VTEAM Model Based In-Memory Computation using Memristors

Chinmay C, Mahantesh S Kenchannavar, Tejaswini A, Kariyappa B S

Abstract— The recent applications in the computing field deals with huge data processing operations, which involves a lot of data transfer from the memory to processing unit and vice versa. If these data transfers between the CPU and memory are reduced, a lot of time could be saved along with advantage of getting desirable outputs at a lower latency. One of the ways to achieve this is to adopt In-memory computational techniques, where most of the computations happen inside memory itself. Memristive crossbar architecture is a memory architecture which makes use of memristors as memory cells to store data with some added computational capabilities. The need for power efficient computing in the present days motivates the search for new nonvolatile universal memories like memristors. In this paper, we present an optimal way of implementing basic ALU operations in the memristive memory block. A detailed description for simulating this is given, which includes the creation of memristor symbol, choosing a suitable memristive logic to implement basic digital logics, creation of memory crossbar and the way to realize basic ALU operations.

Keywords— Memristor, IMPLY, MAGIC, memory crossbar, Memristive Processing Unit.

I. INTRODUCTION

Memristor is a two-terminal nonlinear, passive electrical component that links magnetic flux and electric charge. It remembers the charge that has flown through it and mainly gets its significance because of its non-volatile nature. It is often called stateful memory i.e. it remembers its resistance state until some exciting current or voltage is applied across it [1]. Due to this behavior, it finds its use cases in a lot of applications like neuromorphic computing, non-volatile memory blocks, logic circuits etc. High resistance state is considered as logic '0' and low resistance state considered as logic '1'.
The resistance of the memristor follows a hysteresis loop depending on the applied voltage and retains this resistive state until the next pulse.
As a memory block, its functionalities are different from other non-volatile memories because of different memristive logics which can be implemented in it.

Revised Manuscript Received on November 30, 2019.
***** Correspondence Author
Chinmay C*, Electronics and Communication Engineering, R V College of Engineering, Bengaluru, India. Email: chin9.ind@gmail.com
Mahantesh S Kenchannavar, Electronics and Communication Engineering, R V College of Engineering, Bengaluru, India. Email: Mahanteshs45@gmail.com
Tejaswini A, Electronics and Communication Engineering, R V College of Engineering, Bengaluru, India. Email: leotejaswini@gmail.com
Dr. Kariyappa B.S., Electronics and Communication Engineering, R V College of Engineering, Bengaluru, India. Email: kariyappabs@rvce.edu.in

Using these memristive logics, different logical gates can be implemented, where both the outputs and inputs are in the form of resistance exhibited by the memristors themselves [2]. Out of the different memristive logics, IMPLY and MAGIC (Memristor Aided loGIC) are the popular ones[3].
In this paper, the complete method of implementing ALU operations in the memristive memory is given, from the creation of memristor symbol to implementation of basic ALU operations in memory by applying different excitations at different cycles. The following steps are followed:

- As a first step, a suitable mathematical model and optimal device parameters for the memristor is chosen, in our case VTEAM model because of the reason mentioned in the next section.
- The above selected mathematical model is used to model a memristor symbol in cadence by using Verilog-A.
- Basic gates are implemented using memristors where the logic operations and storage both happen in the memristors themselves. To implement these gates, memristive logics such as IMPLY and MAGIC (Memristor Aided loGIC) are considered and is concluded that the MAGIC best suites our application i.e. in-memory computation.
- A memory crossbar is built using memristors and basic full adder functionality is tested by applying various control voltages at the rows and columns of the crossbar at different cycles to obtain the full adder output at the last cycle.
- Finally, the basic ALU operations in the memory crossbar is tested by applying different control voltages so as to minimize the number or operational cycles and the number of memristor blocks involved in it.

II. MEMRISTOR MODELLING AND MEMRISTOR LOGICS

There are various mathematical models available for modelling a memristor like, TEAM [12], VTEAM [13], Simmons Tunneling Model and Non-Linear Ion Drift Model etc. VTEAM model is selected to meet the objectives, as it is a voltage-controlled model and the I-V characteristics are ideally symmetrical. The parameters of the model were adjusted such that optimum hysteresis loop is obtained. Fig.1. shows the analysis of memristor model in MATLAB. The I-V characteristics of the memristor has a symmetric hysteresis loop after optimizing the parameters [13]. As seen from the figure, there are two cutoff voltages Von and Voff. For the memristor model shown, Von=3.5V and Voff=-2.5V. So, for the voltages greater than 3.5V, the memristor changes its state from Roff to Ron.

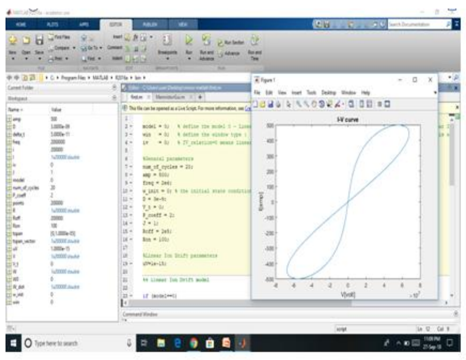

Fig.1. Analysis of Memristor Model in MATLAB.

For voltage less than -2.5V, memristor changes its state from Ron to Roff. And for the voltage between Von and Voff, the memristor state doesn't change [2]. Hence, previously if its state was Ron, then it will remain Ron itself, and if its state was Roff, then it will remain Roff only.Then a symbol in Cadence Virtuoso is created for the memristor from the Verilog-A code. Fig.2. shows the symbol of Memristor in the Virtuoso. It has 3 pins namely 'p','n' and 'w'. Out of which 'p' and 'n' are used to apply the voltage.

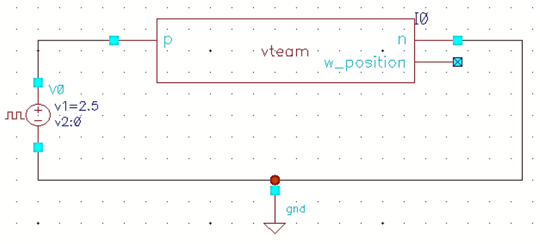

Fig.2. Memristor Symbol in Cadence.

There are various novel techniques to implement a logic within these MPUs, out of which the popular ones are IMPLY logic and Memristor Aided Logic (MAGIC). The performance and quality of these memristor logic depends mainly on two factors, the number of memristors and the number of operational pulses involved required to implement them. Fig. 3. shows the complete implementation of the IMPLY logic. The symbol of IMPLY gate is denoted as "→". It shows the schematic of the IMPLY gate. It has two memristors and a resistor. Two voltage cycles are applied, Vcond and Vset. Imply logic involves the presence of two memristors, which are working memristor and input memristor, which have the following functionalities [4]:

1) Input memristor: This memristor holds the input signal and maintains its resistance state even after computation.

2) Working memristor: This memristor holds the input signal, the operational result, or a constant 0 value of an IMPLY gate. The input value initially stored is erased after computation.

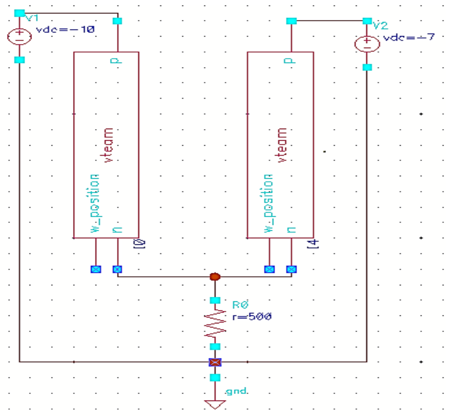

Fig. 3. Imply Logic.

Fig. 4. shows the implementation of MAGIC NOR gate [2] which is a fundamental block required to implement all other combinational logic expressions. It has 2 input memristors where the input logics are stored in them in the form of their resistance and 1 output memristor which holds the output logic in the form of its resistance.By changing the polarity of the applied voltage to the memristors and by arranging the input memristors in series, other Boolean logics like NAND, AND, OR also can be implemented using MAGIC.

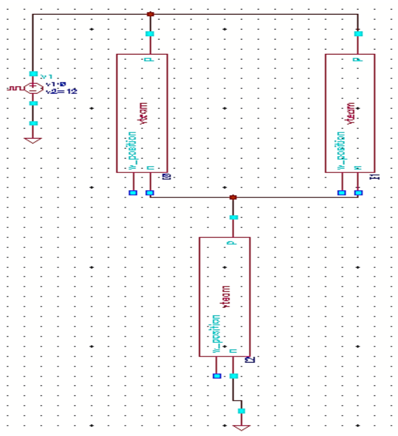

Fig. 4. Magic NOR.

Three operations required to executive MAGIC in a memristive memory are: row isolation, column isolation (only for transpose memory) and MAGIC execution [3]. The sneak path effect and non-ideal wires effect are not discussed in this paper.

Table.1. shows the comparison of IMPLY and MAGIC. As it can be seen from the table MAGIC has no fanout issues, requires less operational pulses and less complex when compared to IMPLY. Also, there is an explicit output memristor in MAGIC where as one of the input memristor acts as the output in case of IMPLY, which means, that input is destroyed after the computation [4]. Due to the above-mentioned advantages of MAGIC [6], we prefer it over the IMPLY logic in the design of combinational circuits.

Table. 1. Imply Logic vs MAGIC

IMPLY logic	MAGIC
It is slower compared to MAGIC.	It is 2.4% faster than IMPLY logic.
It requires more operational pulses.	It requires less operational pulses.
It has fanout issues.	No fanout issues.
Dissipates more energy compared to MAGIC.	Dissipates 66.3% less energy
Has high computational complexity.	Is less complex compared to imply logic.
Requires complicated control circuitry.	Requires simple control circuitry.

III. DESIGN ALGORITHM FOR FULL ADDER

The previous section describes how Boolean logics can be implemented using MAGIC. Thus, different combinational circuits can be implemented if their logic expressions are known. To implement a full adder in memristive memory crossbar, we consider the following two expressions for S (sum) and Cout (carry) which are outputs of a full adder:

Cout = ((A + B)' + (B + C)' + (C + A)')'…………………..(1)

S = [[(A' + B' + C')' + {(A + B + C)' + Cout}']']'………..(2)

Where, A, B and C are the inputs for the full adder which are stored in the respective memristors prior to the computational process. S and Cout are the sum and carry generated because of the computations and are stored in the output memristors till they are disturbed externally by applying triggering voltages to them. Both the S and Cout outputs are obtained solely by using NOR gates and each NOR operation takes one operational pulse. Equations (1) and (2) are used to get the full adder outputs from nor operations. This not only involves the input and output memristors but also other functional memristors which store the intermediate results of NOR operations in them. Fig. 5. Shows how the full adder is implemented in the memory crossbar [9],[14],[10]. Table 1 of supplementary material included in [4] shows the 16 operational pulses required to perform the full adder logic in a transpose memory architecture. The logic operation performed, the input and output memristors, the isolation voltages at every cycle is shown. Number of operational pulses and the additional functional memristorsinvolved in the operation completely depends on the design which is used. The best design has optimal usage of these two.

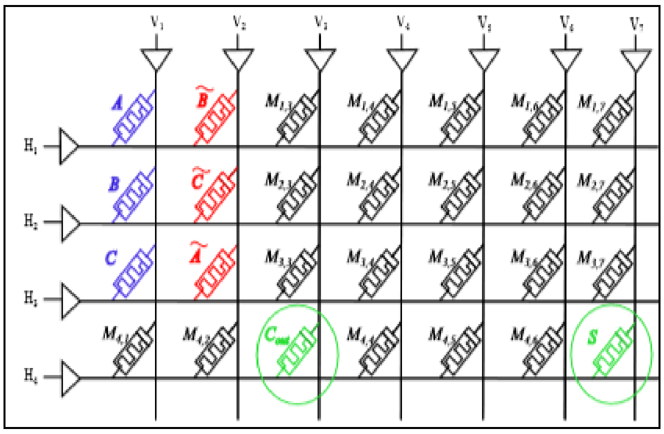

Fig.5. Implementation of Full Adder in Memristive Memory Crossbar. ~A, ~B, C~ are the Inverted Data for the A, B, C inputs and S and Cout are the Sum and Carry Generated. Others are Functional Memristors [4].

IV. ALU OPERATIONS INSIDE MEMRISTIVE MEMORY CROSSBAR

All the arithmetic and logical operations which could be realized in an ALU can be implemented by just giving suitable inputs A, B and Cin to full adder along with the relevant control inputs S0, S1, S2. Fig. 6. shows the Logic diagram of the complete ALU design.

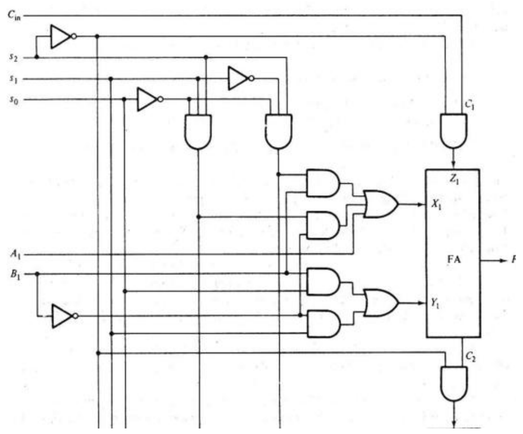

Fig. 6. Logic Diagram of ALU Block [7].

Table 2 shows all the 12 arithmetic and logical operations. The ALU block performs arithmetic operations when S2 bit is 0 and performs logical operations when S2 bit is 1. The entire ALU design can be split into 2steps. The first one being the full adder design and the second one is to implement the Xi, Yi and Zi inputs for the full adder.

Table 2. Arithmetic and Logical Operations Performed in ALU for Different Control Table 2. Arithmetic and Logical Operations Performed in ALU for Different Control Signals.

Selection				Output	Function	Location of Inputs in Memristor Crossbar
S2	S1	S0	C			
0	0	0	0	F=A	Transfer A	M73=A; M41=S2=0; M51=S1=0; M61=S0=0; M32=Cin=0
0	0	0	1	F=A+1	Increment A	M73=A; M41=S2=0; M51=S1=0; M61=S0=0; M32=Cin=0
0	0	1	0	F=A+B	Addition	M73=A; M22=B; M32=Cin=1; M41=S2=0; M51=S1=0; M61=S0=1
0	0	1	1	F=A+B+1	Add with carry	M73=A; M22=B; M32=Cin=1; M41=S2=0; M51=S1=0; M61=S0=1
0	1	0	0	F=A-B-1	Subtract with borrow	M73=A; M22=B; M32=Cin=0; M41=S2=0; M51=S1=1; M61=S0=0
0	1	0	1	F=A-B	Subtraction	M73=A; M22=B; M32=Cin=1; M41=S2=0; M51=S1=1; M61=S0=0
0	1	1	0	F=A-1	Decrement A	M73=A; M41=S2=0; M51=S1=1; M61=S0=1; M32=Cin=0
0	1	1	1	F=A	Transfer A	M73=A; M41=S2=0; M51=S1=1; M61=S0=1; M32=Cin=1
1	0	0	X	F=A OR B	OR	M73=A; M22=B; M41=S2=1; M51=S1=0; M61=S0=0
1	0	1	X	F=A XOR B	XOR	M73=A; M22=B; M41=S2=1; M51=S1=0; M61=S0=1
1	1	0	X	F=A AND B	AND	M73=A; M22=B; M41=S2=1; M51=S1=1; M61=S0=0
1	1	1	X	F=$\overline{A}$	Complement A	M73=A; M41=S2=1; M51=S1=1; M61=S0=1

During the execution, first all the inputs are written in the locations mentioned in the last column of Table 3, where Mxy represents the memristor at row x and column y. For example, to write S2=0 in the M42 memristor, apply V0 to fourth row and GND to second column, similarly to write S2=1, apply GND to fourth row and V0 to second column. The below equations show the Xi, Yi, Zi values in terms of the A, B, C inputs and the controls [6].

Xi = Ai + S2S1'S0'Bi + S2S1S0'Bi'………………………(3)

Yi = S0Bi + S1Bi'…………………………………………..(4)

Zi = S2'Ci…………………………………………………..(5)

These Xi, Yi, Zi, are to be computed inside the memory and to be fed to the already designed Full adder. This is implemented using transpose architecture with 9X4 size and taking 27 operational pulses. Fig. 7. shows the ALU schematic with 4*9 memory crossbar. Table 4. given in the appendix shows the 27 operational pulses and the crossbar functionality at each of them to work as a full adder.

Table 3. Operational Pulses for ALU Implementation in Memristor Memory.

Logic	Vo	GND	Vhs	Vvs

M21 = B'
M31 = C'
	V2	V1	H3-9	--

M42 = S2'
M42 = S1'
M62 = S0'
	V1	V2	H1-3, H7-9	---

M11 = S2'	H1	H4	---	V2-4
M12 = S0	H1	H6	---	V1, V3, V4

M13 = S0'
M23 = B'
	V2	V3	H3-9	---

M71 = M	H7	H1, H2, H5 H6	---	V2 V3 V4
M72 = N	H7	H1247	---	V1 V3 V4
M74 = X'	V1	V4	H1-6, H89	---
M44 = X	H4	H7	---	V123
M81 = Z	H8	H34	---	V234
M84 = Z'	V1	V4	H1-7, H9	---
M64 = Z	H6	H8	---	V123
M92 = Q	H9	H25	---	V134
M93 = P	H9	H12	---	V124
M94 = Y'	V23	V4	H1-9	---
M54 = Y	H5	H9	---	V123
M14 = (X'+Y'+Z')'	H1	H789	---	V123
M74,84,94 = Ron	V4	H789	---	---
M74 = (X+Y)'	H7	H45	V123	---
M84 = (Y+Z)'	H8	H56	V123	---
M94 = (X+Z)'	H9	H46	V123	---
M24 = Cout =((x+y)'+(y+z)' +(z+x)')	H2	H789	V123	---
M34 = (X+Y+Z)'	H3	H456	V123	---
M74,84,94 = Ron	V4	H789		---
M74 = (Cout + (X+Y+Z)')'	H7	H2H3	V123	---
M84 = S' = ((X'+Y'+Z')'+(X +Y)')'	H8	H17	V123	---
M94 = S =M84'	H9	H8	V123	---

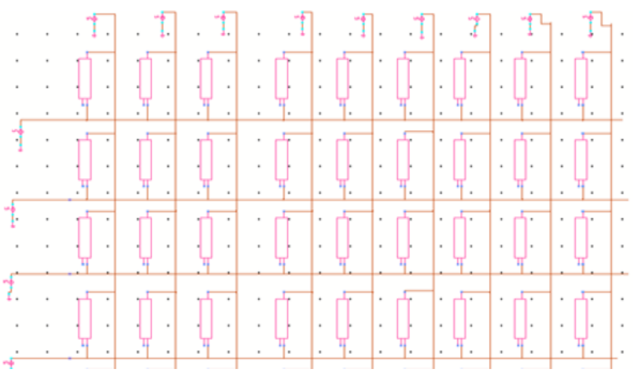

Fig. 7. ALU Schematic with 4*9 Memory Crossbar.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

We have made use of five voltage levels, which are, Vo, Gnd, Z state, Vhs and Vvs which are given as input to every row and column one at time. Four types of voltage values are defined which are: V0=5V, GND=0V, Vhs=1V, Vvs=4V.The designed ALU is of 1-bit which can be extended to 8 bits by instantiating the 1-bit ALU 8 times. For addition operation the value of s2, s1 and s0 are ‘0’, ‘0’ and ‘1’ respectively and Cin is ‘0’. For subtraction operation the value of s2, s1 and s0 are ‘0’, ‘1’ and ‘0’ respectively and Cin is ‘1’.

V. PERFORMANCE ANALYSIS

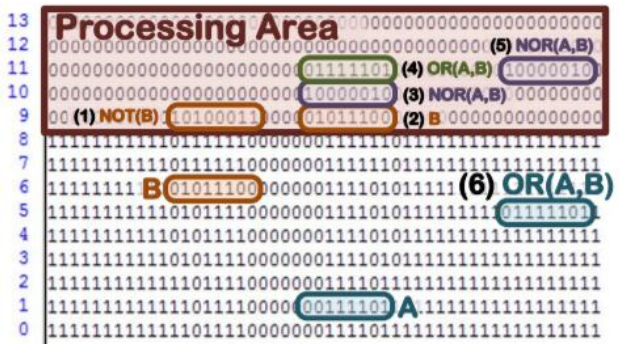

Fig. 8. Figure Depicting OR Operation on A and B (Two 8-Bit Vectors) in the Processing Area of the Memristor. The Result is then Written to the Destination Address. The operation is Realized using the Following Sequence of NOR Operations:

1. NOT(B).
2. NOT(NOT(B))- To Make B Align with A(Steps 1 and 2).
3. NOR(A, B).
4. NOT(NOR(A, B))- OR Result Between A and B (Steps 3 and 4).
5. NOR(OR(A, B))
6. NOT(NOR(A, B))- Copy Result to Destination Address.

Consider a simple CPU functionality of reading a data from memory, processing it and then writing back[5]. The performance analysis of the standard CPU with Von-Neumann architecture and CMOS design is made with the memristive unit. From Fig. 8.it is understood that a memristor based processing unit would take 6 clock cycles to implement the above 8-bit vector OR operation and storage of the result in a memory location. Whereas, to implement the same operation in a CMOS based processor, (8085 for e.g.), it gets implemented by following steps:

Table 4. 8-Bit Vector OR Implementation in 8085.

Processor Operation	No. of T-states
MVI A, 8-bit data	7
MVI B, 8-bit data	7
ORA B	4
STA 16 bit address	13
	Total= 31

Table 4. shows the OR operation of A and B (two 8-bit vectors) in traditional processing unit. The result is written to the destination address. The operation is performed by the above sequence of assembly statements.

Table 5. Performance Memristor Memory Processing Unit Vs Traditional Processing Unit.

Processing Unit	**Number of clock cycles**
Memristive Memory Processing Unit	6
Traditional Processing Unit	31
Improvement with memristor	5.167 times

From Table 5, its clear that memristor based processing units show approximately 5.167 times improvement in the performance based on the number of operational pulses taken for computation of results. CMOS based processors take more cycles because of the additional machine cycles and bus cycles used to access memory and load the contents to CPU. This performance analysis is not a general one and is demonstrated just for the above mentioned 8-bit OR operation. However, there will be some improvements in the performance when the memristive memory processing unit is considered for huge data-intensive operations because the large number of machine cycles involved in the data transfer operations between the processor and memory is eliminated as the computation itself happens in memory. But for processor-intensive applications, traditional processors based on von-Neuman architecture is suited best because the processing operations itself take many operational cycles in memristive memory unit [5]. Therefore, the best way would be to use the MPU as a kind of co-processor or an add-on to CPU where the data preprocessing happens.

VI. CONCLUSION

The paper dealt with the design and implementation of Non-volatile device called memristor and In-memory computation in the memristor crossbar array. First, the behavior and operation of the memristor was successfully understood in terms of its hysteresis loop. The two different logic styles of designing memristor circuits i.e. IMPLY logic and MAGIC where compared by the implementation of NOR Gate and concluded that MAGIC performed better than IMPLY in terms of delay and power requirements. Then all basic Gates such as NAND, AND, OR, NOR and NOT were successfully designed using MAGIC logic. Then memory crossbar architecture of memristor was designed successfully, wherein any location can be read or written with both logic values.
There are two types of realization of crossbar architecture, one is conventional memory crossbar and the other is transpose memory crossbar. In conventional we can perform operations either in horizontal or vertical. Full adder implementation in the crossbar was designed using both type of architectures.

Then 1-bit ALU was designed within the memory crossbar architecture to perform the basic ALU operations. The in-memory computation MPU faster when compared with the traditional Von-Neumann architecture in case of data-intensive operations and von-Neuman architecture is faster for processor-intensive operations. Hence, the best way to use the MPU would be to use it as a kind of co-processor where data-preprocessing happens. In future, applications related to image processing and neuromorphic computation can be efficiently executed within the memristor crossbar. Complete MPU can be realized as a co-processor to work along with the main processor.

REFERENCES

1. L. O. Chua, "Memristor-the missing circuit element," IEEE Trans. Circuit Theory, vol. CT-18, no. 5, pp. 507–519, Sep. 1971.
2. S. Kvatinsky et al., "MAGIC—Memristor-aided logic," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 61, no. 11, pp. 895–899, Nov. 2014
3. S. Kvatinsky et al., "Logic Design Within Memristive Memories Using Memristor-Aided loGIC (MAGIC)" IEEE Transactions on Nanotechnology, Vol 15, no/ 4, pp 635-650, July 2016.
4. Hsin-Pei Wang , Chia-Chun Lin, Chia-Cheng Wu , Yung-Chih Chen , and Chun-Yao Wang, "On Synthesizing Memristor-Based Logic Circuits With Minimal Operational Pulses", IEEE Transactions on Very Large Scale Integration Systems, Volume: Pp, Issue: 99,2018.
5. Rotem Ben Hur , Shahar Kvatinsky , "Memristive Memory Processing Unit (MPU) Controller for In-Memory Processing", ISCEE International Conference on the Science of Electrical Engineering, 2016.
6. Shahar Kvantisky, Dmitry Belousov, Slavik Liman, Guy Satat, Nimrod Wald, Eby G. Fredman, AvinoamKolodny, and Uri C. Weiser, "Magic-Memristor Aided Logic", IEEE Transactions On Circuits And Systems-II, 11 Nov 2014.
7. M Moris Mano, Digital Logic and Computer Design, 2016 Pearson India Education Services Pvt. Ltd .
8. L.ChuaandS.M.Kang, "Memristive devices and systems," Proc. IEEE, vol. 64, pp. 209–223, Feb 1976.
9. S. Paul and S. Bhunia, "A scalable memory-based reconfigurable computing framework for nanoscale crossbar," IEEE Trans. Nanotechnol., vol. 11, pp. 451–462, May 2012.
10. E. Linn et al., "Beyond von Neumann–logic operations in passive crossbar arrays alongside memory operations," Nanotechnology, vol. 23, pp. 305205:1–6, July 2012.
11. S. Kvatinsky et al., "Memristor-based material implication (IMPLY) logic: Design principles and methodologies," IEEE Trans. Very Large Scale Integr. Syst., vol. 22, no. 10, pp. 2054–2066, Oct. 2014.
12. S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, "TEAM: ThrEshold adaptive memristor model," IEEE Trans. Circuits Syst. I: Reg. Papers, vol. 60, no. 1, pp. 211–221, Jan. 2013.
13. S. Kvatinsky, M. Ramadan, E. Friedman, and A. Kolodny, "VTEAM: A general model for voltage-controlled memristors," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 62, no. 8, pp. 786–790, Aug. 2015.
14. Teimoori, M., Amirsoleimani, A., Ahmadi, A., & Ahmadi, M. (2018). A 2M1M Crossbar Architecture: Memory. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 26(12), 2608–2618. doi:10.1109/tvlsi.2018.2799951.
15. Naman S Kumar, Sudhanva N G, Shreyas Hande V, Mallikarjun V Sajjan, Hemanth Kumar C S, and Kariyappa B S, "SRAM design using Memristor and Self-controllable Voltage (SVL) Technique" International Conference on Computational Intelligence & Data Engineering (ICCIDE 2017) Vijayawada, Andhra Pradesh, 978-981-10-6318-3, July 2017, Springer.

AUTHORS PROFILE

I am Chinmay C, an Electronics and Communication Engineering student at R V College of Engineering (2015-2019), Bengaluru. I have completed my 10th Board from Kendriya Vidyalaya Dharwad, Karnataka and Pre-University College degree from Smt. Vidya P Hanchinmani Independent PU Science College Dharwad, KarnatakaI have presented "Testing the Performance of 7280 Controller Testbed" paper in National conference on Communication Standards and Systems and "Parametrics for the choice of Embedded Systems and Control Algorithms for Application Specific Robot Designs" paper in 2nd International Conference on Inventive Systems and Control 2018. I have worked on Distributed Robotics - Multi Swarm Robots at IIT Bombay. I have worked on embedded systems with Atmega chipsets. I have secured 11th Rank in Engineering field of Karnataka Common Entrance Test. I secured Certificate of merit for highest aggregate marks in 10th board. Won runner up place in National Level Smart India Hackathon out of 480 participants in 80 teams. Won 4th place in National Level E Yantra Robotics Competition out of 22,608 Students in 5,652 teams.

I am Mahantesh S Kenchannavar, an Electronics and Communication Engineering student at R V College of Engineering, Bengaluru (2015-2019). I have completed my 10th standard from Kendriya Vidyalaya Dharwad and Pre-University College degree from JSS College, Dharwad. I have designed a system to count the number of straws using image processing implemented on Raspberry Pi, Prediction of Diabetes Using Machine Learning, Smart Kitchen, Speech Recognition, Wireless Home Security System and ALU design using Constant Delay Logic. I have published the paper, "Wearable assistive device for the deaf" in IEEE 2018 : International Conference on Innovations in Engineering, Technology and Sciences (ICIETS – 2018).

I have won the Academic Excellence Award for I and III year Engineering, All India Rank 1 in 3rd year category in the Indian Engineering Olympiad, Won IEEE AAC Hackathon for the project Smart Watch For Deaf People, and selected for best project in 2nd International Conference on Consumer Electronics Asia. Currently working as Digital design engineer in Analog Devices.

I am Tejaswini A, an Electronics and Communication Engineering student at R V College of Engineering, Bengaluru (2015-2019). I have completed my 10th standard from Prarthana Central school, Bengaluru and Pre-University College degree from Sri Bhagwan Mahaveer Jain College, Bengaluru. I have worked on building adapter board for Firebird V robot based on AVR microcontroller, worked on speech recognition system. I was a part of winning team in robotics competition held by NITK, finalist of e-Yantra robotics competition, Runner up of Smart India Hackathon for the project on smart glove which helps the dumb in communicating with the external world. I was also a part of Astra Robotics, a student club while pursuing Engineering degree. Also, I have completed CCNA certifications in routing, switching and data center. Currently working as Application Engineer in Analog Devices.

I am Dr. Kariyappa B.S, Professor at Electronics and Communication Engineering, Rashtreeya Vidyalaya College of Engineering, Bangalore, Karnataka. I have completed Bachelors in Engineering and Masters in Engineering in Electronics and Communication and Ph.D in Digital Power Controller. I have been in the teaching field for 15 years 6 months. I am interested in VLSI and Embedded systems. I have guided 18 undergraduate projects and 12 post graduate projects. I have publications in 13 journals.

Recent ones are "RideNN: A New Rider Optimization Algorithm Based Neural Network for Fault Diagnosis in Analog Circuits", IEEE Transactions on Instrumentation & Measurement, Vol. 68, Issue.1, Janaury 2019, ISSN: 0018-9456 and "Low Power Logically Reduced TSPC Flipflop Design using Dual Threshold CMOS Technology" Journal of Emerging Technologies and Innovative Research (JETIR), Volume 5, Issue 5, May 2018, ISSN: 2349-5162. I have presented in 11 International conferences and 4 National conferences. Recent ones are "Digital Twin Ranorex Test Automation of SIPROTEC 5 Protection Devices", 3rd International Conference on Electronics and Communication and Aerospace Technology, Coimbatore, IEEE, June 2019, ISBN: 978-1-7281-0167-5 and "Optimization of Runtime and Memory Footprint in SOC Verification", International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT-2019), Bengaluru, IEEE, May 2019, ISBN: 978-1-7281-0630-4. I have worked on an R & D project named A Study of defining maneuring parameters of ship model by analyzing wakes using image processing techniques. I was awarded Awarded Certificate for "academic excellence through research publications" from RSST, in the year 2018, 2010 and 2009. I was Honored BHARAT VIKAS AWARD by Institute of Self Reliance (ISR) Bhubaneswar, Odisha for the occasion of "CITIZEN's Day on 19th November 2017. I was Awarded Certificate in recognition for "Obtaining Ph.D. Degree" from ISTE-RVCE chapter, in the year 2012.