2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture

Energy Efficient Data Encoding in DRAM channels exploiting Data Value Similarity

Hoseok Seol Wongyu Shin Jaemin Jang Jungwhan Choi Jinwoong Suh Lee-Sup Kim
Korea Advanced Institute of Science and Technology
{seolhs,wgshin,jmjang,cjwdream,jinwoongsuh}@mvlsi.kaist.ac.kr leesup@kaist.ac.kr

***Abstract*—As DRAM data bandwidth increases, tremendous energy is dissipated in the DRAM data bus. To reduce the energy consumed in the data bus, DRAM interfaces with asymmetric termination, such as Pseudo Open Drain (POD) and Low Voltage Swing Terminated Logic (LVSTL), have been adopted in modern DRAMs. In interfaces using asymmetric termination, the amount of termination energy is proportional to the hamming weight of the data words. In this work, we propose Bitwise Difference Encoding (BD-Encoding), which decreases the hamming weight of data words, leading to a reduction in energy consumption in the modern DRAM data bus. Since smaller hamming weight of the data words also reduces switching activity, switching energy and power noise are also both reduced.**

BD-Encoding exploits the similarity in data words in the DRAM data bus. We observed that similar data words (i.e. data words whose hamming distance is small) are highly likely to be sent over at similar times. Based on this observation, BD-coder stores the data recently sent over in both the memory controller and DRAMs. Then, BD-coder transfers the bitwise difference between the current data and the most similar data. In an evaluation using SPEC 2006, BD-Encoding using 64 recent data reduced termination energy by 58.3% and switching energy by 45.3%. In addition, 55% of the LdI/dt noise was decreased with BD-Encoding.

Keywords-**DRAM, interface, energy, data locality, data similarity, POD, LVSTL, termination, switching activity**

I. Introduction

DRAM is one of the major components in modern computing systems. Over time, memory bandwidth has been increased to meet the needs of various applications. As a result, DRAM data traffic has grown, which causes enormous energy dissipation on the DRAM data bus [1], [2], [3]. Consequently, the DRAM interface has evolved to deal with the energy issues. For example, the interface voltage level has gone down to decrease the data bus energy. In addition, interface schemes such as Pseudo Open Drain (POD) or Low Voltage Swing Terminated Logic (LVSTL) have been employed.

Energy dissipated in the DRAM data bus consists of switching and termination energy. The switching energy refers to the energy needed to charge / discharge the capacitance of the data bus. The termination energy is dissipated in On Die Termination (ODT), the scheme used to mitigate the reflection of a signal in the data bus. Traditionally, the switching energy has been the key component for the energy dissipation in the DRAM data bus [4], [5]. However, as modern DRAMs have adopted termination schemes to increase data rates, termination energy has come to one of the dominant portions of energy dissipated in the DRAM data bus [2], [3].

To reduce the termination energy, modern DRAM interfaces have adopted asymmetric termination designs, which means that the termination resistor is connected to either the VDD or the VSS, instead of both the VDD and VSS symmetrically (POD: DDR4 [6] and GDDR4/5 [7], LVSTL: LPDDR4 [8]). The asymmetric termination consumes the termination energy only in half case of data transfer (i.e. bit-1 in LVSTL and bit-0 in POD). As a consequence, the termination energy is proportional to the numbers of bit-1 (for LVSTL) or bit-0 (for POD) in data words. By exploiting this characteristic, we propose a data encoding approach, called Bitwise Difference Encoding (BD-Encoding), to reduce the hamming weight of data words, and thereby reduce the termination energy[1].

Reducing the hamming weight of a data word also decreases the switching energy and power noise of a DRAM data bus. Both the switching energy and power noise are determined by the number of switching activity (i.e. data toggling) in the DRAM data bus. Since a data word are transferred in a serialized manner through the modern DRAM data bus[2], the number of switching activity is determined by the local difference in a data word. Reducing the hamming weight of a data word decreases the probability for local difference, and thereby reduces the switching energy and power noise.

For BD-Encoding, we exploit the data characteristic of DRAMs. Our key observation is that similar data words are sent over the DRAM data bus within a given time interval. In other words, there is a high probability that the data word to be transferred has low hamming distance from one of the data words recently sent over. Figure 1 shows that 90% data similarity exists among 64 recent data words. Specifically, 90% data similarity means that more than 90% of data bits (58-bits / 64-bits) are the same as those of the data word to be transferred. As shown in Figure 1, the average probability of this similar data occurrence is over 72%.

Based on this observation, BD-Encoding reduces the hamming weight of data words by transferring the bitwise difference between the data word that is currently transferred (the current data word), and the data word most similar to it among data words that have recently been sent over. This operation is executed only when similar data words exist, where the hamming weight of the bitwise difference is smaller than that

[1]For POD, we assume that data is transferred through the data channel in an inverted manner, since this is usually more energy efficient due to the abundance of 0-bits.

[2]Modern DRAMs have adopted a prefetch architecture.

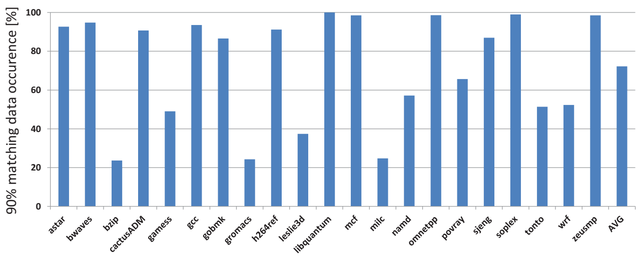


Fig. 1. Probability of 90% data matching among 64 recent data words

of the current data word. Otherwise, BD-Encoding transfers the current data word.

Some prior works have exploited the data characteristic to reduce the switching energy in the data bus. SILENT tried to transfer the XORed data between adjacent data words to reduce the amount of switching activity in the on-chip serial link [9]. However, SILENT is effective only for workloads which have a strong correlation between adjacent data words. In contrast, BD-Encoding is effective for overall workloads, since it exploits the correlation between the original data word and recent data words, and transfers the encoded data only when it is beneficial. Other prior works have exploited the occurrence of repetitive data words to reduce the switching energy in the off-chip memory data bus [4], [5], [10], [11]. Our work is different from these works because our work exploits the data similarity, which is more effective than the data equality used in these works. A qualitative and quantitative comparison between theses approaches will follow in Sections 6 and 7.

We verified BD-Encoding by using SPEC 2006 workloads. In our evaluation, BD-Encoding using 64 recent data reduced termination energy by 58.3% and switching energy by 45.3%. These results show that BD-Encoding is more efficient than prior works such as SILENT [9] (termination 5%, switching 3%), Power Protocol [4] (termination 25%, switching 20%), and VALVE [12] (termination 34.8%, switching 25.7%).

In summary, this paper makes the following contributions.

- We proposed data encoding to reduce the termination energy in the DRAM data bus which use asymmetric termination. To our knowledge, this is the first work to reduce energy in asymmetric termination by using the data characteristic. In addition, our encoding reduces the switching energy and power noise.
- We observed that a similarity exists between the data word to be transferred and the data words recently sent over in the DRAM data bus. The similar data words refer to the data words whose hamming distance is small.
- We have suggested concrete mechanisms to operate BD-Encoding in modern DRAM systems. We also analyzed the overheads of BD-Encoding precisely using HSPICE simulation and layout tools.
- The operation of BD-Encoding has been successfully verified through an evaluation performed using SPEC 2006 workloads. In the evaluation, BD-Encoding decreased the termination energy by 58.3% and the switching energy by 45.3%. In addition, BD-Encoding also decreased the LdI/dt power noise by 55%.

II. Background

In this section, we provide background information to permit easy understanding of our work.

A. DRAM Data Bus Structure

A DRAM system is composed of many potentially independent Dual In-line Memory Modules (DIMMs) [13]. A DIMM is usually controlled by a single memory controller. Each DIMM may contain one or more independent ranks. Each rank is a set of DRAM devices that operate in unison. Each rank has a 64-bit wide data bus. The number of DRAM chips depends on their individual data pin widths (e.g. a rank of x8 DRAMs would consist of 8 DRAM chips as Figure 2).

Modern DRAMs employ prefetch architecture to achieve high data rates. As the I/O interface speed has become faster, the slow DRAM core cycle has limited the overall data rate of DRAMs. To solve this problem, the prefetch architecture fetches a larger width of column data than the data pin width, and transfers the column data in a serialized manner. To support it, there are Serializer / Deserializer (SerDes) circuits in DRAMs and the memory controller (Figure 2). Since DDR3 / DDR4 SDRAM employ 8-bit prefetch architecture, 8-bit data is transferred through a single data line, and 64-byte cache line data is transferred through 64 data lines in a rank for a single READ / WRITE command.

B. Data Termination of modern DRAMs

As the data rate of the DRAM interface increases, transferring data through an off-chip DRAM data bus becomes more difficult. Specifically, the reflection of the signal at the end of the DRAM data bus has been a major obstacle to high speed data transfer. To mitigate the reflection, modern DRAMs have introduced On Die Termination (ODT). The basic ODT scheme is composed of a switch and a termination resistor which has similar impedance to the data bus. The ODT operates by connecting the termination resistor with the data pin by turning on the switch when the data is coming. Turning on the ODT causes a DC current path in the DRAM data bus, which dissipates energy.

Modern DRAM interfaces have evolved to reduce the termination DC current path [7]. Center tapped termination

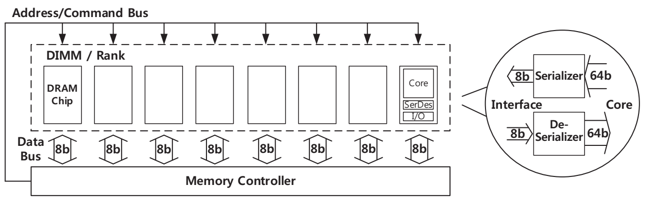


Fig. 2. DRAM Data Bus Structure

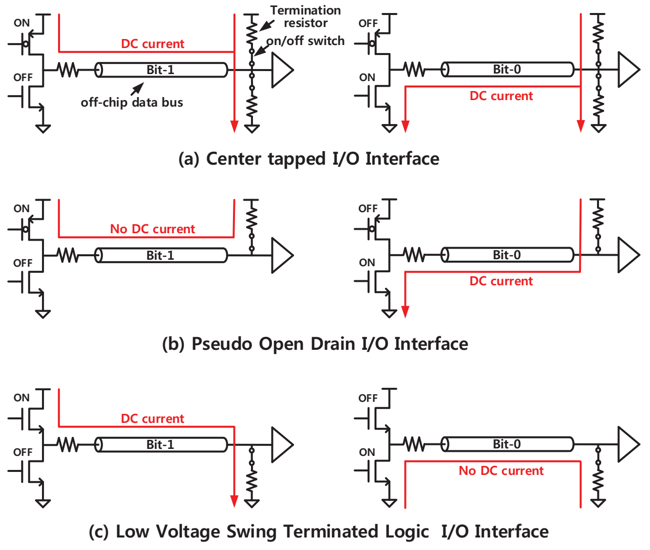


Fig. 3. Various types of termination in DRAMs

is the basic termination scheme adopted in DDR2 / DDR3 (Figure 3(a)). In the center tapped termination, the resistors are connected from data pin to both VDD and VSS, to create a DC current path at all the times the ODT is turned on. Pseudo Open Drain (POD), introduced to DDR4 and GDDR4 / GDDR5, only uses the termination resistor connected from the data pin to the VDD. As a result, the DC current path is formed only when the bit-0 is transferred, as shown in Figure 3(b). Thus, the POD interface decreases the termination energy according to the proportion of bit-1. Low Voltage Swing Terminated Logic (LVSTL), adopted in LPDDR4, only uses the termination resistor connected from the data pin to the VSS (Figure 3(c)). Thus, the DC current path is only formed when the bit-1 is transferred, which reduces the termination energy, similar to the POD.

C. Data Bus Inversion

In modern DRAMs which use POD or LVSTL, Data Bus Inversion (DBI) has been adopted to reduce both the power noise and termination energy. The power noise of DRAMs is generated because the sudden change of current causes a voltage difference across the inductance of the power line (i.e. LdI/dt noise). Since the current generated in ODT occupies the dominant part of the interface current, noise for interface-power occurs when the number of DC current paths is rapidly changed in the data bus. DBI limits the number of simultaneous DC current paths to half, which results in half the maximum interface current. Therefore, the maximum current change is also limited to half, and as a result, power noise becomes half in the worst case. DBI also reduces the termination energy because it reduces the number of DC current paths.

The specific operation of DBI is as follows. On the transmitter side, DBI compares the hamming weight of the original data and the inverted data on 8-bit granularity. Then, it transfers the data word which have less hamming weight. The comparison result is also transferred to the receiver side through an extra DBI line. In the receiver side, the decoder can determine whether the incoming data is inverted data or not by monitoring the DBI line.

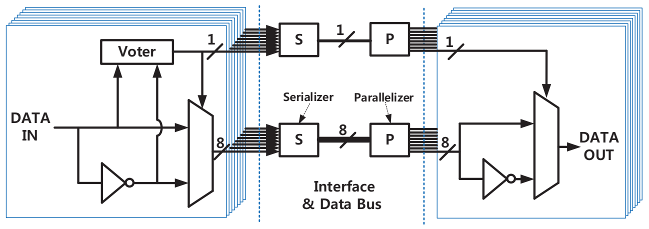


Fig. 4. Logic diagram of Data Bus Inversion

Figure 4 depicts the DBI logic diagram. The DBI encoder / decoder are located in non-serialized data area to process the comparison in parallel, as shown in Figure 4. A single extra line is required per each DRAM chip to transfer the comparison results. The majority voter is a circuit used to compare the hamming weight of two codes, and the latency of it determines the overall latency required for DBI.

III. Motivation

Figure 5 shows the energy dissipated in DRAMs and off-chip data bus (DDR4-2133). These results were calculated by using Micron DDR4 power calculator [14][3] and DDR4 specification [15]. We modified Micron DDR4 power calculator to calculate 1) termination energy changed according to the hamming weight[4], 2) switching energy assuming 15 [pF] channel capacitance and half swing voltage rail[5]. We assumed that the hamming weight and switching activity of data words were half of the max case.

As shown in Figure 5, the energy dissipated in the off-chip data bus represents a huge portion of the total energy dissipated in the DRAM sub-system. Precisely, the termination and switching energy takes 14.1% / 7% of the DRAM sub-system energy, respectively. The sum of both energies takes the proportion over 20%, which is similar to the activate / precharge energy. Therefore, reducing the energy dissipated in the off-chip data bus is crucial.

The energy consumed in DRAMs will be reduced by the shrinking DRAM transistors. However, the energy dissipated in off-chip data bus will remain constant, irrespective of shrinking transistors. Thus, the proportion for the data bus energy will increase without an additional interface technology development. Up to now, the main strategy of DRAM vendors for reducing the data bus energy have been lowering the voltage level of the interface signal. But, the signal voltage level has already been lowered thoroughly, which is hard to

[3]DDR4 power calculator setting: BND_PRE (30%), CKE_LO_PRE (15%), CKE_LO_ACT (15%), PH (20%), RDsch (20%), WRsch (10%)

[4]Micron DDR4 power calculator assumed the hamming weight as 100%.

[5]For calculation, we used the formula, $E = CV^2$.

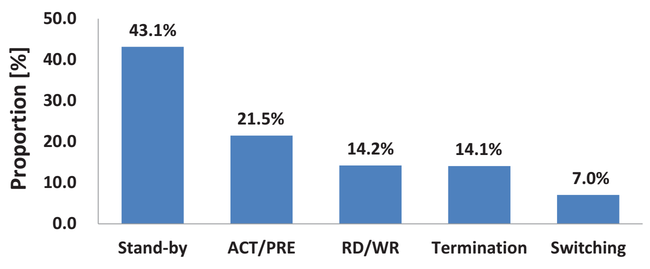


Fig. 5. Energy dissipation analysis in DRAM sub-system

decrease additionally. Therefore, we propose reducing the data bus energy by using the novel data encoding.

IV. Overview

The data bus energy can be decreased by reducing the hamming weight of data words. As mentioned in Section 1, in the interface using asymmetric termination, the termination energy is proportional to the number of DC current paths, and thus, the hamming weight. In addition, the hamming weight reduction reduces the occurrence of switching activity since it reduces the local difference in a data word. Therefore, BD-Encoding works by targeting a reduction in the hamming weight of data words.

We found an opportunity to reduce the hamming weight of data words by observing the similarities among data words recently transferred. As described in Figure 1, the data word passing through a DRAM data bus usually has low hamming distance from one of the data words that have been recently sent over. The main reasons for this similarity are repeated values, data structures whose dynamic ranges are narrow, tables of pointers, images with low color gradient, etc [16]. Thus, the bitwise difference between the current data word and one of the similar recent data words has a lower hamming weight than the current data word in many cases. Therefore, BD-Encoding transfers the encoded data as per the algorithm described in Table I.

The overall structure of BD-coder is depicted in Figure 6. To support the algorithm presented in Table I, recent data tables and one extra index line per DRAM chip are required. The recent data tables are located in both the memory controller and DRAM sides, and are updated equivalently. The recent data table would be implemented to search for the most similar data and deciding whether to transfer the encoded data or not. On the 1-bit index line, the serialized index is transferred in the form of a code in which the hamming weight is one, two, or three, in order to minimize the energy dissipated in the index line.

Figure 7 shows the example in which BD-Encoding reduces the termination and switching energy. More precisely, Figure 7 presents an example involving a 4 consecutive burst data transfer between the DRAM and memory controller. For simplicity, our explanation uses 8-bit data for a data burst, which is transferred through a single data line. In addition, we assume the granularity of BD-Encoding to be 8-bit. Also, we assume BD-Encoding uses two recent data.

TABLE I. BD-Encoding algorithm

```
Transmitter Side

    Search for the most similar data in recent data table.
    Check the bitwise difference between current data and the
    most similar data has less hamming weight than current data.
    If checking result is true
        Transfer the bitwise difference between current data and
        the most similar data through data bus.
        Transfer the table index through index line.
    Else
        Transfer the current data through data bus.
        Index data bus retains default value.
        Store the current data in the table.

Receiver Side

    Check that the index line retains the default value.
    If the checking result is true
        Transfer the incoming data to next stage.
        Store the incoming data in the table.
    Else
        Read the stored data with the index indicates.
        Transfer the bitwise difference between the incoming data
        and read data to next stage.
```

Before the first burst, the tables in the DRAM and memory controller already stored the data which were transferred through the data bus. Since the data for the first burst is similar to the data stored in ❷, BD-coder transfers the bitwise difference, ’00000010’, between the data originally to be transferred, ’00011011’, and the data stored in ❷, ’00011001’. This encoding reduces the hamming weight from 4 to 1, and the switching activity from 4 to 2. At the second burst, since the data is similar with the data stored in ❶, BD-coder transfers the bitwise difference between the data to be transferred and the data stored in ❶. This encoding reduces the hamming weight from 4 to 1, and the switching activity from 6 to 2. However, at the third burst, the data has less hamming weight than the bitwise differences between the data and all the stored value. Therefore, BD-coder transfers the original data ’00000001’ and stores it in table ❷. At the fourth burst, BD-coder reduces the hamming weight from 5 to 2, and the switching activity from 6 to 2. In this example, BD-Encoding reduces the hamming weight from 14 to 5, the switching activity from 18 to 8 in total.

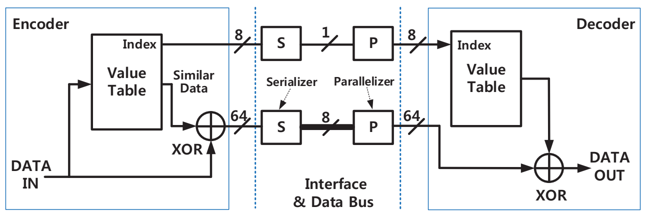
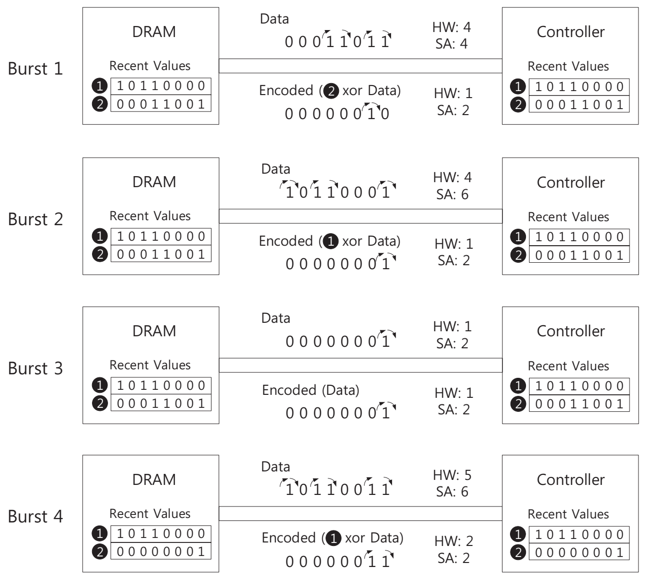


Fig. 6. Overall structure of BD-coder



Total Sum	Data	BD-Encoding
Hamming Weight (HW)	14	5
Switching Activity (SA)	18	8

Fig. 7. Operation example of BD-Encoding

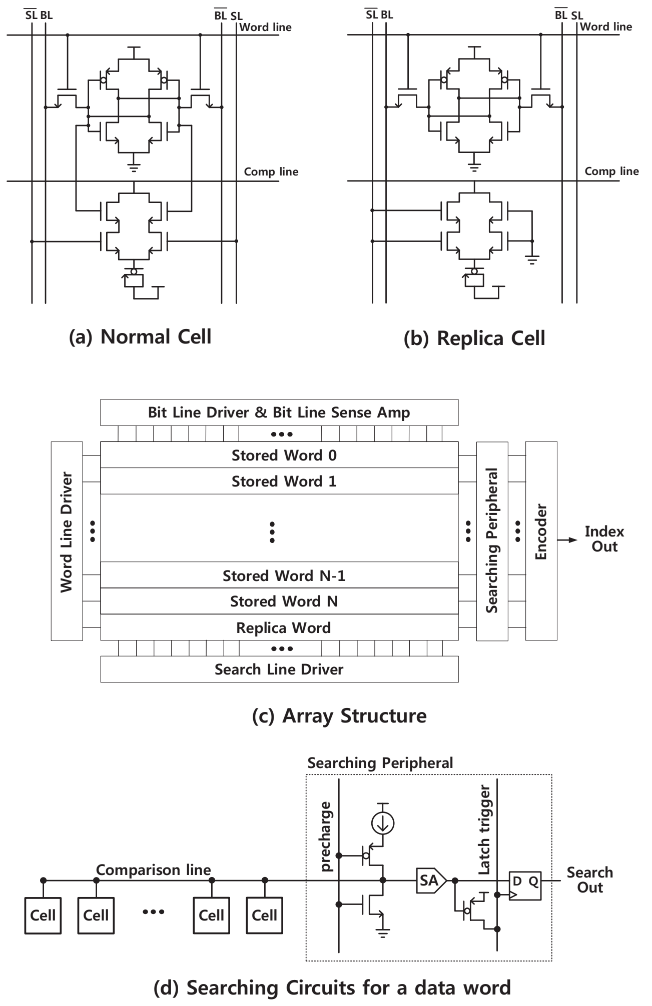


Fig. 8. Implementation of Recent Data Table

V. Bitwise Difference Coder

In this section, we will describe the implementation of BD-coder. In addition, we quantitatively analyze the overhead of BD-coder using HSPICE simulation and layout tool.

A. *Recent Data Table*

The main mechanism for BD-coder involves recent data tables. Thus, we will discuss this feature first.

Cell Structure. The main operations of the recent data table are to store the data, read the data, and search for the most similar data among all the stored data. To enable these operations, the cells of the recent data table were designed as shown Figure 8(a). The cell structure is motivated by that of the NOR binary Content Addressable Memory (CAM) [17]. We modified the CAM cell to search for the most similar value instead of a matched value. The cell includes 6T-SRAM to read and write data. In addition, the cell includes 5 transistors for search operation. Among the 5 transistors, 4 NMOS are used as a comparator, which compare the 1-bit stored value to the 1-bit search input. The other PMOS is used as a capacitor. When the search input and the stored value are different, the capacitor is connected to the comparison line and vice versa. The recent data table searches for the most similar data word by comparing the number of capacitors connected to each comparison line.

The replica cell, which is depicted in Figure 8(b), is placed in a replica word in the array as shown in Figure 8(c). The replica word is required to compare the hamming weight of the search input to that of the bitwise differences. The replica cell has the same transistor layout as a normal cell but has different signal connections. In contrast to a normal cell, the capacitor connection is determined only by the search input (bit-0: disconnect, bit-1: connect). Therefore, the number of capacitors connected to the comparison line is determined by the hamming weight of the search input.

Array Structure. The overall structure of a recent data table is depicted in Figure 8(c). A single entry in the recent data tables consists of 64 cells which corresponds to a 64-bit data word. The peripheral circuits for SRAM and search operation are located around the cell structure. The specific circuit diagram used for search operation is represented in Figure 8(d). The search peripheral includes the precharge nmos, current mirror & pmos switch, sense amplifier, D-flipflop and latch trigger circuit.

Operation. The read and store operations of the recent data table are the same as those of a normal SRAM. Therefore, we will skip the read / store operation and describe the search operation precisely.

The search operation follows the sequence mentioned below. First, before searching, all the comparison lines are discharged to ground level by the precharge nmos. In this step, all the capacitors in the cell structure are connected to comparison lines. Also, the latch trigger line is discharged to ground. Second, as the search input comes in, any capacitor in the cell, whose input bit and stored bit are the same, becomes decoupled from the comparison line. Therefore, the number of connected capacitors in the cell structure is the same as the hamming distance between the input word and the stored word. Third, the current source starts to charge the comparison line. The charging time of the comparison line is proportional to the number of connected capacitors. That is, the comparison line of the most similar data word is charged first. The charging time to the threshold voltage of the sense amplifier can be represented by (1). Fourth, when the comparison line of the most similar data word is charged to the threshold voltage of the sense amplifier, the latch trigger signal is generated. The latch trigger signal causes all D-flipflops to hold the sensing output at that instance. Since the firstly charged comparison line triggers the D-flipflops, only the entry which stores the most similar data shows a *high* search out, and other entries shows *low* search outs. Fifth, the encoder encodes the search outputs to an 8-bit code.

(1) t = Vth (Cparasitic + N*Ccell) / Isource

In the replica word, the number of capacitors which connect with the comparison line is proportional to the hamming weight of the search input. Therefore ’N’ in (1) is determined by the hamming weight of the search input. That is, if the hamming differences between search input and all the stored data are over the hamming weight of the search input, the comparison line of the replica word is charged for the first time. Therefore, the search output of the replica word only shows *high*.

Approximation Technique for Latency Reduction. We improved the searching latency of the recent data table by exploiting a data characteristic. Figure 9 shows the cumulative distribution of the hamming distance between the current data and the most similar data among 64 recent data words, when similar data word exist. As shown in Figure 9, the probability that the hamming distance will be less than 25 is over 99.5% on average. Based on this characteristic, we ignore the case where the hamming distance between the current data and the most similar data is over 24. To implement this algorithm, if no comparison line has been charged to the threshold voltage after a specific time has gone by, the latch trigger signal is automatically generated. In that case, the latched outputs of all entries are *zero*, and BD-coder selects the current data instead of the encoded one. As a result, the worst case latency is reduced from *Vth (Cparasitic + 64*Ccell) / Isource* to *Vth (Cparasitic + 24*Ccell) / Isource*.

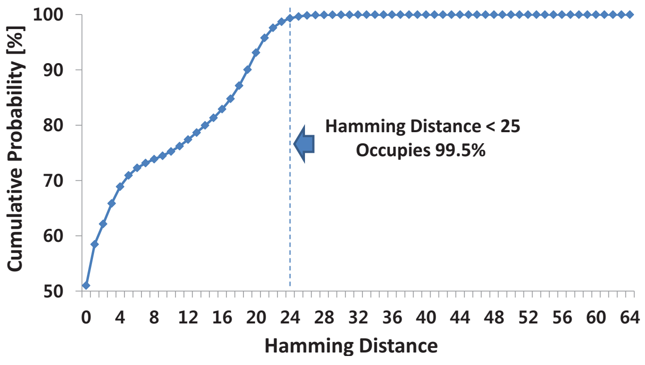


Fig. 9. Cumulative probability distribution of hamming distance between current data word and the most similar data word among 64 recent data words

PVT Variation. Like all electronic circuits, the searching circuits of Recent Data Table can operate incorrectly by PVT variation. However, the process variation can be calibrated by using various calibration techniques. Moreover, the effect of PVT variation is restricted to just decreasing the energy reduction rate slightly. For example, if Recent Data Table searches the stored word which has 1 or 2 more hamming distance than the most similar data words due to the PVT variation, the encoded result has 1 or 2 more hamming weights than the ideal result; it does not cause logical fail in DRAMs. Therefore, the searching circuits have relatively lower risk than other circuits in DRAMs according to the PVT variation.

B. Overall Operation

In the transmitter side, BD-coder first searches the entry which stores the most similar data word. If one of the search outs of stored words shows *high* (i.e. similar value exists), BD-coder reads the entry which searching result indicates. Then, BD-coder generates the bitwise difference between the current data and read out data by using XOR logic and transfers it to the I/O interface circuit. If the search outs of replica word show *high* or all search outs show *low* (i.e. similar value does not exist), BD-coder stores the input data word to the recent data table as a round robin policy. In this case, the sense amplifiers of SRAM are not turned on and output the default value, *zero*. Therefore, the output of BD-coder is the same as the current data since the XORed result between the current data and *zero* is the current data. When multiple search outs show *high*, an encoder logic selects the most recently stored entry as a result.

In the receiver side, BD-coder decodes the index line at first. If the index line is not a default value, BD-coder reads the entry which the index line indicates. Then, BD-coder transfers the bitwise difference between the input data and the read out data to the next stage. If the index line is the default value, BD-coder stores the input data word to the recent data table as round robin policy.

C. Overhead Analysis

Table II shows the overhead of BD-coder. We used a 65 nm process to get HSPICE and layout results. In this section, we precisely described the result of the overhead analysis.

Latency. The data latency increment (2.3 ns) of the transmitter is the combination of the searching (1.6 ns), reading (0.6

TABLE II. OVERHEAD ANALYSIS FOR BD-CODER

	Num Entry	Additional Latency (ns)	Cycle Time (ns)	Energy (pJ)	Area (um^2)
Transmitter	4	2.3	2.4	1.2	4,191
	16			2.4	10,240
	64			7	33,280
Receiver	4	0.7	0.8	0.9	4,191
	16			1.17	10,240
	64			2	33,280

ns), and XORing (0.1 ns) time. This result is slightly larger than the latency increment of DBI scheme (2-clk in DDR4-2133, i.e. 1.875 ns, [15]). The data latency increment of the receiver (0.7 ns) is the combination of the reading (0.6 ns) and XORing time (0.1 ns).

Cycle Time. The cycle time of the transmitter (2.4 ns) is the combination of the searching (1.6 ns), and storing (0.8 ns) time[6]. This cycle time enables the operation up to 6.66 Gbps in LPDDR4 (16-bit prefetch) and 3.33 Gbps in DDR4 (8-bit prefetch). In addition, dividing the recent data tables into each bank-group enables the operation up to 6.66 Gbps in the DDR4 architecture. The cycle time of the receiver (0.8 ns) is the same as the storing time (0.8 ns).

Energy. The energy dissipated in BD-coder is affected by the number of entries in the recent data table. The reasons are as follows. First, the energy involved in search operation is almost proportional to the number of entries. This is because the charge / discharge energy for the comparison lines is proportional to the number of the comparison lines. In addition, the charge / discharge energy for the search line is proportional to the length of the search line, which is proportional to the number of entries. Second, the read / store energy of the recent data table is affected by the number of entries because the length of the bit-line is affected by it.

Area. We drew the layout of BD-coder. In our layout, the size of a single cell of the recent data table is 7 um^2. The size of 4 / 16 / 64 entries BD-coder is 4,191 um^2 / 10,240 um^2 / 33,280 um^2, respectively. According to the DDR4 SDRAM size reported in a recent paper [18], 4 / 16 / 64 entries BD-coder occupies 0.006% / 0.014% / 0.044% of the DRAM area, respectively.

Index pins. For BD-coder, a single extra pin per 8 data pins is required to transfer the table index. To save the overhead, BD-coder can share its index pins with multipurpose pins already existing in commodity DRAMs. DM (Data Mask), DBI, and TDQS, which are the functions of DDR4, require extra pins, and they share the extra pins. The purpose of the extra pins can be programmed through DRAM mode register. Therefore, BD-coder can exploit the multipurpose pins for the index pins, and DRAM users can test and determine the optimal purpose for those pins.

[6]Since the delay for *search-store* case is bigger than that of *search-read* case, *search-store* case limits the cycle time.

D. Architecture for multi-core environments

In the multi-core system, BD-coder can perform the same as in the single-core system by locating a separated recent data table for each core. For column accesses, BD-coder checks core ID[7] and accesses the corresponding core's recent data table. Since requests of a particular core do not influence the contents of other core's recent data table, BD-coder can perform the same as in the single-core system. In this case, the area overhead for the total recent data tables is increased. For example, a BD-coder which has four recent data tables (64-entries) occupies 0.176% of DDR4 DRAM area.

VI. RELATED WORKS

In our study, we exploited a data characteristic of DRAMs to reduce the data bus energy. In this section, we discuss prior works and qualitatively compare them to BD-Encoding.

SILENT. Lee *et al.* proposed a serialized low-energy transmission (SILENT) coding technique to minimize the switching energy of the on-chip serial link [9]. SILENT reduced the hamming weight of data words to reduce the switching energy of the data bus, by exploiting the correlation between adjacent data words. BD-Encoding is distinct from SILENT for several reasons. First, SILENT can reduce energy only in workloads, which always have strong correlation between adjacent data words. In contrast, BD-Encoding encodes data words only when the correlation is strong enough to decrease the energy, and this characteristic extends the scope of workloads. Second, SILENT only exploits correlations between current and precedent data words, while BD-Encoding exploits the correlation between current and recent data words, which makes the correlation much stronger. Third, SILENT attempts to only decrease the switching energy, but we propose reducing both the termination and switching energy, as appropriate to the modern DRAM interfaces. A quantitative comparison will be shown in Section 7.

Coding exploiting data equality. Several prior works have exploited repeated values to reduce the switching energy in the memory data bus [4], [5], [19], [20] and on-chip data bus [10], [21]. They have proposed locating the data tables in both the memory controller and memory sides. When the transferred data is matched in the data table, a table index, which has a small hamming weight, is transferred instead of the data itself. Other prior works have improved these works by using multiple data tables, which store various length of data [11], [12]. Even though all these works exploits data equality, BD-Encoding is fundamentally different from these works since it exploits data similarity. In other words, the prior works can decrease the data bus energy only when the transferred data has been perfectly matched with data in the table. In contrast, BD-Encoding decreases the data bus energy when the transferred data is similar to the recent data. Since the probability for similar data occurrence is much higher than that for the same

[7]The Core ID can be transferred from memory controller to DRAMs by using redundant address lines. Since the number of row address bit is bigger than that of column address bit, there are redundant address lines when the column-address transfers.

data occurrence, BD-Encoding outperforms previous works in energy reduction results. A quantitative comparison will be described in Section 7.

Bus invert coding. Stan *et al.* proposed Bus-Invert Coding to lower peak and average power dissipation in the data bus [22]. This scheme found code which lowered switching activity between the original and inverted codes, and transferred it. Other prior works applied the Bus-Invert Coding to an open drain interface, by comparing the hamming weight of the original and inverted codes rather than the amount of switching activity [23], [24]. These works were adopted in GDDR4 at first, and subsequently in other generations such as GDDR5, DDR4, and LPDDR4 [6], [7], [8]. These works are collectively referred to as Data Bus Inversion (DBI) in the DRAM specification. The quantitative results of the approach will be described in Section 7.

Address / Instruction bus coding. Several prior works have decreased the switching activity on address / instruction bus [25], [26], [27], [28], [29], [30]. However, these works are distinct from data bus encoding since the value characteristic of the address / instruction is different from that of data.

Data compression exploiting data similarity. Pekhimenko *et al.* exploited data similarity to compress data in the cash memory [16]. By storing data in a compressed manner, they have increased on-chip cache capacity, and have decreased on-chip / off-chip bandwidth. They have exploited the data similarity existing in a single cache line. Since BD-coding exploits the data similarity between multiple cache lines, the prior work can be used with BD-coding orthogonally.

VII. Evaluation

In this section, we review our quantitative evaluation of the benefits of BD-Encoding. We also evaluated prior works (DBI [24], SILENT [9], Power Protocol [4], and VALVE [11]) for comparison.

A. Methodology

In this work, our evaluation follows the steps mentioned below. First, we extracted the data trace of the DRAM data bus using a system simulator. We used GEM5 [31] as our simulation platform: the specific system configuration is mentioned in Table III. For the simulation workloads, we used SPEC 2006 benchmarks [32] with reference input size. For the simulation, 10B instructions were fast-forwarded and then 100M instructions were executed. Second, we evaluated the benefits of BD-coder (i.e. the reduction of hamming weight and switching activity) using an in-house tool. All the results of our evaluation include the energy dissipated in the index line. Third, we calculated the overall energy dissipated in the DRAM data bus. The termination energy was calculated using the Micron Power Calculator [14]. The switching energy was calculated based on the formula $E = CV^2$. The specific value used for the calculation is mentioned in Table III.

TABLE III. EVALUATED SYSTEM CONFIGURATION

Processor	**3.3 GHz, Out of order (O3), 192 reorder buffer 32 load/store queue, 8 width issue/commit**
Cache	**L1: I-cache (32KB, 4way) D-cache (64KB, 4way) L2 cache : (2MB, 8way)**
DRAM System	**DDR4-2133, 1 channel, 2 rank, 8 bank, 8GB 15 pF per data-bus Termination Resistance Rzc: 34Ω, RTT_WR: 80Ω, RTT_OTH: ∞ Ω, RS: 10Ω**

For prior works using the data tables, we used the table index which has small hamming weight (Power Protocol: 1, VALVE: 1 or 2). For VALVE, we used 4 tables which store various length of data (64-bit, 48-bit, 32-bit, 16-bit) for a single entry.

B. Termination Energy

Figure 10(a) shows the portion of the termination energy that was reduced (i.e. hamming weight reduction rate) using BD-Encoding, according to the number of table entries. We draw two conclusions. First, as shown in Figure 10(a), BD-Encoding decreases the termination energy in all the workloads. Even in the workload which shows the least effect, bzip2, BD-Encoding using 64 table entries reduces 29% of the termination energy. This means that data similarities exist in most workloads. Second, the hamming weight reduction rate increases as the number of table entries increases. On average, BD-Encoding decreases hamming weight by 28% / 36% / 43.3% / 49.3% / 53.3% / 56.5% / 58.3% for 1 / 2 / 4 / 8 / 16 / 32 / 64 entries in a recent data table.

The termination energy reduction rate is different across the workloads depending on the data characteristic of each workload. First, in the workloads which have high data similarity, such as bwaves, libquantum, mcf, soplex, and zeusmp, BD-Encoding using 64 table entries reduced the termination energy more than 80%. Second, in workloads such as libquantum and soplex, the difference of the hamming weight reduction rate between a 1 table entry and 64 table entries of BD-Encoding is under 10%, whereas those of other workloads are over 20%. This is because, in these workloads, the similarity between consecutive data words is much higher than the similarity between non-consecutive data words. Therefore, in these workloads, BD-Encoding using a single table entry, is sufficiently effective. However, in other workloads, multiple table entries are required to show effective results.

Figure 10(b) shows the termination energy reduction rate of BD-Encoding and prior works (DBI, SILENT, Power Protocol using 64 table entries, VALVE using 64 table entries). Even though SILENT, Power Protocol, and VALVE target to reduce the switching energy, they also reduce the termination energy since the hamming weight of the data words is reduced as a result of the encoding. The results of DBI are from 0.03% (astar) to 23% (zeusmp) and the average result is 10.9%. The results of SILENT are from -47% (sjeng) to 90.2% (libquantum) and the average result is 3.3%. The results of Power Protocol using 64 table entries are from 3.5% (soplex) to 47% (cactusADM), and the average result is 25.5%. The results of VALVE using

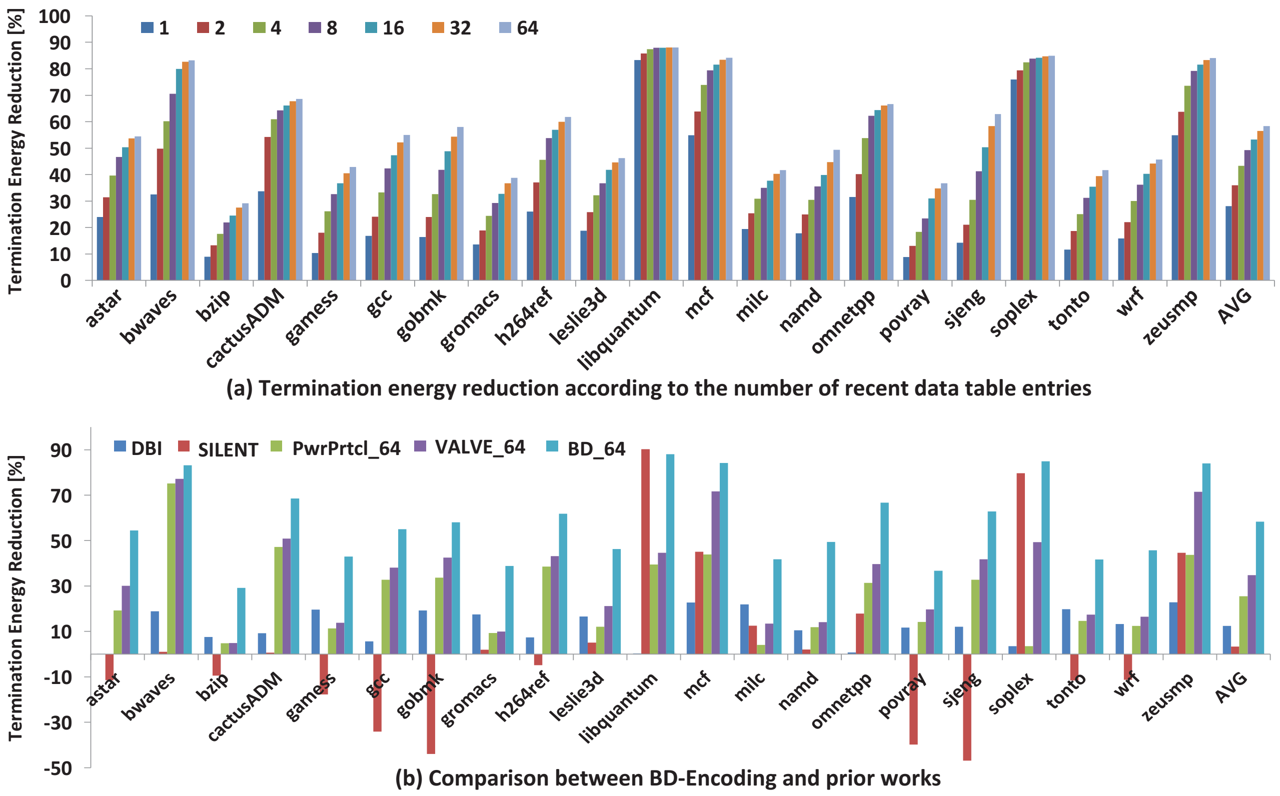


Fig. 10. Termination Energy Reduction by BD-Encoding

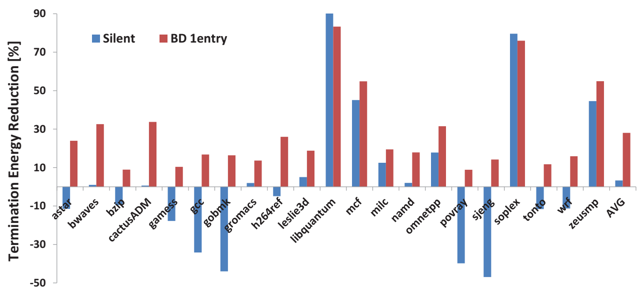


Fig. 11. Comparison between BD-Encoding and SILENT

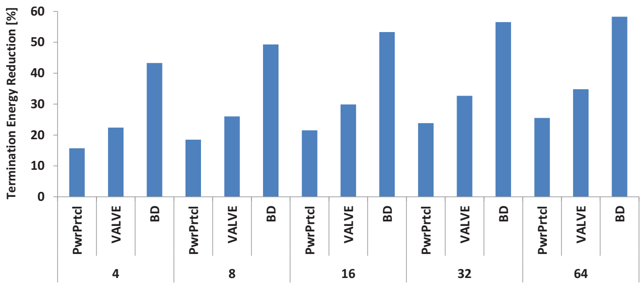


Fig. 12. Comparison between BD-Encoding and Data Equality Coding

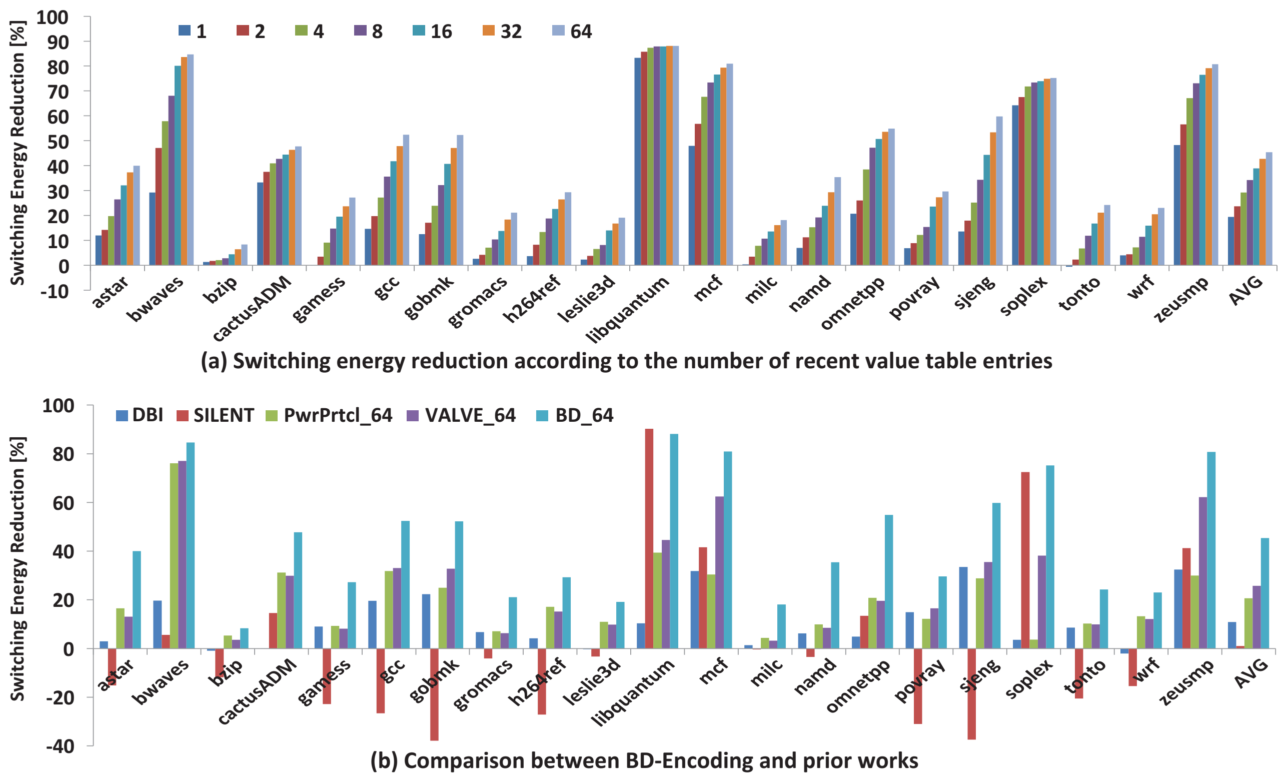
64 table entries are from 10% (gromacs) to 77.7% (bwaves), and the average result is 34.8%. Against the results of prior works, BD-Encoding using 64 table entries shows outstanding results; the results are from 29% (bzip2) to 88% (libquantum), and the average result is 58.3%.

C. Comparison to Prior Works

In this section, we more precisely compare the termination energy reduction rate of BD-Encoding with the prior works.

SILENT. Figure 11 shows the relative proportion of reduced termination energy for SILENT and BD-Encoding using a single table entry. As mentioned in Section 6, SILENT reduces the termination energy effectively if there is strong correlation between adjacent data words. However, if the correlation is weak, it fails to reduce the termination energy. For instance, in the workloads libquantum and soplex, there is usually a strong correlation between adjacent data words, and thus, SILENT works effectively. Otherwise, in most workloads, SILENT does not work effectively and sometimes fails to reduce hamming weight (e.g. astar, bzip, gamess, gcc, gobmk, h264ref, povray, sjeng, tonto, and wrf).

BD-Encoding using a single table entry uses only the precedent data word, just as SILENT does. However, it transfers the bitwise difference only when the hamming distance between the adjacent data words is lower than the hamming weight of the current data word. In other words, it transfers the encoded data only if there is a strong correlation between adjacent data words. Therefore, BD-Encoding using a single table



(a) Switching energy reduction according to the number of recent value table entries

(b) Comparison between BD-Encoding and prior works

Fig. 13. Switching Energy Reduction by BD-Encoding

entry can decrease the termination energy in those workloads which *sometimes* have correlations between adjacent data words, unlike SILENT. On average, SILENT decreases the hamming weight by 3.3%, and BD-Encoding using 1 table entry decreases the hamming weight by 28%.

Furthermore, BD-Encoding exploits the correlation between the current data and other recent data in addition to the correlation between adjacent data words. This property reduces 30.3% more termination energy, as shown in Figure 10(a). Since the datasets for graphic processors which have been targeted in SILENT have a high correlation between adjacent data words, SILENT showed good energy reduction in [9]. However, as shown in our evaluation, workloads which has a strong correlation between adjacent data words are rare (i.e. libqauntum and soplex). Therefore, for application to a wider range of workloads, BD-Encoding is more appropriate than SILENT.

Coding Exploiting Data Equality. Figure 12 shows the averaged termination energy reduction rate of Power Protocol, VALVE, and BD-Encoding according to the number of table entries. As shown in Figure 12, the termination energy reduction results for Power Protocol is the lower than that of VALVE. This is because Power Protocol can work only when the current data word is exactly same as the recent data words. On the other hand, VALVE can reduce the termination energy when current data words is partially matched with the recent data words. BD-Encoding shows the highest termination energy reduction results among the three encodings, since it can reduce the termination energy when the current data word is similar to the recent data words, as well as the cases where they are the same or partially matched. This is important because, the occurrence of similar data is much more frequent than the occurrence of matched data. As a result, BD-Encoding always reduce the hamming weight more than codings exploiting data equality.

D. Switching Energy Reduction

Figure 13(a) presents the proportion of reduction in switching energy (i.e. number of switching activity) based on the number of table entries. As shown in Figure 13, the tendency in switching energy reduction is similar to that of termination energy reduction. As with the termination energy, BD-Encoding can decrease the amount of switching activity in all the workloads; in bzip2, which shows the least effect, BD-Encoding using 64 table entries reduces the switching activity by 8.4%. In addition, as the number of table entries increases, the effect of BD-Encoding increases. On average, BD-Encoding decreases the switching activity by 19.4% / 23.7% / 29.3% / 34.2% / 38.9% / 42.7% / 45.3% for 1 / 2 / 4 / 8 / 16 / 32 / 64 entries of recent data table.

Figure 13(b) shows the proportional reduction in switching energy when using DBI, SILENT, Power Protocol using 64 table entries, VALVE using 64 table entries, and BD-Encoding using 64 table entries. The overall tendency is similar to that

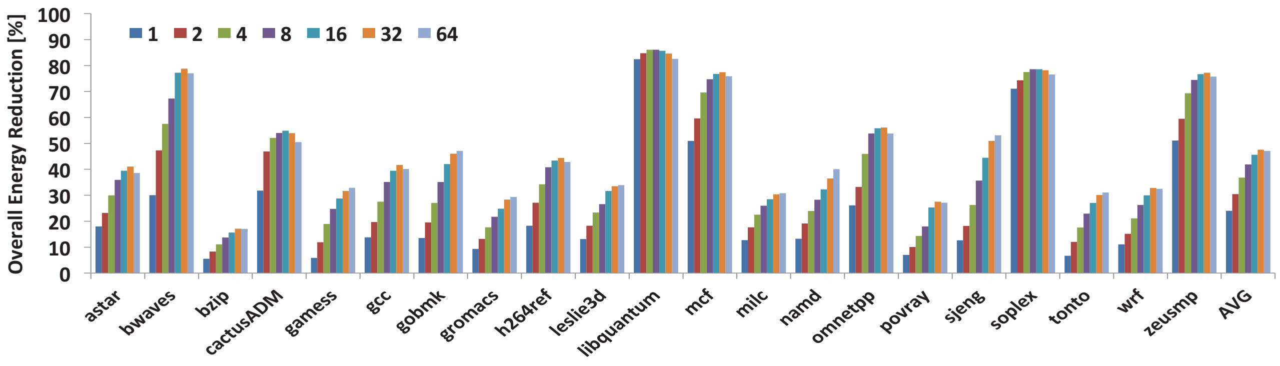


Fig. 14. The Energy Reduction including Termination, Switching, and BD-coder

observed for termination energy reduction. There is a small difference in the results for DBI. Since DBI transfers the optimal data by monitoring the hamming weight of the current data and the inverted data, it always reduces the hamming weight of the data. However, reducing the hamming weights does not decrease the switching activity at all times. Thus, in some workloads, such as bzip and wrf, switching activity is increased by using DBI. This also occurs with BD-Encoding, but since BD-Encoding decreases the hamming weights more effectively than DBI, the DBI level of decreases in switching activity does not occur with BD-Encoding. On average, the switching activity is reduced by 10.9% / 1.1% / 20.7% / 25.8% / 45.3% in DBI / SILENT / Power Protocol / VALVE / BD-Encoding.

E. Energy Reduction in Data Bus and BD-coder

In this section, we describe the overall energy reduction in data bus using POD interface. The energy evaluated in this analysis includes the termination energy, switching energy, and the energy needed by the BD-coder hardware. In our evaluation, the ratio between the termination and switching energy is 63:37 on average.

Figure 14 shows the energy reduction rate in the POD data bus according to the number of table entries. As shown in Figure 14, the optimal number of table entries exists. This is because, 1) the energy reduction effect of increasing the number of table entries becomes saturated, 2) the energy overhead of BD-coder is almost proportional to the number of table entries, 3) the energy dissipated in the index line increases as the number of entries rises. On average, the proportion of the reduced energy is 24% / 30.4% / 36.8% / 41.9% / 45.6% / 47.6% / 47.1% in 1 / 2 / 4 / 8 / 16 / 32 / 64 table entries.

Figure 15 shows the overall energy dissipation of BD-encoding, normalized to the baseline. The baseline is the sum of termination and switching energy without BD-Encoding. Figure 15 also shows the proportion of the BD-coder hardware energy[8]. When the number of table entries is small, the hardware energy for BD-coder is relatively negligible. However, as the number of table entries increases, the proportion that the hardware energy occupies becomes bigger. The averaged hardware energy for BD-coder is 1% / 1.3% / 1.6% / 2.2% / 2.7% / 4.3% / 6.8% in 1 / 2 / 4 / 8 / 16 / 32 / 64 entries[9].

[8]We used the values calculated in SPICE simulation for the BD-coder hardware energy.

Figure 15 also shows the proportion of the energy dissipated in the index line. As shown in Figure 15, the energy dissipated in the index line increases when the number of table entries rises. This is because 1) more indexes are needed to be transferred as more data words are transferred in the encoded manner, 2) the hamming weight of the index can rise when the number of entries increases. The averaged energy dissipated in the index line is 2.8% / 3.6% / 4.3% / 4.7% / 6.1% / 7.2% / 8.4% in 1 / 2 / 4 / 8 / 16 / 32 / 64 entries.

F. Power Noise Analysis

Figure 16 shows the distribution in the number of changed DC current paths in the data bus. The value depicted in Figure 16 is the averaged value for all of the evaluated workloads. As mentioned in Section 2.3, the amount of power noise is almost proportional to the number of changed DC current paths (i.e. LdI/dt noise). For each x8 DRAM chip using an asymmetric termination, the maximum number of DC current path is 8, and the minimum number is 0. Thus, the number of changed DC current paths is also distributed between 0 and 8.

DBI reduces number of changed DC current path to 4 in the worst case. Therefore, the proportion of changed DC current paths between 5 and 8 becomes zero, but the proportion,

[9]The hardware energy for BD-coder can be lowered as the gate-length of transistor shrinks.

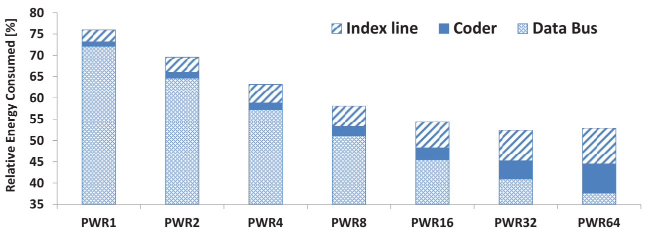


Fig. 15. The proportion of Data Bus and BD-coder Hardware energy consumption

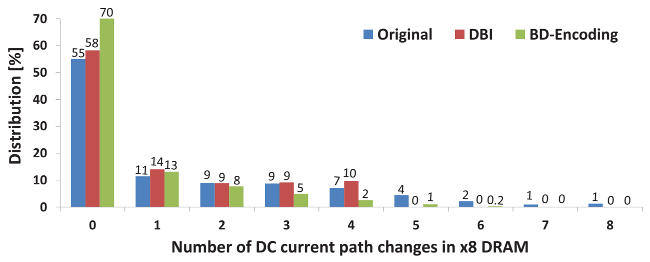


Fig. 16. Distribution for changed DC current paths in the data bus

between 0 and 4, increases, as shown in Figure 16. Meanwhile, BD-Encoding shifts the overall distribution of changed DC current paths to 0, as depicted in Figure 16. The expected value for the number of changed DC current path is 0.15 / 0.11 / 0.068 for Original / DBI / BD-Encoding, respectively. Consequently, BD-Encoding reduces 55% of the power noise on average, which is more effective than DBI (27% reduction).

VIII. Conclusion

We have proposed BD-Encoding, an approach which reduces energy dissipated in DRAM data bus. By reducing the hamming weight of data words, BD-Encoding reduced not only energy in data bus, but also the power noise. BD-Encoding reduced the hamming weight by exploiting the data similarity of DRAMs (i.e. characteristic that similar data words are frequently sent over through a data bus at a given time interval). Based on an evaluation using SPEC 2006, BD-Encoding using an 64 table entries reduced termination energy by 58.3% and switching energy by 45.3%. This result is excellent in comparison to prior data encodings which exploit a data characteristic. Including the energy overhead of the hardware, BD-Encoding reduced overall data bus energy by 48.7%. Consequently, BD-Encoding is an effective data encoding method, which can be applied to next generation DRAMs to save data bus energy.

References

[1] H. Zheng, J. Lin, Z. Zhang, and Z. Zhu, “Decoupled dimm: Building high-bandwidth memory system using low-speed dram devices,” in *ISCA*, 2009.

[2] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu, “Memory power management via dynamic voltage/frequency scaling,” in *ICAC*, 2011.

[3] K. T. Malladi, B. C. Lee, F. A. Nothaft, C. Kozyrakis, K. Periyathambi, and M. Horowitz, “Towards energy-proportional datacenter memory with mobile dram,” in *ISCA*, 2012.

[4] K. Basu, A. Choudhary, J. Pisharath, and M. Kandemir, “Power protocol: reducing power dissipation on off-chip data buses,” in *MICRO*, 2002.

[5] J. Yang, R. Gupta, and C. Zhang, “Frequent value encoding for low power data buses,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 9, July 2004.

[6] K. Sohn, T. Na, I. Song, Y. Shim, W. Bae, and S. K. et al., “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,” *Solid-State Circuits, IEEE Journal of*, vol. 48, no. 1, 2013.

[7] S.-J. Bae, K.-I. Park, J.-D. Ihm, H.-Y. Song, W.-J. Lee, and H.-J. K. et al., “An 80 nm 4 gb/s/pin 32 bit 512 mb gddr4 graphics dram with low power and low noise data bus inversion,” *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 1, 2008.

[8] T.-Y. Oh, H. Chung, J.-Y. Park, K.-W. Lee, S. Oh, and S.-Y. D. et al., “A 3.2 gbps/pin 8 gbit 1.0 v lpddr4 sdram with integrated ecc engine for sub-1 v dram core operation,” *Solid-State Circuits, IEEE Journal of*, vol. 50, no. 1, pp. 178–190, 2015.

[9] K. Lee, S.-J. Lee, and H.-J. Yoo, “Silent: serialized low energy transmission coding for on-chip interconnection networks,” in *ICCAD*, 2004.

[10] V. Wen, M. Whitney, Y. Patel, and J. Kubiatowicz, “Exploiting prediction to reduce power on buses,” in *HPCA*, 2004.

[11] D. Suresh, B. Agrawal, J. Yang, and W. Najjar, “Tunable and energy efficient bus encoding techniques,” *Computers, IEEE Transactions on*, vol. 58, no. 8, 2009.

[12] D. Suresh, B. Agrawal, W. Najjar, and J. Yang, “Valve: variable length value encoder for off-chip data buses,” in *Computer Design: VLSI in Computers and Processors, 2005. ICCD 2005. Proceedings. 2005 IEEE International Conference on*, 2005.

[13] B. Jacob, S. W.Ng, and D. T. Wang, *Memory Systems(Cache, DRAM, Disk)*. Morgan Kaufmann, 1st ed., 2008.

[14] Micron, “DDR4 SDRAM System Power Calculator,” 2014.

[15] Samsung, “4Gb D-die DDR4 SDRAM,” 2014.

[16] G. Pekhimenko, V. Seshadri, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry, “Base-delta-immediate compression: Practical data compression for on-chip caches,” in *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques*, PACT ’12, 2012.

[17] K. Pagiamtzis and A. Sheikholeslami, “Content-addressable memory (cam) circuits and architectures: a tutorial and survey,” *Solid-State Circuits, IEEE Journal of*, vol. 41, no. 3, 2006.

[18] K. Sohn, T. Na, I. Song, Y. Shim, W. Bae, and S. K. et al., “A 1.2v 30nm 3.2gb/s/pin 4gb ddr4 sdram with dual-error detection and pvt-tolerant data-fetch scheme,” in *ISSCC*, 2012.

[19] B. Bishop and A. Bahuman, “A low-energy adaptive bus coding scheme,” in *VLSI, 2001. Proceedings. IEEE Computer Society Workshop on*, 2001.

[20] J. Yang and R. Gupta, “Fv encoding for low-power data i/o,” in *Low Power Electronics and Design, International Symposium on, 2001.*, 2001.

[21] C. Liu, A. Sivasubramaniam, and M. Kandemir, “Optimizing bus energy consumption of on-chip multiprocessors using frequent values,” in *Parallel, Distributed and Network-Based Processing, 2004. Proceedings. 12th Euromicro Conference on*, 2004.

[22] M. Stan and W. Burleson, “Bus-invert coding for low-power i/o,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 3, no. 1, 1995.

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

[24] K. Nakamura and M. Horowitz, “A 50% noise reduction interface using low-weight coding,” in *VLSI Circuits, 1996. Digest of Technical Papers., 1996 Symposium on*, 1996.

[25] L. Benini, G. De Micheli, E. Macii, D. Sciuto, and C. Silvano, “Address bus encoding techniques for system-level power optimization,” in *Design, Automation and Test in Europe, 1998., Proceedings*, 1998.

[26] L. Benini, G. De Micheli, E. Macii, D. Sciuto, and C. Silvano, “Asymptotic zero-transition activity encoding for address busses in low-power microprocessor-based systems,” in *VLSI, 1997. Proceedings. Seventh Great Lakes Symposium on*, 1997.

[27] E. Musoll, T. Lang, and J. Cortadella, “Working-zone encoding for reducing the energy in microprocessor address buses,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 6, no. 4, 1998.

[28] P. Petrov and A. Orailoglu, “Low-power instruction bus encoding for embedded processors,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 12, no. 8, 2004.

[29] Y. Aghaghiri, F. Fallah, and M. Pedram, “Irredundant address bus encoding for low power,” in *Low Power Electronics and Design, International Symposium on, 2001.*, 2001.

[30] M. Mamidipaka, D. Hirschberg, and N. Dutt, “Adaptive low-power address encoding techniques using self-organizing lists,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 11, no. 5, 2003.

[31] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, “The gem5 simulator,” vol. 39, Aug. 2011.

[32] Standard Performance Evaluation Corporation, “SPEC CPU2006,” http://www.spec.org/cpu2006.