This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.2978826, IEEE Transactions on Computers

STFL-DDR: Improving the Energy-Efficiency of Memory Interface

Payman Behnam and Mahdi Nazm Bojnordi

Abstract—Power dissipation is a significant problem limiting the performance of today's computer systems. One of the main contributors to power consumption in microprocessors is data movement in cache and memory interface. Several solutions such as low power interconnects, energy-aware data encoding, and low power signaling have been proposed to mitigate this problem. Almost all of these techniques result in a significant system performance degradation. This article examines the application of a novel technique, called STFL-DDR, for hybrid signaling on low-power DRAM interface. To keep the power consumption low, STFL-DDR employs a high-performance clock rate for transferring data on low power wires. To avoid any signal deterioration, STFL-DDR employs data encoding/decoding to prevent each wire from switching in any two consecutive cycles. STFL-DDR creates new opportunities for optimizing the energy-efficiency of DRAM systems. We compare the efficiency of STFL-DDR with the state-of-the-art methods by simulating a mix of 12 parallel benchmark applications on a muticore system. Our simulation results indicate that STFL can reduce the energy consumption of a contemporary DRAM interface by 17% as compared to an LPDDR baseline while achieving the throughput of a high-performance DRAM. Applying STFL to both last level cache and DRAM interface results in improving the system energy, energy-delay product, and performance by 8%, 15%, and 9% respectively. Compared with a high-performance memory interface, STFL improves the system energy and energy-delay product by 25% and 75%, while reaching 98% of the average performance of the high-performance system.

Index Terms—Memory Interface, Low Power Wires, Energy-efficiency, Hybrid Signaling.

1 Introduction

Transferring data over off-chip wires consumes more than 20% of the overall DRAM energy [1]. Figure 1 shows an energy breakdown for a multicore processor running a set of memory intensive parallel applications. [1] As illustrated in this figure, the last level cache (LLC) and DRAM IO together consume about 33% of the overall system energy. Recent studies show that the data movement energy is a dominant energy consumer for modern computer systems [2], [3], [4].

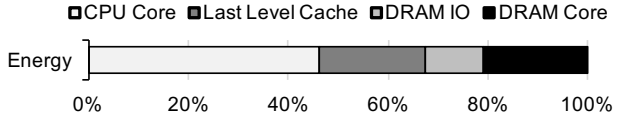

Fig. 1. **Example system energy breakdown for a multicore processor.**

Bit time is defined by the amount of time required for sending a single bit of information over a wire. The peak bandwidth of the wire is defined by the number of possible bit times per second. To increase the bandwidth of the wire, we can increase the number of bit times per second. Increasing the number of bit times may result in an increased number of switchings between a low and a high voltage levels. The higher the switching activity, the more dynamic power consumption. Based on the dynamic power equation $P = \alpha C V^2 f$, numerous techniques have been introduced to reduce power by lowering the switching activity (α), capacitance (C), voltage (V), and frequency (f). In our recent work [5], we propose STFL to improve the bandwidth and energy efficiency of low power wires in LLCs. In this article, we extend the application of STFL to DRAM interfaces using novel microarchitectural techniques.

1. Detailed explanation of the system configurations is provided in Section 5.

DRAM data bus consumes energy due to signal transitions (wire flips) for transferring data bits and on-die termination. Every wire flip expends energy for charging/discharging a wire capacitance. On-die termination circuit consumes energy for impedance matching and mitigating signal reflection in the data wires. A termination circuit enables a higher data rate in modern DRAM interfaces at the cost of dissipating a significant amount of DRAM energy. To alleviate this problem, asymmetric termination designs, such as DDR4 [6], GDDR4/5 [7], and LPDDR4 [8], have been proposed to reduce power consumption in high performance interfaces. However, significant power reductions are only possible through low power DRAM interfaces that leverage low voltage-swing signaling and unterminated wires—*e.g.*, LPDDR3 [9]. Regrettably, the bandwidth and energy efficiency of these techniques are limited mainly due to the significant reduction in the frequency of interface.

We examine a microarchitectural solution to achieve a higher bandwidth in low power DRAM interfaces. The key bottleneck to achieve a high bandwidth in low power wires is the transition speed that relates to the wire characteristic; and is hard to change. Despite this limitation, *we propose to signal data bits at high rates on low power wires.* To avoid signal deterioration when sending consecutive transitions, we pause the transmission by injecting delay cycles after each transition. Furthermore, we propose a simple encoding technique to reduce the number of transitions per transferred data and signal the voltage levels faster to reduce the transfer time. To the best our knowledge, STFL is the first architectural solution for hybrid signaling on low power cache and DRAM interfaces. While the area and power overheads of the encoding/decoding circuits are comparable to the state-of-the-art techniques, the system energy

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.2978826, IEEE Transactions on Computers

and performance potentials of the proposed solution are considerable. We demonstrate the efficiency of our method when we apply STFL to the DRAM interface and both DRAM and cache interfaces. Our simulation results over a mix of 12 parallel benchmark applications running on a multirate system shows that STFL improves the DRAM interface energy by 17% as compared to a low power DRAM interface. In addition, STFL is applied to both last level cache and DRAM interface to improve the system energy, energy-delay product, and performance averages by 8%, 15%, and 9% respectively.

2 Background

This section provides the necessary background on data communication in modern memory systems and the relevant energy optimization techniques.

2.1 Bus Termination in DRAM Interface

Terminated DRAM buses can reach a higher performance than unterminated interfaces because of the reduction in signal reflection within off-chip wires; however, they dissipate more energy due to on-die termination [10]. A basic on-die termination circuit comprises a switch (T) and a resistor (R_{term}) that matches the impedance of data wires. On every data reception, the switch is on and a DC current flows through the resistor that results in energy dissipation. Numerous techniques have been proposed to reduce this DC current. For example, DDR4 has adopted a pseudo open drain technique to address this problem (Figure 2(a)). On transferring a **1**, T_0 and T_1 are on and off, respectively; since R_{term} is connected to VDD, no DC current flows through the resistor. However, on transferring a **0**, T_1 is on that results in a DC current flowing through the resistor, thereby dissipating energy. In contrast, LPDDR3 [9] adopts an unterminated (slower) interface to reduce the power consumption in data wires (Figure 2(b)). On every wire flip, a load capacitance (C) is charged/discharged that results in switching power. The absence of a termination circuit narrows down the interface power only to signal transitions. The proposed STFL-DDR employs unterminated wires for sending data bits to completely remove the termination power in data bus; it leverages a novel frequency aware encoding to improve the bit rates in low power wires and exploits data locality to further reduce the switching activity in unterminated wires.

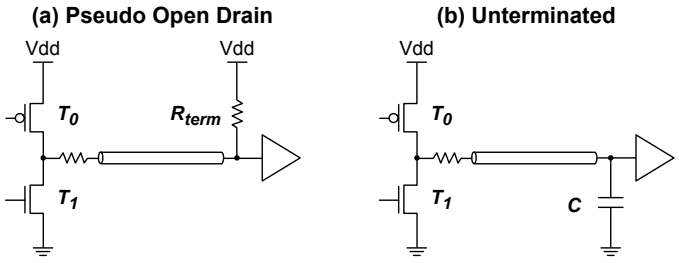

Fig. 2. **Pseudo open drain (a) and unterminated (b) memory interfaces.**

2.2 Energy-Efficient Data Encoding

Data encoding is a popular way to reduce switching activity and dynamic power consumption over the LLC and DRAM interfaces. Numerous techniques have been proposed for energy efficient data encoding. Bus invert coding (BIC) sends the data or its complement: the one that leads to less switching activity [11]. Data bus inversion (DBI) [12] applies the BIC technique to open drain interfaces like GDDR5, DDR4, and LPDDR4 to reduce power consumption.

DESC [13] sends a transition per data chunk (a group of 4 bits) on data wires while using synchronized counters at the sender and receiver to represent data in terms of the elapsed cycles between two consecutive transitions. Due to guaranteeing up to one transition per data chunk, DESC decreases the switching activity of wires; however, the transmission time depends on the value of data chuck that is typically longer than binary encoding. Adaptive time-based encoding [14] monitors the application phases and memory burst over the data bus at run time and applies either binary encoding or DESC encoding to improve energy-efficiency. Flip-N-Write [15] applies a bus inversion coding to the phase-change memories to reduce the write energy. CAFO [16] is a cost aware flip optimization method suitable for non-volatile memories with asymmetric endurance and energy for writing **1** and **0**. The goal of CAFO is to minimize the cost of write operations. SETS [17] makes use of limited weight codes [18] to make the wire energy proportional to the blocks' Hamming weight.MiL [19] has shown an application of sparse encoding to the DDR4 interface.

History based methods utilize the similarities between the past and future data blocks to reduce switching activity. For example, bitwise difference (BD) encoding [20] utilizes a table to detect similarities between data words sent over the bus and sends the difference between the current data and the most similar entry of the table. Recent work on online data clustering and encoding [21] clusters data blocks at the transmitter and receiver sides. It computes the cluster centers and sends the difference between the data block and the closest center along with its ID.

Due to the ever-increasing bandwidth demand in GPU systems, energy consumption of high speed DRAM interfaces has become a significant challenge. Lee *et al.* propose a mechanism that captures data similarities across GPU DRAM transactions to reduce the data movement energy in terminated, pseudo open drain I/O interfaces [22].

3 STFL Coding

STFL proposes a hybrid technique for slow-transition, fast-level (STFL) signaling that creates a balance between power and bandwidth in the last level cache and DRAM interfaces. Transition speed is the key bandwidth bottleneck in low power wires. STFL employs a hybrid technique that transfers signal levels faster and reduces the number of transitions in every data block. Instead of using binary encoding that represents **1**s and **0**s with two different voltage levels, STFL exploits signal transitions that map every **1** to a signal transition on the wire and every **0** to the absence of wire transitions. This creates an opportunity for controlling the bit flips over wires via data encoding. In addition, STFL sends and receives data at a high clock rate. The main problem is the signal deterioration that may happen when transferring consecutive transitions. To address this problem, STFL detects each transition and injects a delay

cycle (**0**) after the transition (**1**). The STFL receiver detects the transitions and removes those inserted *dummy* delay cycles (**0**).

Figure 3 demonstrates an example of sending a four-bit data (**0011**) by making use of high speed wire, low power wire, and STFL interface. The high speed wire sends data with the fastest transmission time (t). However, it needs a high voltage that leads to a high power consumption. The low power wire has the longest transmission time ($2t$) and the least power consumption. STFL is a hybrid data transmission technique that consumes similar power to the low power technique. With the help of the dummy cycles **(D)**, STFL is able to send **0**s at a higher rate than **1**s, thereby reducing the overall transition time from $2t$ to $1.5t$. As a result of optimizing both power and time, STFL is now able to improve the energy efficiency of data transmission compared to the other two techniques.

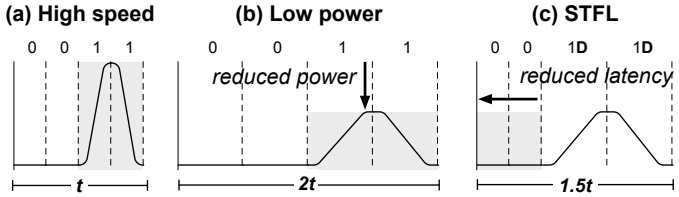

Fig. 3. **Transferring a 4-bit data with transition signaling on high speed wire (a), low power wire (b), and STFL interface (c) [5].**

4 Applying STFL to DRAM Interface

DRAM power and bandwidth are crucial to the energy efficiency and performance of computer systems. Therefore, numerous optimization techniques have been proposed in the literature to improve the efficiency of DRAM subsystems [1], [23]. Memory IO dissipates an increasingly significant amount of DRAM energy as more memory bandwidth is utilized. Figure 4 shows the IO power and the peak memory bandwidth provided by various DRAM generations and the proposed STFL-DDR mechanism. LPDDR interfaces consume a lower power at the cost of limiting the memory bandwidth compared to DDR. Instead, STFL-DDR strikes a balance between power and bandwidth by employing low power wires from LPDDR3 to transfer data bits and high performance wires from DDR4 to carry the clock and control signals. A controller is used to efficiently manage data movement in the proposed memory interface.

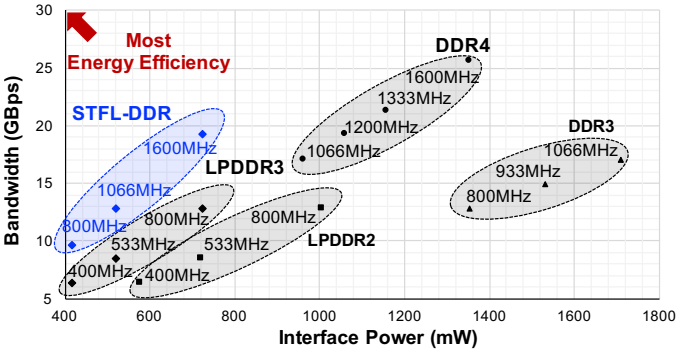

Fig. 4. **Energy efficiency of STFL-DDR and conventional DRAM interfaces.**

Figure 5 illustrates an example data movement between a CPU and an eight-chip DRAM DIMM using STFL-DDR. Similar to DDR4 and LPDDR3, each of the DRAM chips employs nine data wires for transferring eight data bits and a mode bit[2]; therefore, a DIMM provides a 64-bit data bus. STFL-DDR employs the same unterminated wires as used in the LPDDR3 (800MHz) data bus for the data bits and a pseudo open drain wire as used in DDR4 (1600MHz) for the mode bit.

Every data wire is connected to an *STFL-DDR transmitter* and an *STFL-DDR receiver*. STFL-DDR employs the encoder/decoder unit to control every group of eight transmitter-receiver pairs. The transmitter transfers a byte by generating a set of transitions (wire flips) on a data wire; while, the receiver retrieves the original data from those transitions.

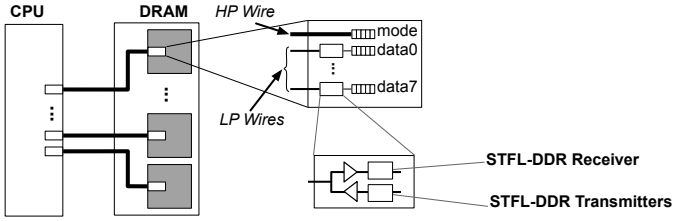

Fig. 5. **Illustrative example of the proposed STFL-DDR interface.**

4.1 STFL-DDR Transmission

As show in Figure 6, the transmitter follows multiple steps to generate appropriate transitions for every input data. STFL-DDR restricts the number of transferred **1**s per transmission to reduce the number of transitions. First, a population counter computes the Hamming weight of each byte; if the result is greater than four, the inverted data (otherwise, the original data) is stored in a parallel-in, serial-out shift register. This inversion is necessary to guarantee that no more than four **1**s will be transferred on the data wire. Then, the contents of the shift register are serially read and converted into wire flips using a transition generator circuit consisting of a latch and an XOR gate. Similar to the STFL-LLC transmitter [5], a delay injector circuit controls the shift register by maintaining the previous output value and disabling the shift operation if the value is a **1**. A transition generator circuit translates STFL codes into switchings in DRAM data bus. Despite using a high speed clock signal, the delay injector logic can avoid generating two transitions in consecutive cycles; as a result, the transmitter can keep the maximum transition rate on every data wire below 1.6 giga transitions per second.

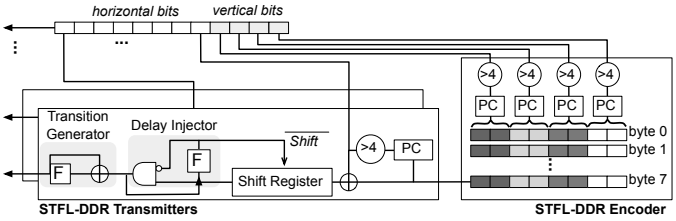

Fig. 6. **Illustrative example of the proposed transmitter for STFL-DDR interface.**

4.2 STFL-DDR Reception

The receiver comprises a transition detector and a serial-in, parallel-out shift register (Figure 7). The transition detector

2. The mode bit in DDR4 and LPDDR3 indicates if data bus inversion (DBI) is applied to the data bits [12].

employs an XOR gate and a flip-flop to convert every transition to a **1** and the absence of a transition to a **0**. The resultant bit stream is then sent to the shift register. STFL-DDR identifies and removes dummy **0**s from the received stream simply by pausing the shift register and overwriting the previously received bit. Finally, the shift register's contents are XORed with the corresponding inversion bit to retrieve the original data. Notice that both the STFL-DDR transmitter and receiver operate at the high interface clock frequency.

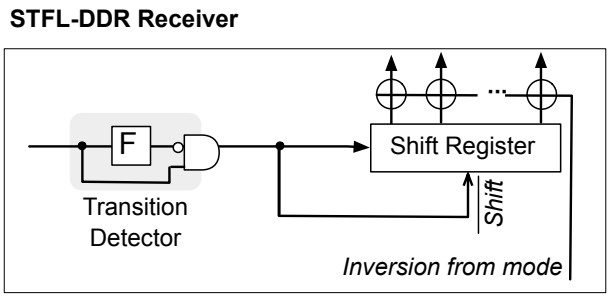

Fig. 7. **Illustrative example of the proposed receiver for STFL-DDR interface.**

4.3 Optimizing STFL Codes for DRAM

In addition to the transmitter and receiver, STFL-DDR employs a simple encoder to further improve power consumption in the memory interface. The key idea is to reduce the number of transitions (**1**s) on data wires by exploiting spatial locality across adjacent data wires. (Notice that all of the inversion control bits are transferred on the mode wire.) As shown in Figure 6, STFL-DDR applies data encoding to every array of 8×8 data bits. Each row of the array is assigned to a data wire. Four vertical and eight horizontal mode bits are generated for every 8×8 data array. STFL-DDR uses two phases to reduce the number of **1**s (transitions) without incurring significant overhead.

In the first phase, every pair of adjacent columns are XORed. For each computed result, if the Hamming weight is greater than four, the leftmost column of the pair is inverted and the corresponding vertical mode bit is set to **1**. As a result of this bitwise inversion, the similarity between adjacent columns increases that helps to reduce the number of ones in the second phase.[3] In the second phase, STFL-DDR computes the Hamming weight of each row separately. If the result is greater than four, the row is inverted and its horizontal mode bit is set to **1** (left of Figure 6). Increasing the similarity between columns in the first phase results in having most bits of the rows set to either **0** or **1**. Hence, the proposed techniques can significantly reduce the total number of **1**s, while it is guaranteed that every row does not contain more than four **1**s.[4]

4.4 STFL-DDR Clock Frequency

The proposed STFL-DDR interface employs a reference 1600 MHz clock frequency. We do not allow a variety of time scales for 0s and 1s. 1s are only allowed to be transferred at the half frequency of zeros; therefore, a single clock reference is sufficient for synchronization.The transmission frequency is set to 1600MHz (the same as the mode wire); however, STFL-DDR encoding guarantees that the required frequency for sending transitions on the data bus does not exceed 800MHz. We inject dummy zeros after each 1 as explained in Section 3. This means that for sending 8-bit with at most 4 ones, we add 4 dummy zeros and send them with the frequency of 1600MHZ. In DDR we can send 2 bits in each cycle time; however, in STFL, we send 8 data bits in 6 cycle time, which means that we can send 1.33 bits in each cycle time. Accordingly, the bandwidth of the STFL (2.13 Gps) is equal to the bandwidth of a DDR that works with the frequency of 1066 MHz. The proposed interface employs a reference 1600 MHz clock and an appropriate encoding and signaling mechanism to significantly improve the energy efficiency of data movement in DRAM interface. As shown in Figure 4, the 1600MHz STFL-DDR provides a memory bandwidth almost equal to that of the 1066 MHz DDR4 interface at a significantly lower power.

Due to the dual-data rate of DRAM interface, STFL-DDR transfers data bits on both edges of the clock. Therefore, transferring an 8×8 data array (i.e., 64 bits) between CPU and a DRAM chip requires 6 DDR clock cycles. (This paper employs a 1600MHz clock for STFL-DDR that provides the same bandwidth as in a DDR4 1066 MHz interface.)

Overall, the proposed mechanism further reduces the dynamic power consumption of STFL-DDR in the unterminated data wires by decreasing the number of transitions (**1**s) while it sends and receives data bits at high performance rate.

5 Experimental Setup

We evaluate area, performance, power, and energy of STFL-DDR based on hardware synthesis with a 45nm CMOS technology library [24]. The synthesis results are scaled to 22nm using the scaling parameters provided by the prior work [25]. Our SPICE models for the interfacing circuits are based on the PTM [26] high-performance 22nm transistors. We use the HSPICE simulator to estimate the delay and energy overheads. To evaluate the overall energy and performance potentials of the interconnects in DRAM, CACTI IO [27], Micron power calculator [28], [29], and DRAM-Power [30] are used. We use McPAT [31] to estimate the overall processor power consumption.

5.1 Processor Architecture

We employ a heavily modified ESESC simulator [32] to model a multicore processor system that includes data along with every memory request for calculating the dynamic energy, accurately. The multicore system comprises four OoO cores with private L1 cache and a shared 4MB LLC interfaced to two DRAM channels. The simulation parameters for the processor are shown in Table 1.

TABLE 1
Processor architectural parameters.

Core	four 4-issue OoO cores, 128 ROB entries, 3.2 GHz
IL1/DL1 cache	32KB, 4-way, LRU, 64B block, hit/miss delay 1/1
shared L2 cache	4MB, 8-way, LRU, 64B block, hit/miss delay 8/2, MESI protocol
Temperature	360 K (77 °C)

3. We also observed that this technique can reduce the total number of transitions per data blocks.

4. This requirement is forced by STFL to limit the transmission delay.

5.1.1 Methodology

We evaluate and compare STFL-DDR interfaces with the state-of-the-art encoding and signaling techniques such as the conventional binary encoding, bus invert coding [11], BD encoding [20], DESC [13], SETS [17], and two-dimensional block coding with CAFO [16]. Furthermore, we model a low power (LP) baseline using low power wires for both last level cache and DRAM interface. We use low voltage-swing wires in the LLC H-trees and unterminated LPDDR3 wires in the DRAM IO interface. We develop two versions of the bus invert coding optimized for power in data wires: DBI-LLC is used to reduce the switching activity of on-chip wires for LLC and DBI-DDR is developed to reduce the total number of transferred **1**s over the DDR4 wires. Similarly, two versions of CAFO and BD encoding are developed for reducing the switching activity and number of **1**s in cache and DRAM. CAFO-LLC is applied to data blocks while the size is 8×8 bits; whereas, CAFO-DDR is optimized for the burst length of DRAM by encoding 8×16 data blocks. We also adopt two optimized versions of history-based BD encoding with 64- and 32-entry tables for DRAM and LLC interfaces, respectively. DESC and SETS require excessive time and wire overheads that make them largely inefficient solutions for DDR interfaces.[5] As a result, these two techniques are applied only to the last level cache interface.

To account for the energy and delay overheads of the STFL coding, we consider reusing the existing components from LPDDR3 and DDR4 DRAM devices. However, more energy-efficiency and better performance are expected to be achievable through a custom design. We use the clocking circuits and pseudo open drain wires for mode bits from a 1600MHz DDR4 device, while the data wires are borrowed from an 800MHz LPDDR3 device.

5.1.2 DRAM Performance and Energy Model

We implement a detailed model of a two-channel DRAM system for the proposed and baseline systems to carry out all the cycle-accurate performance simulations and energy estimation in the ESESC simulator [32]. As the same DRAM core technology is used for all the evaluated systems, we set the timing parameters based on a DDR4-2133 device (Table 2).[6] We calculate the burst time (t_{BURST}) based on the requirements of each interface: it is set to 6 for STFL-DDR and 4 otherwise. Notice that 6 STFL-DDR cycles at 1600MHz result in about the same amount of transmission time as 4 DDR4 cycles at 1066MHz (3.75 ns). However, the 4 LPDDR3 cycles at 800MHz requires 33% more transmission time than STFL-DDR and DDR4 interfaces.

TABLE 2
DRAM timing parameters (ns).

DDR4 [6]	tRCD: 14.16, tCL: 13.32, tWL: 16, tCCD: 4, tWTR: 7.5, tWR: 12, tRTP: 7.5, tRP: 13.32, tRRD: 4, tRAS: 32, tRC: 45.32, tFAW: 30

Table 3 shows the parameters used for DRAM energy calculation using different interfaces. DRAM core parameters are common for all three interfaces. However, the IO related parameters are set according to the components that exist in each interface. For example, IDD3 parameters[7] mainly account for the clocking circuit in the standby mode; therefore, we use the same values as in the 1600MHz DDR4 for STFL-DDR. To compute the data movement energy on wires, STFL-DDR employs two values for each read and write IDD4 parameters. The two values correspond to the mode and data wires.

TABLE 3
DRAM power parameters (mA).

	DDR4	LPDDR3	STFL-DDR		Common in All
IDD3P	33	10	35	**IDD0**	56
IDD3N	57	47	63	**IDD2P**	22
IDD4R	135	265	170/265	**IDD2N**	41
IDD4W	117	294	154/294	**IDD5B**	297

Table 4 shows voltage, energy/bit, termination and switching power values for all the DDR4 and LPDDR3. STFL-DDR employs LPDDR3 (800MHz) low power wires for data only; while, the corresponding clock data recovery circuits [35], [36], [36], [37], [38], [39], [40], [41], [42]—e.g., PLL or DLL units—are connected to high performance wires operating at a $2\times$ faster clock rate (i.e., 1600MHz). Mode bits are sent over DDR4 wires. The fixed ratio between the two frequency domains eliminates the need for transmitting multiple clock references for data recovery. Therefore, similar to the prior work on reliable and low-jitter clock data recovery [43], [44], [45], a frequency divider converts the high-frequency clock to an appropriate reference for recovering data from the low power wires. Therefore, STFL-DDR is able to employ the high frequency clock and division circuits for data equalization [44], [46], [47] and double-edge signal alignment [45], [48] at the receiver.

TABLE 4
Voltage, energy/bit, and termination power values [33], [34]

Parameter	DDR4	LPDDR3
VDDQ	1.2	1.2
energy/bit (PJ/bit)	7.4	2.6
termination Read power (mW)	16.2	-
termination Write power (mW)	14.0	-
switching power (mW)	3.4	4.14

5.1.3 Applications

A mix of twelve parallel applications from Phoenix [49], NAS [50], and SPLASH-2 [51] benchmark suites are used to evaluate the impact of STFL codes on both memory intensive and non-intensive applications. We run the simulations until completion for power, and performance evaluations. Table 5 summarizes the evaluated benchmarks and their input sets.

5.2 Applying STFL to Cache

In this Section, we briefly review how to employ STFL at the last level cache interface [5].

5.2.1 Applying STFL to Large Caches

On-chip last level caches are performance critical components in modern computer systems that occupy significant die area and consume considerable amounts of energy. Due

5. DESC employs a temporal encoding mechanism that requires a maximum of 16 cycles; where, SETS needs either $4\times$ pins or time to transfer a byte.

6. Notice that appropriate conversion of the parameters to DRAM cycles is necessary according to the frequency of each interface.

7. IDD3N represents the current flowing through the DRAM when at least one bank is active; IDD3P is the current when an external clock is on during the power-down mode.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.2978826, IEEE Transactions on Computers

TABLE 5
Applications and data sets.

Label	Benchmarks	Suite	Input
FT	Fourier Transform	NAS OpenMP	Class A
IS	Integer Sort	NAS OpenMP	Class A
MG	Multi-Grid	NAS OpenMP	Class A
CG	Conjugate Gradient	NAS OpenMP	Class A
BT	Block Tri-diagonal	NAS OpenMP	Class A
RAY	Ray Trace	SPLASH-2	car
OCN	Ocean	SPLASH-2	514x514 ocean
FFT	FFT	SPLASH-2	1048576 data points
LU	LU	SPLASH-2	1024×1024 Matrix
BRN	Barnes	SPLASH-2	16K particles
HIST	Histogram	Phoenix	100MB file
WCNT	Word Count	Phoenix	10MB text file

to the large interconnects used for transferring data in last-level caches, accessing a data block necessitates expending a large amount of dynamic energy. To address this problem, STFL-LLC is used to optimize the cache energy by reducing the number of wire flips in the interconnects.

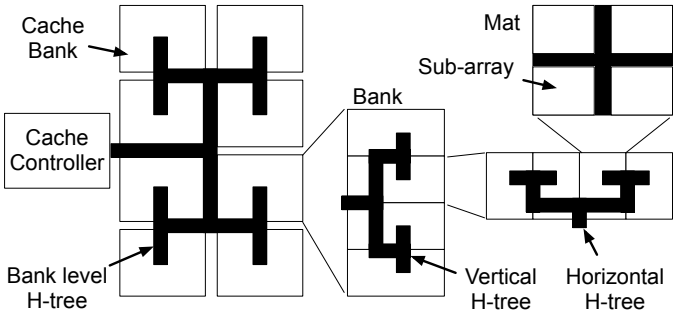

Fig. 8. Hierarchical organization of an example last-level cache with eight banks.

As shown in Figure 8, large caches are typically organized as a hierarchy of banks, sub-banks, mats, sub-arrays, and multiple H-trees that are disciplined by a cache controller. Independent banks are accessed simultaneously through a bank-level H-tree; each bank comprises a group of sub-banks that share the wires of a vertical H-tree; within every sub-bank, multiple mats are connected to a horizontal H-tree and supply different bits of the cache block in a bit parallel fashion. The number of subarrays inside each mat is always 4. The optimal number of banks, subbanks and mats explored through CACTI [52] to minimize energy-delay product (EDP) . Every read and write access requires moving data over long and capacitive wires within the H-trees, which results in significant delay and power consumption [13], [53]. STFL-LLC reduces the overall data movement energy in the cache interconnects through (1) using low power wires in data H-trees, and (2) integrating a set of STFL transmitters and receivers in the mats and cache controller to perform data transmission.

We apply STFL to the input and output data buses transferring a cache block between the cache controller and the selected mats during every cache access. Figure 9 depicts transferring a 64-byte cache block using an STFL-LLC interface with 16 groups. STFL divides every cache block into multiple groups of four bytes. Each group is converted to four STFL codewords transferred over four low power data wires. STFL employs an existing low power wire to transfer the encoding modes used for the four codewords. Finally, the receiver detects the signals and converts the codes to the original data block.

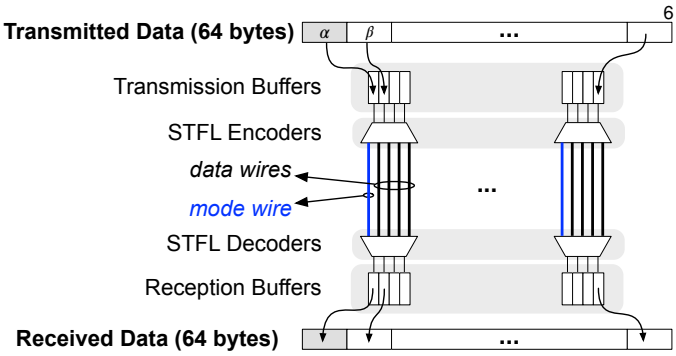

Fig. 9. **Illustrative example of transferring a 64-byte cache block using the STFL-LLC interface.**

5.2.2 Data Encoding with STFL-LLC

The proposed STFL-LLC mechanism exploits the similarities between adjacent bytes (i.e., spatial locality) in every cache block to reduce the Hamming weight of the codewords. This optimization is implemented through defining multiple encoding modes for every byte (α) to be transferred. The STFL-LLC encoder estimates the energy and delay costs for all of the possible codewords through computing the Hamming weight (Φ) of each candidate. Therefore, STFL-LLC selects the codeword with less Hamming weight to be transferred for the data byte. Table 6 shows the three possible encoding modes and the corresponding codewords for every α. The mode is set to 000 if the original data (α) is selected as the codeword. This mode is useful for transferring low Hamming weight bytes, such as 00000000.[8] STFL-LLC employs mode 01**D** for transferring the inverted data ($\overline{\alpha}$) to reduce the number of **1**s in heavy bytes, such as 11111111. The 1**D**0 mode is used for transferring the difference between α and its adjacent byte β within the same group. Notice that the mode bits of each group are serially transferred on a low power wire, thereby requiring a **D** after every **1**.

TABLE 6
STFL-LLC encoding.

Condition	Codeword	Mode
$(\Phi(\alpha) \leqslant 4) \wedge (\Phi(\alpha \oplus \beta) < \Phi(\alpha))$	$\alpha \oplus \beta$	1**D**0
$(\Phi(\alpha) > 4) \wedge (\Phi(\alpha \oplus \beta) \geqslant \Phi(\overline{\alpha}))$	$\overline{\alpha}$	01**D**
Otherwise	α	000

One difficulty in mode 1**D**0 that computes codewords by XORing the adjacent bytes is the possibility of forming long chains of XORs at the decode time. To avoid delay and energy overheads, STFL-LLC limits the XOR coding in mode 1**D**0 to every data group only. The rightmost byte of each group may be XORed with a fixed constant value (01010101) rather than its adjacent byte. [9] Figure 10 illustrates the proposed encoding mechanism for STFL-LLC. The encoder employs two population counters and a simple encoding logic to prepare data prior to transmission on a data wire. Based on Table 6, the logic generates three mode bits indicating which encoding is applied to the data (α). STFL-LLC generates a total of 12 mode bits for all of the bytes in every data group and transmits them using a single mode wire. Similarly, the decoder employs Table 6

8. Prior work [13] shows that about 30% of the transferred bytes may be zero.

9. Our studies indicate that XORing with a constant value that provides a code equally distant from the true and inverted value of the original byte results in a lower number of ones.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.2978826, IEEE Transactions on Computers

to convert the received codewords into the original data. In addition to the encoder and decoder units, STLF employs a transmitter and a receiver to generate and detect the corresponding signals with every codeword.

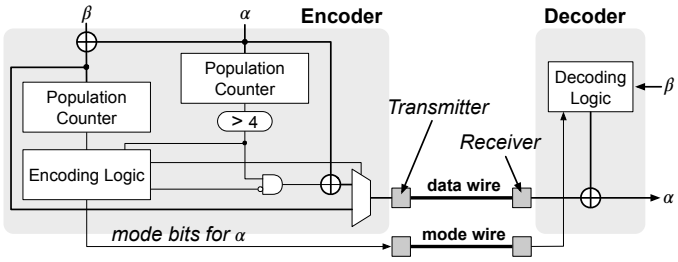

Fig. 10. **Illustrative example of the STFL-LLC encoder and decoder.**

Figure 11 shows how an 8-bit codeword is transferred over an example STFL interface that includes three mechanisms for transmitting codewords, receiving signals, and transferring mode bits. The transmitter sends each codeword by generating a set of transition signals on the data wire; the receiver detects those transitions and recovers the original data.

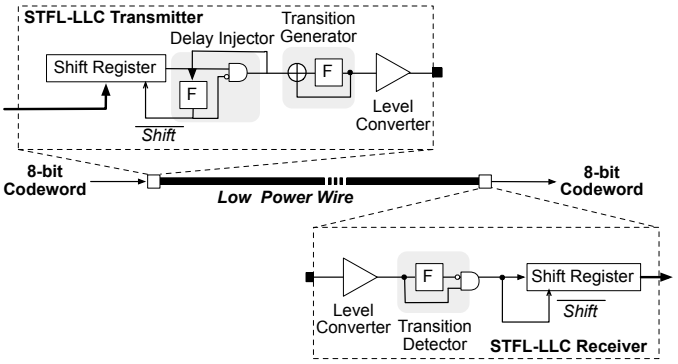

Fig. 11. **Transferring data codewords in STFL-LLC.**

Transmitting Encoded Data. To ensure the transition signals are properly generated for each codeword, multiple steps are followed by the STFL-LLC transmitter. First, STFL-LLC stores the eight-bit codeword generated by the encoder in a parallel-in, serial-out shift register. (Due to using the encoding modes as explained in Table 6, it is guaranteed that every codeword contains no more than four **1**s.) STFL reads the code bits serially read from the shift register and converts into transition signals using a transition generator comprising a latch and an XOR gate. A delay injector controls the shift register and maintains the previous output of the shift register. The delay injector is connected to the shift register via an (active low) shift signal; every **1** transmitted in the previous cycle disables the shift register in the current cycle, thereby injecting a **D** after every **1** in the code. Since the shift register can now contain up to four **1**s, the longest generated codeword is 12 bits long. To avoid the complexity of variable length encoding, the transmitter is set to produce fixed 12-bit codes (zero padding is required for the codes with fewer **1**s). The STFL codes are serially fed into a transition generator circuit that translates every **1** into a flip on its output. Finally, STFL-LLC employs the level converter to prepare the signals prior to transmission on the low-power wires by converting from full to low-swing.[10]

Transmitting Mode Bits. Unlike codewords, mode bits can be directly converted to the transition signals on the wire with no need for delay injection. The STFL-LLC encoder generates a total of 12 mode bits for every four data bytes, where **1**s are spaced out by dummy **0**s in the resultant bit pattern. Similarly to the data codewords, transferring the mode bits requires 12 cycles. Figure 12 shows how the mode bits are transmitted on a low-power wire.

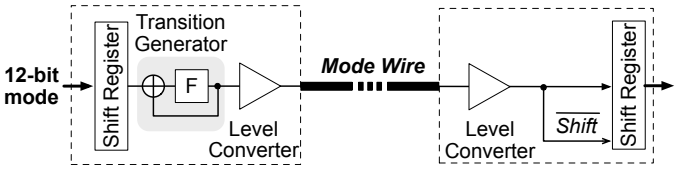

Fig. 12. **Transferring mode bits in STFL-LLC.**

Receiving STFL-LLC Signals. The STFL receiver employs the level converter to convert the domain of signals transferred on the low power wires. It makes use of a transition detector, consisting of an XOR gate and a flip-flop, to convert the transition signals into **1**s (Figures 11 and 12). From data wires, the result is sent to a serial-in, parallel-out shift register. On every cycle, a newly detected bit is fed to the shift register; moreover, the same bit controls the shift operation. Every **0** results in shifting the content and inserting the bit in the register; a **1**, however, disables the shift operation and overwrites the previously sampled value—which is a dummy **0**. Therefore, STFL removes all of the additional delay cycles by the transmitter at the receiver. Finally, the result is sent to the STFL-LLC decoder for extracting the original data block (Figure 10).

6 Evaluation

In this section, we first present the area, energy, and delay overheads of the STFL encoder and decoder as compared to the baseline systems. Next, the energy and performance potentials of STFL and the baseline systems will be explained.

6.1 Synthesis Results

Table 7 shows area, critical path delay, and power consumption of the encoders and decoders used for 64-bit DRAM interfaces using DBI, BD, CAFO, DESC, SETS, as well as STFL. Overall, the area, delay, power, and energy overheads of the encoders and decoders are negligible as compared with the interface circuits and data wires. However, each encoding mechanism may significantly impact the overall system performance through indirect overheads such as consuming significant static energy due to long encoding latency, degrading the energy-efficiency, and increasing the cache and memory footprint. As shown in the table, STFL consumes less area overhead compared to BD and CAFO with comparable delay and power overheads. The impacts of these delay and power overheads are evaluated in the final results.

10. Notice that a shared H-tree is used to connect the sub-banks within every bank (Figure 8). Similarly to the prior work on DESC [13], we employ a re-generator circuit for transferring the transition signals on shared data wires.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.2978826, IEEE Transactions on Computers

TABLE 7
Overhead of various encoders and decoders.

Interface	Encoder			Decoder		
	Area (μm^2)	Delay (ns)	Power (mW)	Area (μm^2)	Delay (ns)	Power (mW)
DBI-DDR	78.044	0.176	0.16	25.536	0.016	0.43
STFL-DDR	1642.284	0.831	0.49	102.144	0.071	0.71
BD-DDR	3195.358	0.74	0.38	3195.358	0.24	0.33
CAFO	2638.72	0.705	1.41	51.072	0.033	0.71

6.2 Energy

This section presents the potential energy savings of STFL as compared with the other baseline mechanisms when applied to the DRAM interface. We also assess the system energy including cache, CPU cores, and DRAM.

6.2.1 DRAM Interface Optimization

Figure 14 shows the impact of various encoding mechanisms on the DRAM interface energy including switching and termination. This experiment accounts for the additional energy required for encoding, decoding, and transferring mode bits. The proposed STFL-DDR codes reduce the DRAM interface energy by an average of 72% for all of the benchmark applications. CAFO and BD are successful in reducing the termination energy of the wires by decreasing the total number of transferred **1**s. These techniques, however, are not able to completely eliminate the termination energy, which is equal to not sending **1**s. In contrast, STFL-DDR and LPDDR3 employ unterminated wires that translate to a lower energy consumption. LPDDR3 has a limited bandwidth due to the unterminated wires; whereas, STFL-DDR recovers the bandwidth loss through STFL codes, and further reduces the wire energy by lowering the switching activity.[11] On average, STFL-DDR achieves a 17% reduction in DRAM interface energy over LPDDR3. DBI reduces the termination energy by decreasing the Hamming weight of the data blocks via bus inversion. CAFO applies a two dimensional bus invert codes to further reduce the Hamming weights. Unlike DBI and CAFO, BD encoding reduces both the number of **1**s and the bit flips by exploiting data similarity. STFL-DDR and LPDDR3 mechanisms significantly benefits from unterminated wires to reduce energy reduction for data transfer. However, STFL-DDR is able to reduce further interface energy due to reducing the switching energy via data encoding. The combination of both data encoding and exploiting unterminated wires results in superior energy savings compared to all of the baseline systems.

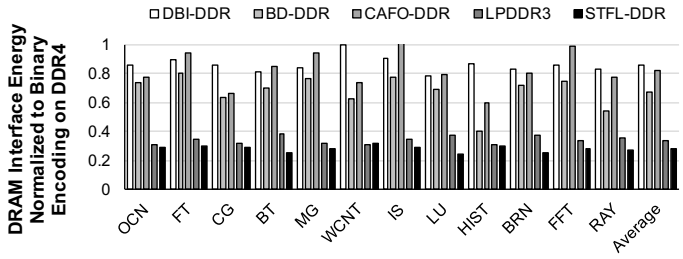

Fig. 14. **Total DRAM interface energy consumed by STFL and other baseline systems when they are applied only to DRAM.**

Figure 15 shows the overall DRAM energy when the proposed STFL -DDR and baseline mechanisms are applied to the DRAM IO wires. STFL-DDR reduces the overall DRAM energy by 26% averaged across all of the evaluated benchmark applications. CAFO and BD encoding techniques achieve 18% and 11% energy reductions, respectively. LPDDR3 achieves an average of 22% energy reduction, which is close to STFL-DDR. The main reason for the marginal improvement of STFL-DDR over LPDDR3 is the fact that most of the evaluated applications can tolerate the bandwidth degradations imposed by LPDDR3.[12] This point becomes clearer if we take a closer look at individual applications. For example, `BT` is a memory intensive application that consumes 15GBps of DRAM bandwidth. Our simulations on `BT` indicate average DRAM energy reductions of 43% and 53% for STFL-DDR and LPDDR3, respectively. As `BT`'s performance heavily depends on the DRAM bandwidth, a bandwidth degradation by LPDDR3 results in consuming more DRAM static power; therefore, a bandwidth boost through STFL-DDR coding becomes important.

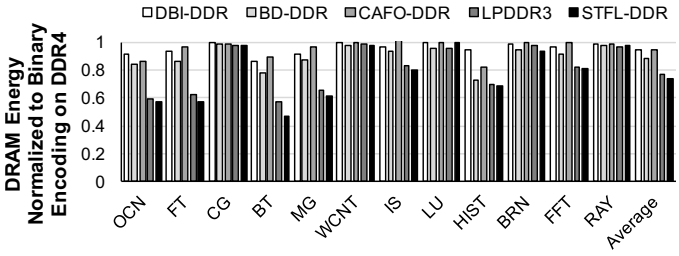

Fig. 15. **Total DRAM energy consumed by STFL and other baseline systems when they are applied only to DRAM.**

6.2.2 System Energy Optimization

To assess the overall energy saving potentials of STFL coding in computer systems, we consider applying STFL and the baseline encodings to both last level cache and DRAM interfaces. Figure 13 shows the system energy consumption of various baselines and STFL normalized to the conventional binary encoding system. The results indicate that STFL improves the average system energy by 25%; whereas, LP saves only 19% of the system energy, on average. When normalized to LP, STFL reduces the overall system energy by 8%. Both these techniques provide better system energy savings as compared with the other baselines (i.e., DBI, CAFO, and BD encoding) that rely on high performance wires.

Figure 16 shows the total system power consumed for STFL and baseline systems. STFL consumes an average power similar to LP. This proves that the efficiency of STFL comes from a reduced execution time while the power consumption is kept low.

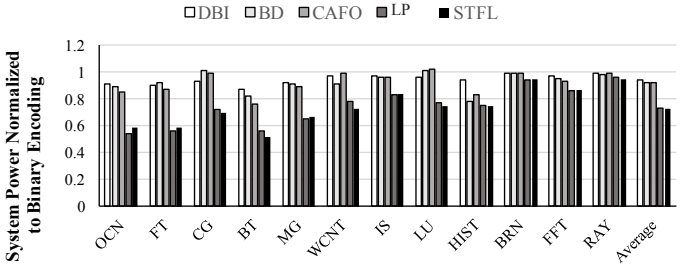

Fig. 16. **Total system power consumed by STFL and other baseline systems when they are applied to both LLC and DRAM.**

11. We observed that more energy is consumed during a switching on an unterminated wire than a terminated one.

12. This was also observed by prior work on DRAM energy proportionality [1].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.2978826, IEEE Transactions on Computers

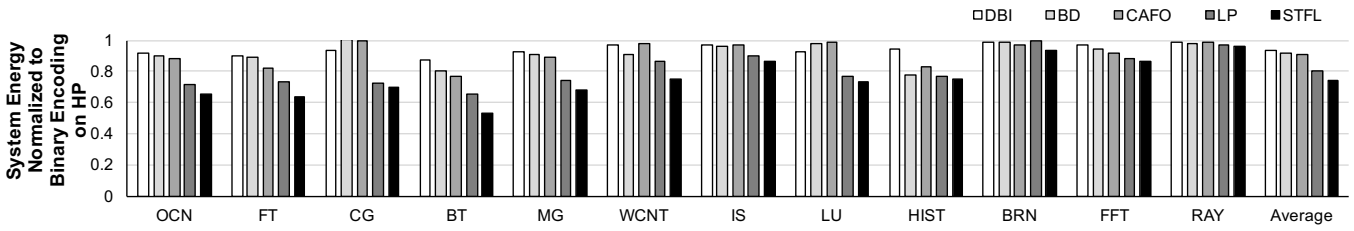

Fig. 13. **Total system energy consumed by the STFL and other baseline systems when they are applied to both LLC and DRAM.**

6.3 Performance

To evaluate the impact of STFL coding on system performance, we compute $1/execution\_time$ for all the benchmark applications in two system configurations: 1) DRAM IO is optimized, and 2) both cache and DRAM interfaces are optimized.

6.3.1 DRAM Interface Optimization

We evaluate the impact of STFL and the baselines on performance when applied to only DRAM IO interface as shown in Figure 17. This experiment shows that STFL provides almost the same performance as that of the high performance (HP) DRAM interface. For different applications, BD has between 1% to 4% better performance than STFL-DDR. Except for OCN, IS, BRN and RAY, STFL-DDR has better performance than CAFO. Notice on average, STFL-DDR and CAFO show the same performance for the evaluated benchmarks. As observed by prior work on PARDIS [54], a delay in processing requests at the memory controller may result in a different schedule that may improve or worsen the system performance within a small margin. The observed performance variance across the evaluated systems is mainly due to this processing delay.

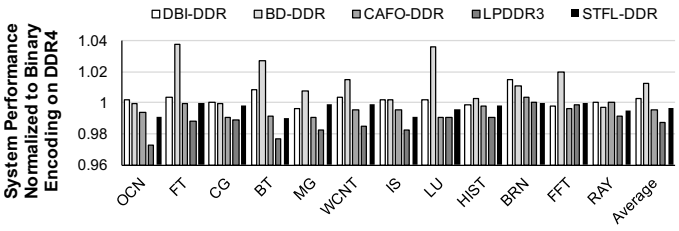

Fig. 17. **System performance of the STFL and other baseline systems when they are applied only to DRAM.**

6.3.2 LLC and DRAM Interface Optimization

Figure 18 shows the average execution time when STFL and the baseline mechanisms are used to optimize both the last level cache and DRAM interfaces. As compared to LP, STFL reduces the end-to-end execution time by an average of 7% across all the evaluated benchmarks. While LP incurs 10% increase in the overall execution time; STFL reduces the loss down to less than 2% and achieves 98% of the high-performance binary encoding baseline. CAFO and BD encoding result in excessive bandwidth overheads in the last level cache interface, thereby increasing the execution time by 4% and 3%, respectively.

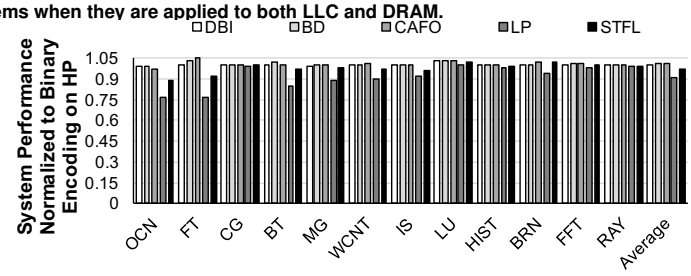

Fig. 18. **Total system performance of the STFL and other baseline systems when they are applied to both LLC and DRAM.**

6.4 Energy-Delay Product

Due to improving both energy and delay, STFL can significantly reduce the energy-delay product (EDP) of a contemporary multicore system. Figure 19 shows that STFL improves the system EDP by 15% and 25% compared to the binary encoding on LP and HP wires, respectively.

7 Discussion

7.1 Adapting STFL Coding to Data Patterns

As explained in Section 5.2.2, STFL-LLC employs three encoding modes to reduce the switching activity on the wires based on the block contents. Figure 20 shows the usage of each mode for all of the evaluated applications.[13] The results indicate that most of the transferred data bytes have low Hamming weights and chose as the best codeword.

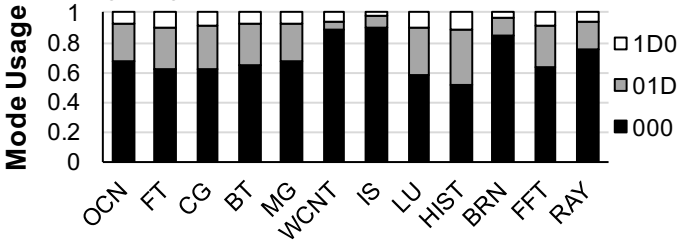

Fig. 20. **Breakdown of STFL-LLC mode usage.**

7.2 Exploiting Spatial Locality

We define*spatial locality* as the repetition of a bit value in the same bit position of neighboring bytes. As a result, XORing the adjacent bytes may lead to more zeros in codewords. Notice that STFL does not solely rely on spatial locality to reduce bit-flips. Most of the energy efficiency is due to using low power signaling and a hybrid signaling/encoding that guarantees a certain numbers of **1**s per byte. Nevertheless, we observed the following average numbers of **1**s transferred over the DRAM across the evaluated benchmark applications (Figure 21).

13. This study is also used to assign bit patterns to each mode: the more frequently used mode is assigned to the code with less switchings (**1**s).

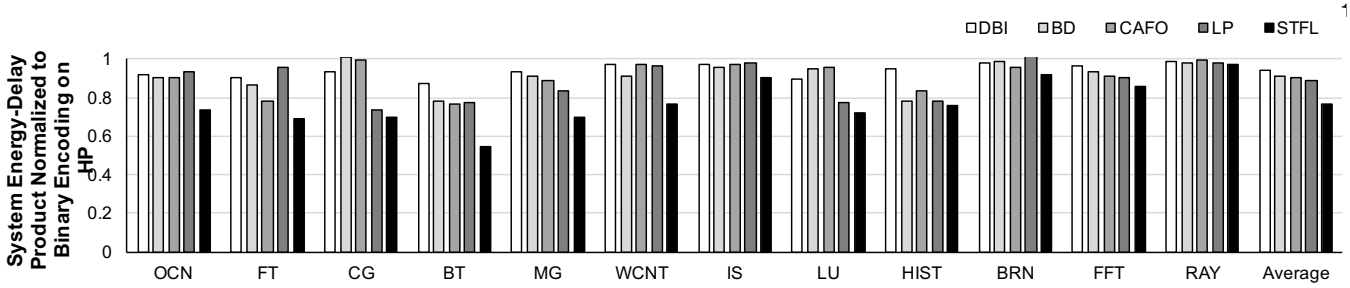

Fig. 19. **Total energy delay product of the STFL and other baseline systems when they are applied to both LLC and DRAM.**

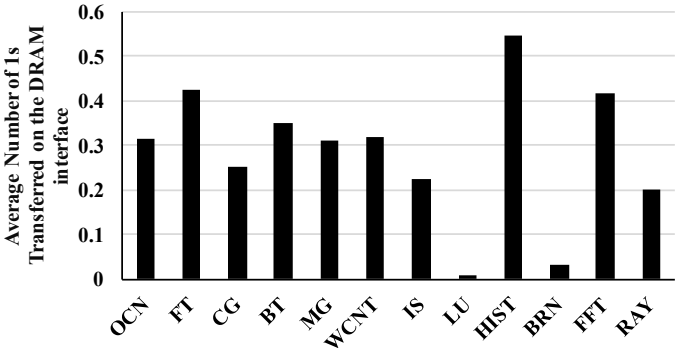

Fig. 21. **Average number of 1s of evaluated benchmarks.**

7.3 DRAM Bandwidth

Figure 22 shows DRAM bandwidth consumption for STFL-DDR and the baselines. STFL-DDR provides the same bandwidth as DDR4 due to operating at 1600MHz. However, LPDDR3 @800MHz encounters a bandwidth drop.

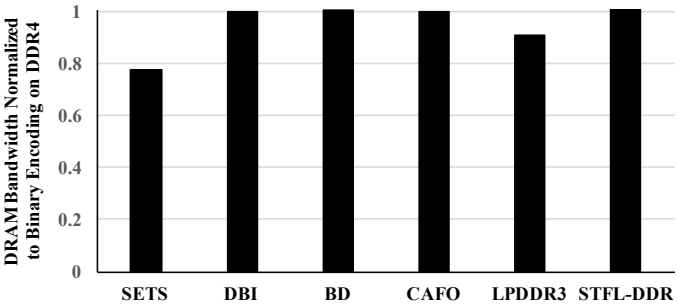

Fig. 22. **Average DRAM Bandwidth consumption for various techniques.**

7.4 Handling ECC

Modern cache and memory systems may require error-correction codes (ECC) to improve system reliability. We expect such mechanism impact an STFL interface in two ways: 1) transferring ECC codewords may increase the entropy of transferred data; and 2) a single error may impact multiple bits of a byte, which may be addressed through bit interleaving per block as explained by the prior work on DESC [13].

8 Conclusions

This paper examined a hybrid technique for slow-transition, fast-level signaling on the DRAM interface called STFL-DDR. The proposed technique improves the performance, bandwidth, and energy efficiency of low power wires compared with state of the art solutions. We also demonstrated the significant effectiveness of our proposed method when it is applied to both LLC and DRAM interface. Since data movement is the major source of energy consumption in modern computer systems, this technique can assist the designers to build more energy-efficient computer systems.

References

[1] 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 *ACM SIGARCH Computer Architecture News*, vol. 40, no. 3, 2012, pp. 37–48.

[2] I. Akturk and U. Karpuzcu, "Amnesiac: Amnesic automatic computer trading computation for communication for energy efficiency," in *ASPLOS 2017*, 2017, pp. 811–824.

[3] "The top ten exascale research challenges," in *Report of the Advanced Scientific Computing Advisory Committee Subcommittee*, 2014.

[4] V. Sze, Y.-H. Chen, J. Emer, A. Suleiman, and Z. Zhang, "Hardware for machine learning: Challenges and opportunities," in *IEEE Custom Integrated Circuits Conference (CICC)*, 2017, pp. 1–8.

[5] P. Behnam and M. N. Bojnordi, "Stfl: Energy-efficient data movement with slow transition fast level signaling," in *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, 2019, p. 42.

[6] "Jedec standard: Ddr4 sdram," *JEDEC Solid State Technology Association*, 2012.

[7] S. J. Bae *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 Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 121–131, Jan 2008.

[8] T. Y. Oh *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," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 178–190, Jan 2015.

[9] S. Dumas, "Mobile memory forum: Lpddr3 and wideio," in *JEDEC mobile forum*, vol. 201, no. 1, 2011.

[10] H. Nguyen, V. Gadde, and B. Lau, "Calibration methods and circuits for optimized on-die termination."

[11] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power i/o," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 3, no. 1, pp. 49–58, 1995.

[12] T. M. Hollis, "Data bus inversion in high-speed memory applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, no. 4, pp. 300–304, 2009.

[13] M. N. Bojnordi and E. Ipek, "Desc: Energy-efficient data exchange using synchronized counters," ser. MICRO-46, 2013, pp. 234–246.

[14] P. Behnam, N. Sedaghati, and M. N. Bojnordi, "Adaptive time-based encoding for energy-efficient large cache architectures," in *Proceedings of the 5th ACM International Workshop on Energy Efficient Supercomputing*, 2017, p. 5.

[15] S. Cho and H. Lee, "Flip-n-write: A simple deterministic technique to improve pram write performance, energy and endurance," ser. MICRO 42, 2009, pp. 347–357.

[16] R. Maddah, S. M. Seyedzadeh, and R. Melhem, "Cafo: Cost aware flip optimization for asymmetric memories," in *HPCA 2015*, 2015, pp. 320–330.

[17] Y. Song, M. N. Bojnordi, and E. Ipek, "Energy-efficient data movement with sparse transition encoding," in *ICCD*, 2015, pp. 399–402.

[18] M. R. Stan and W. P. Burleson, "Limited-weight codes for low-power i/o," in *International Workshop on low power design*, vol. 6, no. 3. Citeseer, 1994, pp. 6–8.

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

[20] H. Seol *et al.*, "Energy efficient data encoding in dram channels exploiting data value similarity," in *ISCA 2016*, 2016, pp. 719–730.

[21] S. Wang and E. Ipek, "Reducing data movement energy via online data clustering and encoding," in *MICRO*, 2016, pp. 1–13.

[22] D. Lee, M. O'Connor, and N. Chatterjee, "Reducing data transfer energy by exploiting similarity within a data transaction," in *HPCA*, 2018, pp. 40–51.

[23] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu, "Memory power management via dynamic voltage/frequency scaling," in *Proceedings of the 8th ACM international conference on Autonomic computing*, 2011, pp. 31–40.

[24] "Free PDK 45nm open-access based PDK for the 45nm technology node," http://www.eda.ncsu.edu/wiki/FreePDK.

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

[26] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45nm design exploration," in *International Symposium on Quality Electronic Design*, 2006.

[27] N. P. Jouppi, A. B. Kahng, N. Muralimanohar, and V. Srinivas, "Cacti-io: Cacti with off-chip power-area-timing models," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 7, pp. 1254–1267, 2015.

[28] "Micron ddr4 power calculator," DDR4SDRAMSystem-PowerCalculator(xlsm)-Micron.

[29] "Micron lpddr3 power calculator," http://www.micron.com/.

[30] K. Chandrasekar, C. Weis, Y. Li, B. Akesson, N. Wehn, and K. Goossens, "Drampower: Open-source dram power & energy estimation tool," *URL: http://www. drampower. info*, 2012.

[31] S. Li *et al.*, "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *MICRO*, 2009, pp. 469–480.

[32] E. K. Ardestani and J. Renau, "Esesc: A fast multicore simulator using time-based sampling," in *HPCA*, ser. HPCA '13, 2013, pp. 448–459.

[33] Micron, "Mobile lpddr3 sdram," https://www.micron.com/-/media/client/global/documents/products/data-sheet/dram/mobile-dram/low-power-dram/lpddr3/178b_8-16gb_2c0f_mobile_lpddr3.pdf.Accessed:2019-03.

[34] ——, "Twindie ddr4 sdram," https://www.micron.com/media/client/global/documents/products/data-sheet/dram/ddr4/ddr4_16gb_x16_1cs_twindie.pdf.Accessed:2019-03.

[35] B. Razavi, "Challenges in the design high-speed clock and data recovery circuits," *IEEE Communications magazine*, vol. 40, no. 8, pp. 94–101, 2002.

[36] S. Ozawa, J.-i. Okamura, Y. Ishizone, and S. Miura, "Transmitter circuit, receiver circuit, clock data recovery phase locked loop circuit, data transfer method and data transfer system," 2009, uS Patent 7,535,957.

[37] J.-H. Chae *et al.*, "A 1.74 mw/ghz 0.11–2.5 ghz fast-locking, jitter-reducing, 180 phase-shift digital dll with a window phase detector for lpddr4 memory controllers," in *2015 IEEE Asian Solid-State Circuits Conference*, 2015, pp. 1–4.

[38] J.-K. Woo *et al.*, "A fast-locking cdr circuit with an autonomously reconfigurable charge pump and loop filter," in *IEEE Asian Solid-State Circuits Conference*. IEEE, 2006, pp. 411–414.

[39] S.-K. Lee *et al.*, "A 650mb/s-to-8gb/s referenceless cdr circuit with automatic acquisition of data rate," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2009, pp. 184–185.

[40] R. F. Payne and B. Parthasarathy, "Interpolator based clock and data recovery (cdr) circuit with digitally programmable bw and tracking capability," Jan. 1 2008, uS Patent 7,315,596.

[41] D. Kim, J. Wei, Y. Frans, T. Bystrom, N. Nguyen, and K. Donnelly, "Clock-data recovery ("cdr") circuit, apparatus and method for variable frequency data," Mar. 6 2012, uS Patent 8,130,891.

[42] E. Guerrero, C. Sanchez-Azqueta, C. Gimeno, J. Aguirre, and S. Celma, "An adaptive bitrate clock and data recovery circuit for communication signal analyzers." *IEEE Trans. Instrumentation and Measurement*, vol. 66, no. 1, pp. 191–193, 2017.

[43] J. Lee and B. Razavi, "A 40 gb/s clock and data recovery circuit in 0.18/spl mu/m cmos technology," in *Solid-State Circuits Conference, 2003. Digest of Technical Papers. ISSCC. 2003 IEEE International*. IEEE, 2003, pp. 242–491.

[44] Y. Shim, D. Oh, T. Hoang, and Y. Ke, "A jitter equalization technique for minimizing supply noise induced jitter in high speed serial links," in *Electromagnetic Compatibility (EMC), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 827–832.

[45] H. Partovi *et al.*, "Data recovery and retiming for the fully buffered dimm 4.8 gb/s serial links," in *IEEE International Solid-State Circuits Conference, ISSCC 2006*. IEEE, 2006, pp. 1314–1323.

[46] R. E. Palmer, "Adaptive equalization using correlation of edge samples with data patterns," Dec. 29 2009, uS Patent 7,639,737.

[47] K. Kaviani, *et al.*, "A 6.4 gb/s near-ground single-ended transceiver for dual-rank dimm memory interface systems," in *IEEE International Solid-State Circuits Conference (ISSCC), 2013*, pp. 306–307.

[48] J.-M. Dortu and A. M. Chu, "Clock latency compensation circuit for ddr timing," Aug. 8 2000, uS Patent 6,100,733.

[49] R. M. Yoo, A. Romano, and C. Kozyrakis, "Phoenix rebirth: Scalable mapreduce on a large-scale shared-memory system," in *Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC)*, 2009, pp. 198–207.

[50] D. H. Bailey *et al.*, "The nas parallel benchmarks—summary and preliminary results," in *Proceedings of the 1991 ACM/IEEE Conference on Supercomputing*, ser. Supercomputing '91, 1991, pp. 158–165.

[51] S. C. Woo *et al.*, "The splash-2 programs: Characterization and methodological considerations," in *ISCA 95*, 1995, pp. 24–36.

[52] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "Cacti 6.0: A tool to model large caches," *HP Laboratories*, pp. 22–31, 2009.

[53] A. N. Udipi, N. Muralimanohar, and R. Balasubramonian, "Non-uniform power access in large caches with low-swing wires," in *2009 International Conference on High Performance Computing (HiPC)*, Dec 2009, pp. 59–68.

[54] M. N. Bojnordi and E. Ipek, "Pardis: A programmable memory controller for the ddrx interfacing standards," in *39th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2012, pp. 13–24.

Payman Behnam received his B.S. and M.S. degrees with distinction in computer science and engineering from Shiraz University, Iran and the University of Tehran, Iran. Currently, he is a graduate research assistant in School of Computing at the University of Utah, UT, USA. His research centers on designing energy-efficient system design and building hardware accelerators for computer vision and machine learning accelerators.

Mahdi Nazm Bojnordi received the Ph.D. degree from the University of Rochester, Rochester, NY, USA, in 2016 in electrical and computer engineering. He is currently an Assistant Professor of School of Computing with the University of Utah, Salt Lake City, UT, USA, where he leads the Energy-Efficient Computer Architecture Laboratory (ECAL). His current research interests include energy-efficient architectures, low-power memory systems, and the application of emerging memory technologies to computer systems. Professor Bojnordi's research has been recognized by two IEEE Micro Top Picks Awards, an HPCA 2016 Distinguished Paper Award, and a Samsung Best Paper Award.