MOEM

Modern Electronic Materials 2024; 10(1): 19–28
DOI 10.3897/j.moem.10.1.113631

Research Article

Optimized register level transfer mechanism for ASIC-based memscomputing

B. M. Shilpa[1,2], Sathisha K. Shet[3]

1 *Sri Krishna Institute of Technology, 29 Hesarghatta Rd, Chikkabanavara, Bangalore, Karnataka 560090, India*

2 *Jagadguru Sri Shivarathreeshwara Academy of Technical Education, JSS Campus Rd, Srinivaspura, Bangalore, Karnataka 560060, India*

3 *Birla Institute of Technology and Science, Vidya Vihar, Pilani, Rajasthan 333031, India*

Corresponding author: B. M. Shilpa (shilpabm244@gmail.com)

Received 3 October 2023 ♦ **Accepted** 9 February 2024 ♦ **Published** 5 April 2024

Citation: Shilpa BM, Shet SK (2024) Optimized register level transfer mechanism for ASIC-based memscomputing. *Modern Electronic Materials* 10(1): 19–28. https://doi.org/10.3897/j.moem.10.1.113631

Abstract

Micro-electromechanical system-based semiconductor sensor systems typically need a reliable on-chip analog read-out circuit to identify and process changes in physical properties. To improve the mobility, robustness, reliability and performance (in terms of power consumption and speed) of the sensor system, the remaining parts of the post-processing logic, such as the digital data processor (or part of it), can also integrated into the chip. Cost and integration are always important considerations in both analog and digital designs, but they become even more important; the subsystem of data processing must be squeezed into a very small space near the structure of the micro-electromechanical system (MEMS) sensor. The paper introduces a new Buffalo-based register-transfer level (BRTL) method that aims to improve the efficiency and reliability of the system of digital data processing for MEMS sensor post-processing algorithm. Initially, the memcomputing system was designed with the needed functional processing elements. Then, a novel Buffalo-based register-transfer level design is modelled in the MEMS architecture. The digital data transmission process is performed to estimate the reliability of the proposed BRTL model. Finally, the performance of the proposed model can be validated.

Keywords

application specific integrated circuit (ASIC), Buffalo-based register transfer level, memcomputing, functional verification

1. Introduction

Memcomputing was one of the technologies, and it would provide noteworthy enhancements for the solution of hard optimization issues in contrast to conventional algorithmic techniques [1]. It stands for computing in and with memory [2]. It was the computing paradigm where the memory obtained the two tasks, such as storing and information processing [3]. Here, the MEMCOMP was a universal computer which simulated both the quantum computers and reservoir computers [4]. The universal memcomputing machine (UMM) was the ideal machine, provided with infinite long memory tape as a substitute for inter-connectivity memory cells, that was capable of performing either analog or digital operations, and these operations were controlled by the controlling unit [5]. The responsibility of this controlling unit was to transmit the signal, whereas the memprocessor's internal state was organized [6]. These UMMs are used to solve all the problems of computing in and with memory [7]. Non-volatile

memory, such as logic-in-memory and neuromorphic computing, has been introduced into the computational hierarchy [8]. Here, the arithmetic logic unit (ALU) and memory were merged into compact memory-ALU to achieve the logic-in-memory [9]. This process was focused on MEMCOMP through Boolean logical functions, including NAND, material implication (IMP) and NOR [10]. The binary-resistant switch memristor was able to perform a two-state transition, and the logical values were stored consequently in a similar functional unit [11]. Digital memcomputing machines (DMM) was defined as non-linear dynamic system which was designed to solve constraint-satisfied issues [12].

The 3-SAT formula was built by the seminal work of Bearden [13], which applies conjunction, disjunction and negation operations to Boolean variables such as TRUE or FALSE. First, transform the Boolean variables into continuous variables. These variables could be realized as voltages on the self-organizing OR gate terminals [14]. This gate could impact the terminals for pushing voltages concerning a configuration satisfied by its OR gate, which regards whether the signal received via the gate originates from the convention input or output [15]. The voltages were constrained with Boolean values recovered through thresholding ranges from (–1, 1) [16]. Application-specific integrated circuits (ASIC) design comprised of processing elements (PE) connected in various topologies provided with latency, incomparable energy efficiency and form factor [17]. Then, the hardware design was created through ASIC with adjusted PE and its connections [18]. It had optimized logic gates and hardware implemented uniquely for its application, and it was fabricated, redesigned and designed. The benefits of ASIC provided accurate computing, re-usage of data, local memory, memory bandwidth was higher, and MAC units were able to be modified based on ASIC [19]. In contrast, it provided a high performance per watt with no reconfigurability, the tape-out process was slow, and the development cost was very expensive.

Also, ASIC designs would diminish power consumption and make it suitable for mobile and embedded systems. In the hardware design, the implementation of data plane functionality in an ASIC [20]. This platform was used to provide high performance because of dedicational hardware components such as ternary content addressable memory chips (TCAM) for efficient packet matching. The application-specific hardware, such as ASIC, considered with the data loaded from memory, was said to be a bottleneck. In several phenomena, Winogad transformation was used as essential for less local data reuse; due to this, it might not be able to obtain the performance gain of the designing process. The ASIC provided with effective use of availability on chip-memory for the increment of data locality. In our study, the most effective ASIC-based MEMCOMP process was analyzed and designed in this study. The key objective of the work is to design the optimized MEMCOMP for the digital applications.

The key contribution of this present study is defined as follows,

– initially, the memcomputing system was designed with the needed functional processing elements;

– then, a novel Buffalo-based register-transfer level design (BRTLD) is modelled in the MEMS architecture;

– the digital data transmission process is performed to estimate the reliability of the proposed RTL model;

– finally, the chief metrics like delay, memory usage and power/energy utilization were measured and compared with other models.

The present work is presented in the form of related work in the second section. The problems that the conventional method faces are exposed in the section three. Then, the solution for the defined problem is elaborated in section four. The validated results for the novel solution are discussed in the section five. The end of the research paper was concluded with section six.

2. Related works

The memristive neuromorphic system included with the chips is implemented through integrated resistant switched memories on the CMOS hardware. The term is neuromorphic has been utilized to recognize analog, digital and mixed models of analog and digital integration design. This network was essential to construct the neurons and artificial synapses that were able to impersonate the complexity term of the biological counterparts. Since this network was inherently volatile, binary and poor scaled. The devices depended on the novel physical principles that were necessary for replicating the biological synapses and neurons in this network, which was consistent with the high-density ultra-connections and complexity processes [21].

The implementation was not available in formal technologies using memristors in circuits, which was one of the major problems. So, to overcome those issues, a Memristor Cellular Non-linear Network (M-CNN) was used to implement the digital realization of CNN by memristors. In this network, the process could balance the time and accuracy in the trade-off manner. For instance, several examples are based on this network for diverse applications such as compression of data; Fast Fourier Transform (FFT) computation, image processing algorithms and chaotic equation estimation or conversion of analog to digital circuits. Also, the consumption of energy, complexity of the circuit and overhead of area for a minimum number of bits were decreased. However, the time was necessary to perform the increasing operation with several bits [22].

The primary network unit, an oscillator circuit-based memristor, will be illustrated. To form the network, couple the oscillators. Then, the network was modified to compensate for the unbalanced connections from coupled node to coupled node and introduce the variability in the device-to-device process, and the original array would be

extended. To implement the reconfigured network, insert the transistors in series provided with the coupled capacitors that allowed switching on and off the connections between the cells. For example, organise the cell in the matrix configuration while selecting the respective control signals through multiplexers by ad-hoc address code. Meanwhile, reprogramming the connection among oscillators based on the electrical signals allowed the implementation of various computing phenomena [23].

The ant colony optimization (ACO) was designed through the memristor crossbar array. Initially, the non-linear voltage control memristor mechanism with the relaxation term and then the optimization process was performed with the padding strategy. Furthermore, the memristor crossbar array was designed with the external control circuits with the ACO strategy that provided high parallel computing and device density. The threshold for generating the edges was selected directly as the mean of the final conductance matrix for deploying the designing process of the memcomputing through the memristor crossbar array [24].

In-memory computing would eliminate the movement of data among the physical separable computing and memory units in traditional computers. The quadrative Eigenvalue problem was attained due to the transfer function of the circuit, whereas the minimum Eigenvalue value is denoted as the dominant pole that dominates the circuit's response time. Also, the size of the problem would affect the circuit response. The computational speed of the circuit was enhanced based on parameter setting and performs the process faster. To speed up the computation of the circuit, the above parameter was operated synergistically. For instance, this parameter was used to determine the computational error and power consumed in the circuit. This circuit was crucial to develop the in-memory machine learning (ML) accelerator applications for designing the memcomputing [25].

3. System model and problem statement

Memcomputing machines operate in continuous time, and the simulation process on modern computers is required to describe the discretization of time. The physical time was said to be continuous time; also, the discrete-time was not the physical quantity of the physical time. The UMM was said to be the Turing-complete that simulates the universal Turing machine (UTM). The UMM consists of analog and digital machines. Meanwhile, analog memcomputing machines had remarkable computation power, and analog systems could not plan for the scalability process, as the size of growing the machines required growing resources to obtain the same accuracy. The DMM was scalable and focused on the dissertation process. Moreover, if the functional operations are not optimised, then it has recorded high power consumption and area usage.

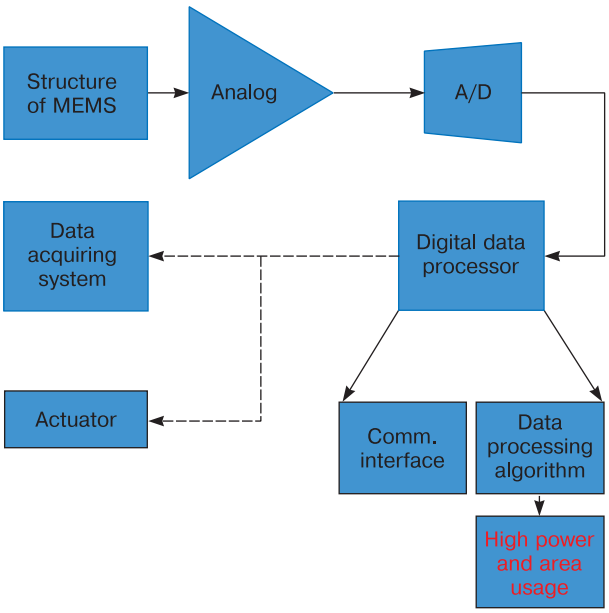

Figure 1. Difficulties of the traditional method

The difficulties of the traditional method are described in Fig. 1. Here, the algorithm for the data processing may result in high power and area usage, which leads to poor outcomes. Thus, the standard method for register-level transfer was very difficult. In addition, the methodology does not provide the outcome because of consuming high power and area usage. To overcome the difficulties of the traditional method, design the optimized MEM computing for the digital applications.

4. Proposed methodology

The first step was to make the idea into the chip in the ASIC design. The specification was provided with goals, constraints, functionality, speed and power, and technology constraints such as size and shape. The next step of this design was structural and functional description. The functionality should match the specifications. Here, a novel Buffalo-based register-transfer level design (BRTLD) is planned to be designed for controlling the ASIC design functional features. Here, the presence of the Buffalo best solution has helped to keep the MEMS architecture in optimal condition by controlling the data overflow. The proposed architecture is defined in Fig. 2.

The proposed methodology developed a novel buffalo-based register transfer level design method aimed at designing the optimized MEMCOMP for digital applications, which is related to the mems sensor's post-processing algorithm. Here, buffalo optimization gives the best solution in the proposed method.

4.1. Process of the proposed methodology

To achieve an effective combination of BRTL and the rapid design steps, the novel description language known

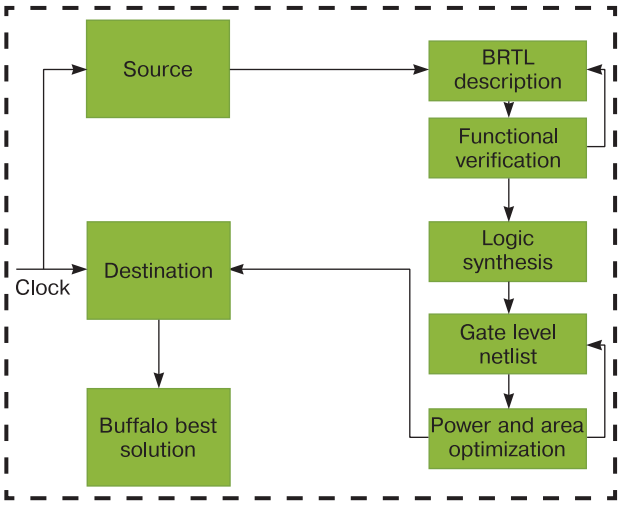

Figure 2. Proposed architecture

as AMDL (*Algorithmic Microarchitecture Description Language*) was developed along with the new pre-synthesis model. AMDL is the register transfer level (RTL) language that is tailored for describing data paths that include algorithmic constructs, which improves the design time and readability. The pre-synthesis algorithm transforms the AMDL model into the structural RTL model represented in the VHDL, which can be synthesized into the higher quality gate level description using actual RTL tools, as defined in Fig. 3.

A well-designed language-based approach can provide a high-level design method and general-purpose algorithmic language to increase the efficiency of hand-optimized BRTL designs. To ensure that high-level optimization is possible beyond the nature of the algorithm, it is important for the designer to have full visibility of the architecture. AMDL tasks contain correct value and left-value expressions that refer to the specific BRTL functional units and data storage elements, which are listed in the resource declaration section of the description.

$$D(x) = x_1, x_2, x_3, \ldots, x_n. \tag{1}$$

The digital data initialization for the design by using the Eq. (1). Where $D(x)$ is denoted as the objective function variable. The binary data was initialized. The AMDL model does not include the accurate functionalities of all operators. The AMDL designers understand the desired behaviour of the design and create an AMDL description that captures this behaviour accordingly. In most cases, an operator is only the library unit, but if its functionality is more complex, the VHDL code can be used to describe the design during the pre-synthesis phase.

Because the designer manually executes scheduling, binding and resource allocation, AMDL is not a programming language in itself but rather a way of writing code that follows algorithmic principles of the BRTL model that describes a higher abstraction level within BRTL. The conversion of AMDL to BRTL is straightforward during the pre-synthesis stage due to the AMDL language's detailed nature. Functional verification is important to a design's success. For the design, there are often billions or millions of potential test cases. The function verification can be done by using the Eq. (2).

$$F = D(x) + T_1(T_s - F_s) + T_2(T_s - F_s). \tag{2}$$

Where F is denoted as the functional verification variable, $D(x)$ is the objective function variables, T_s denoted as the true specification, F_s denoted as the false specification, T_1 and T_2 are test one and test 2. By using the equation, verify that the specific model implements the

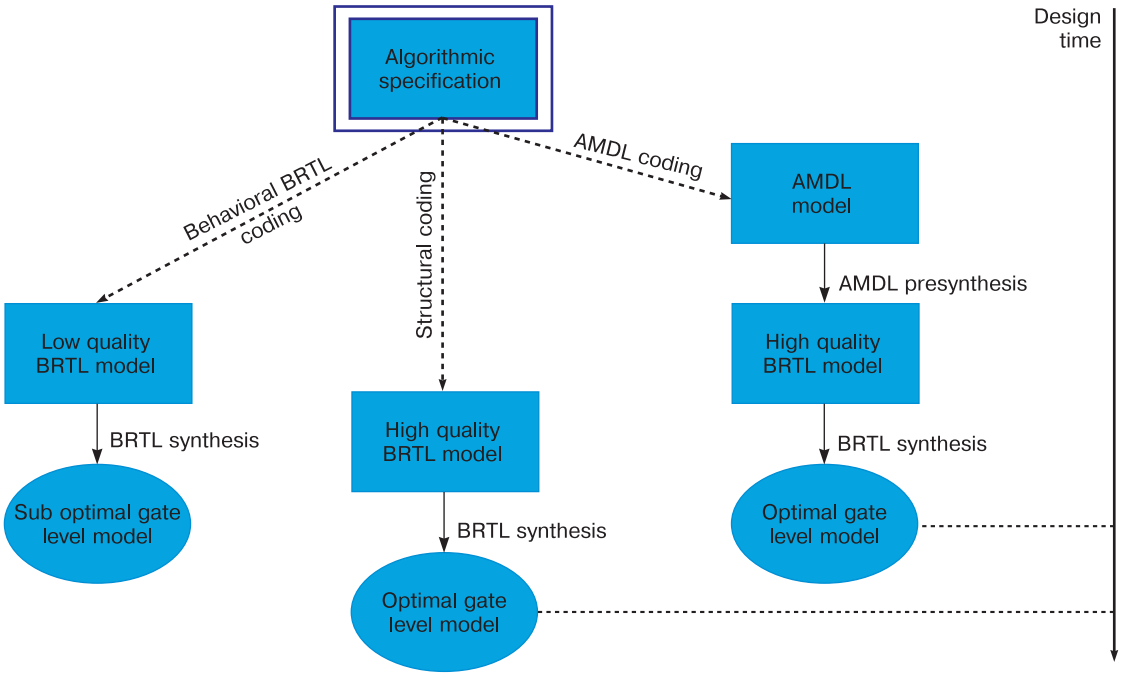

Figure 3. AMDL design objective

specification correctly. The datapath is created from the task statements in the AMDL model. The AMDL designer created the BRTL netlist by explicitly stating the connections between the logic elements and storage. The pre-synthesis algorithm identifies and resolves situations where multiple resources are competing for the same input by inserting multiplexers into the netlist as needed. After the BRTL netlist has been generated, the model generator algorithm creates HDL models of the datapath resource and combines them into a single VHDL entity and architecture. The datapath should be complemented by their VHDL model as well if the AMDL model includes a design-specific operator.

The control unit generation is based on the specific mapping regulation to determine the control states needed, and then the distinct AMDL control structures are implemented. AMDL provides additional control structures that improve the language's expression of concurrency beyond the fundamental elements of structured programming like decisions, loops, statements and sequences. The logic equivalence check can be done by using the Eq. (3).

$$Eq = F\left(\frac{T_1 - T_2}{r}\right). \quad (3)$$

Where Eq is the logic equivalence checking variable, T_1 is denoted as the true specification one, T_2 is the true specification two and r is the random value. A logic equivalence check is a very crucial process to perform after the synthesis process. This process checks the similarity between the original code and the synthesis netlist. To calculate the power by using the Eq. (4).

$$\text{Power} = P_s + P_C + P_l, \quad (4)$$

$$\text{if (power} \leq 0.5 = \text{optimal}). \quad (5)$$

Where P_s is denoted as the power switching, P_C is denoted as the power short circuit and P_l is denoted as the power leakage, by using the Eq. (5) to check the power optimization. A power value of less than 0.5 means optimal. Otherwise, move to gate level netlist and remove the data path using the buffalo optimization for power optimization.

$$\text{if (area} \leq 3.2 = \text{optimal}). \quad (6)$$

To check the optimization of the area by using the Eq. (6). Where 3.2 is the standard area usage value in BRTL. The area value less than 3.2 means optimal. Otherwise, move to the gate level and, reduce the gate size and minimize the logic using the buffalo optimization for area optimization. The control unit generation consists of two main measures: The pre-synthesis steps of the control state allocation process mapped AMDL control structures to VHDL implementation. In this step, the hierarchical structure of FSM fragments is built. The second step reduces the number of clock cycles by finding and using any opportunities where multiple control states can be executed at the same time. The generator of the BRTL model creates the entire implementation of FSM from the optimized tree of control states.

Algorithm 1: BRTL design

```
Start
{
    Initialization()
    {
        int D(x) = 1, 2, …, n;
        // initialize the digital data
    }
    Functional verification()
    {
        int F, Ts, Fs;
        // initialize the function verification variables
        Verify → D(x) – false specification
        // verify the correct specification
        if (specification = true)
        {
        Go to next step
        }else go to previous step
    }
    Logic equivalence check()
    {
        Power optimization()
        {
            Power → P_s + P_C + P_l
            // calculate the power
            if (power ≤ 0.5)
            {
            Optimal
            }else go to gate level netlist
            // optimization of power
        }
        Area optimization()
        {
            if (area ≤ 3.2%)
            {
            Optimal
            }else go to gate level
            // area optimization
        }
    }
}
Stop
```

The flow and step-by-step process of the novel buffalo-based register level transfer design for ASIC-based memcomputing can be described in Fig. 4. After the processing of the described steps, the experimental results of the novel BRTL design have been valued by different measures.

5. Result and discussion

The novel BRTL design is validated in Matlab and executed in the Windows 10 platform. The buffalo-based register transfer level design was implemented to control the ASIC design's functional features. The presented Buffalo best solution helps to keep the MEMs architecture in an optimal condition by controlling the data overflow. First, initialize the binary data.

The performance of the proposed model can be validated by power consumption, area, delay and memory usage. The specification of the execution parameters is given in Table 1.

Table 1. Specification of execution parameters

Descriptions of the parameters	
Programming environment	Matlab
Dataset	Digital data
Operating system	Windows 10
Optimization	Buffalo optimization

5.1. Case study

Developing a new ASIC based memcomputing is the new way of computing that uses memory as a computational resource; implement the BRTL in the memcomputing. Memristor is the type of memory that can compute and store information, making them ideal for memcomputing applications. The combination of ASIC and memristor was successfully implemented in the memcomputing application using BRTL design.

5.2. Algorithmic operation

The memcomputing has some arithmetic operations. The diagram shows how to perform arithmetic operations using a series of algorithms. The algorithms are stored in the memory of the memcomputing device, and they are

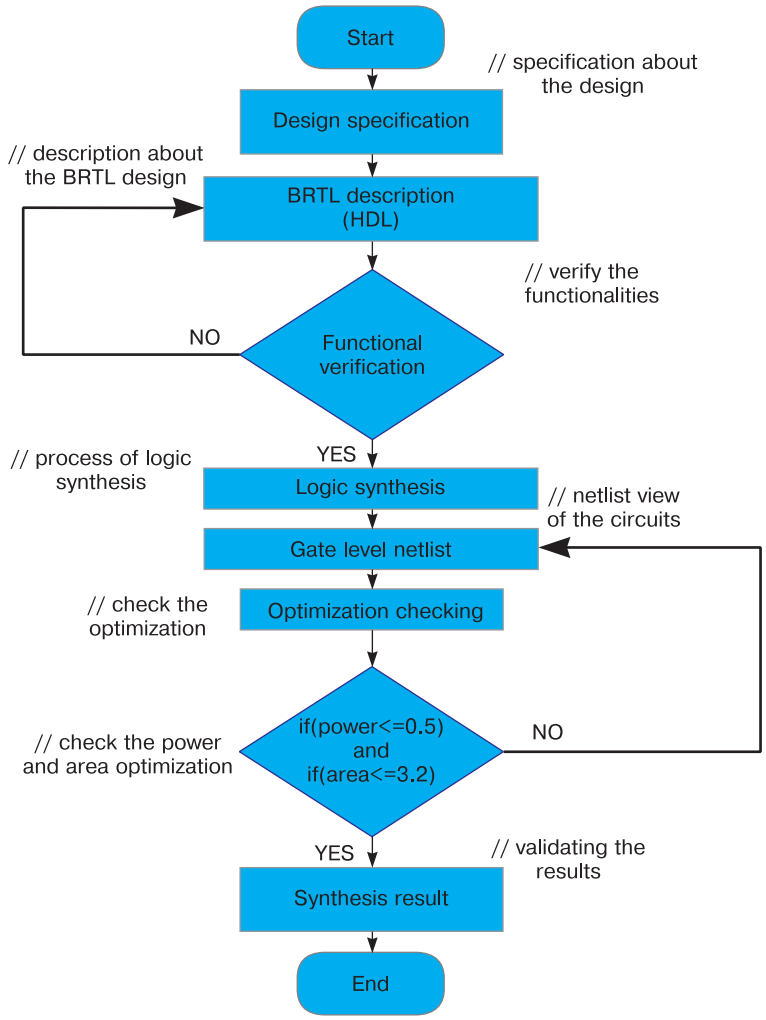

Figure 4. The flow diagram of BRTL

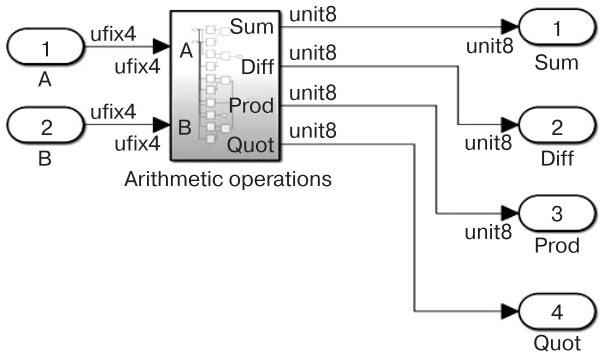

Figure 5. Algorithmic operations

executed one after the other to perform the desired operations.

The diagram shows how to perform arithmetic operations using a series of algorithms. The algorithms are stored in the memory of the memcomputing device, and they are executed one after the other to perform the desired operations. Figure 5 is a simplified representation of a memcomputing device, but it shows the basic principles of how memcomputing works. The memcomputing device has four registers: A, B, Sum and Diff. The registers are used to store the two integers being operated on the result of the operation and the difference between the two integers. The device also has four algorithms: Addition, Subtraction, Multiplication and Division. The first step is to load the two integers into registers A and B. The second step is to call the addition algorithm. The addition algorithm adds the two integers together and stores the result in the register sum. The third step is to display the result of the operation on the screen.

5.3. Arithmetic operations

The memcomputing device has components like a register, algorithm and control unit. In memcomputing, the algorithms are stored in the memory of the device and are executed one after the other. The advantage of using it to perform arithmetic operations is that it is much faster than traditional methods. This is because the algorithms for the desired operations can be stored in memory and executed very quickly. Additionally, memcomputing devices are very energy efficient, so they can be used to perform arithmetic operations on battery-powered devices.

Figure 6 shows the simplified representation of memcomputing devices that can be used to perform arithmetic operations on two 4-bit unsigned integers. Here, the registers are all 4 bits wide, which means that they can store integers from 0 to 15. The algorithms are very simple, and they can be executed very quickly. The control unit is responsible for sequencing the execution of the algorithms. It ensures the execution of the algorithms, making sure that they run in the correct order and that the output of each algorithm is used as input for the next. The diagram can perform any arithmetic operations, not just addition. The only difference is the algorithm called. For example, to perform subtraction, you would call the subtraction algorithm. To perform multiplication, you would call the multiplication algorithm. To perform division, you would call the division algorithm.

5.4. Performance analysis

The presented model is designed using the BRTL implemented on Matlab and runs on the Windows 10 platform. The metrics such as Area, power consumption, memory usage and delay are computed to validate the performance of the proposed method. For validating the performance improvement, take the recently associated model. The existing models such as Spiking Neural Network (SNN) [26], Hardware Optimized and Error Reduced Approximate Adder (HOERAA) [27], Binary Neural Network (BNN) [28], Residue Number System based Memory Loss Distributed Arithmetic (RNS-MLDA) [29].

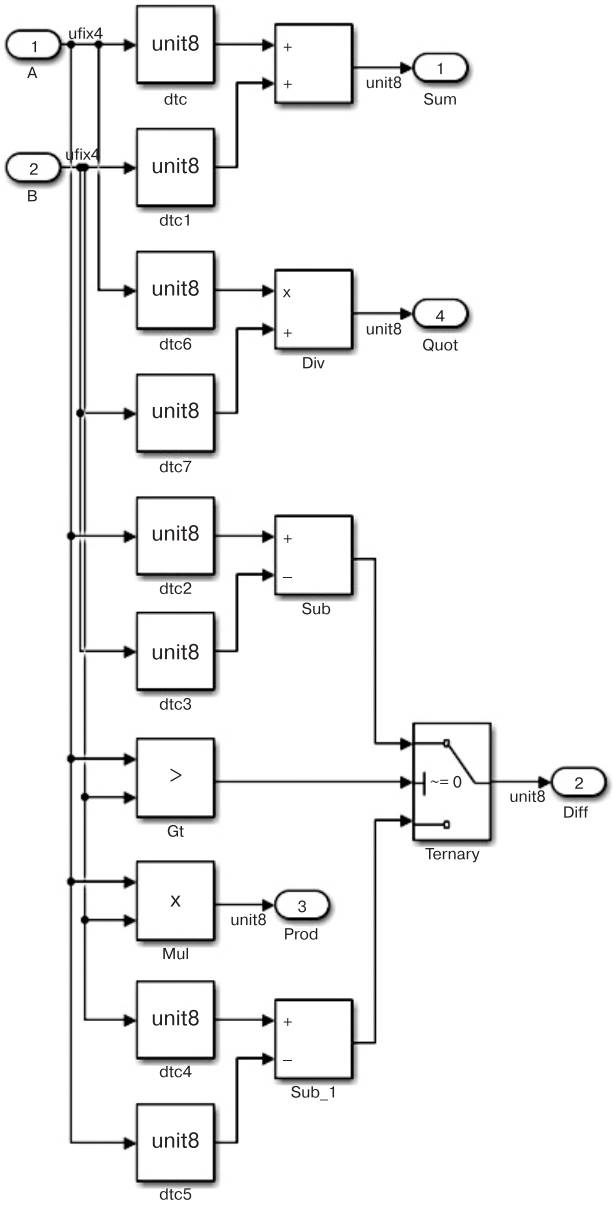

Figure 6. Arithmetic operation

5.5. Power consumption

The power consumption of an electronic device can be calculated as the power supplied to the device minus the power lost by the device. The amount of energy that the circuit uses while it is operating is known as power consumption. It is an essential factor in the design of electronic systems because it can have a significant impact on the overall performance of the system and the life of the battery.

The power consumption can be calculated by using the Eq. (4).

The power consumption of the existing model, such as SNN earned 46 µW, HOERAA earned 28.54 µW, BNN earned 59.75 µW, and RNS-MLDA earned 44.11 µW. The proposed model earned a power consumption of 24.1548 µW, which is less than the existing models. The power consumption comparison with the current model is presented in Fig. 7.

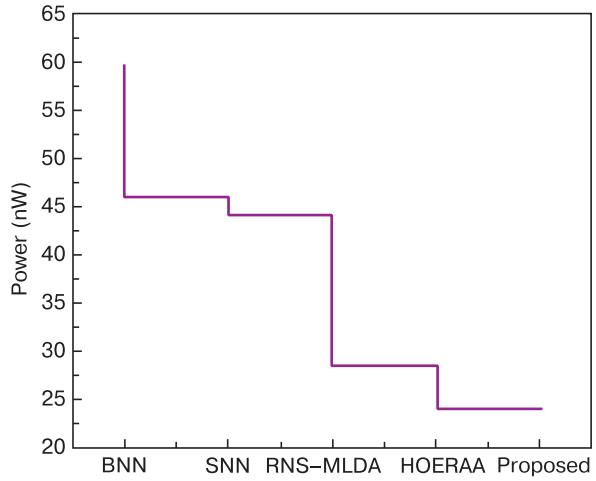

Figure 7. Power comparison with the existing model

5.6. Delay

Delay is the time that it takes for the signal to travel from one point to another in a circuit. It is an important factor in the electronic system design because it has a significant impact on the overall performance of the system.

$$\text{Delay} = D_t + D_w. \tag{7}$$

The delay can be calculated by using the Eq. (7). Where D_t is the transistor delay, which takes the time for a signal to travel through the transistor and D_w is the wire delay, which means it takes time for the signal to travel through a wire.

The delay of existing models such as the RNS-MLDA model earned 1.4 ns, the SNN model earned 0.98 ns, the HOERAA model earned 0.97 ns and the BNN model earned 0.97 ns. At the same time, the proposed model earned a delay of 0.812 ns. The comparison of delay with the existing model is presented in Fig. 8.

5.7. Area

The metrics area is the amount of space that the circuit occupies on a chip. It is an essential factor in the electronic system design because it could have a significant impact on the system cost.

The area usage of the existing model, such as the HOERAA method, earned 418.58 µm^2, the SNN method earned 128 µm^2, the RNS-MLDA method earned 54.32 µm^2 and the BNN method earned 23.18 µm^2. At the same time, the proposed model earned an area of 20.95 µm^2, which is lower than the existing methods. The comparison of the area with the existing method is presented in Fig. 9.

5.8. Memory

Memory is a circuit which can store the data permanently and temporarily. It is an important component in the electronic system design because it can store data for various purposes.

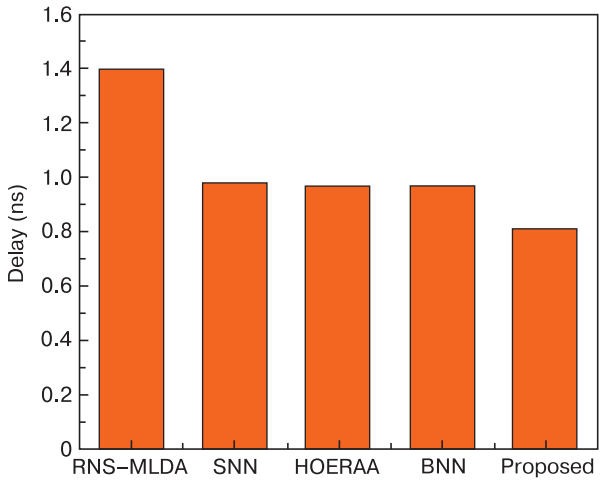

Figure 8. Delay comparison with the existing methods

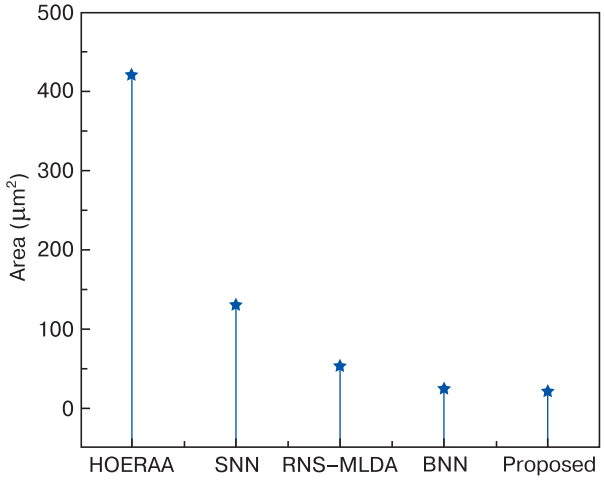

Figure 9. Area comparison with the existing methods

$$\text{Memory} = \frac{D_s N_a}{T_a}. \quad (8)$$

The memory usage can be calculated by the Eq. (8). Where the size of the data is $D_s N_a$ the number of times the memory will be accessed and T_a the time taken for accessing the memory.

The memory usage of the existing model, such as SNN, earned 533 MB. The proposed model earned memory as 259 MB, which is less than the existing method. The comparison of memory usage with the existing method is presented in Figure 10 and Table 2.

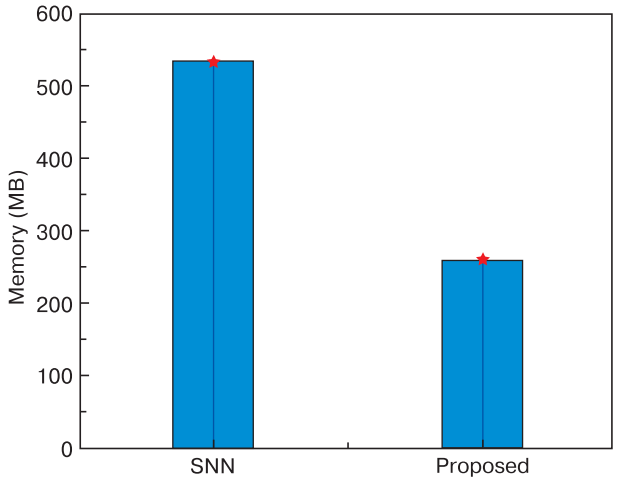

Figure 10. Memory comparison with the existing method

Table 2. Comparison assessment

Comparison statistics				
Methods	**Power (μW)**	**Delay (ns)**	**Area (μm²)**	**Memory (MB)**
SNN	46	0.98	128	583
HOERAA	28.54	0.97	418.58	–
BNN	59.75	0.97	23.18	–
RNS-MLDA	44.11	1.4	54.32	–
Proposed	24.1548	0.812	20.95	259

5.9. Discussion

Overall, the presented method has obtained better metrics scores in all metrics: Deviation time, area, power, frequency and cells. The overall performance of the proposed BRTL model has been tabulated in Table 3.

Table 3. Overall performance

Performance of BRTL	
Parameters	**Performance**
Lines of code	100
Delay (ns)	0.812
Area (μm²)	20.95
Cells	3983
Deviation time (s)	1.21
Power (μW)	24.1548
Frequency (MHz)	28.2
Memory (MB)	259

The overall performance of the proposed methodology is described in Table 3. The proposed model earned power consumption is 24.1548 μW. The power consumption is less when compared with the existing method.

6. Conclusion

High-quality digital data processing RTL models in MEMS sensors are typically hard to develop and keep up-to-date, as they lead to optimal gate-level implementations. The design of BRTL and optimization techniques that are effective in reducing area and lowering power consumption can significantly improve the effort and design time. The paper proposes a new way to model generate BRTL hardware using the AMDL language. The AMDL provides a high-level algorithmic interface for BRTL designers, which can lead to faster and more efficient designs. The proposed pre-synthesis steps for translating the AMDL model to BRTL have been shown to produce more quality output, which leads to efficient subsequent synthesis steps. Initially, the memcomputing system was designed with the needed functional processing elements. Then, a novel Buffalo-based register-transfer level design (BRTLD) is modelled in the MEMS architecture. The digital data transmission process is performed to estimate the reliability of the proposed BRTL model. The proposed model earned power as 24.1548 when compared with the traditional model, improved by 2%. Finally, the chief metrics like delay, memory usage and power/energy utilization were measured and compared with other models. In future, design the high-speed and area-efficient model with the proposed model.

References

1. Muratov D.G., Kozhitov L.V., Zaporotskova I.V., Popkova A.V., Tarala V.A., Korovin E.Yu., Zorin A.V. Synthesis, structure and electromagnetic properties of FeCoCu/C nanocomposites. *Modern Electronic Materials*. 2023; 9(1): 15–24. https://doi.org/10.3897/j.moem.9.1.104721

2. Abryutin V.N., Davydova E.V., Egorov M.A., Maronchuk I.I., Sanikovich D.D. Profound purification of tellurium, zinc and cadmium for electronic applications. *Modern Electronic Materials*. 2022; 8(1): 7–14. https://doi.org/10.3897/j.moem.8.1.89297

3. Kislyuk A.M., Ilina T.S., Kubasov I.V., Kiselev D.A., Temirov A.A., Turutin A.V., Shportenko A.S., Malinkovich M.D., Parkhomenko Yu.N. Degradation of the electrical conductivity of charged domain walls in reduced lithium niobate crystals. *Modern Electronic Materials*. 2022; 8(1): 15–22. https://doi.org/10.3897/j.moem.8.1.85251
4. Silva F., Sanz M., Seixas J., Solano E., Omar Y. Perceptrons from memristors. *Neural Networks*. 2020; 122: 273–278. https://doi.org/10.1016/j.neunet.2019.10.013
5. Romero-Zaliz R., Pérez E., Jiménez-Molinos F., Wenger C., Roldán J.B. Study of quantized hardware deep neural networks based on resistive switching devices, conventional versus convolutional approaches. *Electronics*. 2021; 10(3): 346. https://doi.org/10.3390/electronics10030346
6. Bae J., Kim G., Lee S.J. Real-time prediction of nuclear power plant parameter trends following operator actions. *Expert Systems with Applications*. 2021; 186: 115848. https://doi.org/10.1016/j.eswa.2021.115848
7. Sun K., Chen J., Yan X. The future of memristors: Materials engineering and neural networks. *Advanced Functional Materials*. 2021; 31(8): 2006773. https://doi.org/10.1002/adfm.202006773
8. Finocchio G., Di Ventra M., Camsari K.Y., Everschor-Sitte K., Amiri P.K., Zeng Z. The promise of spintronics for unconventional computing. *Journal of Magnetism and Magnetic Materials*. 2021; 521(Pt 1): 167506. https://doi.org/10.1016/j.jmmm.2020.167506
9. Bonnin M., Traversa F.L., Bonani F. Coupled oscillator networks for von Neumann and non-von Neumann computing. In: Virvou M., Tsihrintzis G.A., Tsoukalas L.H., Jain L.C. (eds.). *Advances in artificial intelligence-based technologies. learning and analytics in intelligent systems*. Springer: Cham; 2022. Vol. 22. P. 179–207. https://doi.org/10.1007/978-3-030-80571-5_11
10. Bohaichuk S., Kumar S. Computing with device dynamics. In: *Memristive devices for Brain-inspired computing*. Woodhead Publishing; 2020. P. 255–273. https://doi.org/10.1016/B978-0-08-102782-0.00010-1
11. Wang C.-Y., Wang C., Meng F., Wang P., Wang S., Liang Sh.-J., Miao F. 2D layered materials for memristive and neuromorphic applications. *Advanced Electronic Materials*. 2020; 6(2): 1901107. https://doi.org/10.1002/aelm.201901107
12. Joksas D., Almutairi A., Lee O., Cubukcu M., Lombardo A., Kurebayashi H., Kenyon A.J., Mehonic A. Memristive, spintronic, and 2D-materials-based devices to improve and complement computing hardware. *Advanced Intelligent Systems*. 2022; 4(8): 2200068. https://doi.org/10.1002/aisy.202200068
13. Elwy O., AbdelAty A.M., Said L.A., Madian A.H., Radwan A.G. Two implementations of fractional-order relaxation oscillators. *Analog Integrated Circuits and Signal Processing*. 2021; 106: 421–432. https://doi.org/10.1007/s10470-020-01640-x
14. Kashchenko A.A. Relaxation cycles in a model of two weakly coupled oscillators with sign-changing delayed feedback. *Theoretical and Mathematical Physics*. 2020; 202: 381–389. https://doi.org/10.1134/S0040577920030101
15. Yang K., Yang J.J., Huang R., Yang Y. Nonlinearity in memristors for neuromorphic dynamic systems. *Small Science*. 2022; 2(1): 2100049. https://doi.org/10.1002/smsc.202100049
16. Cheng C., Tiw P.J., Cai Y., Yan X., Yang Y., Huang R. In-memory computing with emerging nonvolatile memory devices. *Science China Information Sciences*. 2021; 64: 221402. https://doi.org/10.1007/s11432-021-3327-7
17. Maheshwari S., Stathopoulos S., Wang J., Serb A., Pan Y., Mifsud A., Leene L.B., Shen J., Papavassiliou C., Constandinou T.G., Prodromakis T. Design flow for hybrid CMOS/memristor systems. Pt I: Modeling and verification steps. *IEEE Transactions on Circuits and Systems I: Regular Papers*. 2021; 68(12): 4862–4875. https://doi.org/10.1109/TCSI.2021.3122343
18. Camsari K.Y., Debashis P., Ostwal V., Pervaiz A.Z., Shen T., Chen Z., Datta S., Appenzeller J. From charge to spin and spin to charge: Stochastic magnets for probabilistic switching. *Proceedings of the IEEE*. 2020; 108(8): 1322–1337. https://doi.org/10.1109/JPROC.2020.2966925
19. Fu H., Xu Y., Wu G., Liu J., Chen S., He X. Emphasis on the flipping variable: Towards effective local search for hard random satisfiability. *Information Sciences*. 2021; 566(5): 118–139. https://doi.org/10.1016/j.ins.2021.03.009
20. Liao M., Wang C., Sun Y., Lin H., Xu C. Memristor-based affective, associative memory neural network circuit with emotional gradual processes. *Neural Computing and Applications*. 2022; 34: 13667–13682. https://doi.org/10.1007/s00521-022-07170-z
21. Sirakoulis G.Ch., Ascoli A., Tetzlaff R., Yu S. Guest editorial memristive circuits and systems for Edge-computing applications. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*. 2022; 12(4): 717–722. https://doi.org/10.1109/JETCAS.2022.3226900
22. Camps O., Chawa A., Moner M., Stavrinides S.G., Picos R. Stochastic computing emulation of memristor cellular non-linear networks. *Micromachines*. 2022; 13(1): 67. https://doi.org/10.3390/mi13010067
23. Weiher M., Herzig M., Tetzlaff R., Ascoli A., Mikolajick T., Slesazeck S. Improved vertex colouring with NbO_x memristor-based oscillatory networks. *IEEE Transactions on Circuits and Systems I: Regular Papers*. 2021; 68(5): 2082–2095. https://doi.org/10.1109/TCSI.2021.3061973
24. Yu Y., Deng Q., Ren L., Tashi N. Memristor crossbar array based ACO for image edge detection. *Neural Processing Letters*. 2020; 51: 1891–1905. https://doi.org/10.1007/s11063-019-10179-6
25. Wang S., Sun Z., Liu Y., Bao S., Cai Y., Ielmini D., Huang R. Optimization schemes for in-memory linear regression circuit with memristor arrays. *IEEE Transactions on Circuits and Systems I: Regular Papers*. 2021; 68(12): 4900–4909. https://doi.org/10.1109/TCSI.2021.3122327
26. Lin J., Yuan J.-Sh. A scalable and reconfigurable in-memory architecture for a ternary deep spiking neural network with ReRAM-based neurons. *Neurocomputing*. 2020; 375: 102–112. https://doi.org/10.1016/j.neucom.2019.09.082
27. Balasubramanian P., Maskell D.L. Hardware optimized and error-reduced approximate adder. *Electronics*. 2019; 8(11): 1212. https://doi.org/10.3390/electronics8111212
28. Ankit W., Khatri S., Vrudhula S. A configurable BNN ASIC using a network of programmable threshold logic standard cells. In: *2020 IEEE 38th Inter. conf. on computer design (ICCD)*. Hartford, CT, USA; 2020. P. 433–440. https://doi.org/10.1109/ICCD50377.2020.00079
29. Jyothi G.N., Kishore S., Vijayalakshmi A. ASIC implementation of distributed arithmetic based FIR filter using RNS for high-speed DSP systems. *International Journal of Speech Technology*. 2020; 23(2): 259–264. https://doi.org/10.1007/s10772-020-09683-1