Lucas, J., Lal, S., & Juurlink, B.

Optimal DC/AC Data Bus Inversion Coding

Lucas, J., Lal, S., & Juurlink, B. (2018). Optimal DC/AC data bus inversion coding. In 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE.
https://doi.org/10.23919/date.2018.8342169

Optimal DC/AC Data Bus Inversion Coding

Jan Lucas, Sohan Lal, Ben Juurlink
Embedded Systems Architecture
TU Berlin
Berlin, Germany
{j.lucas,sohan.lal,b.juurlink}@tu-berlin.de

***Abstract*—GDDR5 and DDR4 memories use data bus inversion (DBI) coding to reduce termination power and decrease the number of output transitions. Two main strategies exist for encoding data using DBI: DBI DC minimizes the number of outputs transmitting a zero, while DBI AC minimizes the number of signal transitions. We show that neither of these strategies is optimal and reduction of interface power of up to 6% can be achieved by taking both the number of zeros and the number of signal transitions into account when encoding the data. We then demonstrate that a hardware implementation of optimal DBI coding is feasible, results in a reduction of system power and requires only an insignificant additional die area.**

***Index Terms*—Data bus inversion, DDR4, GDDR5, power consumption, termination power**

I. Introduction

Up to 50% of the power used by the memory is consumed by the external interconnect [1]. GDDR4/5/5X [2]–[4] as well as DDR4 [5] memories use a pseudo open drain (POD) electrical interface [6]. While the previously used SSTL interfaces terminate to a voltage at $0.5V_{DDQ}$, the POD interface is terminated to V_{DDQ}. In a terminated SSTL interface, DC current is always flowing, transmitting a zero or a one just changes the path of the current flow. In the POD interface, also illustrated in Fig.1, DC current through the termination resistors is only flowing when transmitting a zero, while transmitting a one does not cause DC current through the termination. Memory using POD signalling reduces the termination current by employing data bus inversion (DBI) [7]. For every 8 DQ (data) lines, a ninth DBI line is added. Transmitting a zero on this line signals that the 8 DQs lines contain an inverted data byte, while a one on the DBI wire indicates transmission of the non-inverted byte. The simplest DBI scheme is called DBI DC and simply counts the number of zeros in each byte and transmits the byte in its non-inverted form, if it contains 4 or fewer zeros. If the byte contains 5 or more zeros, the byte will be inverted. A byte with 5 zeros, will contain 3 zeros after inversion, however, the DBI bit will contain an additional zero indicating the inversion. This scheme guarantees that never more than 4 zeros per byte are transmitted.

In addition to the interface energy consumed by DC termination current, transitions from zero to one or one to zero consume dynamic power by charging and discharging of load capacities. The importance of the load capacities can also be seen in the design of the POD output driver: A regular open drain output would rely solely on the resistor to V_{DDQ} to generate high output state, but the pseudo open drain output actively drives the output to high to provide a faster recharging of the load and thus also a faster signal transitions than what could be achieved by the termination pull-up alone. Instead of reducing the number of transmitted zeros, the DBI signalling can also be used to reduce the number of signal transitions. In the DBI AC scheme, each transmitted byte is inverted, if the inversion reduces the number of signal transitions.

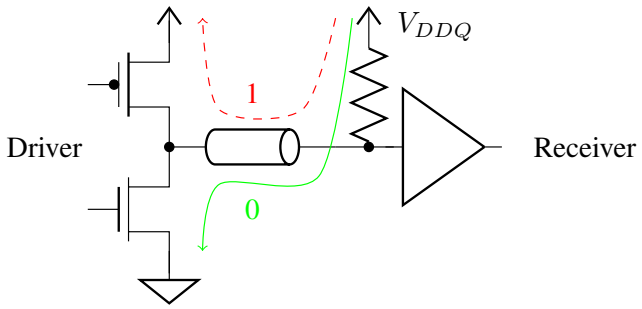

Fig. 1. Pseudo open drain (POD) interface

In this paper, we present a novel DBI encoding scheme. It finds a minimum energy DBI encoding of a burst, if given the ratio between the energy for transmitting a zero and the energy per transition. The paper is organized as follows: We first provide an overview of related work, then we introduce our optimal encoding algorithm and a simplified variant. Then in Section IV we explain how power was modelled and explain a hardware design that is able to perform the new DBI encoding at the required data rates. In the next section, we present our experimental results and finally we conclude our paper.

II. Related Work

Hollis [8] described the DBI DC and DBI AC schemes and recognized that both the number of transmitted zeros and the number of signal transitions are important for the power consumption of the memory interface. The slight increase of the signal transitions in DBI DC and the slight increase of transmitted zeros in DBI AC was also described in the same paper. Hollis proposes to combine DBI AC and DC by switching between DBI DC and DBI AC encoding modes. The proposed DBI ACDC scheme encodes the first byte of a group

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 688759 (Project LPGPU2).

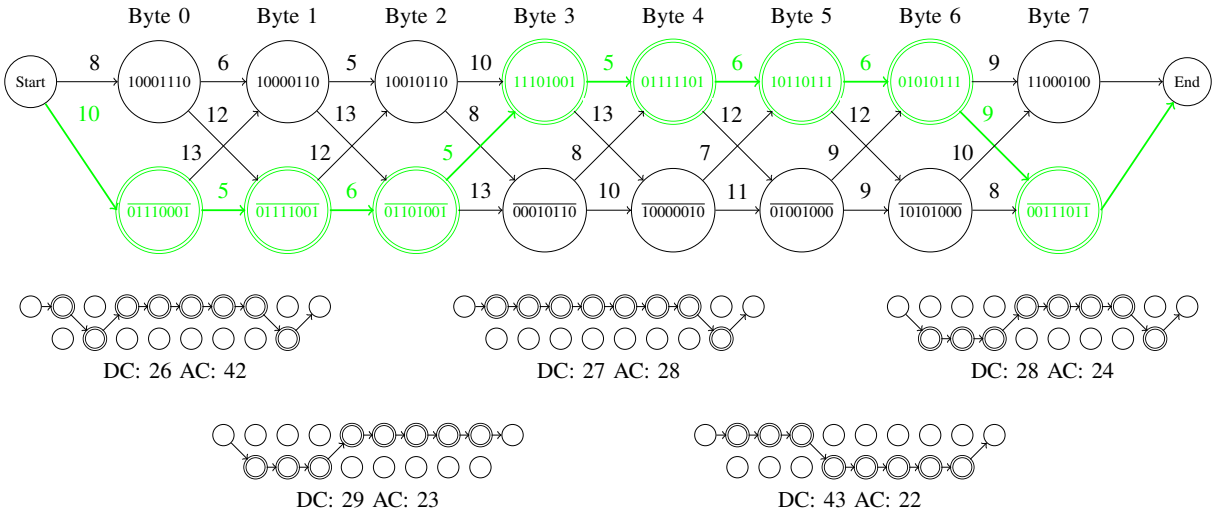

Fig. 2. Optimal DBI encoding as a shortest path problem

of bytes using DBI DC and then encodes the remaining bytes using DBI AC. We found that this scheme indeed provides a slight improvement compared to pure DBI AC. However, the encoding proposed in this paper outperforms the DBI ACDC scheme. In this paper we assume that all lines transmitted ones prior to transmitting the evaluated burst. Due to this boundary condition DBI AC performs identical to DBI ACDC.

Chang et al. [9] propose schemes that aim to reduce both zeros and transitions per burst. However, instead of finding the minimal energy encoding for each burst, they propose heuristic schemes that find good but not necessary optimal encodings.

In a patent Hollis [10] proposes a technique to target both signal transition and zeros. This technique uses additional signal lines and requires a different and more complex decoding process than regular DBI schemes.

Ihm et al. propose an analog circuit for DBI DC encoding [11]. Analog implementation could also reduce the overhead of the technique proposed in this paper and DBI encoding seems to be well suited for analog implementation as rare inaccurate encoding decision are unlikely to causes application errors.

Stan and Burleson [12] provide theoretical background on DBI encoding, however, they only consider the reduction of signal transition and do not consider the reduction of zeros.

Narayanan et al. [13] describe additional coding schemes that can reduce the number of signal transitions beyond DBI, but require an even higher number of lines and more complex encoding and decoding.

Kim et al. describe DBI DC in GDDR4 and show how it reduces simultaneous switching output noise [14].

III. Optimal Encoding

To reduce the power consumption, every burst should be transmitted using as little energy as possible. Each burst of 8 bytes can be encoded using 2^8 different DBI patterns. A naive algorithm would search through all possible encoding options and pick the cheapest one. But the cheapest encoding option can be found much more efficiently as we can reformulate the problem as a shortest path problem on a directed graph with nonnegative weights. This is illustrated in Fig. 2. The topology of the graph only depends on burst length. Two nodes exist for each byte, one node represents the transmission of the byte in its non-inverted representation, while the other node represents the inverted transmission of this byte. The cost of transmitting each byte depends only on the previous byte as well as the byte itself. Only two different previous bytes can exist, either the previous byte was inverted or not. The weight of the edges represents the cost of encoding each byte based on the previous byte. The shortest path from the start to the end node is the encoding with the minimum total energy. Three factors control the weights of the edges: The data that should be transmitted and the coefficients α and β. The α coefficient configures the cost of each signal transition, while the β coefficient sets the cost of each transmitted zero bit. As the shortest path does not change by a uniform scaling of the edge weights, we can freely scale the coefficients as long as the ratio $\frac{\alpha}{\beta}$ does not change. This allows us to use small integer coefficients without a significant loss of encoding efficiency. Our top example shows the shortest path and edge weights for $\alpha = \beta = 1$. This choice of α and β in the example implies that the energy cost of transmitting a zero is identical to the energy cost of a transition. If we vary the coefficients without changing the data, we find 5 other pareto optimal encoding options. The DBI DC algorithm finds an encoding with 26 zeros, but 42 transitions. The DBI AC algorithm finds the encoding with 22 transitions but 43 zeros. But neither of these two previous algorithms are able to identify the three encodings with a more balanced trade-off between zeros and transitions. If we assume $\alpha = \beta = 1$, then the optimal encoding has energy cost of $28 + 24 = 52$, while DBI DC choose an encoding with a cost of $26 + 42 = 68$ and DBI AC selects an encoding with a cost of $43 + 22 = 65$.

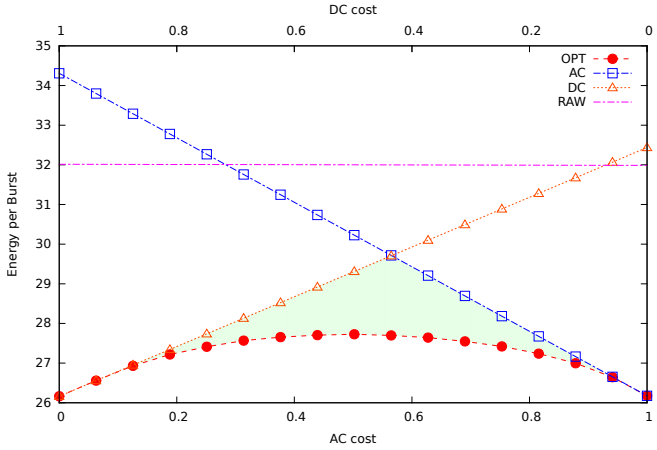

Fig. 3. Energy per Burst using different DBI schemes

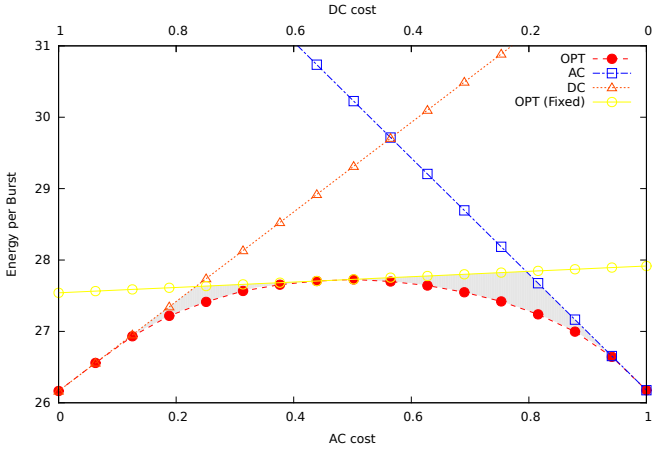

Fig. 4. Energy per Burst for different DBI schemes, shaded area shows loss of efficiency from fixed coefficients

We simulated the different DBI encoding schemes on 10000 random bursts. We varied the cost α per signal transition from 0 to 1 and set the cost $\beta = 1 - \alpha$. The result is shown in Fig. 3. DBI DC behaves identical to optimal DBI (DBI OPT) encoding when the AC cost is 0. This is no surprise as DBI OPT with $\alpha = 0$ and $\beta = 1$ is identical to DBI DC. DBI DC works almost as well as the optimum encoding until the AC cost reaches 0.15. Similar results can be seen for DBI AC. As expected DBI AC performs identical to DBI OPT when the DC cost is 0 and the performance stays close until the DC cost reaches 0.15. Both DBI AC and DBI DC perform worse than unencoded (RAW) data, when used together with high DC cost or AC cost, respectively. DBI AC encoding is cheaper than DBI DC encoding starting from $\alpha = 0.56$. The biggest advantage of optimal DBI encoding is also offered at this point, where the average cost per burst is 2 points or 6.75% lower than with DBI AC or DBI DC. The shaded area in Fig. 3 shows the advantage of DBI OPT encoding compared to the best conventional encoding scheme (DBI DC or AC).

One problem with DBI OPT encoding is the accuracy required for the coefficients. However, as we already saw with DBI AC and DBI DC, the coefficients do not need to be very accurate to still enable almost perfect encoding results. We fixed $\alpha = \beta = 1$ and named this encoding scheme DBI OPT (Fixed). Fig. 4 shows the results. The shaded area indicates the small reduction of performance due to the fixed coefficient. The encoding with fixed coefficients performs better than previous scheme from an AC cost of 0.23 to 0.79. The maximum energy reduction from this encoding is nearly identical at 6.58%.

IV. Experimental Setup

A. Power Model

We estimated the energy consumption based on a model derived from the CACTI-IO model presented by Jouppi et al [1], [15]. We unified all load capacities into a single load capacity and reformulated the equations from power to energy per activity.

E_{zero} is the energy consumed by transmitting a single zero.

$$E_{zero} = \frac{V_{DDQ}^2}{R_{pullup} + R_{pulldown}} \frac{1}{f} \tag{1}$$

$E_{transition}$ is the energy consumed by a single transition from zero to one or one to zero.

$$E_{transition} = \frac{1}{2} V_{DDQ} V_{swing} c_{load} \tag{2}$$

V_{swing} is the signal swing, it is calculated from the output resitance of the pulldown driver ($R_{pulldown}$) and on-die termination resistor. (R_{pullup})

$$V_{swing} = V_{DDQ} \frac{R_{pullup}}{R_{pullup} + R_{pulldown}} \tag{3}$$

The total interface energy per burst is calculated as follows:

$$E_{burst} = n_{zeros} E_{zero} + n_{transitions} E_{transition} \tag{4}$$

c_{load} is the total load capacity. We tested a wide range of values from 1 pF to 8 pF total load. It should be the sum of the effective capacities of the driver in the CPU or GPU, the capacities of the memory devices added to the DQ lines, the capacity of the transmission line connecting memory and CPU/GPU. If a system uses DIMM or similar sockets the extra load of those should also be considered. Amirkhany et al. state a 1.3 pF load for an GDDR5 output driver [16]. CACTI-IO assumes 2 pF for an DDR4 output driver and 1 pF per memory device [1]. Vuong lists a maximum capacity of 1.3 pF for DDR4 [17]. IBIS files from Micron also list similar values per DDR4 input. DIMM sockets and the PCB trace can add a few additional pF.

B. Hardware

To validate that the proposed DBI encoding can be done at the required data rates and add only a small overhead to a CPU or GPU using this scheme, we developed a hardware

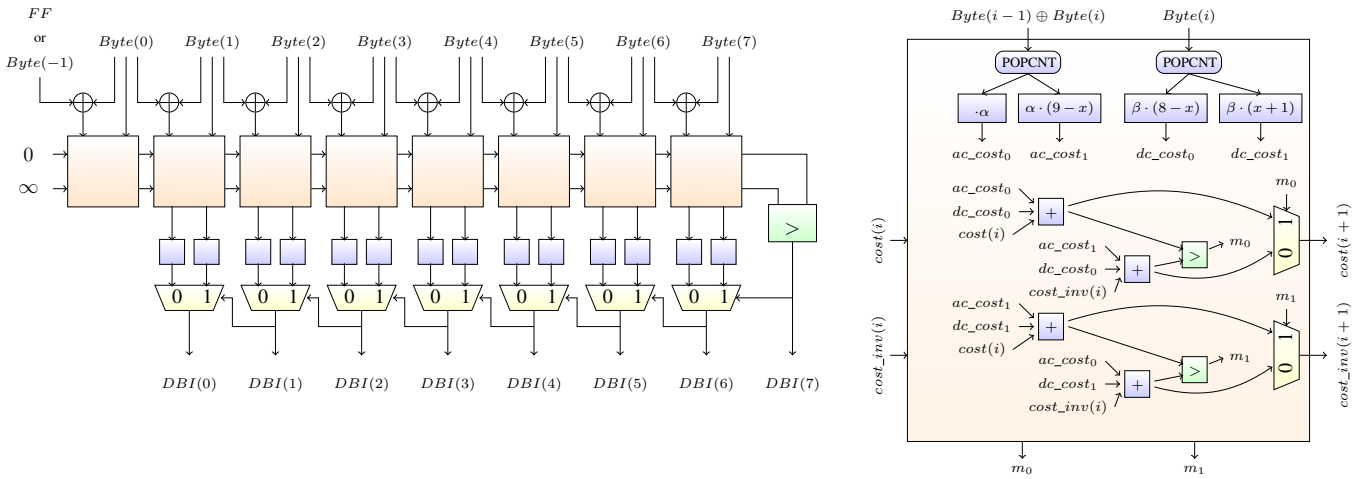

Fig. 5. Hardware architecture of improved DBI encoder

TABLE I
SYNTHESIS RESULTS (32NM)

Scheme	Area (μm^2)	Static Power (µW)	Dynamic Power (µW)	Burst Rate (GHz)	Total (µW)	Energy per Burst (pJ)
DBI DC	275	105	111	1.5	216	0.14
DBI AC	578	170	250	1.5	420	0.28
DBI OPT (Fixed Coeff.)	3807	257	2233	1.5	2490	1.66
DBI OPT (3-Bit Coeff.)	16584	5200	3600	0.5	8800	17.6

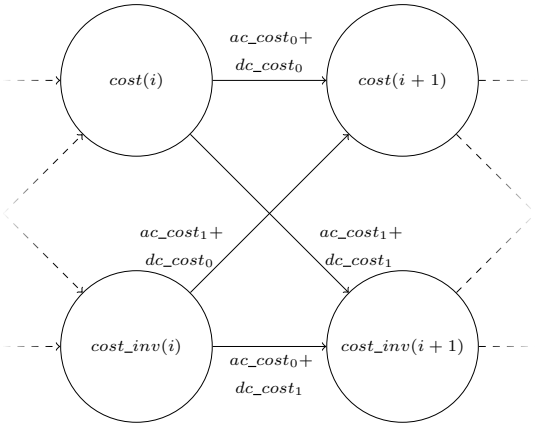

Fig. 6. Mapping of shortest path search to signals

implementation. Our proposed hardware architecture is shown in Fig. 5. Each byte of the burst is processed by one processing block. Each block receives two minimum costs: $cost(i)$ is minimum cost of transmitting bytes 0 to i-1 with the last byte transmitted in non-inverted encoding, while $cost\_inv(i)$ is minimum cost of transmitting those bytes with the last byte inverted. If we consider the problem as a shortest path problem, $cost(i)$ is the cost of the shortest path from the start to the node of the ith byte and $cost\_inv(i)$ is the cost of the shortest path to corresponding inverted node. Each of the processing blocks receives the byte itself as well as the exclusive-or of this byte and the previous byte. Within each processing block, two population count units (POPCNT) count the number of set bits in each of the two inputs on the top. dc_cost_0 is the cost of transmitting the current byte without inversion, i.e., the number of zero bits multiplied with the cost β of each zero. dc_cost_1 is the DC cost of transmitting the current byte inverted. In this case the extra zero transmitted on the DBI signal also needs to be considered, which results in the $+1$ term. Two options also exist for number of signal transitions: Either both the previous byte and the current byte are transmitted in the same way (ac_cost_0) or the DBI bit changed between the two bytes (ac_cost_1). Now the cost of four different encoding options can be calculated (from the top to the bottom): 1. Previous byte was not inverted, current byte also not inverted. 2. Previous byte was inverted, current byte is not inverted. 3. Previous byte was not inverted, current byte is inverted. 4. Previous byte is inverted, current byte also inverted.

The relationship to the graph is also shown in Fig. 6. To calculate the cost of reaching a node via one edge, we need to consider the cost of the edge as well as the minimum cost of reaching the source node of the edge. Two edges lead to each node and we compare their cost and store which of the edges provided the cheapest path. The cheapest path is then forwarded to the next block.

At the last block, we compare which of the two end nodes provides the shortest overall path. This path is backtracked to find the DBI pattern using the muxes below the blocks. This is the same technique that is also used in the Dijkstra's algorithm to reconstruct the shortest path.

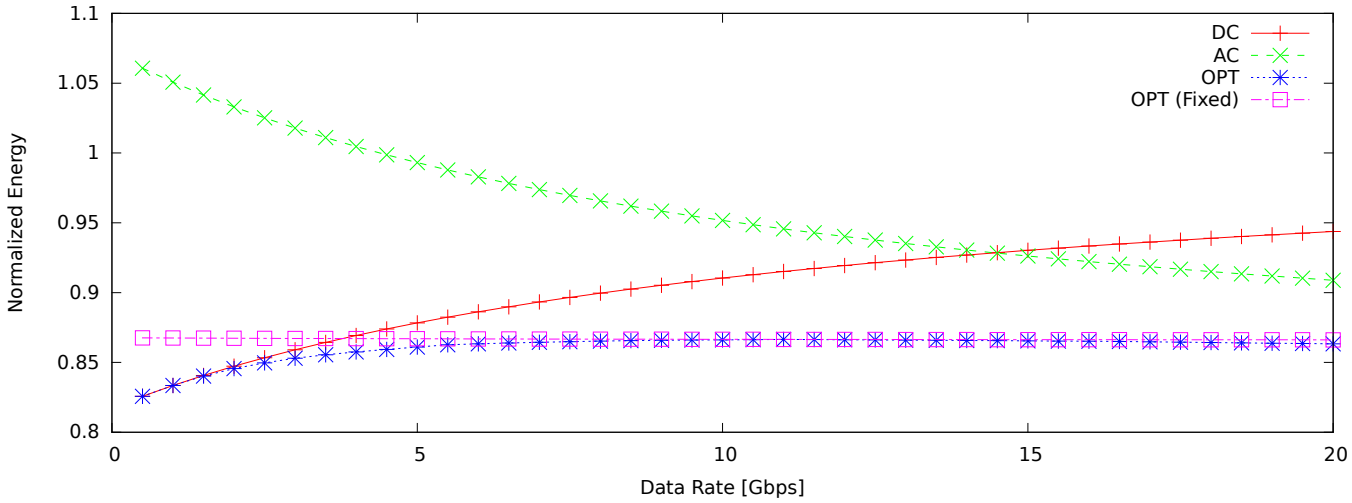

Fig. 7. Interface energy per burst normalized to unencoded transmission for various DBI encoding schemes

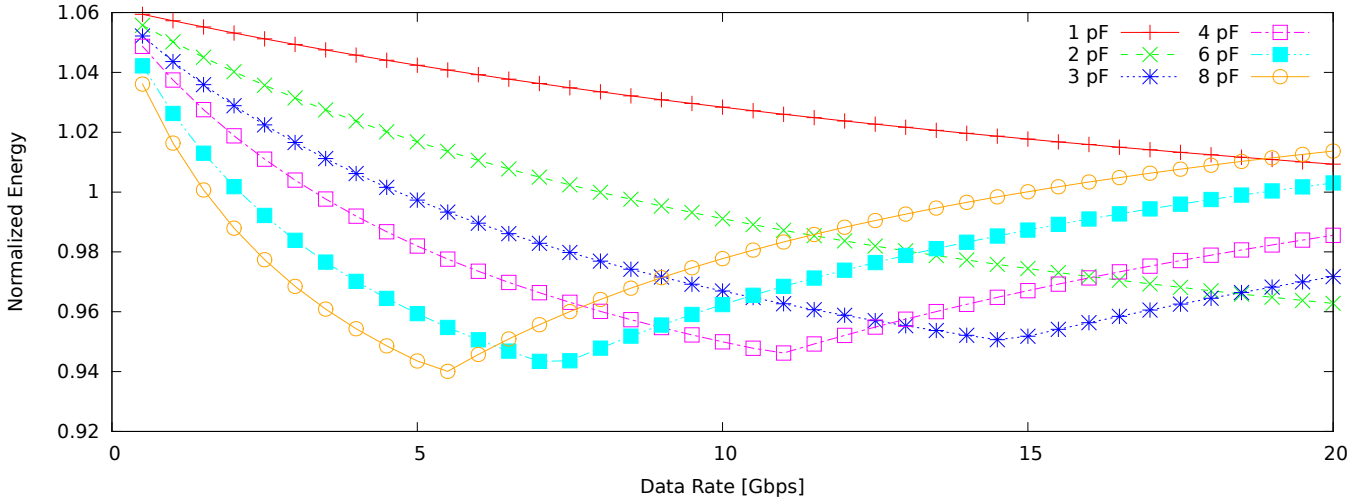

Fig. 8. Energy per burst using optimized encoding, including encoding energy, normalized to best of DBI DC or AC

We described our designs in VHDL and synthesized the designs using Synopsys Design Compiler Ultra K-2015.06-SP4 together with the Synopsys 32nm generic libraries in order to estimate the required die area, power and throughput. We synthesized two variants of our proposed design: One design used configurable 3-bit coefficients for α and β, while the other design fixes $\alpha = \beta = 1$. The fixed coefficients remove multipliers from the design and reduce the bit width of the data path. We added 8 pipeline stages to the output of our design and used the retime option of the synthesis tool to move the registers to an appropriate location. Current GDDR5X uses up to 12 Gbps data rate per pin. Our design encodes 8 bytes per clock cycle, thus a clock frequency of 1.5 GHz is required to meet the required throughput using a single encoding unit. Whether this design adds additional latency, depends on the design of the memory controller, often it should be possible to perform the encoding in parallel with other memory controller tasks. If extra latency is added, this can still be acceptable for GPUs: GPUs already have memory subsystems with hundreds of cycles of latency and their performance is relatively insensitive to additional latency [18].

V. Results

Table I shows the results of our synthesis. DBI DC, DBI AC and DBI OPT with fixed coefficients could meet the 1.5 GHz timing, equivalent to a data rate of 12 Gbps. DBI OPT with 3-Bit configurable coefficients was significantly slower and could only run at 500 MHz (equivalent to 4 Gbps). It also required 4.5x more area than the design with fixed coefficients and used 10.6x more energy per encoded burst than the design with fixed coefficients. Due to the lower frequency, 3 units are required to reach the same throughput, increasing the area requirements even further.

Fig. 7 displays the interface energy per burst normalized to the cost of transmitting the data without any DBI encoding using POD135 (used by GDDR5X) and 3 pF load. However, results for DDR4 with POD12 are almost identical. DBI DC performs better than DBI OPT (Fixed) until 3.8 Gbps. DBI AC would require a significantly higher frequency than 20 Gbps to perform better than this scheme. The maximum gain from this optimized encoding can be found around 14 Gbps.

The previous Fig. 7 does not include the energy required for encoding. If we also consider the energy for encoding, the picture changes. DBI OPT encoding with configurable coefficients encodes the data only slightly better than the fixed coefficient version, however, it uses significantly more energy for encoding each burst. For this reason it always consumes more power than the DBI DC and DBI AC schemes. However, further optimization of the hardware might change this. We used a relatively old 32nm process node for estimating the power consumption and an optimized implementation in a more recent process could provide a significant power reduction, that could make configurable coefficients beneficial.

Fig. 8 shows the energy per burst for DBI OPT with fixed coefficients normalized to the best conventional DBI encoding. Higher capacitive load reduces the frequency where the highest reduction of energy is achieved. At 3 to 8 pF load, the energy is reduced between 5-6% at the operating points with the highest gains.

VI. Conclusions

A novel DBI encoding scheme was presented, it reduces the link power consumption by up to 6%. It has been shown that the problem of finding an DBI encoding with the smallest link energy is equivalent to finding the shortest path in a graph. We presented a hardware design that performs the encoding at the required data rates using an insignificant extra area and energy. Additional optimization to reduce the hardware overhead including partially analog implementation are possible. A design with fixed coefficients provides a very good trade-off between the energy required for encoding and the saved link energy. It can be used without changing existing DDR4, GDDR5 and GDDR5X memories to reduce the interface energy during writes and could be integrated into future memories to also reduce read interface energy.

References

[1] N. P. Jouppi, A. B. Kahng, N. Muralimanohar, and V. Srinivas, "CACTI-IO: CACTI with off-chip power-area-timing models," *IEEE Transactions on VLSI Systems*, 2015.

[2] JEDEC Standard, "Graphics Double Data Rate (GDDR4) SGRAM Standard," *SDRAM3.11.5.8, May*, 2006.

[3] ——, "Graphics Double Data Rate (GDDR5) SGRAM Standard," *JESD212C, February*, 2016.

[4] ——, "Graphics Double Data Rate (GDDR5X) SGRAM Standard," *JESD232A, August*, 2016.

[5] ——, "DDR4 SDRAM Standard," *JESD79-4B, June*, 2017.

[6] JEDEC Standard , "POD15 - 1.5 V PSEUDO OPEN DRAIN I/O," *JESD8-20A*, 2009.

[7] S. J. Bae, Y. S. Sohn, K. I. Park, K. H. Kim, D. H. Chung, J. G. Kim, S. H. Kim *et al.*, "A 60nm 6Gb/s/pin GDDR5 graphics DRAM with multifaceted clocking and ISI/SSN-reduction techniques," in *IEEE ISSCC Digest of Technical Papers*, 2008.

[8] T. M. Hollis, "Data bus inversion in high-speed memory applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2009.

[9] N. Chang, K. Kim, and J. Cho, "Bus encoding for low-power high-performance memory systems," in *Design Automation Conference (DAC)*. ACM, 2000.

[10] T. M. Hollis, "Devices and methods for facilitating data inversion to limit both instantaneous current and signal transitions," 2016, US Patent 9,270,417.

[11] J. D. Ihm, S. J. Bae, K. I. Park, H. Y. Song, W. J. Lee, H. J. Kim, K. H. Kim *et al.*, "An 80nm 4Gb/s/pin 32b 512Mb GDDR4 graphics DRAM with low-power and low-noise data-bus inversion," in *IEEE ISSCC Digest of Technical Papers*, 2007.

[12] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power I/O," *IEEE Transactions on VLSI systems*, 1995.

[13] U. Narayanan, K.-S. Chung, and T. Kim, "Enhanced bus invert encodings for low-power," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2002.

[14] J.-H. Kim, W. Kim, D. Oh, R. Schmitt, J. Feng, C. Yuan, L. Luo, and J. Wilson, "Performance impact of simultaneous switching output noise on graphic memory systems," in *Electrical Performance of Electronic Packaging*. IEEE, 2007.

[15] N. P. Jouppi, A. B. Kahng, N. Muralimanohar, and V. Srinivas, *CACTI-IO Technical Report*. Department of Computer Science and Engineering, University of California, San Diego, 2012.

[16] A. Amirkhany, J. Wei, N. K. Mishra, J. Shen, W. T. Beyene, C. Chen, T. Chin, D. Dressler, C. Huang, V. P. Gadde *et al.*, "A 12.8-Gb/s/link tri-modal single-ended memory interface," *IEEE Journal of Solid-State Circuits*, 2012.

[17] H. Vuong, "Mobile memory technology roadmap," in *JEDEC's Mobile Forum*, 2013.

[18] M. Andersch, J. Lucas, M. Álvarez Mesa, and B. Juurlink, "On latency in GPU throughput microarchitectures," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2015.