ReDRAM: A Reconfigurable Processing-in-DRAM Platform for Accelerating Bulk Bit-Wise Operations

Shaahin Angizi[†] and Deliang Fan[§]
[†]University of Central Florida, Orlando, FL 32816
[§]Arizona State University, Tempe, AZ 85287
angizi@knights.ucf.edu,dfan@asu.edu

Abstract— **In this paper, we propose *ReDRAM*, as a reconfigurable DRAM-based processing-in-memory (PIM) accelerator, which transforms current DRAM architecture to massively parallel computational units exploiting the high internal bandwidth of modern memory chips. *ReDRAM* uses the analog operation of DRAM sub-arrays and elevates it to implement a full set of 1- and 2-input bulk bit-wise operations (NOT, (N)AND, (N)OR, and even X(N)OR) between operands stored in the same bit-line, based on a new dual-row activation mechanism with a modest change to peripheral circuits such sense amplifiers. *ReDRAM* can be leveraged to greatly reduce energy consumption and latency of complex in-DRAM logic computations relying on state-of-the-art mechanisms based on triple-row activation, dual-contact cells, row initialization, NOR style, etc. The extensive circuit-architecture simulations show that *ReDRAM* achieves on average 54× and 7.1× higher throughput for performing bulk bit-wise operations compared with CPU and GPU, respectively. Besides, *ReDRAM* outperforms recent processing-in-DRAM platforms with up to 3.7× better performance.**

I. Introduction

In the last two decades, Processing-in-Memory (PIM) architecture, as a potentially viable way to solve the memory wall challenge, has been well explored for different applications [1]–[3]. The key concept behind PIM is to realize logic computation within memory to process data by leveraging the inherent parallel computing mechanism and exploiting large internal memory bandwidth. The proposals for exploiting SRAM-based [4] PIM architectures can be found in recent literature. However, PIM in context of main memory (DRAM- [2], [3]) has drawn much more attention in recent years mainly due to larger memory capacities and off-chip data transfer reduction as opposed to SRAM-based PIM. Such processing-in-DRAM platforms show significantly higher throughputs leveraging multi-row activation methods to perform bulk bit-wise operations by either modifying the DRAM cell and/or sense amplifier. For example, Ambit [2] uses triple-row activation method to implement majority-based AND/OR logic, outperforming Intel Skylake-CPU, NVIDIA GeForce GPU, and even HMC [5] by 44.9×, 32.0×, and 2.4×, respectively. DRISA [3] employs 3T1C- and 1T1C-based computing mechanisms and achieves 7.7× speedup and 15× better energy-efficiency over GPUs to accelerate convolutional neural networks. However, there are different challenges in such platforms that make them inefficient acceleration solutions for more complex bulk bit-wise logic implementations, e.g. X(N)OR- and addition-based applications such as data-encryption [6], DNA alignment [7], and graph processing [8], [9]. The need for multi-cycle operations, row initialization and add-on dual-contact cells as well as reliability concerns due to multiple row activation mechanisms, etc. motivated us to explore alternative circuit-architecture solutions.

In this work, we propose a high-throughput, reconfigurable PIM accelerator based on DRAM, called *ReDRAM*. The proposed platform exploits a new in-memory computing mechanism called Dual-Row Activation mechanism (DRA) to perform a full-set of 1- and 2-input bulk bit-wise operations (NOT, (N)AND, (N)OR, and X(N)OR) between operands stored in different word-lines. The DRA is developed based on analog operation of DRAM sub-arrays with a modest change in the sense amplifier circuit such that different operations can be efficiently realized on every memory bit-line. This mechanism alleviates the reliability concerns of the voltage deviation on the bit-line and eliminates the need for dual-contact cell rows and multi-cycle operations of counterpart designs [2], [3]. Our main contributions in this work can be summarized as follows.

1) To the best of our knowledge, *ReDRAM* is the first processing-in-DRAM platform that proposes a high-throughput and energy-efficient acceleration solution exploiting DRAM arrays based on dual-row activation mechanism to realize a full-set of bulk bit-wise operations. We develop *ReDRAM* based on a set of novel microarchitectural and circuit-level schemes to realize massive data-parallel computational unit for different applications; 2) We extensively evaluate and compare *ReDRAM*'s performance with various conventional and PIM computing platforms, i.e. a Core-i7 Intel CPU [10], an NVIDIA GTX 1080Ti GPU [11], Ambit [2], DRISA-1T1C [3], and HMC [5], to handle bulk bit-wise operations; 3) We provide case studies of how important graph processing and data encryption workloads can be partitioned and mapped to our architecture and how they can benefit from it.

II. Background

A. Processing-in-DRAM Mechanisms

A DRAM hierarchy at the top level is composed of channels, modules, and ranks. Each memory rank, with a data bus typically 64-bits wide, includes a set of memory chips that are manufactured with a variety of configurations and operate in unison [2]. Each chip is further divided into multiple memory banks that contains 2D sub-arrays of memory cells virtually-organized in memory matrices (mats). Banks within same chips share I/O, buffer and banks in different chips working in a lock-step manner. Each memory sub-array, as shown in Fig. 1a, has 1) a large number of rows (typically 2^9 or 2^{10}) holding DRAM cells, 2) a row of Sense

978-1-7281-2350-9/19/$31.00 ©2019 IEEE

Authorized licensed use limited to: University of Central Florida. Downloaded on June 23,2020 at 18:35:10 UTC from IEEE Xplore. Restrictions apply.

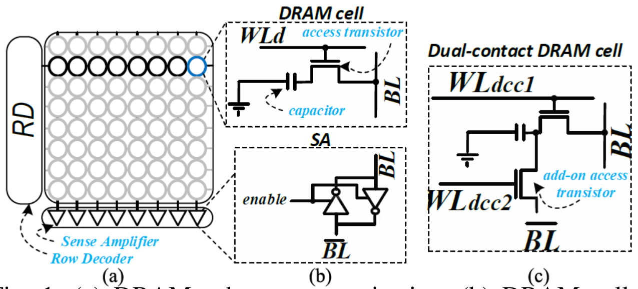


Fig. 1: (a) DRAM sub-array organization, (b) DRAM cell and Sense Amplifier, (c) Dual-contact DRAM cell.

Amplifiers (SA), and 3) a Row Decoder (RD) connected to the cells. A DRAM cell basically consists of two elements, a capacitor (storage) and an Access Transistor (AT) (Fig. 1b). The drain and gate of the AT is connected to the Bit-line (BL) and Word-line (WL), respectively. DRAM cell encodes the binary data by the charge of the capacitor. It represents logic '1' when the capacitor is full-charged, and logic '0' when there is no charge.

•Write/Read Operation: At initial state, both BL and $\overline{BL}$ is always set to $\frac{V_{dd}}{2}$. Technically, accessing data from a DRAM's sub-array (write/read) after initial state is done through three consecutive commands [2] issued by the memory controller: 1) During the activation (i.e. ACTIVATE), activating the target row, data is copied from the DRAM cells to the SA row. Fig. 1b shows how a cell is connected to a SA via a BL. The selected cell (storing V_{dd} or 0) shares its charge with the BL leading to a small change in the initial voltage of BL ($\frac{V_{dd}}{2} \pm \delta$). Then, by activating the $enable$ signal, the SA senses and amplifies the δ of the BL voltage towards the original value of the data through voltage amplification according to the switching threshold of SA's inverter [12]. 2) Such data can be then transferred from/to SA to/from DRAM bus by a READ/WRITE command. In addition, multiple READ/WRITE commands can be issued to one row. 3) The PRECHARGE command precharges both BL and $\overline{BL}$ again and makes the sub-array ready for the next access.

•Copy and Initialization Operations: To enable a fast ($< 100ns$) in-memory copy operation within DRAM sub-arrays, rather than using $\sim 1\mu s$ conventional operation in Von-Neumann computing systems, *RowClone*-Fast Parallel Mode (FPM) [13] proposes a PIM-based mechanism that does not need to send the data to the processing units. In this scheme, two back-to-back ACTIVATE commands to the source and destination rows without PRECHARGE command in between, leads to a multi-kilo byte in-memory copy operation. This operation takes only $90ns$ [13]. This method has been further used for row initialization, where a preset DRAM row (either to '0' or '1') can be readily copied to a destination row. RowClone imposes only a 0.01% overhead to DRAM chip area [13].

•Not Operation: The NOT function has been implemented in different works employing Dual-Contact Cells (DCC), as shown Fig. 1c. DCC is mainly designed based on typical DRAM cell, but equipped with one more AT connected to $\overline{BL}$. Such hardware-friendly design [2], [14], [15] can be developed for a small number of rows on top of existing DRAM cells to enable efficient NOT operation with issuing two back-to-back ACTIVATE commands [2]. In this way, the memory controller first activates the WL_{dcc1} (Fig. 1c) of input DRAM cell, and reads the data out to the SA through BL. It then activates WL_{dcc2} to connect $\overline{BL}$ to the same capacitor and so writes the negated result back to the DCC.

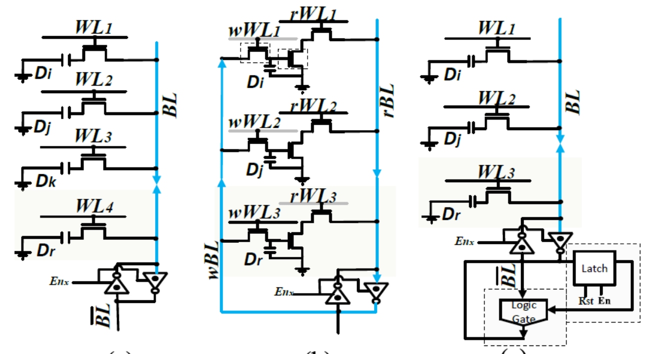


Fig. 2: (a) Ambit's TRA [2], (b) DRISA's 3T1C [3], (c) DRISA's 1T1C-logic [3]. (Glossary- D_i/D_j: input rows data, D_k: initialized row data, D_r result row data.)

•Other Logic Operations: To realize the logic function in DRAM platform, *Ambit* [2] extends the idea of RowClone by implementing 3-input majority function (Maj3)-based operations in memory by issuing the ACTIVATE command to three rows simultaneously followed by a single PRECHARGE command, so-called *Triple Row Activation (TRA)* method. As shown in Fig. 2a, considering one row as the control, initialized by D_k= '0'/'1', Ambit can readily implement in-memory AND2/OR2 in addition to Maj3 functions through charge sharing between connected cells (D_k, D_i and D_j) and write back the result on D_r cell. It also leverage TRA mechanism along with DCCs to realize the complementary functions. However, despite Ambit shows only 1% area over commodity DRAM chip [2], it suffers from multi-cycle PIM operations to implement other functions such as XOR2/XNOR2 based on TRA. Alternatively, *DRISA*-3T1C method [3] utilizes the early 3-transistor DRAM design [16], in which the cell consists of two separated read/write ATs, and one more transistor to decouple the capacitor from the read BL (rBL), as shown in Fig. 2b. This transistor connects the two DRAM cells in a NOR style on the rBL naturally performing functionally-complete NOR2 function. However, *DRISA*-3T1C imposes very large area overhead (2T per cell) and still requires multi-cycle operations to implement more complex logic functions. *DRISA*-1T1C method [3] offers to perform PIM through upgrading the SA unit by adding a CMOS logic gate in conjunction with a latch, as depicted in Fig. 2c. Such inherently-multi-cycle operation can enhance the performance of a single function through add-on CMOS circuitry, in two consecutive cycles. In first cycle, D_i is read out and stored in the latch, and in the second cycle, D_j is sensed to perform the computation. However, this design imposes excessive cycles to implement other logic functions

Authorized licensed use limited to: University of Central Florida. Downloaded on June 23,2020 at 18:35:10 UTC from IEEE Xplore. Restrictions apply.

and at least 12 transistors to each SA.

B. Challenges and Motivations

There are four main challenges in the existing processing-in-DRAM platforms that this work aims to address:

Low Reliability (Challenge-1): By simultaneously activating three cells in TRA method, the deviation on the BL might be smaller than typical one-cell read operation in DRAM. This can elongate the sense amplification state or even adversely affect the reliability of the result. The problem can be even intensified when multiple TRA are needed to implement more complex functions. **Row Initialization (Challenge-2):** Given R=A$op$B function ($op \in$ `AND2/OR2`), TRA-based method [2], [12] takes 4 consecutive steps to calculate one result: 1-RowClone data of row A to row D_i (Copying first operand to a computation row to avoid data-overwritten); 2-RowClone of row B to D_j; 3-RowClone of ctrl row to D_k (Copying initialized control row to a computation row); 4-TRA and RowClone data of row D_i to R row (Computation and Writing-back the result). So, TRA method needs at least 360ns to perform such in-memory operations. **DCC Rows (Challenge-3):** Almost all the existing PIM platforms rely on DCC rows to realize in-memory `NOT` operation. This could impose excessive area overhead to the sub-array and complicate control circuitry. Besides, DCC is not a favorable solution fabrication-wise, as it complicates DRAM fab process considering two types of DRAM cell in every sub-array. **Limited Throughput X(N)OR (Challenge-4):** Due to the intrinsic complexity of X(N)OR-based logic implementations, current PIM designs (such as Ambit [2], DRISA [3], and Dracc [17]) are not able to offer a high-throughput and area-efficient X(N)OR or addition in-memory operation despite utilizing maximum internal DRAM bandwidth and memory-level parallelism for NOT, (N)AND, (N)OR, and MAJ/MIN logic functions. Moreover, while DRISA-1T1C method could implement either XNOR or XOR functions as the add-on logic gate, it requires at least two consecutive cycles to perform the computation, which in turn limits other logic implementation.

III. The Reconfigurable PIM Platform

A. Architecture

ReDRAM is designed to be an independent, high-performance, energy-efficient accelerator based on main memory architecture to accelerate a wide variety of applications. The main memory organization of *ReDRAM* is shown in Fig. 3a based on typical DRAM hierarchy. Each mat consists of multiple computational memory sub-arrays connected to a Global Row Decoder (GRD) and a shared Global Row Buffer (GRB). According to the physical address of operands within memory, *ReDRAM*'s Controller (Ctrl) is able to configure the sub-arrays to perform data-parallel intra-sub-array computations. We divide the *ReDRAM*'s sub-array row space into two distinct regions as depicted in Fig. 3b: 1- Data rows (1016 rows out of 1024) connected to a regular Row Decoder (RD), and 2- Computation rows (8-labeled by $x1, ..., x8$), connected to a Modified Row Decoder (MRD), which enables dual row activation required for bulk bit-wise in-memory operations between operands. *ReDRAM*'s computational sub-array is developed to perform a full-set of bit-wise operations based on the proposed *Dual-Row Activation* (DRA) mechanism leveraging charge-sharing among different rows, as discussed below.

B. Dual Row Activation Mechanism

With a careful observation on the existing processing-in-DRAM platforms, we realized that they are dealing with different challenges which could be alleviated by rethinking about SA circuit. Our key idea is to perform in-memory logic operations through a DRA mechanism to address all four challenges discussed in Section 2.2. To achieve this goal, we propose a new reconfigurable SA, as shown in Fig. 3c, developed on top of existing DRAM circuitry. It consists of a regular DRAM SA equipped with add-on circuits including three inverters, one NAND gate, and one MUX, controlled with five enable signals ($En_M, En_x, En_{mux}, En_{C1}, En_{C2}$). This design leverages the basic charge-sharing feature of DRAM cell and elevates it to implement `NOT`/`AND2`/`OR2`/`XOR2` logic between two selected rows through static capacitive functions in a single cycle. To implement capacitor-based logic, we use two different inverters with shifted Voltage Transfer Characteristic (VTC), as shown in Fig. 4a. In this way, a NAND/NOR logic can be readily carried out based on high switching voltage (V_s)/low-V_s inverters with standard high-V_{th}/low-V_{th} NMOS and low-V_{th}/high-V_{th} PMOS transistors. It is worth mentioning that utilizing low/high-threshold voltage transistors along with normal-threshold transistors have been accomplished in low-power application, and many circuits have enjoyed this technique in low-power design [18], [19].

Fig. 5 gives the detailed control signals of *ReDRAM*'s sub-array to implement different memory and in-memory logic functions (here, e.g. `X(N)OR2`). *ReDRAM*'s ctrl activates En_M and En_x control-bits simultaneously (when MUX is deactivated-En_{mux}=0) to perform typical memory write/read operation. It is worth noting that in such memory operations, MUX's output voltage is high-z and $\overline{BL}$ voltage is solely determined in sense amplification state through two normal-V_s back-to-back inverters, just like normal DRAM's SA mechanism. Therefore, *ReDRAM* can perform the bulk copy operation based on RowClone mechanism, as discussed earlier. Now, consider D_i and D_j operands (in Fig. 5b) are RowCloned from data rows to $x1$ and $x2$ computational rows and both BL and $\overline{BL}$ are precharged to $\frac{V_{dd}}{2}$ (Precharged State). *ReDRAM*'s ctrl first activates two WLs in computational row space (here, $x1$ and $x2$) through the modified decoder for charge-sharing when all the other enable signals are deactivated. During sense amplification state, by setting the proper enable set ($En_M, En_x, En_{mux}, En_{C1}, En_{C2}$), tabulated in Fig. 5a (01111 for `X(N)OR2`), the input voltage of both low- and high-V_s inverters in the reconfigurable SA can be simply derived as $V_i = \frac{n.V_{dd}}{C}$, where n is the number of DRAM cells storing logic '1' and C represents the total number of unit capacitors connected to the inverters (i.e. 2

Authorized licensed use limited to: University of Central Florida. Downloaded on June 23,2020 at 18:35:10 UTC from IEEE Xplore. Restrictions apply.

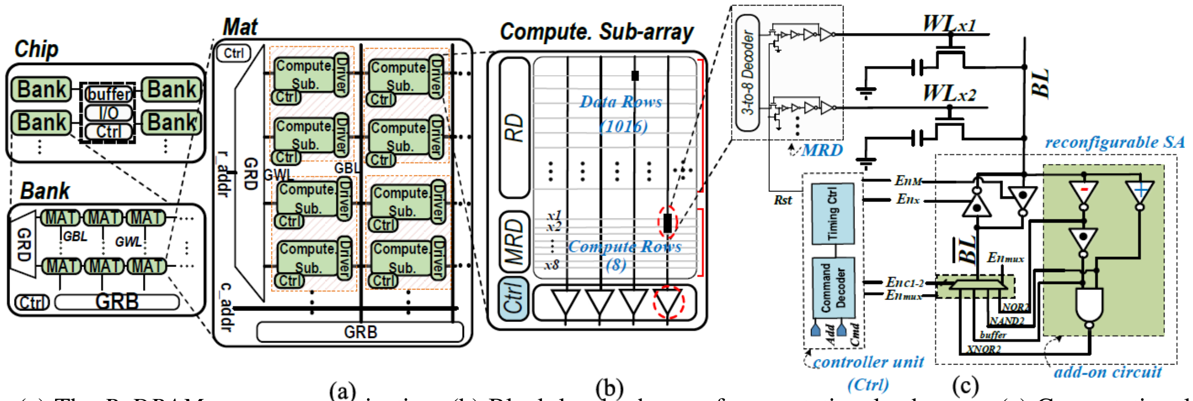
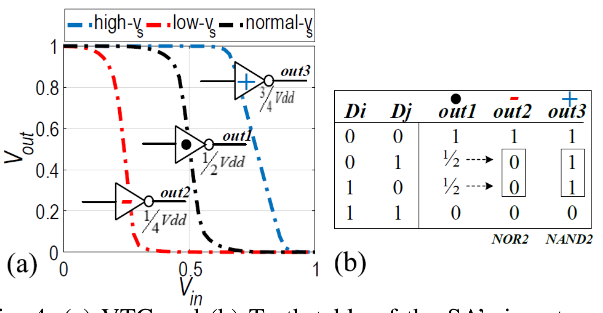


Fig. 3: (a) The *ReDRAM* memory organization, (b) Block level scheme of computational sub-array, (c) Computational Rows and reconfigurable SA.

in DRA mechansim). Now, the low-V_s inverter acts as a threshold detector by amplifying deviation from $\frac{1}{4}V_{dd}$ and realizes a NOR2 function, as tabulated in the truth table in Fig. 4b. At the same time, the high-V_s inverter amplifies the deviation from $\frac{3}{4}V_{dd}$ and realizes a NAND2 function. Accordingly, XNOR2 function of input operands can be realized after CMOS NAND gate. Now, *ReDRAM*'s MUX can be readily reconfigured through the selectors to assign NOR2/NAND2/buffer/XNOR2 value and its complementary logic to $\overline{BL}$ and BL, respectively. As can be seen, by setting enable set to 01111, XOR2 result can be produced in a single cycle on the BL.

The transient voltage simulation results of DRA mechanism to realize single-cycle in-memory operations is shown in Fig. 6. We can observe how NOR2/NAND2/XNOR2 function is produced for two inputs (D_i and D_j). In this case, MUX's selectors are configured to set $\overline{BL}$ voltage with XNOR2 result (Fig. 6). We can see that cell's capacitor is accordingly charged to V_{dd} (when D_iD_j=10/01) or discharged to GND (when D_iD_j=00/11) during sense amplification state. Therefore, DRA mechanism can effectively provide single-cycle logic functions (NOT, AND, OR, XOR) and two-cycle complementary logics to address the *challenge*-2 and -3 discussed in Section 2.2 by eliminating the need for the row initialization and DCC Rows. Note that, NOT function is readily realized on the BL by selecting the corresponding MUX selectors (01110). In addition, *ReDRAM* can perform more complex in-memory logic functions (such as XOR2) in a single memory cycle not relying on multiple TRA-based [2] or NOR-based [3] operations.



Di	Dj	● out1	- out2	+ out3
0	0	1	1	1
0	1	½	0	1
1	0	½	0	1
1	1	0	0	0

Fig. 4: (a) VTC and (b) Truth table of the SA's inverters to realize capacitive NAND2-NOR2 functions.

C. Software Support

From a programmer perspective, *ReDRAM* is more of a third party accelerator that can be connected directly to the memory bus or through PCI-Express lanes rather than a memory unit. Therefore, a virtual machine and ISA for general-purpose parallel thread execution need to be defined. Accordingly, the programs are translated at install time to the *ReDRAM* hardware instruction set discussed here to realize the functions tabulated in Table I. Additionally, Table I lists the corresponding function implementation in Ambit [2] and DRISA [3] platforms. The micro and control transfer instructions are not discussed here.

TABLE I: The basic functions supported by *ReDRAM*, Ambit and DRISA.

		Command Sequence		
Func.	Operation	ReDRAM	Ambit [2]	DRISA‡ [3]
Copy	$D_r \leftarrow D_i$	AAP(D_i, D_r)†	AAP(D_i, D_r)	AP$(D_i, Latch)$ AP$(Latch, D_r)$
NOT	$D_r \leftarrow \overline{D_i}$	AAP$(D_i, D_r, 10)$	AAP$(D_i, dcc2)$ AAP$(dcc1, D_r)$	AAP$(D_i, dcc2)$ AAP$(dcc1, D_r)$
AND	$D_r \leftarrow D_i.D_j$	AAP$(D_i, x1)$ AAP$(D_j, x2)$ AAP$(x1, x2, D_r, 01)$	AAP$(D_i, x1)$ AAP$(D_j, x2)$ AAP$(0, x3)$* AAP$(x1, x2, x3, D_r)$**	AP$(D_i, Latch)$ AAP$(D_j, x1)$ AAP$(Latch, x1, D_r)$
OR	$D_r \leftarrow D_i + D_j$	AAP$(D_i, x1)$ AAP$(D_j, x2)$ AAP$(x1, x2, D_r, 00)$	AAP$(D_i, x1)$ AAP$(D_j, x2)$ AAP$(1, x3)$* AAP$(x1, x2, x3, D_r)$**	N/A
XOR2	$D_r \leftarrow D_{ij}$	AAP$(D_i, x1)$ AAP$(D_j, x2)$ AAP$(x1, x2, D_r, 11)$	AAP$(D_i, x1, dcc2)$ AAP$(D_j, x2, dcc4)$ AAP$(0, x3, x4)$* AP$(dcc1, x2, x3, x2)$** AP$(dcc3, x1, x4, x1)$** AAP$(1, x3)$* AAP$(x1, x2, x3, D_r)$**	N/A

† Size of input vectors are not shown here. ‡ DRISA's 1T1C-logic is realized with add-on AND2 gate. Therefore, it can not implement other functions. *Row initialization steps. **TRA steps.

ReDRAM is developed based on ACTIVATE-ACTIVATE-PRECHARGE command a.k.a. AAP primitives and most bulk bit-wise operations involve a sequence of AAP commands. To enable processor to efficiently communicate with *ReDRAM*, we developed two types of AAP-based instructions:

1- AAP (src, des, size) that runs the following commands sequence: 1) ACTIVATE a source address (src); 2) ACTIVATE a destination address (des); 3) PRECHARGE to prepare the array for the next access. The size of input vectors for in-memory computation must be a multiple of DRAM row size, otherwise the application must pad it with dummy data. The type-1 instruction is mainly used for *copy* function;

2- AAP (src1, src2, des, opcode, size) that performs DRA method by activating two source

Authorized licensed use limited to: University of Central Florida. Downloaded on June 23,2020 at 18:35:10 UTC from IEEE Xplore. Restrictions apply.

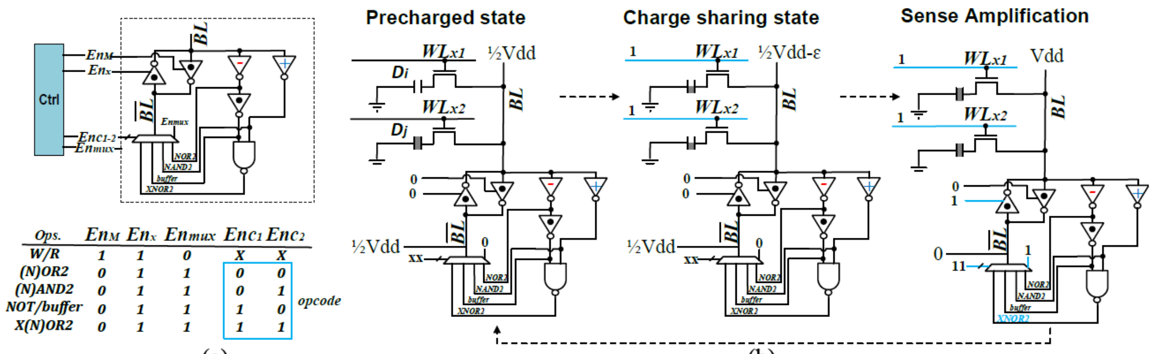

Ctrl
Ops. | En_M | En_x | En_{mux} | Enc_1 | Enc_2
W/R | 1 | 1 | 0 | X | X
(N)OR2 | 0 | 1 | 1 | 0 | 0
(N)AND2 | 0 | 1 | 1 | 0 | 1
NOT/buffer | 0 | 1 | 1 | 1 | 0
X(N)OR2 | 0 | 1 | 1 | 1 | 1
opcode
(a)
Precharged state
Charge sharing state
Sense Amplification
½Vdd
½Vdd-ε
Vdd
(b)

Fig. 5: (a) *ReDRAM*'s control signals and activations in the sense amplification state, (b) Dual Row Activation mechanism. Here, `X(N)OR2` is implemented by setting enable set $(En_M, En_x, En_{mux}, En_{C1}, En_{C2})$ to 01111.

addresses (`src1` and `src2`) and then writes back the result on a destination address (`des`) according to the opcode. Here opcode corresponds to the MUX's selectors (En_{C1} and En_{C2}), as shown in Fig. 5a.

For instance, in order to implement the `XOR2`-in-memory, as tabulated in Table I, *ReDRAM* first copies the input operands from data rows to computational rows in two consecutive cycles using `AAP`-type-1 and then perform the operation in a single cycle using `AAP`-type-2 by setting the opcode to 11. The similar operation requires at least 7 consecutive cycles based on Ambit's TRA mechanism.

D. Reliability

We performed a comprehensive circuit-level simulation to study the effect of process variation on both DRA and TRA methods considering different noise sources and variation in all components including DRAM cell (BL/WL capacitance and transistor, shown in Fig. 7) and SA (width/length of transistors-V_s). We ran Monte-Carlo simulation with 45nm NCSU Product Development Kit (PDK) library [20] in Cadence Spectre (DRAM cell parameters were taken and scaled from Rambus [21]) under 10000 trials and increased the amount of variation from ±0% to ±30% for each method. Table II shows the percentage of the test error in each variation. We observe that even considering a significant ±10% variation, the percentage of erroneous DRA across 10000 trials is *zero*, where TRA method shows a failure with 0.18%. Therefore, *ReDRAM* offers a solution to alleviate *challenge*-1 by showing an acceptable voltage margin in performing operations based on DRA mechanism. By scaling down the transistor size, the process variation effect is expected to get worse [2], [13]. Since *ReDRAM* is mainly developed based on existing DRAM structure and operation with slight modifications, different methods currently-used to tackle process variation can be also applied for *ReDRAM*.

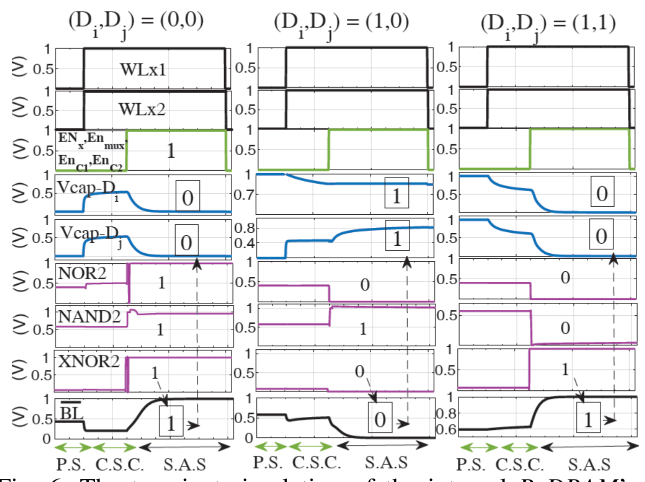


Fig. 6: The transient simulation of the internal *ReDRAM*'s sub-array signals in DRA mechanism.

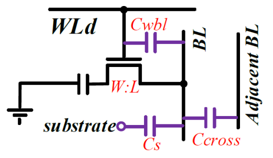


Fig. 7: Noise sources in DRAM cell. Glossary: $Cwbl$, Cs, and $Ccross$ are WL-BL, BL-substrate, and BL-BL capacitance, respectively.

Variation	TRA	DRA
±5%	0.00	0.00
±10%	0.18	0.00
±15%	5.5	1.2
±20%	17.1	9.6
±30%	28.4	16.4

TABLE II: Process variation analysis.

IV. Raw Performance

To assess the performance of *ReDRAM* as a new PIM platform, a comprehensive circuit-architecture evaluation framework and two in-house simulators are developed. 1- At the circuit level, we developed *ReDRAM*'s sub-array with new peripheral circuity (SA, MRD, etc.) in Cadence Spectre with 45nm NCSU Product Development Kit (PDK) library [20] in to verify the DRA mechanism and achieve the performance parameters. 2- An architectural-level simulator is built on top of Cacti [22]. The circuit level results were then fed into our simulator. It can change the configuration files corresponding to different array organization and report performance metrics for `AAP`-based PIM operations. The memory controller circuits are designed and synthesized by Design Compiler with a 45nm industry library. 3- A behavioral-level simulator is developed in Matlab to calculate the latency and energy that *ReDRAM* spends on different tasks. Besides, it has a mapping optimization framework to maximize the performance according to the available resources.

Authorized licensed use limited to: University of Central Florida. Downloaded on June 23,2020 at 18:35:10 UTC from IEEE Xplore. Restrictions apply.

A. Throughput

We evaluate and compare the *ReDRAM*'s raw performance with different computing units and accelerators including a Core-i7 Intel CPU [10] and an NVIDIA GTX 1080Ti Pascal GPU [11]. In PIM domain, we shall restrict our comparison to four recent processing-in-DRAM platforms, Ambit [2], DRISA-1T1C [3], DRISA-3T1C [3], and HMC 2.0 [5], to handle four bulk bit-wise operations, i.e. `NOT`, `AND2`, `OR2`, and `XOR2`. To have a fair comparison, we report *ReDRAM*'s and other PIM platforms' raw throughput implemented with 8 banks with 1024×256 computational sub-arrays. The Intel CPU consists of 4 cores and 8 threads working with two 64-bit DDR4-1866/2133 channels. The Pascal GPU has 3584 CUDA cores running at 1.5GHz [11] and 352-bit GDDR5X. The HMC has 32-10 GB/s bandwidth vaults. Accordingly, we develop an in-house micro-benchmark to run the operations repeatedly for $2^{27}/2^{28}/2^{29}$-bit length input vectors and report the throughput of each platform, as shown in Fig. 8a-d.

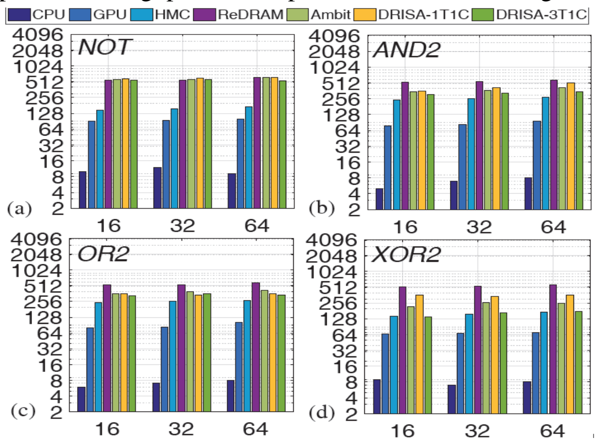


Fig. 8: Throughput of (a) `NOT`, (b) `AND2`, (c) `OR2`, and (d) `XOR2` operations implemented by different platforms. X-axis: Vector Size (MB) and Y-axis: Log Scaled Throughput (GOps/second).

We observe that 1) either the external or internal DRAM bandwidth has limited the throughput of the CPU, GPU, and even HMC platforms. However, HMC outperforms the CPU and GPU with ∼25× and 6.5× higher performance on average for bulk bit-wise operations. Besides, PIM platforms achieve remarkable throughput compared to Von-Neumann computing systems (CPU/GPU) through unblocking the data movement bottleneck. *ReDRAM* shows on average 54× and 7.1× better throughput compared to CPU and GPU, respectively. 2) While the *ReDRAM*, Ambit, and DRISA platforms achieve almost the same performance on performing bulk bit-wise `NOT` function, shown in Fig. 8a, *ReDRAM* outperforms other PIMs in performing `AND2`,`OR2`, and `XOR2` operations. As for `XOR2`, our platform improves the throughput on average by 2.3×, 1.9×, 3.7× compared with Ambit [2], DRISA-1T1C [3], and DRISA-3T1C [3], respectively. This mainly comes from the DRA mechanism that eliminates the need for row the initialization in Ambit and multi-cycle DRISA mechanism. Note that, the add-on logic of DRISA-1T1C is developed with the corresponding logic in the plots [3], however, in practice only one single logic can be accelerated with this platform. That is why DRISA-1T1C shows the second best performance in performing bulk bit-wise `XOR2` operation. To sum it up, *ReDRAM*'s DRA mechanism could effectively address *challenge*-4 by proposing the high-through bulk bit-wise X(N)OR operation.

B. Energy

We estimate the energy that DRAM chip consumes to perform the four bulk bit-wise operations per Kilo-Byte for *ReDRAM*, Ambit [2], DRISA-3T1C [3], and CPU[1]. Fig. 9 shows that *ReDRAM* achieves 2.6× and 2.8× energy reduction over Ambit [2] and DRISA-3T1C [3], respectively, to perform bulk bit-wise `XOR2` operation. Besides, compared with copying data through the DDR4 interface, *ReDRAM* reduces the energy by ∼80×. As for bit-wise in-memory `AND2` operation, *ReDRAM* outperforms TRA-based Ambit, NOR-based DRISA-3T1C, and CPU, respectively, with ∼2.1×, 1.9×, and 82× reduction in energy consumption.

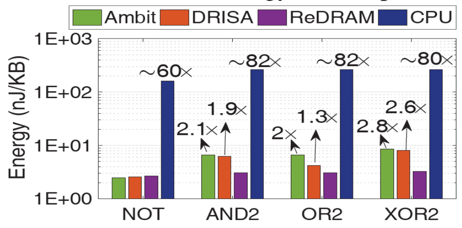


Fig. 9: Energy of different platforms (Y-axis: log scale).

C. Area Overhead

To estimate the area overhead of *ReDRAM* on top of commodity DRAM chip, three hardware cost sources must be taken into consideration. First, add-on transistors to SAs; in our design, each SA requires 30 additional transistors connected to each BL. Second, the 3:8 MRD overhead; we modify each WL driver by adding two more transistors in the typical buffer chain, as depicted in Fig. 3c, so there is only 16 add-on transistors for computational rows. Third, the Ctrl's overhead to control enable bits; ctrl generates the activation bits with MUX units with 6 transistors. To sum it up, *ReDRAM* imposes 31 DRAM rows (31×256 transistors) per sub-array, at the most, which can be interpreted as $\sim 14\%$ of DRAM chip area. Note that Ambit design requires DCC rows with two WL associated with each; based on the estimation made by [14], each DCC row imposes roughly one transistor over regular DRAM cell to each BL. Besides, DRISA-3T1C [3] requires 2 add-on transistors per cell, which essentially triple the area overhead.

V. Application Study

A. Graph Analysis

Real world graph consists of millions of vertices and edges that need to be processed. To efficiently map such graphs into *ReDRAM* architecture, graph partitioning methods are used. Here, we adopt interval-block partitioning method to balance workloads of each *ReDRAM*'s chip and maximize parallelism. We use hash-based method [8] to split the vertices into M intervals and then divide edges into M^2

[1]This energy doesn't involve the energy that processor consumes to perform the operation.

Authorized licensed use limited to: University of Central Florida. Downloaded on June 23,2020 at 18:35:10 UTC from IEEE Xplore. Restrictions apply.

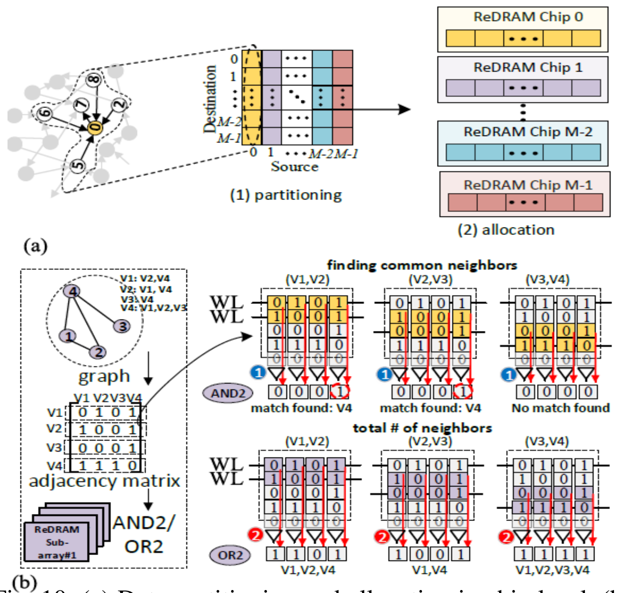


Fig. 10: (a) Data partitioning and allocation in chip level, (b) *ReDRAM*'s mapping and acceleration for finding matching index in sub-array level.

blocks as shown in Fig. 10a. For example,the matching index $M_{i,j}$ quantifies the *similarity* between two vertices (V_i and V_j) based on the number of common neighbors shared by vertices as ($\frac{\sum \text{common neighbors}}{\sum \text{total number of neighbors}}$). The main task here is to find the common and total number of neighbors which can be implemented and accelerated by *ReDRAM*. Fig. 10b provides a straightforward example to elucidate the mapping and acceleration method of *ReDRAM*. After partitioning and allocation, the sample four-vertex network is converted to adjacency matrix and stored in 4 consecutive rows of sub-array. To find the common neighbors of two particular vertices (e.g. V1, V2), *ReDRAM* performs parallel `AND2` on the rows and SA's outputs determine the matches (here, V4). In addition, the total number of neighbors is found by performing `OR2` operation on the same rows. Then, *ReDRAM* can readily process the summation operation based on the ISA.

•**Experiments:** We configure the *ReDRAM*'s memory sub-array with 1024 rows and 256 columns, 4×4 mats (with 1/1 as row/column activation) per bank organized in H-tree routing manner, 16×16 banks (with 1/1 as row/column activation) and 1024Mb total capacity. Therefore, an identical physical memory size (1024Mb) is considered for all PIM implementations henceforth. To estimate the performance of the accelerators, we take three social network data-sets, as tabulated in Table III.

TABLE III: Social Network data-sets.

Dataset	Nodes	Edges	Graph Information
ego-Facebook	4,039	88,234	profiles & friends lists from Facebook
dblp-2010	326,186	1,615,400	scientific collaboration network
amazon-2008	735,323	5,158,388	similarity among books reported by Amazon store

Fig. 11a depicts the energy that four accelerators (Ambit [2], ReDRAM, DRISA-3T1C [3], and GPU) consume to perform matching-index task on different data-sets. We observe that *ReDRAM* obtains the highest energy-efficiency compared to others owning to DRA mechanism. *ReDRAM* consumes on average 2.5× less energy than that of Ambit accelerator. Compared to GPU, it reduces the energy consumption by ~21×. Fig. 11b plots the execution time of the *ReDRAM* and other accelerators. We observe that *ReDRAM* solution is on average 5× faster than that of Ambit solution and 49× faster than GPU. This is mainly because of fast and parallel in-memory operations of *ReDRAM*, specifically for implementing `AND2-OR2` operations. Fig. 11c also reports the Memory Bottleneck Ratio (MBR), which is the time fraction at which the computation has to wait for data and on-/off-chip data transfer obstructs its performance (memory wall happens) running matching index task on three data-sets. The experiment is performed according to the peak throughput for each platform considering number of memory access. The results reemphasize the PIM platform's efficiency for solving memory wall issue. We observe that *ReDRAM* along with other PIM solutions spend less than ~22% time for memory access and data transfer. However, GPU accelerator spends more than 90% time waiting for the loading data.

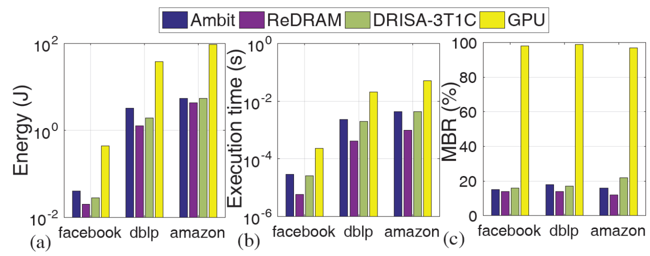


Fig. 11: (a) Normalized energy consumption, (b) Execution time, (c) Memory bottleneck ratio of the accelerators.

B. Data Encryption

We take the Advanced Encryption Standard (AES) algorithm as an example to elucidate the mapping of transformations in *ReDRAM*, which reveals its benefits of energy-efficiency and high-throughput for in-memory data encryption applications. AES is an iterative symmetric-key cipher where both sender and receiver units use a single key for encryption and decryption. AES basically works on the standard input length of 16 bytes (128 bits) data organized in a 4×4 matrix (called state matrix (S_M)) while using 3 different key lengths (128, 192, and 256 bits). For 128-bit key length, AES encrypts the input data after 10 rounds of consecutive transformations enumerated as SubBytes, ShiftRows, MixColumns, and AddRoundKey in Fig. 12.

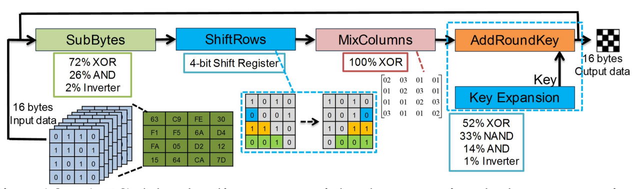


Fig. 12: AES block diagram with the required data organization and gate utilization of each transformation.

To facilitate working with input data, each input byte data is distributed into 8-bit such that eight memory sub-arrays are filled by 4×4 bitmatrices. After mapping, *ReDRAM* can support the required AES bulk bit-wise operations to accelerate each transformations inside the memory. As shown in

Authorized licensed use limited to: University of Central Florida. Downloaded on June 23,2020 at 18:35:10 UTC from IEEE Xplore. Restrictions apply.

Fig. 12, all transformations are mainly based on (N)AND and XOR gates. In SubBytes, MixColumns, and AddRoundKey stages, parallel in-memory `XOR2` and `(N)AND2` operations contribute to more than 90% of the operations. In ShiftRows stage, S_M will undergo a cyclical shift operation by a certain offset. One of the DRAM arrays is considered as a buffer to temporarily save the readout data. In this way, after reading the data from second to fourth row (3 rows), they can be easily rewritten to the memory with desired order using *ReDRAM*'s copy operation.

•**Experiments:** We assess the performance of 128-bit AES implemented by general purpose processor (GPP), ASIC, CMOL [23], Ambit [2], DRISA-3T1C [3], and *ReDRAM*, in terms of energy consumption and number of cycles required for the process. For evaluation of AES performance in GPP, AES C code is compiled, then cycle-accurate gem5 [24] is used to take AES binary and accordingly system level processor power evaluating tool McPAT [25] is used to estimate power dissipation. For evaluation of AES in CMOS ASIC (1.133GHz), Synopsys Design Compiler tool is used. Fig. 13a and Fig. 13b show the breakdown of energy (Y-axis in Log scale) and number of cycles required for different AES transformations after mapping to the different platforms, respectively.

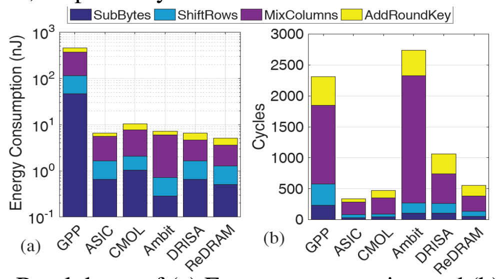


Fig. 13: Breakdown of (a) Energy consumption and (b) Delay of different AES implementations.

The results show the *ReDRAM*'s energy-efficiency (Fig. 13a) compared to other platforms. It reduces the energy consumption by $\sim 23\%$ compared to the CMOS-ASIC. From number of cycles stand point, we observe that MixColumns consumes the most clock cycles as well as energy due to the high number of resources (memory and in-memory `XOR2`) that it takes during operation. In some of the XOR-unfriendly platforms such as Ambit [2], MixColumns contributes to more than 70% of the energy consumption and number of cycles. Overall, *ReDRAM* requires the least number of cycles compared with other processing-in-DRAM platforms and GPP. However, ASIC (with 336 cycles) and CMOL (470) designs show better performance compared to *ReDRAM* (552).

VI. Conclusion

In this work, we presented *ReDRAM*, as a high-throughput and reconfigurable PIM architecture to implement a full set of 1- and 2-input bulk bit-wise operations (NOT, (N)AND, (N)OR, and even X(N)OR). *ReDRAM* exploits a new dual-row activation mechanism and eliminates the need for previous mechanisms based on triple-row activation, dual-contact cells, etc. The simulation results show that as a graph processing accelerator, *ReDRAM* reduces the energy consumption and execution time $\sim 21\times$ and $49\times$, respectively, compared with GPU. As for AES data-encryption, it reduces the energy consumption by 23% compared to CMOS-ASIC implementation.

Acknowledgements

This work is supported in part by the National Science Foundation under Grants No. 1740126 and No. 1908495 and Semiconductor Research Corporation nCORE.

References

[1] P. Chi *et al.*, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *ISCA*. IEEE Press, 2016.

[2] V. Seshadri *et al.*, "Ambit: In-memory accelerator for bulk bitwise operations using commodity dram technology," in *Micro*. ACM, 2017, pp. 273–287.

[3] S. Li *et al.*, "Drisa: A dram-based reconfigurable in-situ accelerator," in *Micro*. ACM, 2017, pp. 288–301.

[4] S. Aga *et al.*, "Compute caches," in *High Performance Computer Architecture (HPCA), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 481–492.

[5] "Hybrid memory cube speci!cation 2.0." [Online]. Available: http://www.hybridmemorycube.org/files/SiteDownloads/HMC-30G-VSR˙HMCC˙Specification˙Rev2.0˙Public.pdf.

[6] S. Angizi *et al.*, "Rimpa: A new reconfigurable dual-mode in-memory processing architecture with spin hall effect-driven domain wall motion device," in *ISVLSI*. IEEE, 2017, pp. 45–50.

[7] S. Angizi, J. Sun, W. Zhang, and D. Fan, "Aligns: A processing-in-memory accelerator for dna short read alignment leveraging sot-mram," in *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, 2019, p. 144.

[8] G. Dai *et al.*, "Graphh: A processing-in-memory architecture for large-scale graph processing," *IEEE TCAD*, 2018.

[9] S. Angizi *et al.*, "Graphs: A graph processing accelerator leveraging sot-mram," in *DATE*. IEEE, 2019, pp. 378–383.

[10] "6th generation intel core processor family datasheet." [Online]. Available: https://www.intel.com/content/www/us/en/products/processors/core/core-vpro/i7-6700.html

[11] "Geforce gtx 1080 ti." [Online]. Available: https://www.nvidia.com/en-us/geforce/products/10series/geforce-gtx-1080-ti/

[12] V. Seshadri *et al.*, "Fast bulk bitwise and and or in dram," *IEEE Computer Architecture Letters*, vol. 14, 2015.

[13] V. Seshadri, Y. Kim *et al.*, "Rowclone: fast and energy-efficient in-dram bulk data copy and initialization," in *46th Micro*. ACM, 2013, pp. 185–197.

[14] H. B. Kang and S. K. Hong, "One-transistor type dram," Apr. 20 2010, uS Patent 7,701,751.

[15] S. Angizi and D. Fan, "Graphide: A graph processing accelerator leveraging in-dram-computing," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*. ACM, 2019, pp. 45–50.

[16] G. Sideris, "Intel 1103-mos memory that defied cores," *Electronics*, vol. 46, no. 9, pp. 108–113, 1973.

[17] Q. Deng *et al.*, "Dracc: a dram based accelerator for accurate cnn inference," in *55th DAC*. ACM, 2018, p. 168.

[18] M. W. Allam *et al.*, "High-speed dynamic logic styles for scaled-down cmos and mtcmos technologies," in *ISLPD*. ACM, 2000, pp. 155–160.

[19] K. Navi *et al.*, "A novel low-power full-adder cell with new technique in designing logical gates based on static cmos inverter," *Microelectronics Journal*, vol. 40, pp. 1441–1448, 2009.

[20] (2011) Ncsu eda freepdk45. [Online]. Available: http://www.eda.ncsu.edu/wiki/FreePDK45:Contents

[21] . DRAM Power Model. https://www.rambus.com/energy/.

[22] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi, "Cacti 5.1," Technical Report HPL-2008-20, HP Labs, Tech. Rep., 2008.

[23] Z. Abid *et al.*, "Efficient cmol gate designs for cryptography applications," *IEEE TNANO*, vol. 8, no. 3, pp. 315–321, 2009.

[24] N. Binkert *et al.*, "The gem5 simulator," *SIGARCH*, vol. 39, pp. 1–7, 2011.

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

Authorized licensed use limited to: University of Central Florida. Downloaded on June 23,2020 at 18:35:10 UTC from IEEE Xplore. Restrictions apply.