1. Introduction

Data movement in computer systems dissipates a substantial amount of energy through charging and discharging the interconnect capacitance [1–3]. Prior studies [4–6] have shown that data movement may consume more than ten times the energy compared to arithmetic and logical computation within a processor. The discrepancy is expected to increase further with CMOS technology scaling [7–11], and it is, therefore, of a great practical importance to reduce the energy for data movement.

Data bus inversion (DBI) [12–19] is a well-known bus coding technique that lowers the energy that data movement consumes. DBI encodes a group of data bits using an extra bit called a control bit, which indicates whether the current data bits are to be transmitted over a bus as they are or in an inverted form. For example, in the case of DBI that encodes a group of 8 data bits, the most common case in practice [20–23], 8-bit data $u(t)$ is converted to 9-bit codeword $v(t)$ as shown in Figure 1. If the number of bit toggles between the previous codeword $v(t-1)$ and the current data $u(t)$ is greater than four, then the control bit is set to one and the data bits are encoded in an inverted form. Otherwise, the control bit is set to zero and the data bits are encoded as equal to the original $u(t)$. This technique reduces switching activity on a bus by 18% on average for random data bits and thus reduces the energy consumption accordingly [12].

To achieve this, DBI encoders conventionally use majority voter to count the bit toggles. However, a majority voter, which is usually comprised of population count circuitry and comparator, results in a high encoding latency. As will be shown later, the conventional 8-bit DBI encoder takes more than 0.9 nanoseconds when synthesized in a 65-nm CMOS process. It must be noted that this high latency issue can be compounded in the large-scale network on-chip buses where arbitration occurs at each individual router [24–28].

Since DBI was first introduced in 1995 [12], there have been active studies on low-energy bus coding techniques. Stan et al. [13] presented low-power coding techniques in which redundancy can be added in space, time, and voltage. Lee et al. [14] presented XOR-based coding that possibly leads to fewer 1s in the codeword. Song et al. [15] took advantage
of data bus under-utilization based on that DDR data bus utilization typically falls below 60%. They proposed to opportunistically exploit a bus in idle cycles as redundancy for sparse encoding that can achieve high switching activity savings at the expense of large extra-bit overhead. Ghosh et al. [16] proposed a low-power coding scheme dedicated for serial bus. The proposed scheme accounts for correlations in data, and accordingly, reduces the switching activity up to 25% with overhead of two extra lines. Kwon [17] proposed a coding technique optimized for OR-chained bus that can collect multiple modules without multiplexers. The proposed technique jointly considers two switching activities, one due to a change in valid data values and the other due to data parking, which translates to 3% additional saving of switching activity in average. Shin et al. [18] presented partial DBI coding, where the conventional DBI is applied only to a selected subset of bus lines in order to avoid unnecessary data inversion of inactive bus lines.

![Figure 1. Conventional 8-bit DBI encoding when \( u(t) \), \( v(t-1) \), \( v(t) \) are the current data, the previous codeword, and the current codeword.]

While aforementioned prior works have proven the potential of bus coding for energy-efficient data movement, the encoding latency may be a bottleneck in high-speed applications or largescale networks. In this work, we address the latency issue by proposing new voters that consist of only two gate levels.

Main contributions of this paper are as follows. We propose two new DBI encoders that are designed to optimize latency. The new encoders are based on circuits that we call approximate voters that operate with lower latency while allowing small errors in the majority vote decision. The errors in the majority vote decision do not corrupt the data being transferred because the DBI decoder can still recover the original data by performing a bitwise XOR operation on each bit of the codeword and the control bit. Thus, our proposed design maintains functional correctness while achieving a tradeoff between the encoding latency and the switching activity. Implemented in a 65-nm process, two proposed DBI encoders improve the latency more than three and two times, respectively, over the conventional DBI encoder made of a population counter and a comparator. Our proposed designs reduce switching activity on average by 9% and 11%, respectively, for a sequence of random data.

The rest of this paper is organized as follows. Section 2 reviews existing methods to design a majority voter for DBI encoding. Section 3 presents two proposed DBI encoders based on new latency-optimized voters. Section 4 analyzes behavior of the new voters in comparison with the existing majority voter. Section 5 shows functional correctness of the proposed DBI encoders. Section 6 presents the simulation results using various metrics including latencies, areas, operating powers, and switching activities. Section 7 concludes the paper.
2. Majority Voter

Majority voter, the Boolean circuit that evaluates a logical 1 if more than half of input bits are 1 and a logical 0 otherwise, is a main component that requires a high latency within a DBI encoder. One possible approach to a majority voter design is to use a logic synthesis tool that transforms a high-level code in hardware description language (HDL), e.g., Verilog HDL [29], into a combination of logic gates and wires. Shown in Figure 2 is the example of such approach for a 9-bit majority voter that results in eight gate levels on the critical path.

![Figure 2. Example of 9-bit majority voter using a logic synthesis. Critical path is shown in red color.](image)

Some prior works have explored alternative design methods through hierarchical decomposition [30–32]. Parhami et al. [30,31] showed that a majority voter can be constructed from multiple smaller voters along with a multiplexer. Moreover, Choudhary et al. [32] showed that an \( n \)-bit majority voter can be built from an \( (n - 2) \)-bit voter together with extra logic gates as illustrated in Figure 3. While these suggestions have shown that a complex majority voter can be designed in a hierarchical fashion, they still pose multiple gate levels on the critical path, leading to a high latency on DBI encoding.

![Figure 3. Hierarchical approach in [32] for 9-bit majority voter design.](image)

In order to address the long-latency issue of DBI encoder, we propose two new voters comprising only two gate levels. The first gate level contains AND-OR-INVERTER (AOI) gates in parallel with the aim of reducing four adjacent input bits to a single output bit, i.e., 4:1 compression. In the second gate level, each proposed voter uses either a NAND gate or an AOI gate to effectively approximate the true majority voter.
3. Proposed Designs
3.1. Basic Idea

Let MAJ\textsubscript{n} be the majority voter on \( n \)-bit inputs. Say that the \( n \) inputs satisfy adjacency condition if at least an adjacent pair of inputs is both 1, that is, both \( i \)th and \( i + 1 \)th inputs are 1, for some \( i = 0, 1, \ldots, n - 2 \). If the majority voter outputs 1, then it is very likely that the adjacency condition holds. When \( n \) is odd, it is precisely the necessary condition of the majority. When \( n \) is even, there is only one input pattern 1010 \ldots 101, which betrays the case. In fact, if we name the inputs cyclically so that 0th and \( n - 1 \)th bits are also adjacent, then the adjacency condition becomes precisely the necessary condition for the majority, regardless of the parity of \( n \). Although it is only a necessary condition, we take advantage of this observation to approximate the majority voter.

Consider the logical OR of two AND gates in Figure 4a, and call it AND-OR pattern detector. This simple circuit approximates MAJ\textsubscript{4}, the majority voter on four bits as shown in Figure 4b; they differ on 0011 and 1100 as shown in Figure 4c. Note that the AND-OR pattern detector outputs 1 if and only if both 2\textsuperscript{th} and \( 2i + 1 \)th bits are 1 for some \( i = 0, 1 \). Thus, it can be regarded as a partial detector of the adjacency condition on 4-bit inputs, and it is a more accurate approximation of the majority voter as shown in Figure 4c.

![Diagram](image)

**Figure 4.** AND-OR pattern detector versus 4-bit majority voter. (a) the logical OR of two AND gates; (b) the majority voter on four bits; (c) comparisons between majority voter, AND-OR gate, and adjacency condition.

3.2. Proposed Encoders

Figure 5 shows two proposed encoders that employ approximate voters, comprised in two gate levels, thus achieving lower latencies. They exploit the simplicity of AND-OR pattern detector. The first one in Figure 5a, comprised of two AOI gates, receives an eight-bit input from the difference between \( v(t - 1) \) and \( u(t) \), ignoring the last one among the nine bits of the original majority voter for simplicity’s sake, and breaks them into two groups, each being fed into the AND-OR patterns. The second level is simply a single NAND gate that sets the control bit to one if any of the two AND-OR patterns is detected. This encoder, termed AOI-NAND ENC, predicts that the number of bit toggles between \( v(t - 1) \) and \( u(t) \) is greater than four if both 2\textsuperscript{th} and \( 2i + 1 \)th bits toggle, for some \( i = 0, 1, 2, 3 \). Although this prediction is not always accurate, this encoder still leads to 9\% of switching activity reduction on average for random data while achieving a lower encoding latency.

The second proposed encoder, named AOI-AOI ENC, includes more circuits for higher reduction in switching activity. However, it still maintains two gate levels as shown in Figure 5b. The first gate level has four AOI gates in parallel where two of the gates receive
which indicates that there are four-bit toggles, and both the approximate voters output whichever form it is encoded. That is, the functional correctness of the proposed encoders.

The voter $f_b$ ignores the input bit $x_8$ since the corresponding circuit accepts only the eight input bits $x_7 \ldots x_0$, that is,

$$f_b \left( 0x_7 \ldots x_0 \right) = f_b \left( 1x_7 \ldots x_0 \right),$$

for any 8-bit patterns $x_7 \ldots x_0$. Among the 512 possible input patterns, $f_a$ agrees with the majority function on 386 inputs, about 75.4% of all the possible inputs, and $f_b$ agrees on 401 inputs, about 78.3%. That is, $f_b$ approximates better the majority function than $f_a$. Given an encoder as in Figure 1 with a Boolean function that determines the control bit, in place of the majority voter, a minimum switching activity is achieved when the majority function is used [12]. Thus, we can expect that approximations $f_a$ and $f_b$ result in more switching activities, and $f_b$, a better approximation, results in less switching activities than $f_a$.

Note also that the majority function is unbiased in the sense that it outputs 0 and 1 with the same probability 0.5 on random input patterns. In other words, the resulting control bit $f \left( x_8x_7 \ldots x_0 \right)$ is 0 on 256 input patterns and 1 on the other 256 input patterns. The approximate voter $f_a$ is biased toward 1; it makes the control bit 1 on 350 input patterns. The voter $f_b$ is biased toward 0; its value is 1 on 185 input patterns.

5. Functional Correctness of the Proposed Encoders

Even though our proposed DBI encoders make errors in the majority vote decision, the bus decoder in the encoders is designed to recover the original data from the codeword whichever form it is encoded. That is, the functional correctness of the proposed encoders is maintained.

Suppose that, for example, a previous codeword $v(t − 1)$ is 001101101 and a data $u(t)$ is 00101111 as shown in Figure 6. Then, the first layer of XOR gates outputs 000110011 which indicates that there are four-bit toggles, and both the approximate voters output
1, a misprediction; the correct majority decision is 0. The misprediction further leads to a codeword $v(t)$ of 11010001 encoded in an inverted form, as shown in Figure 6a, which is opposite to the one encoded by a true majority voter-based DBI, as shown in Figure 6b. Nevertheless, since a control bit indicates whether the codeword is equal to an original data or in an inverted form, the bus decoder restores the codeword in any of two different forms to the same $u(t)$ of 00101111 by performing XOR operations with the control bit. In fact, an arbitrary Boolean function in place of the approximate voters does not affect the correctness of the encoder because of the way the decoder works; only the energy efficiency may suffer.

![Figure 6. Bus decoding with (a) approximate voter and (b) majority voter.](image)

**6. Results**

The proposed encoders were designed and synthesized with a commercial 65-nm process and standard library to analyze the performance in terms of latency, area, and operating power. For comparisons, we also designed three conventional DBI encoders: logic synthesis-based encoder (SYN-ENC) shown in Figure 2, multiplexer-based encoder (MUX-ENC) proposed in [30,31], and hierarchical design (HIE-ENC) based on [32] using the same design methodology.

The evaluation of switching activities of the five encoders was performed on ten million uniformly generated 8-bit random data bits. As a major performance metric, the savings in switching activities were measured compared to direct (non-DBI) data movements. Table 1 summarizes also latencies, areas, and operating powers of the five designs.

| 65-nm Process            | Latency (nsec) | Area ($\mu$m²) | Operating Power (µW) | Savings in $\alpha$ (%) |
|-------------------------|----------------|----------------|-----------------------|-------------------------|
| SYN-ENC [29]            | 0.95           | 95.68          | 46.9                  | 18                      |
| MUX-ENC [30,31]         | 0.91           | 103.68         | 38.0                  | 18                      |
| HIE-ENC [32]            | 1.05           | 151.04         | 72.2                  | 18                      |
| Proposed AOI-NAND ENC   | 0.29           | 48.96          | 14.1                  | 9                       |
| Proposed AOI-AOI ENC    | 0.41           | 56.00          | 16.6                  | 11                      |

Two proposed DBI encoders achieve lower latencies and smaller areas compared to conventional encoders. On the other hand, as a tradeoff, since the approximate voters make errors in majority decision, the proposed encoders show lower performance on
the switching activity savings by 9% and 11%, respectively, compared to 18% for the conventional encoders. However, the degradation in switching activity savings is mitigated again by operating power savings. The proposed encoders require lower powers of 14.1 µW and 16.6 µW, respectively, while the conventional encoders require 38.0 µW, 46.9 µW, and 72.2 µW, respectively.

To compare the designs in terms of operating energy efficiency, the power-delay product (PDP) and the energy-delay product (EDP) were obtained [33,34]. As shown in Figure 7, the proposed encoders outperform the conventional ones.

![Figure 7. PDP and EDP comparisons. Y axes are in log scale.](image)

7. Conclusions

We proposed two new encoders for data bus inversion (DBI), which conventionally uses a majority voter to reduce switching activities in data movement and thus reduces the corresponding energy consumption. We report various experiment data based on 65-nm process implementations, including latencies and powers, regarding the two proposed encoder designs and three conventional ones.

The new encoders employ simpler approximate voters, which are based on the idea that AND-OR pattern detector can approximate the majority and the adjacency on 4-bit inputs. Both approximate voters are comprised in two gate levels. Hence, they improve latency more than twice and the resulting encoders still achieve energy savings compared to direct data movement. Of course, the energy savings is not as much as the conventional DBI design. But we can see that there is a predictable tradeoff between latency and energy savings, and there must be a sweet spot to achieve overall optimality when we design circuits for data movement.

**Author Contributions:** Conceptualization, K.-W.K.; methodology, S.-i.P. and K.-W.K.; software, S.-i.P. and K.-W.K.; validation, S.-i.P. and K.-W.K.; formal analysis, S.-i.P. and K.-W.K.; investigation, S.-i.P. and K.-W.K.; resources, S.-i.P. and K.-W.K.; data curation, S.-i.P. and K.-W.K.; writing—original draft preparation, S.-i.P. and K.-W.K.; writing—review and editing, S.-i.P. and K.-W.K.; visualization, S.-i.P. and K.-W.K.; supervision, S.-i.P. and K.-W.K.; project administration, S.-i.P. and K.-W.K.; funding acquisition, S.-i.P. and K.-W.K. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported in part by Hongik University and in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government under Grant 2016R1A1A1B01016531 and Grant NRF-2019R1G1A1008751. This work was also partly supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korean Government (MSIT) (No. 2019-0-00933, Research on CPU vulnerability detection and validation).

**Conflicts of Interest:** The authors declare no conflict of interest.

**References**

1. Borkar, S.; Chien, A.A. The future of microprocessors. *Commun. ACM* **2011**, *54*, 67–77. [CrossRef]
2. Borkar, S. Role of interconnects in the future of computing. *J. Lightwave Technol.* **2013**, *31*, 3927–3933. [CrossRef]
3. Dally, B. Power, programmability, and granularity: The challenges of exascale computing. In Proceedings of the 2011 IEEE International Test Conference, Anchorage, AK, USA, 16–20 May 2011; p. 12. [CrossRef]
4. Keckler, S.W.; Dally, W.J.; Khailany, B.; Garland, M.; Glasco, D. GPUs and the future of parallel computing. *IEEE Micro* **2011**, *31*, 7–17. [CrossRef]
