the *nw-** becomes 10.8 and 14.6 percent, respectively. In contrast, the *prop-** shows much more graceful performance degradation compared to the cases of the *nw-**; performance losses of the *prop-** are only 3.8 and 6.4 percent in the case of the fault rate = 40% and 50%, respectively. In terms of performance, the *prop-** is superior to the *nw-** particularly under severe process variations.

In all benchmarks, the *prop-** reduces a performance loss more than the *nw-**. Because an off-chip main memory access accompanies a huge latency overhead, a last-level cache miss rate reduction of the *prop-** results in better performance. However, when executing *mcf* and *omnetpp*, a

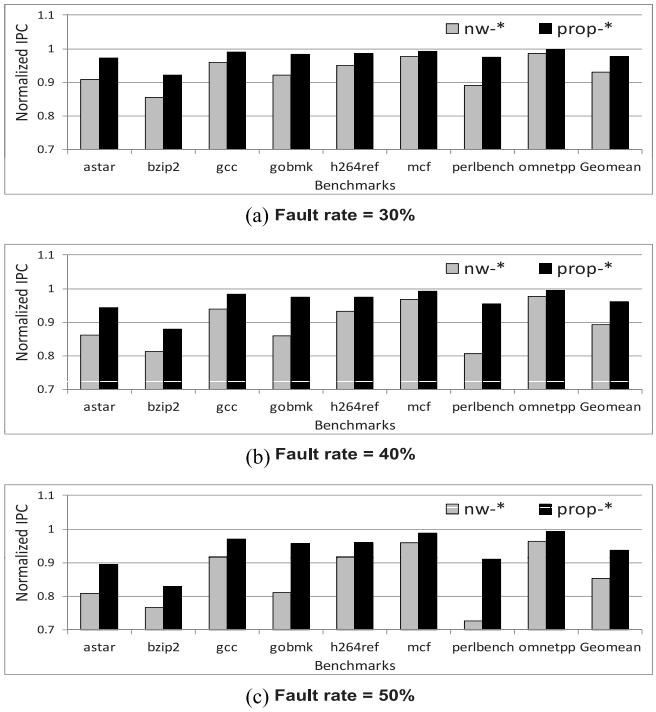


Fig. 12. Normalized IPC (instruction per clock cycles) results under different fault rates: (a) fault rate = 30%, (b) fault rate = 40%, and (c) fault rate = 50%.

performance gap between the *prop-** and *nw-** is less compared to executing the other benchmarks. The main reason is that these benchmarks are relatively insensitive to the number of the available cache lines in the L2 cache. On the other hand, the other benchmarks show a relatively large performance gap between the *prop-** and *nw-**. In the case of *perlbench*, a performance gap between the *prop-** and *nw-** is highest among the eight benchmarks (18.6 percent in the case of fault rate = 50%). Note that performance also significantly affects energy consumption since the leakage energy consumption is closely related to the execution time. Consequently, the *prop-** brings more energy reduction compared to the *nw-**, as shown in Section 6.2.

We also present L2 cache miss rates for each benchmark across various fault rates in Fig. 13. An overall trend of cache miss rates is consistent with the IPC results. In most of the benchmarks, the *prop-** shows a little increase of the cache miss rate compared to the baseline while the *nw-** shows huge miss rate increase. As the fault rate increases, one can see a steep increase of the L2 cache miss rate in the case of the *nw-**. In contrast, the *prop-** shows only a small miss rate increase, which is translated into better performance than the *nw-**. In addition, the *prop-** leads to less memory accesses and contention in the interconnection than the *nw-**, which may in turn result in better energy efficiency for the whole computer system.

Cache line invalidations in the case of cache write hit also contribute to performance degradation of the *prop-**. However, average cache line invalidation rates (= # of invalidations / # of cache accesses) are only 0.06, 0.09, and 0.11 percent in the case of the fault rate = 30%, 40%, and 50%, respectively. It means a negative impact of the cache line invalidations in our proposed architecture is negligible.

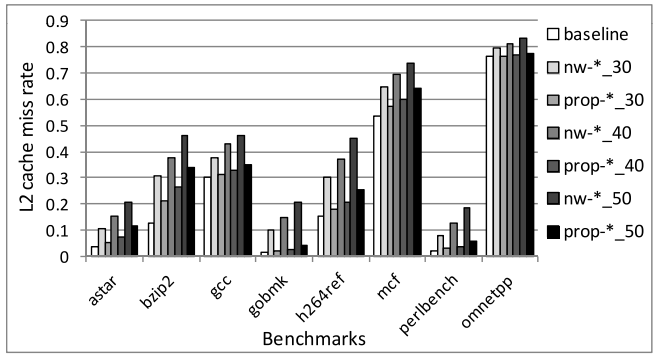


Fig. 13. L2 cache miss rates for each benchmark across various fault rates. The number which follows *nw-*_* or *prop-*_* corresponds to the fault rate (percent).

6.4 Area

Our proposed architecture has a small area overhead. For implementing the *prop-bs*, the fault bitmap occupies the largest area (6.04 percent of the entire L2 cache area, which is obtained from CACTI [31]) among our additional logic components. The zero-extension logic needs 2-to-1 MUXs, comparators, and some logic gates (e.g., XOR gates for a fault bit addition), which are negligible area overhead compared to the entire L2 cache area. The narrow-width value checker logic needs comparators and AND gates, which is also trivial. The fit checker logic can be implemented by small modification from the conventional victim selection logic. We estimate the additional logic overhead for the *prop-bs* as ~7 percent of the entire L2 cache area.

To additionally support the *prop-op*, we need more area not only to store the data type information but also to deploy the control logic components including the MD-positioning logic. The data type bitmap (32 KB) occupies ~3 percent of the entire L2 cache area. The other control logic components can be easily implemented with a small modification in the fit-checker logic, control signal generator, and data routing crossbar. According to our conservative estimation, the *prop-op* needs ~10 percent more area compared to the entire L2 cache area.

There can also be a trade-off between area and energy. If the microprocessor is supposed to be used in area-constrained environment, one can only adopt the *prop-bs* instead of the *prop-op* to reduce area overhead. Otherwise, for more energy- and yield-efficiency, one can fully adopt the *prop-op* by paying a little more area overhead. Another possible method for the trade-off is to adjust a mapping granularity of the fault bits, which has been already introduced in [10]. In the case that two cache bit subblocks are mapped to one fault bit, the area overhead is reduced from 10 to 7 percent in the case of the *prop-op*.

7 Related Work

Most studies for mitigating process variation are focused on 2D chips [10], [14], [15], [17] rather than 3D chips. There have been a few studies that are aimed at 3D microprocessors. In [13], Zhao et al. proposed a DRAM-based LLC architecture to mitigate process variation in 3D microprocessors. Their target is DRAM-based cache where data-intensive server application is preferred for their design. Ferri et al. [40] also proposed several 3D integration techniques to improve

performance as well as sales profit of chips. Ozdemir et al. [2] proposed a Cross LAyer Path Splitting (CLAPS) technique for variation-aware 3D microprocessor design. Their technique is focused on critical path splitting for yield improvement in 3D microprocessors. Though those techniques we introduce above are aimed at mitigation of process variation in 3D microprocessors, our proposed architecture can be differentiated with those techniques due to the efficient utilization of the narrow-width values for process variation-tolerant cache architecture.

The narrow-width value feature has been extensively explored for power/performance efficiency or soft-error protection. Brooks and Martonosi [19] utilized the narrow-width value feature for power reduction and performance improvement of the microprocessor data path. A register packing technique [20] of which main goal is performance improvement also utilizes narrow-width values. Hu et al. [22] also proposed an in-register duplication technique for soft-error tolerant register file design. There have been also several studies that apply the narrow-width value feature to cache memories. Ergin et al. [21] explored a soft-error resilient L1 data cache design. Islam and Stenstrom [23] proposed a separated narrow-width cache structure for energy reduction and performance improvement. However, the techniques introduced above are applied to register files or L1 data caches. In addition, the main goal of those techniques is not process variation (hard-error) resilience, but performance improvement, energy reduction, or soft-error resilience.

As SRAM-based cache memories consume huge leakage energy, many architectural-level cache leakage management techniques have also been proposed. Gated-Vdd [4] reduces cache leakage power by gating the power supply to the SRAM cells. A cache decay [5] technique manages cache leakage power in a cache line granularity considering program behaviors. Drowsy cache [6] maintains the supply voltage of the SRAM cells as a near-threshold level to not only manage cache leakage but also minimize a cache miss rate. Chung and Skadron proposed an on-demand wake-up policy for the L1 instruction cache, which optimizes both cache leakage and performance [7]. Meng and Joseph proposed a process variation-aware leakage reduction technique via cache way-level prioritization [8]. In [9], voltage-scaled cache designs for process variation-awareness are proposed. Though they achieved high power savings which have a potential for leakage-induced yield loss recovery, it has a limited applicability because our architecture considers various causes of the yield losses (both SRAM failures and leakage failures). Though another voltage-scaled cache design [10] also achieved a high yield loss recovery, their design also considers only delay and leakage failures. Our leakage-optimized 3D cache architecture in this paper manages cache leakage in much finer-grained granularity (*cache bit subblock-level*: 16-bit granularity). By considering both SRAM cell-level failures (delay, read, and write) and cache-level failures (leakage), our cache architecture will bring much higher yield compared to the previously proposed techniques.

8 Conclusion

In this paper, we presented a novel last-level cache architecture to mitigate process variation in 3D die-stacked microprocessors. Our proposed architecture disables faulty cache bit subblocks (16-bit) to improve cache yield. Our proposed architecture significantly improves yield by up to 48.3 percent in an energy-/performance-efficient manner with a small area overhead. Our architecture also enables leakage energy optimization with a little more area overhead, which results in additional yield improvement by up to 10.37 percent. In the future process technologies in which higher process variations are inevitable, our proposed cache architecture can be a good alternative for future 3D microprocessor design. As our future work, we will further optimize our architecture through: i) evaluating our proposed architecture with larger LLC capacity, ii) devising a customized/optimized turn-off policy considering the unique characteristics of each chip under process variations, iii) investigating a dynamic support referring to workload characteristics, and iv) evaluating a trade-off between the design-time technique and dynamic support.

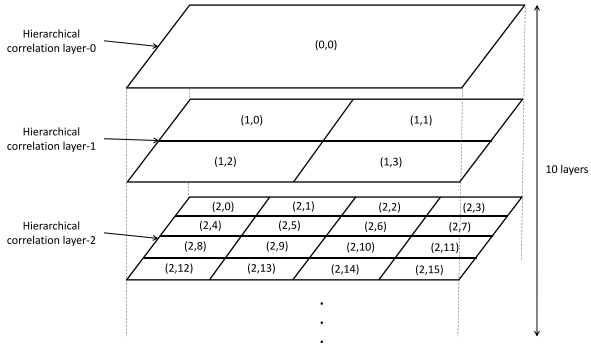


Fig. A1. Ten hierarchical correlation layers to model spatial correlations of process variation.

Appendix A
More Details on Our Evaluation Methodology

A.1 Process Variation Modeling

In this section, we describe our evaluation methodology on process variation modeling in detail.

Regarding the within-die variation, there are two types of variation, systematic and random variation. To model the systematic variation, we build variation maps by using the model described in [30]. In a die (there are total four dies in the L2 caches), cache data arrays are divided into 512×512 grids with 10 hierarchical correlation layers.[2] The 10 hierarchical correlation layers to build a process variation map are shown in Fig. A1.

Within one hierarchical correlation layer, the random variables that correspond to the grids follow Gaussian distributions with $N(0, {\sigma_{within-layer}}^2)$. To generate the spatially correlated variation map, the random variables (which are in the same horizontal location) across 10 hierarchical correlation layers are vertically added up. This added value is

2. Hierarchical correlation layers are virtual layers to model the within-die variations in a die. Assuming there are N hierarchical correlation layers (from 0 to N-1) to model the variation in one die, the number of grids in hierarchical correlation layer x is 4^x. For more details, please refer to and [3], [8], and [30].

TABLE 6
Five Levels of the Process Variation Severity

	Level 0	Level 1	Level 2	Level 3	Level 4
$\sigma_{within-layer}$	0.00	0.03	0.06	0.09	0.12
σ_{Vth}	0.02*Nominal V_{th}	0.04*Nominal V_{th}	0.06*Nominal V_{th}	0.08*Nominal V_{th}	0.1*Nominal V_{th}
σ_{Leff}	0.0111111*Nominal L_{eff}	0.0166666*Nominal L_{eff}	0.0222222*Nominal L_{eff}	0.0277777*Nominal L_{eff}	0.0333333*Nominal L_{eff}

denoted by M_{grid}. In our work, since there are 10 hierarchical correlation layers, 10 independent Gaussian random variables are added to generate a value for a grid. Assuming

$$X_{layer-n} \sim N(0, {\sigma_{within-layer}}^2), \tag{4}$$

where n is a hierarchical correlation layer number, M_{grid} can be calculated as follows:

$$M_{grid} = X_{layer-0} + X_{layer-1} + ...X_{layer-8} + X_{layer-9.} \tag{5}$$

Now, M_{grid} is assigned to each grid in the variation map. With this variation map, mean V_{th} (M_{grid_Vth}) and mean L_{eff} (M_{grid_Leff}) for each grid are calculated as follows:

$$M_{grid\_Vth} = Nominal\ Vth + \sigma_{Vth} \times M_{grid,} \tag{6}$$

$$M_{grid\_Leff} = Nominal\ Leff + \sigma_{Leff} \times M_{grid.} \tag{7}$$

Note that V_{th} and L_{eff} are generated from the same variation maps in our work, since as V_{th} increases, L_{eff} also tends to be increased, and vice versa [17].

Each grid is mapped to 1-byte (8-bit SRAM cells) in the cache arrays and each layer is formed by 512×512 grids. M_{grid_Vth} and M_{grid_Leff} are mapped to the appropriate SRAM cells in the L2 cache according to the layer-partition scheme. Thus, with the same variation map, the yield results are different according to the layer-partition schemes.

We also model random variation effects such as random dopant fluctuation (RDF). Within 8-bit cells that are mapped to the same grid, we assign V_{th} and L_{eff} to each device by generating Gaussian random variables with $N(M_{grid\_Vth}, {\sigma_{Vth}}^2)$ and $N(M_{grid\_Leff}, {\sigma_{Leff}}^2)$, respectively.

We assume that four different dies constitute the L2 cache. Each die generated from the above procedure has quite different characteristics due to D2D variations, thus representing both D2D and WID variation in the L2 cache of each processor.

We perform a Monte Carlo simulation to evaluate yield across five variation severities with 240 chips for each variation severity level (denoted as 'PV level'). Thus, total 1,200 chips are simulated (the total number of the variation maps: 1,200 $\times$ 4 layers = 4,800). By using different standard deviation values such as σ_{Vth} and σ_{Leff}, we adjust PV levels. PV level 0 is the case of the lowest process variation severity while PV level 4 is the case of the highest. Table 6 summarizes the parameters used for modeling different degrees of the PV severities.

A.2 SRAM Failure Models

For yield evaluation, we adopt four SRAM failure models: delay [32], read [33], write [34], and leakage (BSIM leakage model [35]) failures.

For delay model, we use Sarangi et al.'s SRAM delay model [32]. This model is a simplified version of [33], proposed by Mukhopadhyay et al. The delay cutoff boundary is set to be μ_{delay}(*the mean delay without process variation*) $+1.5 \times \sigma_{delay}$. In order to find the delay failing cells, we calculate the delay of each SRAM cell in the L2 cache and compare it with the delay cutoff boundary. If the calculated delay of the cell is higher than the delay cutoff, this cell is regarded as delay failing cells.

When the read failure occurs, the cell data is destroyed during the read operation (a destructive read). Since it adversely affects functionality of the microprocessor, consideration on the read failure is also important. For a read failure model, we use the long channel transistor read failure model proposed by Mukhopadhyay et al. [33]. We calculate V_{read} and V_{triprd} of each SRAM cell by using the model in [33] and find the cells of which V_{read} is higher than V_{triprd} (the condition in which the read failure occurs).

The write failure occurs when the write delay (T_{write}) is slower than the wordline activation delay (T_{write}). The write failures also negatively affect the functionality of the microprocessor since wrong values may be stored in the microprocessor caches. For a write failure model, we use a simplified write failure model proposed by Agarwal and Nassif [34]. If the calculated write delay of the SRAM cell is higher than the wordline activation delay, this cell is regarded as a write failing cell. Assuming the mean write delay is 1.0, we use 1.32 as a cutoff wordline activation delay in our work.

The above three SRAM failures occur at the cell-level. However, leakage failure occurs at the chip-level. In order to calculate chip-level leakage power consumption, we calculate the leakage power of each SRAM cell by using the BSIM leakage model [35]. The calculated leakage power values of each cell are added to calculate leakage power consumed by the L2 cache arrays. For our parametric yield simulation (shown in Section 6.1), if consumed leakage power of the chip is higher than $3 \times$ *leakage consumption of the chips without process variation*, this chip is regarded as a yield loss.

Many of the previous studies only modeled delay and/or leakage failures [2], [8], [10], [14], [16], [17], [18]. However, in this work, we additionally model the read and write failures that also have negative impacts on cache yield. Consequently, it leads to more accurate cache yield simulations. For full details on the SRAM failure models, please refer to the related papers and documents [32], [33], [34], [35].

TABLE 7
Energy Parameters for Architectural Simulations

Energy parameters for cache array			
	Set-partition & Way-partition	**Bit-partition (including our proposed scheme)**	
Dynamic energy per access (J)	0.133012e-9	0.184780e-9	
Vertical routing energy (J)	0.000875e-9	0.001110e-9	
Leakage power (W)	0.013785	0.013785	
Energy parameters for additional logics			
	Naïve wayreduction schemes	**Our proposed scheme w/o energy optimization**	**Our proposed scheme w/energy optimization**
Dynamic energy (J)	0.001349e-9	0.018703e-9	0.033437e-9
Leakage power (W)	0.0000984	0.0013810	0.0019816

A.3 Energy Parameters

Table 7 shows the derived per-access dynamic energy and leakage power for three different layer-partition schemes. We obtain the dynamic energy values from CACTI and properly scale them for 3D configurations to reflect reduced routing energy in 3D chips. Note that in the set-partition and way-partition schemes, per-access dynamic energy consumption is assumed as same since their main difference of dynamic energy consumption comes from vertical routing between different layers (i.e., which layer among four layers is accessed). We also model the vertical routing access energy including TSV accessing energy by properly scaling the routing energy obtained from CACTI, in order to reflect a reduced wire length effect in 3D chips. Regarding the leakage power, we assume that the same leakage power is consumed across three layer-partition schemes because data and tag arrays have same area and capacity regardless of which layer-partition scheme is used. However, actual leakage energy consumption of each chip will be different by selectively applying the Gated-Vdd [4] to unused cache bit subblocks (lines).

As shown in Table 7, we also derived the energy overhead of the additional logic components for naive way-reduction schemes and our proposed architecture. In our architecture, the energy overhead mainly comes from the fault bitmaps. Since the fault bitmap structure is similar to the cache data array structure, we also derived energy consumption of the fault bitmap from CACTI. Note that the other additional logic components (the narrow-width value checker logic, zero-extension logic, and fit checker logic) have negligible energy overhead.

We also reflect the energy overhead from our additional logic which is introduced for our leakage optimization. We derived the energy overhead of the data type bitmap from CACTI. For the rest of the logic, we added an additional energy overhead (only for dynamic energy) by 30 percent of the data type bitmap energy consumption. Note that it is very conservative assumption since the data type bitmap consumes a huge portion of the energy overhead for our additional logic. We ignored leakage energy consumption from the logic except the data type bitmap, since leakage energy consumption is negligible for the control logic.

ACKNOWLEDGMENTS

This work was supported by a National Research Foundation of Korea (NRF) grant funded by the Ministry of Science, Education, and Technology (NRF-2012R1A2A2A01013816) and Samsung Electronics. This work was also supported in part by the ARO STIR (W911NF-14-1-0456), National Science Foundation (NSF) (CCF-1116858), and ONR (N00014-11-1-0885). We would also like to thank the editor and anonymous reviewers for their helpful feedback. Sung Woo Chung is the corresponding author of this paper.

REFERENCES

[1] J. Kong, S. W. Chung, and K. Skadron, "Recent thermal management techniques for microprocessors," *ACM Comput. Surv.*, vol. 44, no. 3, pp. 13:1–13:42, 2012.
[2] S. Ozdemir, Y. Pan, A. Das, G. Memik, G. Loh, and A. Choudhary, "Quantifying and coping with parametric variations in 3D-stacked microarchitectures," in *Proc. 47th Des. Autom. Conf.*, 2010, pp. 144–149.
[3] J. Kong and S. W. Chung, "Exploiting narrow-width values for process variation-tolerant 3-D microprocessors," in *Proc. 49th Des. Autom. Conf.*, 2012, pp. 1197–1206.
[4] M. Powell, S.-H. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar, "Gated-Vdd: A circuit technique to reduce leakage in deep-submicron cache memories," in *Proc. Int. Symp. Low Power Electr. Des.*, 2000, pp. 90–95.
[5] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache decay: Exploiting generational behavior to reduce cache leakage power," in *Proc. 28th Int. Symp. Comput. Archit.*, 2001, pp. 240–251.
[6] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge, "Drowsy caches: Simple techniques for reducing leakage power," in *Proc. 29th Int. Symp. Comput. Archit.*, 2002, pp. 148–157.
[7] S. W. Chung and K. Skadron, "On-demand solution to minimize i-cache leakage energy with maintaining performance," *IEEE Trans. Comput.*, vol. 57, no. 1, pp. 7–24, Jan. 2008.
[8] K. Meng and R. Joseph, "Process variation aware cache leakage management," in *Proc. Int. Symp. Low Power Electr. Des.*, 2006, pp. 262–267.
[9] A. Sasan, H. Homayoun, A. M. Eltawil, and F. J. Kurdahi, "Process variation aware SRAM/cache for aggressive voltage-frequency scaling," in *Proc. Des., Autom. Test Eur.*, 2009, pp. 911–916.
[10] J. Kong, Y. Pan, S. Ozdemir, A. Mohan, G. Memik, and S. W. Chung, "Fine-grain voltage tuned cache architecture for yield management under process variations," *IEEE Trans. VLSI Syst.*, vol. 20, no. 8, pp. 1532–1536, Aug. 2012.
[11] K. Puttaswamy and G. H. Loh, "Implementing caches in a 3D technology for high performance processors," in *Proc. 23rd Int. Conf. Comput. Des.*, 2005, pp. 525–532.
[12] G. H. Loh, "Extending the effectiveness of 3D-stacked DRAM caches with an adaptive multi-queue policy," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchit.*, 2009, pp. 201–212.

[13] B. Zhao, Y. Du, Y. Zhang, and J. Yang, "Variation-tolerant non-uniform 3D cache management in die stacked multicore processor," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchit.*, 2009, pp. 222–231.
[14] Y. Pan, J. Kong, S. Ozdemir, G. Memik, and S. W. Chung, "Selective wordline voltage boosting for caches to manage yield under process variations," in *Proc. 46th Des. Autom. Conf.*, 2009, pp. 57–62.
[15] A. Agarwal, B. C. Paul, H. Mahmoodi, A. Datta, and K. Roy, "A process-tolerant cache architecture for improved yield in nanoscale technologies," *IEEE Trans. VLSI Syst.*, vol. 13, no. 1, pp. 27–38, Jan. 2005.
[16] A. Das, S. Ozdemir, G. Memik, J. Zambreno, and A. Choudhary, "Microarchitectures for managing chip revenues under process variations," *Comput. Archit. Lett.*, vol. 7, no. 1, pp. 5–8, 2008.
[17] E. Humenay, D. Tarjan, and K. Skadron, "Impact of parameter variations on multi-core chips," in *Proc. Workshop Arch. Support Gigascale Integr.*, 2006.
[18] B. F. Romanescu, M. E. Bauer, S. Ozev, and D. J. Sorin, "Reducing the impact of intra-core process variability with criticality-based resource allocation and prefetching," in *Proc. Conf. Comput. Frontiers*, 2008, pp. 129–138.
[19] D. Brooks and M. Martonosi, "Dynamically exploiting narrow width operands to improve processor power and performance," in *Proc. 5th Int. Symp. High Perform. Comput. Arch.*, 1999, pp. 13–22.
[20] O. Ergin, D. Balkan, K. Ghose, and D. V. Ponomarev, "Register packing: Exploiting narrow-width operands for reducing register file pressure," in *Proc. 37th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2004, pp. 304–315.
[21] O. Ergin, O. S. Unsal, X. Vera, and A. González, "Exploiting narrow values for soft error tolerance," *Comput. Arch. Lett.*, vol. 5, no. 2, p. 12, 2006.
[22] J. S. Hu, S. Wang, and S. G. Ziavras, "In-register duplication: Exploiting narrow-width value for improving register file reliability," in *Proc. Int. Conf. Dependable Syst. Netw.*, 2006, pp. 281–290.
[23] M. M. Islam and P. Stenstrom, "Characterization and exploitation of narrow-width loads: The narrow-width cache approach," in *Proc. Int. Conf. Compilers, Archit. Synth. Embedded Syst.*, 2010, pp. 227–236.
[24] X. Jiang, A. K. Mishra, L. Zhao, R. Iyer, Z. Fang, S. Srinivasan, S. Makineni, P. Brett, and C. R. Das, "ACCESS: Smart scheduling for asymmetric cache CMPs," in *Proc. 17th Int. Symp. High Perform. Comput. Archit.*, 2011, pp. 527–538.
[25] K. Puttaswamy and G. H. Loh, "Thermal herding: Microarchitecture techniques for controlling hotspots in high-performance 3D-integrated processors," in *Proc. IEEE 13th Int. Symp. High Perform. Comput. Archit.*, 2007, pp. 193–204.
[26] ARM Cortex-A series [Online]. Available: http://www.arm.com/products/processors/cortex-a/
[27] Intel atom processor specification [Online]. Available: http://www.intel.com/content/www/us/en/processors/atom/atom-processor.html
[28] H. Park, S. Yoo, and S. Lee, "A novel tag access scheme for low power L2 cache," in *Proc. Des., Autom. Test Eur.*, 2011, pp. 655–660.
[29] H. Chang and S. S. Sapatnekar, "Prediction of leakage power under process uncertainties," *ACM Trans. Des. Autom. Electr. Syst.*, vol. 12, no. 12, 2007.
[30] A. Agarwal, D. Blaauw, V. Zolotov, S. Sundareswaran, M. Zhao, K. Gala, and R. Panda, "Path-based statistical timing analysis considering inter- and intra-die correlations," in *Proc. 8th ACM/IEEE Int. Workshop Timing Issues Specification Synth. Dig. Syst.*, 2002, pp. 16–21.
[31] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "CACTI 6.0: A tool to model large caches"—HP Laboratories, 2009.
[32] S. R. Sarangi, B. Greskamp, and J. Torrellas, "A model for timing errors in processors with parameter variation," in *Proc. 8th Int. Symp. Qual. Electron. Des.*, 2007, pp. 647–654.
[33] S. Mukhopadhyay, H. Mahmoodi-Meimand, and K. Roy, "Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS," *IEEE Trans. CAD Integr. Circuits Syst.*, vol. 24, no. 12, pp. 1859–1880, Nov. 2005.
[34] K. Agarwal and S. R. Nassif, "Statistical analysis of SRAM cell stability," in *Proc. 43th Des. Autom. Conf.*, 2006, pp. 57–62.
[35] BSIM MOSFET Model [Online]. Available: http://www-device.eecs.berkeley.edu/~bsim3/bsim4.html
[36] Y.-F. Tsai, Y. Xie, N. Vijaykrishnan, and M. J. Irwin, "Three-dimensional cache design exploration using 3DCacti," in *Proc. Int. Conf. Comput. Des.*, 2005, pp. 519–524.
[37] J. J. Sharkey, D. Ponomarev, and K. Ghose, "M-Sim: A flexible, multithreaded architectural simulation environment," Dept. Comput. Sci., State Univ. New York at Binghamton, Tech. Rep. CS-TR-05-DP01, 2005.
[38] SimpleScalar toolset [Online]. Available: http://www.simplescalar.com
[39] S. Li, J.-H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and many core architectures," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchit.*, 2009, pp. 469–480.
[40] C. Ferri, S. Reda, and R. I. Bahar, "Parametric yield management for 3D ICs: Models and strategies for improvement," *ACM J. Emerging Technol. Comput. Syst.*, vol. 4, no. 4, pp. 19:1–19:22, 2008.

Joonho Kong received the BS degree in Computer Science from Korea University, Seoul, Korea, in 2007. He received the MS and PhD degrees in Computer Science and Engineering from Korea University, Seoul, Korea, in 2009 and 2011, respectively. He also worked as a postdoctoral research associate in the Department of Electrical and Computer Engineering, Rice University. He is now an assistant professor in the School of Electronics Engineering at Kyungpook National University. His research interests include computer architecture design, temperature-aware microprocessor design, reliable microprocessor cache design, and hardware security. He is a member of IEEE.

Farinaz Koushanfar received the PhD degree in electrical engineering and computer science and the M.A. degree in statistics, both from University of California Berkeley, in 2005, and the M.S. degree in electrical engineering from the University of California Los Angeles. She is currently an Associate Professor with the Department of Electrical and Computer Engineering, Rice University, Houston, TX, where she directs the Texas Instruments DSP Leadership University Program. Her research interests include adaptive and low power embedded systems design, hardware security, and design intellectual property protection. Prof. Koushanfar is a recipient of the Presidential Early Career Award for Scientists and Engineers (PECASE), the ACM SIGDA Outstanding New Faculty Award, the National Academy of Science Kavli Foundation fellowship, the Army Research Office (ARO) Young Investigator Program Award, the Office of Naval Research (ONR) Young Investigator Program Award, the Defense Advanced Project Research Agency (DARPA) Young Faculty Award, the National Science Foundation CAREER Award, MIT Technology Review TR-35, an Intel Open Collaborative Research fellowship, and a best paper award at Mobicom.

Sung Woo Chung received the BS, MS, and PhD degrees in Electrical Engineering and Computer Science from Seoul National University, Seoul, Korea, in 1996, 1998, and 2003, respectively. He is currently an associate professor in the Department of Computer and Radio Communication Engineering at Korea University. His research interests include low-power design, temperature-aware design, and user-aware design. He is a senior member of the IEEE and member of the IEEE Computer Society. He serves (and served) on the technical program committees in many conferences, including International Conference on Computer Design, International Symposium on Quality Electronic Design, and Asia and South Pacific Design Automation Conference. He is currently an Associate Editor of IEEE Transactions on Computers.