Reducing Data Movement Energy via Online Data Clustering and Encoding

Shibo Wang, Engin Ipek
Department of Computer Science
University of Rochester
Rochester, NY 14627 USA
{swang, ipek}@cs.rochester.edu

***Abstract*—Modern computer systems expend significant amounts of energy on transmitting data over long and highly capacitive interconnects. A promising way of reducing the data movement energy is to design the interconnect such that the transmission of 0s is considerably cheaper than that of 1s. Given such an interconnect with asymmetric transmission costs, data movement energy can be reduced by encoding the transmitted data such that the number of 1s in each transmitted codeword is minimized. This paper presents a new data encoding technique based on online data clustering that exploits this opportunity.**

The transmitted data blocks are dynamically clustered based on the similarities between their binary representations. Each data block is expressed as the bitwise XOR between one of multiple cluster centers and a *residual* with a small number of 1s. The data movement energy is minimized by sending the residual along with an identifier that specifies which cluster center to use in decoding the transmitted data. At runtime, the proposed approach continually updates the cluster centers based on the observed data to adapt to phase changes.

The proposed technique is compared to three previously proposed energy-efficient data encoding techniques on a set of 14 applications. The results indicate respective energy savings of 5%, 9%, and 12% in DDR4, LPDDR3, and last level cache subsystems as compared to the best existing baseline encoding technique.

I. Introduction

Data movement is a major contributor to the total system energy consumption in deeply scaled CMOS ICs [1]. Studies show that for scientific and mobile applications, 40% [2] and 35% [3] of the total system energy is consumed by data movement, respectively. The energy cost of data movement is substantially higher than that of computation. For example, when performing a double precision addition on a graphics processing unit (GPU) implemented at the 22nm technology node, fetching the two operands from memory consumes 50× greater energy than moving the operands from the edge of the chip to its center, which in turn consumes another 10× higher energy than the addition [4]. This orders of magnitude energy gap between data movement and computation is expected to widen in the future with technology scaling [5]. Thus, reducing data movement energy is critical to future computer systems.

A promising way of reducing the data movement energy is to design the interconnect such that the transmission of **0**s is considerably cheaper than that of **1**s. Given such an interconnect with asymmetric transmission costs, data movement energy can be reduced by encoding the transmitted data such that the number of **1**s in each transmitted codeword is minimized. Existing coding techniques that can exploit this asymmetry to reduce data movement energy include the DBI coding technique adopted by DDR4 DRAM [6], [7], two dimensional bus invert (CAFO) coding [8], and recent value encoding [9]. However, these techniques either use fixed coding methods that cannot adapt to the data patterns across different applications, or are restricted to exploring limited set of data patterns at runtime. These shortcomings leave room for designing a more effective data encoding technique to reduce the energy consumption due to data movement.

This paper proposes a new data encoding method based on online data clustering. In the proposed coding scheme, the transmitted data are dynamically grouped into different clusters based on their similarities, as shown in Figure 1. The similarity is evaluated based on the Hamming distance (*i.e.*, the number of bit positions that differ) between two data blocks. Each cluster has a center with a bit pattern close to those of the data blocks that belong to that cluster. These cluster centers are learned online and stored at both the transmitter and the receiver. A transmitted data block is encoded by the residual that results from XORing the contents of the block with the nearest cluster center, concatenated with a center identifier. The original data block is recovered by XORing the received residual with the identified center. By dynamically learning the cluster centers and assigning each data block to the nearest center, the sum of the Hamming distances to the cluster centers is minimized. As a result, the total number of **1**s in the transmitted data blocks is significantly reduced, leading to substantial savings in data movement energy.

Online data clustering allows the proposed coding scheme to adapt to changes and to customize data encoding to different applications. As compared to coding techniques that use fixed frequent data values [10] or recent data values [9] to represent the data, the proposed data encoding scheme can capture data patterns that appear both recently and in the distant past. In addition, the cluster centers in the proposed scheme are not required to be present in the actual data that is transmitted. For example, given the data patterns 1110, 1101, and 1011, the proposed approach can place a cluster center at 1111, which results in a Hamming distance of 1 to each of the three data

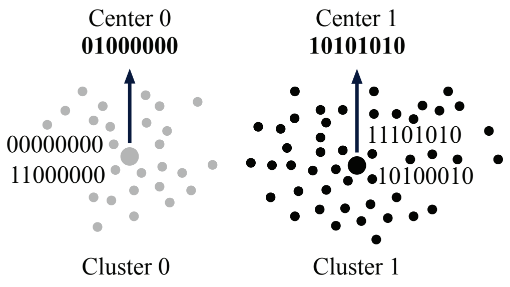


Fig. 1. Illustrative example of clustering 8-bit data blocks based on similarity. The small data points represent the transmitted data blocks. The data points in different colors belong to different clusters.

patterns in that cluster. This feature increases the number of possible data patterns that can be chosen as a cluster center, thereby improving the odds of finding a representation with a few ones for each encoded data block. All of these advantages make the proposed data encoding scheme better at reducing the energy consumption of data movement as compared to existing techniques.

The proposed coding scheme is applied both to DRAM interfaces and the main data H-tree in a last level cache (LLC) for evaluation. The proposed approach reduces the average energy consumption of data movement by 40% as compared to conventional DBI coding. The overall energy of DDR4 and LPDDR3 DRAM systems are respectively reduced by 9% and 15%. Depending on the cache size, the energy consumption of the LLC is reduced by 15% to 21%. These energy reductions are achieved at a cost of less than 0.5% performance degradation as compared to a system using conventional DBI coding. Furthermore, as compared to two other recently proposed encoding techniques, CAFO [8] and recent value encoding [9], the proposed coding scheme respectively reduces the data movement energy by 30% and 20%.

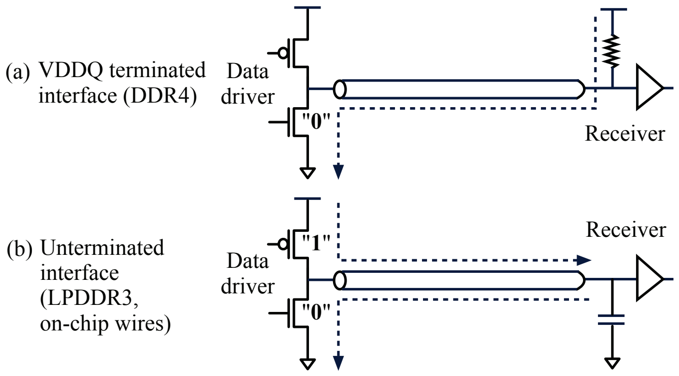
II. Background and Related Work

To reduce the energy consumption of data movement, it is important to understand how the energy is consumed when data are transmitted on different types of interconnects, and how existing energy efficient data encoding techniques are designed. The basics of the data clustering algorithm that inspired the proposed online clustering and encoding scheme are also reviewed in this section.

A. Energy Consumption of Data Movement

Different types of interconnects consume energy in different ways. Some DRAM standards provide basic coding schemes that improve the energy efficiency of the IO interface.

1) Terminated Interconnects: Many DRAM interfaces adopt on-die termination (ODT) to maintain signal integrity at a high clock speed. Recent DDR3/4, GDDR4/5, and LPDDR4 interfaces all support ODT in different forms. DDR4 and GDDR5 adopt a pseudo open drain (POD) signaling scheme with VDDQ termination [6], [11], as shown in Figure 2 (a). The IO interface consumes energy when transmitting a **0** as the current flows from VDD to GND; transmitting a **1** is effectively free. This asymmetric energy cost provides the opportunity for coding techniques to reduce the energy consumption of DDR4 and GDDR5 interfaces by reducing the number of transmitted **0**s. Different from DDR4 and GDDR5, LPDDR4 adopts a low-voltage-swing terminated logic (LVSTL) with VSSQ termination to satisfy speed requirements with noise immunity [12]. It consumes energy when transmitting a **1**; transmitting a **0** is free. For simplicity and without loss of generality, a terminated interface that consumes energy during the transmission of **1**s is used as an example for the rest of this paper.



(a) VDDQ terminated interface (DDR4)
Data driver
"0"
Receiver
(b) Unterminated interface (LPDDR3, on-chip wires)
Data driver
"1"
"0"
Receiver

Fig. 2. Energy consumption of different interconnects: (a) a VDDQ terminated interface; (b) an unterminated interface.

To reduce the IO energy and simultaneous switching output (SSO) noise, DDR4 chips with $\times 8$ and $\times 16$ configurations [6], [7] adopt data bus inversion (DBI) coding. DBI coding can be used to minimize either the number of **0**s or **1**s in a transmitted block of data; in the case of DDR4, DBI is used to guarantee that the number of **0**s in each encoded byte is less than or equal to four. DBI coding is performed by checking the number of **0**s within each byte of data that needs to be transmitted. If the number is greater than four, the data block is inverted and transmitted over the interconnect. A DBI bit, which is transmitted over an extra pin, is set to **0** to signal that the transmitted data block is in the inverted form. Otherwise, the data block is transmitted in its original form with the DBI bit set to **1**.

2) Unterminated Interconnects: Many mobile systems use a point-to-point unterminated DRAM interface such as LPDDR3. As shown in Figure 2 (b), the energy consumption of the unterminated LPDDR3 IO interface is caused by charging and discharging the load capacitance of the data bus [13], [14]. Unlike the asymmetric energy consumption of a terminated interface, the energy consumption of the LPDDR3 IO interface is context-dependent, and is proportional to the number of $\mathbf{0} \rightarrow \mathbf{1}$ and $\mathbf{1} \rightarrow \mathbf{0}$ transitions on the data bus. Since the energy consumption of an unterminated interface is not directly related to the number of **0**s in the transmitted data, applying DBI coding is not guaranteed to reduce the data movement energy.

Although LPDDR3 in its unterminated configuration does not provide native coding methods, bus invert (BI) coding [15] can be used to save energy. BI coding can be considered a special version of DBI coding that is designed for reducing context-dependent data movement energy. Similar to DBI coding, BI coding is also applied to an 8-bit data block with the help of an extra pin, which is used to transmit the BI bit. However, rather than checking the exact value of the data to be transmitted, BI coding checks whether the number of bits that differ between the current and the previously transmitted data is greater than four. If so, each bit of the original data block is flipped and transmitted with the BI bit set to **1**. Otherwise, the BI bit is set to **0** and transmitted with the original byte of data.

High Bandwidth Memory (HBM) [16] and Wide I/O 2 [17] are memory standards that use wide interfaces to deliver high bandwidth with high power efficiency. Both standards use unterminated interfaces, and adopt the BI coding scheme (called DBIac in the standards) to reduce the interface energy consumption [16], [17].

Transition signaling (discussed in Section IV-C) is another solution to save the energy consumption of an unterminated interface. When transition signaling is used, the number of transitions on the interconnect is equal to the number of **1**s in the transmitted data, which allows an unterminated interface to adopt coding techniques that reduce data movement energy by reducing the number of transmitted **1**s.

On-chip interconnects such as the H-trees in caches are also unterminated interconnects, and consume energy in the same way as unterminated DRAM interfaces. The techniques that are used by unterminated DRAM IO to reduce data movement energy can also be applied to on-chip interconnects to improve their energy efficiency.

3) Other Interconnects: Hybrid Memory Cube (HMC) [18], [19] employs SerDes links in its IO interface to provide high bandwidth and energy efficiency. However, due to the high static power consumption of SerDes links, HMC consumes more energy than conventional memory interfaces when the data bus is underutilized. In addition, the high static interface power prevents the energy efficient data encoding techniques from effectively reducing data movement energy. Udipi *et al*. [20] propose a memory interface similar to HMC that uses photonic interconnects instead of the SerDes links. Photonic interconnects require significant hardware changes and also dissipate high static power. The proposed energy efficient coding scheme is not applicable to SerDes links or to photonic interconnects.

B. Energy Efficient Data Encoding Techniques

Since BI coding [15] was introduced, energy efficient coding techniques have been heavily studied. Comprehensive surveys on data encoding schemes that can reduce on/off-chip interconnect power can be found in the literature [21], [22].

The M limited-weight codes (M-LWCs) [23] are a class of codes that guarantee the Hamming weights of the codewords are no more than M. They can achieve a relatively small M for energy savings at the cost of a wide codeword that may need to be transmitted in multiple cycles. For example, a 3-LWC encodes a byte of original data into a 17-bit codeword [24]. More is Less [25] exploits data bus underutilization by opportunistically using wide, sparse codes without significant performance degradation. In contrast, the coding scheme proposed herein does not require a wide codeword, and keeps the extra pin overheads the same as the DBI coding used in conventional memory interfaces.

Frequent value (FV) encoding [10] maintains a table to store a set of frequent data values that can be identified offline or online. Before transmitting the original data, an encoder compares the data to the values stored in the table. If there is a hit, the identifier of the table entry is transmitted using a one-hot code (1-LWC). Otherwise, the original data is transmitted. VALVE and TUBE improve FV by detecting variable length frequent values [26]. Constrained to using the exact data values that have appeared in the past transmissions, these coding techniques do not fully exploit data similarity.

A previously proposed XOR-based coding technique [27] exploits the temporal locality of the transmitted data by encoding each data block as the XOR between the current and the perviously transmitted data values. Komatsu *et al*. [9] propose a recent value coding technique that extends the XOR-based coding by keeping the most recent data values in a table. The original data is encoded as the identifier of the table entry that has the smallest Hamming distance from the original data, concatenated with the residual that results from XORing the table entry and the original data. In contrast, by using the learned cluster centers rather than the most recent data as the coding bases, the proposed method can capture representative data patterns that appear both recently and in the distant past. In addition, the proposed coding scheme can use data values that do not exist in the transmitted data set as centers, which further increases the potential to find representative data patterns.

Not limited to energy efficient data movement, data encoding schemes also find applications in the area of emerging memory technlogies. Flip-N-Write [28] uses BI coding to improve the write bandwidth, energy efficiency, and endurance of phase change memory (PCM). The recently published CAFO [8] coding uses an up-to-date two dimensional BI coding scheme to provide high endurance and reliability for PCM and spin-transfer torque random access memory (STT-RAM). The two dimensional BI coding organizes each data block as a matrix, and iteratively applies BI coding to each row and column until no more Hamming weight reduction can be achieved.

C. K-Majority Clustering

Data clustering is a machine learning technique that groups a set of data points into clusters such that points within each cluster are similar to each other. In centroid-based clustering, each cluster is represented by its centroid, which may not necessarily be a member of the clustered data set. K-means is a clustering algorithm in which each data point is assigned to the nearest cluster center such that the sum of the squared distances from the cluster centers is minimized [29], [30]. K-majority is a variant of k-means clustering proposed for binary data. It replaces Euclidean distance with Hamming distance to evaluate the similarity between bit-strings, and uses the majority vote rather than the arithmetic mean to recalculate the cluster centers. Similarly to the k-means clustering algorithm, k-majority proceeds by alternating between two steps: assignment and update [31]. In the assignment step, each binary data point is assigned to the nearest center based on the Hamming distance. In the update step, a new center is regenerated for each cluster using a majority vote—each bit of the new center is set based on the majority of the data points that are assigned to that cluster. This iterative process stops when the centers stop changing.

III. Key Idea: Learning the Encoding at Runtime

The goal of the proposed runtime encoding technique is to arrive at a sparse representation of the data—*i.e.*, a representation with few **1**s—prior to transmission on an interconnect with asymmetric energy costs. The key idea is to dynamically group similar data into clusters, and to represent each transmitted data block as the XOR between the nearest cluster center (*i.e.*, the most similar data pattern) and a residual. Copies of the learned cluster centers are kept at both the transmitter and the receiver. Instead of transmitting the original data blocks, only the residuals and the center identifiers are transmitted as the encoded data. The original data is recovered by XORing the received residual and the identified cluster center. By learning the proper cluster centers, the Hamming distance between each transmitted data block and the nearest cluster center is kept small. As a result, the number of **1**s in the transmitted residual is substantially reduced, leading to significant energy savings.

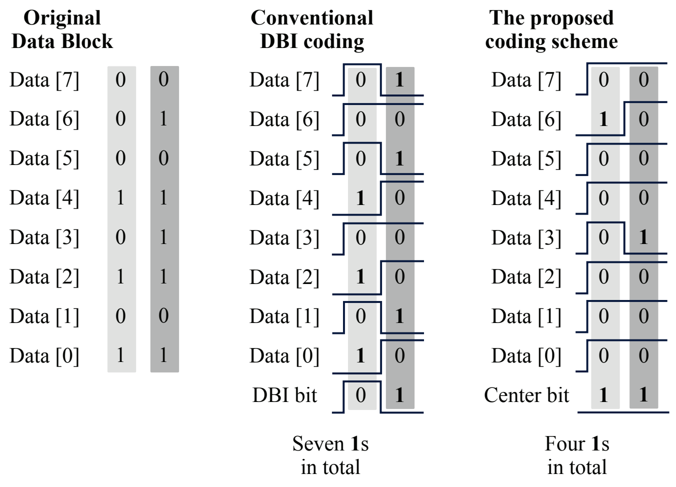


Fig. 3. Illustrative example of the key idea. The goal of the shown DBI coding is to minimize the number of 1s.

DBI coding can be considered a special case of the proposed coding scheme with two fixed centers: all **0**s and all **1**s. By dynamically learning centers that more effectively represent the data patterns that appear in the data stream, the proposed coding scheme significantly outperforms DBI coding. Figure 3 shows an example of data transmission over an interconnect with asymmetric transmission costs under DBI coding [1] and the proposed coding scheme. In the example, two bytes of data (00010101 and 01011101) are sent from the transmitter to the receiver. Under DBI coding, the first byte remains in its original form with the DBI bit set to **0**; the second byte is flipped with the DBI bit set to **1**. In total, the transmission results in seven **1**s. Under the proposed coding scheme, one of the learned centers is 01010101, and is recorded at both the transmitter and the receiver. Since 00010101 is the result of XORing the center 01010101 with the residual 01000000, and 01011101 is the result of XORing the center 01010101 with the residual 00001000, the two residuals are transmitted through the interconnects with the center bit set to **1**. The total number of transmitted **1**s is four, which is 43% smaller than that of DBI coding. This simple example suggests that the proposed coding scheme can reduce data movement energy by reducing the number of transmitted **1**s by learning accurate data cluster centers.

IV. Online Data Clustering and Encoding

Figure 5 illustrates an example system using the proposed data encoding scheme. The coding technique is applied to both the DRAM interface and the LLC. Data communication between the memory controller and each DRAM chip is performed through a codec. The cache controller and each cache bank are also equipped with the proposed codec for transmitting data on the main data H-tree. The proposed coding scheme is applied to only the main data H-tree—which is a major contributor to the overall LLC energy—to limit the coding overhead.

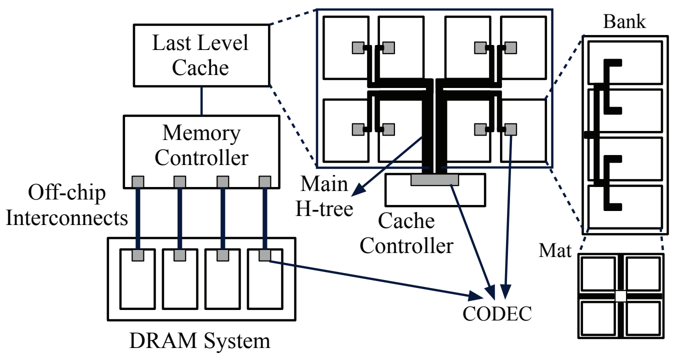


Fig. 5. An example system using the proposed coding technique.

A. Encoding and Decoding

Accurately decoding the encoded data requires the learned cluster centers to be kept synchronized at the transmitter and receiver. Given the current set of cluster centers, the encoder encodes each new data block as the XOR between the nearest cluster center (*i.e.*, the center with the smallest Hamming

[1]Recall that DBI coding can be used to minimize either the number of 1s or 0s. In DDR4, it is used to minimize the number of zeroes, whereas in this example, it is used to minimize the number number of 1s.

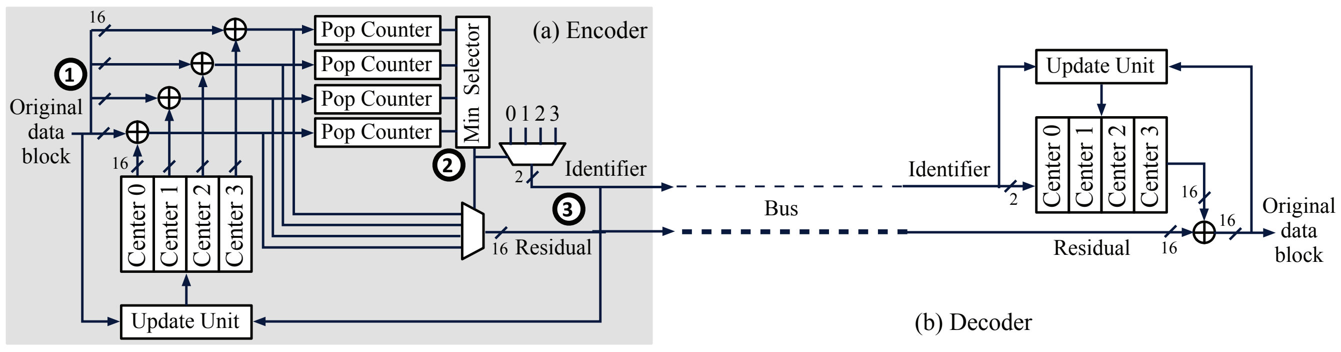


Fig. 4. Illustrative example of the proposed codec.

distance to the block) and the residual. The residual, which typically contains a small number of **1**s, is transmitted with the corresponding center identifier. At the receiver, the decoder recovers the data by XORing the specified cluster center with the transmitted residual.

Figure 4 (a) shows an encoder with four centers as an example. In this example, to keep the overhead due to extra pins the same as that of DBI coding, each cache line is chopped into 16-bit data blocks before encoding. The encoder needs to find the nearest center for each transmitted data block to reduce the energy cost of data movement. To fulfill this need, a data block is first XORed with each stored center to generate a candidate residual (step ① in Figure 4). The Hamming weights (number of **1**s) of the generated residual blocks are compared to determine the residual with the lowest Hamming weight (step ② in Figure 4). The center used to generate the selected residual block thus has the shortest Hamming distance to the original data block. Its identifier and the residual are concatenated to form the encoded data block, and are transmitted to the receiver (step ③ in Figure 4).

Figure 4 (b) illustrates how the decoder recovers the original data at the receiver side. The center identifier in the encoded data is used as the index to retrieve the center used for encoding. The original data is recovered by XORing the center and the residual in the encoded data.

B. Learning the Cluster Centers

Learning proper cluster centers that can capture representative data patterns in the transmitted data set is vital to the proposed coding scheme. An online clustering method inspired by the offline k-majority algorithm is proposed to solve the problem. The original offline k-majority algorithm performs clustering by iteratively assigning each data block to the nearest cluster center, and relocating each center based on the data blocks that belong to the corresponding cluster. Although it is perceivable that k-majority could be applied to the entire memory system once (or periodically) to determine the cluster centers, the required transfer of data to perform the clustering itself could become a significant source of energy dissipation. Instead, the proposed online data clustering scheme dynamically assigns each transmitted data block to the nearest cluster center, and adjusts the center based on this newly observed data. The assignment step is automatically performed by data encoding since the encoder already calculates the Hamming distance to the nearest cluster center. The adjustment step adopts the majority vote method used by the offline k-majority algorithm: each bit value of the new center is determined by the majority of the corresponding bit values within all of the data blocks that belong to that cluster. Hardware counters are required to keep the majority vote results of old data blocks that have been assigned to each cluster. At the receiver side, the corresponding cluster center is adjusted by following an identical set of steps to keep the centers at the transmitter and receiver synchronized.

Choosing the number of centers. In the proposed coding scheme, the number of cluster centers is a parameter defined at design time. A larger number of centers provides the opportunity to learn and make fine distinctions among a greater number of data patterns. However, to keep the extra coding bit overhead the same as that of DBI coding, the coding data block size also needs to be increased, which may reduce the efficacy of the clustering. In addition, more centers and larger coding blocks lead to high coding overheads, which may outweigh the benefits achieved by data encoding. Based on the results from a sensitivity study (Section VI-E1), we choose sixteen centers, which provides the best energy efficiency for the evaluated systems.

Initializing the centers. Similar to the original k-majority clustering algorithm, the cluster centers are randomly initialized in the proposed coding scheme. The centers can also be initialized based on prior knowledge of the transferred data to learn the cluster centers faster. Here we use random initialization to keep the coding scheme simple. An analysis on the impact of center initialization can be found in Section VI-E4.

Updating the centers. Instead of performing the data clustering offline, the cluster centers in the proposed coding scheme are dynamically updated based on the transmitted data to adapt to the changes in data patterns during the execution. The proposed online center updating method is designed based on the update rule of the offline k-majority algorithm, as shown in Figure 6. Each bit of every cluster center is associated with a saturating counter, which maintains sufficient statistics to compute the majority vote. When a transmitted data block is assigned to the nearest cluster center for encoding, it also updates the counters that belong to the cluster center in the background. Depending on whether the corresponding bit value within the data block is a **0** or a **1**, every counter is respectively decremented or incremented by 1. The most significant bit of every counter is used to represent the result of the majority vote.

The saturating counters provide the ability to "forget" stale associations with old data, because a number of updates in the opposite direction decreases the weights of old data in each cluster, helping the cluster center to "drift" as the data changes. Therefore, the size of the saturating counter affects the efficiency of the proposed coding scheme. On the one hand, if the size is too large, the old information may require a long time to be forgotten, which may prolong the time needed to find the proper centers for more recent data. On the other hand, if the size is too small, it might be hard for the learning procedure to capture important patterns that may have appeared in the distant past. In addition, large counters introduce higher coding overheads, which may offset the energy reduction achieved by the data encoding. A sensitivity analysis on the size of the counter is presented in Section VI-E2. For the configuration with sixteen centers, the proposed coding scheme that uses 4-bit counters achieves the highest energy improvement.

The two update units located at the transmitter and the receiver provide a simple and effective way for synchronization. If there is a transmission error, the centers at the transmitter are updated first. The transmission error is then corrected — *e.g.*, by retransmission or ECC checks, depending on the error correction mechanism employed by the system. After this, the centers at the receiver side are updated based on the correct data value. As long as the correct data can be recovered, the centers can be correctly synchronized by the same learning procedure. [2]

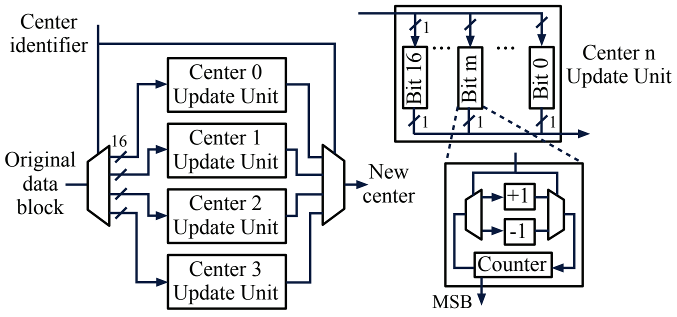


Fig. 6. Illustrative example of the center update unit.

Reducing the coding overheads. The proposed coding scheme applies sampling on top of the basic center update procedure to reduce the energy overheads. Fixed interval sampling is employed to keep the learning process simple. Rather than collecting information and performing updates based on all of the transferred data, the counters and the cluster centers are updated every N memory requests. Due to the sampling, the coding scheme loses some data value information in the center learning process. The sampling interval (N) is chosen to be sixty-four, which limits the lost opportunity in energy savings to 5% of an ideal solution that samples every memory access (Section VI-E3).

C. Transition Signaling for Unterminated Interconnects

For the unterminated LPDDR3 interface and for on-chip interconnects, the energy consumption of data movement is proportional to the number of transitions on the wires instead of the number of transmitted **0**s or **1**s. However, transition signaling [36] can be employed to make the energy consumption of each transmitted data context-independent. Different from *level signaling*, which uses high and low voltage levels to represent logical **1**s and **0**s, *transition signaling* uses a voltage transition on the interconnects and the lack thereof to represent a logical **1** and **0**. Under transition signaling, minimizing the number of transitions on the wires reduces to the problem of minimizing the number of **1**s in the transmitted data. With transition signaling, energy efficient coding techniques designed for unterminated interconnects can be readily applied to save data movement energy by reducing the number of **1**s in each transmitted data block. This makes the proposed coding scheme applicable to an unterminated LPDDR3 interface, and to the main data H-tree within the LLC.

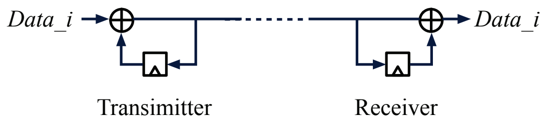


Fig. 7. Transition signaling. $Data\_i$ represents the original data bit.

Transition signaling requires an extra XOR gate and a register at both the transmitter and the receiver, as shown in Figure 7. Each transmitted bit of data is encoded as the XOR of the current bit value and the previous value transmitted on the interconnect. At the decoder, the original bit of data is recovered by XORing the received bit value and the previously transmitted value.

V. Experimental Setup

Analyzing the performance, energy, and area of the proposed clustering based data encoding scheme requires both architectural and circuit-level design and evaluation.

A. Architecture

We use a heavily-modified version of the SESC simulator [37] with a cycle accurate DRAM timing model to evaluate the baselines and the proposed coding scheme. The efficacy of the proposed coding scheme depends on the contribution of the memory interface and the main data H-tree of the LLC to the system energy consumption. Both a DDR4 based server system and an LPDDR3 based mobile system are evaluated (configuration parameters are shown in Table I). McPAT 1.0 [38] is used to estimate the processor energy at the 22nm technology node. CACTI 6.5 [39] is used to evaluate the energy and latency of caches. [3] The energy consumption of DDR4 and LPDDR3 systems is calculated following the

[2]The overhead of maintaining the two update units is taken into account in the evaluation.

[3]LSTP devices are used to model the last level caches [40], [41], [42], [43], [44].

TABLE I
EVALUATED SYSTEM CONFIGURATIONS.

	Mobile System [32], [33]	Microserver System [34]
Core	8 out-of-order cores, 1.6GHz, single thread per core fetch/issue/commit width 3/3/3	8 in-order cores, 3.2GHz, 4 threads per core fetch/issue/commit width 4/4/4
IL1 cache	32KB, direct-mapped, 64B line, hit/miss delay 1/1	
DL1 cache	32KB, 4-way, 64B line, write back, hit/miss delay 2/2	
Coherence	Snoopy bus with MESI protocol	
L2 cache	2MB, 8-way, 8 banks, 64B line, hit/miss delay 11/2	8MB, 16-way, 8 banks, 64B line, hit/miss delay 19/4
Memory controller	FR-FCFS scheduling [35], open-page policy, page-interleaved address mapping	
DRAM	LPDDR3-1600 [13] channels/ranks/banks = 2/2/8, page size 4KB, tCL/tWL/tRCD/tRP/tRAS/tRC/tFAW/tREFI/tRFC/BL = 12/6/15/16/34/50/40/3120/168/8 DRAM cycles	DDR4-3200 [7] channels/ranks/banks = 2/2/8, page size 8KB, tCL/tWL/tRCD/tRP/tRAS/tRC/tFAW/tREFI/tRFC/BL = 20/16/20/20/52/72/48/12480/416/8 DRAM cycles

methodology described in the respective power calculators provided by Micron [45], [14], with parameters taken from prior work [7], [13]. The three coding schemes discussed in Section II are implemented and evaluated as baselines for comparison: 1) the conventional DBI coding [6], [7]; 2) the recently proposed CAFO coding [8] (two dimensional bus invert coding); and 3) coding based on the k recent values [9]. [4] All of the data encoding techniques, including the proposed one, keep the extra pin overhead the same as that required by the conventional DBI coding.

B. Synthesis

The codec logic of different evaluated data encoding schemes and the center updating unit used by the proposed data encoding technique are designed and verified in Verilog RTL, and synthesized with the FreePDK 45nm library [46] using the Synopsys Design Compiler [47]. The area, timing and power numbers are scaled from 45nm to 22nm DRAM process technology using the methodology and scaling parameters reported in prior work [40], [48], [49], [50], [51]. [5] CACTI 6.5 [39] and FabMem [48] are used to model the SRAM tables that store cluster centers.

C. Applications

We evaluate a mix of fourteen applications that are readily portable to our simulator with different memory intensities, as shown in Table II. We use SimPoint [52] to find a representative 1 billion instruction region from each SPEC2006 benchmark to reduce the simulation time.

VI. EVALUATION

In this section, we compare the energy and performance of the proposed architecture to the baseline systems. We also analyze the latency, area, and power overheads of the components in the proposed data encoding technique.

[4] XOR-based coding [27] is a special case of k recent value coding, with k equal to 1. The k recent value coding is evaluated in the paper since it is a stronger baseline.

[5] ITRS LSTP devices are used for the codec on the DRAM side to model the impact of implementing logic circuits using a memory process [40]. For simplicity, the same synthesis results (using LSTP devices) are conservatively used for the processor side as well.

TABLE II
APPLICATIONS AND DATA SETS.

Benchmarks	Suite	Input
SCALPARC	NuMineBench [53]	A32-D125K
GUPS	HPCC [54]	2^{25} table, 1048576 updates
LINEAR	Phoenix [55]	100MB file
FFT	SPLASH-2 [56]	2^{20} complex data points
OCEAN	SPLASH-2	514×514 ocean
CG	NAS OpenMP [57]	Class A
MG	NAS OpenMP	Class A
ART	SPEC OpenMP [58]	MinneSpec-Large
EQUAKE	SPEC OpenMP	MinneSpec-Large
SWIM	SPEC OpenMP	MinneSpec-Large
LBM	SPEC 2006 [59]	Reference
MCF	SPEC 2006	Reference
MILC	SPEC 2006	Reference
SOPLEX	SPEC 2006	Reference

A. Hardware Overheads

Table III lists the power, area, and latency overheads of the evaluated data encoding techniques. The data shown for DBI coding is based on an 8-bit coding block size. All of the other coding techniques exhibit the same extra bit overhead (1 extra bit per every 8 bits) as DBI coding. The data shown for CAFO coding is based on a configuration with four iterations for encoding [6]. The data shown for the proposed coding scheme is based on a configuration with a 32-bit coding block size and sixteen cluster centers. Since recent value coding also uses a 32-bit coding block size and keeps the most recent sixteen data blocks, it has the same encoder and decoder overheads as the proposed solution. The center update unit of the proposed coding technique works in the background; thus, its latency is not on a performance critical path. Compared to the long DRAM access latency, the codec latency is short; notably, the codec does not constitute a bandwidth bottleneck. All of these overheads are taken into account in the subsequent performance and energy evaluations.

B. Performance

The system performance with different energy efficient coding techniques are evaluated. Figure 8 shows the execution time normalized to the DBI coding baseline on the server systems. For the server systems, CAFO, recent value coding, and the proposed coding scheme respectively degrade the average performance by 1.2%, 0.4%, and 0.4%. The applications `art`, `cg`, `mg`, `swim`, `mcf`, and `milc` are more sensitive to the extra

[6] CAFO coding in this configuration achieves the same average energy reduction as the original CAFO without the extra latency penalty.

TABLE III
POWER, AREA AND LATENCY OVERHEADS OF DIFFERENT DATA ENCODING SCHEMES AT 22NM.

	Power (mW)	Area (mm^2)	Latency (ns)
Proposed Encoder	2.8	0.004	0.9
Proposed Decoder	0.81	0.0005	0.18
Center Update Unit	4.4	0.002	0.38
DBI Encoder	0.19	0.00006	0.18
DBI Decoder	0.35	0.00003	0.06
CAFO Encoder	0.64	0.0001	1.76
CAFO Decoder	0.88	0.0002	0.04

latency added by the encoding and decoding processes. Most of the sensitive applications are memory intensive.

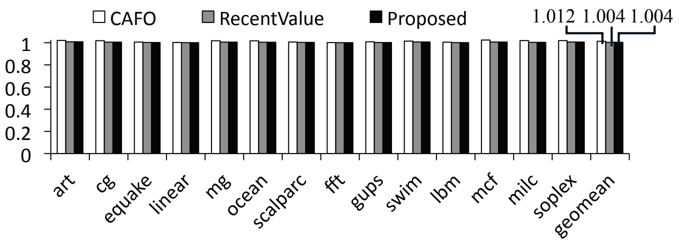


Fig. 8. Execution time normalized to the DBI coding baseline on the evaluated server systems.

The performance results achieved on the mobile systems are shown in Figure 9. As compared to the server systems, the performance degradation slightly increases due to a smaller number of hardware threads and a smaller LLC. On average, CAFO, recent value coding, and the proposed coding scheme respectively degrade the performance by 1.3%, 0.5%, and 0.5%. The applications `art`, `mcf`, and `milc` suffer from the largest performance degradations (1.1%, 0.9%, and 0.9%) when using the proposed coding scheme. However, the system using the proposed coding scheme still outperforms the system using CAFO coding, which respectively degrades the performance by 3.9%, 3.6%, and 1.8% for these three applications.

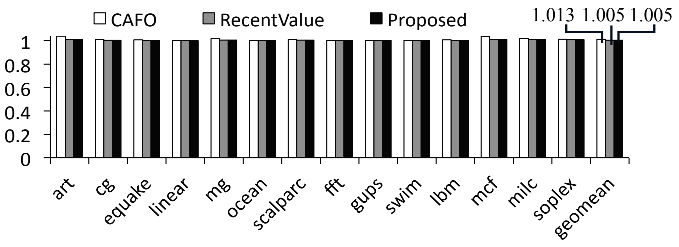


Fig. 9. Execution time normalized to the DBI coding baseline on the evaluated mobile systems.

C. Energy Impact on Server Systems

Figure 10 illustrates the number of transmitted **0**s under three different coding schemes in the DDR4 DRAM system. The numbers are normalized to the baseline with DBI coding. On average, the proposed coding scheme respectively reduces 40%, 30%, and 20% of the number of transmitted **0**s as compared to DBI, CAFO and recent value coding techniques. For applications `art` and `equake`, the difference between the recent value coding scheme and the proposed approach is small (about 10%). These applications have relatively good temporal data value locality, which makes the recent value coding competitive. In addition, the sampling used in the proposed coding scheme has nontrivial effects on these applications (discussed in Section VI-E3), preventing the proposed coding scheme from eliminating a larger number of transmitted **0**s.

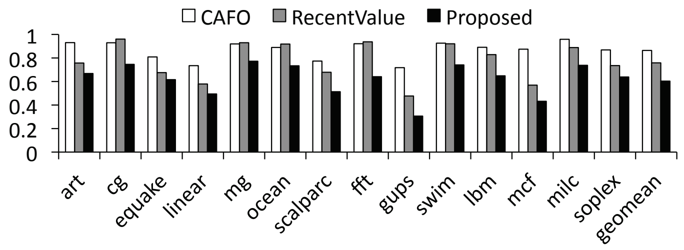


Fig. 10. The number of transmitted **0**s normalized to that of the DBI coding.

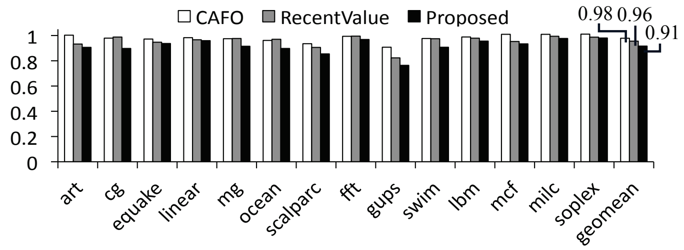


Fig. 11. Energy consumption of the DDR4 DRAM system normalized to that with DBI coding.

The energy consumption of the DDR4 system is shown in Figure 11. CAFO, recent value, and the proposed coding schemes reduce the DDR4 system energy by 2%, 4%, and 9% as compared to the system with DBI coding. Figure 12 shows the energy breakdown of the DDR4 system. Since the DRAM core energy, especially the background energy, contributes substantially to the overall DDR4 energy, the IO energy savings achieved by energy efficient coding schemes are not as high as the percentage of eliminated **0**s. The coding overheads added to the DRAM system is too small to be visible in the figure. The overheads of the proposed coding scheme account for 0.15% of the overall DDR4 system energy.

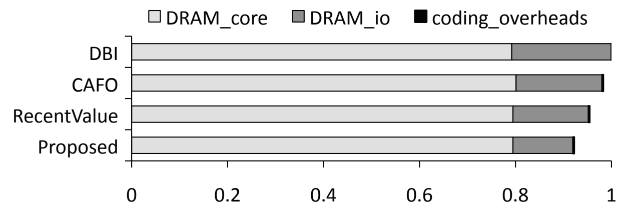


Fig. 12. Energy breakdown of the DDR4 DRAM system.

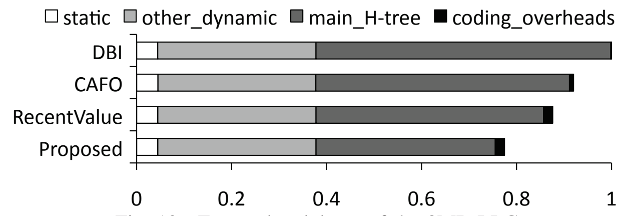


Fig. 13. Energy breakdown of the 8MB LLC.

The energy efficient coding techniques are also applied to the main data H-tree in the LLC. The proposed coding scheme respectively reduces 40%, 30%, and 21% of the transitions on the interconnects as compared to DBI, CAFO, and recent value coding. Since the main data H-tree is a big contributor to the overall LLC energy, as shown in Figure 13, the coding techniques achieve significant energy savings on the LLC. As illustrated in Figure 14, CAFO, recent value and the proposed coding schemes respectively reduce the LLC energy by 7%, 11%, and 21% as compared to the DBI coding baseline. Since the LLC is more frequently accessed than the DRAM system, and the overall energy is smaller than that of the DRAM system, the proportion of coding overheads in the overall LLC energy become larger as compared to those in the overall DDR4 energy. CAFO, recent value and the proposed coding scheme add 0.8%, 2%, and 2% energy overheads to the overall LLC energy (Figure 13).

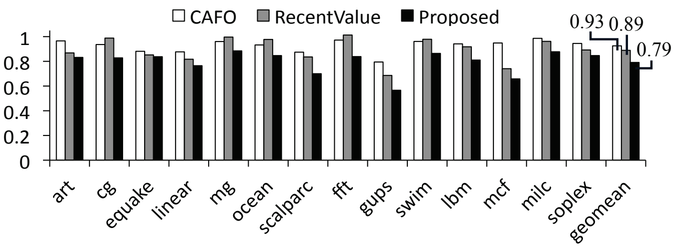


Fig. 14. Energy consumption of the 8MB LLC normalized to that with DBI coding.

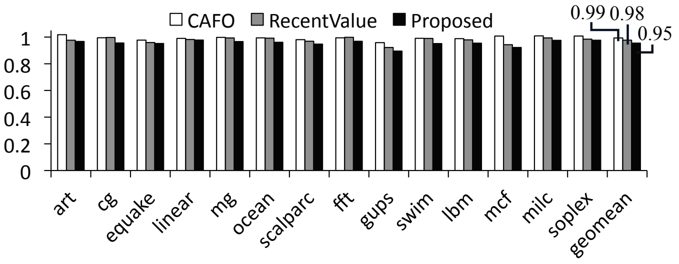


Fig. 15. Energy consumption of the server system normalized to that with DBI coding.

Figure 15 shows the energy consumption of the evaluated server systems. As compared to DBI coding, CAFO, recent value and the proposed coding schemes respectively save 1%, 2%, and 5% of the system energy on average. Besides the reduction in the number of transmitted **0**s and transitions, the energy savings achieved by the proposed coding scheme also depend on the proportion of the DRAM and LLC energy in the overall system energy. For memory-nonintensive applications, such as `linear`, the DRAM and LLC energy are not significant. For such applications, the overall system energy reduction is modest although the proposed coding scheme eliminates more than half of the number of transmitted **0**s and transitions. For memory-intensive applications, such as `gups` and `mcf`, the proposed coding scheme can achieve high system-wide energy savings (11% and 8%).

D. Energy Impact on Mobile Systems

When the proposed coding scheme is applied to a LPDDR3 DRAM system, it respectively reduces the number of transitions on the interface by 39%, 30%, and 21% as compared to

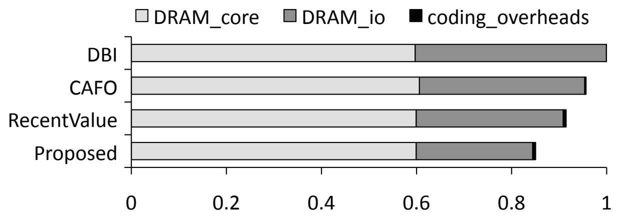


Fig. 16. Energy breakdown of the LPDDR3 DRAM system.

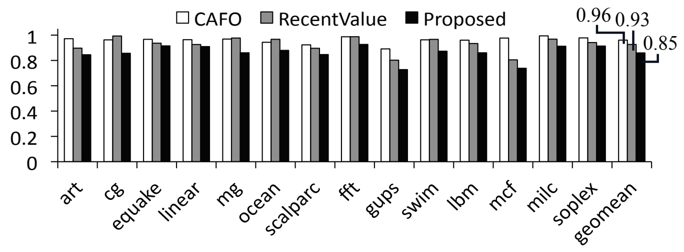


Fig. 17. Energy consumption of the LPDDR3 DRAM system normalized to that with DBI coding.

DBI, CAFO, and recent value coding. In an LPDDR3 system, DRAM IO is a big contributor to the overall DRAM system energy (Figure 16). As a result, coding techniques can achieve much higher energy savings than they do in a DDR4 system. Figure 17 shows that on average, CAFO, recent value, and the proposed coding scheme respectively reduce the LPDDR3 system energy by 4%, 7%, and 15%. The coding overheads remain low in the LPDDR3 DRAM system: 0.2%, 0.6%, and 0.6% for CAFO, recent value and the proposed coding scheme, respectively.

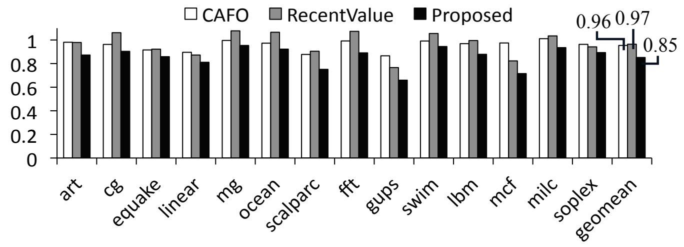


Fig. 18. Energy consumption of the 2MB LLC normalized to that with DBI coding.

In the mobile system, the proposed coding scheme reduces the number of transitions on the main data H-tree of the LLC by 41%, 31%, and 23% as compared to DBI, CAFO, and recent value coding schemes. Although the number of transitions is similar to those in the server system, the energy savings on the LLC are smaller in the mobile system. As shown in Figure 18, as compared to the DBI coding baseline, the average LLC energy reductions achieved by CAFO, recent value, and the proposed coding schemes are 4%, 3%, and 15%, respectively. Because of the smaller size and lower energy consumption of the LLC in the mobile system, the coding overheads become nontrivial and offset parts of the benefits achieved by data encoding. The energy breakdown in Figure 19 shows that the coding overheads of CAFO, recent value, and the proposed coding schemes are respectively responsible for 4%, 10%, and 10% of the overall LLC energy in the mobile system.

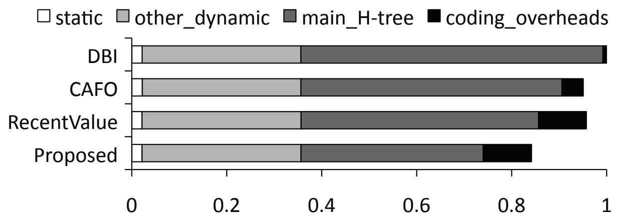


Fig. 19. Energy breakdown of the 2MB LLC.

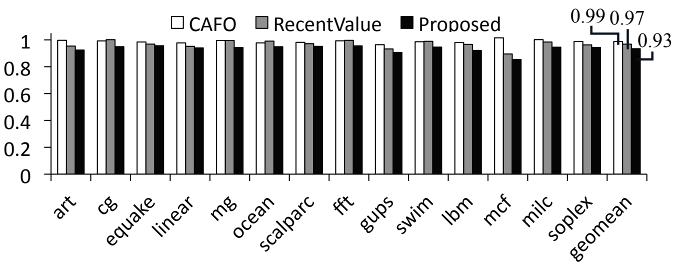


Fig. 20. Energy consumption of the mobile system normalized to that with DBI coding.

Due to the large energy savings on LPDDR3 and the high energy efficiency of mobile cores, the proposed coding scheme achieves higher system-wide energy reduction in the mobile system as compared to that of the server system. Figure 20 shows that as compared to DBI coding, the proposed coding scheme reduces the overall mobile system energy by 7%, which is higher than both CAFO (1%) and recent value coding (3%). Some memory-intensive applications, such as `gups` and `mcf`, achieve over 10% system-wide energy savings using the proposed coding scheme.

E. Analysis and Discussion

We performed sensitivity studies and analyses on LLCs, LPDDR3, and DDR4 DRAM systems in both mobile and server systems. Only the LPDDR3 results are shown here for brevity; DDR4 and LLCs demonstrate similar characteristics.

1) Number of Centers: In the proposed coding scheme, the number of centers affects the magnitude of the achieved energy reductions. Figure 21 illustrates the average number of transitions under configurations with different numbers of centers. Since the performance of the recent value coding scheme also depends on the number of stored data values, its results are also shown in the figure for comparison. The configuration with sixteen centers achieves the best result for both recent value and the proposed coding scheme. As compared to recent value coding, the proposed coding technique is less sensitive to the number of centers. Even with only two centers, it can still achieve a 31% reduction in the number of transitions as compared to DBI coding.

Figure 22 shows the average LPDDR3 system energy under coding schemes with different configurations. The configuration with sixty-four centers suffers from a significant coding overhead; consequently, this configuration is not applicable in practice and is not shown in the figure. Since the coding overheads of the remaining configurations are much smaller than

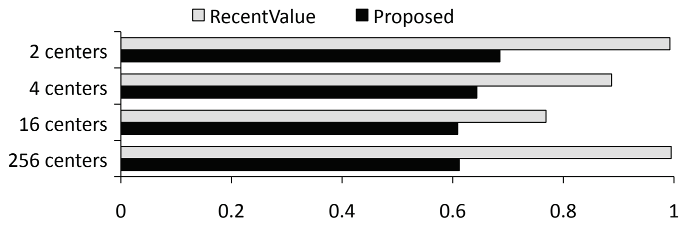


Fig. 21. Sensitivity of the number of transitions to the number of centers. The results are normalized to those achieved by DBI coding.

the benefits achieved by reducing the number of transitions, the LPDDR3 system energy under different configurations closely follows the number of transitions. The proposed coding scheme consistently achieves lower energy consumption and is less sensitive to the number of centers as compared to recent value coding.

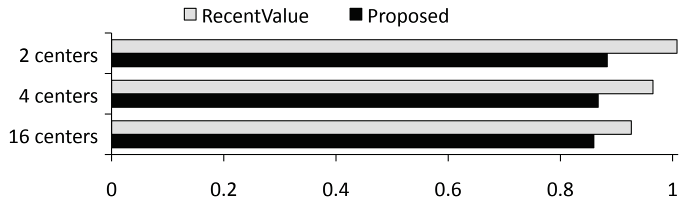


Fig. 22. Sensitivity of LPDDR3 system energy to the number of centers. The results are normalized to those achieved by DBI coding.

Since the configuration of the proposed coding scheme with two centers can nevertheless reduce the number of transitions significantly, it can be applied to small subsystems in which the coding overheads are nontrivial.

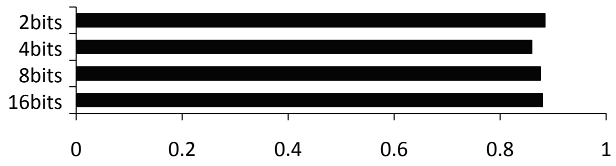


Fig. 23. Sensitivity of LPDDR3 system energy to the size of the counter used for center update. The results are normalized to those achieved by DBI coding.

2) Counter Size: The proposed coding scheme uses saturating up/down counters to dynamically update the centers. Figure 23 illustrates the impact of counter size on the LPDDR3 system energy. On average, the 4-bit counter configuration performs better than the other three configurations. The main reason is that the 4-bit counter configuration can find a better balance between adapting to recent data values and forgetting the past information, which allows it to reduce a larger number of transitions on the interconnects.

The best counter size is affected by the number of centers used in the proposed coding scheme. With a decreasing number of centers, the best configuration tends to use larger counters. On average, the 4-bit counter configuration performs best for the proposed coding schemes with sixteen and four centers; the 8-bit counters perform best with two centers.

3) Impact of Sampling: Figure 24 illustrates the impact of sampling on the proposed coding scheme. On average, the

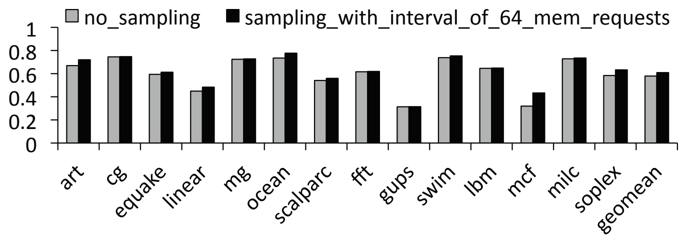


Fig. 24. The impact of sampling on the number of transitions for the LPDDR3 interface. The results are normalized to those achieved by DBI coding.

proposed coding scheme with a sampling interval of sixty-four memory requests increases the number of transitions on the interconnects by 5% as compared to the proposed coding scheme without sampling. Most of the applications are not very sensitive to sampling, which allows the proposed coding scheme to reduce center update overheads without significantly sacrificing the energy reduction benefits achieved by the data encoding.

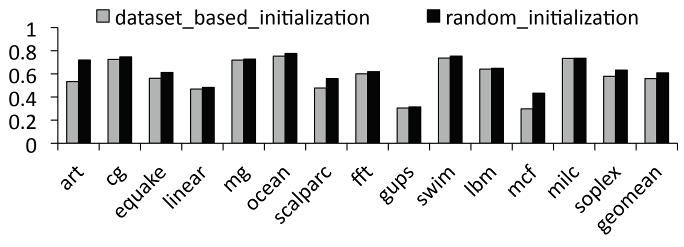


Fig. 25. The impact of center initialization on the number of transitions for the LPDDR3 interface. The results are normalized to those achieved by DBI coding.

4) Impact of Initial Center Selection: The proposed coding scheme can achieve greater energy reductions if it is given some guidance at the center initialization step. If the centers can be initialized closer to the representative data patterns, the learning process will become shorter and the learned centers will be more accurate, which leads to more energy reduction. The information used to guide center initialization can be achieved based on information collected from previous executions of an application, or some knowledge about the data sets stored in the memory that are going to be accessed by an application.

Figure 25 shows the comparison between the proposed coding scheme with random and guided center initialization. The evaluated data encoding scheme with guided center initialization assumes the data sets used by the applications are known ahead of time based on programmers' knowledge or previous executions. A single iteration of offline k-majority analysis is performed on the data sets to generate the offline cluster centers. These centers are used for initialization under the assumption that the data transmitted over the interconnects share some patterns and characteristics with the data stored in the memory. On average, the new center initialization method achieves a 9% higher reduction on the number of transitions as compared to the random center initialization method. The applications `art` and `mcf` achieve significantly higher savings (35% and 46%, respectively), as their data sets that are stored in the memory exhibit similar data patterns as those in their transmitted data sets. For applications that are going to be executed multiple times with a large amount of transferred data, the guided center initialization might be a better option. Although it requires extra overheads to collect information on the data sets, the energy savings may still be higher than that achieved by the random center initialization method due to more accurate cluster centers.

5) Impact of Multiprogrammed Workloads: A set of multiprogrammed workloads was generated by choosing pairs of applications from the evaluated SPEC2006 benchmarks. Weighted speedup [60] is used as the metric to evaluate the performance.

Similarly to the other evaluated applications, the performance of the multiprogrammed workloads is slightly degraded due to the latency added by the encoding and decoding processes. For the mobile systems, CAFO, recent value coding, and the proposed coding scheme respectively degrade the average performance by 0.9%, 0.1%, and 0.1% as compared to the system using DBI coding.

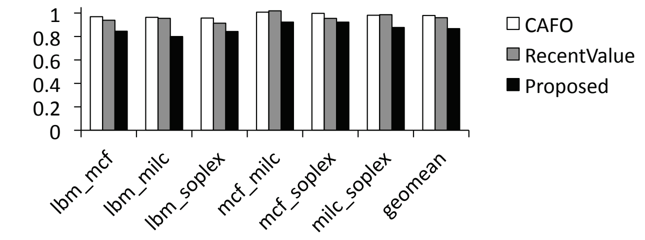


Fig. 26. The impact of multiprogrammed workloads on LPDDR3 system energy. The results are normalized to those achieved by DBI coding.

The mixture of data patterns from different applications increases the difficulty of the learning process. Figure 26 shows that on average, CAFO, recent value, and the proposed coding scheme respectively reduce the LPDDR3 system energy by 2%, 4%, and 13%. The coding schemes that only rely on temporal locality, such as the evaluated recent value coding baseline, can only achieve limited data movement energy reduction. In contrast, the proposed coding scheme not only exploits temporal locality, but also can find patterns that appear in the distant past by keeping statistical information on each cluster. When running multiprogrammed workloads, the different patterns exhibited by different workloads are still captured and form different clusters during the online learning process. Consequently, the proposed coding scheme can still achieve significant energy savings for multiprogrammed workloads.

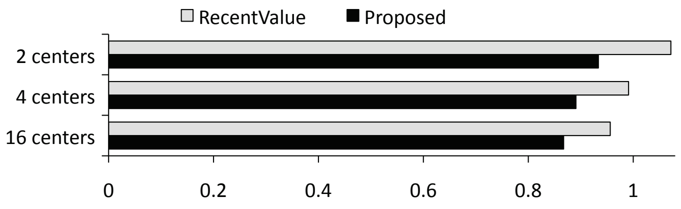


Fig. 27. LPDDR3 system energy of multiprogrammed workloads with different numbers of centers. The results are normalized to those achieved by DBI coding.

Figure 27 shows the average LPDDR3 system energy under coding schemes with different configurations for multiprogrammed workloads. With a small number of cluster centers, recent value coding fails to reduce the data movement energy. However, the proposed coding scheme can still achieve moderate energy savings. As compared to other evaluated applications (Figure 22), multiprogrammed workloads are more sensitive to the number of cluster centers, and the optimal number of clusters depends on the mixture of applications.

VII. Conclusions

The asymmetric energy consumption of data movement provides the opportunity to save energy by reducing the number of **1**s in the transmitted data. Leveraging this feature, we propose a novel data encoding scheme based on online data clustering. By dynamically grouping similar data blocks into clusters, and encoding each data block as the XOR between the nearest cluster center and a sparse residual, the proposed coding technique can significantly reduce data movement energy. We conclude that the proposed online data clustering and encoding approach holds the potential to improve the energy efficiency of data movement in future computer systems.

Acknowledgment

The authors would like to thank Yanwei Song and anonymous reviewers for useful feedback. This work was supported in part by NSF grant CCF-1533762.

References

[1] B. Dally, "Power, programmability, and granularity: The challenges of exascale computing," in *Parallel Distributed Processing Symposium (IPDPS), 2011 IEEE International*, pp. 878–878, May 2011.

[2] G. Kestor, R. Gioiosa, D. J. Kerbyson, and A. Hoisie, "Quantifying the energy cost of data movement in scientific applications," in *Workload Characterization (IISWC), 2013 IEEE International Symposium on*, pp. 56–65, Sept 2013.

[3] D. Pandiyan and C. J. Wu, "Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms," in *Workload Characterization (IISWC), 2014 IEEE International Symposium on*, pp. 171–180, Oct 2014.

[4] W. J. Dally, "Challenges for future computing systems," Jan 2015.

[5] "Top ten exascale research challenges, DOE advanced scientific computing advisory committee (ASCAC) subcommittee report," tech. rep., U.S. Department of Energy, February 2014.

[6] K. Sohn, T. Na, I. Song, Y. Shim, W. Bae, S. Kang, D. Lee, H. Jung, S. Hyun, H. Jeoung, K.-W. Lee, J.-S. Park, J. Lee, B. Lee, I. Jun, J. Park, J. Park, H. Choi, S. Kim, H. Chung, Y. Choi, D.-H. Jung, B. Kim, J.-H. Choi, S.-J. Jang, C.-W. Kim, J.-B. Lee, and J. S. Choi, "A 1.2 v 30 nm 3.2 gb/s/pin 4 gb DDR4 SDRAM with dual-error detection and PVT-tolerant data-fetch scheme," *IEEE Journal of Solid-State Circuits*, vol. 48, pp. 168–177, Jan 2013.

[7] Micron Technology, "4Gb DDR4 SDRAM," 2014.

[8] R. Maddah, S. M. Seyedzadeh, and R. Melhem, "CAFO: Cost aware flip optimization for asymmetric memories," in *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*, pp. 320–330, 2015.

[9] S. Komatsu, M. Ikeda, and K. Asada, "Low power chip interface based on bus data encoding with adaptive code-book method," in *VLSI, 1999. Proceedings. Ninth Great Lakes Symposium on*, pp. 368–371, 1999.

[10] J. Yang, R. Gupta, and C. Zhang, "Frequent value encoding for low power data buses," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 9, pp. 354–384, July 2004.

[11] Micron Technology, "TN-40-03: DDR4 networking design guide," 2014.

[12] K. Song, S. Lee, D. Kim, Y. Shim, S. Park, B. Ko, D. Hong, Y. Joo, W. Lee, Y. Cho, W. Shin, J. Yun, H. Lee, J. Lee, E. Lee, N. Jang, J. Yang, H. k. Jung, J. Cho, H. Kim, and J. Kim, "A 1.1 v 2y-nm 4.35 gb/s/pin 8 gb LPDDR4 mobile device with bandwidth improvement techniques," *IEEE Journal of Solid-State Circuits*, vol. 50, pp. 1945–1959, Aug 2015.

[13] Micron Technology, "16Gb: 216-ball, dual-channel mobile LPDDR3 SDRAM," 2014.

[14] Micron Technology, "LPDDR3 power calculator," 2015.

[15] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power I/O," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 3, pp. 49–58, March 1995.

[16] JEDEC, "High Bandwidth Memory (HBM) DRAM," 2015.

[17] JEDEC, "Wide I/O 2 (WideIO2)," 2014.

[18] Hybrid Memory Cube Consortium, "Hybrid Memory Cube specification 2.1," 2014.

[19] J. Jeddeloh and B. Keeth, "Hybrid memory cube new DRAM architecture increases density and performance," in *VLSI Technology (VLSIT), 2012 Symposium on*, pp. 87–88, June 2012.

[20] A. N. Udipi, N. Muralimanohar, R. Balasubramonian, A. Davis, and N. P. Jouppi, "Combining memory and a controller with photonics through 3D-stacking to enable scalable and energy-efficient systems," in *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ISCA '11, pp. 425–436, 2011.

[21] G. D. Micheli and L. Benini, *On-Chip Communication Architectures: System on Chip Interconnect*. Morgan Kaufmann Publishers Inc., 2008.

[22] W.-C. Cheng and M. Pedram, "Memory bus encoding for low power: a tutorial," in *Quality Electronic Design, 2001 International Symposium on*, pp. 199–204, 2001.

[23] M. R. Stan and W. P. Burleson, "Limited-weight codes for low-power I/O," in *International Workshop on low power design*, pp. 209–214, 1994.

[24] M. R. Stan and W. P. Burleson, "Coding a terminated bus for low power," in *VLSI, 1995. Proceedings., Fifth Great Lakes Symposium on*, pp. 70–73, 1995.

[25] Y. Song and E. Ipek, "More is less: Improving the energy efficiency of data movement via opportunistic use of sparse codes," in *Proceedings of the 48th International Symposium on Microarchitecture*, MICRO-48, pp. 242–254, 2015.

[26] D. C. Suresh, B. Agrawal, J. Yang, and W. A. Najjar, "Tunable and energy efficient bus encoding techniques," *IEEE Transactions on Computers*, vol. 58, pp. 1049–1062, Aug 2009.

[27] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj, "A coding framework for low-power address and data busses," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, pp. 212–221, June 1999.

[28] S. Cho and H. Lee, "Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pp. 347–357, 2009.

[29] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, 1967.

[30] S. P. Lloyd, "Least squares quantization in PCM," *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 129–137, 1982.

[31] C. Grana, D. Borghesani, M. Manfredi, and R. Cucchiara, "A fast approach for integrating ORB descriptors in the bag of words model," in *IS&T/SPIE Electronic Imaging*, pp. 866709–866709, 2013.

[32] A. L. Shimpi, "Qualcomm Snapdragon 805 Performance Preview," 2014. http://www.anandtech.com/show/8035/qualcomm-snapdragon-805-performance-preview.

[33] Qualcomm, "Snapdragon 808," 2016. https://www.qualcomm.com/products/snapdragon/processors/808.

[34] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: a 32-way multithreaded Sparc processor," *IEEE Micro*, vol. 25, pp. 21–29, March 2005.

[35] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens, "Memory access scheduling," in *Proceedings of the 27th Annual International Symposium on Computer Architecture*, ISCA '00, pp. 128–138, 2000.

[36] M. R. Stan and W. P. Burleson, "Low-power encodings for global communication in CMOS VLSI," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 5, pp. 444–455, Dec 1997.

[37] J. Renau, B. Fraguela, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, S. Sarangi, P. Sack, K. Strauss, and P. Montesinos, "SESC simulator," January 2005. http://sesc.sourceforge.net.

[38] 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 manycore architectures," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pp. 469–480, Dec 2009.

[39] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0," in *International Symposium on Microarchitecture*, (Chicago, IL), Dec. 2007.

[40] S. Thoziyoor, J. H. Ahn, M. Monchiero, J. B. Brockman, and N. P. Jouppi, "A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies," in *Computer Architecture, 2008. ISCA '08. 35th International Symposium on*, pp. 51–62, June 2008.

[41] M. N. Bojnordi and E. Ipek, "DESC: Energy-efficient data exchange using synchronized counters," in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-46, pp. 234–246, 2013.

[42] S. Sardashti and D. A. Wood, "Decoupled compressed cache: Exploiting spatial locality for energy-optimized compressed caching," in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-46, pp. 62–73, 2013.

[43] M.-T. Chang, P. Rosenfeld, S.-L. Lu, and B. Jacob, "Technology comparison for large last-level caches (l3cs): Low-leakage sram, low write-energy stt-ram, and refresh-optimized edram," in *Proceedings of the 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, HPCA '13, pp. 143–154, 2013.

[44] A. Arelakis and P. Stenstrom, "Sc2: A statistical compression cache scheme," in *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ISCA '14, pp. 145–156, 2014.

[45] Micron Technology, "DDR4 power calculator," 2014.

[46] J. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. Davis, P. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal, "FreePDK: An open-source variation-aware design kit," in *Microelectronic Systems Education, 2007. MSE '07. IEEE International Conference on*, pp. 173–174, June 2007.

[47] Synopsys, "Synopsys design compiler," 2010. http://www.synopsys.com/Tools/Implementation/RTLSynthesis/DesignCompiler/Pages/default.aspx.

[48] N. K. Choudhary, S. V. Wadhavkar, T. A. Shah, H. Mayukh, J. Gandhi, B. H. Dwiel, S. Navada, H. H. Najaf-abadi, and E. Rotenberg, "FabScalar: composing synthesizable RTL designs of arbitrary cores within a canonical superscalar template," in *Proceeding of the 38th annual international symposium on Computer architecture*, ISCA '11, pp. 11–22, 2011.

[49] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ISCA '11, pp. 365–376, 2011.

[50] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45nm design exploration," in *Quality Electronic Design, 2006. ISQED '06. 7th International Symposium on*, pp. 6 pp.–590, March 2006.

[51] ITRS, "International Technology Roadmap for Semiconductors: 2010 Update."

[52] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically characterizing large scale program behavior," in *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS X, pp. 45–57, 2002.

[53] J. Pisharath, Y. Liu, W. Liao, A. Choudhary, G. Memik, and J. Parhi, "NU-MineBench 2.0," tech. rep., Northwestern University, August 2005. Tech. Rep. CUCIS-2005-08-01.

[54] P. Luszczek, J. J. Dongarra, D. Koester, R. Rabenseifner, B. Lucas, J. Kepner, J. McCalpin, D. Bailey, and D. Takahashi, "Introduction to the HPC challenge benchmark suite," *Lawrence Berkeley National Laboratory*, 2005.

[55] R. M. Yoo, A. Romano, and C. Kozyrakis, "Phoenix rebirth: Scalable MapReduce on a large-scale shared-memory system," in *Proceedings of the 2009 IEEE International Symposium on Workload Characterization*, 2009.

[56] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," in *ISCA-22*, 1995.

[57] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga, "The NAS parallel benchmarks—summary and preliminary results," in *Proceedings of the 1991 ACM/IEEE Conference on Supercomputing*, Supercomputing '91, pp. 158–165, 1991.

[58] Standard Performance Evaluation Corporation, "SPEC OMP2001," 2001. https://www.spec.org/omp2001/.

[59] Standard Performance Evaluation Corporation, "SPEC CPU2006," 2006. https://www.spec.org/cpu2006/.

[60] A. Snavely and D. M. Tullsen, "Symbiotic jobscheduling for a simultaneous multithreaded processor," in *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS IX, pp. 234–244, 2000.