electronics

Article

Majority Approximators for Low-Latency Data Bus Inversion

Sung-il Pae and Kon-Woo Kwon *

Department of Computer Engineering, Hongik University, Seoul 04066, Korea
* Correspondence: konwoo@hongik.ac.kr

Abstract: Data bus inversion (DBI) is an encoding technique that saves power in data movement in which the majority function plays an essential role. For a latency optimization, the majority function can be replaced by a majority approximator that allows for a small error in majority voting to obtain a faster encoder that still saves power. In this work, we propose two systematic approaches for finding high-performance majority approximators. First, we perform an exhaustive search of all possible Boolean functions to find an optimal approximator based on a certain circuit structure comprised of fifteen logic gates. The approximator found by the systematic search can be implemented using compound gates, resulting in a latency-efficient design with only two gate levels. Compared with prior works using a heuristic idea, the proposed circuit runs at the same speed but achieves greater switching activity savings. Second, we propose another majority approximator using the average of three randomly permuted copies of the approximator found in the first approach. We show that the second proposed approximator achieves even higher savings in switching activity as its function is closer to a true majority voter. We report various performance metrics of the newly found majority approximators based on syntheses using a 65 nm process.

Keywords: approximator; data bus inversion; latency; majority function; switching activity

Citation: Pae, S.-i.; Kwon, K.-W. Majority Approximators for Low-Latency Data Bus Inversion. *Electronics* **2022**, *11*, 3352. https://doi.org/10.3390/electronics11203352

Academic Editors: Juan M. Corchado, Byung-Gyu Kim, Carlos A. Iglesias, In Lee, Fuji Ren and Rashid Mehmood

Received: 30 September 2022
Accepted: 14 October 2022
Published: 17 October 2022

1. Introduction

Data bus inversion (DBI) is an encoding technique that saves power in data movement [1–7]. Let $u(t) = (u_0(t), \ldots, u_7(t))$ be 8-bit data to move at time t, and consider a 9-bit encoding $v(t) = (v_0(t), \ldots, v_8(t))$ of the data $u(t)$, which is computed by the encoder given in Figure 1.

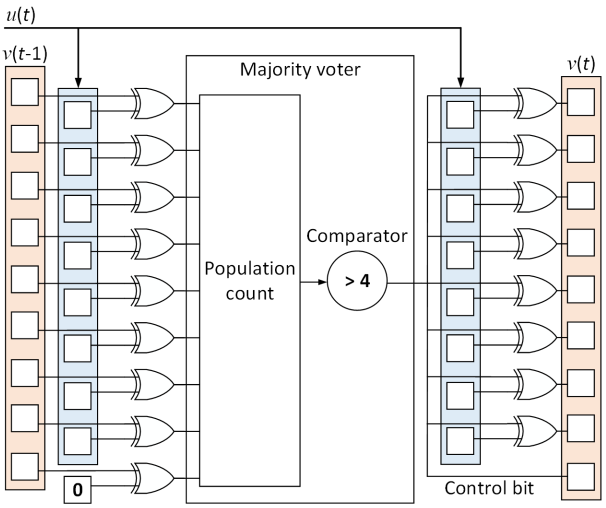

Figure 1. A DBI encoder.

The encoder first computes

$$x(t) = (v_0(t-1), \ldots, v_7(t-1), v_8(t-1)) \oplus (u_0(t), \ldots, u_7(t), 0), \tag{1}$$

the bitwise XOR of the previous encoding $v(t-1)$ and the current data $u(t)$ with 0 appended at the end. The first 8 bits of $x(t)$ are the bit toggles between the previous encoding and the current data. Now, the toggle bits $x(t)$ are fed to the majority voter MAJ and

if MAJ$(x(t)) = 1$, then, (if $x(t)$ has more than four 1's,)
  $v_8(t) = 1$, and, $v_i(t) = \overline{u_i(t)}$, for $0 \le i \le 7$, (set control bit to 1 and invert)
otherwise,
  $v_8(t) = 0$, and, $v_i(t) = u_i(t)$, for $0 \le i \le 7$. (set control bit to 0)

The last bit $v_8(t)$ of the encoding $v(t)$ is the *control bit* that contains the information on whether the data are inverted or not, and the majority function determines the control bit based on the number of toggles.

We are interested in the number of bit toggles that occur *on the bus*, as it is proportional to the energy dissipation through charging and discharging the interconnect capacitance, which is known to be a substantial portion of the energy consumed by digital circuits [8–13].

If the DBI encoding is not used, then the data $u(t)$ are directly sent to a 8-bit bus. If DBI encoding is used, then we assume the encoded data $v(t)$ are sent to a 9-bit bus. The above encoding reduces the average number of bit toggles even though it has the extra control bit, and we can see this as follows. If the data $u(t)$ are uniformly random, then the bit toggles are random, and thus the total amount τ_u of bit toggles over all possible 8-bit values is

$$\tau_u = 0\binom{8}{0} + 1\binom{8}{1} + \cdots + 8\binom{8}{8} = 8 \cdot 2^7 = 1024.$$

(Take the derivative of $(y+1)^n = \sum_{k=0}^{n} \binom{n}{k} y^k$, and obtain $n(y+1)^{n-1} = \sum_{k=0}^{n} k\binom{n}{k} y^{k-1}$.)

If the data $u(t)$ are uniformly random, we can show that the encoding $v(t)$ becomes uniformly random quickly. The total amount τ_v of bit toggles over all possible 9-bit values of $x(t)$ is

$$\tau_v = 0\binom{9}{0} + \cdots + 4\binom{9}{4} + 4\binom{9}{5} + \cdots + 0\binom{9}{9} \tag{2}$$

$$= 2\left(0\binom{9}{0} + \cdots + 4\binom{9}{4}\right) = 1674. \tag{3}$$

Now, to make a proper comparison, we need the *average* amounts of toggles per data movement, which are

$$\tau_u / 2^8 = 1024/2^8 = 4,$$
$$\tau_v / 2^9 = 1674/2^9 \approx 3.270.$$

So, the encoding reduces the number of bit toggles by 18.3% on average, when compared to the direct data movement.

Latency-Optimized DBI Encoding

In the above encoding scheme, replace MAJ by an arbitrary Boolean function f on 9-bit inputs. The resulting encoding still functions correctly; if the control bit was set to 1, then the inverted data are recovered by decoding. However, the number of toggles can increase. For example, if f is identically 0, then the encoder passes the original data as it is with the control bit 0, and the total number of toggles over all input patterns is the same as the direct data movement with 8-bit bus.

In the following, we are concerned with a DBI encoder for 8-bit data that use a Boolean function f in place of MAJ. Let $\tau(f)$ be the total sum of toggles on all possible 9-bit patterns for encoding the corresponding DBI encoder and let us call it *total switching activity* for f.

The total switching activity is minimized when f is MAJ. For each input, the majority function gives a minimal switching and thus the total switching activity over all inputs is minimal. For the toggle bits x of (1), let $\mathrm{wt}(x)$ be its Hamming weight. Note that the switching activity is either $\mathrm{wt}(x)$ or $9 - \mathrm{wt}(x)$. More precisely,

$$\text{switching activity} = \begin{cases} \mathrm{wt}(x) & \text{if } f(x) = 0 \\ 9 - \mathrm{wt}(x) & \text{if } f(x) = 1 \end{cases} \tag{4}$$

If $\mathrm{wt}(x) \leq 4$, then the switching activity is minimized if and only if $f(x) = 0$. If $\mathrm{wt}(x) > 4$, then the switching activity is minimized if and only if $f(x) = 1$. Such a Boolean function f is exactly the majority function MAJ, and this proves our claim that the total switching activity is minimized when f is MAJ.

However, the majority function is rather complicated to implement as a circuit and thus increases the latency of data transmission (of course, in favor of power saving). In [14], the authors replaced it with simpler approximate circuits to decrease the latency at the cost of switching activity. Let us call a Boolean function f that replaces majority function in DBI encoder a *majority approximator*. As observed above, even if a majority approximator f makes an error in majority voting, the data are transmitted correctly. If we come up with a low-latency circuit that is close enough to majority function, then we can make a more quickly running DBI encoder and thus a more quickly running system, while still saving power for data transmission. Furthermore, a lower latency itself can save energy by saving the time and thus the energy to sustain the system.

In this paper, we aim to find low-latency majority approximators that result in as low switching activity as possible. We have seen $\tau(\mathrm{MAJ}) = 1674$ in (3). If MAJ is replaced by an approximate voter f, by the above reasoning in (4), for x such that $f(x) \neq \mathrm{MAJ}(x)$, the additional switching activity compared to the correct DBI encoder is $|2\mathrm{wt}(x) - 9|$. So, the total switching activity with respect to f,

$$\tau(f) = \tau(\mathrm{MAJ}) + \sum_{f(x) \neq \mathrm{MAJ}(x)} |2\mathrm{wt}(x) - 9|.$$

Let us call

$$\delta(f) = \tau(f) - \tau(\mathrm{MAJ}) = \sum_{f(x) \neq \mathrm{MAJ}(x)} |2\mathrm{wt}(x) - 9|$$

additional total switching activity of f, and we use this as a performance measure for a majority approximator. Given a Boolean function f, it is often easier to compute the additional total switching activity of f than the total switching activity. Of course, when f is close to the majority function, that is, its Hamming distance to the majority function is small, its additional total switching activity tends to be small. In our search for good majority approximators, we use the two measures, the Hamming distance and the additional total switching activity.

Since DBI was first proposed by [1], many researchers have presented bus coding schemes in various forms for energy-efficient data movement. Shin et al. [15] proposed a scheme named partial DBI in which the DBI coding is applied to a subset of total bus lines to reduce the energy waste caused by encoding inactive bus lines. Kwon [7] proposed a modified DBI scheme suitable for OR-chained buses that require extra switching activity for data parking. Lee et al. [16] proposed a scheme named dynamic bus inversion dedicated for a pseudo open drain interface in which sending the logical value 1 consumes more energy than sending the logical value 0. Based on the observation that there exists a similarity between multiple data elements sent in a single transaction, the authors in [16] proposed XOR-based encoding such that the number of logical 1's can possibly reduce in the code word. Song et al. [17] proposed using a bus in idle cycles as a resource for

sparse bus encoding that can outperform the DBI at the expense of many extra bits of overhead. Pae et al. [14] proposed low-latency DBI encoders by replacing a true majority voter with a majority approximator. Specifically, two majority approximators F_A and F_B were suggested and implemented as circuits with significantly lower latencies than the majority function, while among the $2^9 = 512$ inputs, they create error for 126 and 111 inputs, respectively. They were devised using the heuristic idea that an input is likely to satisfy so-called adjacency condition if the input has more 1's than 0's. In this paper, we aim to find better majority approximators through systematic searches of certain classes of circuits.

The rest of this paper is organized as follows. Section 2 describes an exhaustive search on all possible Boolean functions based on the circuit scheme that imitates the structure of the approximate voters in the previous work [14]. As a result, we could come up with majority approximators, which we call F_0 and F_1, with a similar latency but with less errors (93). Section 3 describes a search on circuits made by parallelly combining three of the newly found approximators F_0 but randomly rearranging inputs, and then by taking the majority of the three outputs, we came up with three new approximators with only 39 errors. In this case, the search is not exhaustive because the search space is too huge, and therefore there is no guarantee that they are the best approximators. Section 4 reports the relevant metrics including the latencies, areas, operating powers, and switching activities of the circuits based on syntheses using a commercial 65 nm process. Section 5 concludes the paper, and Appendix A presents a table of abbreviations and symbols.

2. Search for Majority Approximators

Consider the schematic shown in Figure 2. Substituting the fifteen rectangles with appropriate logic gates g_i, we obtain a circuit with nine inputs. In a functional form, the resulting Boolean function is

$$\begin{aligned} f(x) = & g_{15}(g_{13}(g_9(g_1(x_0, x_1), g_2(x_2, x_3)), g_{10}(g_5(x_1, x_2), g_6(x_3, x_4))), \\ & g_{14}(g_{11}(g_3(x_4, x_5), g_4(x_6, x_7)), g_{12}(g_7(x_5, x_6), g_8(x_7, x_8))). \end{aligned} \tag{5}$$

For example, consider NAND and NOR as candidates for the gates. Then, there exist $2^{15} = 32,768$ possibilities, and we can make comparisons with the majority function to find the best approximators among them. As a result, we found six circuits, shown in Table 1, with a Hamming distance of 93 to the majority function, which is the minimum among the possible circuits. As the next-best circuits, there are two functions with a distance of 94. Furthermore, then there are four circuits with a distance of 96, etc. A similar exhaustive search was performed with respect to the additional total switching activity. Figure 3 shows the tally counts of the functions with respect to the number of errors and the additional total switching activity.

Table 1. The six circuits that are nearest to majority function using the search scheme in Figure 2. For the gates $g_1, \ldots, g_{15}$, 0 stands for NAND and 1 stands for NOR.

f	$(g_1, g_2, \ldots, g_{15})$
F_0	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
F_1	(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
F_2	(0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0)
F_3	(0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0)
F_4	(1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1)
F_5	(1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1)

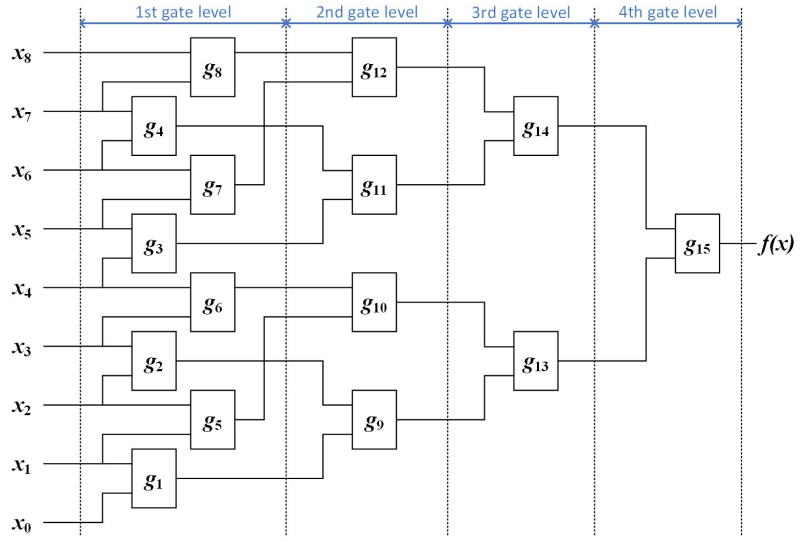

Figure 2. A search scheme for majority approximators.

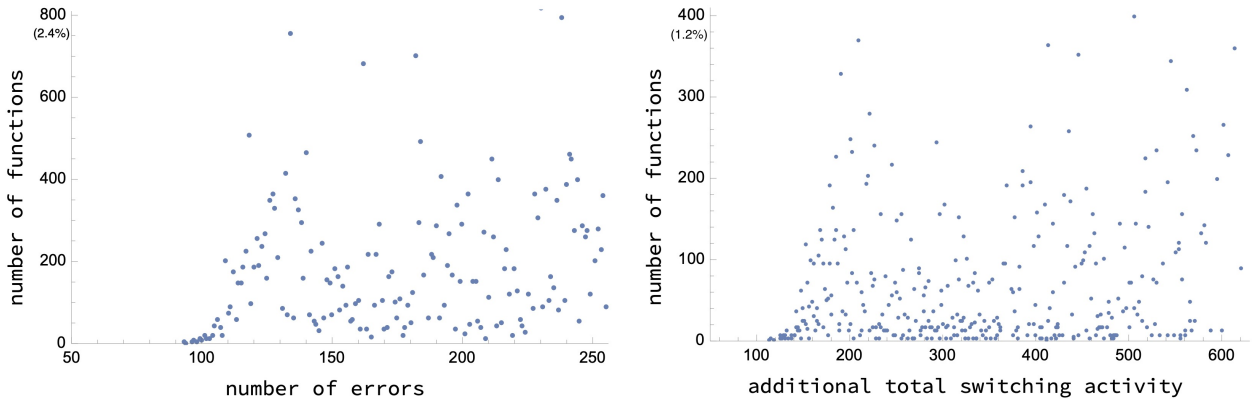

Figure 3. Tally counts according to the number of errors to MAJ and the additional total switching activity, among $2^{15} = 32,768$ possible circuits using NAND and NOR gates; test.

We have chosen the scheme presented in Figure 2 because the approximate voters F_A and F_B proposed in the previous work [14] used a similar structure that could exploit the adjacency condition, especially the better approximator F_B. In fact, using the convention of Table 1, the approximator F_B can be represented as $(0, 0, 0, 0, 0, 0, 1, 1, 1)$.

Among the six circuits in Table 1, F_0 and F_1 stand out because they have all the same gate for g_i. In fact, in terms of switching activity, they are the two best approximators among $2^{15} = 32,768$ circuits. The other four circuits in Table 1 have slightly higher switching activities, although they have the same number of errors. See Table 2 for the specific numbers.

Table 2. Hamming distances and additional switching activities to the majority function.

Approximator f	Hamming Distance to MAJ	Additional Total Switching Activity
F_A	126	198
F_B	111	161
F_0	93	113
F_1	93	113
F_2 to F_5	93	115

Recall that F_A and F_B from [14] have Hamming distances of 126 and 111, respectively, and thus, F_0 and F_1 are more accurate majority approximators. Even with more extensive searches on various combinations of two or three gates among NAND, NOR, AND, and OR, F_0 and F_1 are found to be the best approximators in terms of both Hamming distance and switching activity. For example, when NAND, AND, and OR gates are candidates, there are $3^{15} = 14,348,907$ possibilities (see Figure 4). An exhaustive search found 75 circuits with the same Hamming distance of 93, as with F_0 and F_1. Among them, 45 circuits have the same additional total switching activity of 113 as with F_0 and F_1, and those are all the circuits with the minimum additional total switching activity. Interestingly, those 45 circuits are all equivalent to either F_0 or F_1.

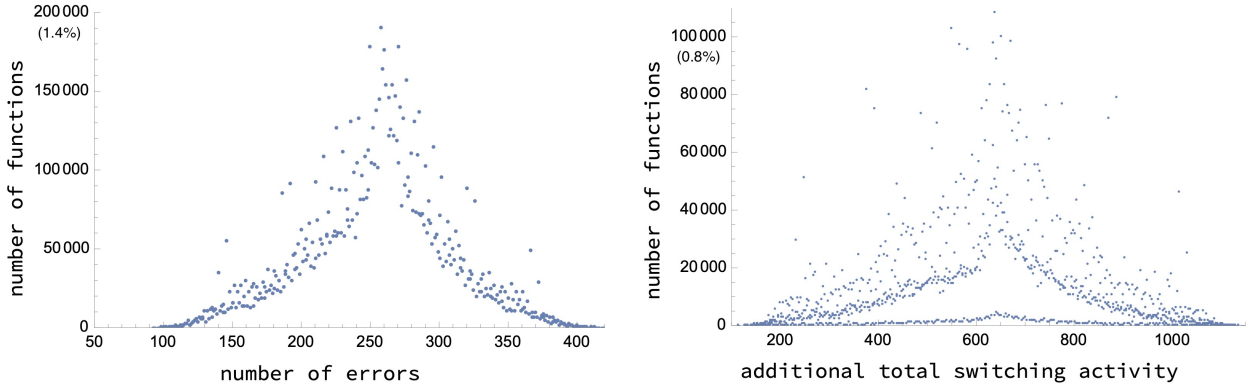

Figure 4. Tally counts according to the number of errors to MAJ and the additional total switching activity among $3^{15} = 14,348,907$ possible circuits using NAND, AND, and OR gates.

The Boolean function F_0 can be implemented as a low-latency circuit using the idea used in [14] (see Figure 5). We start with the four-level circuit (a) with all NAND gates. By De Morgan's law, replace the NAND gates with AND gates at the first and third levels and with OR gates at the second and fourth levels. This equivalent circuit (b) can be viewed as a majority approximator comprising five AND–OR pattern detectors.

For a latency-efficient design, NOR gates are preferred over the OR gates in the second level of (b) because a NOR of two AND gates can be implemented as a single compound AND-OR-INVERTER (AOI) gate. At each wire connecting the second and third levels, we add two bubbles, of which one is used for converting OR to NOR in the second level while the other is pushed to the third level as shown in (c). The pushed bubbles are combined with two ANDs and one OR gate, resulting in a single compound gate of OR-AND-INVERTER (OAI). Circuit (d) now consists of only two gate levels: the first stage contains four AOI gates in parallel and the second one contains an OAI gate. In general, AOI and OAI gates are available as a standard cell that includes only two transistors in series for both low-to-high and high-to-low transitions as shown in Figure 6 (see that a simple NAND gate also needs two transistors in series for a high-to-low transition).

Similarly, F_1 can be implemented as a low-latency circuit using the compound gates as shown in Figure 7. Four levels of NOR gates (a) are replaced by appropriate gates that result in an equivalent circuit (b), and the shaded regions can be regarded as OAI and AOI gates.

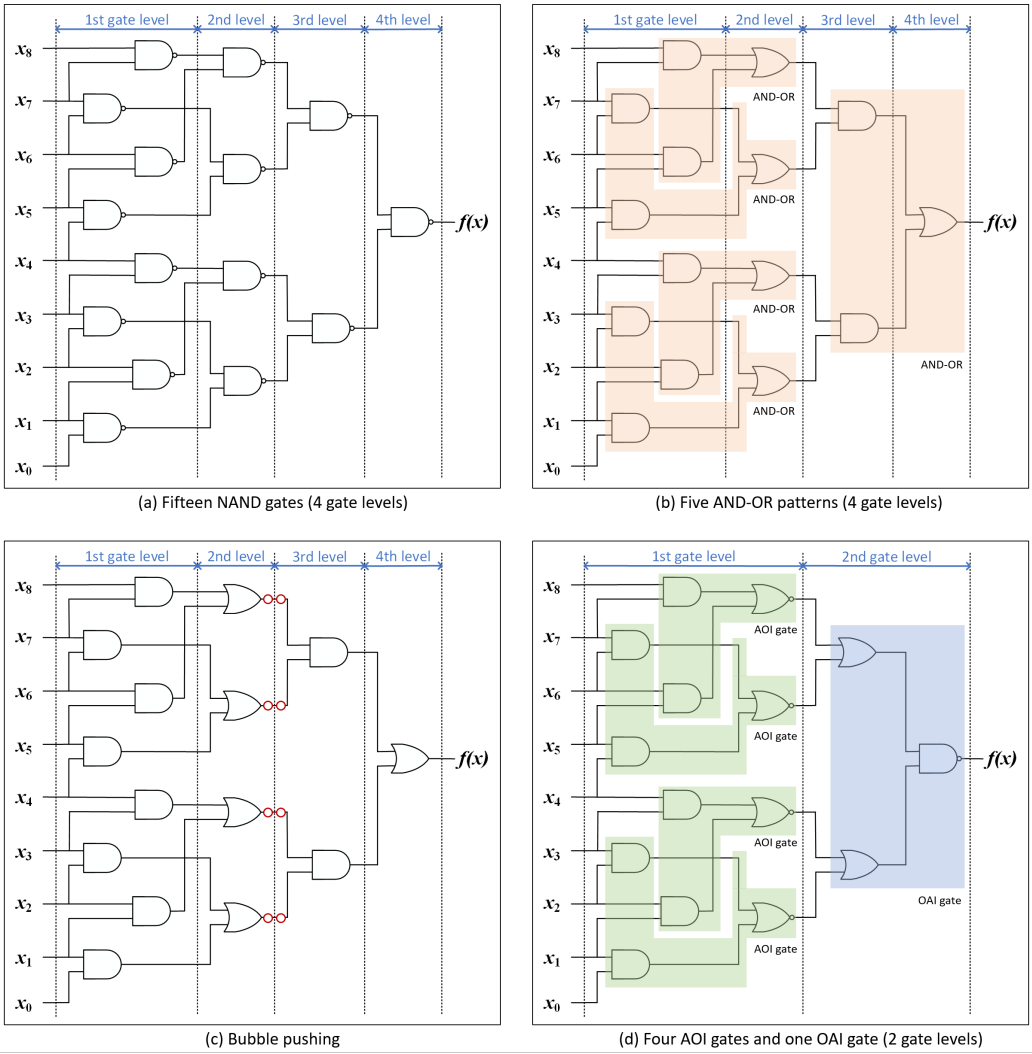

Figure 5. A low-latency implementation of F_0.

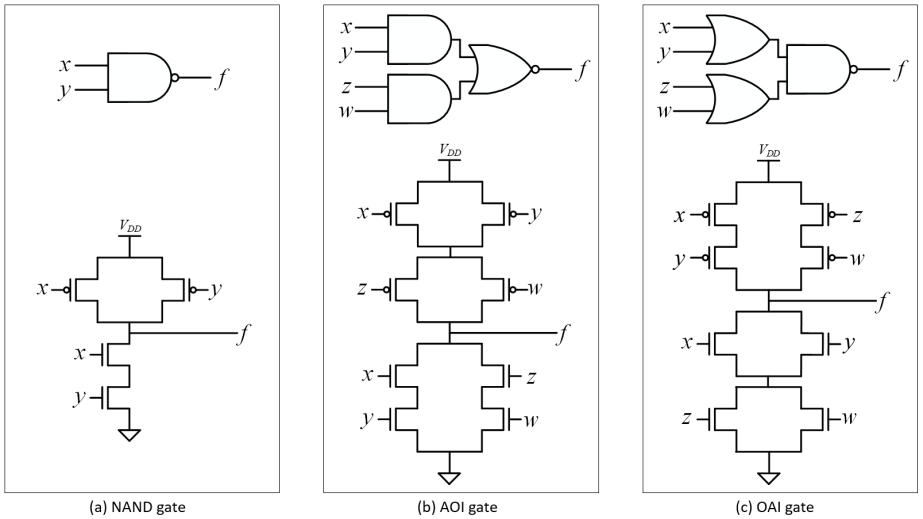

Figure 6. An efficient compound implementation of AOI gate and OAI gate.

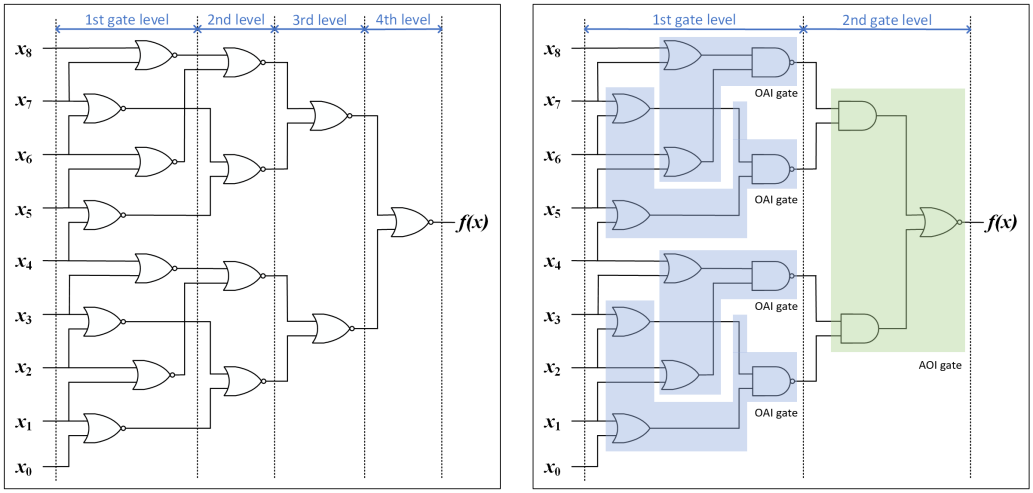

Figure 7. A low-latency implementation of F_1.

3. Averaging Approximators

Let f_1, f_2 and f_3 be majority approximators that make errors independently, in the sense that, for $i \neq j$,

$$\Pr(f_i \text{ makes an error} \mid f_j \text{ makes an error}) = \Pr(f_i \text{ makes an error}),$$

and assume that the three functions all create errors on 18% of the inputs. Consider the function

$$f(x) = \mathrm{MAJ}(f_1(x), f_2(x), f_3(x)).$$

This function takes a 9-bit input x, feeds it into three approximate voters, and then takes a majority vote of their outputs. The function f makes an error if and only if at least two of the f_is make errors. Let E_i be the event that f_i makes an error, and let E be the event that f makes an error. Then,

$$E = (E_1 \cap E_2) \cup (E_2 \cap E_3) \cup (E_3 \cap E_1),$$

and

$$\Pr(E) = \Pr(E_1 \cap E_2) + \Pr(E_2 \cap E_3) + \Pr(E_3 \cap E_1) - 2\Pr(E_1 \cap E_2 \cap E_3).$$

By the independence, for $i \neq j$, $\Pr(E_i \cap E_j) = \Pr(E_i)\Pr(E_j) = (0.18)^2$, and similarly, $\Pr(E_1 \cap E_2 \cap E_3) = (0.18)^3$. Therefore,

$$\Pr(E) = 3 \cdot (0.18)^2 - 2(0.18)^3 = 0.085536.$$

So, the function f is a better approximator that makes only about the half as many errors as f_is. Although the three approximate voters can make errors, by taking their majority, we expect that the possible errors are averaged out. This idea holds if the constituent functions f_i make relatively small errors. Let us call such an approximate voter f an averaging approximator.

Using this idea, consider now the following approximator:

$$f(x) = \mathrm{MAJ}(F_0(x), F_0(\pi_1(x)), F_0(\pi_2(x))), \tag{6}$$

where π_1 and π_2 are permutations of nine bits of $x = (x_0, \ldots, x_8)$. We have a good candidate for the constituent function for an averaging approximator, namely F_0 and F_1. However, we need three functions that are stochastically independent.

For a Boolean function g and a permutation π, in general, $g(x) \neq g'(x) := g(\pi(x))$. However, a global metric such as the Hamming distance to the majority function does

not change by permutation of the inputs. For the permutations π_1 and π_2, let $F_0'(x) = F_0(\pi_1(x))$, $F_0''(x) = F_0(\pi_2(x))$. Then, F_0' and F_0'' have the same performance as F_0 as approximate voters, in the sense that they have the same Hamming distance to the majority function and the additional total switching activity as F_0. Moreover, by taking random permutations, we expect that the three functions F_0, F_0', and F_0'' are relatively independent, if not strictly independent as we assumed before.

Figure 8 shows a tally count of the Hamming distances for a million approximate voters (6) obtained by generating pairs of random permutations. The Hamming distance ranges from 42 to 94, and the frequencies are distributed in a bell shape similar to a normal distribution. The distribution peaks at 66 and thus we can expect that a randomly chosen averaging approximator is likely to have a Hamming distance close to 66 with a high probability. This number of errors is about 12.9% of all 512 inputs. This result does not exactly agree with the approximately 9% that we made above assuming the strict independence, and it is probably because F_0, F_0', and F_0'' are not strictly independent (note that F_0 makes an error on about 18% of the inputs). However, this experiment results in approximators with a wide range of performances and shows a possibility of finding a good approximator.

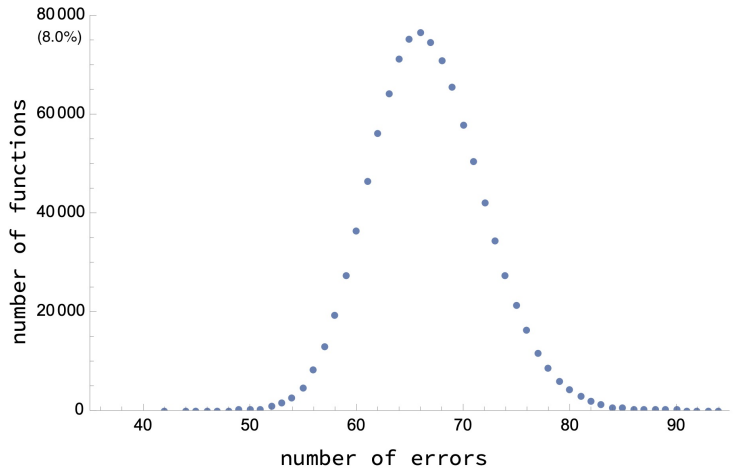

Figure 8. A tally count according to the number of errors to MAJ among the averaging approximators using one million random pairs of permutations.

The search was performed to find the two permutations that result in a majority approximator (6) that is as close to the majority function as possible. Note that there are $9! \times 9! = 131,681,894,400$ possibilities, which is too huge to make an exhaustive search. Instead, we generated a few hundred million pairs of random permutations, and then measured how close the resulting approximator (6) is to the majority function. As a result, we found three averaging approximators with a Hamming distance of 39, or a 7.6% error rate, and that was as close as we could obtain.

One of the three best averaging approximators that we found was determined by the following permutations:

$$\begin{aligned} \pi_1 &= (x_4, x_3, x_5, x_8, x_0, x_6, x_2, x_1, x_7), \\ \pi_2 &= (x_1, x_5, x_8, x_2, x_4, x_7, x_6, x_3, x_0). \end{aligned} \tag{7}$$

Let us call the corresponding approximate voter F_M, and it can be implemented as in Figure 9 where three copies of F_0 with appropriate permutations operate in parallel then their majority is chosen by using four NAND gates. The use of more circuits than F_0 or F_1 increases the latency of F_M. However, as will be discussed in Section 4, our implementation based on F_M still runs much faster than a true majority voter-based implementation.

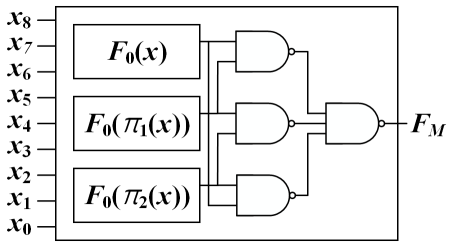

Figure 9. Implementation of F_M.

4. Implementation and Performance Metrics

The proposed encoders based on F_0, F_1, and F_M were designed in Verilog HDL and synthesized using standard cells in commercial 65 nm CMOS process technology to analyze latency, area, and operating power. For fair comparisons, two prior designs based on F_A and F_B as well as another two conventional DBI encoders were also resynthesized using the same CMOS technology. For the seven different DBI encoders, savings in switching activity with respect to a direct data movement were evaluated on an input sequence of ten million uniformly random 8-bit data.

Table 3 summarizes the performance of the seven DBI encoders. The encoders based on true majority voters, namely, SYN-ENC [14,18] and MUX-ENC [19,20], achieve the highest savings in switching activity of 18.3% at the expense of 0.91 ns or higher latency. Prior works on approximators based on F_A and F_B improved latencies significantly, however, as a trade-off, they showed lower savings in switching activity of 9.5% and 11.2%, respectively. Our first proposed encoders based on F_0 and F_1 enhance the trade-off provided by F_B, that is, they achieve 12.8% savings in switching activity without incurring any penalty in the encoding latency. The enhancement stems from the systematic search scheme as opposed to the heuristic solution of F_B. Our second proposed encoder based on F_M achieves 16.4% savings in switching activity, the best among five approximate approaches. Due to the use of majority function on three outputs from F_0, F_0', and F_0'', the F_M-based design requires 0.53 ns latency which is longer than other approximators; however, this is still almost half the encoding latency of the true majority voter-based designs.

Table 3. Latency, area, power, and savings in switching activity of the encoders based on the proposed approximators and the majority voters.

65 nm Process	Latency (ns)	Area (μm^2)	Operating Power (μW)	Savings in Total Switching Activity (%)
SYN-ENC [14,18]	0.95	95.68	46.9	18.3
MUX-ENC [19,20]	0.91	103.68	38.0	18.3
F_A [14]	0.29	48.96	14.1	9.5
F_B [14]	0.35	65.28	26.2	11.2
Proposed F_0	0.35	65.28	26.7	12.8
Proposed F_1	0.35	65.28	26.4	12.8
Proposed F_M	0.53	113.60	65.2	16.4

We also obtained the power-delay product (PDP) and the energy-delay product (EDP) as shown in Figure 10 to compare the seven encoders in terms of operating energy efficiency [21,22]. Obviously, the five approximate designs outperform the three conventional designs based on a true majority voter, e.g., even the F_M-based design shows a significantly lower EDP than the conventional designs.

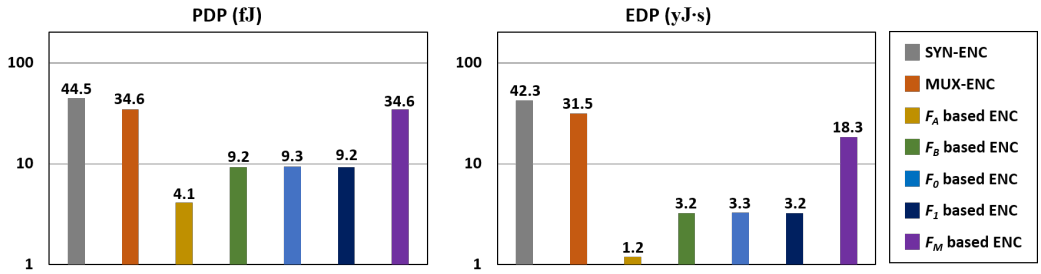

Figure 10. PDP and EDP comparisons. Y axes are in log scale.

5. Conclusions

We proposed two search methodologies for latency-efficient majority approximators that can replace the true majority function for 9-bit inputs in the DBI encoder for 8-bit data. First, motivated by our previous work in [14], we performed an *exhaustive* search on a class of circuits based on a structure of the approximate voter in [14] and found two new majority approximators F_0 and F_1 that are optimal in terms of the total switching activity. F_0 and F_1 can be implemented using compound gates of AOI and OAI, resulting in only two gate levels. The proposed encoders based on F_0 and F_1 improve the switching activity without incurring any extra latency compared to the previous work using a heuristic idea. Second, we considered another class of majority approximators made by averaging three input-permuted copies of F_0. Since the class is too large to perform an exhaustive search, we randomly picked a few hundred million such approximators and found a good majority approximator, namely, F_M. The appropriately chosen permutations by our systematic search achieve a functionality close to the true majority function, leading to higher savings in switching activity than F_0 and F_1.

The proposed encoders were designed and synthesized using standard cells in 65 nm CMOS technology. The F_0- and F_1-based encoders both achieved 12.8% savings in switching activity, compared to 11.2% for the previous encoder with the same encoding latency of 0.35 ns. The F_M achieved 16.4% savings by averaging out the possible errors produced by constituent approximators. Although the F_M is designed with more circuits than F_0 and F_1, it is still latency efficient at two compound gate levels in addition to two NAND gate levels, requiring about half the latency of true majority function-based DBI encoders.

Author Contributions: Conceptualization, S.-i.P. and K.-W.K.; methodology, S.-i.P. and K.-W.K.; software, S.-i.P. and K.-W.K.; vali-dation, 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 NRF-2016R1D1A1B01016531 and Grant NRF-2022R1F1A1074728. 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. 2019-0-00533, Research on CPU vulnerability detection and validation).

Data Availability Statement: Not applicable

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

Appendix A

Table A1. List of Abbreviations and Symbols

Abbreviation or Symbol	Definition
DBI	data bus inversion encoding; Section 1
MAJ	the majority function; Section 1
$\tau(f)$	total switching activity for f; Section 1
$\delta(f)$	additional total switching activity for f; Section 1
F_A, F_B	majority approximators proposed in [14]; Section 2
F_0, F_1	best majority approximators by scheme of Figure 2; Table 1
AOI	AND-OR-INVERTER; Figure 6
OAI	OR-AND-INVERTER; Figure 6
F_M	an averaging approximator based on F_1; Figure 9
SYN-ENC	a majority function implementation proposed in [18]
MUX-ENC	a majority function implementation proposed in [19,20]

References

1. Stan, M.R.; Burleson, W.P. Bus-invert coding for low-power I/O. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **1995**, *3*, 49–58. [CrossRef]
2. Stan, M.R.; Burleson, W.P. Low-power encodings for global communication in CMOS VLSI. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **1997**, *5*, 444–455. [CrossRef]
3. Bae, S.J.; Park, K.I.; Ihm, J.D.; Song, H.Y.; Lee, W.J.; Kim, H.J.; Kim, K.H.; Park, Y.S.; Park, M.S.; Lee, H.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. *IEEE J. Solid-State Circuits* **2008**, *43*, 121–131. [CrossRef]
4. Hollis, T.M. Data bus inversion in high-speed memory applications. *IEEE Trans. Circuits Syst. II Express Briefs* **2009**, *56*, 300–304. [CrossRef]
5. Lucas, J.; Lal, S.; Juurlink, B. Optimal DC/AC data bus inversion coding. In Proceedings of the 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 19–23 March 2018; pp. 1063–1068.
6. Sohn, K.; Na, T.; Song, I.; Shim, Y.; Bae, W.; Kang, S.; Lee, D.; Jung, H.; Hyun, S.; Jeoung, H.; 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. *IEEE J. Solid-State Circuits* **2012**, *48*, 168–177. [CrossRef]
7. Kwon, K.W. Optimal bus coding for OR-chained buses. *IEICE Electron. Express* **2021**, *18*, 20200378. [CrossRef]
8. Borkar, S.; Chien, A.A. The future of microprocessors. *Commun. ACM* **2011**, *54*, 67–77. [CrossRef]
9. Borkar, S. Role of interconnects in the future of computing. *J. Light. Technol.* **2013**, *31*, 3927–3933. [CrossRef]
10. Dally, B. Power, programmability, and granularity: The challenges of exascale computing. In Proceedings of the 2011 IEEE International Test Conference, Anaheim, CA, USA, 20–22 September 2011; p. 12.
11. 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]
12. Kestor, G.; Gioiosa, R.; Kerbyson, D.J.; Hoisie, A. Quantifying the energy cost of data movement in scientific applications. In Proceedings of the 2013 IEEE International Symposium on Workload Characterization (IISWC), Portland, OR, USA, 22–24 September 2013; pp. 56–65.
13. Lucas, R.; Ang, J.; Bergman, K.; Borkar, S.; Carlson, W.; Carrington, L.; Chiu, G.; Colwell, R.; Dally, W.; Dongarra, J.; et al. *Doe Advanced Scientific Computing Advisory Subcommittee (Ascac) Report: Top Ten Exascale Research Challenges*; Technical report; Office of Science, U.S. Department of Energy: Washington, DC, USA: 2014.
14. Pae, S.i.; Kwon, K.W. Latency-Optimized Design of Data Bus Inversion. *Electronics* **2022**, *11*, 1205. [CrossRef]
15. Shin, Y.; Chae, S.I.; Choi, K. Partial bus-invert coding for power optimization of application-specific systems. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2001**, *9*, 377–383. [CrossRef]
16. Lee, D.; O'Connor, M.; Chatterjee, N. Reducing Data Transfer Energy by Exploiting Similarity within a Data Transaction. In Proceedings of the 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA), Vienna, Austria, 24–28 February 2018; pp. 40–51. [CrossRef]
17. Song, Y.; Ipek, E. More is less: Improving the energy efficiency of data movement via opportunistic use of sparse codes. In Proceedings of the 2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Waikiki, HI, USA, 5–9 December 2015; pp. 242–254. [CrossRef]
18. Palnitkar, S. *Verilog HDL: A Guide to Digital Design and Synthesis*; Prentice-Hall, Inc.: Upper Saddle River, NJ, USA, 1996.
19. Parhami, B. Voting networks. *IEEE Trans. Reliab.* **1991**, *40*, 380–394. [CrossRef]
20. Balasubramanian, P.; Maskell, D. A distributed minority and majority voting based redundancy scheme. *Microelectron. Reliab.* **2015**, *55*, 1373–1378. [CrossRef]

21. Nagendra, C.; Owens, R.M.; Irwin, M.J. Power-delay characteristics of CMOS adders. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **1994**, *2*, 377–381. [CrossRef]
22. Laros III, J.H.; Pedretti, K.; Kelly, S.M.; Shu, W.; Ferreira, K.; Vandyke, J.; Vaughan, C. Energy Delay Product. In *Energy-Efficient High Performance Computing*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 51–55.