A Fast, Reliable and Wide-Voltage-Range In-Memory Computing Architecture

William Simon, Juan Galicia, Alexandre Levisse, Marina Zapater and David Atienza
Embedded System Laboratory (ESL), Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland.
william.simon@epfl.ch

ABSTRACT

As the computational complexity of applications on the consumer market, such as high-definition video encoding and deep neural networks, become ever more demanding, novel ways to efficiently compute data intensive workloads are being explored. In this context, In-Memory Computing (IMC) solutions, and particularly bitline computing in SRAM, appear promising as they mitigate one of the most energy consuming aspects in computation: data movement. While IMC architectural level characteristics have been defined by the research community, only a few works so far have explored the implementation of such memories at a low level. Furthermore, these proposed solutions are either slow (<1GHz), area hungry (10T SRAM), or suffer from read disturb and corruption issues. Overall, there is no extensive design study considering realistic assumptions at the circuit level. In this work we propose a fast (up to 2.2Ghz), 6T SRAM-based, reliable (no read disturb issues), and wide voltage range (from 0.6 to 1V) IMC architecture using local bitlines. Beyond standard *read* and *write*, the proposed architecture can perform *copy*, *addition* and *shift* operations at the array level. As *addition* is the slowest operation, we propose a modified carry chain adder, providing a 2× carry propagation improvement. The proposed architecture is validated using a 28nm bulk high performances technology PDK with CMOS variability and post-layout simulations. High density SRAM bitcells (0.127μm) enable area efficiency of 59.7% for a 256×128 array, on par with current industrial standards.

KEYWORDS

In-Memory Processing, IMP, In-Memory Computing, IMC, bitline computing, fast carry ripple, adder, local bitline, read disturb

ACM Reference format:
William Simon, Juan Galicia, Alexandre Levisse, Marina Zapater and David Atienza. 2019. A Fast, Reliable and Wide-Voltage-Range In-Memory Computing Architecture. In *Proceedings of ACM Design Automation Conference, Las Vegas, Nevada USA, June 2019 (DAC'19),* 6 pages.
DOI: 10.1145/3316781.3317741

1 INTRODUCTION AND RELATED WORK

Data intensive workloads such as artificial intelligence or real-time video streaming and rendering applications require more and more computational capabilities both in so-called "edge devices" and in servers. In this context, increasing efficiency, understood as reducing energy consumption while maintaining or enhancing performance, is a major concern in both scenarios. Recently, In Memory Computing (IMC) solutions have been proposed as a method for substantially increasing computational capabilities while simultaneously reducing energy consumption [1, 5, 11]. A subset of these innovations utilize in-SRAM computing (in-cache or scratchpad memory), namely *bitline computing*, which has been proposed as a method for accelerating neural networks and finite state automata [6, 13]. While [1] and [2] clearly define the need for SRAM-based IMC solutions, 6T SRAM-based IMC solutions [6, 7] struggle with read disturb data corruption, limiting profitability for ultra-low power or high performance applications in terms of degraded read stability and reduced speed. To address this issue, [2] proposes using 10T SRAM bitcells which do not suffer from read disturb issues, at the expense of reducing the bitcell array density (at least 2× lower than 6T bitcells), confining it to low density applications. The same work also proposes the use of a Carry Ripple Adder (CRA) to perform in-memory *additions*. However, they do not propose any implementation, layout, or performance considerations from a circuit perspective. Finally, in [8], a capacitive adder is proposed; however, such analog-based approaches suffer from high variability, limited supply range, and slow operation.

This work proposes a dense, reliable, and fast SRAM array that performs IMC operations at 2.3 to 2.8 × the frequency of current state-of-the-art solutions [6, 7]. This work also enables fast in-memory addition: 2.2Ghz for 8-16bits, down to 1.2Ghz for 64bits. To accomplish this, we implement a fast carry adder pitched underneath the memory array which outperforms a standard carry ripple adder [2] by 60 to 70% depending on its depth. The proposed architecture is designed and laid out in 28nm high performance CMOS PDK technology from TSMC. Its functionality is validated through electrical simulations accounting for variability and layout parasitic effects.

Overall, the main contributions of this work are:

- We propose an innovative in-SRAM IMC architecture enabling in-memory bitwise, *addition*, *shift* and *copy* operations. We implement it in a 28nm bulk high performance CMOS technology PDK and demonstrate its operation through CMOS variability and layout aware simulations.
- We propose the use of local read bitlines to (i) avoid data corruption issues during in-SRAM processing operations and (ii) enable high frequency operations (2.2Ghz at 1V).
- We propose a method to mask the carry propagation delay of in-memory *addition* and implement a fast carry adder to improve its performance (60-70% improvement).
- We explore the design space of the proposed IMC memory architecture and provide energy, area and speed values for various configurations.

DAC'19, Las Vegas, Nevada USA
 978-1-4503-6725-7/19/06…$15.00
DOI: 10.1145/3316781.3317741

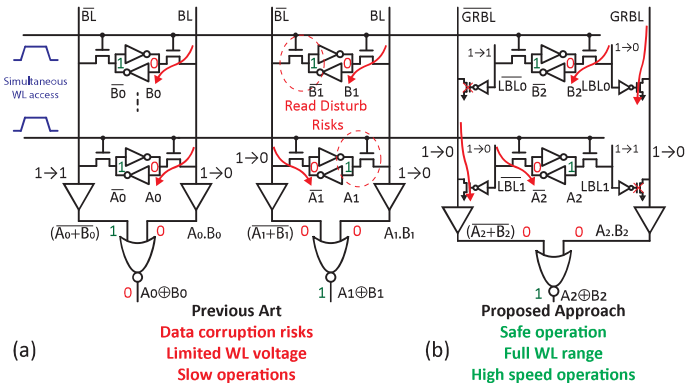

Figure 1: Diagram schematic showing the concept of bitline computing used in this work. (a) Standard bitline computing with highlighted read disturb risks and (b) proposed local bitlines-based bitline computing.

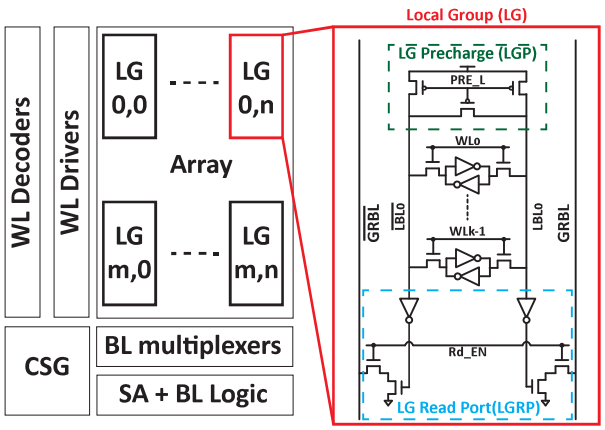

Figure 2: Proposed memory organization with detailed Local Group schematic diagram (red). The Local Group Precharge (LGP) and the Local Read Port (LRP) are identified in green and blue respectively.

The remainder of this paper is organized as follows. Section 2 presents bitline computing and introduces the proposed LG-enabled IMC operations scheme. Section 3 presents the proposed memory architecture while Section 4 details the BL logic considered in this work. Section 5 explains the work's functional validation methodology. Section 6 presents the performance trade-off of the proposed architecture. Finally, Section 7 concludes the paper.

2 RELIABLE BITLINE COMPUTING

Beyond standard read and write operations, in-memory computing operations can be performed on one (shift, copy) or two operands (NOR, AND, XOR, ADD). This section presents how such operations are performed in the proposed memory architecture.

Standard Operations Read and write operations are performed in a standard manner by accessing a single wordline (WL). As the proposed memory uses local bitlines, before each read operation, both local and global Read BLs (LBLs, GRBLs) are precharged to Vdd. Then, the WL is activated. One of the LBLs discharges, activating one of the Local Read Ports (LRP), in turn discharging of one of the GRBLs, as demonstrated in [12, 14]. Two single ended Sense Amplifiers (SA) connected to the GRBLs output the data. Overall, the proposed memory enables simultaneous 1 read + 1 write operation (2-ports), or 2 read + 1 write (3-ports) by accessing the GRBL independently if all the accessed words belongs to different LBLs.

Slow IMC operations In SRAM memories, IMC operations can be performed by simultaneously accessing two WLs after a precharge phase to Vdd. Depending on the states of the accessed bitcells, one or both of the BLs are discharged as shown Figure 1-a. In the end, the BL (respectively $\overline{BL}$) carries the result of an AND operation (respectively NOR). These operations form the basis of bitline computing. When applied to standard 6T bitcells in opposite states, if the PMOS transistor of one bitcell is weaker than the access and pulldown transistors of the other bitcell, the cell may flip, leading to data corruption. This effect can be shown with variability simulations, particularly in the slowP-FastN CMOS corner. In [7] and [1], data corruption is avoided by strongly reducing the WL voltage (i.e., the operation frequency, lower than 1Ghz).

Proposed fast and reliable IMC scheme We propose an innovative approach to performing bitline computing-based IMC operations with 6T SRAM bitcells (contrary to [2]) while avoiding read disturb risks. In this work, the two accessed bitcells are always connected to different LBLs, eliminating the risk of bitcell shorting, as shown Figure 1-b. Within an LBL, an IMC operation is identical to a standard *read*. Then, either one or both of the GRBLs are discharged through the read ports depending on the states of the accessed bitcells. As previously described, the GRBL (resp. $\overline{GRBL}$) carries the result of an AND (resp. NOR) between the two operands. This approach enables full WL dynamic voltage range without read disturb issues while avoiding the use of 10T bitcells.

3 PROPOSED ARCHITECTURE

3.1 Memory Array organization

Figure 2 shows the organization of the proposed memory array. Besides the memory array, it contains: *(i)* WL decoders and drivers, *(ii)* BL multiplexers to enable data interleaving, *(iii)* Sense Amplifiers (SA) and *(iv)* a BL logic block. Finally, the control signals (such as the local group precharge PRE_L and enable Rd_EN signals) are generated in the Control Signal Generation (CSG) block. Inside the memory array, the bitcells are organized into Local Groups (LGs). Each LG contains k bitcells along an LBL, an LG Read Port (LGRP), a Local Group Precharge (LGP), and is connected to a GRBL through its LGRP. The GBRLs are connected to two single ended SAs, which are in turn connected to the BL logic block. The BL logic block (described in Section 4) performs bitwise NOR, AND, and XOR operations as well as more complex operations, such as *copy*, *shift*, and *addition*.

3.2 Decoding Logic and Precharge Management

In this work we consider a straightforward WordLine (WL) decoding scheme consisting of two WL Decoders (WDs) connected to a single WL driver. Two encoded wordline addresses to the operation

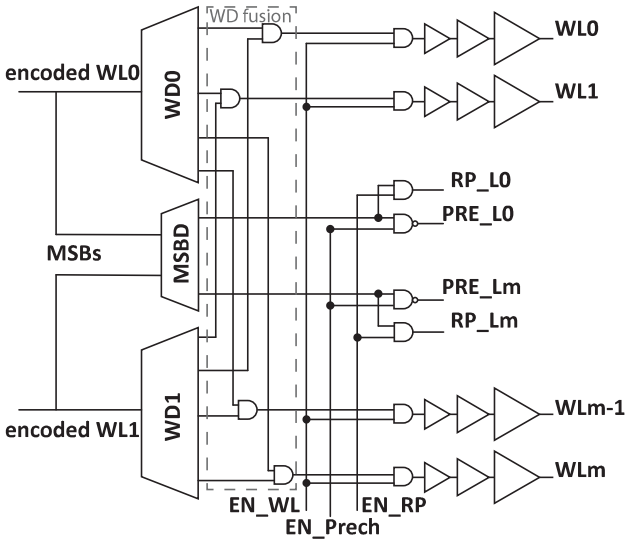

Figure 3: Block schematic of the modified WL decoder used in this work.

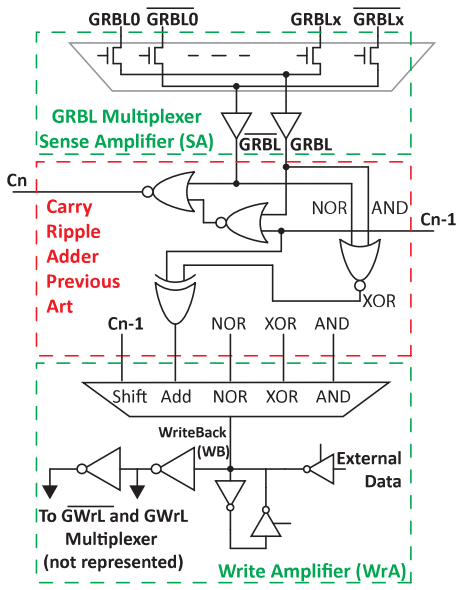

Figure 4: Block schematic of the BL logic considered in this work

operands are decoded by WD0 and WD1. A AND operation between the decoded addresses is performed before driving the WLs (highlighted as "WD fusion" in Figure 3).

In order to minimize local precharge energy consumption, only the accessed LG LBLs are precharged. This is accomplished by decoding the address MSBs separately to generate *(i)* the PRE_L signals controlling the LGP circuits and *(ii)* the Rd_EN activating the LRP circuit. Figure 3 presents the architecture of the decoding structure considered in this work. More optimized WD implementations can be envisioned, however this strategy is sufficient for the purpose of this paper. Further optimizations are left for future works.

4 BITLINE LOGIC

4.1 Architecture

Figure 4 presents a schematic of the proposed IMC logic. The IMC logic contains several subblocks connected in series: *(i)* a GRBL multiplexer and dual single ended SAs consisting of inverter-based buffers that amplify the GRBL signals, *(ii)* a BL Logic block that performs bitwise and complex operations, and *(iii)* an operation multiplexer, WriteBack latch, and WriteBack multiplexer (not represented, identical to GRBL multiplexer) for latching selected operation results, generating write patterns, and writing back to memory via GWrLs. The operation multiplexer selects the data to be written back or sent to the CPU via the memory H-tree. Copy and shift operations access only one WL and write back the AND and carry ripple signals respectively.

In [2], *addition* is performed in 3 cycles; first, the data is read from the memory. Then, ripple carry propagation occurs over the next cycle. Finally, the addition logic is completed in the last cycle. These three operation are performed in a pipeline manner, resulting in a 1-cycle operation once the pipeline is filled. However, such a solution adds a non-negligible area overhead (i.e. 3 latches per BL logic block) and limits the memory frequency to the slowest of the three pipeline stages' frequencies (i.e. the CRA, which can exceed 1*ns* for more than 32 bit additions).

In this work, we propose directly connecting the adder to the memory SA outputs. Consequently, addition starts immediately as data is read from the memory, masking a portion of carry propagation time within the memory read cycle (as described in Section 5). In this context, the adder carry logic must be as fast as possible in order to "hide" its latency in the in-compressible periods of the memory access.

4.2 Proposed Fast Carry Ripple Stage

To maximize carry propagation masking, we propose a fast carry adder based on a dynamic Manchester Carry Chain (MCC) adder [10] implemented in buffered 4-bits configuration, illustrated in Figure 5-a. It is directly connected to the SA outputs (NOR<0:3>and AND<0:3>signals). *Generate<0:3>* and *Propagate<0:3>* signals are simply generated with a single *nor* gate thanks to bitline computing, greatly reducing the area overhead typically associated with such an architecture. Figure 5-b presents the corresponding layout in 28nm bulk CMOS technology. 4 MCC blocks are needed per 16 bitcell columns, as columns are mux-4 multiplexed. Remaining space is used to fit inter-MCC signal buffers as well as decoupling capacitors (10fF MOS capacitor). It should be noted that the same precharge signal (EN_prech) is used in both the proposed MCC and in the memory array for LBL and GRBL precharging.

Figure 6-a shows the gain provided by the proposed MCC versus standard CRA adder for various MCC configurations and adder depths. Both are pitched under the array with post-layout parasitic and process variations included in the simulation. With near 80% performance improvement versus standard CRA, we choose a 4-bit buffered configuration as the most efficient topology. Figure 6-b presents the performance gain of the considered 4-bit buffered topology versus the adder depth and supply voltage. While a diminishing supply voltage reduces performance (as the serial transistors

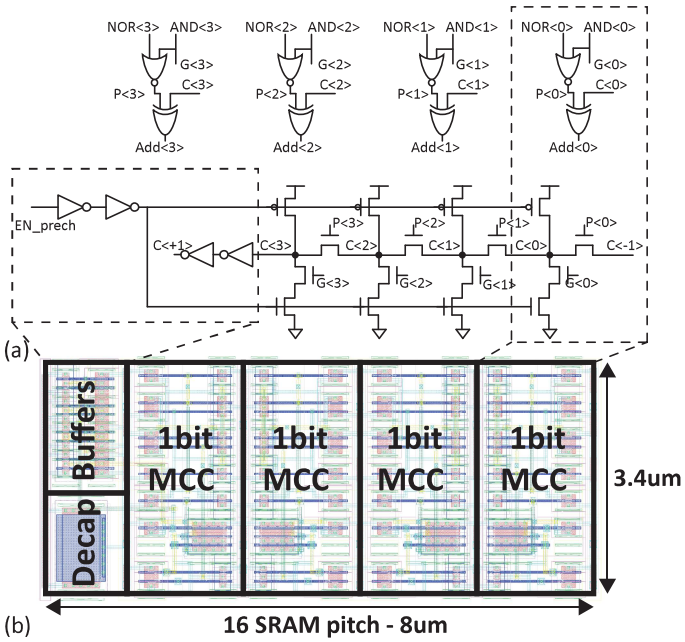

Figure 5: Schematic diagram and physical layout of the proposed fast carry ripple and full adder in buffered 4 stages configuration.

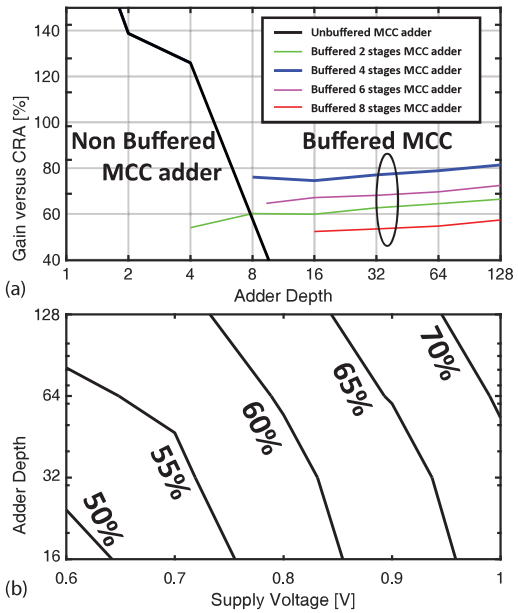

Figure 6: (a) Gain provided by the proposed MCC adder versus CRA for various buffer configuration and adder depth. (b) Gain provided by a buffered 4 stages MCC adder versus adder depth and supply voltages.

become more resistive) a 54% improvement is still observed for 64 bit additions at 0.6V.

5 FUNCTIONAL VALIDATION

We validate the functionality of the proposed memory by implementing a 256WL×64BL (32WL per LG) memory array with its periphery and IMC logic, and simulate its critical path at 300K with 10,000 Monte-Carlo runs, accounting for CMOS variability and equivalent layout parasitics. To optimize the simulation time for validation, only memory critical paths are simulated (WL decoders and drivers, equivalent WLs, equivalent GRBLs, two local groups, and BL logic) in a netlist containing more than 8,000 elements. We model the propagation time of the signals in the memory and periphery by creating equivalent circuits for the lines with corresponding gates and extracted RC networks.

We implement the memory using thin oxide transistors from a TSMC 28nm bulk CMOS high performance technology PDK. While the memory array is implemented in the Regular Voltage Threshold (RVT) technology flavor to limit static leakage, we utilize the Low Voltage Threshold (LVT) technology flavor for the peripheral circuitry in order to optimize performance.

The memory array is designed considering 28nm high density bitcells with modified SRAM rules (we design a 0.127μm SRAM bitcells inspired from [4] and [9] and pitched the periphery on the bottom and sides of the memory array, i.e., on 500nm and 260nm). Additionally, to account for the required spacing between the SRAM and logic design rules, we consider a 500nm spacing between the memory array and the periphery.

Figure 7 presents a chronogram of successive operations performed in the worst-case/slowest BL of the memory array (i.e. the furthest bitcell from the WL driver). The two bitcells considered are the first bitcells of the top LGs (furthest from the LGRP and BL logic). In this chronogram, write operations are performed sequentially to initialize the two bitcells to '1', a NOR IMC operation is performed, and the result is written back to the first bitcell (switching its state). During the NOR operation, both $\overline{LBL0}$ and $\overline{LBL1}$ are discharging, in turn discharging the $\overline{GBL}$ which triggers the NOR output. At the end of the cycle, the NOR data is ready to be latched before the next operation cycle.

Figure 8 shows the operation of the proposed MCC adder tied to the memory BL logic output signals with variability simulations (the MCC output shown is the slowest carry ripple among 10,000 Monte-Carlo runs). Carry ripple propagation begins simultaneously with the first BL logic block; however, carry propagation progresses slower than the rate at which data becomes ready from the memory array (simulation shows an average of 20ps propagation time between the 1st and 256th BL). In addition, the WL discharge time is from 60 to 90ps for the considered sizing (as proposed in [3], the WL in metal 1 is doubled with metal 3, in order to optimize the charging and discharging time). Taking into consideration mandatory design margins, we demonstrate that a 16bit addition (100 to 150ps with variability and post layout parasitics) can be performed during the IMC operation without any decrease in memory frequency. This trade-off is explored Section 6.2.

6 PERFORMANCE RESULTS

The proposed memory architecture is benchmarked on three axes: *(i)* Area, *(ii)* Speed and *(iii)* Energy per operation. Table 1 presents

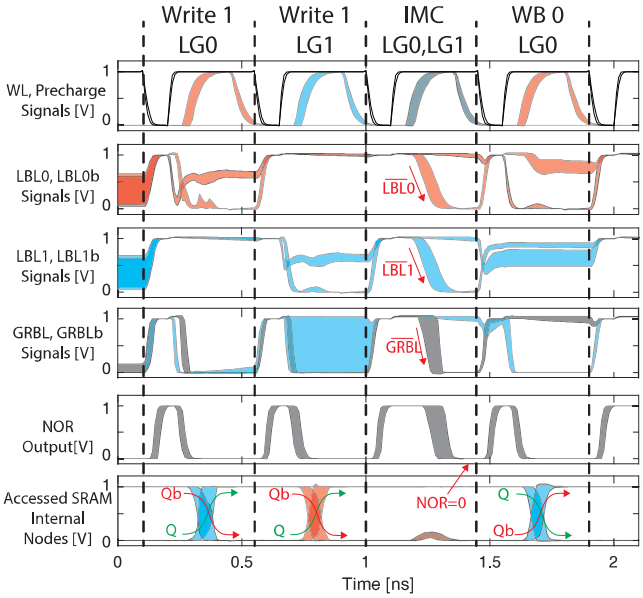

Figure 7: Transient simulation of a 256x64 memory array (i.e. with two LG) with presented standard write, standard read, IMC operations and write back of in memory computed results. Envelopes of the variability simulations are shown in red for LG0 and blue for LG1.

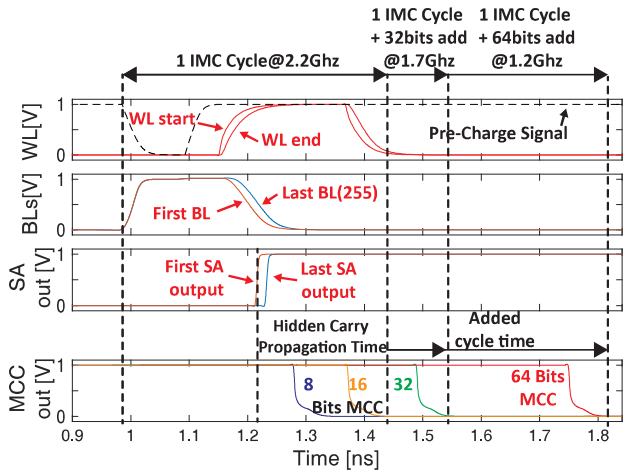

Figure 8: Propagation timing through the memory array for the first and the last of a 256BL memory array. Added timing required to perform 32 and 64bits additions is highlighted.

the specification of the previously published IMC circuit architectures compared to the proposed solution.

6.1 Area Estimation

Figure 9 shows the layout view of a 256BLx128WL array spread over 4 LGs of 32WLs. Each LG is 11.7μm × 128.5μm with its WL drivers. When considered alone, the a 32WL LG results in 71% area efficiency while 16, 64 and 128 WL LGs feature 55.6, 83.4 and 91% area efficiencies respectively, to which must be added WL drivers

Table 1: Summary of the proposed architecture features compared to previously reported in-SRAM IMC memories

	Jeloka [7]	Akyel [2]	This Work
Bitcell Density	6T	10T	6T
Memory Access	1Rd/1Wr	1Wr+2Rd	1Wr+1Rd (1 LG) 1Wr+2Rd(2 LG)
Sensing	Single Ended		
Array Size	64×64	Not Simulated	256×64
Max Freq.	800Mhz	Max Addition Freq.	2.2Ghz
Add. type	no addition	Pipeline	Direct
Add. Freq.	-	Not Simulated	2.2Ghz@8,16bits 1.7Ghz@32bits 1.2Ghz@64bits

Table 2: Worst Case Energy and Frequency Values of IMC Operations in a 256×64 array.

Operation	Rd	Wr	IMC	Add			
Type	Bitwise			8b	16b	32b	64b
E/op[fJ]	23.5	25.9	23.8	20.7	41.6	83.3	167.3
Carry Prop. Time [ps]	-	-	-	64	130	258	512
Array Leak./op [fJ]	88.9					136.9	162.9
Freq.[Ghz]	2.2	2.2	2.2	2.2	2.2	1.7	1.2

and BL logic (we do not consider the WL decoder as it will be shared with the neighbor array). While a larger LG enables better area efficiency, the LG size is limited by a decreasing read margin, as demonstrated in [12]. Also, a co-optimization between circuit and application design is necessary to find optimal LG size. Overall, the 256BL by 128WL (32WL per LG) array shown in Figure 9 provides a 59.7% area efficiency, of which the BL logic accounts for only 4%, while the simulated 256BL by 64WL (32WL per LG) array provides a 53.5% area efficiency.

Periphery Pitch Under the Array. On the bottom of the memory, memory I/O (comprised of SA, BL logic, and WrA) is 6.3μm tall, spread across 16 SRAM pitch (i.e. 8μm as shown Figure 5). While laying out the BL logic, we identified that a mux-4 ratio between the periphery and the array is the most efficient approach regarding the adder as it provides enough space (2μm) to optimize its layout.

6.2 Speed Evaluation

6.2.1 Bitwise operations. Figure 10 shows the maximum clock frequency of different proposed IMC architectures compared to this work at various supply voltages. The red solid curve shows the maximum frequency achievable by this work's architecture performing bitwise IMC operations while accessing simultaneously two LGs. On the other hand, the triangle markers refer to literature solutions not using LGs [1, 7]. Overall, LG-enabled IMC operation provides a 2.8 × improvement versus standard architectures, while also enabling operation at 416MHz and 0.6V, where standard architectures no longer function. As a reference, we simulate this work's

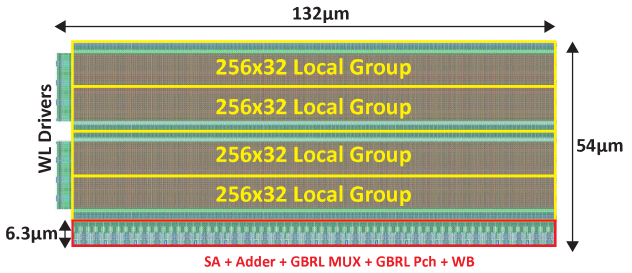

Figure 9: Layout of a 256BLx128WL memory array using the proposed in memory processing architecture. This configuration, enables a 59.7% area efficiency.

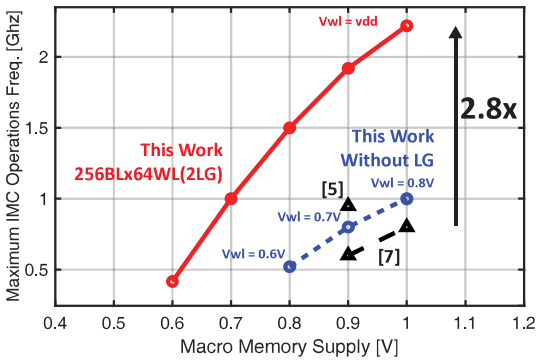

Figure 10: Maximum frequency of bitwise operations versus memory macro supply voltage

architecture without using LGs (blue dotted curve), obtaining similar performance to previously published IMC solutions (1Ghz at 1V with a WL voltage reduced to 0.8V). Higher performances can be achieved; for example, while keeping the same sizing for the peripheral driver circuits, we show that shorter WLs enable faster operations (2.5Ghz at 1V for 64 BLs, not shown in Figure 10).

6.2.2 Addition. Table 2 shows the worst-case ripple carry time and maximum computational frequency of the 4-bit buffered MCC adder with consideration for CMOS variability and layout effects. While 8/16 bit addition can be completed during the read pulse, 32/64bit addition require reduced operating frequencies: 1.7Ghz (1.2Ghz for 64bits). It can be noted here that considering smaller memories while keeping the same driver sizing (i.e., shorter discharge) would reduce the profitability of the proposed addition scheme as the ripple time does not scale with the WL charging-discharging time.

6.3 Energy Assessments

We extract the energy figures for each of the operations performed in the simulated 256BL by 64WL (32 WL per LG) memory and display them in Table 2. We include in these simulations the leakage of the entire memory array as the addition frequency is reduced with increasing bit depth. Bitwise IMC operations are performed at 2.2Ghz while addition frequency is adapted depending on its depth. The adder energy scales almost linearly with its depth (leakage contribution from decreased frequency is <5% when moving from 2.2 to 1.2Ghz), enabling a wide range of addition configurations (e.g. 8 parallel 8bit additions or 1 single 64bit addition) without impacting the memory power budget.

7 CONCLUSION

While it is believed that IMC is one of the most promising topics for the future of computing architectures, there is a lack of circuit and architecture evaluations of such memories in the community. In this context, we propose a new architecture that relies on local bitline based IMC to perform a wide range of operations (AND, NOR, XOR, shift, copy, add). The proposed memory architecture operates within a wide range of supply voltages (0.6-1V) without any added reliability degradation compared to standard SRAM architectures. Additionally, we identify a way to improve the performance of IMC addition by implementing an MCC enhanced fast adder within the memory BL logic. The added BL logic reduces area efficiency by 4%. Overall the proposed architecture achieves up to 2.2Ghz bitwise IMC and 16bit addition or 32bit (resp. 64) addition at 1.7Ghz (resp. 1.2Ghz) at 1V. Moreover, the proposed architecture can perform reliable bitwise operations down to 0.6V at a frequency of 416MHz.

ACKNOWLEDGMENTS

This work has been supported by the ERC Consolidator Grant COMPUSAPIEN (GA No. 725657).

REFERENCES

[1] S. Aga, S. Jeloka, et al. 2017. Compute Caches. *IEEE International Symposium on High Performance Computer Architecture (HPCA)* (2017).
[2] Kaya Can Akyel, Henri-Pierre Charles, et al. 2016. DRC2: Dynamically Reconfigurable Computing Circuit based on memory architecture. *IEEE International Conference on Rebooting Computing (ICRC)* (2016).
[3] J. Chang, Y. Chen, et al. 2017. A 7nm 256Mb SRAM in high-k metal-gate FinFET technology with write-assist circuitry for low-VMIN applications. *IEEE International Solid-State Circuits Conference (ISSCC)* (2017).
[4] M. Chang, C. Chen, et al. 2015. A 28nm 256kb 6T-SRAM with 280mV improvement in VMINusing a dual-split-control assist scheme. *IEEE International Solid-State Circuits Conference (ISSCC)* (2015).
[5] Quan Deng, Lei Jiang, et al. 2018. DrAcc: A DRAM Based Accelerator for Accurate CNN Inference. *IEEE/ACM/EDAC Design Automation Conference (DAC)* (2018).
[6] Charles Eckert, Xiaowei Wang, et al. 2018. Neural Cache: Bit-Serial In-Cache Acceleration of Deep Neural Networks. *ACM/IEEE International Symposium on COmputer Architecture (ISCA)* (2018).
[7] Supreet Jeloka, Naveen Bharathwaj Akesh, et al. 2016. A 28 nm Configurable Memory (TCAM/BCAM/SRAM) Using Push-Rule 6T Bit Cell Enabling Logic-in-Memory. *IEEE Journal of Solid-State Circuits (JSSC)* (2016).
[8] Mingu Kang, Eric P Kim, et al. 2015. Energy-efficient and high throughput sparse distributed memory architecture. *IEEE International Symposium on Circuits and Systems (ISCAS)* (2015), 2505–2508.
[9] H. Pilo, C. A. Adams, et al. 2013. A 64Mb SRAM in 22nm SOI technology featuring fine-granularity power gating and low-energy power-supply-partition techniques for 37% leakage reduction. *IEEE International Solid-State Circuits Conference (ISSCC)* (2013).
[10] M. Schlag and P. Chan. 1990. Analysis and Design of CMOS Manchester Adders with Variable Carry-Skip. *IEEE Transactions on Computers (TC)* (1990).
[11] Vivek Seshadri, Donghyuk Lee, et al. 2017. Ambit: In-memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology. *IEEE/ACM International Symposium on Microarchitecture (MICRO)* (2017).
[12] Mahmut E. Sinangil, Hugh Mair, et al. 2011. A 28nm high-density 6T SRAM with optimized peripheral-assist circuits for operation down to 0.6V. *IEEE International Solid-State Circuits Conference (ISSCC)* (2011).
[13] Arun Subramaniyan, Jingcheng Wang, et al. 2017. Cache Automaton. *IEEE/ACM International Symposium on Microarchitecture (MICRO)* (2017).
[14] Shang-Lin Wu, Kuang-Yu Li, et al. 2017. A 0.5-V 28-nm 256-kb Mini-Array Based 6T SRAM With Vtrip-Tracking Write-Assist. *IEEE Transactions on Circuits and Systems I (TCASI)* (2017).