Digital Logic Implementation in Memristor-based Crossbars

Tezaswi Raja and Samiha Mourad, *Member, IEEE*
Electrical Engineering, Santa Clara University, Santa Clara, CA 95053
(traja, smourad)@scu.edu

***Abstract*— Memristor is a passive electronic device that was proposed and described by Leon Chua in 1971. The first practical implementation has been realized by Stan Williams' group at HP Labs in 2008. This paper is intended as a tutorial on how to use memristor crossbars for logic design and is a consolidation of various recent publications. The goal of this paper is to give the reader a brief introduction to the possibilities of logic design using memristors.**

I. Introduction

THE crossbar structure shown in Fig. 1 has been used to implement digital logic since the 1970s. The basic structure contains a mesh of wires with switches that may be present at junctions. The state of the switch can be open or closed. The crossbar can be used to compute logic based on the placement of these switches on the wire junctions and their state. Current crossbar based logic designs have inherent disadvantages to conventional CMOS design in terms of performance (wire delay is dominant in crossbar designs), density (large unused areas) and power dissipation (most of the unused switches in junctions are inactive and hence consume leakage power.) However, the disadvantages are mainly due to the device choice used to implement the switch. Many devices have been tried as the switch in the crossbar. First, transistors were used and the structures were known as PLAs and PLDs. However, they were too slow to compete with conventional CMOS. FPGAs used pass transistors to implement crossbar logic but the increased leakage due to pass transistor design limits the viability of this choice [Xilinx, Actel].

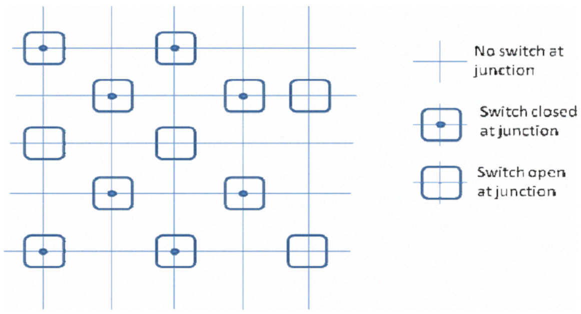


Fig. 1 Generic cross-bar structure used for logic computation. The legend is shown on the right. The switch mentioned can be implemented using diodes, transistors, or other switches. The notation is copies form [12]

More recently, with the advent of Nano-scale devices there have been many alternatives proposed for implementing the switch in the crossbar [5-8,16]. One such alternative is the *Memristor* which is based on a theoretical device formulated by Chua [1]. A practical implementation of this device has been demonstrated by Stan Williams *et al.* [9,10]. This device is the focus of this paper. Crossbar logic using a memristor has been proposed by Snider *et al.*[12]. This paper is aimed at consolidating the knowledge of these various publications.

The rest of the paper is organized as follows: Section II describes the theory and technology of the memristor. Section III describes the basic logic operations using memristors. Section IV describes a memristor based crossbar and its uses for logic computation with examples. Summary and conclusions are reported in the final Section.

II. The HP Memristor

A. Theory of Memristors

"Memristor" was coined by Leon Chua in a 1971 seminal paper [1] for a two-terminal element characterized by a relation of the type $g(f,q)=0$. It is charge (or flux) controlled if it can be expressed as a single valued function of q or f. The voltage across the device is $V(t) = M(q(t))I(t)$, where the memristance is given by:

$$M = d\Phi/dq = (d\Phi/dt)/(dq/dt$$

It can be inferred from this relation that memristance is simply a charge-dependent resistance.

As an alternating voltage is applied at the terminal of the device, the *I-V* curve is characterized by a *pinched hysteresis loop* that passes through the origin as illustrated in Fig. 2. This is a 2-valued function of the current. Around the origin, the device acts as a traditional resistance (linear relation). However, at $V = + V_c$, the device retains its resistance and hence, the memory attribute of the memristor. Accordingly, the conductance is either zero or at a higher value depending on the device.

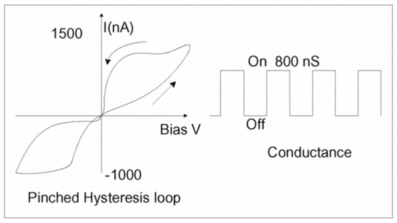


Fig. 2 (a) Memristor I-V characteristics of a memristor; (b) Its conductance in response to the application of alternating pulses.

More work on theoretical memristors has been continued by Chua and professional colleagues [2-4]. In these articles, Chua mentions a long list of papers where hysteric-type

Manuscript received March 27, 2009. This work was supported in part by William and Janice Terry funds.

curves associated with nano-devices were mistakenly reported as amoralities. We cite here two main examples: representation of a CNT as having an inductance and a resistance that are function of frequency [17]; and Huxley nerve model [18].

B. HP Oxygen Vacancy Switch

The most acclaimed memristor realization was announced recently by HP [10]. This TiO_2/TiO_{2x-} switch, shown in Fig. 3 is redrawn from [11] and it is a voltage regulated device. The resistance between the terminals may be large enough (TiO_2) to represent the switch in an open position or a very low value to represent the switch closed. By applying a voltage pulse to the terminal of the switch, the resistance can be changed. The final resistance is retained and can be kept, practically speaking indefinitely.

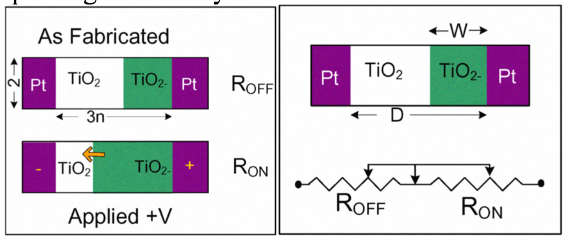


Fig. 3 (a) Memristor; (b) Its Model [11]

The memristance of this device is given by:

$$M(q) = R_{OFF}(1 - q(t)[\mu R_{ON}/D^2])$$

III. LOGIC DESIGN USING HP MEMRISTORS

In this Section, we first look at the Memristor as a state element that can store logic data. We describe the basic operations on a single memristor that can change its state. The implementation of a wired-AND using multiple memristors is also illustrated.

A. Memristor as a state element

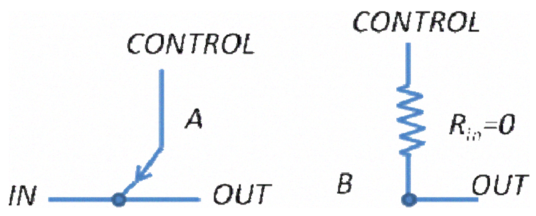


Fig. 4: Memristor as a logical state element. A) A closed memristor showing logic "0" and B) Equivalent circuit during readout.

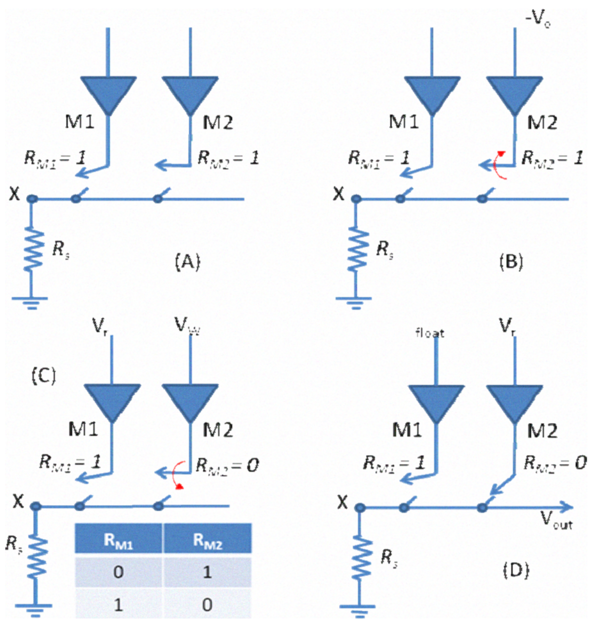
Consider the memristor in Figure 4(A). As shown the memristor is a two terminal device connected between terminals *control* and *in*. Note that even though terminals *in* and *out* are on the same net, they will be used differently depending on the operation being performed on the memristor. Now a memristor can be used as a memory element or a latch. However, the memristor holds logical state as an impedance value and not as a voltage. If a memristor is closed(as in Figure 4(A)), it has zero impedance across *in* and *control*, and hence represents a logic "0". This logic state "0" can be read out by applying a positive voltage at *control* and reading the Voltage at *out*. A high voltage at *out* designates a logic "0"(since impedance Rin=0). The equivalent circuit during readout is shown in Figure 4(B). Note that the output of a memristor latch is a Voltage. Similarly, an open memristor contains a non-zero impedance and designates logic "1".

B. Basic Logic Operations on two Memristors

The basis for any logical computation is the reliable transfer of state from one state element to the next. This transfer of state can be achieved from one memristor to the other in two possible ways: *inverting* and *non-inverting*. The differences between these two configurations described below are the presence of the extra Resistance R_s and the voltages applied at the terminals.

1) Inverting Configuration

Two memristors can be connected in an inverting configuration as shown in Figure 5(A).



R_{M1}	R_{M2}
0	1
1	0

Figure 5: Inverting configuration of two memristors. (A) Showing driving memristor M1 and receiving memristor M2. (B) The memristor M2 is unconditionally opened. (C) The memristor M2 is conditionally closed based on state of M1. The inverting truth table of operation is also shown. (D) The value stored in M2 is read out at terminal Vout.

The driving memristor M1 is placed in read out mode where its value is not changed. The receiving memristor M2 can change state based on the value of memristor M1, thus creating a logical computation. The logic computation can be achieved in three stages as shown below.

a) Unconditional Open

The memristor M2 is preset to the open state by forcing a high impedance on *control* of M1 and by applying a high negative voltage $-V_o$ at the *control* of M2. The voltage $-V_o$ is above the threshold required to open the memristor and hence the state of the memristor M2 is set to "1" irrespective of its previous state. This case is shown in Figure 5(B).

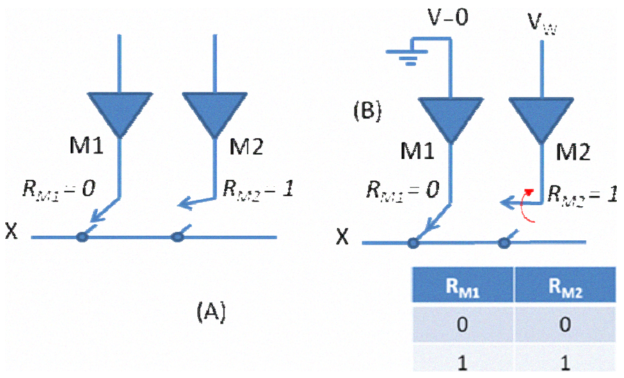
b) Conditional Close

This stage as shown in Figure 5(C), has a voltage V_r applied at the *control* terminal of M1 which places the memristor M1 in read out mode(its state cannot be changed). Now a Voltage V_W is applied at the *control* node of M2. There can be two scenarios that happen based on the state of memristor M1. If memristor M1 is closed, then the impedance R_{M1} = 0. This makes the voltage at the intermediate node X to be close to V_r due to the presence of the resistor R_s. The voltage across the M2 is (V_W - V_r) which is not enough to close the memristor M2 and it remains open. The other scenario is when M1 is open (R_{M1} = 1). The voltage on node X is close to 0 and the voltage across M2 is V_W. This voltage drop is over the threshold of the memristor M2 and it closes the memristor (R_{M2} = 0). Thus, the logical action of the inverting configuration is as shown in the truth table in Figure 5(C). The memristor M2 takes the inverted value of memristor M1.

c) Read out

The inverted value of M1 has been written in to M2 as in the previous step. Now the stored value in M2 is used as input to the next stage of computation by configuring as shown in Figure 5(D). The control node of M1 is placed in high impedance state to not have an effect on the intermediate node X. Now a voltage V_r is applied at the control terminal of M2. This will result in a Voltage at *out*, that can be used for further computation (Voltage high = Logic 0 and vice versa).

2) Non-inverting configuration



R_{M1}	R_{M2}
0	0
1	1

Figure 6: Non-inverting configuration of two memristors. (A) Showing driving memristor M1 and receiving memristor M2. (B) The memristor M2 is conditionally closed based on state of M1. The non-inverting truth table of operation is also shown.

Two memristors combined in this configuration are shown in Figure 6(A). It is similar to the inverting configuration described earlier, but with the resistance R_s removed. It takes three stages for the logical computation in this configuration as described below.

a) Unconditional open

This can be achieved by applying a high Voltage -V_o at the *control* terminal of M2(same as inverting configuration).

b) Conditional close

This can be achieved by connecting the *control* terminal of M1 to GND(V=0) and connecting the *control* of M2 to voltage V_W. The two scenarios are shown in Figure 6(B). If M1 is closed (R_{M1} = 0), then the intermediate node voltage V_X=0 and this will create the voltage across M2 to be V_W. This voltage is enough to close the memristor M2 making its logic state R_{M2}=0. If M1 were open(R_{M1}=1), then the node V_X is un-driven and floating. This makes the Voltage across M2< V_W and it leaves the memristor M2 in open state. The truth table of the computation is shown in Figure 6(B). It can be noted that in this configuration, the logical value of M1 is copied into M2.

c) Read out

The read out in the non-inverting configuration is similar to the inverting configuration. The *control* of M1 is un-driven(high impedance state) and the *control* of M2 is driven with V=*0*(GND). This results in a voltage at the *Out* terminal of M2, which can be used as input to another memristor.

C. Logic operation on multiple memristors (wired-AND logic)

In the previous Section, we have seen two memristors combined to create both inverting and non-inverting logic functions. In this Section, we look at combining the state of multiple memristors to create the wired-AND logic function. This function will form the basis of the crossbar logic module which will be described later.

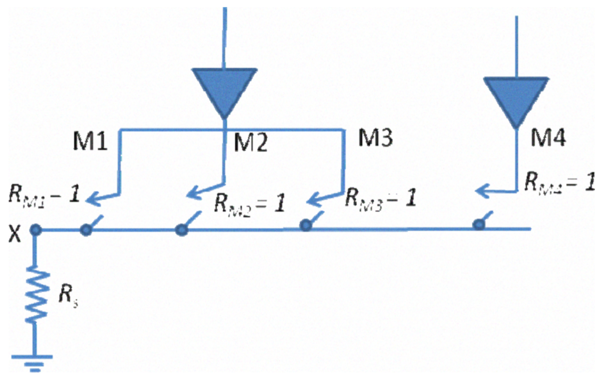


Figure 7: Wired-AND logic implementation. The memristors M1-M3 are connected in the Inverting configuration as shown. The output will be stored in RM4 which will be (RM1.RM2.RM3)'.

Consider the set of memristors shown in Figure 7. The configuration is similar to the inverting configuration shown earlier in Figure 5A. The memristor M1 in Figure 5A is replaced by a set memristors M1-M3, which are all connected in parallel.

Note that the control terminals of M1-M3 are shorted together. The working principle is similar in this configuration too. The computation can be performed in three stages as before.

The memristor M4 is unconditionally open by applying a high negative Voltage V_o at the *control* terminal. The conditional close phase is achieved by applying a voltage V_r at the common *control* terminal and V_W at the *control* terminal of M4. Now consider the scenario where all memristors M1-M3 are open($R_{M1} = R_{M2} = R_{M3} = 1$). In this case, the voltage at the common terminal X is close to 0. The voltage drop across the memristor M4 is V_W, which is enough to close the memristor(R_{M5}=0). Now consider the scenario where one memristor M1 is closed and M2 and M3 are still open. The intermediate node settles close to V_r(see Footnote1) and the voltage drop across M4 is not enough to close the memristor. Thus, the output of the computation is that M4 remains open or R_{M4}=1. It is trivial to extrapolate this result to all cases in which at least one of the memristors M1-M3 is closed. So, the result is R_{M4}=1 for all cases except when R_{M1}= R_{M2}= R_{M3}=1. Hence, the logical computation can be treated as:

$$M4 = (M1.M2.M3)'$$

which is nothing but a NAND gate. This configuration is referred to as a "wired-AND" as the various inputs can be wired together to produce the result.

IV. Memristor Crossbar based logic design

In the previous section, we have looked at the underlying principles of logic design using memristors. In this Section, we will describe the characteristics of Memristor based crossbar arrays, and how they can be used for logical computation.

A. Memristor Crossbar Array

An example of a memristor based crossbar array is shown in Figure 8. It is similar to a regular crossbar array but the wire junctions contain memristors instead of the usual diodes or transistors. As shown in the legend of Figure 8, any wire junction can have a memristor that is open, closed or in an unknown state and some junctions do not have a memristor altogether(see Footnote2). Each wire in the crossbar contains drivers that can possibly be connected to various voltages as needed for computation. The crossbar shown is a small tile that can be replicated to create larger tiles for more complex operations.

Footnote1: To be strict, when only one memristor is closed, a potential divider exists between the closed memristor and the resistance Rs. Hence, the voltage at the intermediate node is close to Vr/2. But in this case too, the drop across M4 is not enough to close the switch and M4 remains open.

Footnote2: From a manufacturability perspective, it might be expensive to fabricate memristors on certain junctions only. So, it is conceivable that all wire junctions are manufactured with memristors but the memristors in certain junctions are destroyed(made permanently open) by applying a drastically high voltage at the terminals.

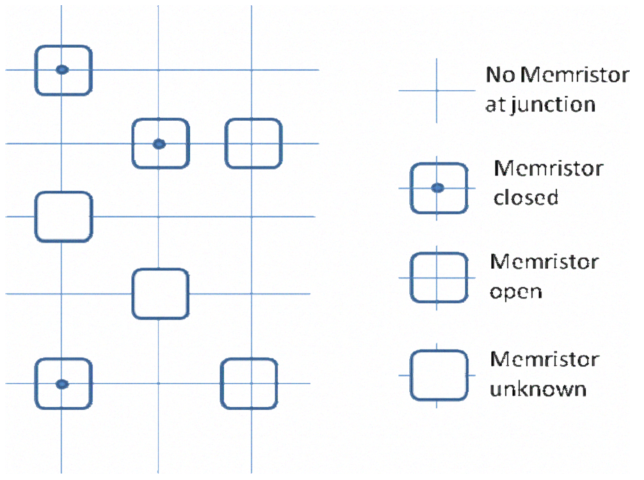


Figure 8: An example crossbar with Memristors at wire junctions. The legend is shown on the right. Each wire junction can contain no memristor, or one memristor which is closed, open or in an unknown state.

B. Implementing a NAND using a crossbar

A 3-input NAND gate implementation using a Memristor crossbar array is shown in Figure 9(A).

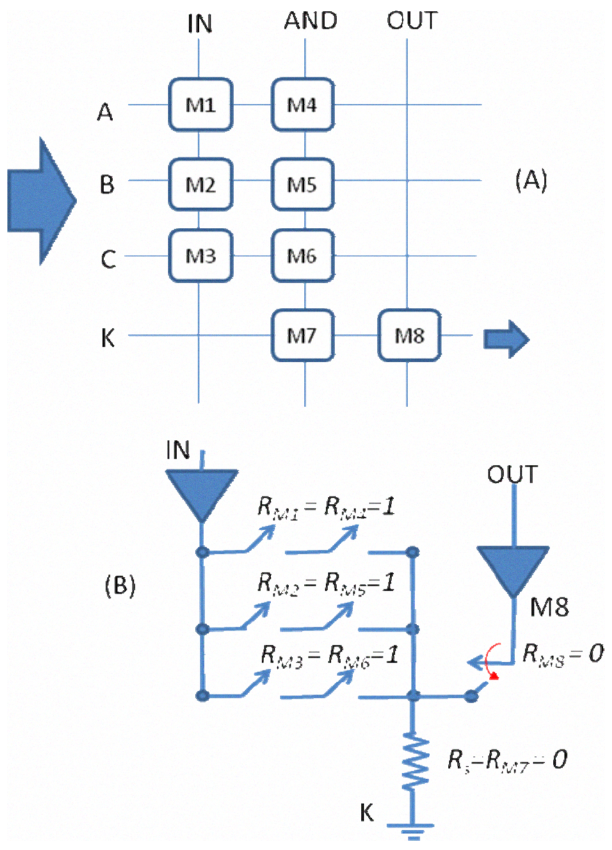


Figure 9: Memristor crossbar based implementation of a 3-input NAND gate. (A) The crossbar needs a 4x3 tile as shown. (B) The equivalent circuit of the NAND gate with the case where all inputs A=B=C=1. The memristors are connected in the inverting configuration and a conditional close operation is performed on memristor M8.

It contains 8 memristors connected as shown in the Figure. The circuit shown can function as a NAND gate by applying the following sequence.

1) *Unconditional Open*

Open all junctions by applying a large negative voltage at all control terminals.

2) *Latch inputs*

In this stage, the input data from the three input terminals A,B,C is stored in the memristors M1-M3. To accomplish this, we need to apply a voltage V_W at the input terminal *IN*. Considering that the inputs A,B,C are outputs of other memristors (which are also impedance encoded), the logic state of signals A,B,C is latched in to M1-M3 respectively. This is similar to the conditional close in the non-inverting configuration as described in Section III.

3) *Copy inputs & Close-AND*

In this stage, we try to copy the state of M1-M3 to M4-M6 respectively. This can be achieved by connecting the M1-M3 in non-inverting configuration with M4-M6 as described earlier. A voltage of V_W is applied at the *AND* terminal and $V=0$ applied at *IN*. The other objective of this stage is to close the memristor M7. This can be achieved by applying $V=0$ at K and V_W at *AND*. This closed memristor will be used in the next stage as the Rs in inverting configuration.

4) *Evaluate and Capture*

In this stage, a voltage V_r is applied at *IN* and V_W is applied at the *OUT* terminal. The equivalent circuit for this stage is shown in Figure 9(B). Notice the similarity between this configuration and the wired-AND configuration of Figure 7. It can easily be identified that the memristors M1 and M4 can be simplified to a single resistance, as both of them contain the same state. The combined resistance of M1 and M4 (call it M1) can be thought similar to the memristor RM1 in Figure 7. Similarly the pairs M2-M5 and M3-M5 in Figure 9B are analogous to M2 and M3 in Figure 7. Now the memristor M7 in Figure 9B acts as the R_s in Figure 7. The result in both cases is the "inverted wired-AND" value of the logic stored in the memristors. As noted before, the output (which is stored in M8 in this case) can be expressed as: M8 = (M1.M2.M3)'
Thus, the NAND function of the inputs is captured in the memristor M8.

5) *Open AND*

In order to be able to read the output of the memristor M8, we need to disable the memristor M7. This can be achieved by applying a large negative voltage on AND and opening memristors M4-M7.

6) *Read Output*

The output of the memristor M8 can be read out by applying a voltage $V=0$ at OUT and using the output terminal as an input to a subsequent crossbar logic gate.

V. CONCLUSION REMARKS AND COMPARISONS

Although so far we stayed close to HP logic implementations as explained in many of their public presentations and in their US patent [12], it is actually possible to use other perspectives.

A. *Implementing an output that is a "Sum of Products"*

For, example, we can organize the crossbar plane into AND-blocks and OR-blocks (see for example Fig. 1 of [16]) and , use the same tools available for PLA/PLD design.

B. *Use of Implication logic*

It would be more challenging however, to take advantage of the attribute of HP memristor to implement logic in material implication, since it is more likely that this approach will help in reducing the circuit size even further (fewer switches needed when compared to a crossbar using transistors.)

C. *Advantage of the Technology*

There are many: drastic reduction of size, higher speed of operation and lower power dissipation. In addition, the architecture is defect tolerant. More importantly, the design, verification and test tools do not demand a drastic change from current practices when compared to the change encountered by the manufacturing.

REFERENCES

[1] L.O, Chua, Memristor-missing circuit element, IEEE Tans. Circuit Theory, Vol. 18, 1971, pp. 507-519.
[2] L.O. Chua, and S. M. Kang, Memrestive devices and systems, Proc. Of the IEEE, Vol. 64, 1976, pp.209-223.
[3] L.O, Chua, Device modeling via basic nonlinear circuit elements, IEEE Tans. Circuit & Systems, Vol. 27, 1980, pp. 1014-1444.
[4] L.O. Chua, Nonlinear circuit foundations for nanodevices, Proc. Of IEEE, Vol. 91, 2003, pp. 1830-1859.
[5] Y. Chen, et. al., Nanoscale molecular-switch crossbar, Nanotechnology, Vol. 14, 2003, pp. 462-468.
[6] D.R. Stewart, et al., Molecule independent electrical switching in Pt/Organic monolayer/Ti devices, Nano-Letters, Vol. 4, 2004, pp. 133-136.
[7] C,A, Richter, D.R. Stewart, D.A.A. Ohlberg, and R.S. Williams, Electrical characterization of Al/AlOx/molecule/Ti/Al devices, Appl. Phys. A, Vol. 80, 2005, pp. 1355-1369.
[8] J.J. Blackstock, et al., Infernal structure of a molecular junction device, Phys. Chem. Letters, Vol. 111, 2007, pp. 16-20.
[9] W. Robinett, G.S. Snider, P.J. Kuekes, and R. S. Williams, Computing with a trillion crummy components, Com. Of the ACM, Vol. 50, 2007, pp. 35-39.
[10] D.B. Stukov, G.S. Snider, D.R. Steward R.S. Williams, The missing memristor found, Nature, Vol. 453, 2008, pp. 80-83.
[11] S.William,http:www.cpmt.org/scv/meetings/cpmt0902.html.
[12] Greg Snider et al., Architecture and methods for computing with reconfigurable resistor crossbar. US Patent 7,203,789, 2007
[13] R. H. Chen, A. N. Korotov, K. K. Likharev, Single-electron transistor logic,Appl. Phys. Lett. **68**, 1996, pp. 1954
[14] Islamshah, et. al., Digital Logic Gate Using Quantum-Dot Cellular Automata Science Vol. 284,1999, pp. 289 – 291
[15] C.P Collier et. al., Electronically Configurable Molecular-Based Logic Gates, Science Vol. 285. 16, 1999, pp. 391 – 394.
[16] A. DeHon, Array-based architecture for FET-based Nanoscale electronics, IEEE Trans. Nanotechnology, Vol. 2, 2003, pp. 23-32.
[17] Y. Myamoto, A. Rubio, S.G. Loiie, a nd M. L. Cohen, Self inductance of chiral conducting nanotubes, Phys. Rev. B, Vol. 60, 1999, pp 13885-13889.
[18] A.L. Hodgkin, and A.F. Hyxley, A quantative description of membrane curent and its application to conduction in nerve, J. Phys.. Vol. 117, 1952, pp. 500-544.