CODEWORD ASSIGNMENT FOR CELL-WRITE REDUCTION IN NON-VOLATILE MEMORY TECHNOLOGIES

by

Nathan Altay Hunter

B.S., University of Pittsburgh, 2012

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science

University of Pittsburgh

2014

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This thesis was presented

by

Nathan Altay Hunter

It was defended on

April 18, 2014

and approved by

Kartik Mohanram, PhD, Associate Professor

Department of Electrical and Computer Engineering

Steven P. Levitan, PhD, Associate Professor

Department of Electrical and Computer Engineering

Jun Yang, PhD, Associate Professor

Department of Electrical and Computer Engineering

Thesis Advisor: Kartik Mohanram, PhD, Associate Professor

CODEWORD ASSIGNMENT FOR CELL-WRITE REDUCTION IN NON-VOLATILE MEMORY TECHNOLOGIES

Nathan Altay Hunter, M.S.

University of Pittsburgh, 2014

This thesis explores a context-independent limited-weight codeword assignment architecture for cell-write reduction in emerging single-level cell (SLC) and multi-level cell (MLC) non-volatile memories (NVMs). Cell-write reduction in NVMs has many practical benefits, including lower write latency, lower dynamic energy, and enhanced endurance. The proposed architecture, which is integrated into the memory controller, relies on an a priori analysis of memory access patterns and a remapping table to minimize overwrites in NVM cells. The baseline for comparison used for comparing our algorithms is a technique known as data-comparison write (DCW) which performs a cell-wise comparison of the write data before each operation. This reduces cell-writes by allowing only the cells whose value changes to be rewritten. Similarly, Flip-N-Write (FNW), a technique which allows each word written to memory to be optionally inverted, serves as the state-of-the-art technique our schemes outperform. Our first algorithm relies on the different frequencies with which each value is written to memory to perform a frequency-based assignment (FBA) of codewords. Based on a set of training data, the most frequently occurring values are mapped to limited-weight codes (LWC) to reduce the number of bit-writes. Our second algorithm further improves upon FBA by considering the sequence in which values transition to one another to perform a sequence-based assignment (SBA) of codewords. All three methods are then modified to provide similar improvements in multi-level cell (MLC) memories. Since bit-write reduction is distinct from cell-write reduction in MLC, the LWCs are changed for FBA, the algorithm used by SBA to calculate node weights are modified, and a different approach to cell inver-

sion for FNW is considered. Trace-based simulations of the SPEC CPU2006 benchmarks show a 33× reduction in raw bit-writes in SLC and a 28× reduction in raw cell-writes in 2-bit MLC. These correspond to a 19% and 15% improvement over the best state-of-the-art method respectively.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 INTRODUCTION

The goal of this thesis is to explore the use of codeword substitution techniques in nonvolatile memories. Although many nonvolatile memory technologies are poised to replace complementary metal-oxide-semiconductors (CMOS), each has its own technical hurdles to overcome ranging from low endurance to high energy consumption. However, regardless of the technology these problems can be mitigated if the number of cell accesses can be algorithmically reduced. A reduction in cell writes directly reduces both cell wear and energy cost, and can also result in a reduction in latency. Performing analysis of expected memory access patterns in advance to create a codeword assignment table in the memory controller can allow these improvements to occur. This thesis presents codeword substitution techniques which improve the performance of nonvolatile memories by reducing the number of cell writes.

1.1 MOTIVATION

Over the last few years, computing power has increased substantially, placing greater demands on the technologies that drive these systems. One area with critical issues is main memory, especially in embedded systems where memory performance is critical. DRAM has played a major role in supporting the demands on memory capacity and performance for decades. However, scaling DRAM below 22 nm is currently unknown [2], placing limits on its maximum capacity and making it less suitable for next generation main memory. Several new non-volatile memory (NVM) technologies have been considered to address these problems, such as phase change memory (PCM) [13], resistive RAM (ReRAM) [3], and spin-transfer torque RAM (STT-RAM) [12].

However, the performance of these new technologies is affected by a number of problems. Some, such as PCM or ReRAM, suffer from poor cell endurance. In PCM, this is a result of the physical stresses on the phase change material during write operations. The repeated heating and cooling of each cell causes the phase change material to slightly expand and contract. Under enough cycles, these stresses can create gaps in the material or cause the material to break away from the heating element entirely. This ruins the electrical conductivity of the cell, causing it to be "stuck" in one state. PCM and ReRAM cells are only able to sustain on the order of 10^8 writes before failure, compared to the ability of DRAM cells to sustain on the order of 10^{15} writes [30]. Other technologies, such as STT-RAM, suffer from long write latency and high write energy. During write operations, STT-RAM requires a polarized current pulse to switch the spin direction of the free ferromagnetic layer in the magnetic tunneling junction. The length and magnitude of this pulse varies, with faster switching times requiring higher currents, resulting in write latencies 1.25–2× worse and write energies 5–10× worse than DRAM [12]. In all of these technologies, limits on the maximum amount of energy that the device can consume requires the memory to only write a limited amount of data at one time, further impacting the overall write latency of the device. Solutions to these problems must be designed before we can realize widescale adoption of any of these technologies in main memory systems.

Across these NVM technologies, existing solutions use various approaches to address the endurance, latency, and energy problems in NVMs. In PCM, for example, one set of solutions tackle the endurance problem through complex data migration or address translation techniques [18, 25, 19], while other proposals reduce the write latency directly through architectural improvements [11, 29]. With STT-RAM, a candidate replacement for SRAM cache or embedded DRAM [31], architectural improvements similar to those applied to PCM have been considered [12]. However, many of these proposals specifically address the endurance, latency, or energy problem, often with performance penalties in the other categories, limiting the potential improvement. A few proposals have shown improvements in all areas of energy, endurance, and latency. In contrast, Flip-N-Write (FNW) [6], is a solution that improves energy, endurance, and write latency of PCM by reducing the maximum number of bits changed during a write access through conditionally "flipping" the data to be written, reducing the number of bit-writes between the old and new data by up to half. However, FNW is a context-dependent coding scheme, which requires additional bits to indicate whether the stored data has been "flipped". Furthermore, the improvement in FNW is dependent on the both the existing data and the new data to reduce bit-writes.

1.2 WORK OVERVIEW

This thesis proposes an NVM architecture based on context-independent limited-weight codes to further reduce bit-writes and improve write energy, write latency, and write endurance in NVMs. The proposed architecture, which is integrated into the memory controller, relies on an LWC remapping table to encode the write data and reduce the number of bit-writes that occur during each write access to memory. The codewords in the LWC remapping table are limited-weight values (codewords with a limited number of ones) [7] selected from a larger code space in order to reduce the switching activity on NVM cells. When constructing this remapping table, i.e., codeword assignment, a priori, the frequency/sequence analysis on memory access patterns is performed for codeword generation, so that pairs of values that follow each other frequently are assigned codewords that are close to each other.

Thus, the number of bit-writes that are potentially necessary during each write access is reduced significantly. Since our proposed coding schemes are context-independent, they do not require additional bits for storing context information.

The same architecture is then extended to take advantage of the characteristics of multi-level cells (MLC). Storing multiple bits in each cell creates substantial asymmetry in the energy requirements of write data. Simple modifications to the codeword assignment algorithms allow the asymmetry to be taken into account, reducing the energy cost of write accesses in MLC NVMs.

When a write access is received by the memory controller, each half-word (16-bit) is passed through the remapping table to determine the correct codeword to which the half-word must be converted. Before committing the converted value to NVM cells, the memory controller uses a read-modify-write operation [26]. This operation compares the new value against the existing value in memory and only updates the changed bits. Combining the read-modify-write operation with the proposed LWC remapping, our proposed architecture can reduce bit-writes beyond FNW.

We evaluated our method across a variety of benchmarks from the SPEC CPU 2006 suite using an in-house simulator, with memory traces generated with the Intel Pin instrumentation toolset [15]. These trace-based simulations show a 33× reduction in raw bit-writes, which corresponds to a 2× (3×) improvement over state-of-the-art methods like FNW [6] (data-comparison write [26]).

1.3 PROBLEM STATEMENT

This thesis sets out to address the problem of unnecessary write operations in main memory, specifically in non-volatile memories where excessive write operations are problematic due to latency, dynamic energy, and endurance. This problem of excessive write operations corresponds directly bit-writes in SLC, since only overwritten cells require write operations to occur. In MLC it only corresponds to cell-writes, since each cell corresponds to multiple bits. There are four areas of interest to address:

- How can a simple codeword assignment scheme be designed to reduce bit-writes in SLC?
- How can an optimal codeword assignment scheme be designed to minimize bit-writes in SLC?
- How can a simple codeword assignment scheme be designed to reduce cell-writes in MLC?
- How can an optimal codeword assignment scheme be designed to minimize cell-writes in MLC?

1.4 WORK PLAN

The following work plan is used to address the areas of interest in the problem statement:

- Codewords are sorted by weight and assigned to the most frequently occurring values, resulting in a frequency-based codeword assignment.
- The interconnection matrix of the training data is analyzed to assign the lowest cost codeword transitions to the most commonly occurring sequences of values, resulting in a sequence-based codeword assignment.
- The codewords are sorted by their energy cost to the MLC cells before performing a frequency-based codeword assignment.
- The interconnection matrix is analyzed taking into account the direction as well as the sequence of transitions and the most commonly occurring transitions are used to assign the lowest energy cost codewords.

1.5 THESIS ORGANIZATION

In Chapter 2 we provide the background to our work by discussing the current state of non-volatile memories. We first describe the operation of PCM, ReRAM, and STT-RAM, and then provide an overview of the difficulties and trade-offs involved in replacing DRAM with each technology. We also consider coding techniques which have been proposed for memory devices and describe the architecture necessary to implement a context-independent coding technique in a memory device.

In Chapter 3 we describe two codeword assignment techniques based on context-independent codes. We begin by explaining how codewords can be generated and assigned, then describe the details of the two codeword assignment algorithms. We also discuss a hybrid approach which can be implemented for reduced assignment complexity.

In Chapter 4 we explore the application of the codeword assignment techniques to MLC. We provide motivation by explaining how codeword assignment in MLC is unique from SLC, then describe the changes to each technique.

In Chapter 5 we describe the framework used to perform our simulations, then present and discuss the results of our evaluations. Each coding technique is compared to the others using the criteria of bit-writes in SLC, cell-writes in MLC, and energy costs in MLC. We also perform a sensitivity analysis to demonstrate how adjustments to the size of the codewords and the complexity of the codeword assignment algorithm affect the results.

In Chapter 6 we draw conclusions from our work and discuss future work.

2.0 BACKGROUND

In this chapter we discuss the operation of new non-volatile memory (NVM) technologies. Because our method focuses on reducing the number of bit-writes, it can potentially be applied to any byte-addressable NVM technology to improve performance. We also provide a brief background on data encoding techniques and their use in reducing bit-writes in memory systems.

2.1 NON-VOLATILE MEMORIES

NVMs, such as phase change memory [13], resistive RAM [3], and spin-transfer torque RAM [12], use a variety of new technologies to store information. Unlike DRAM, which retains information as charge within a capacitor, NVM technologies typically store information using different states of a physical system. As a result, the information in these NVMs will remain intact for a far longer time than in DRAM, where the capacitors lose information (i.e., charge) in a matter of milliseconds.

2.1.1 PCM

Phase Change Memory technology uses the unique properties of chalcogenide glass to store information. This material typically consists of an alloy of germanium, antimony, and tellurium, such as $Gb_2Sb_2Te_5$, called GST [13]. When the material is in a crystalline (SET) state it exhibits low resistance when subjected to an electric current, and when it is in an amorphous (RESET) state it exhibits high resistance. These resistance levels are drastically different, with the resistance in the amorphous state being as much as 5 times that of the crystalline state.

A typical PCM cell consists of a piece of chalcogenide glass, a joule heater, and a transistor or diode to control access to the cell. To write information to the PCM cell, current is passed through a heating element to melt the chalcogenide material. When performing a RESET operation a short pulse of high current is passed through the heater, raising the temperature of the chalcogenide material above the melting point. This pulse is then quickly terminated to allow the melted material to rapidly cool, resulting in the chalcogenide material being programmed into the amorphous state. In contrast, when performing a SET operation a longer but weaker current pulse is applied to the heating element. This raises the temperature of the chalcogenide material above its crystallization temperature but below its melting point. This allows the chalcogenide material to slowly cool into the crystalline state.

A read operation is performed simply by measuring the current flow through the cell. The wide gap between the resistance levels of the amorphous and crystalline states allows us to quickly detect the large differences in current flow. This allows read operations to complete with a latency on the order of tens of nanoseconds.

2.1.2 ReRAM

ReRAM technology operates on similar principles as PCM, but instead uses the electrical switching properties of metal oxides to store information. ReRAM can be constructed using a variety of metal oxides, such as ZnO or TiO_2. Like PCM, the metal oxide material can be in two different states: a high resistance (RESET) state with the metal oxide material in its regular uniform structure, and a low resistance (SET) state caused by forming oxygen depleted conductive paths through the material.

To perform write operations in ReRAM cells, a voltage difference is applied across the cell to move oxygen ions in the metal oxide into or out of the cell. When performing a SET operation, a positive voltage difference is applied, forcing oxygen ions out of the metal oxide and into the electrode material. These oxygen depleted zones in the metal oxide form conductive metal-only filaments through the material, creating a low resistance state similar to the crystalline state in PCM. During RESET operations, a negative voltage difference is applied, forcing the oxygen ions out of the electrode material and back into the oxygen holes, placing the metal oxide material back into the high resistance state.

2.1.3 STT-RAM

Spin-Transfer Torque RAM technology uses the physical phenomenon of electron spin polarization to store information in a magnetic tunnel junction (MTJ). An MTJ is created by placing a thin layer of insulating material between two layers of ferromagnetic material. One of these magnetic layers has a fixed magnetic polarization, while the other has a polarization that is free to switch when subjected to a polarized current. The insulating later prevents the two layers from interfering with each other, but is thin enough to allow electrons to tunnel through so current can flow [12]. In an MTJ, the polarization of the free layer relative to the fixed layer affects the resistance of the cell. When the free layer is polarized parallel to the fixed layer, the cell is in a low resistance (RESET) state, and when the free layer is polarized anti-parallel to the fixed layer, the cell is in a high resistance (SET) state. Note that the resistance states are flipped with their respective write operations compared to PCM and ReRAM.

To perform write operations in STT RAM, a spin polarized current is injected across the MTJ device. By injecting this current from the free layer to the fixed layer, the spin polarization of the current forces the polarization of the free layer to the anti-parallel, or high resistance, state. Conversely, if this current is injected from the fixed layer to the free later, it forces the polarization of the free layer to the parallel, or low resistance, state.

2.1.4 Overview

While read operations in all of these new technologies are comparable to DRAM, write operations are the cause of their energy, endurance, and latency issues. First, both SET and RESET operations in all of these NVM technologies require a much higher current than DRAM to change the states of the device. Modern memory modules usually include energy consumption limits on their operation to prevent the modules from drawing too much power from the power supply, and to prevent overheating. As a result, the devices are restricted to writing a limited amount of data at once, iterating over the memory array to complete writes. This restriction increases the write latency, impacting performance [29]. Second, SET and RESET operations in PCM require long pulses to ensure that the material cools properly, up to hundreds of nanoseconds. As both SET and RESET operations occur simultaneously during a single write, the next write cannot start until all of the longer SET operations are complete, creating a bottleneck in individual write operations [30]. Finally, during the operation of both PCM and ReRAM, frequently changed cells are subject to physical stresses. In PCM, repeated heating and cooling puts thermal stress on the chalcogenide material, while in ReRAM, repeated programming can cause undesirable expansion in the conductive filaments. These stresses can result in poor data retention or can eventually lead to cell failure. This limits the lifetime of these memory cells to around 10^8 write cycles before cell failure.

2.2 MEMORY CODING TECHNIQUES

Various solutions have been proposed to improve the endurance, latency, and energy problems at the architecture level [18, 25, 19, 11, 29, 31, 12, 6]. Data encoding, combined with architectural design, has the potential to further improve the performance of NVM. Thus, data encoding studies specifically for NVM have become a significant area of research [23, 17, 16, 21]. However, most of the coding schemes in NVM are context-dependent codes that use a one-to-many mapping in which the codeword is chosen based upon both the current and previous data values to cross the interconnect. Thus, additional bits are necessary to store the context information for decoding. Two frequency-based remapping encoding algorithms for PCM were proposed to adjust the remapping table dynamically [23, 21]. The dynamic remapping encoding approach for ReRAM [17] converts only 6 states into a 3-bit multi-level cell (MLC), avoiding the high-power-consuming data patterns. An energy-aware coding scheme was proposed for PCM to convert the written data to low-power-consuming patterns based on the previously stored data [16]. Thus, they require additional bits to indicating the remapping type [23, 21], or data line transformation information [17], or prefix [16]. For frequency-based remapping encoding algorithms [23, 21], frequencies of values written into the memory may not profile the memory access pattern sufficiently, as we will discuss later in this thesis, since they do not represent the bit-level difference, i.e., Hamming distance, between two frequent values that are consecutively written to the same address.

Data encoding techniques have been extensively studied for minimizing the energy consumption of the data bus [4, 24, 28, 27, 22, 20]. The energy optimization in memory systems can be achieved by reducing switching activity, i.e., bit transitions on the interconnect, which is analogous to reducing bit-writes in NVM, showing its potential in addressing the energy, endurance, and latency issues in NVM. Context-dependent, double-ended codes have been proposed for bit-transition reduction [4, 24, 28, 27, 22, 20]. Since the context of a transmission is only known at the time of the transfer, such context-dependent codes must also be double-ended, meaning that the receiver must participate in the code to recover the original data value. Modern memory systems, however, cannot afford to use context-dependent double-ended codes over the processor-memory interconnect, which makes them less useful.

A set of context-independent single-ended coding schemes have been proposed to reduce bit-switch on memory data bus, which do not require additional context information bits [7]. In this thesis, we adapt the context-independent single-ended limited-weight coding (LWC) schemes for reducing the number of bit-writes in NVM. To the best of our knowledge, our design is the first work in studying context-independent coding schemes in NVM, which reduces the number of bit-writes by assigning codewords based on the frequency/sequence profiling without requiring additional bits.

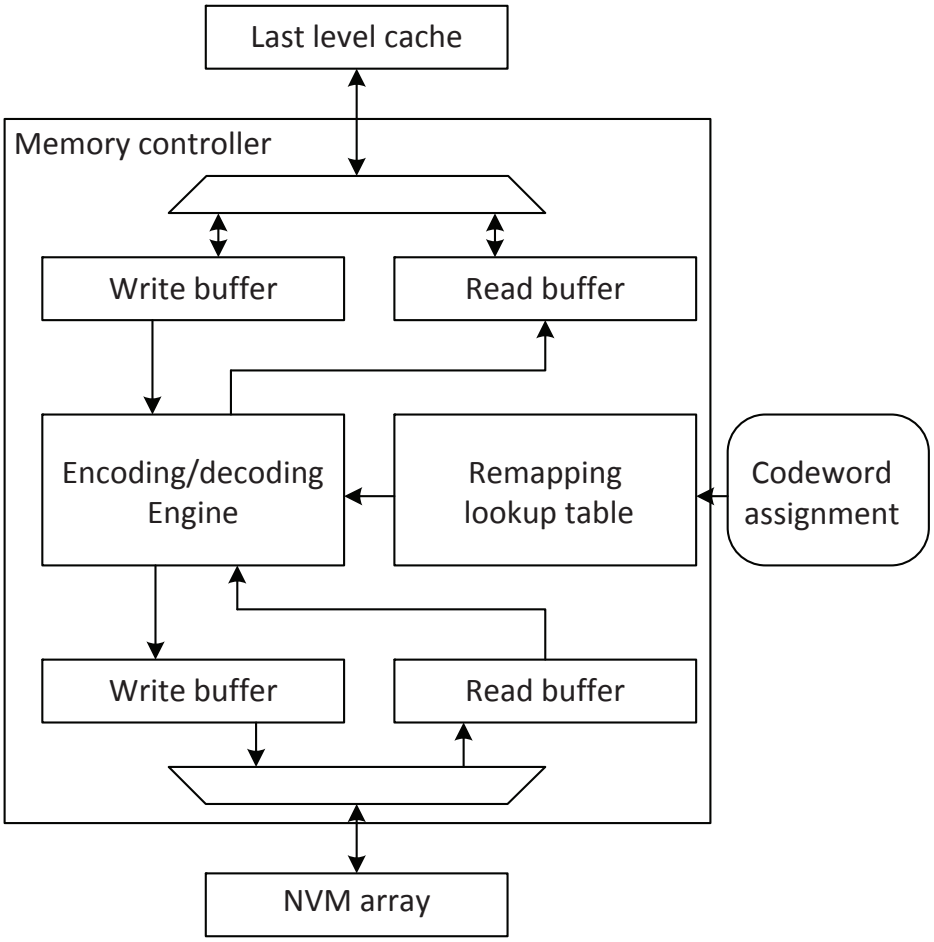

Figure 1: Context-independent coding architecture.

2.3 CONTEXT-INDEPENDENT CODING ARCHITECTURE

The proposed context-independent coding architecture is integrated into the memory controller as illustrated in Fig. 1. In this architecture, the encoding/decoding engine is integrated between the last level cache and the NVM array, allowing data to be seamlessly encoded and decoded during write operations. The encoding/decoding engine uses the remapping lookup table to perform the encoding/decoding operation. The context-independent codeword assignment is used to fill the lookup table. Extra circuitry is added to facilitate the new data path through the encoding/decoding engines. The encoding/decoding engine takes up only 10% additional area in a typical memory controller and requires 3 additional cycles to perform the encoding/decoding operation [14]. The reduction in bit-writes generated by the proposed architecture ensures that more words can be written in one write operation within the power budget limit [30], thereby compensating for the 3 extra cycles of delay incurred by the encoding/decoding engine. Any existing context-independent coding schemes can be used in this proposed NVM architecture for reducing the number of bit-writes.

3.0 LIMITED-WEIGHT CODES FOR SLC NVM

In this chapter we discuss reassigning values in non-volatile memories to limited-weight codewords in order to reduce the number bit-writes in a single-level cell memory. We discuss how the limited-weight codewords will be generated and detail the different methods of assignment.

3.1 MOTIVATION

The main idea of this thesis is to take advantage of context-independent, single-ended coding schemes to reduce the number of bit-writes that occur as each half-word in memory is overwritten by a new half-word. State-of-the-art methods, data-comparison write (DCW) [26] and Flip-N-Write (FNW) [6], have been successful in reducing the number of bit-writes during memory operation to improve performance. These methods show how the non-volatility of these new memory technologies can be leveraged by only performing write operations on the bits that need to be changed, while leaving the other bits unmodified. We show an example of LWC in Fig. 2. Without LWC coding, the write operation of the new data incurs 8 bit-writes, since all bits have to be change when writing with DCW. In the LWC assignment, we assign two codewords that have Hamming distance of 1 between each other to the old data and the new data. With LWC, we only need 1 bit-write, which is a significant improvement from architecture without LWC. The details of the LWC generation and assignment are discussed in in the remainder of this section.

	Without LWC	LWC encoded
Old data in the memory	0b11101110	0b00000001
New data	0b00010001	0b00000011
Write result	0b00010001	0bxxxxxx1x
Number of bit-writes	8 bits	1 bit

Figure 2: Illustration of a write operation without and with LWC.

3.2 CODEWORD GENERATION

Limited-weight codes (LWCs) provide an effective set of codewords that minimize the number of ones in each codeword [20]. LWCs are single-ended, context-independent codes. Consider a k-bit wide data bus with 2^k information symbols. A m-LWC is a one-to-one mapping where every k-bit in the 2^k input space maps to a codeword such that the Hamming weight (i.e., the number of ones in the codeword) is less than or equal to m. Since the source entropy must remain unchanged, i.e., since every information symbol must have a unique codeword, the following inequality must be satisfied by all m-LWCs:

$$\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{m} \geq 2^k \tag{3.1}$$

Here, n is the minimum number of bits ($m \leq n$) required to satisfy the inequality (3.1). Therefore, n determines the width of the bus needed to implement a m-LWC.

A perfect m-LWC satisfies (3.1) above with equality, i.e., all the codewords of length n with weight less than or equal to m are used in the mapping. For example, a 4-LWC where k equals 8 is a perfect 4-LWC when the codeword bus width, n, equals 9. The degenerate case where m equals k is a simple remapping. For example, an 8-LWC where k equals 8 includes the same codewords as the input space, but they can be potentially reassigned to reduce energy consumption in NVM.

3.3 TRACE-BASED ASSIGNMENT

As discussed in the introduction, context-dependent codes use context information to assign codewords in order to minimize bit-writes. However, single-ended codes require the assignment to be independent of the current context. This can be done with no a priori information about the memory behavior of the system, as in the originally proposed limited-weight code [20]. In contrast, this thesis proposes assigning codewords based on an analysis of memory traces taken from the system.

Two important sets of information can be collected from a memory trace: the frequency of occurrence of each value, i, that crosses the interconnect (f_i) and the frequency with which each value, i, that crosses the interconnect is followed by every other value, j (t_{ij}). This information can be represented by a graph. Each node in the graph represents a single data value, i. The weight of i is the frequency of occurrence of that data value, f_i. Each node, i, is also connected to every other node, j, including itself, with a directed edge. The weight of an edge from node i to node j is the the frequency with which the data value j follows the data value i over the interconnect, t_{ij}. Since the same number of bit-writes will occur regardless of the order in which two values cross the interconnect, the two directed edges that connect each pair of nodes can be collapsed into a single edge whose weight is the sum of the two directed edges, t'_{ij}:

$$t'_{ij} = t'_{ji} = t_{ij} + t_{ji} \quad \forall i \neq j \tag{3.2}$$

Note that the frequency of occurrence of each value is then related to the transition frequencies in the following way:

$$f_i = \sum\nolimits_j t_{ij} = t_{ii} + \frac{1}{2} \sum\nolimits_{j \neq i} t'_{ij} \tag{3.3}$$

Note that $\sum_{j \neq i} t'_{ij}$ is halved because t'_{ij} includes all the edges that both enter and leave node i, and each occurrence of i is accompanied by both an entrance and an exit from i.

3.3.1 Frequency-based Assignment

Codewords can be assigned based on the frequency of occurrence of each information symbol. Several previously proposed codes have performed such frequency-based assignment [4, 24, 28, 27, 22, 20, 23, 21]. However, they have used different code generation techniques. Since most of the values remain uncoded, frequent value coding also uses a decorrelator to reduce bit-writes using context information [7]. Single-ended versions (without the decorrelator) of such half-word-level frequent value coding schemes do not perform as well as simpler coding schemes.

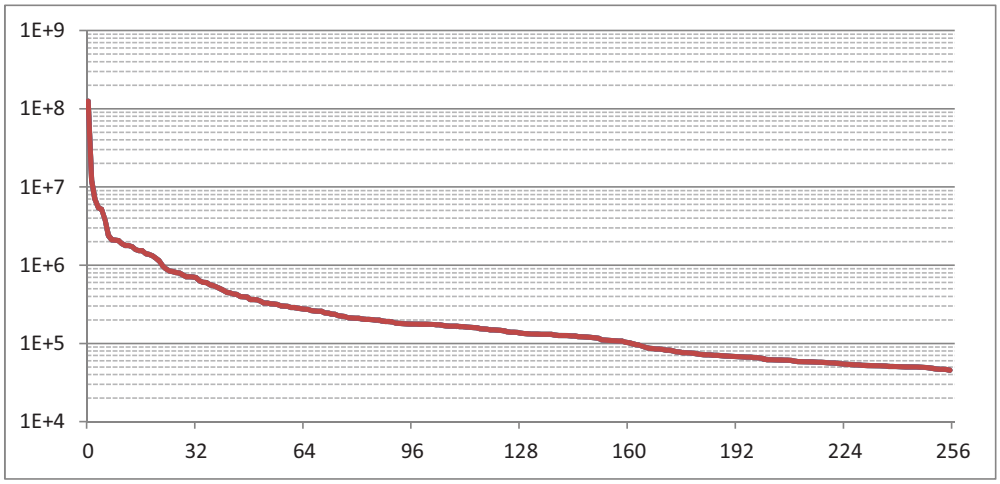

Figure 3: Number of occurrences of codewords.

Fig. 3 presents the frequency of occurrence of byte (8-bit) values in the basicmath benchmark from the MiBench suite [10]. This data helps illustrate the limitations and opportunities of frequency-based coding techniques. As shown in Fig. 3, the 13 most frequent values make up over 75% of the values written, and the 62 most frequent values account for over 90% of all values written. This is a significant observation which motivates the scheme we propose.

Note that although the code can be assigned either at the half-word level or the byte level, it is far more practical to code at the byte level, since there are only 256 possible values.

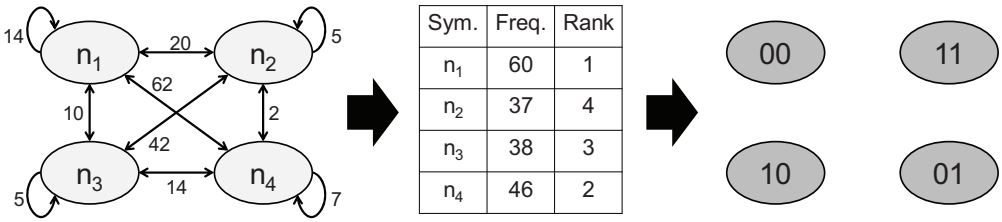

Figure 4: Example of frequency-based assignment.

A frequency-based, limited-weight code exploits this frequency distribution by ordering the information symbols by decreasing frequency of occurrence, f_i. Each information symbol is then assigned, in order, to the LWC codeword with the least weight that remains. The use of an k-LWC allows a simple remapping in which information symbols are reassigned based on their frequency of occurrence. The use of a $k/2$-LWC can be more effective, as the weight of the codewords is reduced.

Frequency-based remapping is illustrated with an example shown in Fig. 4. There are 4 nodes in the graph and the code-space consists of all 2-bit codewords, i.e., a 2-LWC where $n = 2$. The second column of the table in the figure shows the frequency of occurrence, f_i, of each value, and the third column ranks the nodes in descending order of f_i. Based upon this, the nodes are processed in the order n_1, n_4, n_3, and n_2 and the assigned codewords are illustrated in the resulting graph with shaded nodes. The expected number of bit-writes for this assignment of codewords, if the relative frequency of occurrence remains unchanged, is 184, in comparison to the expected number of bit-writes of 254 without frequency-based assignment.

3.3.2 Sequence-based Assignment

The set of codewords can also be assigned to information symbols based on the sequence in which they occur. Specifically, the frequency with which pairs of values follow each other on the interconnect can be used to ensure that pairs of values that follow each other frequently will be assigned codewords that are close to each other (i.e., have a small Hamming distance between them). Similarly, pairs of values that do not follow each other very frequently can be assigned codewords that are not close to each other.

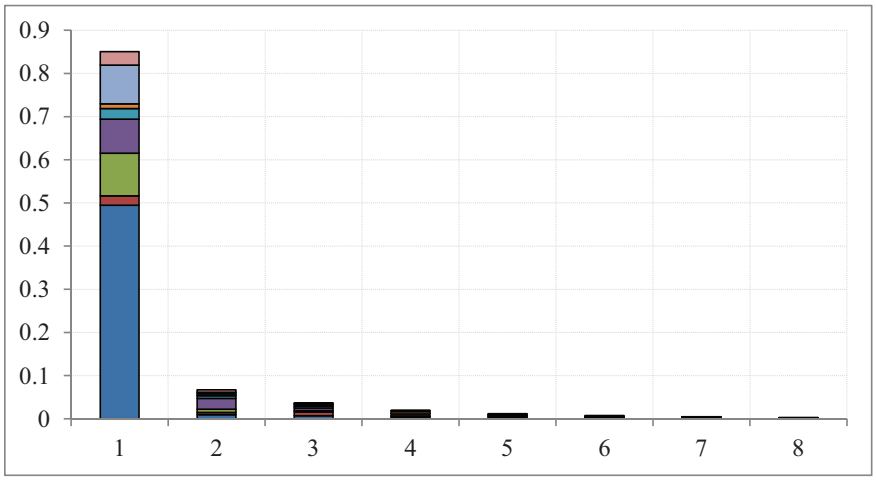

Figure 5: Frequency of transitions between value groups.

Figure 5 shows the frequency of transitions between bytes as a function of the frequency of occurrence of the bytes. The 256 8-bit information symbols are arranged in 8 groups of 32 symbols each in decreasing order of frequency of occurrence. For example, group 1 contains the 32 most frequently occurring bytes and group 8 contains the 32 least frequently occurring bytes. Along the y-axis, the vertically stacked bars represent the fraction of the transitions that occur between a symbol in group i and a symbol in group j, where $j \geq i$. From the figure, it is clear that over half of all transitions the transitions are completely contained within group 1. However, the number of transitions from group 1 to group 3 (third bar from bottom in group 1 bar) is substantially greater than the number of transitions contained within group 3 (bottom bar in group 3 bar). This is a very strong observation that motivates codeword assignments for bytes in groups 2 through 8 that reduce the Hamming distance to

the codewords assigned to bytes in group 1. This is, however, not addressed in frequency-based assignment, since the codewords are ranked in ascending order of their weights and assigned to information symbols ranked in descending order of frequency of occurrence. As a result, frequency-based assignment minimizes the Hamming distance between codewords assigned to bytes in a group, at the expense of the Hamming distance between codewords assigned to bytes across groups.

$\mathcal{C}$ – set of generated codewords, e.g., 8-LWC, 4-LWC for assignment
$\mathcal{U}$ – set of information symbols with no codeword assignment
$\mathcal{A}$ – set of information symbols with assigned codewords

$\mathcal{U} \leftarrow$ {all nodes in subject graph}
initialize $n \leftarrow \max_{i \in \mathcal{U}}(f_i)$, i.e., n is the most frequent node
code$[n] \leftarrow 0$; $\mathcal{C} \leftarrow \mathcal{C} \backslash \{0\}$; $\mathcal{U} \leftarrow \mathcal{U} \backslash \{n\}$; $\mathcal{A} \leftarrow \{n\}$

while $\mathcal{U} \neq \phi$ **do**
$n \leftarrow \max_{i \in \mathcal{U}} \left(\sum_{j \in \mathcal{A}} t'_{ij} \right)$
code$[n] \leftarrow \min_{c \in \mathcal{C}} \left(\sum_{j \in \mathcal{A}} (c \oplus \text{code}[j])\, t'_{nj} \right)$
$\mathcal{C} \leftarrow \mathcal{C} \backslash \{\text{code}[n]\}$; $\mathcal{U} \leftarrow \mathcal{U} \backslash \{n\}$; $\mathcal{A} \leftarrow \mathcal{A} \bigcup \{n\}$

Figure 6: Pseudo-code for sequence-based assignment.

From the graph of trace information, sequence-based assignment is formally equivalent to the minimization of the following objective function:

$$\sum_{i=1}^{256} \sum_{j=i+1}^{256} (\text{code}[i] \oplus \text{code}[j])\, t'_{ij} \tag{3.4}$$

where code$[i]$ and code$[j]$ are the codewords assigned to information symbols i and j, respectively, and t'_{ij} is the frequency of occurrence of the sequences $i \rightarrow j$ and $j \rightarrow i$, as defined in equation (3.2). This problem is intractable and belongs to the class of $\mathcal{NP}$-hard problems.

In practice, however, such problems respond fairly well to heuristics that proceed with assignments to information symbols one-at-a-time. In [7], the authors proposed a linear pass heuristic algorithm, shown in Fig. 6. The central idea is to proceed by making codeword assignments to nodes (information symbols) one-at-a-time in the subject graph. All nodes begin in the set of unassigned nodes, $\mathcal{U}$. Once a codeword assignment is determined for

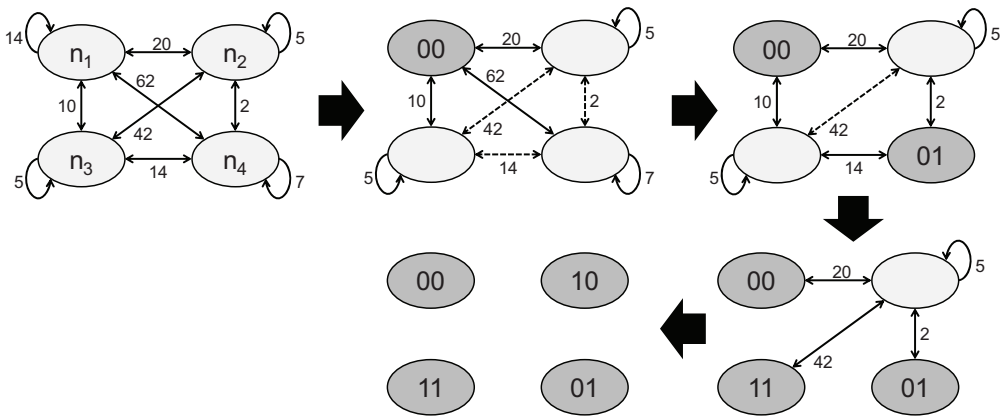

Figure 7: Example of sequence-based assignment.

a node, that node is transferred to the set of assigned nodes, $\mathcal{A}$. The first symbol to be assigned a codeword is the most frequently occurring byte. Although the first codeword can be randomly assigned, 0 is chosen for simplicity. On each subsequent pass, the node, n, with the maximum sum of transition frequencies to $\mathcal{A}$ given by $\sum_{j\in\mathcal{A}} t'_{nj}$ is chosen for codeword assignment. Ties are broken by selecting the n with higher frequency of occurrence, f_n. The codeword, c, is chosen such that it has the minimum cumulative weighted Hamming distance to the nodes in $\mathcal{A}$. The weighted Hamming distance of the codeword c to node $j \in \mathcal{A}$ is the Hamming distance between c and code$[j]$ weighted by the corresponding t'_{nj}. By summing over all $j \in \mathcal{A}$, the cumulative weighted Hamming distance of c to $\mathcal{A}$ is determined. The search for the codeword that minimizes the cumulative weighted Hamming distance is exhaustive over the pool of unassigned codewords. Once a codeword is assigned to n, it is moved from $\mathcal{U}$ to $\mathcal{A}$. The updates are performed as indicated in the pseudo-code, and a new node, n, is chosen for codeword assignment on the next pass.

The heuristic is illustrated with an example shown in Fig. 7, starting with the same graph as Fig. 4. There are 4 nodes in the graph and the code-space consists of all 2-bit codewords. Each graph in the figure represents one pass through the algorithm of Fig. 6. The shaded nodes belong to the set of assigned nodes, $\mathcal{A}$, and the unshaded nodes belong to the set of unassigned nodes, $\mathcal{U}$. At each step, the solid edges are edges that are considered

in selecting the next node, n, and the dashed edges are ignored at that step, but will be used in future steps. For clarity, edges that are no longer needed are removed from each graph. The expected bit-writes obtained for this assignment of codewords is 162. This is a 12% improvement over the frequency-based assignment for the same graph, and is attributable to the interchange of the assignments of 10 and 11 to nodes n_2 and n_3 in the subject graph. Sequence-based assignment thus results in the best possible assignment, since edges between the two pairs of codewords with Hamming distance 2 ({01,10} and {00,11}) have the two lowest weights (2 and 10) in the subject graph.

3.3.3 Rapid-SBA

In our proposed heuristic for SBA, the optimal codeword is assigned to the value with the lowest Hamming distance to all previously-assigned values in each step. This requires every unused codeword to be checked with every assigned code, making SBA substantially more computationally complex than FBA. SBA has a runtime of about 4 seconds at the byte level, but would take over 7 months (estimation) to run to completion at the half-word level. The drastic increase in computational complexity for the half-word level code assignment is due to the exponential increase in the number of values which must be considered, as well as the increase in the total number of codewords.

To reduce the computational complexity of SBA and provide a fast code assignment, we propose a greedy heuristic, called n-rapid-SBA. In the each step of the proposed n-rapid-SBA, instead of checking all unused codeword, we compare only the 2^n lowest Hamming-weight codewords. Also, when checking an unused codeword, we only calculate the sum of Hamming distance between this code and the first 2^n assigned codes, instead of calculating the sum of Hamming distance between this code and all assigned codes. When $n = 16$, the n-rapid-SBA is identical to SBA at the half-word level. For the byte level code assignment, it was possible to increase n all the way to 8. However, for the half-word level assignment, we believe that a feasible value of n is 12, which takes about 2 hours to complete.

4.0 CODEWORD ASSIGNMENT FOR MLC NVM

In this section we discuss the reasons our bit-write reductions will not provide improvements in multi-level cell memories and the proposed modifications to our codeword assignment algorithms in order to achieve the same level of improvement.

4.1 MOTIVATION

The algorithms discussed so far have made the assumption that a reduction in bit-writes will result in a reduction in write overhead. However, this assumption only holds true if each bit of the data corresponds to a single cell. If multiple bits are stored in each cell a distinction can be made between a reduction in bit-writes and a reduction in cell-writes. For example, when overwriting the byte 00011011 with 01111110 in single-level cells (SLC), half of the cells remain unaltered, but in 2-bit MLC every cell must be overwritten. A reduction in bit-writes therefore has no impact on MLC unless it also corresponds to a reduction in cell overwrites.

Since programming MLC cells requires setting them to more precise values than SLC, program-and-verify (P&V) is commonly used in almost all NVM technologies [9, 23, 17]. The P&V scheme works by first setting a cell to a fully SET or fully RESET state, followed by a series of smaller writes. Each write operation is followed by a read operation to determine when the intended value has been reached. As a result, the value of the data written to a cell has a significant impact on the latency and energy cost of the write access. The closer a state is to fully SET or fully RESET, the fewer write iterations are required to reach it, the lower the latency and energy cost. Table 1, Table 2, and Table 3 show the energy cost of writing

each value to a cell in SLC PCM [5], 2-bit MLC PCM [23], and 3-bit MLC ReRAM [17] respectively. It demonstrates that the central-most codewords (01 and 10 in 2-MLC, 011 and 100 in 3-MLC) have the highest energy cost in MLC NVMs. This discrepancy in the energy costs of the write data motivates modifications to FNW, FBA, and SBA to lower the number of high-energy writes and thereby reduce the energy cost of writes in MLC.

Table 1: Energy cost for SLC writes in PCM.

SLC	0	1
Energy (pJ)	19.73	14.03

Table 2: Energy cost for 2-bit MLC writes in PCM.

2-MLC	00	01	10	11
Energy (pJ)	36	307	547	20

Table 3: Energy cost for 3-bit MLC writes in ReRAM.

3-MLC	000	001	010	011	100	101	110	111
Energy (pJ)	2.0	6.7	19.3	35.1	35.6	19.6	8.5	1.5

Accounting for the differences in energy costs of each state requires one fundamental alteration to FBA and SBA. Rather than considering the Hamming weight of each node when performing the codeword assignment, the algorithms must consider the energy cost of the codewords. In the case of FBA this change simply requires the available codewords to be sorted in terms of energy cost instead of Hamming weight before assignment begins. Since SBA considers the transitions from state-to-state of the codewords, it is able to take into account the fact that same-cell overwrites in MLC have no energy cost, making the computed edge weight dependent on both the cell-wise Hamming distance between the considered nodes and the energy costs of the cells in the overwriting state.

When performing codeword assignment to reduce the number of bit-transitions in SLC NVMs, the direction of transitions between nodes was inconsequential, since only the number of bit-transitions was relevant to the analysis. In the case of an energy-based codeword assignment scheme for an MLC NVM, the direction is relevant to the analysis since the energy cost of a transition depends on the codeword being written. This distinction is not important to FBA which only considers the frequency of occurrence for each codeword, but is necessary for SBA since it is concerned with the sequence in which codeword transitions occur.

4.2 FREQUENCY-BASED ASSIGNMENT

Performing FBA in an MLC context requires codeword assignments to be performed in order of energy cost rather than Hamming weight. The SLC-based FBA begins by sorting the nodes by frequency, then generates limited weight codes and assigns them in order such that the highest frequency nodes are given the lowest Hamming weight codewords. The same principle can be applied in MLC, but since Hamming weight is irrelevant in MLC the codewords are sorted instead by the cost of their write energies. Unlike with LWCs, the complexity of generating the next lowest write-energy codeword is sufficient that it is most practical to simply generate all codewords, compute their costs, and pre-sort them before assignment begins.

Figure 8 shows an example of FBA using 2-bit MLC codewords. The transition matrix on the left represents the number of transitions from each node to every other node, similar to the graph in Figure 4. The values of the nodes are shown in hexadecimal, so the uppermost cell shown in gray represents that the value 0000 is overwritten by 0001 90 times. A 16-node graph is necessary for this example, since is performed using 2-bit MLC cells and a minimum of 2 cells per codeword is necessary to fully demonstrate the codeword assignment algorithms.

A list of codewords sorted by energy cost is shown next to the transition matrix. The order of the codewords is determined by summing the cost to write each cell value contained

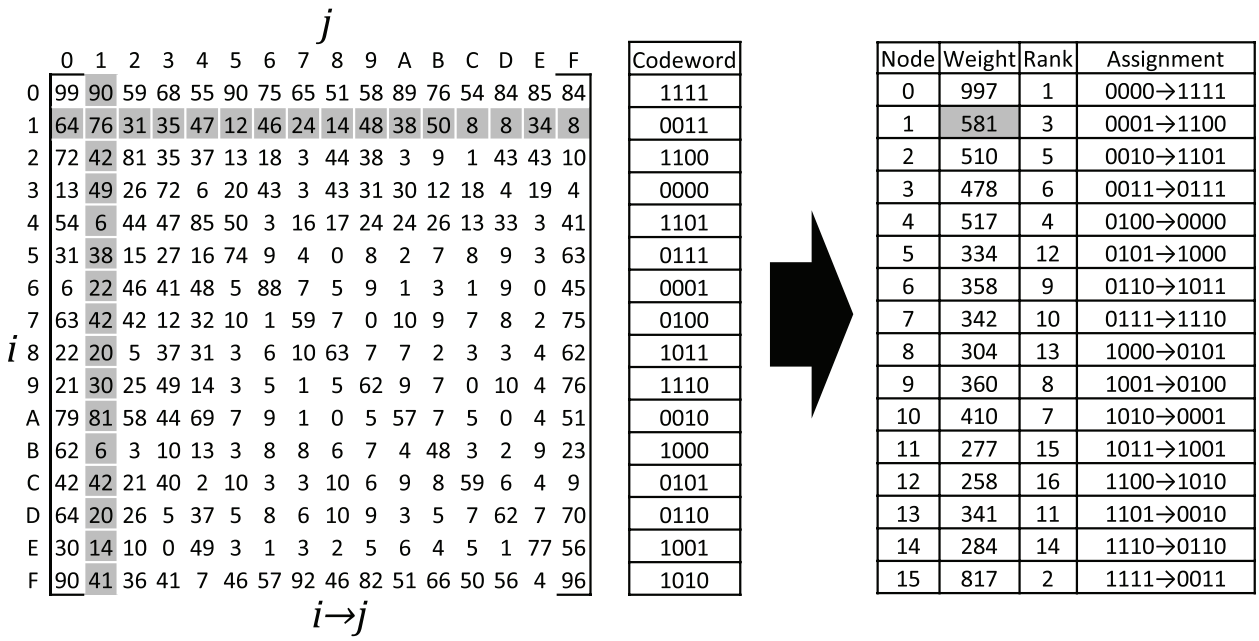

Figure 8: Example of FBA in 2-bit MLC.

in the codeword. For instance, 1111 is listed first because its energy cost is 40 pJ, less than any other codewords. Codewords comprised of the same set of values (e.g. 1100 and 0011) have the same energy cost, so they are sorted by binary value. The sorted list of codewords is used to assign the best codewords to the highest ranked nodes, as in the SLC-based FBA algorithm.

Node ranking is performed identically to SLC-based FBA, using equation 3.3. The gray cells in the transition matrix in Figure 8 show the transitions involved in calculating the weight of the node 1. Ranking the calculated weights from highest to lowest provides the order in which the codewords will be assigned, resulting in the assignments shown in the final column of the table. These assignments reduce the total energy consumption of the transitions in the graph from 20.5 µJ to 18.9 µJ.

4.3 SEQUENCE-BASED ASSIGNMENT

The algorithm given for SLC-based SBA in Figure 6 can be applied in MLC with a single modification, shown in Figure 9. Determining the importance of a node based strictly on the sum of the products of its Hamming distances from previously assigned nodes to the corresponding transition frequencies is too simplistic to minimize the energy cost in MLC. Instead of using the product of Hamming distances with the corresponding two-directional transition frequencies, the MLC algorithm takes the sum of the product of each energy cost with the corresponding one-directional transition frequency. Note that the energy function in Figure 9 is the cost to overwrite its first parameter with its second. This is equivalent to the energy cost of simply its second parameter except that same-cell-value overwrites do not incur an energy cost (e.g. overwriting 1000 with 1011 requires only 20 pJ).

Figure 10 shows the application of MLC-based SBA to the transition matrix shown in Figure 8. The table labeled reference lists the nodes and codewords referenced by the transitions and weights respectively in the following steps. Step 1 has been omitted because it is identical to the procedure in Figure 8; the most frequent node is assigned to the lowest energy codeword. However, instead of continuing to assign each subsequent codeword to the next node, the most interconnected node is determined in each step and the lowest energy codeword is calculated based on the previous codeword assignments.

Codeword 1111 was assigned to node 0 in step 1, so they are each missing from step 2. After step 1 the transition frequencies are calculated based on the previously assigned nodes. In step 2 each of the transition frequencies is the total number of transitions to and from node 0. Node 15 has a total of 174 transitions with node 0, so it will be assigned the next codeword. The weights of the codewords are calculated using the equation in Figure 9.

$$\text{code}[n] \leftarrow \min_{c \in \mathcal{C}} \left(\sum_{j \in \mathcal{A}} (\text{energy}(n, \text{code}[j]) t_{nj} + \text{energy}(\text{code}[j], n) t_{jn}) \right)$$

Figure 9: Modification to the pseudo-code for MLC SBA.

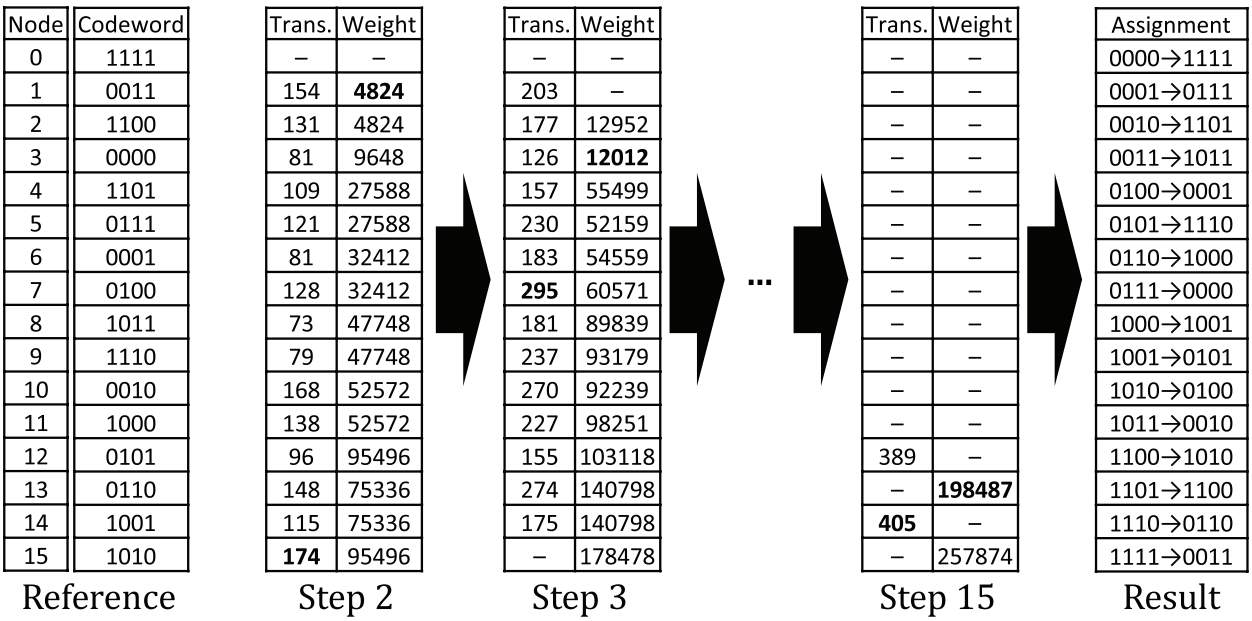

Figure 10: Example of SBA in 2-bit MLC.

Both 0011 and 1100 have a cost of only 36 pJ transitioning from, and 20 pJ transitioning to 1111. Multiplying these costs with the corresponding transition frequencies and taking the sum of each direction results in a weight of 4824 for both codewords, resulting in 0011 being assigned to node 15. The algorithm is complete once every node has been assigned; in this example it reduces the energy consumption from 20.5 µJ to 18.5 µJ.

4.4 FLIP-N-WRITE

Attempting to implement FNW in 2-bit MLC raises the question of what exactly a flip means in the context of 4-state cells. Figure 11 shows the two different approaches to implementing FNW in 2-bit MLC. A bitwise flip can be implemented just as with SLC, where a 00 cell would become 11 and a 01 cell becomes 10 if flipped. Alternatively, an energy-cost flip can be implemented to allow the cells to avoid the highest energy states. In this case a 00 cell would become 01 when flipped and a 10 cell would become 11 when flipped. Providing

these transitions instead of the former allows the flip operation to reduce the energy costs of transitions by giving the option of mapping the cells written to much lower energy states. However, since the lowest energy cost cell states typically occur the least frequently, which approach further reduces energy consumption depends on the characteristics of the data. We simulate and compare each approach in our results section.

bitflip	$00 \leftrightarrow 11, 01 \leftrightarrow 10$
cellflip	$00 \leftrightarrow 01, 10 \leftrightarrow 11$

Figure 11: Different modes of FNW-MLC.

5.0 EVALUATION AND RESULTS

In this chapter we describe the framework used to simulate our algorithms and discuss the results of the simulations.

5.1 SIMULATION FRAMEWORK

The proposed architecture is evaluated using an in-house trace-driven simulator. The traces used are generated from benchmarks in the SPEC CPU2006 [1] benchmark suite. These benchmarks reflect a variety of real integer and floating-point based workloads used by modern computing systems. The memory traces from the benchmarks are recorded using the Intel Pin Binary Instrumentation Tool [15] on a machine running a 2.3GHz Intel Core i7 CPU. Our tool captures memory accesses from the processor and simulates a typical cache system, recording only those accesses sent to main memory. This tool simulates a separate 32 KB 4-way associative I-cache and 32 KB 8-way associative D-cache at L1, with a shared 256 KB 8-way associative L2 cache, and an 8 MB 16-way associative L3 cache. When gathering the traces, the benchmarks are first run through 1 billion instructions to avoid memory accesses from program initialization, then are run until 1 million memory write operations have been recorded.

We run simulations to evaluate and compare the performance improvement of our method with two state-of-the-art methods, Flip-N-Write [6] (FNW) and Data-Comparison Write [26] (DCW). These two methods are similar to ours because they also perform a bitwise comparison to only update changed bits. Our simulator evaluates each method by processing each write access in the trace file, performing the required data modification where appropri-

ate (bitwise inversion for FNW and codeword substitution for our method), and storing the data in a data structure representing the contents of memory. For all methods, our simulator compares the new data against the existing data in the data structure before replacing it to determine which bits are different and would undergo a bit-write operation in a real device.

5.2 BIT-WRITES

The results of the half-word-level codeword assignment are shown in Figure 12. 12-rapid-SBA performs best, which consistently incurs fewer bit-writes than three other methods, resulting in a 54% improvement over DCW, a 40% improvement over FNW, and a 9.9% improvement over FBA. FBA also consistently outperforms DCW and FNW by 56.4% and 38.2%, respectively.

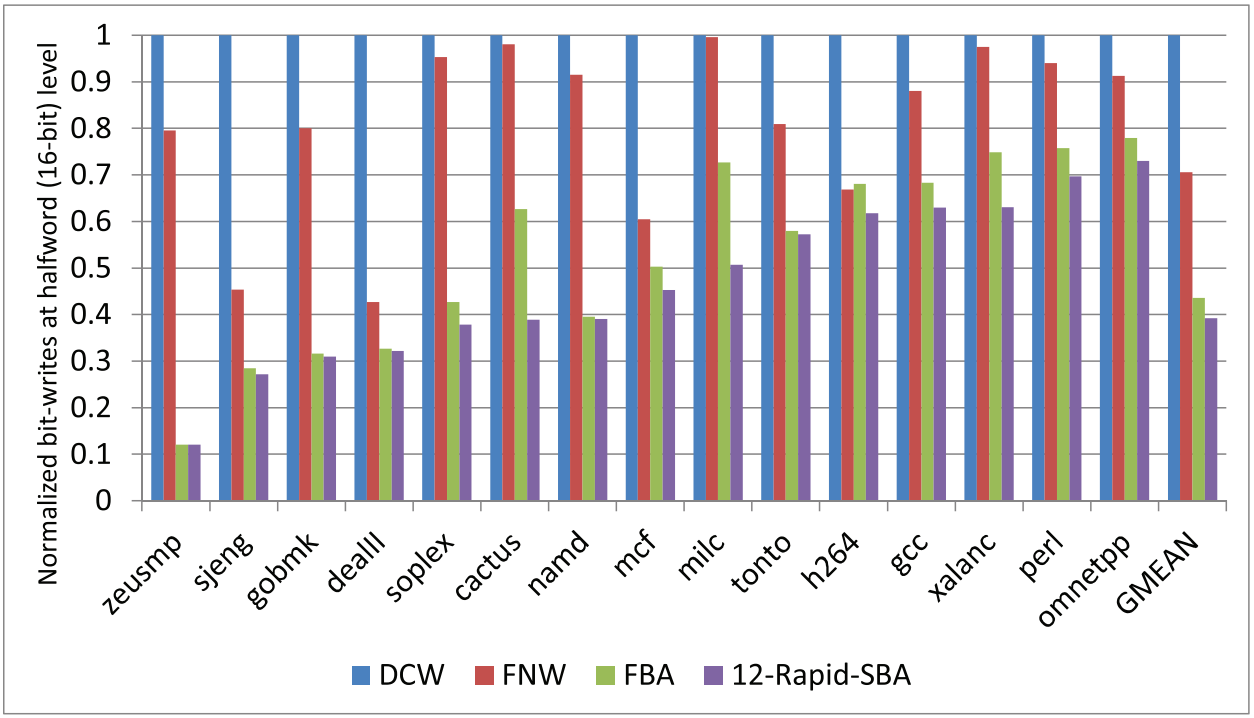

Figure 12: 12-rapid-SBA and FBA at the half-word level with 16-LWC codewords.

The results of the byte-level codeword assignment are shown in Figure 13. In 11 out of 16 benchmarks, SBA results in the smallest number of bit-writes, in comparison to DCW, FNW, and FBA. On average, it reduces the number of bit-writes by 45.7%, 19.3%, and 11.9% over DCW, FNW, and FBA respectively. FBA also outperforms FNW in 7 benchmarks, resulting in 38.4% and 8.5% improvements over DCW and FNW on average. Note that FBA has the worst performance in milc benchmark, while SBA reduces the number of bit-writes by half in comparison to DCW. The reason is that the most frequently occurring values do not transition to each other frequently in the milc benchmark, as previously described in Figure 5. In a few benchmarks with the even distribution of frequencies, which is opposite to the distribution shown Figure 3, neither SBA nor FBA is able to reduce the number of bit-writes over FNW at the byte level.

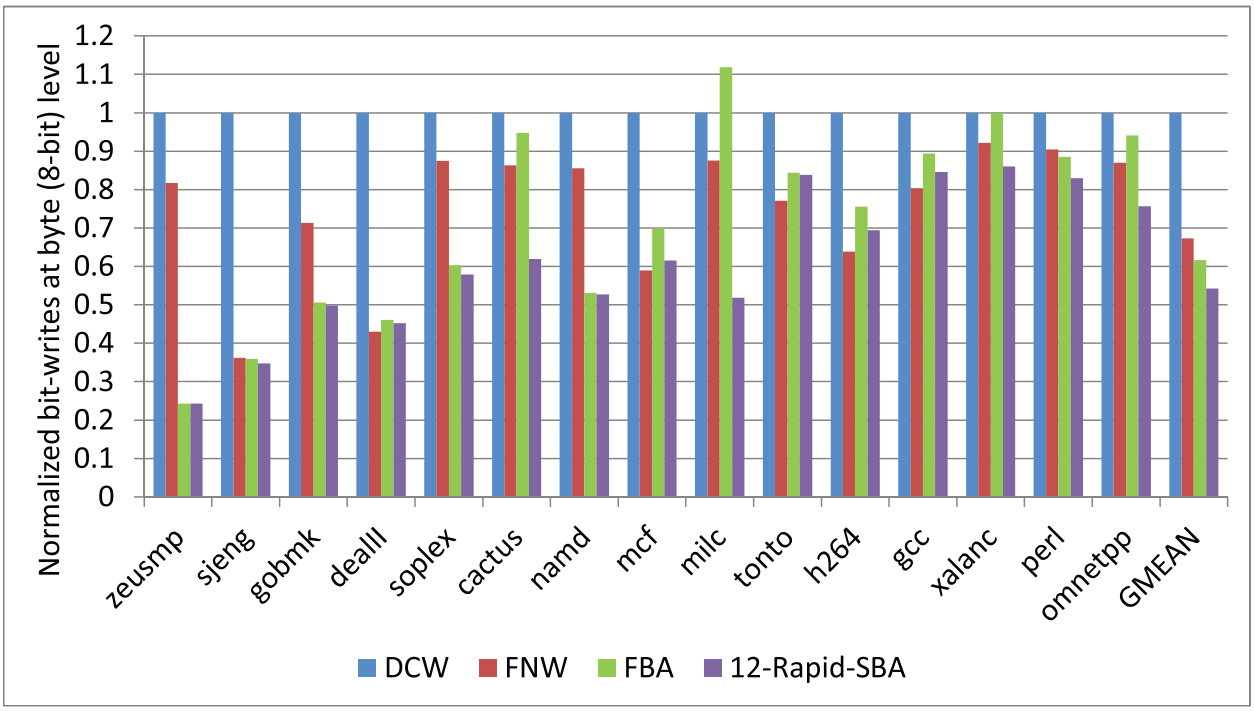

Figure 13: SBA and FBA at the byte level with 8-LWC codewords.

Similar results have been reported in NVMs using a compression-based architecture. Using a frequent-pattern compression algorithm on the data written to memory resulted in a 20% reduction in bit-writes compared to FNW [8], only marginally better the 19.3% reduction achieved by SBA.

5.3 CELL-WRITES

The results of the 2-bit MLC codeword assignments can be seen in Figure 14. SBA outperformed DCW, FBA, and both versions of FNW in 7 of the benchmarks. On average it resulted in a 41.5% reduction over DCW, a 9.1% reduction over FBA, a 15.8% reduction over bitflip FNW, and a 35.6% reduction over cellflip FNW. FBA also performed well on average, providing a 35.7% reduction over DCW, a 7.3% reduction over bitflip FNW, and a 29.2% reduction over cellflip FNW. The xalanc benchmark was the only case where FBA caused

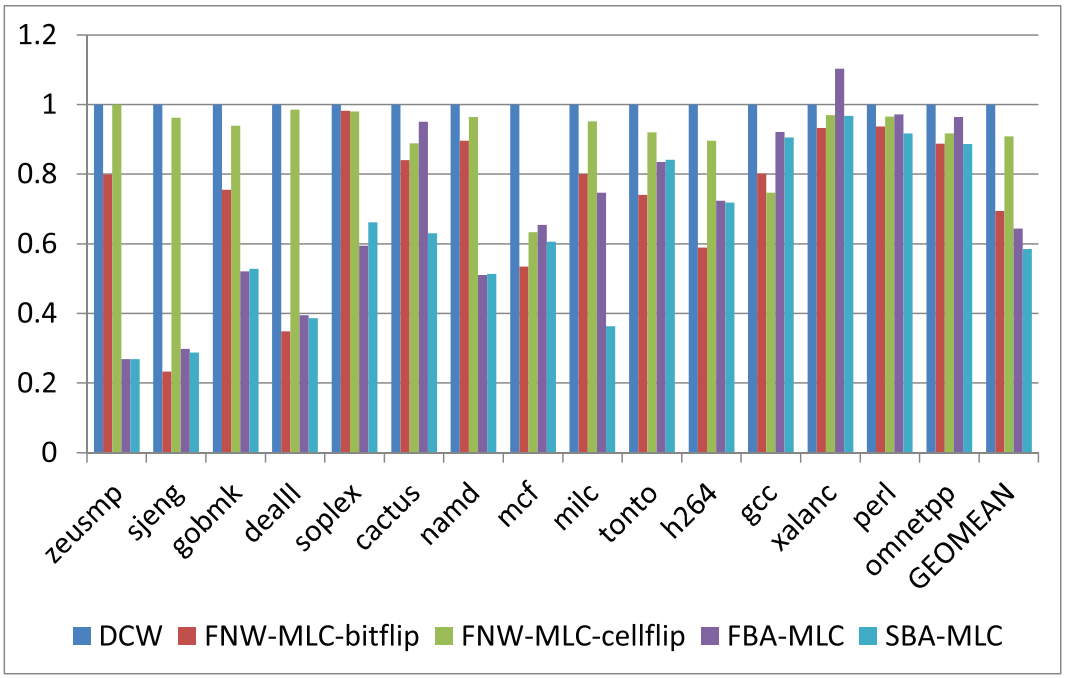

Figure 14: Comparison of the number of cell-writes, normalized by that of DCW.

more cell-writes than DCW, due to the combination of the imprecise nature of FBA (when compared to SBA) and the lack of room for improvement by codeword mapping at a 2-bit cell level in the xalanc benchmark. The fact that cellflip FNW performed poorly in terms of cell-write reduction should not be surprising, since it only provides the option of switching from frequently-occurring states to infrequently-occurring states. A large reduction in cell writes is only possible if flips can occur frequently, which bitflip FNW allows much more often than cellflip FNW.

5.4 ENERGY COSTS

Figure 15 shows the energy costs of the 2-bit MLC codeword assignments. On average SBA reduced the write energy by 57.2% over DCW, 25.3% over bitflip FNW, 33.9% over cellflip FNW, and 10.6% over FBA. It was able to provide the greatest reduction in write energy in

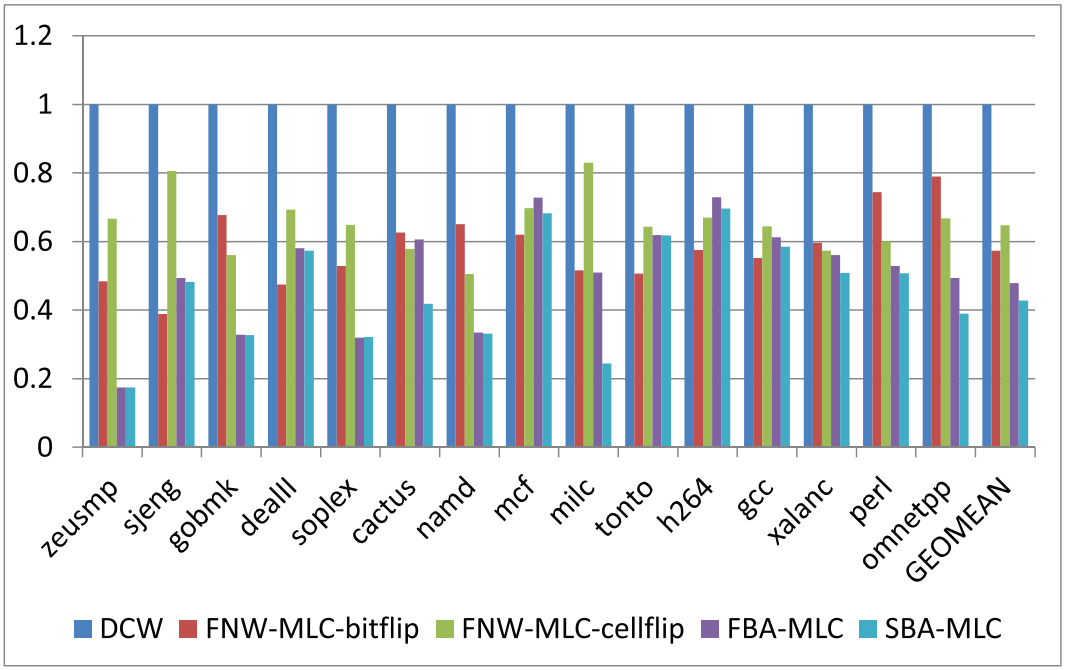

Figure 15: Comparison of the energy costs, normalized by that of DCW.

11 of the 16 benchmarks. Similarly, FBA outperformed all benchmarks other than SBA in 8 of the benchmarks, and outperformed DCW, bitflip FNW, and cellflip FNW in the average case by 52.1%, 16.4%, and 26.0% respectively. Although cellflip FNW was able to reduce energy consumption with respect to bitflip FNW in 6 of the benchmarks, in the average case it increased the energy cost by 12.9%. Its ability to flip between the highest and lowest energy states allowed each flip to conserve more energy, but the scarcity of high-energy states in most benchmarks resulted a significant reduction in the number of flips available.

5.5 SENSITIVITY ANALYSIS

Two different values of m were considered for the m-LWC codes used by SBA at the byte level. 8-LWC has no bit overhead and merely consists of a direct remapping of the values based on their frequencies. The 4-LWC uses a single bit of overhead per byte to ensure a maximum Hamming distance of 4 bits per write. This is the same amount of bit overhead

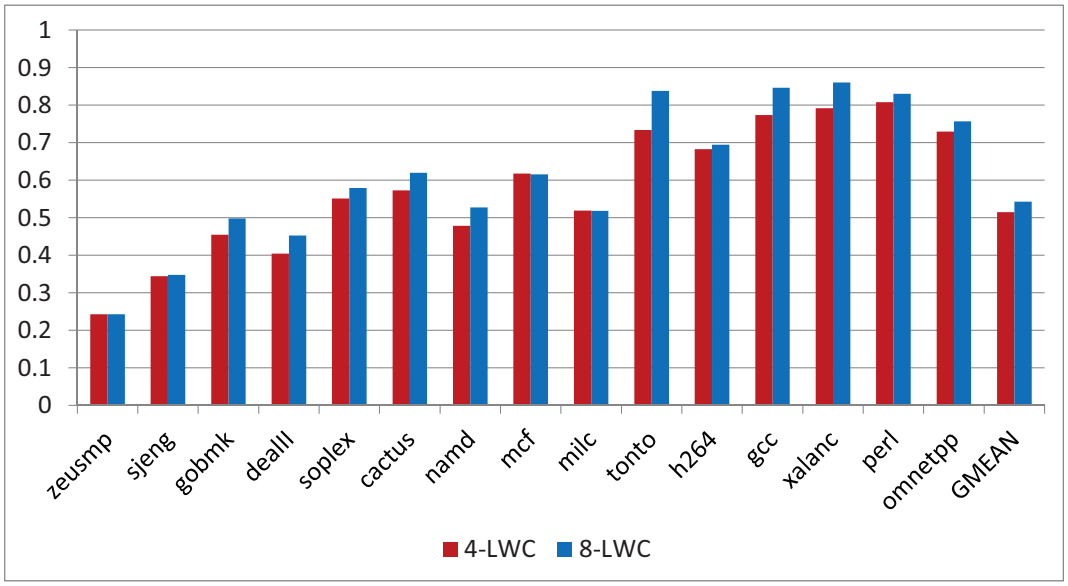

Figure 16: Numbers of bit-writes with 4-LWC and 8-LWC SBA at the byte level.

as FNW would require at the byte level. The numbers of bit write with 4-LWC and 8-LWC, normalized by that with DCW, are shown in Figure 16. 4-LWC provide a 5% decrease in the number of writes compared to 8-LWC. However, the 12.5% overhead in capacity required by the 4-LWC coding makes the 8-LWC a more compelling option.

We also evaluate our propose n-rapid SBA at the half-word level by performing the sensitivity analysis of bit-write reduction to the value of n. Figure 17 shows the numbers of bit-writes of n-rapid SBA with $n = 8/10/12$, normalized by that of DCW. The bit-write reduction in some of the benchmarks, such as zeusmp and soplex, plateaued immediately, since the several most frequent half-words are so disproportionately common. In general, the improvement in bit-write reduction, as n increases, is limited, meaning that our proposed n-rapid SBA can provide reasonably good performance while significantly reducing the computational overhead.

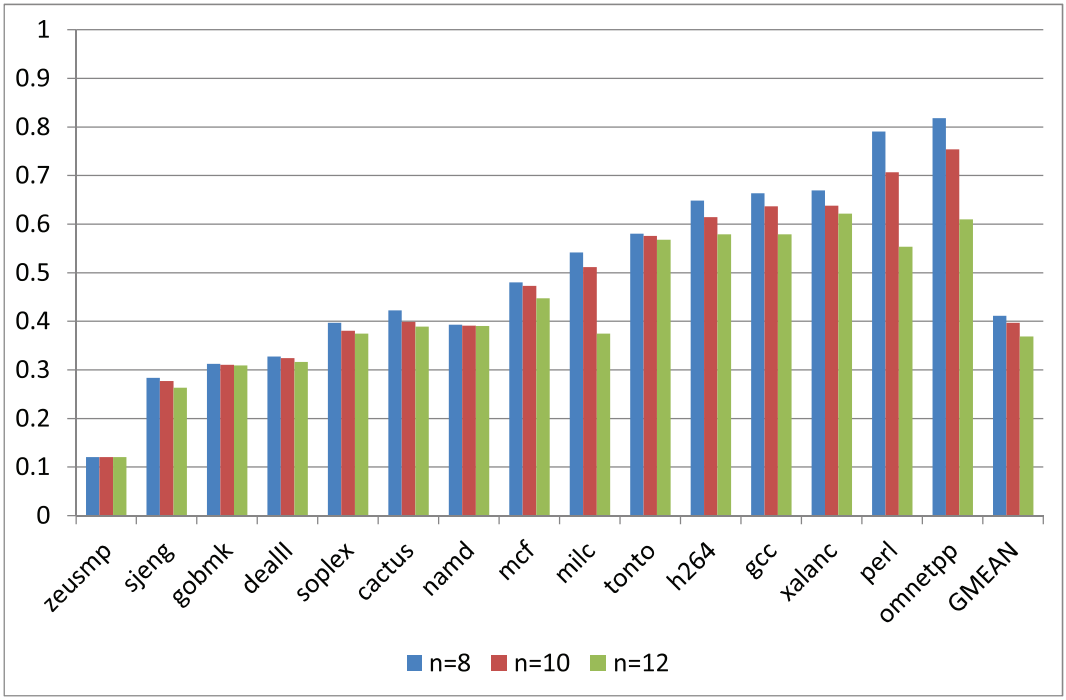

Figure 17: Numbers of bit-writes with n-rapid SBA at the half-word level.

6.0 CONCLUSIONS AND FUTURE WORK

In this chapter we provide a summary of our work, and a discussion of our contributions and future work.

6.1 SUMMARY

In Chapter 2 we provide the background to our work by discussing the current state of non-volatile memories. We first describe the operation of PCM, ReRAM, and STT-RAM, and then provide an overview of the difficulties and trade-offs involved in replacing DRAM with each technology. We also consider coding techniques which have been proposed for memory devices and describe the architecture necessary to implement a context-independent coding technique in a memory device.

In Chapter 3 we describe two codeword assignment techniques based on context-independent codes. We begin by explaining how codewords can be generated and assigned, then describe the details of the two codeword assignment algorithms. We also discuss a hybrid approach which can be implemented for reduced assignment complexity.

In Chapter 4 we explore the application of the codeword assignment techniques to MLC. We provide motivation by explaining how codeword assignment in MLC is unique from SLC, then describe the changes to each technique.

In Chapter 5 we describe the framework used to perform our simulations, then present and discuss the results of our evaluations. Each coding technique is compared to the others using the criteria of bit-writes in SLC, cell-writes in MLC, and energy costs in MLC. We also perform a sensitivity analysis to demonstrate how adjustments to the size of the codewords and the complexity of the codeword assignment algorithm affect the results.

6.2 CONTRIBUTIONS

The contributions made to non-volatile memories are as follows:

- The application of frequency-based assignment of codewords to memory devices, which allows for a reduction in bit-writes. A reduction in the number of bit-writes mitigates many of the problems with current NVMs including cell wear and energy cost.
- The application of sequence-based assignment of codewords to memory devices, which further reduces the number of bit-writes. This approach requires much more complex analysis of the dataset, but provides optimal assignments which minimize the number of bit-writes.
- A hybrid version of frequency-based and sequence-based assignments, called Rapid-SBA. This approach allows a trade-off to be made between the complexity of the analysis and the optimality of the codeword assignments. Codeword assignments performed at a level larger than one byte have a high level of complexity which require a hybrid approach to reduce the overhead of the a priori analysis.
- The development of a frequency-based assignment algorithm for multi-level cells. Due to the differences between SLC and MLC, simply using limited-weight codewords to perform assignment neither reduces cell-writes nor energy costs. Instead, an energy-cost-based codeword assignment is our proposed approach.
- The development of a sequence-based assignment algorithm for multi-level cells. In addition to the different codeword weights, sequence-based assignment for MLC requires a modification to its cost computation due to the lack of commutativity in multi-level cells.

- An exploration of an energy-based approach to Flip-N-Write in multi-level cells. The original design of Flip-N-Write attempts to reduce bit-writes by allowing for cell inversion. We explore an energy-cost-based approach to cell inversion in MLC with the goal of reducing the energy consumed.

6.3 CONCLUSIONS

In this thesis, we presented a codeword assignment architecture for cell-write reduction in emerging SLC and MLC non-volatile memories. Cell-write reduction has many practical benefits, including lower write latency, lower dynamic energy, and enhanced endurance. The proposed architecture relies on a priori analysis of memory access patterns and a remapping table to minimize the switching activity on NVM cells and reduce bit-writes. Trace-based simulations using of the SPEC CPU2006 benchmarks using sequence-based assignment show a 33× reduction in raw bit-writes in SLC, which corresponds to a 2–3× improvement over the best state-of-the-art methods and a 25% reduction of energy in MLC.

6.4 FUTURE WORK

Future expansion to our work can be performed in the following areas:

- When considering single-level cell assignments we considered 8-LWC and 4-LWC assignments. Decreasing the weights of the codewords increases the overhead, but also reduces the number of bit-writes. Exploring increasingly limited weight codewords (e.g. 2-LWC) is a potential avenues for further improvement.
- Codeword assignment was performed at 8-bit and 16-bit levels only. Codeword replacements at a 16-bit level performed better, so future work can be done to explore even further extending the size at which substitutions occur, such as word size or cache line size.
- The multi-level cells modeled in our simulations were only 2-bit MLC. Extending this work to higher cell densities is an addition area for future development of this work.

BIBLIOGRAPHY

[1] SPEC CPU2006, 2006.

[2] International Technology Roadmap for Semiconductors, 2011.

[3] I.G. Baek et al. Highly scalable nonvolatile resistive memory using simple binary oxide driven by asymmetric unipolar voltage pulses. In *Proc. Intl. Electron Devices Meeting*, 2004.

[4] K Basu et al. Power protocol: reducing power dissipation on off-chip data buses. In *Proc. Intl. Symposium on Microarchitecture*, 2002.

[5] Jie Chen et al. Energy-aware writes to non-volatile main memory. *ACM SIGOPS Operating Systems Review*, 2012.

[6] S. Cho and H. Lee. Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance. In *Proc. Intl. Symposium on Microarchitecture*, 2009.

[7] M. Choudhury et al. Single-ended coding techniques for off-chip interconnects to commodity memory. In *Proc. Design, Automation & Test in Europe Conference*, 2007.

[8] D. Dgien et al. Compression architecture for bit-write reduction in non-volatile memory technologies. In *Proc. Non-Volatile Memories Workshop (NVMW)*, 2014.

[9] Guiqiang Dong et al. Using data postcompensation and predistortion to tolerate cell-to-cell interference in mlc nand flash memory. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 2010.

[10] M. Guthaus et al. MiBench: A free, commercially representative embedded benchmark suite. In *Workshop on Workload Characterization*, 2001.

[11] L. Jiang et al. Improving write operations in MLC phase change memory. In *Proc. Intl. Symposium on High Performance Computer Architecture*, 2012.

[12] E. Kultursay et al. Evaluating STT-RAM as an energy-efficient main memory alternative. In *Proc. Int. Symposium on Performance Analysis of Systems & Software*, 2013.

[13] B. Lee et al. Architecting phase change memory as a scalable DRAM alternative. In *Proc. Intl. Symposium on Computer Architecture*, 2009.

[14] H.-G. Lee et al. A compression-based hybrid MLC/SLC management technique for phase-change memory systems. In *Proc. IEEE Annual Symposium on VLSI*, 2012.

[15] C.-K. Luk et al. Pin: Building customized program analysis tools with dynamic instrumentation. In *Proc. Conference on Programming Language Design and Implementation*, 2005.

[16] A. Mirhoseini et al. Coding-based energy minimization for phase change memory. In *Proc. Design Automation Conference*, 2012.

[17] D. Niu et al. Low power multi-level-cell resistive memory design with incomplete data mapping. In *Proc. Intl. Conference on Computer Design*, 2013.

[18] M. Qureshi et al. Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In *Proc. Intl. Symposium on Microarchitecture*, 2009.

[19] N. Seong et al. Security refresh: Protecting phase-change memory against malicious wear out. *IEEE Micro*, 2011.

[20] Mircea R Stan and Wayne P Burleson. Limited-weight codes for low-power I/O. In *Proc. Intl. Workshop on Low Power Design*, 1994.

[21] G. Sun et al. A frequent-value based PRAM memory architecture. In *Proc. Design Automation Conference*, 2011.

[22] Dinesh C Suresh et al. A tunable bus encoder for off-chip data buses. In *Proc. Intl. Symposium on Low Power Electronics and Design*, 2005.

[23] J. Wang et al. Energy-efficient multi-level cell phase-change memory system with data encoding. In *Proc. Intl. Conference on Computer Design*, 2011.

[24] Victor Wen et al. Exploiting prediction to reduce power on buses. In *IEE Proceedings-Software*, pages 2–13, 2004.

[25] G. Wu et al. CAR: Securing PCM main memory system with cache address remapping. In *Proc. Intl. Conference on Parallel and Distributed Systems*, 2012.

[26] B.-D. Yang et al. A low power phase-change random access memory using a data-comparison write scheme. In *Proc. Intl. Symposium on Circuits and Systems*, 2007.

[27] Jun Yang et al. Frequent value encoding for low power data buses. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 2004.

[28] Jun Yang and Rajiv Gupta. Frequent value locality and its applications. *ACM Transactions on Embedded Computing Systems*, 2002.

[29] J. Yue and Y. Zhu. Making write less blocking for read accesses in phase change memory. In *Proc. Intl. Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems*, 2012.

[30] J. Yue and Y. Zhu. Accelerating write by exploiting PCM asymmetries. In *Proc. Intl. Symposium on High-performance Computer Architecture*, 2013.

[31] P. Zhou et al. Energy reduction for STT-RAM using early write termination. In *Proc. Intl. Conference on Computer-Aided Design*, 2009.