RESEARCH ARTICLE OPEN ACCESS

Imply Logic Implementation of Carry Save Adder Using Memristors

*[1]A.Lavanya, [2]BG.Gopal,
[1]PG Scholar, [2,] Associate Professor,
Vel Tech Multi Tech Dr. Rangarajan Dr. SakunthalaEngineering College, Avadi, Chennai, India

Abstract-
Memristor is a two terminal fundamental passive element that provides arelation between electric charge'q' and magnetic flux 'Φ'. It is a type of resistor with some memory property. Implication logic can be easily developed using memristors. Implication logic can be implemented with less number of memristors in comparison with silicon transistors. It has been proved that any Boolean expressions can be implemented with this logic [3]and[4]. Carry save adder (CSA) circuits is built up with IMPLY logic with the basic gates obtained with memristors and their working are briefly explained. The total number of memristors required iscompared with the number of transistors required in constructing the same circuit with CMOS logic.

Keywords-- Memristor, flux, charge, memory, IMPLY logic, nonlinear dopant drift, logic gates.

I. INTRODUCTION

Memristor is a two terminal fundamental passive element that line up with basic electrical and electronic elements. It is passive because it just drops the inputs applied and not goingtoimprove thesignalit. The fundamental elements known before inventing memristor are Resistor, Capacitor and Inductor. All of them follow Ohm's law in its own way. Resistor provides relation between voltage and current; Capacitor provides relation between charge and voltage; Inductor provides relation between flux and the current. And the missing relation is between charge and the flux which is provided by memristor and hence it is also said to be a fundamental element [1]. The resistance of the element is varied with the applied potential and its polarity. It is the property of the memristor that once the applied potential is removed the resistance state is stored and for the next applied potential the memristor starts to work from the last stored resistance. This is observed from the hysteresis curve obtained on application of potential.

II. MEMRISTOR STRUCTURE AND CHARACTERISTICS

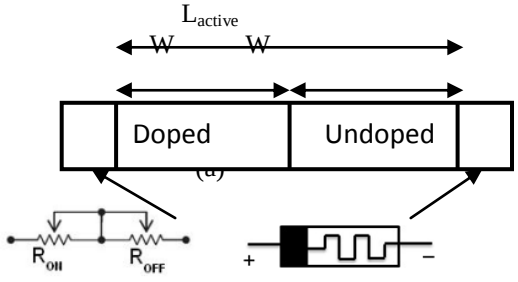
The structure for memristor proposed by HP is shown in the Fig. 1(a). It consists of a sandwiched structure of doped and undoped Titanium Dioxide (TiO_2) that is placed between Platinum electrodes. Here TiO_2 is an intrinsic (pure) semiconductor which has high resistivity contributes the undoped region. The doped area is formed by same TiO_2 that is doped by making oxygen vacanciesV_oS [12]. 'L_{active}' is the total length of the device and the lengths of the doped and undoped area are equal specified as 'w'. Also the doped and undoped areas are considered to be positivepoleand negative pole respectively. Depending on the time for which the type of charge passing through the device, the lengths of the doped and undoped area varies from 'w'. The movement of oxygen atoms follows the mechanism of dopant drift. [6] i.e., Oxygen vacancies in the TiO_2 prefer to move by rearranging its position on application of potential.This variation in length of the doped region is specified by 'l_{doped}'. The boundary line 'x' between the doped and undoped region shall vary. The variation in the boundary is provided by the state equation. $\frac{dw}{dt} = k.i(t).f(x)$ where, $k = \mu_v \frac{R_{ON}}{L^2_{active}}$

Here 'k' gives the relation between the mobility$\mu_v$of ions, R_{ON}, and L_{active}.

The memristance 'M' of the memristor can be expressed by the relation

$$M = R_{doped}\left(\frac{l_{doped}}{L_{active}}\right) + R_{undoped}\left(1 - \frac{l_{doped}}{L_{active}}\right)$$

For the given applied voltage, according to the M, we can obtain the associated current.



(b) (c)

Fig 1: a) Structure for memristor b) Equivalent circuit for memristor c) Symbol

The equivalent circuit of the memristor consists of two seriesresistances namely R_{ON} andR_{OFF}. Both

these resistances are the values just before the application of the voltage. On applying the voltage, the instantaneous values of resistances are termed as R_{doped} and $R_{undoped}$. The R_{doped}may attain maximum of $R_{OFF}$and$R_{undoped}$to the minimum ofR_{ON}.R_{ON} is the minimum resistance of the element and R_{OFF} is the maximum resistance of the element. For the applied positive potential to the positive terminal of the memristor, the oxygen vacancies tend to move towards the undoped area from the doped area. Thus the doped area width $l_{doped}$is increased thereby reducing the resistance R_{OFF}.

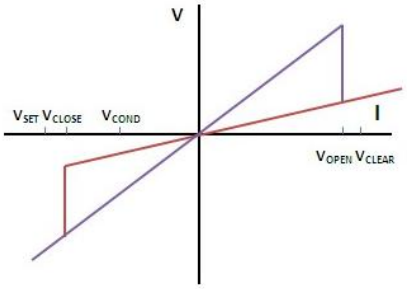
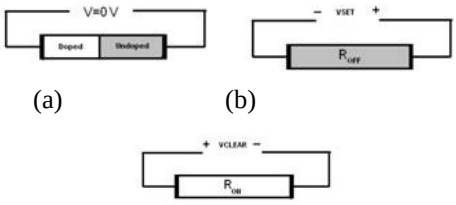


Fig 2: IV characteristics of memristor

This can be explained with the IV characteristic curve of Figure 2. The IV curve of the memristor has two linear resistances R_{ON} and R_{OFF} characterized by the slope of the two straight linesmentioned by the violet and red curves respectively. For the positive voltage the circuit exhibits a low resistance R_{ON}.Once after reaching a particular threshold value, the resistance toggles from the low resistance state R_{ON} to the high resistance state $R_{OFF}$and vice versa.



(a) (b)

(c)

Fig3:Operation of memristora)When no potential is applied b) For V_{SET} potential total resistance is R_{OFF}c)For V_{CLEAR} total resistance is R_{ON}

III. IMPLY LOGIC WITH MEMRISTORS IMPLICATION LOGIC

IMPLY logic is the digital logic. It is defined by the theorem

"If 'P' is TRUE then output follows 'Q' otherwise it is TRUE"

It stands for "***P*** then ***Q"*** and is denoted by p -> q. By classical logic we say IMPLY gate function is equivalent to ***¬P V Q***((**NOT** of **P**) **ORQ**).IMPLY logic is said to be functionally complete with which any logic functions can be implemented as shown in the figure.

Truth Table

P	Q	P→Q	¬PVQ
0	**0**	**1**	**1**
0	**1**	**1**	**1**
1	**0**	**0**	**0**
1	**1**	**1**	**1**

Table. 1 Truth Table for IMPLY logic

LOGIC IMPLEMENTATION

The implementation of IMPLY logic with memristors include two different sets of memristors namely input and work memristors. Input memristors also works as work memristors as and when required. R_g is the resistor that provides a voltage divider setup which aids in changing the states of the output memristors. The resistance of the memristor gives the logic values of '1' and '0'. If the resistance of the memristor is low i.e., R_{ON} then the logic value is considered to be '1' and if the resistance is high i.e., R_{OFF} then the logic value is considered to be '0'.

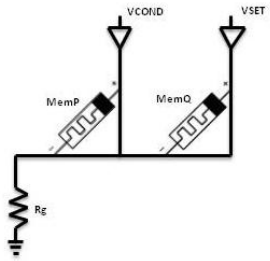


Fig. 3:Schematic of IMPLY gate with memristor

The working of this logic starts with setting the initial states of the memristors.The initial states of the memristors are the inputs to the circuit and the final states of the memristors are the output of the circuit which is being taken at the appropriate memristor. V_{COND} and V_{SET} are the two voltages applied to memristors to change the state and obtain the output state of the output memristor. V_{COND} is the voltage which is sufficient enough to maintain the state of the memristor but insufficient to shift the resistance. V_{SET} is the voltage which is sufficient enough to shift the resistance of the memristor. This is provided to the work memristor or the output memristor of the circuit which requires a shift of resistance with the input.

Input '00' -- Initially both memristors MemP and MemQ are set to initial state of 0. V_{COND} is applied to the input memristor memP and V_{SET} is applied to the work (output) memristor. V_{COND} which is insufficient to make the shift of resistance leaves memP in OFF state (0) and the voltage at the common node is ≈ 0V.

V_{SET} which is sufficient to make the shift of resistance makes memQ reach its ON state.

Input '01' -- MemP and MemQ are in 0 and 1 initial states respectively. V_{COND} again leaves memP in OFF state (0) and the voltage at the common node is ≈ 0V. Here MemQ is in ON state and hence MemQ remains in ON state for applied VSET.

Input '10' -- MemP andMemQ are in 1 and 0 initial states respectively. Now, VCOND is the voltage at the common node. MemQ being in OFF state experiences a potential difference of VSET-VCOND which just leaves MemQ in OFF state.

Input '11' -- MemP and MemQ are in initial states of 1 respectively. Now, VCOND is the potential at the common node. VSET –VCOND which is insufficient to make the state change leaves MemQ in ON state (Logic '1').

IV. BASIC LOGIC GATES USING MEMRISTORS

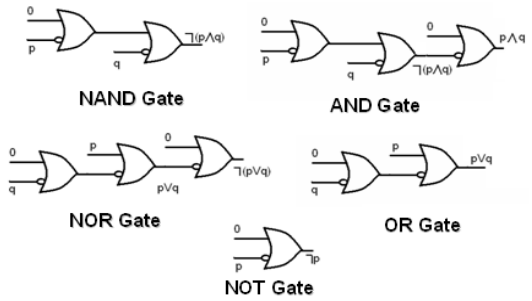


Fig. 4: Basic logic gates framed with IMPLY logic

NOT gate with memristor

A basic IMPLY gate with 2 memristors can be put to use as a NOT gate as shown in the Fig. 5 Here, MemQ is fixed to be in OFF state initially. Input is given to MemP. When memP is in OFF state and the potential at the common node is of ≈ 0V. And VSET applied to MemQ changes the resistance from OFF to ON state. i.e., for an input of 0, the output is 1. Now for an input of 1 (ON state), the potential at the common node is VCOND. Again VSET-VCOND which is insufficient to make the state change leaves behind MemQ in OFF state. i.e., for an input of 1, the output is 0. Hence the NOT gate is implemented using IMPLY logic with memristors.

The IMPLY expression for NOT gate is given by

$$NOT\ p\ = p \rightarrow {}'0'$$

The computational sequence is

$$MemP \rightarrow MemOUT$$

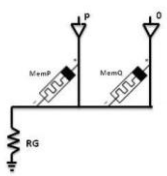


Fig. 5:IMPLY logic gate schematic

NAND gate with memristors

The implementation of a 2 input NAND gate requires about 3 memristors MemP, MemQ and MemR as shown in the Fig. 6 Here, MemP and MemQ are input memristors and MemR acts as a work (output) memristor. Initially both memristors MemP and MemQ are provided with inputs by setting the initial states. MemP is now given with VCOND and MemR is given with VSET which implies the first input and sets the initial state of OUT. VCOND is then applied to MemQ and VSET to OUT. The final state of the memristor MemR is the output of the NAND gate with memristors.

Similarly all the other gates can be implemented using memristors as shown in theFig. 6

The IMPLY expressions of NAND is given by

$$p\ NAND\ q = p \rightarrow (q \rightarrow {}'0')$$

The sequence of computation for NAND is given by

$$p \rightarrow r; q \rightarrow r$$

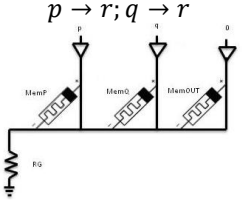


Fig. 6: NAND schematic with IMPLY logic

XOR gate with memristor

The IMPLY expression for XOR gate can be written as this

$$p\ XOR\ q = (p \rightarrow q) \rightarrow ((q \rightarrow p) \rightarrow' 0')$$

The implementation of XOR gate requires about 7 memristors as shown in Fig. 7 without making use of additional reset in between operation. First two are the input memristors. Next four are work memristors and the last one can be called as the output memristor since output is taken from it. Work and output memristors are initialized to '0'. The operations $p \rightarrow q$ requireupdatingMemQ whose initial state is required for the next operation. Hence the duplication of the inputs p and q are required. It uses about 4 memristors. The duplication is done over MemT1 and MemT2 respectively.

The sequence of computation for XOR gate is given by

$$MemP \rightarrow MemW1, \quad MemW1 \rightarrow MemT1,$$
$$MemQ \rightarrow MemW2, \quad MemW2 \rightarrow MemT2,$$
$$MemP \rightarrow MemT2, \quad MemQ \rightarrow MemT1,$$
$$MemT1 \rightarrow MemOUT, \quad MemT2 \rightarrow MemOUT.$$

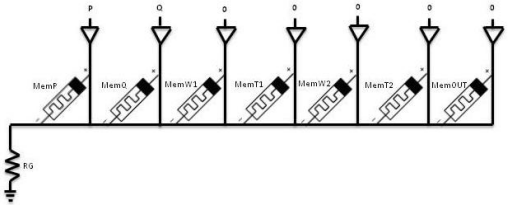


Fig. 7: XOR gate schematic with IMPLY logic

Thus the XOR operation requires 7 memristors and 9 computational steps including initialization of states of all the memristors.

AND gate with memristor

AND gate requires four memristors where the first two are input memristors, one is of work memristor and the last one is the output memristor.

The IMPLY expression for AND gate is given by

$$p\ AND\ q = (p \rightarrow (q \rightarrow 0)) \rightarrow 0$$

The computation sequence is given by

$$MemQ \rightarrow MemT, \quad MemP \rightarrow MemT,$$
$$MemT \rightarrow MemOUT.$$

OR gate with memristor

OR gate requires three memristors. Two input memristors and the other one is the output memristor.

The IMPLY operation for OR gate is given by

$$p\ OR\ q = (p \rightarrow '0') \rightarrow q$$

The computational sequence is given by

$$MemP \rightarrow MemOUT, \quad MemOUT \rightarrow MemQ.$$

V CARRY SAVE ADDER

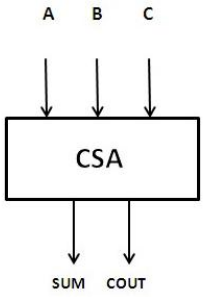
Carry save adder is the three bit adder circuit.The symbol for which is shown in the fig. 8 (a).The carry save adder is the high speed adder that involves parallelism in adding 3 consecutive bits of three 3-bit numbersIt includes two stages one will be a parallel stage of three one bit full adders that adds the consecutive bits initially without considering the carry bit of the previous stage as shown in the fig. 8 (b). The next stage is the ripple carry adder stage that adds the carry bit of each full adder with the carry bit of the previous stage.

With memristors, the carry save adder circuit requires 111 memristors. The same circuit on implementing with CMOS transistors requires 152 transistors.

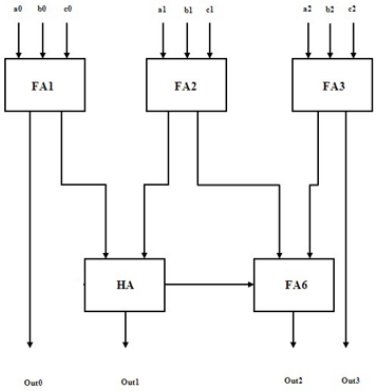
The Computation steps required for developing a Full Adder is

$$MemP \rightarrow MemW1; MemW1 \rightarrow MemT1$$
$$MemQ \rightarrow MemW2;\ MemW2 \rightarrow MemT2$$
$$MemT1 \rightarrow MemT11;\ MemT2 \rightarrow MemT22$$
$$MemP \rightarrow MemT2; MemQ \rightarrow MemT1$$
$$MemT1 \rightarrow MemSOUT; MemT2 \rightarrow MemSOUT$$
$$MemT22 \rightarrow MemTOUT; MemT11 \rightarrow MemTOUT$$
$$MemTOUT \rightarrow MemCOUT$$
$$MemP\_ \rightarrow MemW1\_; MemW1\_ \rightarrow MemT1\_$$
$$MemQ\_ \rightarrow MemW2\_;\ MemW2\_ \rightarrow MemT2\_$$
$$MemT1\_ \rightarrow MemT11\_;\ MemT2\_ \rightarrow MemT22\_$$
$$MemP\_ \rightarrow MemT2\_; MemQ\_ \rightarrow MemT1\_$$
$$MemT1\_ \rightarrow MemSOUT\_; MemT2\_ \rightarrow \boldsymbol{MemSUM}$$
$$MemT22\_ \rightarrow MemTOUT\_; MemT11\_ \rightarrow MemTOUT\_$$
$$MemTOUT\_ \rightarrow MemCOUT\_$$
$$MemTA \rightarrow MemTCarry; MemTB \rightarrow \boldsymbol{MemTCARRY}$$

This makes use of about 25 memristors to perform the full addition operation. The first seven lines perform the half addition operation.



(a)



(b)

Fig. 8: (a) CSA block diagram (b) CSA implementation with full adders and half adders

VI CONCLUSION

Performance measures

On no memristors reused for further computational steps, the number of memristors required to construct basic logic circuits are compared with the transistors with CMOS logic and is shown in the Table.

The drawback of thememristive logic circuits is that every single input must be given at consecutive time interval for which additional care is to be taken.

Sl. No.	GATES	HALF ADDER	FULL ADDER	CSA
1	IMPLY Logic with memristors	11 memristors	25 memristors	111 memristors
2	CMOS Logic	14 transistors	28 transistors	152 transistors

Table 2: Comparison of number of elements used

Reference

[1] L. Chua. Memristor-the missing circuit element. Circuit Theory,IEEE Transactions on, 18(5):507 – 519, sep 1971.

[2] Kvatinsky, S,.Kolodny, A. Weiser, U.C. Friedman, E.G. Memristor-based IMPLY logic design procedure. Computer Design

(ICCD), 2011 IEEE 29th International Conference on Date of Conference:9-12 Oct. 2011

[3] Lehtonen, E, Laiho, M, Stateful implication logic with memristors. Nanoscale Architectures, 2009.NANOARCH '09. IEEE/ACM International Symposium

[4] E. Lehtonen, J.H. Poikonen, and M. Laiho. Two memristors suffice to compute all boolean functions. Electronics Letters,46(3):239 –240, 4 2010.

[5] Strukov, D. B., Snider, G. S., Stewart, D. R. & Williams, R. S. The missing memristor found. Nature 453, 80–83 (2008).

[6] *Zdeněk BIOLEK, Dalibor BIOLEK, Viera BIOLKOVÁ*, SPICE Model of Memristor with Nonlinear Dopant Drift. RADIOENGINEERING, VOL. 18, NO. 2, JUNE 2009

[7] M. Klimo and O. Such, "Memristors Can Implement Fuzzy Logic," arXiv:1110.2074 [cs.ET], October 2011.

[8] L. O. Chua and S. M. Kang, "Memristive Devices and Systems," Proceedings of the IEEE, Vol. 64, No. 2, pp. 209- 223, February 1976.

[9] S. Kvatinsky, N. Wald, G. Satat, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Hybrid CMOS-Memristor Logic," submitted to IEEE Transactions on Very Large Scale Integration (VLSI), 2013.

[10] JOGLEKAR, Y.N., WOLF, S. J. The elusive memristor: properties of basic electrical circuits. arXiv:0807.3994 v2 [cond-mat.meshall] 13 January 2009, p.1-24.

[11] JulienBorghetti, Gregory S. Snider, Philip J. Kuekes, J. Joshua Yang, Duncan R. Stewart, and R. Stanley Williams. 'memristive' switches enable 'stateful' logic operations via materialimplication. Nature, 464:873 –876, 4 2010.

[12] Martin Setvín et al.,Reaction of O_2 with Subsurface Oxygen Vacancies on TiO2 Anatase,Science 341, 988 (2013);DOI: 10.1126/science.1239879, www.sciencemag.org