International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No. (01), January 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

IMPLEMENTATION OF COMPUTATION-IN-MEMORY PARALLEL ADDER

[1]Ms. K.Jeyabala, [2]Dr. P.Ramesh Kumar,

[1] PG Scholar, ME VLSI design,
Sengunthar Engineering College, Tiruchengode, Tamilnadu.(India)
[2] Professor, Department of ECE,
- Sengunthar Engineering College, Tiruchengode,Tamilnadu(India)

ABSTRACT

Computation-In-Memory (CIM) architecture, based on the integration of storage and computation in the same physical location using non-volatile memristor technology offers a potential solution for the memory bottleneck. A Computation-In-Memory (CIM) based parallel adder and showed its potentials and superiority for intensive computing and massive parallelism by comparing it with state-of-the-art computing systems including multicore, GPU and FPGA architecture. The results show that Computation-In-Memory (CIM) based parallel adder can achieve at least two orders of magnitude improvement in computational efficiency, energy efficiency, and area efficiency. Computation-in-memory (CIM) is a novel architecture that tries to solve/alleviate the impact of these challenges using the same device (i.e., the memristor) to implement the processor and memory in the same physical crossbar. To analyze its feasibility in depth, this paper proposes two memristor implementations of a data-intensive arithmetic application (i.e., parallel addition). To the best of our knowledge, this is the first paper that considers the cost of the entire architecture including both crossbar and its CMOS controller. The results show that CIM architecture in general and the CIM parallel adder, in particular, have a high scalability. CIM parallel adder achieves at least two orders of magnitude improvement in energy and area in comparison with a multicore-based parallel adder. Moreover, due to a full Variety of memristor design methods tradeoffs can be made between the city, delay, and energy consumption.

Key Words: Computation-In-Memory (CIM), Implication Logic, Memristor, The Parallel Adder.

I.INTRODUCTION

Computation-In-Memory (CIM) to design a parallel adder and illustrate the massive potential of such an architecture for a simple case study intensive arithmetic operations (additions). The structure uses a revolutionary approach based on the integration of storage and computation in the same physical location, and non-volatile memristor technology. It is worth noting that combining multiple numbers is a basic yet very representative operation in essential data applications. In architectures (e.g., multicore, GPU, and FPGA), simple actions such as adding multiple numbers already face the memory bottleneck. As the processors have to fetch vast amounts of data from memory, the intrinsic parallelism cannot be exploited fully in such architectures. The CMOS technology is reaching its physical –if not economical- limits. Down-scaling devices

have led to many challenges such as leakage power, reliability, fabrication process and turnaround time, test complexity, cost for mask and design, and yield. Furthermore, the performance gain by increasing clock speed has saturated since early today, speed-up is no longer the result of a faster clock, but rather an effect of parallelization on multi-core and many-core systems.

The number of parallel cores that can be programmed

And the computation efficiency that can be extracted is tending to saturate as well. Apparently, all today's computing systems are mainly built on John von Neumann stored-program computer concept. All of these motivate the need for a new architecture being able to eliminate the communication bottleneck and support massive parallelism to increase the overall performance, reduce the energy inefficiency to improve the computation efficiency. Logic-In-Memory (LIM) was initially introduced as a memory accelerator, i.e., add some processing units To alleviate the memory bottleneck and provide practical and efficient solutions for data-intensive applications, many architectural solutions have been proposed.

The processor-in-memory (PIM) was introduced as an architecture that consists of a host CPU, main memory, and a number of accelerators close to the main memory to prevent intensive communication with the However, the effectiveness of this architecture strongly depends on the technology to fabricate the accelerators and foremost minds, which is called merged-logic Unfortunately, this merged technique still suffers from a high cost and low density. Second, near data architectures, the emerging non-volatile memory technology, either using traditional processor approach or using novel neural computing approach. Note that all efforts above both at architectural and technology level tried to close the gap between processor and memory speed. Stored-program/von Neumann concept, the computation is still carried out in a separate physical unit. Therefore, the memory bottleneck is still affecting the computer performance and energy.Third, as a result of this resistive computing architectures were introduced to alleviate the memory bottleneck using memristor technology. Although the memristor technology is not mature yet, several publications focused on the design of circuits and architectures such as programmable logic-in-memory architecture; resistive GP-SIMD is processing-in memory architecture and computation-in-memory (CIM) computing paradigm. However, these works mostly focus on the memristor part. They ignore the controlling and peripheral circuit overheads. Du Nguyen et al. have analyzed the performance of a parallel memristor adder based on the crossbar (i.e., a snake that takes the sum of multiple inputs and produces a single output) using the CIM computing paradigm.

The adder outperforms state-of-the-art multicore, GPUs, and memristor implementations with two orders of magnitude. All the adder implementations were based on high-level assumptions. The feasibility of implementing two different parallel adders in the memristor crossbar, i.e., one based on implication logic and one on Boolean logic. To the best of our knowledge, this is the first publication that considers the cost of the entire system (i.e., the crossbar and CMOS controller). The delay, area and energy costs of the CMOS controller, peripheral circuits (required to read and write to the crossbar), and the crossbar itself are analyzed. Also, we evaluate the scalability of the parallel adder for both implementations the rest of this paper is structured as follows. The crossbar is specialized to perform computation and storage operations in cells

organized in rows and columns. Each cell can be a computational unit (such as an adder or multiplier) or storage location (such as a memory cell). The cells in a row or column can be configured with the same or different functionality. The communication in CIM architecture has maximum flexibility. The structure allows bi-directional communication in both horizontal and vertical direction. The controller contains a router and a finite state machine (FSM). The router provides the FSM with a communication scheme for data distribution and movements. The FSM fetches instructions from an instruction memory (e.g., hard disk), converts brought instructions to control signals for the row/column voltage controller.

II LITERATURE REVIEW

The low complexity design of weighted modulo Parallel adder, derived by decomposition of parallel-prefix computation into several blocks of smaller input bit-widths. Besides, we have proposed a novel enhanced circular carry generation unit to process the carry-bits produced by all the parallel prefix computation units (of small input bit-widths) to obtain the final modulo sum efficiently regarding area-delay product. We have implemented the CMOS technology, and from the synthesis results, we find that our proposed adder outperforms the previously reported weighted modulo parallel adder. A low-complexity design of weighted modulo parallels adder decomposition of the n-bit parallel prefix computation of the summation stage into several blocks of parallel-prefix calculation with fewer input bit-widths. Moreover, using a novel circular carry generation scheme, we have derived the final modulo sum efficiently by a shorter combinational path. The saving of area-delay product by the proposed method is due to the significant reduction of the number of carrying nodes resulting from the decomposition of parallel-prefix computation, and modification of a combinational path by the proposed circular carry generation scheme. Conventional radix-based number systems mainly for the implementation of arithmetic operations on large integers. The that various arithmetic operations, e.g., the addition, subtraction, and multiplication can be performed without carrying propagation in residue representation.

III. PROPOSED SYSTEM

Computation-In-Memory (CIM) developed to design a parallel adder and illustrate the massive potential of such architecture for a simple case study: intensive arithmetic operations (additions). The structure uses a revolutionary approach based on the integration of storage and computation in the same physical location, and non-volatile memristor technology. It is worth noting that adding multiple numbers is a basic yet very representative operation it has essential data applications. In architectures (e.g., multicore, GPU, and FPGA), simple actions such as adding multiple numbers already face the memory bottleneck. As the processors have to fetch vast amounts of data from memory, the intrinsic parallelism cannot be exploited fully in such architectures. The challenges using the same device (i.e., the memristor) to implement the processor and memory in the same physical crossbar. The memristor implementations of a data-intensive arithmetic application (i.e., parallel addition). That considers the cost of the entire architecture including both crossbar and its CMOS controller. The results show that CIM architecture in general and the CIM parallel adder, in

particular, have a high scalability. CIM parallel adder achieves at least two orders of magnitude improvement in energy and area in comparison with a multicore-based parallel adder. Moreover, due to a wide variety of memristor design methods (such as Boolean logic), tradeoffs can be made between.

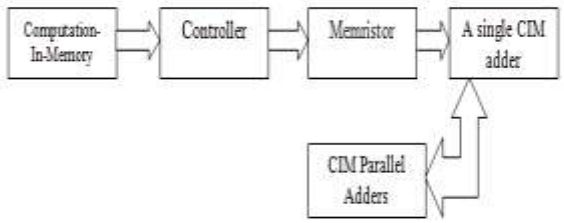

Figure.1 Proposed Circuit diagram

3.1 Computation In Memory

One of the most critical challenges for today's and future data-intensive and significant data problems (ranging from economics and business activities to public administration, from national security to many scientific research areas) is data storage and analysis. The primary goal is to increase the understanding of processes by extracting highly useful values hidden in the vast volumes of data. The increase of the data size has already surpassed the capabilities of today's computation architectures which suffer from the limited bandwidth (due to communication and memory-access bottlenecks), energy inefficiency and limited scalability (due to CMOS technology). The basic computational unit is an n-bit adder, which is surrounded by some memory cells (latches). An n-bit adder contains three n-bit latches two for the inputs and one for the sum, a 1-bit carry-in and a 1-bit carry-out latch. The computation in-parallel memory adder arranges multiple computations in memory adders in a binary tree network. The carry-in and carry-out registers of an adder are correctly connected to generate correct addition results.

3.2 Controller

The crossbar is specialized to perform computation and storage operations in cells organized in rows and columns. Each cell can be a computational unit (such as an adder or multiplier) or storage location (such as a memory cell). The communication in CIM architecture has maximum flexibility. The structure allows bi-directional communication in both horizontal and vertical direction. The controller contains a router and a finite state machine (FSM). The router provides the FSM with a communication scheme for data distribution and movements. The FSM fetches instructions from an instruction memory (e.g., hard disk), converts brought instructions to control signals for the row/column voltage controller.

3.3 Memristor

A fourth device existed to provide conceptual symmetry with the resistor, inductor, and capacitor. This equality follows from the description of essential passive circuit elements as defined by a relation between two of the

four fundamental circuit variables. A device was linking charge and flux (themselves defined as time integrals of current and voltage), which would be the memristor, was still hypothetical at the time.The discovery of a switching memristor. Based on a thin film of titanium dioxide, it has been presented as an approximately ideal device. The reason that the memristor is radically different from the other fundamental circuit elements is that, unlike them, it carries a memory of its past. When you turn off the voltage to the circuit, the memristor still remembers how much was applied before and for how long. That's an effect that can't be duplicated by any circuit combination of resistors, capacitors, and inductors, which is why the memristor qualifies as a fundamental circuit element.

3.4 Need for Memristor

A memristor is one of four essential electrical circuit components, joining the resistor, capacitor, and inductor. The memristor, short for "memory resistor" was first theorized by student Leon Chua in the early 1970s. He developed mathematical equations to represent the memristor, which Chua believed would balance the functions of the other three types of circuit elements.The known three fundamental circuit elements as the resistor, capacitor, and inductor relate four underlying circuit variables as electric current, voltage, charge and magnetic flux. In that, we were missing one to compare the cost of magnetic flux. That is where the need for the fourth fundamental element comes in. This feature has been named as the memristor.

The Memristance (Memory + Resistance) is a property of an Electrical Component that describes the variation in Resistance of a component with the flow of charge. Any two-terminal electrical element that exhibits Memristance is known as a Memristor. Memristance is becoming more relevant and necessary as we approach smaller circuits, and at some point when we scale into nanoelectronics, we would have to take memristance into account in our circuit models to simulate and design electronic circuits properly. An ideal memristor is a passive two-terminal electronic device that is built to express only the property of memristance (just as a resistor expresses resistance and an inductor expresses inductance). The water itself is analogous to electrical charge, the pressure at the input of the pipe is similar to voltage, and the rate of flow of the water through the tube is like an electrical current.

3.5 Single Computation In Memory Parallel Adder

The parallel processing is continually concerned about how to supply all the processing nodes with data. Many of the applications favor particular data patterns that could be accessed in parallel. This is utilized in parallel memories, where the idea is to increase memory bandwidth with several memory modules working in parallel and feed the processor with only necessary data. Traditional identical memories are application specific and support just fixed data access requirements. In this Thesis, memory flexibility is increased to give support for several algorithms by adding run-time configurability to parallel memories. The multitude of data access templates and module assignment functions can be used within a single hardware implementation, which has not been possible in prior embedded parallel memory systems. The design reusability of the memories is also

improved since the same memory system is applicable in several separate implementations.

Three novel parallel memory architectures are presented in this Thesis: one traditional application specific type and two with run-time configurability. The results show that runtime configurability can be included in parallel memories with a reasonable cost. As a case study with four memory modules, the normalized complexity of the proposed configurable identical memories is 63–80% less than the conventional type of parallel memory. Moreover, in configurable parallel minds, the complexity increase in permutation networks is expressed to become the most critical when increasing the memory module count. According to evaluations, up to 79% of the total parallel memory gate count is consumed by the permutation networks excluding memory cells.

3.6 Parallel Memory Principles

A generalized block diagram of parallel memory architecture is depicted in the function is an Address Computation unit, N memory modules S0, S1… SN-1 and a Data Permutation unit. Depending on the access format F and the location of the first element (scanning point) r, the Address Computation unit computes the addresses and directs them to the appropriate memory modules. The Data Permutation unit organizes the data into correct order specified by the access format and Scanning point.

3.7 Memory system classification

The memory system classification on interleaved and parallel memory architecture used in this Thesis. Both memory configurations utilize N memory modules S0, S1… SN-1 was working in parallel. Moreover, M processing elements PE0, PE1… PEM-1 are being used with parallel memories ($M \leq N$). Interleaved memories use time-multiplexed identical memory modules that receive access requests serially one by one. Traditionally, vector computers utilize interleaved memories. Respectively, parallel memories are defined as space-multiplexed memories that are used, for example, in SIMD processing. Identical memories have the full address and data buses, and the memory modules receive access requests in parallel. In some multifactor supercomputers, a memory system with plenty of memory modules may be arranged in both space- and time-multiplexed manner. This Thesis concentrates on parallel memories.

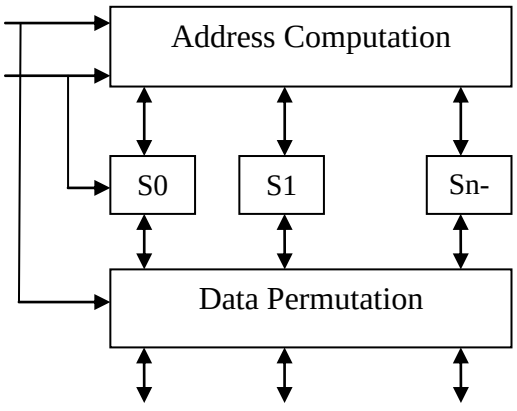

Figure.2 A generalized block diagram of a parallel memory architecture

International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No. (01), January 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

3.8 Simulation Results

ModelSim PE, our entry-level simulator, offers VHDL, Verilog, or mixed-language simulation. Coupled with the most popular HDL debugging capabilities in the industry, ModelSim PE is known for delivering high performance, ease of use, and outstanding product support. Model Technology's award-winning Single Kernel Simulation (SKS) technology enables transparent mixing of VHDL and Verilog in one design. ModelSim's architecture allows platform independent compile with the outstanding performance of the native compiled code. An easy-to-use graphical user interface enables you to quickly identify and debug problems, aided by dynamically updated windows. For example, selecting a design region in the Structure window automatically updates the Source, Signals, Process, and Variables windows. These cross-linked ModelSim windows create a powerful easy-to-use debug environment. Once a problem is found, you can edit, recompile, and re-simulate without leaving the simulator. ModelSim PE fully supports the VHDL and Verilog language standards. ModelSim PE also endorses all ASIC and FPGA libraries, ensuring accurate timing simulations.

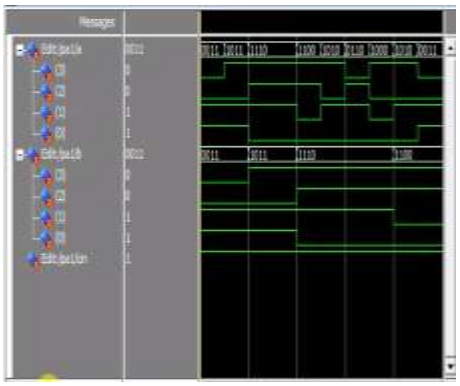

Figure.3 output waveform for 25% duty cycle

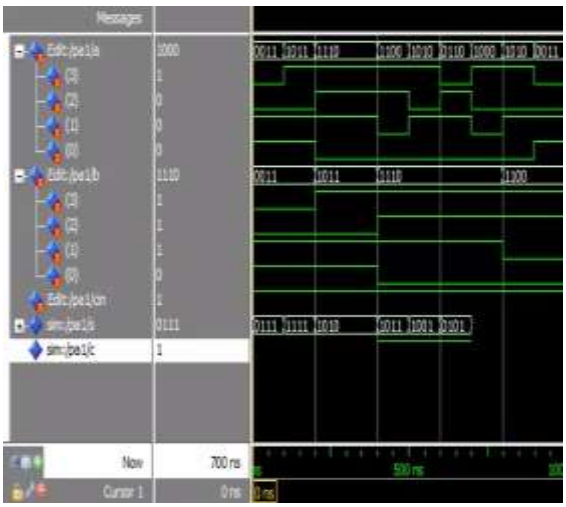

Figure.4 output waveform for duty cycle

IV . CONCLUSION

The Computation-In-Memory (CIM)-based parallel adder and estimated its performance. Despite the simplicity of the case study, the results show that CIM architecture has an enormous potential and orders of magnitude improvements. This is mainly due to reducing/eliminating memory accesses, using the non-volatile technology, and exploiting the high level of parallelism. CIM architecture seems to be very promising and could enable computation of current infeasible finally, it has significant data applications, fuelling critical societal changes.

REFERENCES

[1] H. Esmaeilzadeh et al., "Dark silicon and the end of multicore scaling," SIGARCH Comput. Archit. News, vol. 39, pp. 365–376, 2011.

[2] S. Borkar, "Design challenges of technology scaling," IEEE MICRO, vol. 19, pp. 23–29, Jul 1999.

[3] J. W. McPherson, "Reliability trends with advanced CMOS scaling and the implications for design," in IEEE CICC, 2007, pp. 405–412.

[4] S. Borkar, "Design perspectives on 22nm CMOS and beyond," in DAC. ACM, 2009, p. 9394.

[5] G. Declerck, "A look into the future of nanoelectronics," in Symposium on VLSI Technology, Digest of Technical Papers, 2005, pp. 6–10.

[6] G. Gielen et al., "Emerging yield and reliability challenges in nanometer CMOS technologies," to DATE. ACM, 2008, p. 13221327.

[7] S. Kaxiras, Architecture at the End of Moore, ser. Advances in Atom and Single Molecule Machines. Springer Berlin Heidelberg, 2013, pp. 1–10.

[8] J. W. Janneck, "Computing in the age of parallelism: Challenges and opportunities," Keynote talk, 2013.

[9] A. W. Burks et al., Preliminary discussion of the logical design of an electronic computing instrument (1946). Ablex Publishing Corp., 1989, pp. 39–48.

[10] S. A. McKee, "Reflections on the memory wall," in CF. ACM, 2004.

[11] W. H. Kautz, "Cellular logic-in-memory arrays," IEEE Transactions on Computers, vol. C-18, pp. 719–727, 1969.

[12] D. G. Elliott et al., "Computational RAM: implementing processors in memory," IEEE Design Test of Computers, vol. 16, pp. 32–41, 1999.

[13] P. M. Kogge et al., "Pursuing a petaflop: point designs for 100 TF computers using PIM technologies," in Symposium on the Frontiers of Massively Parallel Computing, 1996, pp. 88–97.

[14] D. Keitel-Schulz and N. Wehn, "Embedded DRAM development: Technology, physical design, and application issues," IEEE Des. Test, vol. 18, pp. 7–15, 2001.

[15] P. M. Kogge, "EXECUBE-a new architecture for scaleable mpps," in ICPP, vol. 1, 1994, pp. 77–84.

[16] D. Patterson et al., "A case for intelligent RAM," IEEE Micro, vol. 17, pp. 34–44, 1997.

[17] Y. Kang et al., "FlexRAM: Toward an advanced intelligent memory system," in ICCD, pp. 5–14.

[18] J. Draper et al., "The architecture of the diva processing-in-memory chip," in ICS. ACM, pp. 14–25.

[19] T. L. Sterling and H. P. Zima, "Gilgamesh: A multithreaded processor in-memory architecture for petaflops

computing," in Supercomputing Conference, pp. 48–48.

[20] N. Venkateswaran et al., "Memory Processor: A novel design paradigm for supercomputing architectures," ser. MEDEA '03. New York, NY, USA: ACM, pp. 19–26.

[21] E. Upchurch et al., "Analysis and modeling of advanced PIM architecture design tradeoffs," in Innovative Architecture for Future Generation High-Performance Processors and Systems, 2003, pp. 66–75.

[22] S. Hamdioui et al., "Memristor-based Computation-in-Memory architecture for data-intensive applications," to DATE, 2015.

[23] L. O. Chua, "The fourth element," JPROC, vol. 100, pp. 1920–1927, 2012.

[24] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs. Oxford University Press, 2000.

[25] A. Siemon et al., "A complementary resistive switch-based crossbar array adder," IEEE JETCAS, 2014.

[26] A. A. El-Slehdar et al., "Memristor-based N-bits redundant binary adder," Microelectronics Journal, vol. 46, pp. 207–213, 2015.

[27] "The international technology roadmap for semiconductors ITRS," 2011.

[28] C. Meinhardt and R. Reis, "FinFET basic cells evaluation for regular layouts," in IEEE LASCAS, 2013, pp. 1–4.

[29] A. Muttreja et al., "CMOS logic design with independent-gate FinFETs," in ICCD, 2007, pp. 560–567.