Design and Performance analysis of Memristor and IMPLY Adder based 64-bit Vedic Multiplier and CAM Memory with Gbps throughput on FPGA

SHRUTHI K. N., R. BHAGYALAKSHMI, ROOPASHREE D.
Department of Electronics and Communication,
Government Engineering College, Hassan, Karnataka,
INDIA

Abstract: - Memristors are a new area with various intriguing properties that make them useful for both storage and computing. We propose a semi-serial IMPLY-based adder that uses Memristor to design high speed and high throughput with minimal latency 64-bit Vedic multiplier and provide a detailed study of its benefits and proposed system focuses on the design of Content Addressable Memory. A fundamental property of the given adder, in comparison to state-of-the-art adders, is its simplicity. Based on a quality factor that gives the series of steps and the requisite die area equal weight researchers indicate that the suggested multiplier outperforms prior attempts. The proposed system is validated using key metrics including Figures of Merit, and detailed comparison analyses are carried out to better understand centered mathematical entities, their features, strategy aspects, and benefits and downsides when equated. This makes it easier for scientists in charge of layout and investigators in the field to create, or select, appropriate units. Domain-specific logic circuits based on memristors may conduct logic operations and store logic values, providing an attractive prospect for the creation of complex intellectual architectures. A novel stateful logic implementation based on memristors has been proposed in this paper. Single-input NOT and COPY operations and multi-input AND, OR, NAND, NOR, and CAM memory manipulations are all possible with the proposed technique. Non-volatile memristor resistances are employed as output and input states in each logic gate, allowing stateful logic operations to be performed. When compared to other methods, the suggested method can result in a multi-functional stateful logic circuit that can conduct many stateful logic operations at the same time. The effectiveness of the proposed design is illustrated using MATLAB to verify the basic characteristics of Memristor and synthesized in Vivado Design Suite 2018.1 platform and compared with theoretical calculations. Based on obtained outcomes in terms of hardware utilization and speed, throughput, and latency, 11% improvement in throughput, 31% improvement in speed, 9% in latency, and a 15% reduction in area.

Key-Words: - Gates IMPLY based Adder, 64-bit Vedic Multiplier, CAM, FoM, Memristor, FPGA, and Clock gated techniques

Received: July 21, 2021. Revised: June 15, 2022. Accepted: July 21, 2022. Published: September 1, 2022.

1 Introduction

Coefficients and Adder circuits are fundamental for constructing frames of ALU since almost each arithmetic operation on an Addition and multiplication is required by software systems. Due to the extensive use of computation-intensive machine learning applications like Neural Networks, the efficiency of these two units is now even more critical. As a result, we concentrate on efficient full adder design in this study, demonstrating that the suggested proficient adder can be employed directly in multipliers and provides enactment comparable to optimal multiplier implementations. We picked Material Implication logic [31] and [33] from a wide spectrum of memristor-based logic [21] and [30] because of their compatibility with Integrated marketing communications and crossbar structure [31] - [32]. Nevertheless, we should point out that INDICATE isn't the only logic having these characteristics; others, such as Memristor-Aided Logic [25] and Firm and Sense which requires less energy in Memory[21], also have them. In *a _ b*, the ASSUME operation leads to logic ‘zero’ High Resistance State or *Roff*, only if, *a has* a logical value of ‘one’ Low Resistance State or *Ron* and *b* have a rational worth of 'zero'. To accomplish this function in a Memristive circuit, connect memristors a and b as indicated in Figure one and apply 2 static powers to those Memristors, respectively VCOND and VSET. [31]– [33] provides thorough evidence on SUGGEST process and in what way to pick the 2 static powers and resistor Gate resistance.

The power consumption of a Field Programmable Gate Array is made up together with stagnant and active modules. Even though stagnant control has increased with technological advancements, it still accounts for just 10% of the overall power consumed by Field Programmable Gate Arrays. Moreover, recognized strategies for lowering it have previously been applied to Field Programmable Gate Arrays, such as power supply that can be changed, different gate oxide stiffness, multiple threshold levels for transistors, and diversion of energy. Dynamical strength, alternatively, accounts for above 90% of the wattage used by FPGA and is the primary source of their power inadequacy. This inappropriate use of energy is due to the significant latency in computing of Field Programmable Gate Arrays, which includes the MOSFET circuits that allow data to flow, multiplexers, and delays, and in the programmable routing fabric, configuration memory is used, which takes up 50% to 80% of the silicon surface. In comparison to cell-based systems, this outcome in significantly destinations that are larger and, as just a consequence, significantly higher capacitive loading. As a consequence, the programmable routing fabric consumes 60 percent to 80 percent of the total FPGA power supply. As a result, the FPGA's configurable routing system consumes a lot of power. Because of the inclusion of programmable logical constructs and reconfigurable interrelate in the programmable routing structure, more power is consumed. The connectors that can be programmed in FPGAs account for around ninety percent of the overall size, about eighty percent of the entire latency, and about eighty-five percent of the total power consumption, according to research. Memory Static Random Access Memory, off-chip DRAM, or Flash memory access time, density, and power consumption have a direct impact on the Field Programmable Gate Array's performance. Because of the substantial use of Static Random Access Fragments of recollection coding, multiplexers or pass transistors, and buffers in interconnects, typical FPGAs suffer greatly from their programmable interconnects. SRAM cells can have up to seven transistors, however, they can only store one bit of data. As SRAM-based storage has a low density, it adds to the area overhead of FPGA programmability, resulting in longer routing paths and longer connection delays. Furthermore, because static random access memory is a sort of memory that is in constant flux, it adds to the high usage of energy in standby mode. Dynamic RAM has the drawback of storing data as an electric charge that must be refreshed regularly.

2 Related Work

SRAM-based Field Programmable Gate Array and CAM devices make up an existing technology. Static random-access memory is a type of electronic memory that uses bi-stable gate systems to record every value. It is distinguished from Dynamic RAM, which must be updated regularly [1]. Data remanence is present in SRAM; however, it is still fragile in the fact that when the memory is turned off, information is missing. Because Static Random Access Memory is more expensive and less dense than Dynamic RAM, it isn't employed in high-capacity, low-cost applications like personal computers' system memory. Six MOSFETs make up a standard SRAM cell. SRAM stores each bit on two cross-coupled inverters made up of four transistors [2]. There are two stable states in this storage cell, which are denoted by the numbers 0 and 1. In a six-transistor (6T) SRAM, two additional access transistors regulate access to a storage cell during reading and write operations. A Static Random Access Memory cell might be in one of 3 states. It can be in one of three states: standby (when the circuit is not in use), reading (when data is required), or writing (updating the contents). Readability and write stability are required for the SRAM to function in both read and write operations. SRAM-based programmable interconnects and a fixed buffer pattern characterize typical FPGAs. As a result, the FPGA's configurable routing system comprised of SRAM consumes a lot of power. The existence of programmable logic blocks and programmable intersects in the programmable routing structure consumes a greater amount of electricity [3]. The adaptive connectors in FPGAs account for around ninety percent of the overall size, about eighty percent of the entire latency, and about eighty-five percent of the total usage of energy, according to research. Memory SRAM, off-chip DRAM, and Flash memory time complexity, compactness, and leakage current are all factors to consider all having an impact on the FPGA's actual quality. Conventional Field Programmable Gate Arrays suffer considerably from their programmable intersects due to the broad use of SRAM-based pieces of coding, multiplexers, and barriers in connections. One static random access cell can have up to seven transistors, but it can only store 1 bit of data. Because static random access-based storage has a low density, it adds to the area overhead of FPGA programmability, resulting in longer routing paths and longer connection delays. Furthermore, because static random access is a sort of unstable recollection, it adds to high standby energy usage [4].

Energy Consumption in Conventional SRAM: The SRAM's energy consumption can be separated into two types: dynamic and stagnant dynamism ingestion. Stagnant dynamism expenditure owing to SRAM leakage current and active usage of power due to capacitance charging and de-charging during query processing.

Dynamic Energy Consumption during reading Operation: When reading operations, there are four primary sources of energy dissipation: 1-energy dissipation during charging and draining data-line capacitance. 2- Dissipation of energy during word-line capacitance charging and discharging 3- Dissipation of energy during word-line capacitance charging and discharging 4- When the row address is decoded and one word-line is asserted in a memory array row, all cells in that row are connected to a bit-line, causing the unselected bit-line to discharge, resulting in energy being dissipated on these circuits. Associated with the four major causes of energy dissipation during reading activity stated above, dynamic energy consumption in SRAM for a normal read operation is calculated as follows:

$$Energy_{consumption} = C_{w_t}V_{dd}^2 + C_{DL}V_{dd}^2 + C_{BL}V_{dd}^2 - \frac{1}{2}V_{BL}^2(C_{BL} + C_{DL}) + (2^{col} - 1)x[C_{BL}V_{dd}^2 - \frac{1}{2}\frac{(C_{BL}V_{DD} - I_{PT}\Delta T)^2}{C_{BL}}] \quad (1)$$

VBL is the voltage of the bit-line at the end of the read operation, VDD is the external stream power, ΔT is the interval time that the cell's access transistor is turned on, and IPT is the current of the cell's in the saturation zone of a transistor, and these values are obtained using the following equation:

$$I_{PT} = \frac{\mu_n C_{ox}}{2}\frac{W}{L}(V_{GS} - V_m)^2 \quad (2)$$

Where n is the motion of electrons, Oxide capacitance is the gate capacitance per unit area, W is the width of the cell's access transistor, L is the length of the cell's access transistor, and VGS is the gate-source voltage of the cell's access transistor, and Vtn is the threshold voltage of the cell's access transistor.

Dynamic Energy Consumption during Write Operations: For a regular write operation in conventional SRAM, the dynamic energy consumption is given by:

$$Energy_{write} = C_{w_t}V_{dd}^2 + C_{DL}V_{dd}^2 + C_{BL}V_{dd}^2 + (2^{col} - 1)x[C_{BL}V_{dd}^2 - \frac{1}{2}\frac{(C_{BL}V_{DD} - I_{PT}\Delta T)^2}{C_{BL}}] \quad (3)$$

The formula is based on the fact that the voltage on the bit-line, word-line, and data-line changes during a write operation, resulting in energy consumption similar to a read operation. External SRAM has two major drawbacks in an FPGA-based embedded system: cost and board space. Other high-capacity memory formats, such as SDRAM, are more expensive per M Byte than SRAM systems. It also takes up more board space per M byte than SDRAM and FPGA on-chip memory, both of which take up none. In a Field Programmable Gate Arrays-based embedded system, information that is stored on the chip provides the best quantity and the shortest delay. It usually just has a one-clock-cycle lag. Recollection interactions able to be piped, resulting in a typical transaction rate of one per clock cycle. The dual-port mode can be used to access some types of on-chip memory, using distinct terminals for reading and writing dealings. Mode with two ports pairs the memory's wideband possibility by permitting it to be printed on one port and read on the other. On-chip memory can help developers save time and money. Moreover, several forms of on-chip memory can be automatically initialized with customized information during FPGA construction. The storage can be used to store small amounts of wader code or Lookup Table data that must be retained at reset.

The inclusion of static random access memory-based interconnects in the field-programmable gate array construction leads to substantial usage of energy, which is one of the major disputes in the activities. Due to the bandwidth, complexity, and power supply, indulgence of memory SRAM, and off-chip Dynamic RAM, have a direct impact on the overall performance of the Field Programmable Gate Arrays, these linked works are mostly focused on reducing or replacing the transistor count in the static random access memory. Traditional Field Programmable Gate Arrays suffer greatly from their programmable interconnects due to the extensive use of SRAM-based programming bits, multiplexers, or passes transistors and shields. One static random access memory cell can have up to seven transistors, but it can only store one bit of data. Because SRAM-based storage has a low density, it adds to the area overhead of FPGA programmability, resulting in longer routing paths and longer connection delays. Furthermore, because Static Random Access Memory is a sort of unstable reminiscence, it adds to high power consumption in standby mode. To address the shortcomings of previous work, a novel approach is developed in which the programmable interconnect of memristor-based FPGA architecture use only newly discovered

circuit elements, such as memristors and metal wires, rather than the static random access memory-based interconnects used in the traditional field-programmable gate array construction. When compared to previous ways, the key benefit of this strategy is that the complete FPGA architecture consumes less power. Because the memristor is a nanodevice, it does not take up more space than similar works. The use of Resistive RAM rather than Static RAM is the key benefit of this technology.

3 Proposed Hybrid Techniques for Optimization of Power, Area, Latency, and Improvement of Throughput

The proposed methodology aims to reduce the number of CLB in FPGA for area reduction by employing a Memristor-based FPGA design.

3.1 Memristor Multiplier and CAM Memory Design

The neoteric Memristor-based Field Programmable Gate Array architecture was employed in this project. Instead of programmable interrelates consisting of SRAM memory cells, Memristor-based FPGAs use Memristor interconnects. The proposed technology uses a three-D memory architecture that relies on crossbars and a three-D technique for mounting sheets, as well as protecting inductance from unneeded routing paths. Memristor-based connectors are customizable field-programmable gate array architecture use only the newly discovered trail component, namely Memristor and metal wires, rather than the static random access memory-based intersects used in conservative field-programmable gate array architecture, resulting in significant reductions in overall area power consumption, area, interconnect delay, and FPGA speed. Because Memristor is a Nano device, it does not take up more space than the existing system. The use of Resistive RAM rather than Static RAM is the key advantage of this method. The key benefits of using a Memristor and an IMPLY-based adder for memory design are as follows:

- Exceptionally compact, negligible leakage, CMOS compatibility, and high power efficiency.
- During read-write operations in the resistive crossbar avoids sneak-path.
- For high speed, it employs an adaptive buffer insertion mechanism.
- Eliminates the problem of increased area and interconnect delay, as well as increased power usage.

In this paper, a separate device analysis of both the SRAM and the Memristor is performed first, followed by the design of a reliable FPGA with only one transistor using the HSPICE simulation tool. The project's main goal is then accomplished by using the Vivado Design Suite software tools to design both SRAM and Memristor-based application circuits in FPGA. Furthermore, a comparison of static random access memory and Memristor-based field-programmable gate array architectures is performed to demonstrate that the memristor-based FPGA architecture consumes less power. Content addressable memory is a type of solid-state memory in which data is accessible by its content rather than by its physical location. It takes search data in the form of a search term and provides the address of a comparable word stored in its data bank [1]. READ, WRITE, and COMPARE are the three basic operation modes of a CAM, with "COMPARE" being the most significant of the three because CAMs rarely read or write.

3.2 Architectural Level Delay Reduction Techniques using Memristor

We have numerous solutions for decreasing the critical route in the data path, as well as power gating, at the design level. This section discusses some of the delay reduction approaches that will help CAM overcome its overall searching speed.

There are only three passive elements with two terminals each available in circuit theory: resistor, capacitor, and inductor. These elements are defined by the relationship between fundamental circuit variables such as current, voltage, charge, and flux. Prof. Leon Chua anticipated that a 4th essential component of a network would be needed to establish the relationship between charge and discharge of magnetism in 1971. The nano-machine memristor is a non-active component that remembers the previous state it was used in. The Memristor was a type of element of Memristive element that varied its resistance in response to the quantity of control flowing over it. The Memristor functions similarly to a linear resistor with memory, but it also has several fascinating nonlinear properties. The Memristive, memcapacitative, and meminductive subsystems make up the Memristive class. Such components are regarded as a single port element whose properties are determined by the charge and flux linkage's time differential. The relationship between fundamental circuit parts is

depicted in Fig.1, which also fills in the gap between charge and flux linkage. The Memristor's key characteristics are as follows:

- Has Resistive Random Access Memory.
- Memristors are tiny and scalable down to 5x5 nm.
- At 180nm, it can be programmed in 5ns and consumes less power.
- It necessitates a very low biassing voltage and quick switching - up to six orders of magnitude - due to the very non-linear rate of switching.
- Because of its nano size, it performs better than traditional Non-Volatile Memory (NVM) elements.

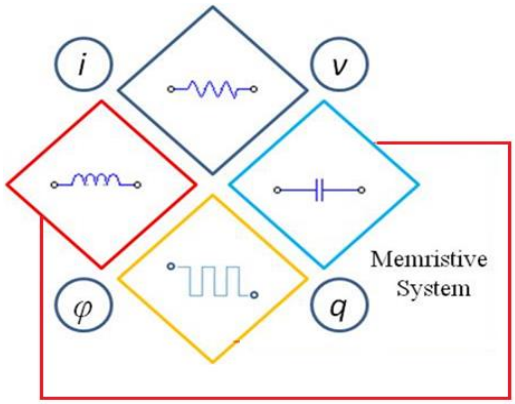


Fig. 1: Relationships between Four Fundamental Circuit Elements

The Memristor was projected using the regularity values of two of the four fundamental electrical measures: current I voltage, charge, and flux. A memristor is a semiconductor reedy picture inserted among two metallic associates with an entire distance of D of TiO2 film with doped low and un-doped high resistance regions with a total length of D of TiO2 film. Symbol 2 depicts the fleshly assembly and its comparable track perfect. The memristor can raise resistance in one direction of current while decreasing resistance in the other. So when ambient energy that has been applied is removed, the memristor returns to its previous condition, indicating that it has resistive memory. To put it another way, a memristor is an analog resistor whose resistance may be altered by changing the direction of applied voltage or current. The basic geometrical structure of a memristor is shown in Figure 2. Figure 3 depicts the simulation findings. The wideness of the entire constituent is denoted by D, whereas the fatness of the nobbled coating is denoted by w. To adequately illustrate the essence of the models and simulations, a mathematical memristor model must be presented.
As seen below, the memristor functioned as a recollection resistor by connecting the voltage across the component and the current flowing finished it,

$$v = M(w)i \ ) \quad (4)$$

The memristance functions similarly to resistance, with the exception that it is dependent on a parameter, which in Chua's sources was either the charge. Because charge and current are connected in the following way,

$$\frac{dq}{dt} = i \quad (5)$$

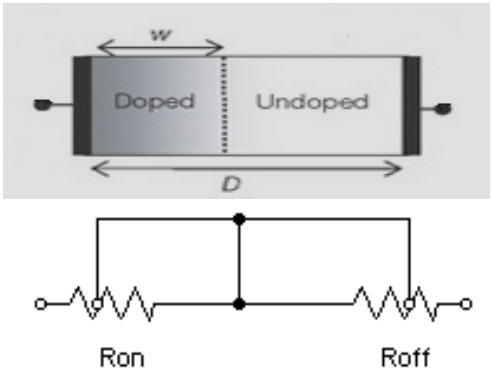


Fig. 2: HP Laboratory's Memristor Structure and Its Corresponding Ideal

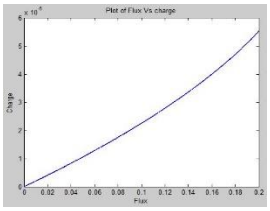
The memristor acts like a resistor with memory since M is dependent on the whole history of the current going finished component. The nonlinear memristance M is a function of charge q, and no RLC element combination mimics or copies such a feature, making it a fundamental circuit element. Chua later demonstrated that memristors are part of a larger class of systems known as memristive systems, as defined by,

$$v = M(w,i)i \quad \frac{dw}{dt} = f(w,i) \quad (6)$$

Equations will be used to express the hysteresis loop of the memristor (6). Equations (7) and (8) define the optimal mathematical model of a memristor (8).

$$V(t) = \left[\left(\frac{Ron\, W(t)}{D}\right) + Roff\ \left(1 - \frac{W(t)}{D}\right)\right] i(t) \quad (7)$$

$$\frac{dw(t)}{dt} = \frac{\mu v\, Ron}{D}\, i(t) \quad (8)$$



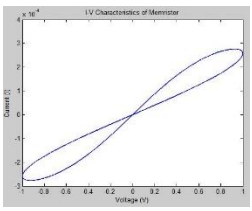


Fig. 3: Simulative Flux V/S Charge and Simulative Current V/S Voltage Plots

3.3 Mathematical Modeling of a Memristor

The fourth Memristor along with RLC components is two pins circuit and it can be defined in terms of two variable charges, voltage and current is given by

$$q(t) = \int_{-\infty}^{t} i(\tau)d\tau$$

$$\varphi(t) = \int_{-\infty}^{t} v(\tau)d\tau$$

It's worth noting that "q" and "are mathematically defined and don't need any physical meanings. Nonetheless, we refer to the charge and flux of the memristor as q and f, respectively, because these correspond to the formulas connecting charge to current and flux to voltage. Charge-controlled or flux-controlled memristors are described as such.

$$\varphi = \overline{\varphi(q)}$$

$$v = \frac{d\varphi}{dt} = R(q)i$$

Where $i = G(\varphi)v$

Here ϕˆ (q) & qˆ (ϕ) represents the unremitting and piecewise-differentiable purposes with restricted grades is entitled the inductance at ϕ, and have the unit of Siemens (S). It's worth noting that it's comparable to Ohm's law, but the resistance is different R (q) at a given time t = t0 is determined by the full functionality of i(t) from t = t −∞ t = t0. Similarly, in eqn, the conductance G (ϕ) is determined by the full history of v (t) from t =−∞ to t = t0. The charge-controlled memristor is thus characterized as analogous to the charge reliant on Ohm's law. It is important to note that if a memristor with a resistance R0 is opened or short-circuited for $t=t_0$ to bring the Memristor to equilibrium condition and at this condition, the values of V and I are zero therefore Memristor device does not lose its data. Whenever power is turned off then V and I go to non-zero values but minimum i.e negligible, but rather retains the value at q0 and ϕ0. As a result, the passive memristor has nonvolatile memory. At the three-phase, the charge and memristance are identical and its state map with the state equation dq/dt = I are corresponding to a Memristor in the sense that given applied current source input signal i(t) for all times from t =−∞, or equivalently, for positive times from t = 0, plus the initial charge q(0) which represents the time integral of i(t) from t =−∞to t = 0, one can calculate the (t). Inversely, if R > 0, the inverse constitutive relation q = ˆq (ϕ) is a continuous function, and given any v (t), the corresponding i (t) may be calculated. Most of the waveforms and hysteresis loops, on the other hand, are just memristor manifestations and cannot be used to predict the voltage response given any excitation waveforms other than I = Ascent, with A = 1 and ω = 1. Changing the parameter A, or the waveforms of I (t), or both, would produce radically different reactions. For example, as the hysteresis loop reduces until it collapses into a unit-slope straight line through the origin, we will see that q (t) tends to zero, v (t) tends to sin t, and R (t) goes to 1. Indeed, the charge q (t) and flux (t) would both trend to the origin and remain immobile thereafter, as they always do. In this special case, the memristor degenerates into a linear resistor, where R is simply the slope of the –q curve at the origin, i.e. R = 1.

4 IMPLY Based 64-Bit Adder with Memristor

The most common IMPLY-based adder designs are either serial or parallel. Parallel techniques are faster but require a large number of work Memristors, whereas serial systems require a minimum group of Memristors and hence compromise velocity. The purpose of our semi-serial adder design is to incorporate the benefits of both serial and parallel approaches to produce a more efficient design with an advanced Figure of Merit. In contrast to serial or parallel designs, the input variables ai and bi is separated into two sections in our semi-serial adder, while the five work memristors w1 to w4 and c and the carry-in memristor cin are separated into a third section. Our approach requires 2n + 6 Memristors: 2n for input and output variables, 4 work Memristors, and two carry Memristors. Each input section has its own work resistor RG, as shown in Figure 5, and each memristor in the independent third section can be linked to any of the two input sections, a and b. To store the resultant sum and carry, the input section a and the carry-in memristor cin are recycled. The formulations of summation and carry in SUGGEST logic, which we utilize to create the quasi adder, are shown in Equations (1) and (2).

$$Sum = \left[(\bar{a} \to b) \to \left((a \to \bar{b})\right)\right] \to \left(\overline{a \oplus b}\right) \to \bar{c} \quad (9)$$

$$Carry_{out} = [\overline{((a \to \bar{b}) \to c) \to \overline{\bar{b} \to a}}] \quad (10)$$

Because each bit is calculated serially, we call this structure semi-serial. Work memristors, on the other hand, are included in a distinct third segment, allowing for parallelism beyond what is feasible in serial design. Table I lays out the algorithm in detail. C is employed and transmitted throughout the main body of the program. The supposition is accurate in the intermediate phases because the algorithm creates and propagates c, although cin is normally provided at the beginning (not c). Furthermore, cout is usually preferred over cout at the end of the day. As a result, we propose one more step for the inversion of cin at the start of the algorithm and one additional inversion step at the end of the algorithm to comply with this. These phases, which are only performed once at the start and end of the algorithm execution, are underlined in bold blue in Table I. As a result, the total number of steps equals 10n + 2. For each step in the algorithm, Table II shows the connection status of memristors in the work memristors section (c, cin, w14). The letter "U" indicates that the memristor is connected to the top segment (a memristor), implying that the higher switch is closed while the lower switch is open. Similarly, an "L" indicates that the respective memristor is connected to the bottom portion, indicating that the top switch is open and the lower switch is closed. A dash ("-") signifies a "don't care" state, suggesting that the memristor could be attached to either of the sections because it is not used in that phase.

5 64-Bit Vedic Multiplier using Imply Adder and Memristor

The article's fourth addition is a novel multiplier design based on our semi-serial adder, which is shown there. A 2n 1 bit semi-serial adder circuit is reproduced 2n – 1 time for a and b, where a and b are binary values of size n, to form an nn-multiplier. The I -th adder starts with a2i and a2i+1 in its work memristors wi,0 and wi,2. Every adder has b moved to the left (one bit) in both summand registers 4 where b is the second input of the product a b. Initially, each adder calculates the relevant set of partial products, i.e., the partial products a2ib j and a2i+1b j are calculated simultaneously in the I -th adder, where j is a natural number in the range [0, n1]. As a result, the operation akxb j = ak_b j, which represents the calculation of one partial product, is executed. The statement has been rephrased as Three IMPLY steps are required for this logic process. The adders calculate the overall product by summing all partial products step by step after computing the component products. This means that each adder's sets of partial products are totaled first, and then two by two adders are connected to calculate intermediate sums until all sets of partial products are totaled. The output or calculated product of a b is represented by this total sum. The result of the multiplication and how it was done will be stored in the cin & memristors of the system's initial adder (cin as the most significant bit or carry-out, and an as the rest of the output value). Table VII shows a two-bit example of the algorithm that is used to calculate partial products. The adders resume their semi-serial adding procedure once partial products have been determined. Figure 8 depicts a 44-bit multiplier made up of two semi-serial adders and one additional switch used to connect the adders during the summing phase. The semi-serial adder circuit does not need to be changed except for the additional switch. In other words, for an n n-bit multiplier, the adder is only repeated n/2 times. The total number of additional switches required is n-1/2.

6 Results and Discussion

The proposed multiplier design was simulated at two different levels of abstraction. A Matlab simulation was used to test the behavioral accuracy of the multiplier method, assuming optimal memristor behavior. Because the multiplier approach is split down into partial product calculation and subsequent addition, we confirmed the accuracy of partial product computation on the circuit level by simulating it in Vivado platform using the Modelsim software, using the same configuration as in Section 4. Our semi-serial adder algorithm, which was verified in Section 5, is used to do the succeeding addition. The LT Spice simulation of the calculation of one partial product a b, where a = 1 and b = 1, is shown in Figure 9. The work memristors w1 and w2 are expected to be initialized to HRS (logic '0') in this example. In w2, the result of a b is saved. For all input combinations of a and b, Figure 10 shows the proper calculation of w2 = a b. In both simulations, an IMPLY step takes 30 seconds. The technique was carried out using a SPICE implementation of the VTEAM model, as in Section III-B. The same variables were used as in the previous example.

Table VIII compares based on multiplier architecture to others in the literature for a 32-bit multiplier in terms of features and performance metrics. We couldn't determine the number of

required memristors, steps, or switches for n = 32 because didn't supply any equations. The FoMs in Table X are calculated using n = 32, but the FoMs in Table XI are generated using n = 8, because, as previously stated, the source for the Dadda multiplier only offers efficiency and characteristic figures for an 8-bit multiplier. In both tables, the best design according to each Figure of Merit is boldfaced to make it easier to spot. The equation was used to determine the improvements (8). As shown in Table X, our multiplier design outperforms array- and Dadda-type multipliers by a factor of 50, owing to significant reductions in the number of memristors and switches used. A Shift & Add multiplier with a 32-bit implementation surpasses all other multiplier designs in 4 out of 5 Figure of Merit, while its 8-bit version outperforms others in 3 out of 5 Figure of Merit. This is caused by a few factors;

- The original performance of the Shift & Add coefficients,
- The design's basic components, for an example multiplexers and transferal records, have a high level of optimization built-in, and
- Multiplexers and shift registers are common examples of external CMOS circuitry.

The final component, in particular, makes a comparison with our suggested multiplier problematic, as our design makes far less use of external Complementary metal–oxide–semiconductor motherboard. Furthermore, we have not considered, and will not examine, the Complementary metal–oxide–semiconductor tracks that are required to create the state machine that controls the Shift&Add, Array, and Dadda multiplier. It can tip the scales even further in our favor, particularly in the case of FoMA. Even when the aforementioned parameters are ignored, our suggested multiplier beats all previous designs in terms of FoMA in both 8-bit and 32-bit versions. When compared to the Shift&Add multiplier [40], for example, according to the majority of Figure of Merit is the best design, our approach is 532 percent superior in terms of FoMA. In other terms, our proposed solution saves over 5 times the amount of space as the Shift&Add multiplier. The Area-centered Diagram, we believe, is correct Merit from Equation (7) delivers the furthermost truthful estimate of worth when it comes to the die area because it not only considers Complementary metal–oxide–semiconductor circuitry but also the fact that additional Complementary metal–oxide–semiconductor motherboard is buried beneath the memristor crossbar due to the most common practice of using memristors in BEOL. Apart from that, we designed our multiplier by just duplicating our serial adder without adding any additional building blocks, keeping CMOS circuitry basic. The contrast of our concept with Array Multiplier is another example of one parameter's lack of representativeness in the worth of a design. Our approach is 58 percent slower than Array Multiplier, but because we use 70 percent fewer memristors, we exceed it in four out of five Figures of Merit. These two examples demonstrate that existence improved or worse in one aspect does not provide us with a whole picture of architecture's worth. As a result, a designer must select a Figure of Merit that best represents the design constraints, evaluate any design using the proper Figure of Merit, and make design choices that assist the system in achieving the criteria.

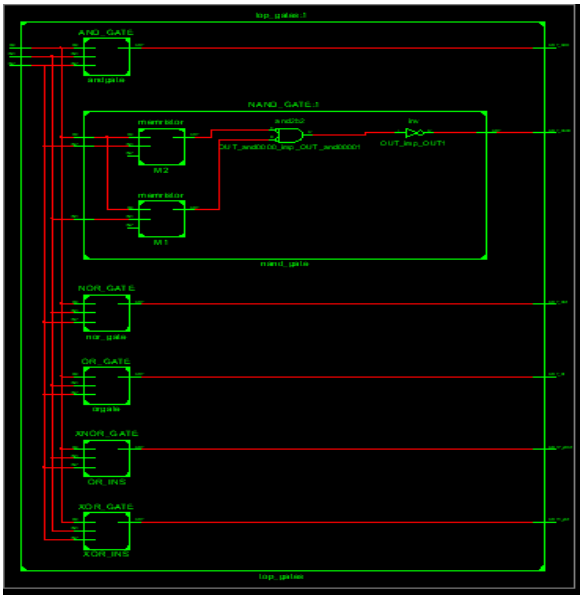


Fig. 4: RTL schematic diagram of Memristor-based gates.

Fig.4 shows the RTL diagram of Memristor-based gates and each gate consumed two Memristors for two inputs. The resistor values vary from 100 Ohm's 40KOhm to perform the exact logic of gates. The simulated results of all gates are shown in Fig.5.

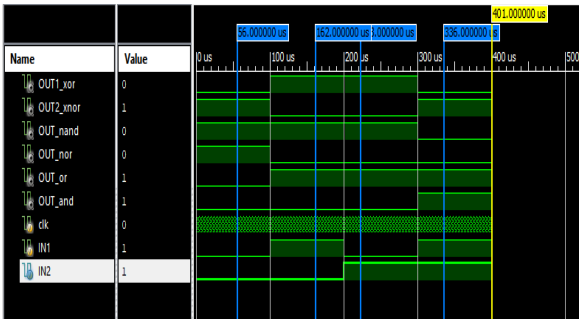


Fig. 5: Simulated results of Memristor based Gates

The analysis of proposed Memristor-based gates and multiplier and CAM memory is analyzed with three main metrics such as latency, throughput, and area utilization. The throughput is given and their calculated values are shown in Table 1

$$\text{Throughput} = \frac{\text{Frequency of operation}}{Latency} x Size\ of\ input\ data$$

Table 1. Comparison between Existing and Proposed Gates and 64-bit Vedic multiplier and Memristors with IMPLY based adder

Parameter	Vedic with PPA based adder [12,9,24]	Vedic with IMPLY based adder and Memristor	Conventional Gates without Memristor	Memristor based Gates
Slice Registers (Area)	7809	7818	204	192
Slice LUT's	16472	14491	409	398
Flip-Flops	7940	7808	215	206
Delay in ns	17.313 ns	17.208	7.3	6.175
Power in Watts	0.491	0.491	0.88	0.450
Frequency in MHz	185.4	250.627	185.8	291.886
Throughput in Mbps	205.6	227.8	34.5	47.2
Latency	19.4	17.6	6.4	6.4

Table 2. Comparison between existing floating and proposed floating-point multiplier.

Parameter	Floating-point Vedic multiplier with PPA based adder [6,10,16]	Double precision floating point Vedic multiplier with IMPLY based adder and Memristor
Slice Registers (Area)	4816	3819
Slice LUT's	5002	4951
Flip-Flops	2451	2100
Delay in ns	4.6	3.841
Power in Watts	0.89	0.881
Frequency in MHz	217.5	261.4
Throughput in Gbps	3.2	4.3
Latency	5.61	4.651

The Memristor with Vedic multiplier is applied for the design of a double-precision floating-point multiplier and validated with a different floating number in both MATLAB environment and Vivado Design Suite 2018.1 software. As per Table.2, there is a 6% improvement in the area, a 4% improvement in LUT, and a 13% reduction in latency.

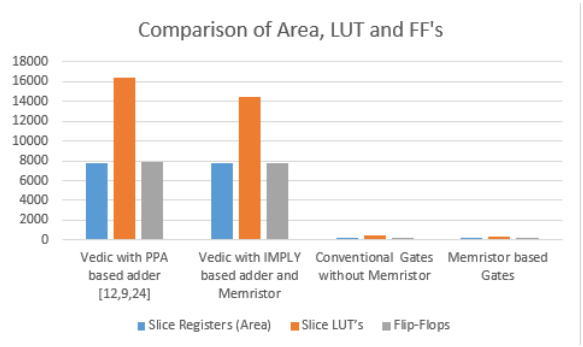


Fig. 5: Comparison of area, LUT, and flip-flops.

Figure 5 shows the suggested system's performance in terms of area, LUT, and flip-flops.

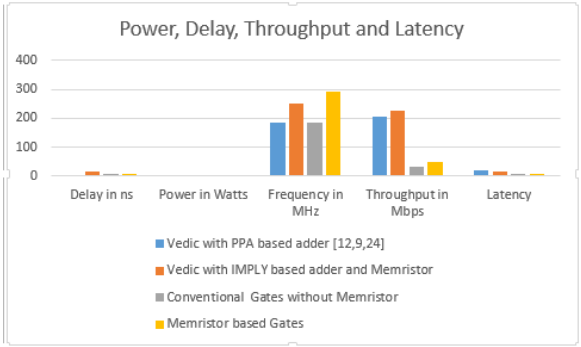


Fig. 6: Performance analysis of the planned system in positions of control, deferral, throughput, and latency.

6Conclusion

The paper presents a stateful Boolean logic implementation method for AND, OR, NAND, NOR, COPY, and NOT, as well as a Vedic

Multiplier built with Memristor and IMPLY-based adders. Each recommended logic gate uses a simple memristor connection, with the output memristor's logic state changing in reaction to the input memristor's logic states. Not only can the suggested logic gates conduct logic functions, but they can also store sense standards. Multi-input state complete logic gates are available in addition to 1-input and 2-input logic gates. Previous memristor-based logic designs are compared to the suggested approach. According to the results of the comparison, the provided approach can execute 6 simple stateful Boolean sense processes with compact circuit topologies. Furthermore, the future enterprise can be used to create a multi-functional journey, reducing the number of memristors required. The suggested product's validity is demonstrated by computer simulation results. The whole scheme presents a novel way for creating memristor-based stateful logic that combines logic value storage with logic operation to build an unconventional cognitive framework. The assessment of our semi-serial 3 out of 5 Figures of Merit for the SUGGEST-created Memristive full-adder design, which outperforms extra enterprises in the research, is enhanced in this study. The complete design is synthesized using Vivado Design suite platform, based on obtained results, the area is reduced by 13%, latency is decreased by 31% and power is minimized by 15%.. Proposed four new Figures of merit to improve the parallel of memristive systems in terms of efficiency. Regarding the design aim, a designer might select the most appropriate FoM. Furthermore, they can be inspired by the proposed FoMs to create a new FoM that more accurately depicts the particular of their architectural restrictions and trade-offs. We also reviewed the literature on SUGGEST-based multiplier designs and planned a new multiplier founded on our semi-sequential full-adder architecture. This repeater concept outperforms other approaches in the field when the area slide created by supplementary obligatory Complementary metal–oxide–semiconductor switches is taken into consideration. Calculations of power consumption and die area should be carried out in the future helps improve replicability and quality of work.

References:

[1]. S. Hamdioui, H. Aziza and G. C. Sirakoulis, "Memristor based memories: Technology, design and test," 2014 9th IEEE International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS), 2014, pp. 1-7, doi: 10.1109/DTIS.2014.6850647.

[2]. D. Niu, Y. Chen and Y. Xie, "Low-power dual-element memristor based memory design," 2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED), 2010, pp. 25-30, doi: 10.1145/1840845.1840851.

[3]. Prithivi Raj, M., Kavithaa, G. RETRACTED ARTICLE: Memristor based high speed and low power consumption memory design using deep search method. *J Ambient Intell Human Comput* **12,** 4223–4235 (2021). https://doi.org/10.1007/s12652-020-01817-2

[4]. M. R. Mahmoodi, A. F. Vincent, H. Nili and D. B. Strukov, "Intrinsic Bounds for Computing Precision in Memristor-Based Vector-by-Matrix Multipliers," in IEEE Transactions on Nanotechnology, vol. 19, pp. 429-435, 2020, doi: 10.1109/TNANO.2020.2992493.

[5]. D. Radakovits, N. TaheriNejad, M. Cai, T. Delaroche and S. Mirabbasi, "A Memristive Multiplier Using Semi-Serial IMPLY-Based Adder," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 5, pp. 1495-1506, May 2020, doi: 10.1109/TCSI.2020.2965935.

[6]. J. Vista and A. Ranjan, "Flux Controlled Floating Memristor Employing VDTA: Incremental or Decremental Operation," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 40, no. 2, pp. 364-372, Feb. 2021, doi: 10.1109/TCAD.2020.2999919.

[7]. O. Leitersdorf, R. Ronen and S. Kvatinsky, "MultPIM: Fast Stateful Multiplication for Processing-in-Memory," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 69, no. 3, pp. 1647-1651, March 2022, doi: 10.1109/TCSII.2021.3118215.

[8]. Chang *et.al*, Ultra-low-voltage, low power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits, IEEE Trans. Circuits Syst. I, Fundam. Theory Appl., 2004, 51, (10), pp. 19851997.

[9]. Raphael, D *et.al*, A Power-Efficient 4-2 Adder Compressor Topology, 15th IEEE (NEWCAS), Strasbourg, France, 2017, pp. 281-284.

[10]. O. Leitersdorf, R. Ronen and S. Kvatinsky, "MultPIM: Fast Stateful Multiplication for Processing-in-Memory," in IEEE

Transactions on Circuits and Systems II: Express Briefs, vol. 69, no. 3, pp. 1647-1651, March 2022, doi: 10.1109/TCSII.2021.3118215.

[11]. Shaahin, A *et.al*, Majority-Based Spin-CMOS Primitives for Approximate Computing, IEEE Trans. on Nanotech., 17(4), 2018, pp. 795-806.

[12]. Avinash, L *et.al*, Parsimonious Circuits for Error-Tolerant Applications through Probabilistic Logic Minimization, Int. Workshop on PATMOS 2011, pp.204-213.

[13]. DarinDarjn, E *et.al*, Approximate Multipliers Based on New Approximate Compressors, IEEE Trans. on CAS-I: Reg. Pap, PP(99), 2018, pp. 1-14.

[14]. Meijia Shang, Xiaoping Wang, A memristor-based circuit design for generalization and differentiation on Pavlov associative memory,

[15]. Neurocomputing, Volume 389, 2020, Pages 18-26, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2019.12.106.

[16]. Liang, J *et.al*, New Metrics for the Reliability of Approximate and Probabilistic Adders, IEEE Trans. on Comp., 63(9), 2013, p. 1760-1771.

[17]. Zervakis, G *et.al*, Design-Efficient Approximate Multiplication Circuits Through Partial Product Perforation, IEEE Trans. on VLSI Systems, 24(10), 2016, pp. 3105-3117.

[18]. X. Xu, X. Cui, M. Luo, Q. Lin, Y. Luo and Y. Zhou, "Design of hybrid memristor-MOS XOR and XNOR logic gates," 2017 International Conference on Electron Devices and Solid-State Circuits (EDSSC), 2017, pp. 1-2, doi: 10.1109/EDSSC.2017.8126414.

[19]. Y. Cho and M. Lu, "A Reconfigurable Approximate Floating-Point Multiplier with kNN," 2020 International SoC Design Conference (ISOCC), 2020, pp. 117-118, DOI: 10.1109/ISOCC50952.2020.9332978.

[20]. M. Hajizadegan and P. Chen, "Harmonics-Based RFID Sensor Based on Graphene Frequency Multiplier and Machine Learning," 2018 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting, 2018, pp. 1621-1622, DOI: 10.1109/APUSNCURSINRSM.2018.8608604.

[21]. K. Paramasivam, N. Nithya and A. Nepolean, "A Novel Hybrid CMOS-Memristor Based 2-Bit Magnitude Comparator using Memristor Ratioed Logic Universal Gate for Low Power Applications," 2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA), 2021, pp. 1-5, doi: 10.1109/ICAECA52838.2021.9675534.

[22]. N. C. Dao and D. Koch, "Memristor-based Pass Gate Targeting FPGA Look-Up Table," 2021 International Conference on Electronics, Information, and Communication (ICEIC), 2021, pp. 1-4, doi: 10.1109/ICEIC51217.2021.9369751.

[23]. N. C. Dao and D. Koch, "Memristor-based Pass Gate Targeting FPGA Look-Up Table," 2021 International Conference on Electronics, Information, and Communication (ICEIC), 2021, pp. 1-4, doi: 10.1109/ICEIC51217.2021.9369751.

[24]. M. Teimoory, A. Amirsoleimani, A. Ahmadi and M. Ahmadi, "A hybrid memristor-CMOS multiplier design based on memristive universal logic gates," 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), 2017, pp. 1422-1425, doi: 10.1109/MWSCAS.2017.8053199.