Reconfigurable Architecture in Resistive Switching Crossbar

THESIS

Submitted in partial fulfilment

of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

MANE PRAVIN SAKHARAM

Under the Supervision of

Dr. Ramesha C. K.

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

2016

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the thesis entitled **Reconfigurable Architecture in Resistive Switching Crossbar** which is submitted by **Mane Pravin Sakharam** ID No **2009PHXF420G** for award of Ph.D. of the Institute embodies original work done by him under my supervision.

Signature of the Supervisor:

Name in capital block letters : **DR. RAMESHA C. K.**

Designation : **ASSISTANT PROFESSOR**

Department of Electrical & Electronics Engineering

Date : 10/08/2016

Declaration

I, Mane Pravin Sakharam, hereby declare that this thesis entitled "Reconfigurable Architecture in Resistive Switching Crossbar" submitted by me under the guidance and supervision of Dr. Ramesha C. K. is a bonafide research work. I also declare that it has not been submitted previously in part or in full to this University or any other University or Institution for award of any degree.

Signature of the Student: 

Name of the student : MANE PRAVIN SAKHARAM

Reg. No.: 2009PHXF420G

Date : 10.08.2016

Abstract

Major reasons behind inferior performance of FPGA over ASIC in terms of delay, area and power consumption are extensive use of SRAM and programmable interconnects. In order to improve the performance gap between ASIC and FPGA so as to increase the share of FPGAs in the market, new emerging devices are being investigated as a replacement for SRAM and programmable interconnection switches. Memristor is one of the most attractive device which can act as nonvolatile memory (by storing data in the form of resistance) and as logic (switching) element. But it is passive device and CMOS circuits are required to implement functions using it.

Out of the available models, VTEAM model found to be more simple, accurate and flexible, and hence used in the work carried out in this research. Stateful NOR, an universal logic gate, can be implemented with memristors. Memristors are seldom used as standalone devices and are fabricated in the form of crossbar over CMOS layer using nanoimprint lithography. Implementation of stateful NOR gate on memristive crossbar requires write, evaluate (imply) and read operations, but memristors can not be isolated from crossbar for these operations.

Memristive crossbar arrays are analyzed for write and evaluate (imply) operations in order to find the effect of them on contents of memristors of crossbar that are not part of logic and limitations on the size of crossbar to keep them unaltered. Sneak path problem is common in such crossbar and analysis of read operation has confirmed that the size of crossbar has to be restricted in order to read the state of memristor correctly. Specialized memristive architectures which logically restricts the size of crossbar array are also investigated for use in implementation of NOR operation.

Crossbar array made up of Complementary Resistive Switches (CRSs) are free from sneak path problem. Novel stateful NOR operation with CRSs using a single voltage source is proposed in this work. The analysis of write, read and evaluate (NOR) operations on CRS crossbar to implement stateful NOR gate is carried out in order to find the effect of such operations on CRS cells that are not involved in operation. The integrity of such cells can be maintained in CRS crossbar by size restriction on crossbar and self-resetting read scheme.

Memristive/CRS crossbar based 3-input pipelined reconfigurable logic block architecture has been proposed in this work for implementation of logic functions. For logic function with more than three inputs, the multiplexing of proposed 3-input logic blocks for its implementation is explained.

Automation algorithm for proposed 3-input logic block architecture and its multiplexing are also presented.

CRS crossbar is the most natural choice for implementation of proposed architecture as it is free from sneak path problem. The performance analysis of implementation of proposed architecture on memristive and CRS crossbar is carried out with respect to delay, power and area, and is compared with performance of LUT based CLB architecture used in many commercial FPGAs. For array size of 128 $\times$ 128, memristive crossbar implementation has shown 1.45 times improvement in delay for large data set, 2.68 times improvement in power while CRS crossbar implementation has shown 1.28 times improvement in delay (for large data set), 2.04 times improvement in power, over LUT based CLB architecture. For area, this improvement is from 1.8 times to 5.6 times in both type of crossbars.

Acknowledgment

I would like to express my sincere gratitude to my supervisor and mentor Dr. Ramesha C. K. for the continuous support in conducting this research, for his patience and motivation. He had whole heartedly helped me in this endeavor at all stages of this work. His guidance helped me in all the time of research and writing of this thesis. From bottom of my heart, I extend my sincere thanks to him for all his efforts and help at various levels without which this work not have been completed.

I am very grateful to members of my Doctoral Advisory Committee for thesis mentorship and guidance, Prof K. R. Anupama and Dr. Narayan Manjarekar for sparing their valuable time in reviewing my thesis and giving constructive suggestions to improve it. Their valuable suggestions have helped in greatly enhancing the quality of the thesis.

I would extend my sincere thanks to Prof. M.K. Deshmukh, Professor and Head, Dept.of EEE for his constant support and encouragement at various levels.

I thank Prof. Sauvik Bhattacharya Vice-Chancellor, BITS Pilani, Prof. Sasikumar Punnekkat, Director, BITS Pilani-K. K. Birla Goa Campus, Prof. K. E. Raman, Former Director, BITS Pilani-K. K. Birla Goa Campus, Prof. Ashoke Kumar Sarkar, Director, BITS Pilani - Pilani Campus, Prof. G. Raghurama, Former Director, BITS Pilani - Pilani Campus, Prof. S. K. Verma, Dean, ARD, BITS Pilani - Pilani Campus, Prof. Prasanta Kumar Das, Associate Dean, ARD, Prof. Sunil Bhand, Dean, SRCD, Prof. D. M. Kulkarni, Dean, Administration, BITS, Pilani - K. K. Birla Goa Campus, and Prof. S. D. Manjare for giving me an opportunity to carry out research studies at the institute and also by providing necessary infrastructure and facilities to carry out my work.

I would extend my sincere thanks to Prof. V.K. Deshpande, former-Head, Dept.of EEE for his constant support at various levels.

I also extend my sincere thanks to Dr. Nitin Sharma, DRC Convener, for support and motivation.

I also thank Prof. Dipankar Pal, Dr. Amalin Prince, Dr. Gautam Bacher, Dr. Anita Agrawal, Dr. Chandram, Mr. C. Balakrishna Moorthy, Mr.Meghanand Bhamare, Mr. Sarang Dhongdi, Mr. Sankhar Reddy, Ms. Meetha Shenoy for their constant support, motivation and suggestions.

I acknowledge with due gratitude to Mr. Nishil Talati, Ameya Riswadkar, Mr. Ramesh Raghu, Mr. Sudeep Mishra, Mr. Ravish Deliwala, Mr. Vikas Khairnar, Mr. Abhishek Joshi for their constant support and motivation.

I would extend my sincere thanks to Mr.Pusharaj Paradkar, Mr. Sameer Chodankar, Mr. Anil Lamani and Mr.Shivaraj Rathod, lab technicians for their constant help during my research work carried out in the lab.

I acknowledge all my colleagues for their continuous support, encouragement and motivation.

I would like to express my sincere gratitude to my loving parents Mr. Sakharam Mane and Mrs. Ratnamala Mane, for always believing in me, for their continuous support, encouragement, motivation and prayers.

I thank my wife Mrs. Swati for taking responsibilities of family on my behalf and giving encouragement in carrying out work. I thank my beloved son, Pratyush Mane. Without their support, I could not have made it here.

I thank my family members Mrs. Prabhavati Jagtap, Mrs. Suvarna Patil, Ms. Shrutika Shinde, Mrs. Pooja Sawant, Mrs. Shradha Dalvi, Ms. Tirtha, Ms. Shreya and all other members for their pertinence and patience during this research. They deserve special thanks for their continuous encouragement and motivation.

I am thankful to all my friends and relatives who directly or indirectly helped me in completing my thesis.

Above all, my gratitude towards the almighty that worshiped me with blessings and mercy.

Mane Pravin Sakharam

Contents

List of Figures

List of Tables

Abbreviations

VLSI	**V**ery **L**arge **S**cale **I**ntegration
FPGA	**F**ield **P**rogrammable **G**ate **A**rray
ASIC	**A**pplication **S**pecific **I**ntegrated **C**ircuit
LUT	**L**ook **U**p **T**able
CLB	**C**onfigurable **L**ogic **B**lock
SRAM	**S**tatic **R**andom **A**ccess **M**emory
ALU	**A**rithmetic **L**ogic **U**nit
CMOS	**C**omplementary **M**etal **O**xide **S**emiconductor
LB	**L**ogic **B**lock
CB	**C**onfiguration **B**lock
SB	**S**witch **B**lock
BLE	**B**asic **L**ogic **E**lement
CPU	**C**entral **P**rocessing **U**nit
CRS	**C**omplementary **R**esistive **S**witch
LRS	**L**ow **R**esistance **S**tate
HRS	**H**igh **R**esistance **S**tate
ReRAM	**Re**sistive **R**andom **A**ccess **M**emory
NEM	**N**ano **E**lectro **M**echanical
NVM	**N**on **V**olatile **M**emory
MRAM	**M**agnetic **R**andom **A**ccess **M**emory
FeRAM	**Fe**rroelectric **R**andom **A**ccess **M**emory
PCRAM	**P**hase **C**hange **R**andom **A**ccess **M**emory
PCM	**P**hase **C**hange **M**emory

MLC	**M**ulti-**L**evel **C**ell
SLC	**S**ingle-**L**evel **C**ell
CMOL	**C**MOS/ **MOL**ecular
FPNI	**F**ield **P**rogrammable **N**anowire **I**nterconnect
FPSLA	**F**ield **P**rogrammable **S**tateful **L**ogic**A**rray
TEAM	**T**hr**E**shold **A**daptive **M**emristor
BEOL	**B**ack **E**nd **O**f **L**ine

Symbols

R	resistance	Ω
C	capacitance	F
L	inductor	H
i, I	current	A
v, V	voltage	V
q	charge	C
ϕ	flux	Vm
p, P	power	W
t	time	s
R_{HRS}	resistance of memristor in OFF state (High Resistance State)	Ω
R_{LRS}	resistance of memristor in ON state (Low Resistance State)	Ω
$\mathrm{V_{th1,M}}$	threshold voltage of memristor to switch from HRS to LRS	V
$\mathrm{V_{th2,M}}$	threshold voltage of memristor to switch from LRS to HRS	V
$\mathrm{V_{th1,C}}$	threshold voltage of CRS to switch from HRS-LRS to LRS-LRS state	V
$\mathrm{V_{th2,C}}$	threshold voltage of CRS to switch from HRS-LRS to LRS-HRS state	V
$\mathrm{V_{th3,C}}$	threshold voltage of CRS to switch from LRS-HRS to LRS-LRS state	V
$\mathrm{V_{th4,C}}$	threshold voltage of CRS to switch from LRS-HRS to HRS-LRS state	V

Chapter 1

Introduction

1.1 Introduction

Three approaches are available for design of applications in digital domain.

- ASIC based approach.
- General purpose microprocessor based approach.
- FPGA based approach.

In ASIC approach, the application is implemented using dedicated hardware. The optimization is carried out with respect to one or more parameters like speed (delay), area (density), power consumption, cost, noise margin etc. These circuits are generally the fastest, the densest and the most power efficient for the specific application. However, such circuits are becoming increasingly expensive with progress in technology (fabrication cost is increasing exponentially due mask cost, quality of materials required in processing, environmental requirements in fabrication lab, stringent requirements over the light sources used in photo-lithography). Leakage current is posing a problem in continuing with Moore's law. Also, the implementation of application on ASIC takes longest time.

Microprocessors are prefabricated circuits, which are programmed by a series of instructions for specific applications. These instructions are stored in memory while data may be present in memory or provided by input devices. A control unit in the form of built-in finite state machine reads instructions from memory and executes them sequentially in the specific order defined in the program. All arithmetic and logical computations are performed in datapaths consisting of an arithmetic logic unit (ALU), a floating-point unit, and a load-store unit. This approach has advantages like flexibility and short application development time because of higher level of abstraction used for application design. However performance and energy efficiency of such system is severely affected by processor-memory bottlenecks and computing overhead.

FPGAs combine some of the best properties of microprocessors and ASICs. Like microprocessors, they are prefabricated and programmable circuits and hence very cost-efficient. Similar to ASICs, they can be customized for efficient datapaths *i.e.* they can be fine grain customized. Also massively parallel operations can be efficiently implemented on FPGA.

1.2 FPGA: Basic Architecture

Figure 1.1 shows generic high-level architecture of an FPGA having uniform structure consisting of logic blocks, interconnection network and input/output blocks. Logic blocks are used to implement logic functions, may be in the form of sea-of-gates, look-up tables (LUT) or programmable NAND and NOR planes. The LUT based logic block structures are commonly used in most commercial FPGAs. If large size functions (many inputs and/or many outputs) are to be implemented, then interconnections between logic blocks are necessary and is done by programmable interconnects. The interconnection of data to input/output devices is done through I/O blocks.

Multiplication is very common and frequent operation in many applications. When implemented on generic FPGA, it occupies more area with more delay from input to output. Hence dedicated hardwired multiplier blocks optimized for speed, area and power consumption are embedded in the architecture of FPGA. Dedicated memory blocks are also the part of architecture of most commercial FPGAs. Some FPGAs have processor cores in it hence hardware-software co-design approach can be used to improve the overall system performance. Critical functions are

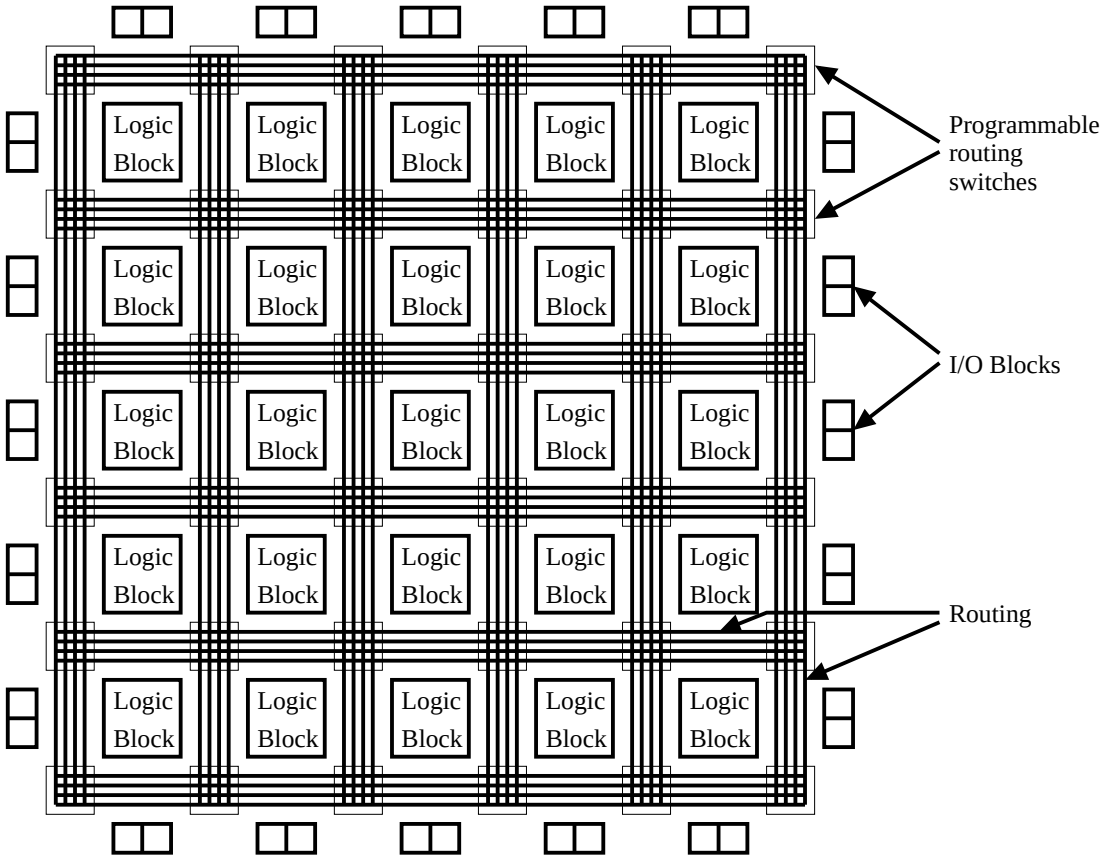


Figure 1.1. Generic homogeneous FPGA consisting of logic blocks, programmable interconnects and I/O blocks [1].

implemented on logic blocks and multiplier blocks while other functions can be simultaneously handled by embedded processors. Such heterogeneous high-level FPGA architecture is shown in Figure 1.2.

The simplified architecture of CLB with 3-input LUT is given in Figure 1.3 along with surrounding interconnection network. LUT is essentially a $k \times 2^n$ memory array, where k is size of each memory location in bits and n is number of address lines. In Figure 1.3, $k = 1$ and $n = 3$. LUT can be viewed as programmable logic gate, which can perform n-input k-output logic function by storing its truth table. Input data to such gate is used as binary address to read the contents of addressed memrory location and put it on output lines through multiplexer. The output of multiplexer is either made available directly on any interconnection lines or through flip-flop for sequential operations with the help of multiplexer. Tristate buffers are used to select the output lines to put results of logic block.

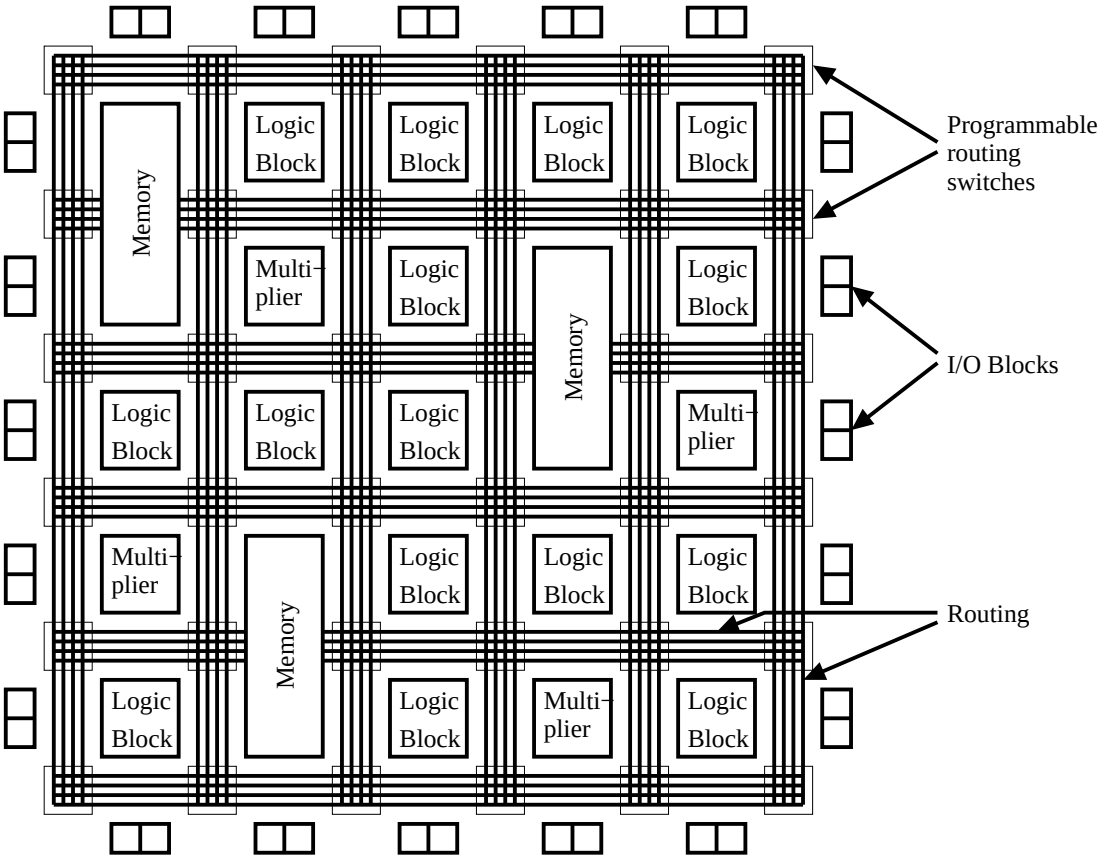


Figure 1.2. Heterogeneous FPGA consists of embedded memory blocks, dedicated hardware blocks (and processor blocks not shown here) along with logic blocks, programmable interconnects and I/O blocks [1].

Figures 1.4 and 1.5 show programmable interconnection switch which can be used to electrically connect any horizontal (vertical) line segment to adjacent horizontal and/or vertical line segment in desired way through configuration bits [3]. Inputs to CLB may be fetched from any of the four adjacent vertical or horizontal wires by configuring the input multiplexer. Similarly output from CLB can be routed to any horizontal segment through tristate buffers programmed through configuration bits. All of these configuration bits are implemented using SRAM cells. The circuit required for programming these bits is not shown in figure for simplicity. Programming FPGA for certain functionality i.e. loading configuration data and running computation on it after configuration are two independent tasks. Generally loading of configuration information is done using serial method for less area overhead.

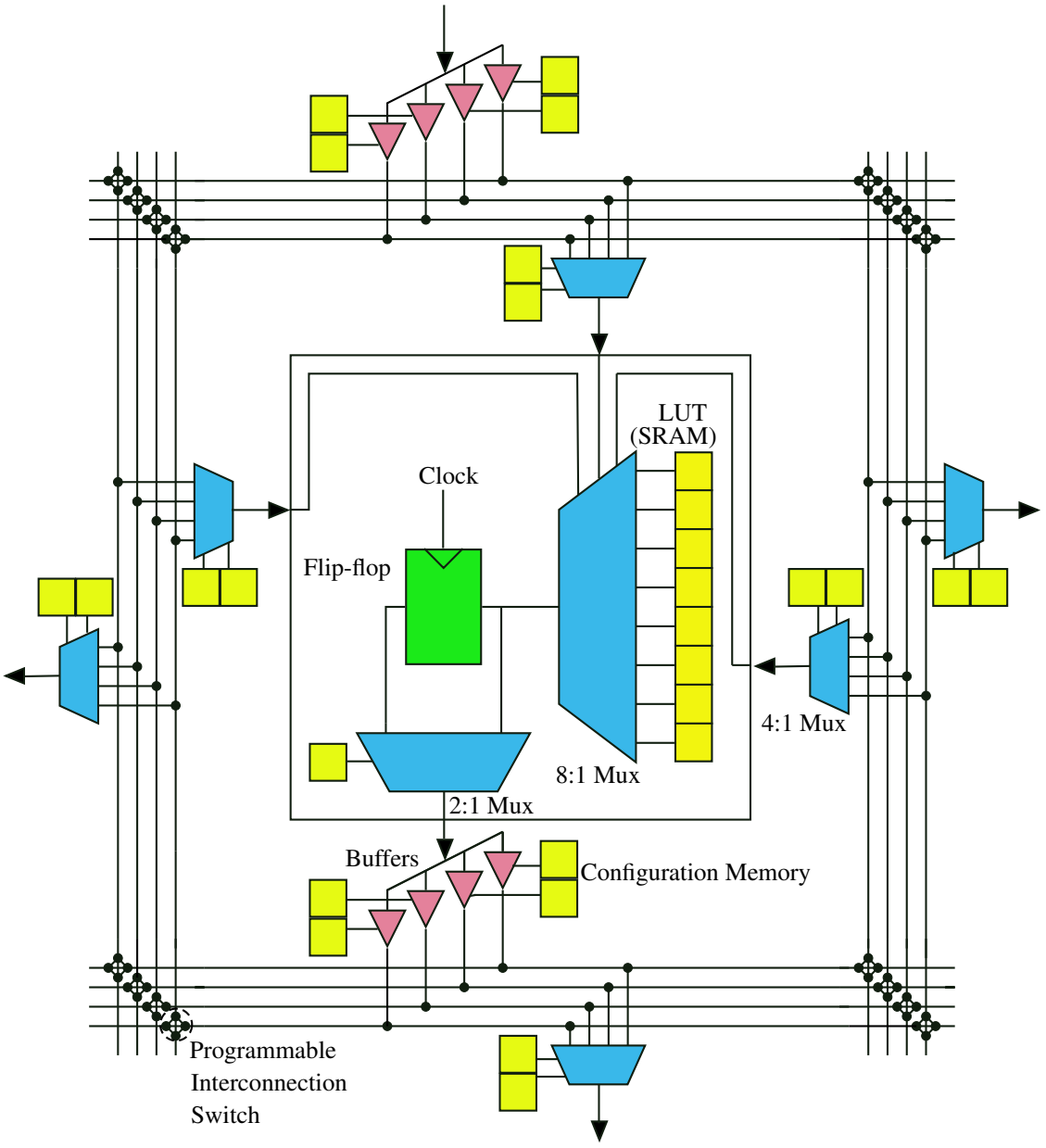


Figure 1.3. Conventional FPGA CLB Structure consisting of LUTs, multiplexers, flip-flop, configuration memory and programmable interconnects [2].

1.3 Advantages of FPGA Approach

The important features like programmability after fabrication in microprocessor and fine-grain customization in ASICs are combined in FPGA. In applications where functionality can be divided into small independent modules, concurrent executions can be implemented efficiently on FPGA due to fine grain customization. In applications such as network processing, signal and image processing, scientific computing, bioinformatics etc., this style is very useful. The execution time can be greatly reduced as a result of parallel execution. Also, in applications like Boolean

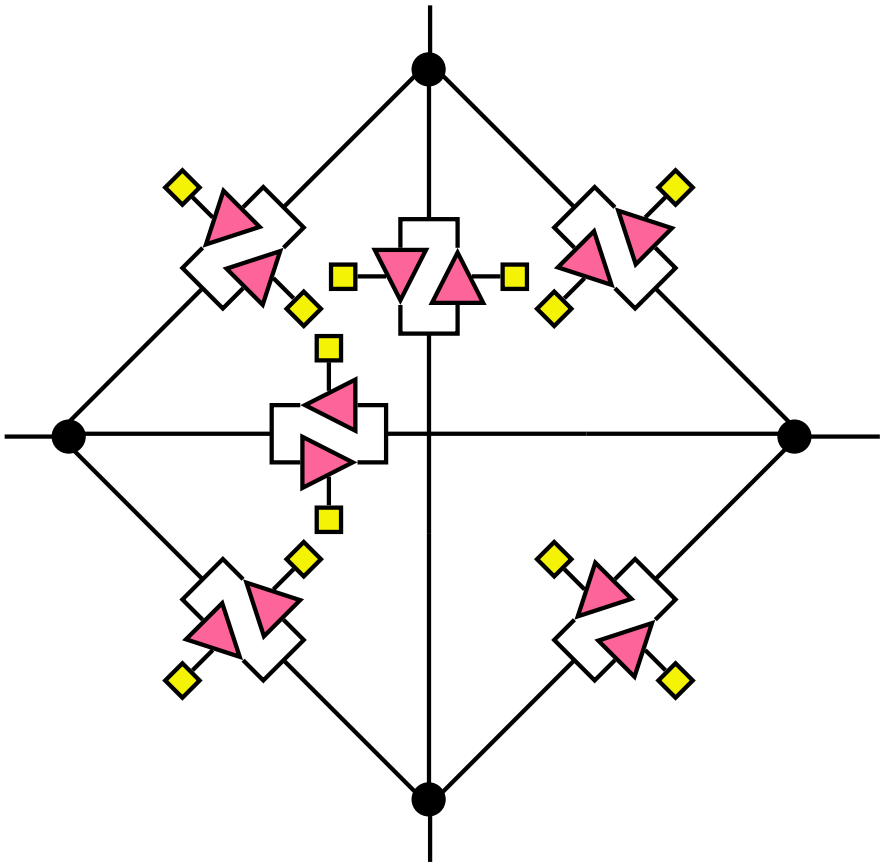


Figure 1.4. Programmable interconnection switch (version 1) consisting of buffers and programmable memory bits to control connectivity and direction of data within interconnects [3].

satisfiability (SAT) solvers, cryptography, and logic emulation, fine-grain customization at bit level is very effective. Fine-grain customization at word level has shown 90 % decrease in power dissipation in comparison to fixed-word-length implementation [4].

In applications where functions cannot be divided into independent modules, FPGAs can be used for such applications using pipelining and throughput can be improved by overlapping the execution. For example, in deeply pipelined implementations of finite impulse response filters, Fourier transforms and discrete cosine transform, FPGAs are standard platform [4].

In applications where information to be processed is not known in advance (e. g. keys in encryption algorithms, signatures by deep packet network inspectors etc.), FPGA implementation may be even denser than ASICs. In FPGA, such information is used for reconfiguration while ASICs makes use of dedicated general-purpose multipliers, decryption circuitry, or pattern-matching engines and thus has hardware overhead [4].

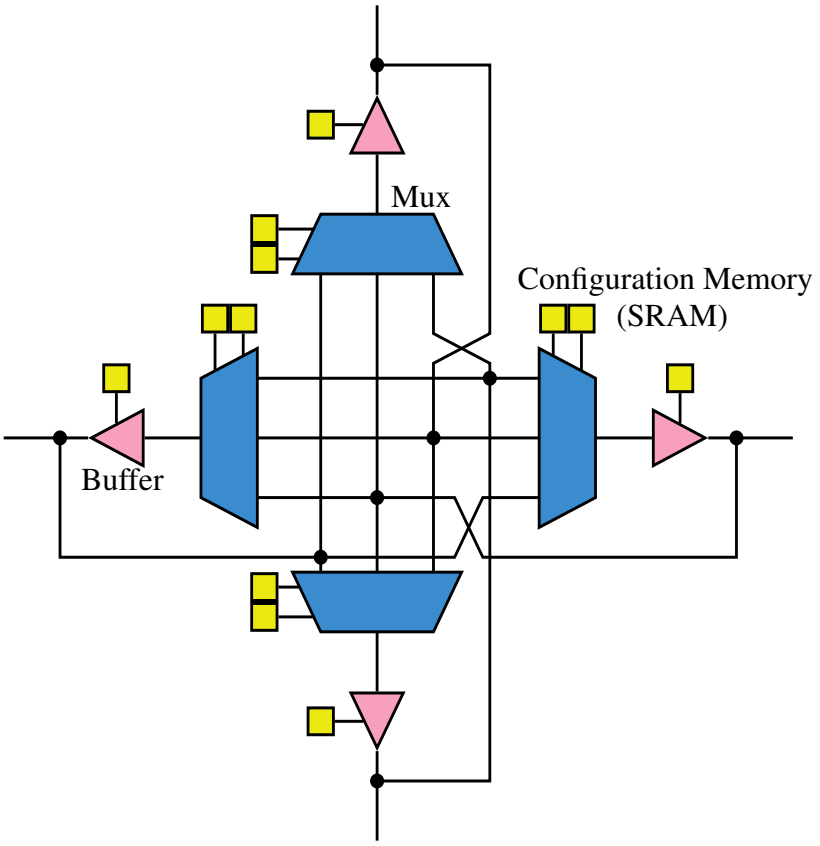


Figure 1.5. Programmable interconnection switch (version 2) consisting of multiplexers, buffers and configuration memory bits [3].

FPGAs have faster time-to-market as no layout, masks or other manufacturing steps are needed. Its design cycle is simple as the software handles much of the routing, placement, and timing tasks. The project cycle is more predictable because of elimination of potential re-spins, wafer capacities, etc. Non-recurring expenses (NRE) are not involved in FPGA designs. Also, new bitstream can be uploaded remotely due to field-programmability.

1.4 Challenges in FPGA Approach

The major challenges of FPGA approach is delay, power dissipation and area (density) overheads associated with configurability. Configuration bits occupy large portion of the configurable fabric and can be as high as 50 % [5] while majority of the remaining area is devoted to configurable routing. The available area for logic implementation is very less (from 5 % to 15 % of the total chip area) [5, 6]. Because of configurability, ASICs are two to three times denser than FPGAs, for

same function implementation [5]. The interconnect resources can not be reduced beyond certain limit as it affects routability.

Also, loading configuration information in configuration memory is slow process as writing to such memory is done sequentially to reduce area overhead of writing circuit. Also configuration word length is much higher for FPGAs. In applications where same operations is to be carried out on large set of data, FPGAs have been preferred as less reconfiguration is required. If frequent reconfiguration is required, FPGA is not good choice for such applications.

In spite of disadvantages specified above, FPGAs are a more attractive option than ASICs for certain applications is mainly because of post-fabrication programmability (so that new functionality can be added or old function can be modified), less time-to-market for application (typically much shorter (weeks) compared to that of ASICs (months)). Also, non-recurring cost is not involved in application design.

The major reason behind the inferior performance of FPGA is programmable routing fabric and extensive usage of SRAM cells in their architecture. Performance gap between ASIC and FPGA needs to be improved in order to increase the share of FPGAs in the market. New emerging devices are being investigated as a replacement for SRAM and programmable interconnection switches. Memristor is one of the most attractive device which can act as nonvolatile memory (by storing data in the form of resistance) and as logic (switching) element. But it is passive device and CMOS circuits are required to implement functions using it. In the present work, memristive devices in nanocrossbar are investigated for the development of reconfigurable architecture.

1.5 Objectives of Research Work

Several efforts have been made in the past to improve the performance of FPGA. These efforts can be broadly classified in one or combinations of the following methods :

1. Optimizing the conventional CMOS based FPGA architecture with respect to one or more parameters like area, delay, cost, power dissipation etc. by changing the number of inputs to logic element, by changing the size of cluster, by changing the size of wire segments etc.

2. Using 3D integration technology in CMOS process to reduce the size of footprint and making them denser.

3. Keeping the logic elements as it is in conventional SRAM based FPGAs and modifying the programmable interconnects by replacing the switching elements (CMOS transistors) and/or configuration memory (SRAM) by new emerging devices.

4. Modifying the design of logic elements by using emerging devices.

Objectives of the research work carried out in this thesis are listed below:

1. To study the memristor as logic element and as memory cell in order to use it in building the reconfigurable architecture (reconfigurable architectures require logic elements as well as configuration memory cells).

2. To analyze the implementation of universal logic gate using memristors on passive memristive crossbar in order to investigate the limitation on the size of crossbar due to sneak path problem commonly found in memristive crossbar. Also analysis will help in determining the size of crossbar for maintaining the integrity of memristor cells not involved in logic operation.

3. To develop universal logic gate using CRSs that can be implemented on CRS crossbar, which are free from sneak path problem, and to investigate the effect of implementation of developed universal gate on other CRSs that are not part of logic in CRS crossbar.

4. To develop reconfigurable architecture using universal logic gate as basic building block that can be implemented on memristive/CRS crossbar, along with its automation algorithm.

5. To compare the performance of developed reconfigurable architecture with conventional LUT based CLB commonly used in most FPGAs.

1.6 Organization of Thesis

The thesis is organized as follows :

- **Chapter 1:** This chapter explains the basic architecture of LUT based FPGA, advantages of FPGA approach over ASIC in application design and the challenges in improving the FPGA performance. Objectives of research work are also listed in the end.

- **Chapter 2:** As an overview of methods adopted in the past to improve the performance of reconfigurable circuits (FPGAs), detailed literature review is presented in this chapter.

- **Chapter 3:** This chapter explains the working of memristor in general, its model and window function used in this work, and use of memristor as logic element to built universal gate (stateful NOR).

- **Chapter 4:** In order to implement stateful NOR gate on passive memristive crossbar, a detailed analysis of all basic operations required to perform stateful NOR is carried out in this chapter. The limitations on size of crossbar due to sneak path problem so as to perform operations without error, and in order to maintain the integrity of memristors not involved in operation are also investigated.

- **Chapter 5:** Passive CRS crossbar free from sneak path problem are analyzed for implementation of stateful NOR gate on it in this chapter. Also limitations on size of crossbar are investigated in order to maintain the integrity of CRS not involved in operation.

- **Chapter 6:** In this chapter, novel stateful NOR logic based pipelined reconfigurable architecture is proposed which can be implemented on memristive/CRS crossbar. Also, the automation algorithm for proposed architecture is presented.

- **Chapter 7:** In this chapter, performance analysis of proposed pipelined reconfigurable architecture implemented on memristive/CRS crossbar is carried out and compared with LUT based CLB used in conventional FPGA.

- **Chapter 8:** The outcomes of research work carried out are summerized in this chapter. Also, further improvements that can be done in this research area are listed.

Chapter 2

Literature Survey

2.1 Introduction

There are mainly three types of commercial FPGAs [7]:

- Antifuse-based FPGAs : These are non-volatile FPGAs but are not reconfigurable i.e. they can be programmed only once. Examples of antifuse-based FPGAs are Axcelerator family of FPGA manufactured by Actel.
- Flash-based FPGAs : These are reconfigurable and also non-volatile but their integration with CMOS process is very difficult. Also low logic density, inadequate performance, and the lack of bit-level programmability prevent flash-based FPGAs to be widely used [8]. They consume less power and more tolerant to radiation effects. Unauthorized bitstream copying can be avoided using flash-based FPGAs. Examples of flash-based FPGAs are Igloo and ProASIC3 family of FPGA manufactured by Actel.
- SRAM-based FPGAs : These are volatile FPGAs and can be reconfigured many times during their lifetime. They can be fabricated using standard CMOS process and are currently very popular. They suffer from long configuration-loading time and excessive leakage power during stand-by. Xilinx Virtex and Spartan families, Altera Stratix and Cyclone are examples of SRAM-based FPGAs.

2.2 Literature Review

Area, delay and power consumption of FPGAs are 21 times, 4 times and 12 times higher than those of ASICs, respectively [9] and the main reason of it is programmable routing structure [5, 10–12] which account for up to 90 % of the total area [5], up to 80 % of the total delay [10, 11] and up to 85 % of the total power consumption [12]. If FPGA routing structure is improved by replacing SRAM (which is main cause of performance inferiority) with alternative memory having properties better than SRAM, the performance gap between FPGA and ASIC can be reduced.

The efforts made in the past to improve the performance of FPGA in terms of delay, area, power consumption and cost using different techniques are given below.

- Changing the size of LUT and cluster [11, 13–15] has shown limited improvement in performance of FPGA.
- Three Dimensional (3D) integration in ICs can increase the performance in terms of functionality, density and speed, and has emerged as promising means to handle the interconnection related problems and thereby has improved the performance of FPGA [16–19]. Because of smaller tile area of FPGA and shorter interconnect distance between tiles, it provided 1.7 times performance gain. However, the use of monolithic stacking is currently limited as high temperature required in the fabrication of transistors in an upper layer may destroy the transistors and metal layers already fabricated in the lower layer [20]. Also, 3D stacking will result in excessive increase in heat density and corresponding degradation in performance if proper thermal solution is not provided [21].
- C. Chen et al. [22] replaced SRAM with NEM relays in programmable routing of FPGA. NEM relay has zero leakage power and potentially low ON-resistance. Also its fabrication is a low-temperature process and can be safely monolithically integrated above CMOS layer. This 3D NEM based FPGA has provided 43.6% footprint area reduction, 37% leakage power reduction and up to 28% critical path delay reduction compared to SRAM-based FPGA with CMOS technology at 22 nm node [23]. Also, apart from experimentally verified zero leakage, NVM on-resistance values were predicted to be smaller than that of the NMOS pass transistors [24, 25]. However its mechanical switching delay is large (>1 ns) [24, 26].

This drawback can be avoided if NEM relays are used only in FPGA routing switches as they do not change their states after configuration.

- Emerging technologies, especially non-volatile memory (NVM) technologies with zero boot-up delay, real-time reconfigurability and superior energy efficiency, are being explored as possible candidate for replacement of SRAM and it include MRAM, FeRAM, PCRAM and ReRAM [27]. Generally they provide high logic density and moderate to high performance compared with existing technologies. They also have the desirable property of non-volatility and can be turned off during stand-by to save power. But their manufacturing require usually new materials and separate processes, and thus complicate the fabrication of FPGA. Y. Chen et al. [20] proposed 3D Non-Volatile FPGA ARchitecture (3D-NonFAR) using Phase Change Memory (PCM) as universal replacement for SRAM in FPGA. PCM is high performance, high density and highly scalable non-volatile memory with bit-level programmability, the feature not available in flash memory [28, 29]. Multi-Level Cell (MLC) type PCM, which has slightly slower write speed than Single-Level Cell (SLC) but higher density (16 times to that of SRAM [30]) was used for configuration memory in LBs, CBs and SBs as writing to these bits only happen at configuration time. SLCs were used in embedded RAM blocks as read/write speed of them is critical for the performance of FPGA. Area breakdown of Xilinx Virtex-4 FPGA is given in Table 2.1 and it can be seen that only replacing SRAM with PCM can significantly improve the density (by around 14%) apart from improvement due to 3D technology.

Table 2.1. Area breakdown of Xilinx Virtex-4 Platform FPGAs. In 3D-NonFAR, SRAM has been replaced by nonvolatile, highly dense and bitwise programmable PCM [20].

Logic Block (LB)		**Routing Resources (RR)**						
Logic	**Memory**	**Switch boxes**	**Inter-connects**	**Memory**	**BRAM**	**DSP**	**PowerPC**	**Clock + I/O**
4.6%	8.1%	15.1%	9.9%	20.3%	14.9%	5.3%	10.6%	11.2/%

- Among the emerging NVM technologies, ReRAM is considered to be the most promising and is made up of memristors. Memristor is scalable below 30 *nm* [31] and can be programmed within 5 *ns* at 180 *nm* technology node [32]. Its fabrication is compatible

with CMOS and its size can be as small as F^2 (F is the feature size) [33]. Because of these features, efforts have been made to use memristors in system integration [34–37]. Fabrication of memristor do not require high temperature and thus will not destroy the transistors and metal wires already fabricated below memristor layer and can be integrated with back-end-of-line (BEOL) compatible fabrication [22, 33, 38–40]. Like PCM [20], memristors have been used to replace SRAM in FPGA [36] and have shown 1.1 times and 2 times overall performance improvement before and after 3D integration. Abid et al. [41] proposed nMOS logic family using memristor which has provided 4 times density improvement over CMOS logic with similar delay and power dissipation. 3D CMOS/Nanomaterial hybrid FPGA was proposed by C. Dong et al. [21] where the transistors of interconnects were seperated from those of logic blocks and redistributed them into different dies. Nanowire crossbars and face-to-face 3D integration technology was used to provide connections between these dies. With certain power overhead brought by the large capacitance of crossbar array, this architecture has provided 2.6 times performance gain compared to the conventional FPGA architecture [21].

- J. Cong and B. Xiao introduced new memristor based reconfiguration FPGA (mrFPGA) [42]. The concept is shown in Figure 2.1. The programmable interconnects consisting of CBs and SBs were made up of only memristors and metal wires, and were fabricated over logic blocks in the same die as shown in Figure 2.2. The nonvolatile state of memristor (high resistance state (HRS) or low resistance state (LRS)) will decide the connectivity and is programmable. The detailed design of CBs and LBs in mrFPGA is shown in Figure 2.3. This structure is feasible for fabrication and CMOS compatible [39]. The total area of mrFPGA will be approximately equal to total area of logic blocks and is only 10 % to 20 % of the conventional FPGA area [5] as shown in the Table 2.2. This 2D architecture has shown 5.18 times area savings, 2.28 times speedup and 1.63 times power savings over conventional FPGA. The improvement in speed is primarily because of reduction in interconnection delay. The reduction in tile area by 5.5 times has reduced the length of wire segments and programmable interconnects by 2.35 times, contributing to reduction of 5.5 times in RC delays of wire segments. Replacement of transistors in the programmable interconnects (multiplexers and SRAM) by nonvolatile memristor and less capacitance on routing path due to capacitance shielding effect of memristor, average 40 % power savings has been achieved.

Further if several mrFPGA stacks are integrated together using 3D technology, at least 10 times density and 4.5 times speedup improvement is possible [19].

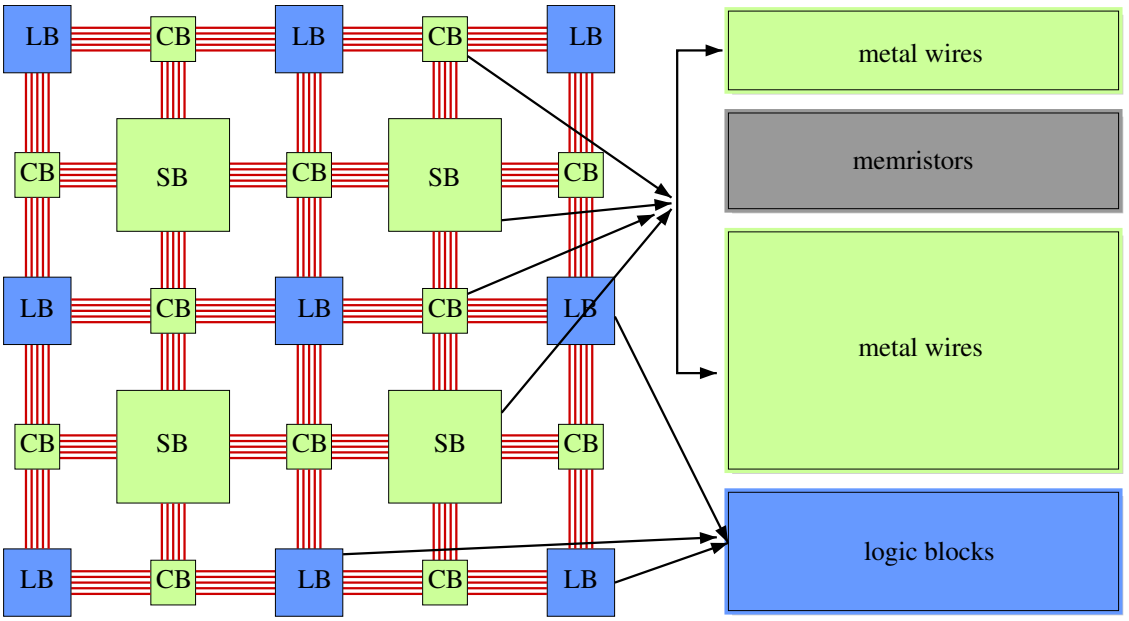


Figure 2.1. Conceptual presentation of mrFPGA architecture. Programmable interconnects (CBs and SBs) were made up of only memristors and metal wires and are placed over the logic blocks [42].

Table 2.2. Total area of mrFPGA is approximately equal to total area of logic block which is around 10% to 20% of area of conventional FPGA (area in μm^2) [42].

	Logic Block	**Switch Block + Connection Block**	**Buffer**	**Total**
Conventional FPGA (Virtex-6)	2082.6	11911	-	13993.6
mrFPGA (unbuffered)	2082.6	-	-	2082.6
mrFPGA (buffered)	2082.6	-	464.86	2547.46

- Many architectures integrating nanowire crossbar and CMOS chip have been proposed [43–46]. The major challenges in these architectures were splitting the functionality between nanowire crossbar and CMOS, and making the interconnections between CMOS and nano layers. In some architectures, demultiplexers were implemented in nanocrossbar layer to control large number of nanowires using small number of pins [43, 47, 48]. Although the work has been carried out in this direction [49–52], they present architectural challenges because simultaneously they have to configure selected nanowire junction and

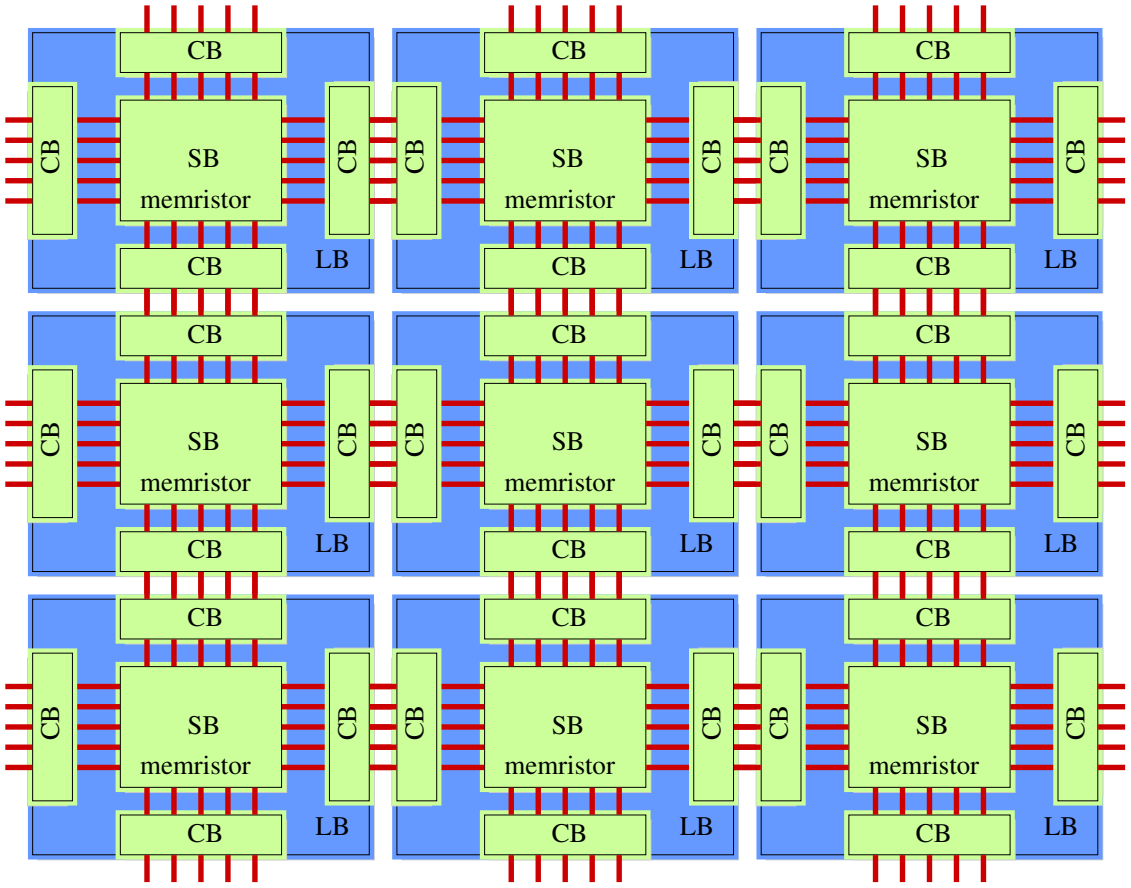


Figure 2.2. The programmable interconnects made up of CBs and SBs were placed over logic blocks in the same die in mrFPGA Architecture [42].

move the data between CMOS and nanolayers. Also demultiplexers are difficult to design without nonlinear devices. D. B. Strukov and K. K. Likharev proposed CMOL hybrid reconfigurable architecture [46, 53] where demultiplexing, logic inversion and gain functions were implemented in CMOS layer while signal routing and wired-OR logic were implemented in nanocrossbar layer. It consists of lower CMOS layer and upper two layers of parallel nanowires perpendicular to each other. Each crosspoint of nanowire has memristor as shown in Figure 2.4. Tapered nanopins were used to connect CMOS with nanowires as shown in Figure 2.5 where each connect exactly to one nanowire. This reconfigurable architecture is uniform fabric as shown in Figure 2.6 of four-transistor CMOS cells at semiconductor level and memristors layer above it. CMOL FPGA architecture provided approximately double density in comparison to conventional CMOS FPGAs with similar performance [53]. However, CMOL presents some operational and fabrication issues. The nanopins with just few nanometers diameters are difficult to fabricate. The wired-OR logic requires diode like nonlinear devices at nanowire junctions.

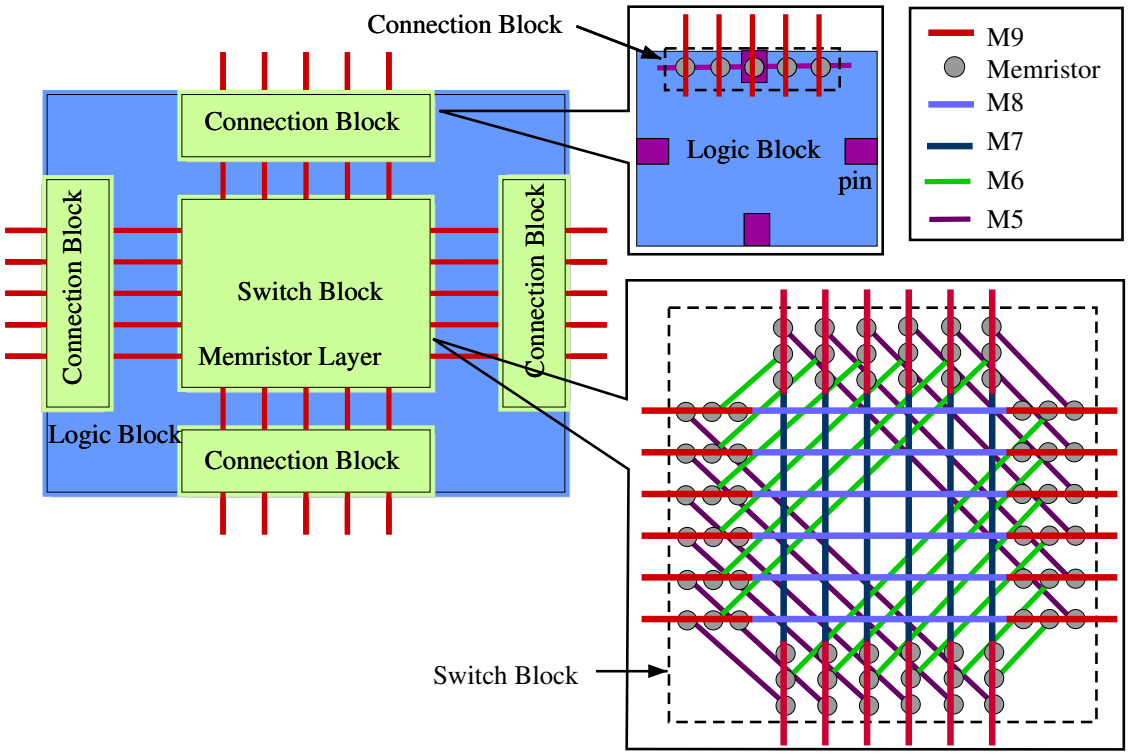


Figure 2.3. The tile of mrFPGA where CBs and SBs were placed over LB. The detailed structure of CB and SB is given. LB uses M1-M4 metal layers while interconnects use M5-M9 metal layers, memristors fabricated between M8-M9 [42].

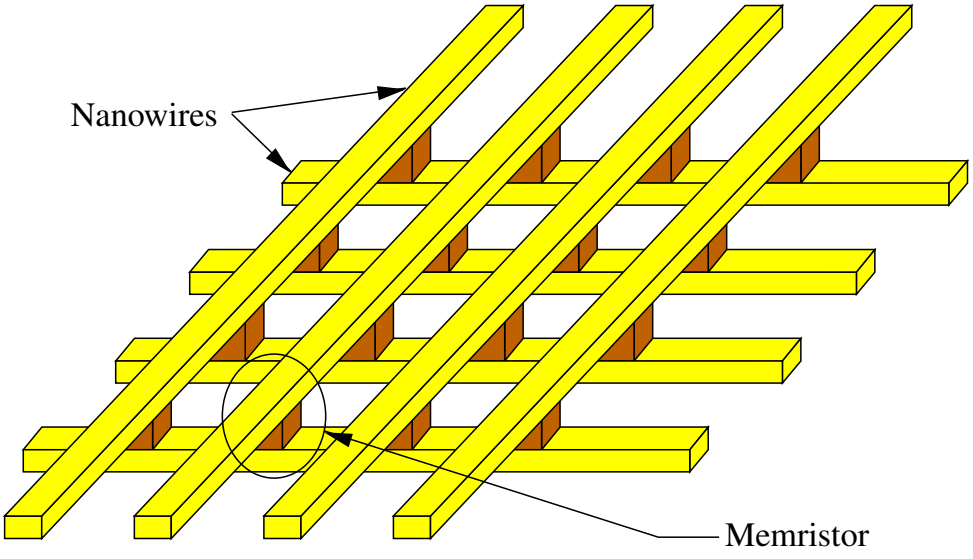


Figure 2.4. The nanocrossbar with two sets of parallel nanowires perpendicular to each other and memristor at every crosspoint

- FPNI [54] architecture is generalization of CMOL architecture. In it, all logic is implemented in CMOS and routing interconnects only in nanowires. Memristors at crosspoint were fabricated using nanoimprint technology [55]. The nanowire at CMOS

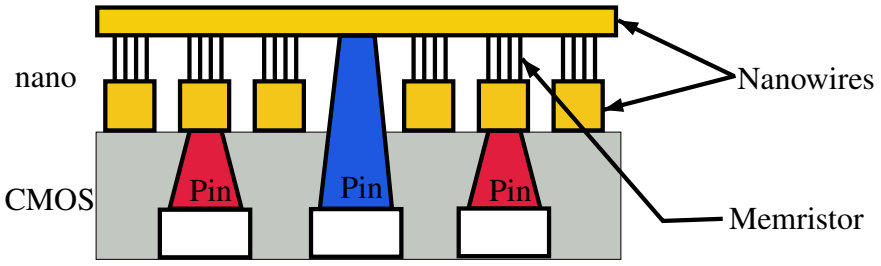


Figure 2.5. Tapered nanopins used in CMOL architecture to connect CMOS with nanowires. Each pin connects exactly to one nanowire [46, 53].

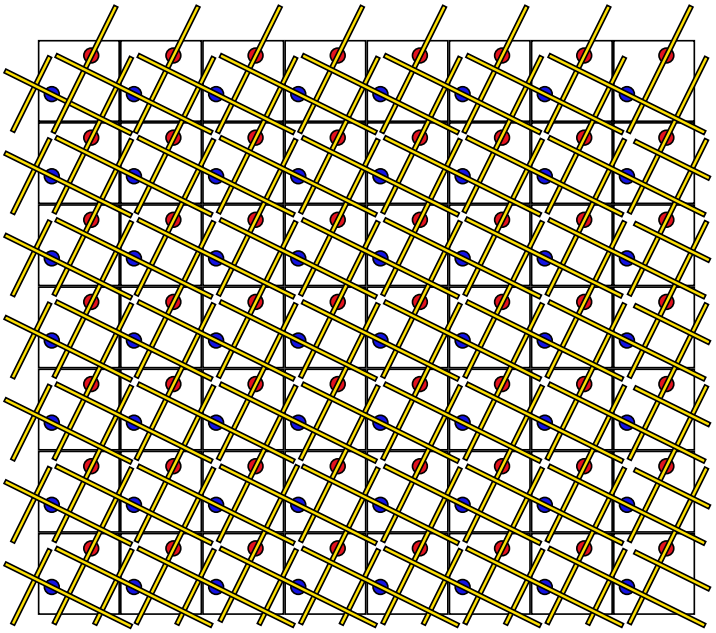


Figure 2.6. CMOL structure where CMOS layer is uniform fabric of 4 cells and nanocrossbar above it. Red and blue colored pins connect CMOS layer to nanowires [46, 53].

nanowire contact were broadened into pads such that the alignment of nanowire and CMOS pin requires accuracy no more than CMOS. The pin structure in FPNI is shown in Figure 2.7 while the top view of FPNI fabric is shown in Figure 2.8. In this fabric also one pin connects to only one nanowire. FPNI architecture is easy to fabricate, provide flexibility in the selection of nanoscale devices and it is possible to use more conservative process parameters. Its has shown 8 times to 25 times improvement in area, reduced power consumption, slightly lower clock speeds when standard benchmark circuits were compiled on it. Also in comparison to defect free chip, FPNI with 20% defective junctions and 20% broken nanowires has an effective yield of 75% without significant degradation of speed along critical path.

- K. Kim et. al [56] proposed stateful logic pipelined architecture using memristive switches.

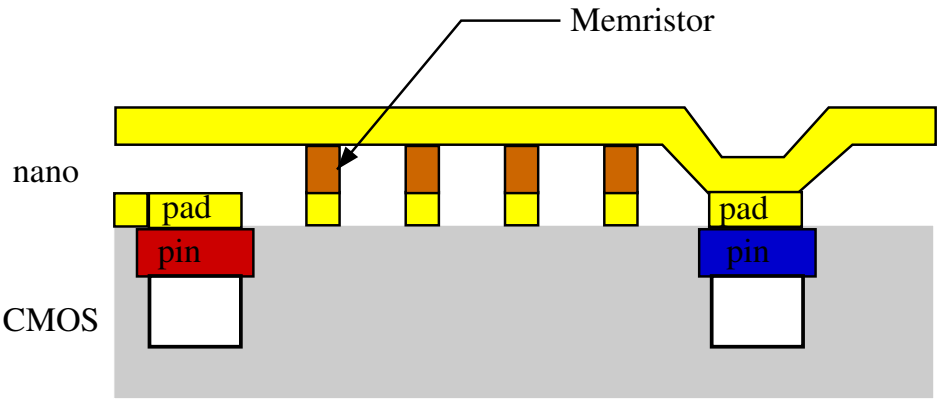


Figure 2.7. Crossbar on top of CMOS gates and buffers in FPNI. The pads are used to avoid the tapering of CMOS pins as in CMOL. CMOS level accuracy is sufficient for fabrication of nontapered pins [54].

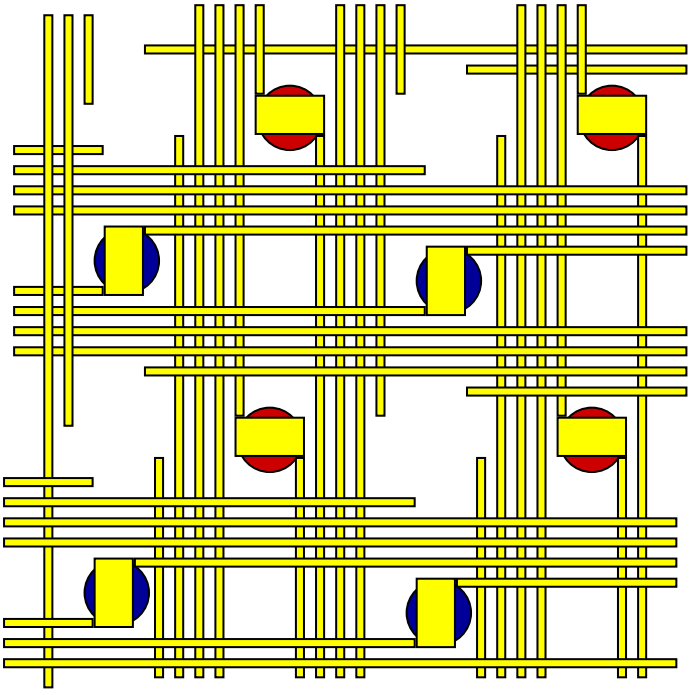


Figure 2.8. The nanowires are flattened into pads so that CMOS pins can be used to connect CMOS with nanowire without tapering. The resolution required will be of CMOS process in this FPNI architecture [54].

The architecture was mapped to FPNI fabric to produce FPSLA. All logic functions were implemented in nanocrossbar by configuring nonvolatile switches (memristors) while CMOS control switches were used to isolate stateful logic units so that multiple logic operations can be executed in parallel. Due to fan-out limitations of material implication, a basic stateful logic operation, new basic AND operation has been proposed to duplicate output. A large fan-in OR or NOR gates were implemented concurrently in nanocrossbar

to execute the given functionality. When implementing multilevel logic circuit, FPSLA can achieve logic density similar to that of FPNI. If inherent latches in FPSLA pipeline were taken into account, its density has been three times more. However the logic duplication overhead is very large.

2.3 Summary

The literature survey is summerized below.

- To develop low cost applications rapidly, the performance of FPGAs needs to be improved so that they can be preferred over ASICs.
- The main cause for performance gap between popular conventional SRAM-based FPGAs and ASICs in terms of area, speed and power consumption is programmable interconnects and extensive use of SRAM. The improvement in performance gap by changing the number of inputs to LUT and/or by changing the size of cluster in clustered FPGAs is limited. Also, with improvement in technology, handling leakage power is becoming difficult in conventional CMOS process and hence sustaining the Moore's law is becoming impossible day by day. Even though footprint size can be improved by 3D integration, the problem of leakage power remains the same in SRAM-based volatile FPGA in addition to handling of power density in 3D technology.
- High performance embedded blocks like multipliers, transceivers, block memory, processor cores etc. have been added to heterogeneous FPGAs to improve the performance of FPGA, but balancing their percentage is very difficult and if not used in application, that area of FPGA is wasted and can not be used in reconfiguration.
- Efforts have been made to improve the performance of FPGA by modifying programmable routing structure. The SRAM is replaced by devices having better performance parameters. NEM in place of SRAM in routing has shown improvement in terms of area and power dissipation but switching delay limits its usage. With the invention of memristor as nonvolatile switching device and its compatible fabrication with CMOS process, it has been used as replacement of SRAM in programmable routing as in mrFPGA and FPNI

architecture. Although this technique has shown improvement in FPGA, the use of memristor as logic element has not been explored in these architectures. The diode like behavior requirement of memristors for wired-OR logic implementation on CMOL architecture limits its use in reconfigurable architectures as such memristors are difficult to fabricate.

- Apart from its use as nonvolatile switch in programmable interconnect, memristor is used as logic element to implement universal NOR/NAND function. Any multilevel logic operation can be implemented using these universal gates. But as material implication operation with memristor can not fan-out (as output of logic operation is in the form of resistance state), overhead of AND operation to duplicate result will add up wherever the output goes as input to multiple pins as in FPSLA architecture. This limits the performance of architecture. But this kind of logic implementation enables to think of in-memory calculations, a technique against von-Neumann architecture.

At this time, any technique to implement logic functions only using nanocrossbar is not known. In this work, attempt have been made to explore other architectural possibilities using hybrid CMOS/Nanomaterial technology in order to improve the performance in terms of area, speed and power consumption. The basic blocks should be repeatable and should be generated during runtime in nanocrossbar with the help of CMOS layer to implement logic functions. At other times, the nanocrossbar can be used as resistive RAM.

Memristor is used in the design of reconfigurable architecture. Next chapter explains fundamentals of memristor, its model used in the simulation and its use as logic element.

Chapter 3

Introduction to Memristor

3.1 Introduction

Memristor is emerging as an alternative candidate for SRAM cell. Unlike SRAM, where information is stored in the form of voltage, memristor stores data in the form of resistance. Also it is possible to perform logical operations using memristors and hence there is possibility of in-memory calculations.

This chapter is organized as follows: The memristor structure, its working and ideal hysteretic characteristics are explained in the following section. In order to utilize memristors in design of memory and logic circuits sufficiently accurate, computationally efficient model should be available. Next section describes in detail the model used in simulation along with window function necessary to incorporate observed nonlinarity at boundary in fabricated memristors. The use of memristor as basic logic element is described at the end.

3.2 Memristor Fundamentals

There are three traditional fundamental passive circuit elements namely resistor (R), capacitor (C) and inductor (L), and four basic circuit variables namely the current (i), voltage (v), charge (q) and flux (ϕ). The relationship between any two circuit variables is either defined as fundamental

passive circuit element ($R = dv/di, C = dq/dv, L = d\phi/di$) or time dependent definition of circuit variables ($i = dq/dt, v = d\phi/dt$). Thus out of six one-to-one possible relationships between above four circuit variables, five were well defined. For the sake of completeness, Leon Chua in 1971 first postulated the existence of sixth relationship in terms of fourth passive basic circuit element and named it memristor, contraction for memory resistor [57]. In this seminal paper, he proved passivity, uniqueness, existence of memristor and electromagnetic interpretation of its characteristics based on Maxwell's equations. The memristance M of memristor is defined as

$$M = \frac{d\phi}{dq}. \tag{3.1}$$

The defining relationship diagram is given in Figure 3.1.

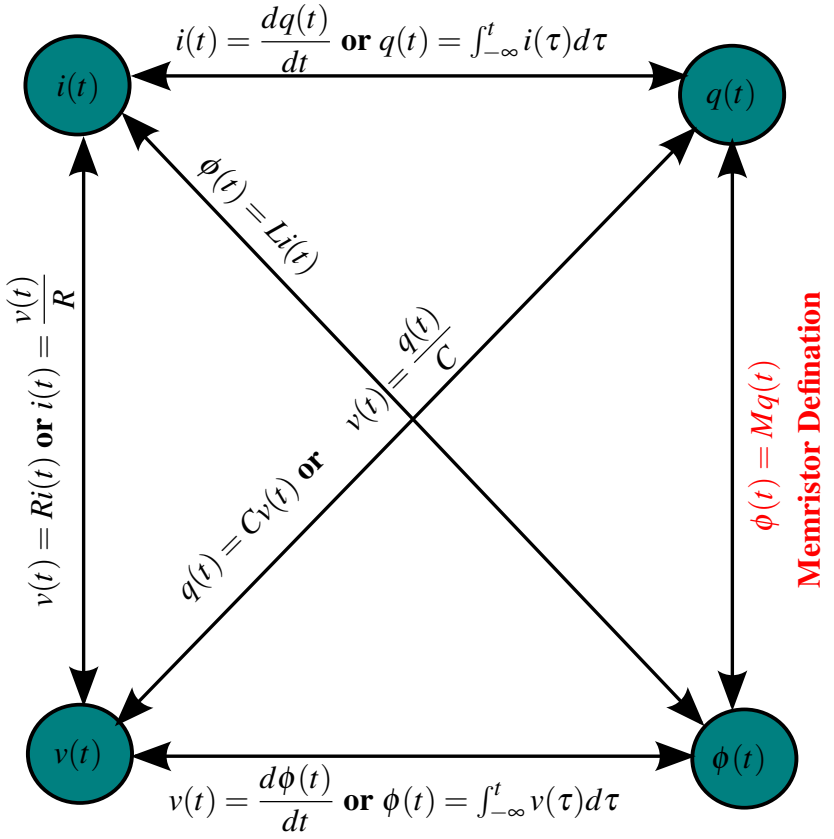


Figure 3.1. Out of six one-to-one relationships between four circuit variables viz. current (i), voltage (v), charge (q) and flux (ϕ), five were well defined in terms of three passive elements resistor (R), capacitor (C) and inductor (L), and time dependent circuit variables ($i = dq/dt, v = d\phi/dt$). The missing sixth relationship was postulated by Leon Chua in terms of memristor as fourth passive element.

To establish the link between this theoretical definition and physical realization of memristor, it took almost 40 years. Research group at HP lab successfully fabricated memristor and presented its

physical model in 2008 [58]. This memristor has two-layer nanoscale structure, the bottom layer consists of stoichiometric titanium dioxide (TiO_2), which is an electrical insulator and the upper layer also consists of TiO_2 but doped with oxygen deficiencies and thus forming TiO_{2-x} layer with high conductance [59]. The conceptual diagram is given in Figure 3.2. The TiO_2-TiO_{2-x} layers are sandwiched between two platinum plates.

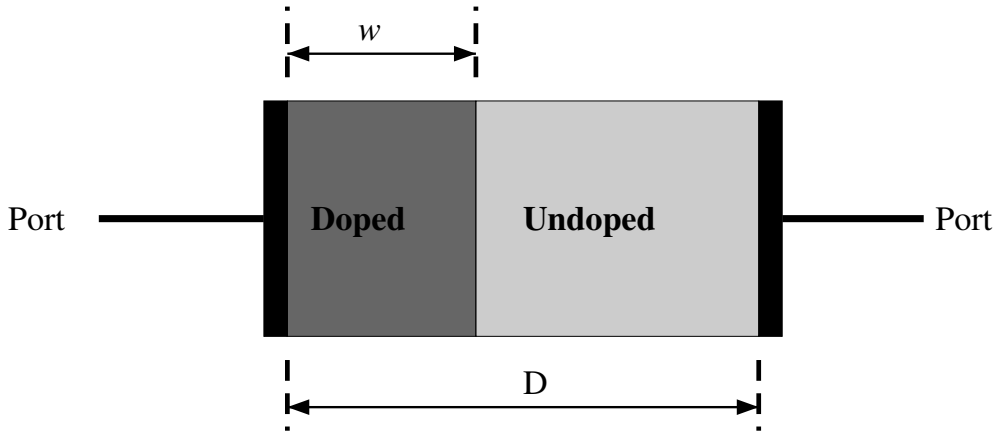


Figure 3.2. Memristor consist of memristive material sandwiched between metal plates, for example, TiO_2 sandwiched between two platinum plates. One part of TiO_2 is doped with oxygen deficiencies to form TiO_{2-x} and becomes good conductor while remaining part is TiO_2 which is insulator. The boundary between these regions move in either directions based on the polarity of applied voltage. When whole area between metal plates is occupied by doped region, the memristor will be in LRS state, otherwise it will be in HRS.

If strong enough electric field is applied across memristor, dopants can be shifted bidirectionally by changing the direction of electric field [59]. The oxygen vacancies are electron donors, so vacancies are positively charged. When a positive voltage is applied to metal plate on TiO_{2-x} side, it will repel the oxygen vacancies towards TiO_2 side, increasing the width of TiO_{2-x} region and reducing width of TiO_2 region. The application of negative voltage will have opposite effect. The position of separating area between undoped and doped region decides the memristance (resistance) of memristor. Apart from TiO_2 [60–62] such memristive effect is observed in nickel oxide [39, 63] and other materials [64–66]. The memristive effect becomes very dominant in nanometer scale than in micrometer scale, hence thickness of layers between electrodes is in nanometer range [58]. Due to nanometer thickness, most of the materials show only two distinct values of resistance: R_{HRS} or R_{LRS}, where R_{HRS} is resistance of memristor in High Resistance State (OFF state resistance) and R_{LRS} is resistance of memristor in Low Resistance State (ON state resistance). The reported values of R_{LRS} and R_{HRS} in TiO_2 material are of the order 10^2 Ω

and 10^6 Ω, respectively [67]. All these materials show characteristic hysteresis curve in their I-V plots due to change in resistance.

Memristive property arises naturally in systems for which the electronic and dopant equations of motion in a semiconductor are coupled in the presence of an applied electric field. Regardless of materials or physical mechanisms utilized, all two-terminal nonvolatile memory devices based on resistive switching effects (ReRAM), are essentially memristors [68]. The first intentional working examples of these devices, along with a simplified physics-based working model (to explain the working principle), were described in 2008 [58, 69].

Figure 3.3 shows idealized current-voltage characteristics of memristor having two distinctive resistance states, the resistance with low-resistance state (R_{LRS}) and with high-resistance state (R_{HRS}). A voltage pulse of sufficient width and amplitude higher than the threshold voltage $V_{th1,M}$ will switch the memristor to R_{LRS} state, while pulse with amplitude smaller than threshold voltage $V_{th2,M}$ will switch it to R_{HRS} state. In the notations used for threshold voltage, the letter M in suffix indicate that it is for memristor. On the other hand, for the applied voltage with amplitude between $V_{th1,M}$ and $V_{th2,M}$, the device remains in its previous state with no change of resistive state. Memristive materials show unipolar or bipolar behavior to switching [67]. Various materials showing memristive property along with the electrodes used are given in Appendix E. For unipolar memristors both $V_{th1,M}$ and $V_{th2,M}$ are positive with $V_{th2,M} < V_{th1,M}$, while for bipolar memristor $V_{th1,M}$ is positive and $V_{th2,M}$ is negative. Memristor acts as nonvolatile memory with information stored in the form of resistance.

Certain materials shown analog behavior i. e. they show gradual change in their resistance when they operated with low voltages and display a controllable hysteresis in their I-V characteristic [58, 67] e.g. tungsten oxide [70]. Other materials show discrete behavior, i.e. they show two state resistance values (R_{HRS} or R_{LRS}) when overdriven by large voltages and/or current as explained in ideal I-V characteristics. The resistance values at the two states are usually two orders of magnitude apart [39], or even up to six orders of magnitude apart [40].

The magnitude of the nonlinear charge dependent component of memristance in a semiconductor film is proportional to the inverse square of the thickness of the film (as in (A.7)), and thus becomes very dominant at the nanometer scale [58]. Hence memristor phenomena is available in nanometer-scale devices.

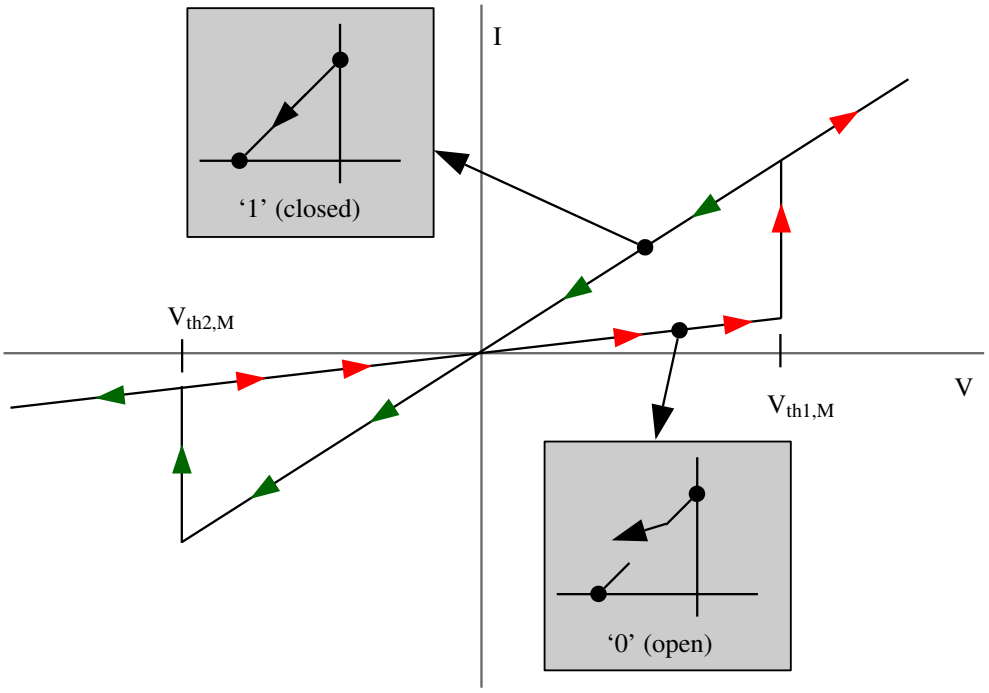


Figure 3.3. Ideal I-V characteristics of memristor showing hysteresis. When voltage across the memristor in HRS state is increased (shown by red arrows), it will switch to LRS state as the voltage across it crosses positive threshold voltage$V_{th1,M}$. If it is in LRS, state will not change upon increasing voltage across it. Similarly when voltage across the memristor in LRS state is decreased (shown by green arrows), it will switch to HRS state as the voltage across it crosses negative threshold voltage$V_{th2,M}$. If it is in HRS, state will not change upon decreasing voltage across it.

In Figure 3.2, variable w is width of doped region and D is total width of memristor. When $w = 0$ the memristor is in HRS state and when $w = D$, memristor will be in LRS state.

3.3 Memristor Models and Window Functions

A large amount of efforts have been spent in the research community to derive a suitable model of memristor that is computationally efficient, accurate and captures the nonlinear dynamics of it. It should also put insight into the physical phenomena associated with its working. The availability of accurate, general and simple models is crucial for the investigation of the nonlinear dynamics of memristor-based circuits [71, 72], to develop novel hybrid hardware architectures combining memory storage and data processing in the same physical location and at the same time [73], and to explain the memristive behavior of biological systems [74, 75]. Hewlett Packard (HP) Labs in 2008, proved the existence of non-volatile memristive behavior in nature, specifically in a Titanium

dioxide-based nano-film and presented its first physical model [58]. Since then, the industries are in search of novel materials and technologies for the manufacture of these nanodevices [76]. The memristor exhibits basic fingerprints [77, 78] : (1) under bipolar periodic signal excitation, it shows pinched hysteresis in its I-V characteristics, (2) As the excitation frequency increases, area under pinched hysteresis decreases monotonically, (3) When frequency of excitation tends to infinity, pinched hysteresis loop shrinks to a single-valued function. Among these a current-voltage pinched hysteretic loop under periodic excitation is important which is common in all kinds of memory devices [79].

Every memristor model should have bound on region of operation. For example, the linear ion drift model should work within bound [0, D]. Thus in order to bound the state variable in certain range and to include the nonlinearity and asymmetry observed in practical memristors at bounds, window function is used. The state variable derivative is multiplied by window function to limit the working interval and to add nonlinearity and asymmetry.

This section describes the basic defining model and TEAM model of memristor along with Kvatinsky's window function used in the simulation. The additional models and window functions found in literature are given in Appendix A.

The charge-flux relation uniquely defines memristor [68]. The charge q and its flux ϕ are related by

$$\phi = f(q), \tag{3.2}$$

where the charge is defined as

$$q(t) = \int_{-\infty}^{t} i(\tau)d\tau, \tag{3.3}$$

and flux is defined as

$$\phi(t) = \int_{-\infty}^{t} v(\tau)d\tau. \tag{3.4}$$

Taking the derivative on both sides of (3.2), we obtain

$$\frac{d\phi}{dt} = \frac{df(q)}{dq}\frac{dq}{dt}, \tag{3.5}$$

which leads to

$$v(t) = \frac{df(q)}{dq}i(t) = \frac{d\phi}{dq}i(t) = M(q)i(t). \tag{3.6}$$

$M(q)$ is a charge controlled memristance, defined as

$$M(q) = \left.\frac{df(q)}{dq}\right|_{q=q_Q} = \left.\frac{d\phi}{dq}\right|_{q=q_Q}. \tag{3.7}$$

Since the memristance depends on the operating point $q = q_Q$, and remains fixed when $v(t) = 0$ and $i(t) = 0$, the device can be used as nonvolatile memory. Thus, the resistance $M(q)$ is called the memristance, an acronym for memory resistance.

The most basic mathematical model of current-controlled memristor was given by Leon Chua [57] as

$$v = R(w)i, \tag{3.8}$$

and

$$\frac{dw}{dt} = i. \tag{3.9}$$

where w is the state variable of the device and R is a generalized resistance that depends upon the internal state of the device. In this case state variable w is just the charge q. In 1976, Leon Chua and Kang generalized the memristor concept to a much broader class of nonlinear dynamical systems called memristive systems, described by the equations [80] as

$$v = R(w,i)i, \tag{3.10}$$

and

$$\frac{dw}{dt} = f(w,i). \tag{3.11}$$

Current controlled memristors are represented by (3.10) and (3.11). Similarly voltage controlled memristor is given by equations

$$i = G(w,v)v, \tag{3.12}$$

and

$$\frac{dw}{dt} = f(w,v), \tag{3.13}$$

where $G(w,v)$ is conductance of memristor known as memconductance.

From (3.9), $w = q$, i.e. the state variable (w) is simply charge (q). It is difficult to physically realize this relationship and is the shortcoming of model proposed by Leon Chua.

3.3.1 TEAM Model

ThrEshold Adaptive Memristor (TEAM) model [81] is generic and simple model which fits almost all models mentioned in Appendix A with acceptable accuracy. For analytical simplification and computational efficiency, following two assumptions were made in this model : (1) Below threshold level, there is no change in the state variable. (2) The memristor current and internal state drift derivative are related by polynomial function instead of exponential function. The nonzero state derivative consist of multiplication of two degenerate functions : One is function of current and the other is function of state itself.

There are two current voltage relationships in this model. In first relationship, memristance is linearly related to state variable x and is given by [81]

$$v(t) = \left[R_{\mathrm{LRS}} + \frac{R_{\mathrm{HRS}} - R_{\mathrm{LRS}}}{x_{\mathrm{HRS}} - x_{\mathrm{LRS}}} (x - x_{\mathrm{LRS}}) \right] i(t) \tag{3.14}$$

In second relationship, memristance is exponentially related to state variable x and is given by [81]

$$v(t) = R_{\mathrm{LRS}} \exp\left(\lambda \frac{x - x_{\mathrm{LRS}}}{x_{\mathrm{HRS}} - x_{\mathrm{LRS}}} \right) i(t) \tag{3.15}$$

where $e^{\lambda} = \dfrac{R_{\mathrm{HRS}}}{R_{\mathrm{LRS}}}$ and R_{LRS} and R_{HRS} are expressed in Ω.

The state equation is given by [81]

$$\frac{dx(t)}{dt} = \begin{cases} k_{\mathrm{HRS}} \left(\dfrac{i(t)}{i_{\mathrm{HRS}}} - 1 \right)^{\alpha_{\mathrm{HRS}}} f_{\mathrm{HRS}}(x), & \text{if } 0 < i_{\mathrm{HRS}} < i, \\ 0, & \text{if } i_{\mathrm{LRS}} < i < i_{\mathrm{HRS}}, \\ k_{\mathrm{LRS}} \left(\dfrac{i(t)}{i_{\mathrm{LRS}}} - 1 \right)^{\alpha_{\mathrm{LRS}}} f_{\mathrm{LRS}}(x), & \text{if } i < i_{\mathrm{LRS}} < 0. \end{cases} \tag{3.16}$$

where x is the state variable which denotes the length of tunnel barrier, k_{HRS}, k_{LRS}, α_{HRS} and α_{LRS} are constants, i_{HRS}, i_{LRS} are current thresholds. The constant parameter k_{HRS} is a positive number, while constant parameter k_{LRS} is a negative number. The functions $f_{\mathrm{HRS}}(x)$ and $f_{\mathrm{LRS}}(x)$, dependent on state variable x, behave like window functions and constrains the state variable to

the bounds of $x \in [x_{\text{LRS}}, x_{\text{HRS}}]$. These functions are given by [81]

$$f_{\text{HRS}}(x) = \exp\left(-\exp\left(\frac{x - x_{\text{HRS}}}{w_c}\right)\right) \tag{3.17}$$

$$f_{\text{LRS}}(x) = \exp\left(-\exp\left(\frac{x_{\text{LRS}} - x}{w_c}\right)\right) \tag{3.18}$$

where x_{HRS} and x_{LRS} are positive parameters with the same dimensions as space. R_{LRS} and R_{HRS} are effective resistance at bounds x_{LRS} and x_{HRS}, respectively.

The TEAM model is accurate enough (0.2 % mean error) and computationally efficient as it improves the simulation runtime by 47.5 % [81]. It satisfies the requirements of memristive systems and convergence conditions.

The above TEAM model describe current controlled memristors. The TEAM model for voltage controlled memristors (VTEAM) is given by following relations [82].

The linear relationship between state variable x and resistance is given by [82]

$$i(t) = \left[R_{\text{LRS}} + \frac{R_{\text{HRS}} - R_{\text{LRS}}}{x_{\text{HRS}} - x_{\text{LRS}}}(x - x_{\text{LRS}})\right]^{-1} v(t) \tag{3.19}$$

The exponential relationship between state variable x and resistance is given by [82]

$$i(t) = \frac{e^{-\frac{\lambda}{x_{\text{HRS}} - x_{\text{LRS}}}(x - x_{\text{LRS}})}}{R_{\text{LRS}}} v(t), \tag{3.20}$$

where λ is the fitting parameter , and $e^{\lambda} = R_{\text{HRS}}/R_{\text{LRS}}$.

The derivative of state variable is given by [82]

$$\frac{dx(t)}{dt} = \begin{cases} k_{\text{HRS}}\left(\dfrac{v(t)}{v_{\text{HRS}}} - 1\right)^{\alpha_{\text{HRS}}} f_{\text{HRS}}(x), & \text{if } 0 < v_{\text{HRS}} < v, \\ 0, & \text{if } v_{\text{LRS}} < v < v_{\text{HRS}}, \\ k_{\text{LRS}}\left(\dfrac{v(t)}{v_{\text{LRS}}} - 1\right)^{\alpha_{\text{LRS}}} f_{\text{LRS}}(x), & \text{if } v < v_{\text{LRS}} < 0. \end{cases} \tag{3.21}$$

where k_{HRS}, k_{LRS}, α_{HRS} and α_{LRS} are constants, v_{HRS}, v_{LRS} are voltage thresholds. The constant parameter k_{HRS} is a positive number, while constant parameter k_{LRS} is a negative number. The

functions $f_{\mathrm{HRS}}(x)$ and $f_{\mathrm{LRS}}(x)$, dependent on state variable x, behave like window functions and constrains the state variable to the bounds of $x \in [x_{\mathrm{LRS}}, x_{\mathrm{HRS}}]$. The I-V characteristics of memristor using TEAM model is shown in Figure 3.4.

Similar to TEAM model, VTEAM model is generic, simple and flexible. It can characterize different voltage-controlled memristors. This model is accurate (below 1.5% in terms of the relative root-mean-square error). It is computationally efficient as compared with existing memristor models. It can be used to describe the experimental results describing different memristive technologies. It can fit to most of the published models including BCM model described above with sufficient accuracy [82]. The comparison of different models is given in Table 3.1. VTEAM model is used in the simulation carried out in this work. The parameter values for VTEAM model are given in Table 3.2.

Table 3.1. Comparison of different memristor models as given in [81]. The TEAM model is suitable for simulation of memristor based systems as it simple, generic, convergent, computationally efficient, provide sufficient accuracy and can fit to most of the available models by choosing proper fitting parameters.

Model	**Linear ion drift**	**Nonlinear ion drift**	**Simmons tunneling barrier**	**TEAM**
State variable	$0 \leqslant w \leqslant D$	$0 \leqslant x \leqslant 1$	$a_{\mathrm{HRS}} \leqslant x \leqslant a_{\mathrm{LRS}}$	$x_{\mathrm{LRS}} \leqslant x \leqslant x_{\mathrm{HRS}}$
Control mechanism	Current	Voltage	Current	Current/Voltage
I–V relation	Explicit	Explicit	Ambiguous	Explicit
Memristance relation	Explicit	Ambiguous	Ambiguous	Explicit
Generic	No	No	No	Yes
Accuracy	Lowest	Low accuracy	Highest	Sufficient
Threshold exists	No	No	Yes	Yes

3.3.2 Kvatinsky's Window

This window function is used in TEAM model to fit the behavior of Simmons tunnel barrier model. Two window functions were defined, one for ON switching and other for OFF switching.

Table 3.2. Values of parameters used for the simulation using VTEAM model.

Parameter	Numerical Value	Parameter	Numerical Value
R_{LRS}	100 Ω	α_{LRS}	3
R_{HRS}	1 $M\Omega$	k_{HRS}	$-0.8\ pm/s$
V_{HRS}	$-0.7\ V$	k_{LRS}	$0.8\ pm/s$
V_{LRS}	$0.7\ V$	$D_{\text{Memristor}}$	$3\ nm$
α_{HRS}	3	L_{CMOS}	$65\ nm$

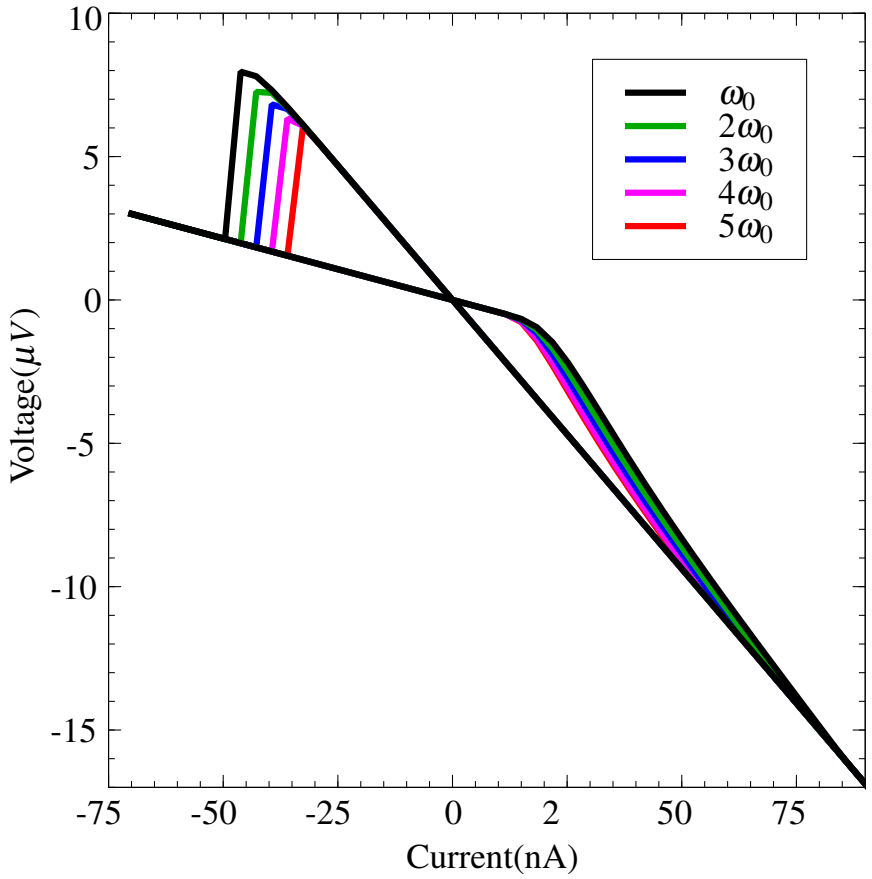


Figure 3.4. I-V characteristics of memristor for different excitation frequencies using TEAM model. It shows pinched hysteresis and area under the lobe decreases with increase in frequency of excitation signal.

This enables asymmetric switching as in Simmons tunnel barrier model. The window functions are given as follows [81].

$$f_{\text{HRS}}(x) = \exp\left(-\exp\left(\frac{x - x_{\text{HRS}}}{w_c}\right)\right) \tag{3.22}$$

$$f_{\text{LRS}}(x) = \exp\left(-\exp\left(\frac{x_{\text{LRS}} - x}{w_c}\right)\right) \tag{3.23}$$

where x_{HRS} and x_{LRS} are positive parameters with same dimensions as space. R_{LRS} and R_{HRS} are effective resistance at bounds x_{LRS} and x_{HRS}, respectively.

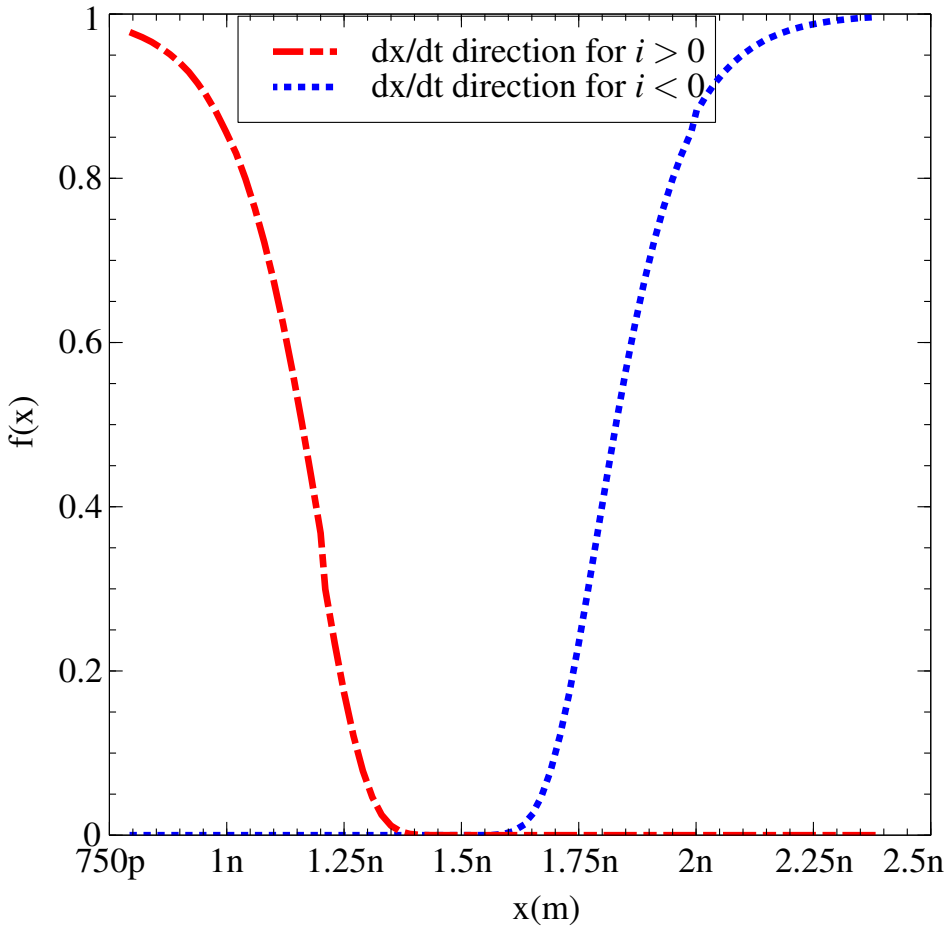


Figure 3.5. Plot for Kvatinsky's window function given by (3.22) and (3.23). The values of parameters used in this plot are $w_c = 0.107$ *nm*, $x_{LRS} = 1.2$ *nm*, $x_{HRS} = 1.8$ *nm*. The parameter D is assumed to be 3 *nm*.

3.4 Memristor as Logic Element

The International Technology Roadmap for Semiconductors (ITRS) 2009 report [83] have challenged the computing research community to find new physical state variables (other than charge or voltage), new devices, and new architectures that offer memory and logic functions [84–86] beyond those available with standard transistors. This section describe memristor as logic element to perform imply logic operation and how imply logic can be used to perform universal logic operations like NOR and NAND.

3.4.1 Implication logic

In 1910, Whitehead and Russell described four fundamental logic operations in their book Principia Mathematica [87]. Out of these, three were chosen by Shannon for his work and called them as Boolean logic operations. Fourth operation was called as 'material implication' (IMPLY) by Russell. If p and q are two binary variables, then p IMPLY q is logically equivalent to $\overline{p}+q$. p IMPLY q (iff p then q) is denoted as p→q. The corresponding truth table is given in Table 3.3. Implication logic can be naturally implemented using memristors and resistor [88]. It becomes 'stateful' logic as memristor store logic values as well as perform logical operations in the form of resistance as a state variable. Circuit diagram of IMPLY logic using memristor is given in Figure 3.6. The resistance R_G is selected such that $R_{LRS} < R_G < R_{HRS}$ and $V_{COND} < V_{SET}$. The detailed analysis of IMPLY logic using memristors is given in Appendix B.

Table 3.3. IMPLY gate truth table. p and q are states of memristors P and Q, respectively. Logic '0' is R_{HRS} while logic '1' is R_{LRS}. The destination memristor Q toggles only if both inputs are logic '0'Àlso only logic '0' to logic '1' transition is possible.

p	**q**	**p → q**
0	0	1
0	1	1
1	0	0
1	1	1

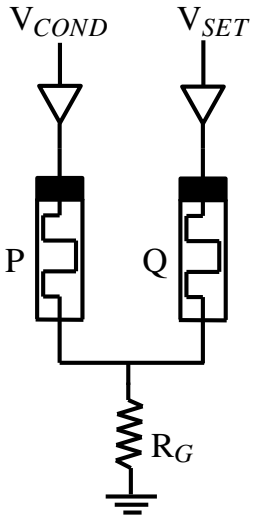


Figure 3.6. Imply gate using memristors. States of memristors P and Q are p and q, respectively.

In IMPLY logic implementation using memristors, the inputs and outputs are in the form of resistance. Let p and q be the state of memristors P and Q, respectively. Logic '0' refers to

R_{HRS} and logic '1' refers to R_{LRS}. The inputs are written to the memristors P and Q (Figure 3.6) and the final output is stored in the memristor Q. To implement this operation, voltages V_{COND} and V_{SET} are applied simultaneously to memristors P and Q, respectively. The simulation results of IMPLY gate are given in Figure 3.7. The IMPLY logic is destructive operation as can be seen from simulation results (specifically last case when input='11').

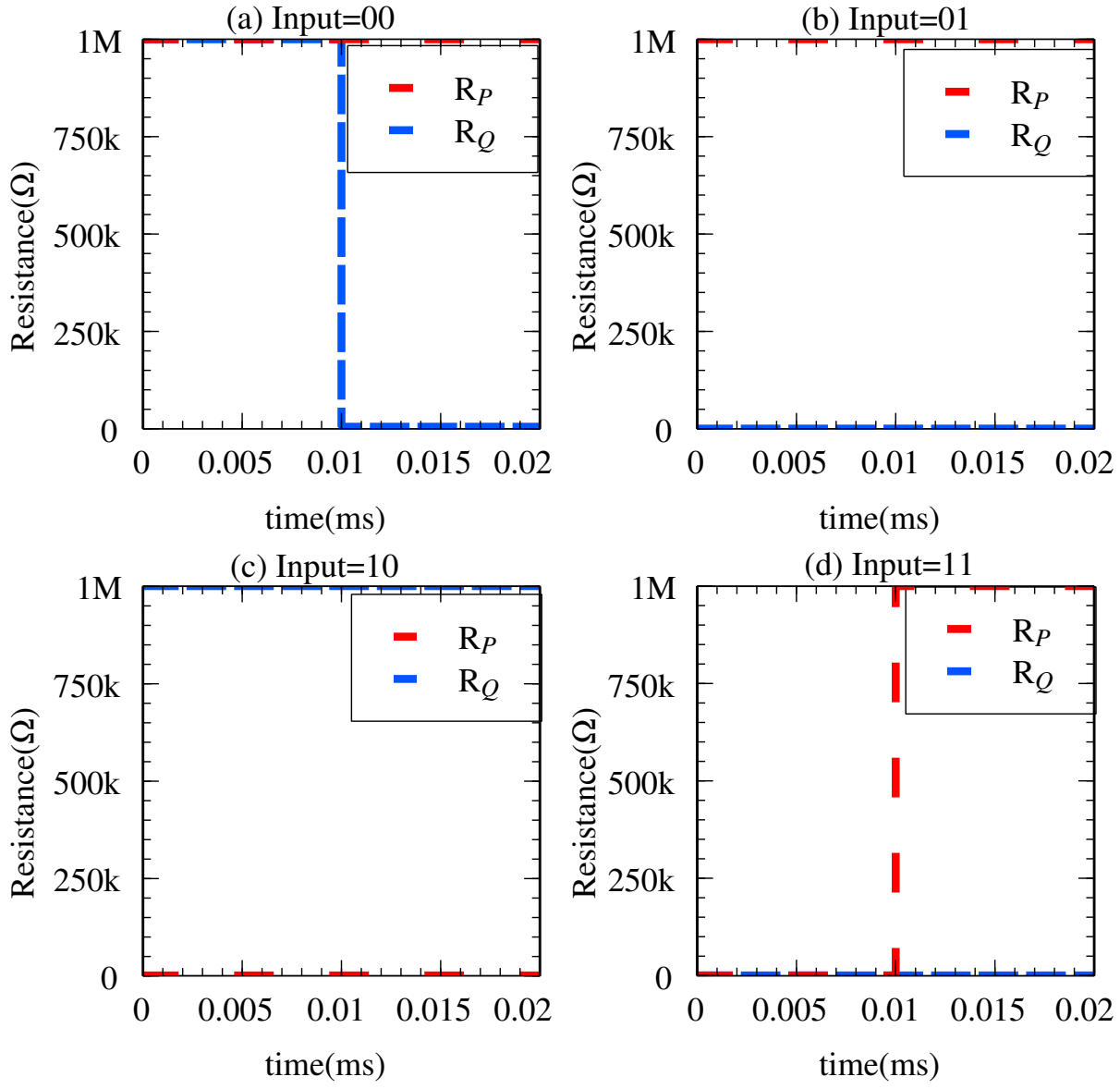


Figure 3.7. Simulation of IMPLY logic. Implication operation is carried out at t=0.01 ms (V_{SET} and V_{COND} is applied). The result is stored in Q memristor in the form of resistance. (a) Input='00'(R_P=1 MΩ, R_Q=1 MΩ) before operation; output='1'(R_Q=100 Ω) after operation. (b) Input='01'(R_P=1 MΩ, R_Q=100 Ω) before operation; output='1'(R_Q=100 Ω) after operation. (c) Input='10'(R_P=100 Ω, R_Q=1 MΩ) before operation; output='0'(R_Q=1 MΩ) after operation. (d) Input='11'(R_P=100 Ω, R_Q=100 Ω) before operation; output='1'(R_Q=100 Ω) after operation.

IMPLY together with FALSE operation form computationally complete set of operations. Universal NAND and NOR logic operations can be implemented using IMPLY and FALSE logic. The output memristor (in this case, it is also one of the input memristors) toggles only if both the inputs are logic '0'. Also, output toggles from only '0' to '1' and its '1' to '0' transition is not possible. These observations are used to implement NAND and NOR gates using memristors.

3.4.2 Stateful NAND Logic using Memristors

NAND operation is executed in three sequential steps [88]. The circuit diagram for 2-input NAND gate using memristors is shown in Figure 3.8, where P and Q are input memristors and S is output memristor. The three sequential steps are given in Table 3.4 with the assumption that inputs are already available in the memristors P and Q. In step 1, voltage V_{CLEAR} (as in Table 3.3) is applied to destination memristor S to implement FALSE operation ($R_{\mathrm{S}} = R_{\mathrm{HRS}}$ or logically $s = 0$). In step 2, voltages V_{COND} and V_{SET} are applied simultaneously to memristors P and S, respectively to implement $s = p \rightarrow s$ which is logically equivalent to $s = \overline{p} + s = \overline{p} + 0 = \overline{p}$. In the final step 3, voltages V_{COND} and V_{SET} are applied simultaneously to memristors Q and S, respectively to implement $s = q \rightarrow s$ which is logically equivalent to $s = \overline{q} + s = \overline{q} + \overline{p} = \overline{pq}$. Thus final output in memristor S will be NAND of logical content of memristors P and Q.

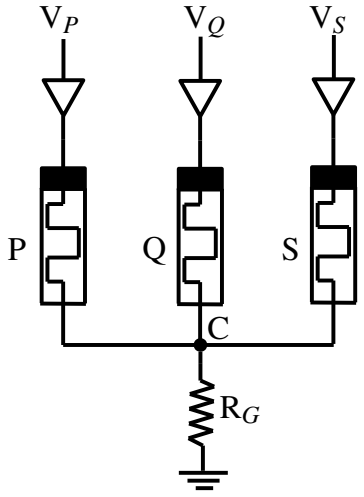


Figure 3.8. Stateful NAND gate using material implication. P and Q are input memristors ans S is output memristor.

Table 3.4. Steps to implement stateful NAND logic. Step 1 is FALSE operation (s=0). Step 2 performs $s = p \rightarrow s$ (i.e. $s = \overline{p} + s$). Step 3 is $s = q \rightarrow s$ (i.e. $s = \overline{q} + s = \overline{q} + \overline{p} + s = \overline{pq}$).

	$\mathbf{V_P}$	$\mathbf{V_Q}$	$\mathbf{V_S}$
Step 1			V_{CLEAR}
Step 2	V_{COND}		V_{SET}
Step 3		V_{COND}	V_{SET}

3.4.3 Stateful NOR Logic using Memristors

NOR operation can be executed in two sequential steps. The circuit will be same as for NAND gate (Figure 3.8) with the two steps as given in Table 3.6. In step 1, voltage V_{CLEAR} is applied to destination memristor S to implement FALSE operation ($R_{\mathrm{S}} = R_{\mathrm{HRS}}$ or logically $s = 0$). In step 2, voltages V_{COND} and V_{SET} are applied simultaneously to memristors P and S. The effective voltage at point C in Figure 3.8 in step 2 is given by

$$V_C = V_{\mathrm{COND}} \frac{R_{\mathrm{HRS}} || R_{\mathrm{G}}}{(R_{\mathrm{P}} || R_{\mathrm{Q}}) + (R_{\mathrm{HRS}} || R_{\mathrm{G}})} + V_{\mathrm{SET}} \frac{R_{\mathrm{P}} || R_{\mathrm{Q}} || R_{\mathrm{G}}}{R_{\mathrm{HRS}} + (R_{\mathrm{P}} || R_{\mathrm{Q}})} \tag{3.24}$$

Table 3.5. Truth table of stateful NOR logic gate in terms of resistance of memristors involved. The destination memristor S is initialized to R_{HRS} in the first step for NOR execution.

	$\mathbf{R_P}$	$\mathbf{R_Q}$	$\mathbf{R_S}$	**Resultant $\mathbf{R_S}$**
Case 1	R_{HRS}	R_{HRS}	R_{HRS}	R_{LRS}
Case 2	R_{HRS}	R_{LRS}	R_{HRS}	R_{HRS}
Case 3	R_{LRS}	R_{HRS}	R_{HRS}	R_{HRS}
Case 4	R_{LRS}	R_{LRS}	R_{HRS}	R_{HRS}

Table 3.6. Steps to implement stateful NOR logic. Step 1 is FALSE operation ($R_{\mathrm{S}} = R_{\mathrm{HRS}}$ or logically $s = 0$). Step 2 performs material implication on equivalent resistance of $R_{\mathrm{P}} || R_{\mathrm{Q}}$ and $R_{\mathrm{S}} = R_{\mathrm{HRS}}$ with memristor S as destination.

	$\mathbf{V_P}$	$\mathbf{V_Q}$	$\mathbf{V_S}$
Step 1			$\mathrm{V_{CLEAR}}$
Step 2	$\mathrm{V_{COND}}$	$\mathrm{V_{COND}}$	$\mathrm{V_{SET}}$

For case 1 in Table 3.5, $R_{\mathrm{P}} || R_{\mathrm{Q}} = R_{\mathrm{HRS}}/2$ and with condition $R_{\mathrm{G}} << R_{\mathrm{HRS}}$, (3.24) becomes

$$V_C = V_{\mathrm{COND}} \frac{R_{\mathrm{G}}}{\frac{R_{\mathrm{HRS}}}{2} + R_{\mathrm{G}}} + V_{\mathrm{SET}} \frac{R_{\mathrm{G}}}{R_{\mathrm{HRS}} + \frac{R_{\mathrm{HRS}}}{2}} \approx 0 \tag{3.25}$$

Hence the voltage drop across the memristor S is $V_{\mathrm{SET}} - V_C \approx V_{\mathrm{SET}}$ which is more than the threshold voltage to turn it LRS state.

For case 2 and 3 in Table 3.5, $R_{\mathrm{P}} || R_{\mathrm{Q}} \approx R_{\mathrm{LRS}}$ and with condition $R_{\mathrm{LRS}} << R_{\mathrm{G}}$, (3.24) becomes

$$V_C = V_{\mathrm{COND}} \frac{R_{\mathrm{G}}}{R_{\mathrm{LRS}} + R_{\mathrm{G}}} + V_{\mathrm{SET}} \frac{R_{\mathrm{LRS}}}{R_{\mathrm{HRS}} + R_{\mathrm{LRS}}} \approx V_{\mathrm{COND}} \tag{3.26}$$

The voltage drop across memristor S is approximately equal to $V_{\mathrm{SET}} - V_{\mathrm{COND}}$ which is less than threshold voltage of memristor and hence memristor S will remain in HRS state.

For case 4 in Table 3.5, $R_{\mathrm{P}}||R_{\mathrm{Q}} = R_{\mathrm{LRS}}/2$. Hence (3.24) becomes

$$V_C = V_{\mathrm{COND}} \frac{R_{\mathrm{G}}}{\frac{R_{\mathrm{LRS}}}{2} + R_{\mathrm{G}}} + V_{\mathrm{SET}} \frac{\frac{R_{\mathrm{LRS}}}{2}}{R_{\mathrm{HRS}} + \frac{R_{\mathrm{LRS}}}{2}} \approx V_{\mathrm{COND}} \tag{3.27}$$

Thus in this case also memristor S will remain in HRS state.

3.5 Other Applications of Memristors

Fabrication of very high density memories using semiconductor or insulator thin film have been reported in the literature [89–93]. It has been shown in [94] that memristor-MOS technology (MMOST) is capable of outperforming CMOS implementations in terms of power and size. The scaling potential of Resistive Random Access Memory (ReRAM) and its capacity of storing multiple bits of information per cell enables the exploitation of the technology in signal processing [95], programmable circuits [96], and neuromorphic circuits [97–99]. As an example, ReRAM has been studied as a leading candidate in nonvolatile memory applications [100, 101] as a replacement for the flash memory technology due to attributes such as high density [91, 102], fast operation [103, 104], low power [103], and CMOS compatibility [104]. Digital memory devices have been achieved in various material systems [100, 102, 105, 106], some of which also show multilevel switching [103, 104, 107, 108], allowing further increase of storage density. Recently, memristors have also been studied in biologically inspired neuromorphic circuits [109–112]. Important synaptic learning rules such as spike-timing dependent plasticity (STDP) have already been demonstrated [110–112].

3.6 Summary

The fundamental aspects of memristor are described in this chapter. Universally acceptable and mature model of memristor is not available yet. To carry out simulation of memristor

based systems, TEAM model is chosen as it is simple, generic, sufficiently accurate, convergent, computationally efficient and can fit to almost all models available in the literature. The use of memristor as logic element in imply operation and implementation of stateful NOR and NAND gate by making use of imply logic is explained.

Memristors are seldom used as standalone devices. They are part of nanocrossbar arrays made up of two sets of nanowires perpendicular to each other. At every crosspoint, there is a memristor. In order to implement the stateful logic gates on such crossbar, write, read and evaluate operations need to be performed. Memristors in crossbar are difficult to isolate. Next chapter gives the complete analysis of implementation of these operations on memristive crossbar in order to find out the effect on other memristors of crossbar that are not involved in a logic implementation.

Chapter 4

Logic Implementation on Memristive Crossbar Array

4.1 Introduction

Memristors are passive circuit elements and cannot supply energy on their own in a circuit. Therefore they need to be integrated with active circuit elements such as transistors in circuits to implement any logic function. Hybrid circuits consisting of memristors and transistors can deliver functionality with fewer components and thus save chip area and power consumption. The crossbar array formed by two sets of parallel metal wires crossing over each other perpendicularly with memristor at each crosspoint of metal wires is ideal platform for use in such hybrid circuits. Because of their ultimate scaling potential and their ability to allow fusion of logic and memory, crossbar array are promising candidates for development of applications with low energy consumption [93, 113, 114].

Many nanoscale architectures with functions such as memory and logic system has been implemented on crossbars [89, 115–121]. Crossbars are scalable down to the molecular size [122, 123], regular in structure, can be reconfigured to tolerate defects in the circuit [124–126] and can be fabricated inexpensively with nanoimprint lithography [89, 91, 127] because of structural simplicity. Ultrahigh density memory and crossbar latches have been previously

demonstrated [128]. J. Borghetti et al. [129] illustrated compound logic operation implementation on memristor/transistor based hybrid structure where memristor crossbars were fabricated at 40 *nm* half-pitch using nanoimprint lithography. Memristor crossbars were used to perform logic functions, as a routing fabric for interconnecting the transistors and for storing the information. The result of logic operation carried out in crossbar can be routed back onto a target memristor inside the array, and hence the self-programmed logic array design is possible. Thus the same memristor in a crossbar circuit can be configured to act as logic, component for signal routing and memory, and the circuit can even reconfigure itself. Memristor can be used in wired logic and routing in a configured crossbar [115, 118, 119, 121, 125], reconfigurable architectures [46, 54, 130], memristor based logic [131] and synaptic circuits [109, 132].

In order to implement stateful NOR gate on crossbar, three basic operations are required : (1) Writing the inputs to memristors in crossbar (in the form of resistance states). (2) Performing the logic evaluation (using material implication as basic step). (3) Reading the result of operation. The memristors not involved in logic implementation can not be isolated. While performing write and evaluate operations, the contents of memristors not involved in logic operation may get altered and while reading the state of destination memristor in logic operation, the state may be interpreted erroneously due to false paths arround destination memristor. This is called sneak path problem. The objectives of this chapter are to analyze write, read and evaluate opetations to be performed on memristive crossbar in order to implement stateful NOR gate on it, to find the way to maintain the integrity of memristors not involved in logic implementation and to find the limitation on size of crossbar in order to read the state of memristor without error.

This chapter is organized as follows. The general structure of passive memristive crossbar is described in the first section followed by description of sneak path problem in it. Different schemes for write operation found in literature, read and evaluate operations are described in the next section and thorough analysis is carried out for these operations by equivalent resistive network method to check the feasibility of implementing these operations on memristive crossbar. Owing to the limitations on size of crossbar found through analysis, the implementation of stateful NOR gate on specialized crossbar architecture is given at the end.

4.2 Passive Memristive Crossbar Array

The passive memristive crossbar is made up of two sets parallel nanowires, one set being perpendicular to other and each crosspoint of nanowires has memristor as shown in Figure 2.4. It consists of *n* columns known as bit lines and *m* rows known as word lines as shown in Figure 4.1. Every word line is connected to every bit line through memristor. The memristor at any crosspoint is accessed through its word line and bit line.

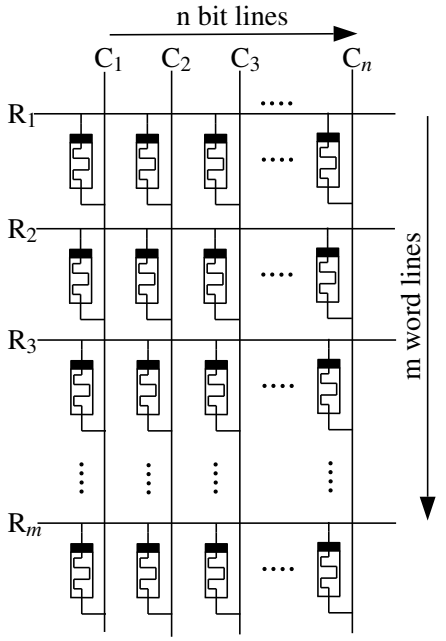


Figure 4.1. Memristive crossbar of size $m \times n$. The rows are considered as word lines while columns are considered as bit lines.

4.3 Sneak Path Problem

The difficulty in accessing individual device in a crossbar and doing correct operation on it is called sneak path problem. The memristors in one row have common top electrode and hence are connected each other. Similarly memristors in one column are connected each other by the common bottom electrode. The concept is illustrated in Figure 4.2. The memristor in HRS is shown in red color and surrounded by memristors in LRSs. This is one of the situations. For example, in order to read the state of memristor in red, voltage V_{READ} is applied to top electrode of memristor shown in red color and current is measured through it (I_M) but at bottom electrode.

In this case false path for current is formed and current I_{sneak} is added to I_M to form I_{read} at bottom electrode. Ideally, I_{read} should be equal to I_M to identify the state (LRS or HRS) of selected memristor correctly but in this case

$$I_{read} = I_M + I_{sneak1} + I_{sneak2} + \ldots + I_{sneakn}; \tag{4.1}$$

where I_{sneak1}, I_{sneak2},..., I_{sneakn} are currents through different false paths around the target memristor. The number of sneak paths depend on distribution of states of memristors around target memristor, the worst being all surrounding memristors in LRS. The value of each sneak current component depends on the physical length it has to travel from top electrode to bottom electrode used in the measurement and number of LRS memristors in its path. The problem is in identifying HRS of memristor as the resultant current I_{read} in (4.1) might be large enough to falsely identify the state of selected memristor as LRS instead of HRS. This is an inherent disadvantage of passive crossbar arrays. They significantly restrict size of a crossbar array (maximum numbers of rows and columns) because the current and voltage drop over the addressed memristor is strongly function of multiple parallel sneak paths [133, 134]. Also, as the overall resistance of crossbar is function of number of LRS and HRS states of memristors, the power consumption increases as number of memristors in LRS increases.

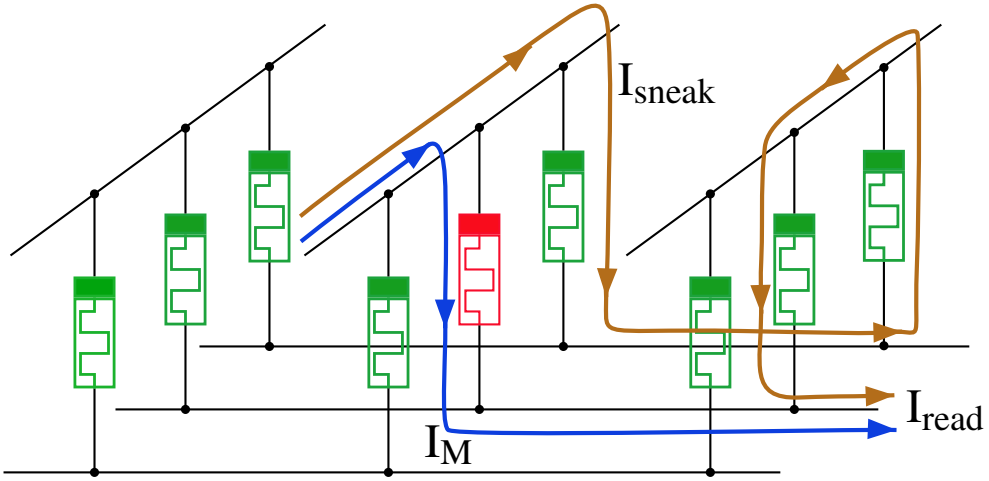


Figure 4.2. Sneak path problem in a memristive crossbar architecture. Only the addressed element in the centre of the crossbar array is in the HRS (red), all surrounding elements are in the LRS (green). This is one possible worst case pattern. When reading, the current flows through the addressed memristor (I_M), but also a significant current will flow through the neighbouring elements I_{sneak}. From the periphery it is not possible to distinguish between both currents and it might lead to a wrong interpretation of the stored bit[135].

Various methods have been tested to deal with sneak path problem. The use of rectifying elements in series with memristor could solve the sneak path problem, but materials with comparable scalability with memristor has not found yet [136–138]. Also some materials showing nonlinear switching characteristics such as Pt/TiO_2/Pt [139, 140] or BPDN-DT (bipyridyl-dinitro oligophenylene-ethynylene dithiol) [141, 142] has been explored, but nonlinearity is not sufficient for proper operation of large crossbar arrays. Other methods include restricting the size of crossbar array and using crossbars with interrupted electrodes as in CMOL and FPNI architecture.

4.4 Analysis of Operations on Memristive Crossbar

In order to implement stateful NOR gate on memristive crossbar, following three oprations need to be perormed in sequence. (1) Write inputs on which NOR logic is to be performed to memristors in a crossbar in the form of resistance states. LRS is logic '1' while HRS is logic '0'. (2) Execute stateful NOR using material implication as basic operation. (3) Read the state of destination memristor in order to know the result of stateful NOR operation on inputs. In this section, write schemes found in literature, read operation and evaluate operation are described and analyzed. Ideally write and evaluate operation should not alter the contents of memristors not taking part in stateful NOR operation and read operation should be able to identify the correct state of memristor for any array size and for any states of surrounding memristors. But this is not possible because of sneak path problem and hence effect of these operations on other memristors needs to be investigated through analysis. The challenge in analyzing the memristive crossbar lies in the fact that the overall characteristics of crossbar is dependent on the data pattern stored in it.

The memristors may be in HRS or LRS. Their resistance will be shown by R_{HRS} and R_{LRS}, respectively. The symbol of resistance equivalent for memristor with dot to represent the orientation of memristor in memristive crossbar is shown in Figure 4.3. With this notation, if positive voltage greater then $V_{th1,M}$ is applied to side representing dot, memristor will change its state from HRS to LRS.

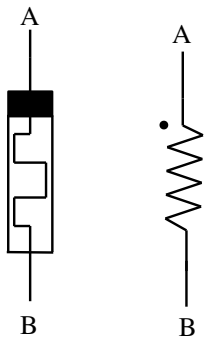


Figure 4.3. The symbol for resistance equivalent for memristor. The dot on one side of symbol indicate the orientation of memristor in equivalent circuits of memristive crossbar.

4.4.1 Write Operation

In order to write LRS into selected memristor in a crossbar, voltage/s should be applied to its top and bottom electrode such that effective voltage across memristor is greater than positive threshold voltage $V_{th1,M}$. To write HRS, the effective voltage across it should be less than negative threshold voltage $V_{th2,M}$. Different write schemes are described and analyzed below. The notations used in the analysis of write operation on memristive crossbar are given in Table 4.1.

Table 4.1. Notations used in the analysis of memristive crossbar for write operation.

Notation	Meaning	Voltage notation
R_{SELECT}	Resistance of selected memristor.	V_{SEL}
R_{WORD}	Resistance of each memristor on selected word line except selected memristor.	V_{WORD}
R_{BIT}	Resistance of each memristor on selected bit line except selected memristor.	V_{BIT}
R_{NSWB}	Resistance of each memristor on unselected word lines and bit lines.	V_{NSWB}

4.4.1.1 Floating Write Scheme

In this scheme, word line of memristor to be written is driven with voltage V_{WRITE} while its bit line is grounded. Other word lines and bit lines are kept floating. The scheme and its resistive equivalent circuit are shown in Figures 4.4 and 4.5, respectively. It can be seen from Figure 4.5 that the voltage across selected memristor is V_{WRITE}. The circuit is complex and it is difficult to find out voltage across each of the remaining memristors as it depends on the state of each

memristor in the circuit. Each memristor can be in LRS or HRS before the application of V_{WRITE} voltage. If V_{WRITE}=V_W and R_{SEL}=R_{HRS} before the application of V_{WRITE}, then it should switch to LRS state i.e. R_{SEL}=R_{LRS} after application of write voltage. If R_{SEL}=R_{LRS} before application of voltage, it should not change its state. Similarly if V_{WRITE}=$-V_W$ and R_{SEL}=R_{LRS} before the application of V_{WRITE}, then it should switch to HRS i.e. R_{SEL}=R_{HRS} after application of write voltage. If R_{SEL}=R_{HRS} before application of voltage, it should not change its state. All the

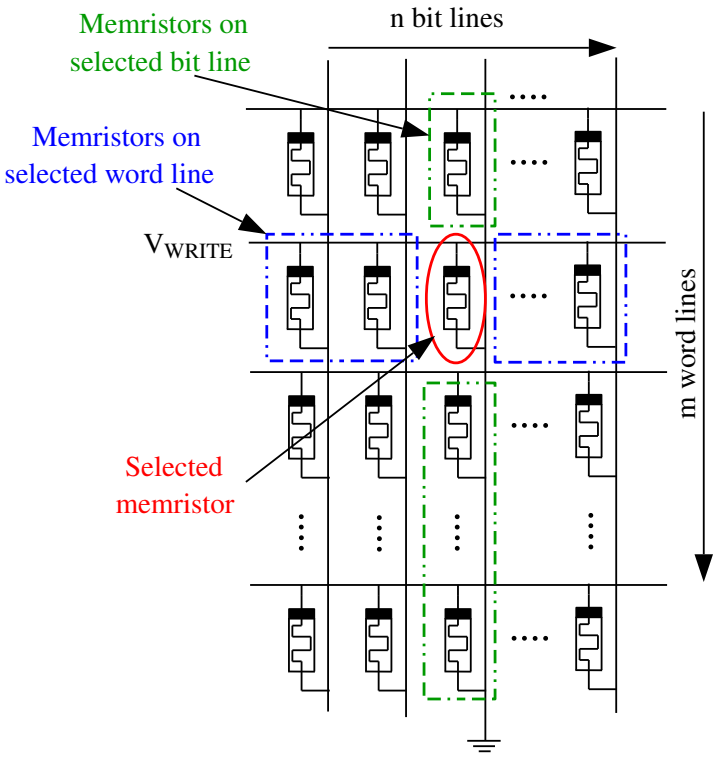


Figure 4.4. The floating write scheme for memristive crossbar. The selected word line is driven by write voltage while selected bit line is grounded. Remaining word lines and bit lines are kept floating.

remaining memristors should not change their states. If V_{WRITE}=$\pm V_W$, then after application of V_{WRITE} voltage

$$V_{th2,\,M} < V_{WORD} < V_{th1,\,M}, \tag{4.2}$$

$$V_{th2,\,M} < V_{BIT} < V_{th1,\,M}, \tag{4.3}$$

and

$$V_{th2,\,M} < V_{NSWB} < V_{th1,\,M}, \tag{4.4}$$

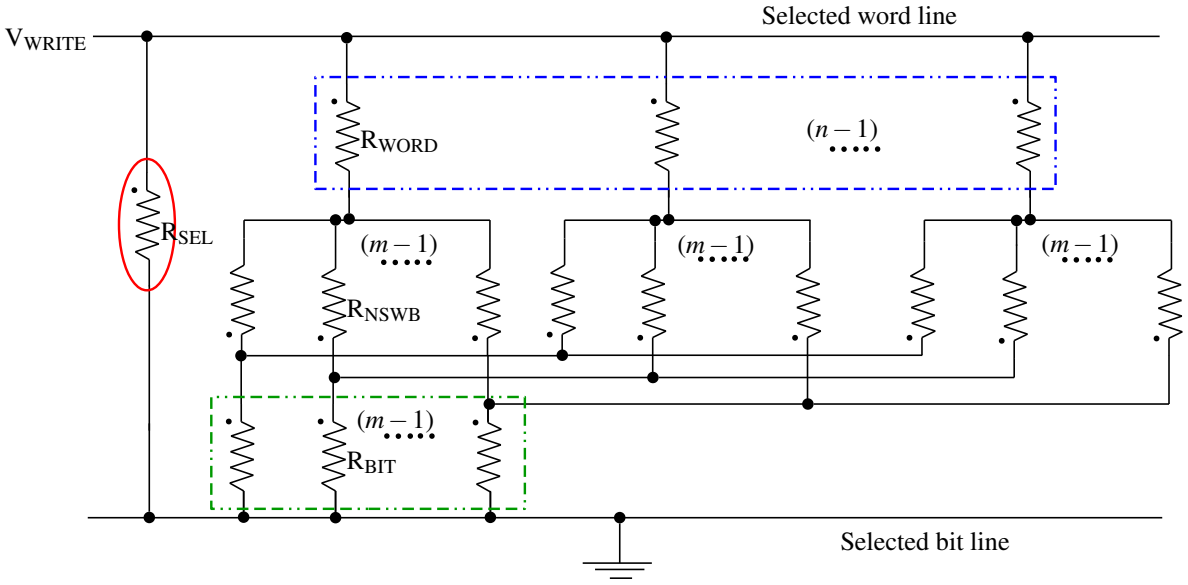


Figure 4.5. The equivalent resistive circuit for floating write scheme in memristive crossbar. The memristors with resistance R_{WORD} and R_{BIT} are prone to change their states from HRS to LRS if voltage across them cross positive threshold voltage $V_{th1,\,M}$ while memristors with resistance R_{NSWB} can switch from LRS to HRS if voltage across them cross negative threshold voltage $V_{th2,\,M}$ for V_{WRITE}=V_W. Similarly for $V_{WRITE} = -V_W$, memristors with resistance R_{WORD} and R_{BIT} can switch from LRS to HRS (upon crossing negative threshold voltage $V_{th2,\,M}$) while memristors with resistance R_{NSWB} can switch from HRS to LRS (upon crossing positive threshold voltage $V_{th1,\,M}$).

so that memristors with resistance R_{WORD}, R_{BIT} and R_{NSWB} will not change their states. If R_{WORD}=R_{LRS} or R_{BIT}=R_{LRS}, application of V_{WRITE}=V_W will not change their states but R_{NWB}=R_{LRS} can change its state because of its polarity. The maximum voltage drop across R_{BIT}, when it is in HRS, is given by

$$V_{\text{BIT, max}}\Big|_{V_{\text{WRITE}}=V_W} = \frac{(n-1)nR_{\text{HRS}}^2}{(n-1)nR_{\text{HRS}}^2 + (mn-m+2)R_{\text{HRS}}R_{\text{LRS}} + (m-2)R_{\text{LRS}}^2} V_W. \tag{4.5}$$

The maximum voltage drop across R_{WORD}, when it is in HRS, is given by

$$V_{\text{WORD, max}}\Big|_{V_{\text{WRITE}}=V_W} = \frac{(m-1)mR_{\text{HRS}}^2}{(m-1)mR_{\text{HRS}}^2 + (mn-n+2)R_{\text{HRS}}R_{\text{LRS}} + (n-2)R_{\text{LRS}}^2} V_W. \tag{4.6}$$

If the memristor with resistance R_{NSWB} is in HRS, any resultant voltage across it due to application of V_W, is not going to alter its state. The maximum voltage drop across R_{NSWB} when it is in LRS, is given by

$$V_{\text{NSWB, max}}\Big|_{V_{\text{WRITE}}=V_W} = \frac{A_1R_{\text{HRS}}^2 + A_2R_{\text{HRS}}R_{\text{LRS}} + A_3R_{\text{LRS}}^2}{C_1R_{\text{HRS}}^2 + C_2R_{\text{HRS}}R_{\text{LRS}} + C_3R_{\text{LRS}}^2} V_W \tag{4.7}$$

where,

$A_1 = 2m + 2n - 4,$

$A_2 = (2m-4)n - 4m + 8,$

$A_3 = (2-m)n + 2m - 4,$

$C_1 = 6m + 6n - 12,$

$C_2 = 2n^2 + 2m^2 + (6m-16)n - 16m + 24,$

$C_3 = (m-2)n^2 + (m^2 - 7m + 10)n - 2m^2 + 10m - 12.$

In Figure 4.6, (4.5), (4.6) and (4.7) are plotted as a function of array size ($m \times n$) for $m = n$ along with threshold voltages $V_{th1,\,M}$ and $V_{th2,\,M}$. It can be observed that the memristors in bit line and word line gets affected while other memristors are not affected in writing LRS to selected memristor. Thus this method has limited use in writing to memristive crossbar.

Similarly in order to write HRS in selected memristor, $V_{WRITE} = -V_W$. The maximum voltage across bit line memristor, when it is in LRS, is given by

$$V_{\text{BIT, max}}\Big|_{V_{\text{WRITE}}=-V_W} = -\frac{(n-1)R_{\text{HRS}}}{(n+1)R_{\text{HRS}} + (m-2)R_{\text{LRS}}} V_W. \tag{4.8}$$

The maximum voltage across word line memristor, when it is in LRS, is given by

$$V_{\text{WORD, max}}\Big|_{V_{\text{WRITE}}=-V_W} = -\frac{(m-1)R_{\text{HRS}}}{(m+1)R_{\text{HRS}} + (n-2)R_{\text{LRS}}} V_W. \tag{4.9}$$

The maximum voltage across memristor other than on word line and bit line, when it is in HRS, is given by

$$V_{\text{NSWB, max}}\Big|_{V_{\text{WRITE}}=-V_W} = \frac{A_1 R_{\text{HRS}}^3 + A_2 R_{\text{HRS}}^2 R_{\text{LRS}} + A_3 R_{\text{LRS}}^3}{C_1 R_{\text{HRS}}^3 + C_2 R_{\text{HRS}}^2 R_{\text{LRS}} + C_3 R_{\text{HRS}} R_{\text{LRS}}^2 + C_4 R_{\text{LRS}}^3} V_W, \tag{4.10}$$

where,

$A_1 = -2m - 2n + 8,$

$A_2 = (4-2m)n + 4m - 12,$

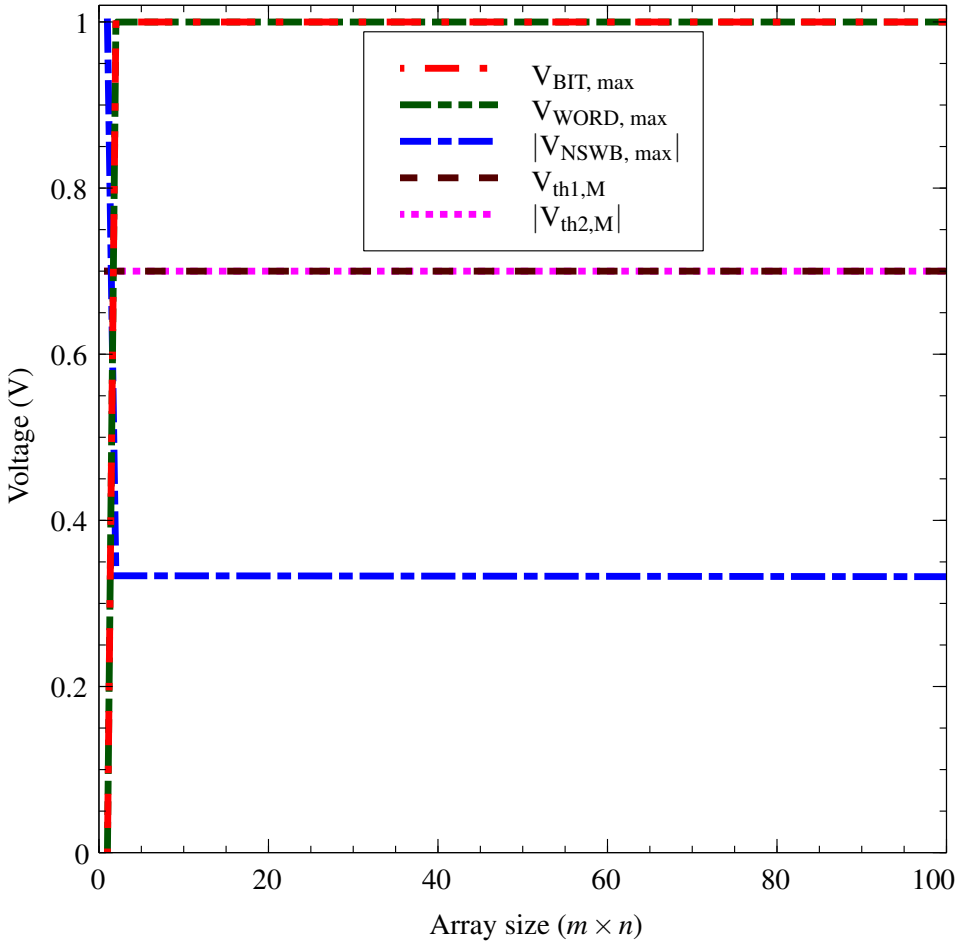


Figure 4.6. The voltage variations across unselected memristors in memristive crossbar as function of array size ($m \times n$) with $m = n$ for positive write voltage in floating write scheme to write LRS to selected memristor. In this plot V_W=1 V and $V_{th1,M}$=0.7 V, $V_{th2,\,M}$=−0.7 V. Bit line and word line memristors are affected while other memristors are not disturbed.

$$A_3 = (m-2)n - 2m + 4,$$

$$C_1 = 2m + 2n - 8,$$

$$C_2 = n^2 + (4m-8)n + m^2 - 8m + 12,$$

$$C_3 = (m-1)n^2 + (m^2 - 6m + 6)n - m^2 + 6m,$$

$$C_4 = mn - 4.$$

In Figure 4.7, (4.8), (4.9) and (4.10) are plotted as a function of array size ($m \times n$) for $m = n$ along with threshold voltages $V_{th1,\,M}$ and $V_{th2,\,M}$. In order to write HRS, we apply V_{WRITE}=−V_W and it can be observed from the Figure 4.7 that the write operation does not affect bit line and word line memristors for approximate array size up to 8×8. But the major problem is with other memristors

as they are affected even for small array size in an attempt to write HRS in selected memristor. Again because of this drawback, floating write scheme has limited use.

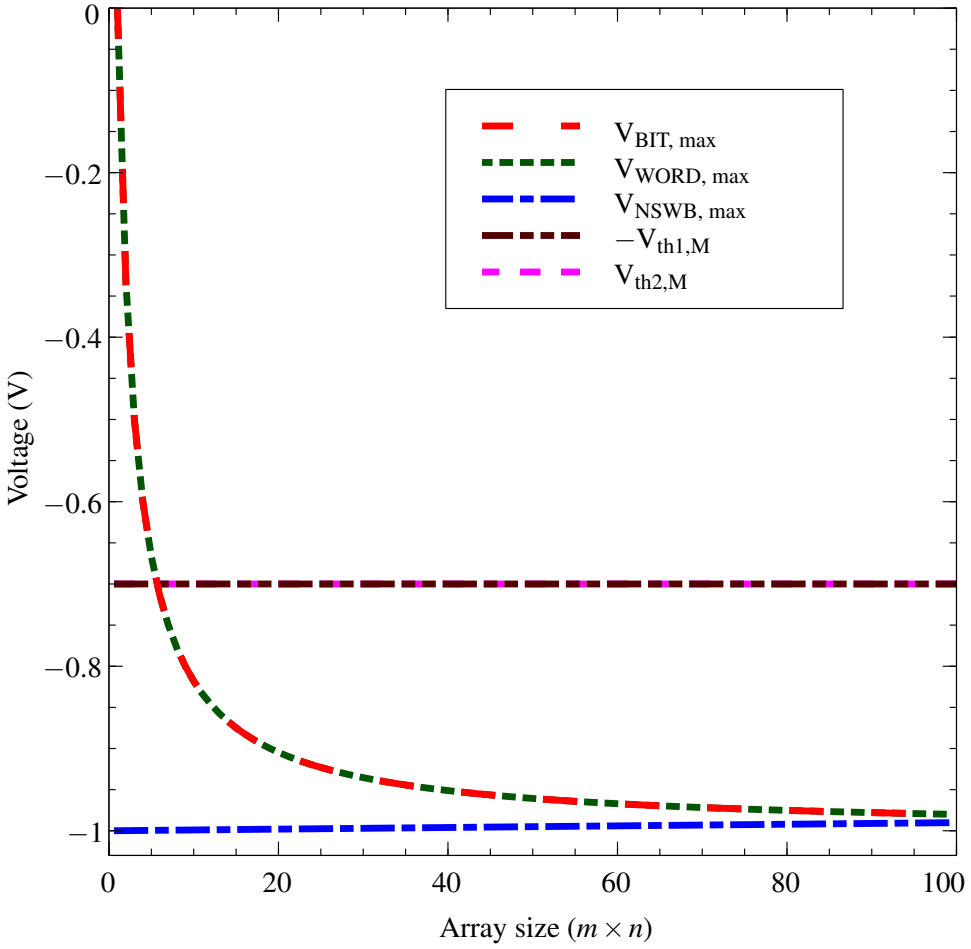


Figure 4.7. The voltage variations across unselected memristors in memristive crossbar as function of array size ($m \times n$) with $m = n$ for negative write voltage in floating write scheme to write HRS to selected memristor. In this plot V_{WRITE}=$-V_W$=-1 V and $V_{th1,\,M}$=0.7 V, $V_{th2,\,M}$=-0.7 V. Bit line and word line memristors are not affected up to array size 8×8 for selected parameters but other memristors are affected even for small size of array.

4.4.1.2 1/3 Write Scheme

In this scheme, the selected word line is driven by voltage V_{WRITE} while selected bit line is grounded. At the same time unselected word lines are driven by voltage $V_{WRITE}/3$ and unselected bit lines are driven by voltage $2V_{WRITE}/3$. Thus always the voltage across selected memristor will be V_{WRITE} while voltage across each unselected memristor will be $|V_{WRITE}/3|$. This scheme is shown in Figure 4.8. For writing LRS, V_{WRITE}=V_W and for writing HRS V_{WRITE}=$-V_W$. The

voltage $\pm V_W$ is selected such that $V_W > V_{th1,M}$, $-V_W < V_{th2,M}$, and $V_{th2,M} < |\pm V_W/3| < V_{th1,M}$. This method will not alter the states of memristors other than selected memristor, unlike the floating write scheme, and hence widely used for writing LRS and HRS states in memristors.

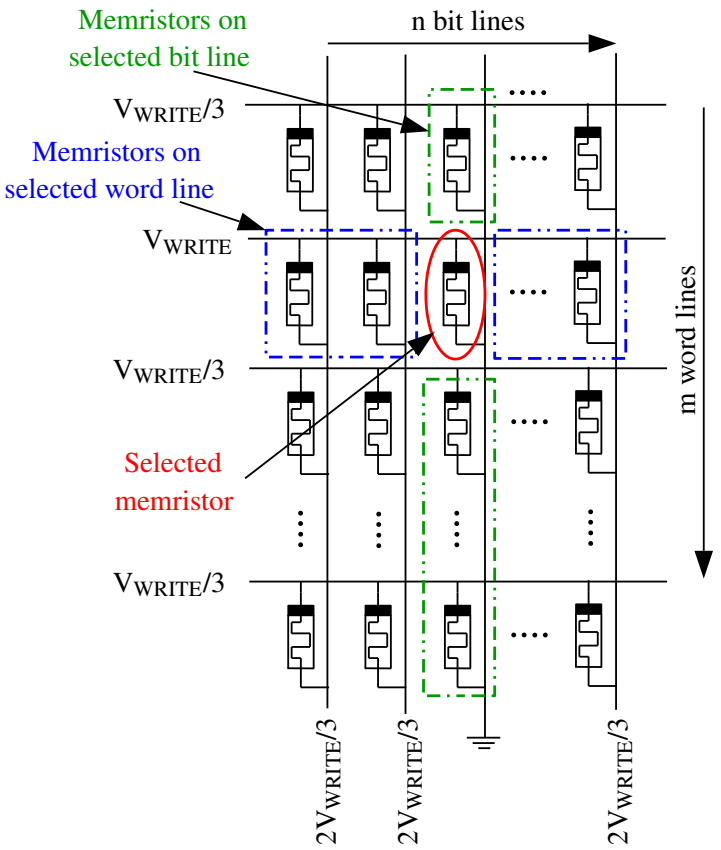


Figure 4.8. The 1/3 write scheme for memristive crossbar. The selected word line is driven by write voltage V_{WRITE} while selected bit line is grounded. Remaining word lines are driven with voltage $V_{WRITE}/3$ and bit lines are driven with voltage $2V_{WRITE}/3$. The effective voltage across each unselected memristor will be $|V_{WRITE}/3|$.

The power consumed in write operation for memristive crossbar is given by

$$P_{\text{total,mem}} = P_{\text{SEL}} + P_{\text{BIT}} + P_{\text{WORD}} + P_{\text{NSWB}} \tag{4.11}$$

The power consumed in memristive crossbar is function of array size (m $\times$ n), the threshold voltage of memristor (as it restricts the minimum value of voltage to be used in write operation) and R_{HRS}/R_{LRS} values of memristor. The maximum power consumed in write operations as a function of array size (m $\times$ n) for symmetric array and for different write schemes is shown in

Figure 4.9. The 1/3 write method is widely used as it does not change the state of other memristors as explained previously in the description of 1/3 write scheme.

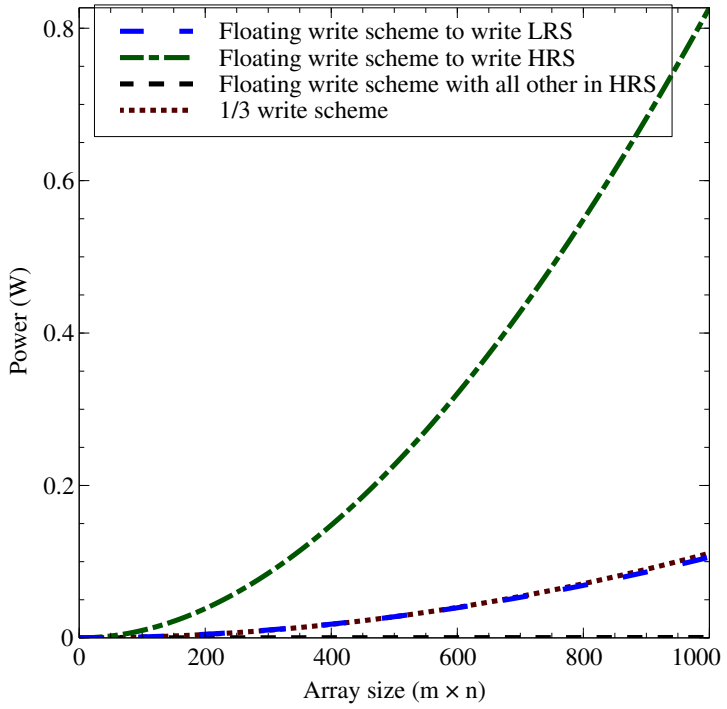


Figure 4.9. Power consumption for different write schemes in memristive crossbar as function of array size (m $\times$ n) for symmetric crossbar. The 1/3 write scheme is most widely used as it does not change the state of unselected memristors.

4.4.2 Read Operation

To read the state of memristor, voltage V_{READ}=V_R, where $V_{th2,\,M} < V_R < V_{th1,\,M}$ (in order to maintain the stste of memristor to be read)is applied to its word line, while resulting voltage is sensed across resistance R_S connected to its bit line. The scheme is shown in Figure 4.10 and its equivalent circuit is shown in Figure 4.11.

The problem with memristive crossbar is sneak path. Consider the worst case scenario where all unselected memristors are in LRS. If memristor selected for read operation is in LRS then the sensed voltage across R_S for read voltage V_{READ}=V_R is given by

$$V_{R_S} = \frac{mnR_S}{mnR_S + (m+n-1)R_{LRS}} V_R \tag{4.12}$$

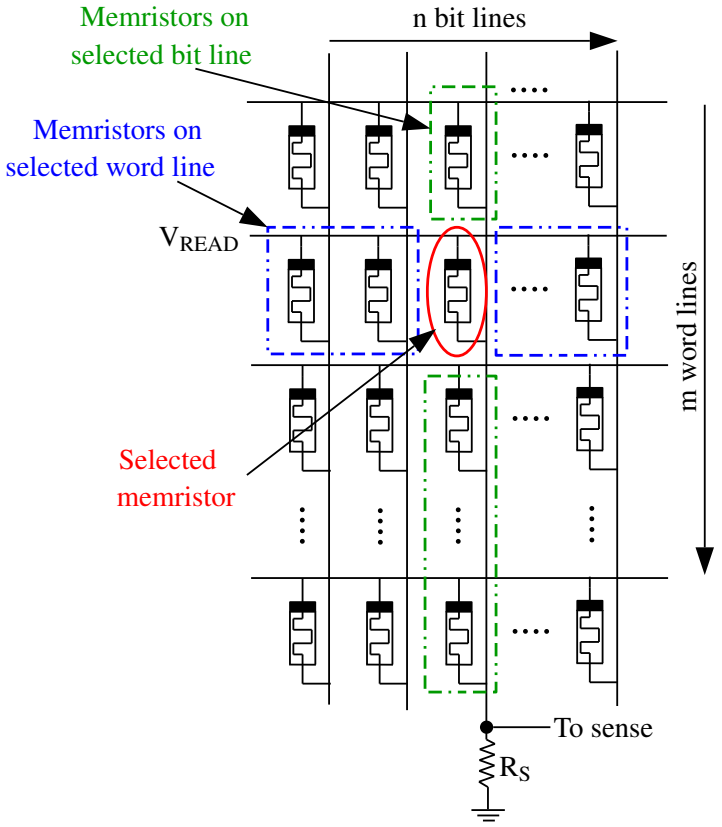


Figure 4.10. The read voltage V_{READ}=V_R, where $V_{th2,\,M} < V_R < V_{th1,\,M}$ is applied to word line of memristor to be read while voltage is sensed across the resistance R_S connected to its bit line. If sensed voltage is large, the selected memristor is in LRS otherwise it is in HRS.

If memristor to be read is in HRS then

$$V_{R_S} = \frac{(m-1)(n-1)R_{HRS}R_S + (m+n-1)R_{LRS}R_S}{(m-1)(n-1)R_{HRS}R_S + (m+n-1)(R_{HRS}+R_S)R_{LRS}}V_R \tag{4.13}$$

in Figure 4.12, (4.12) and (4.13) are plotted as a function of array size ($m \times n$) with $m = n$. The sensed voltage becomes indistinguishable for the read memristor in LRS or HRS as the array size becomes larger (in this case around 10×10). Thus the sneak path problem limits the size of array that can be read without error.

4.4.3 Evaluate Operation (Stateful-NOR Operation)

In this section, the effect of implementation of 3-input NOR function on memristors in memristive crossbar is analyzed. The implementation of 3-input NOR function (without CMOS supporting circuit) on memristive crossbar is shown in Figure 4.13 and its resistive equivalent is shown in

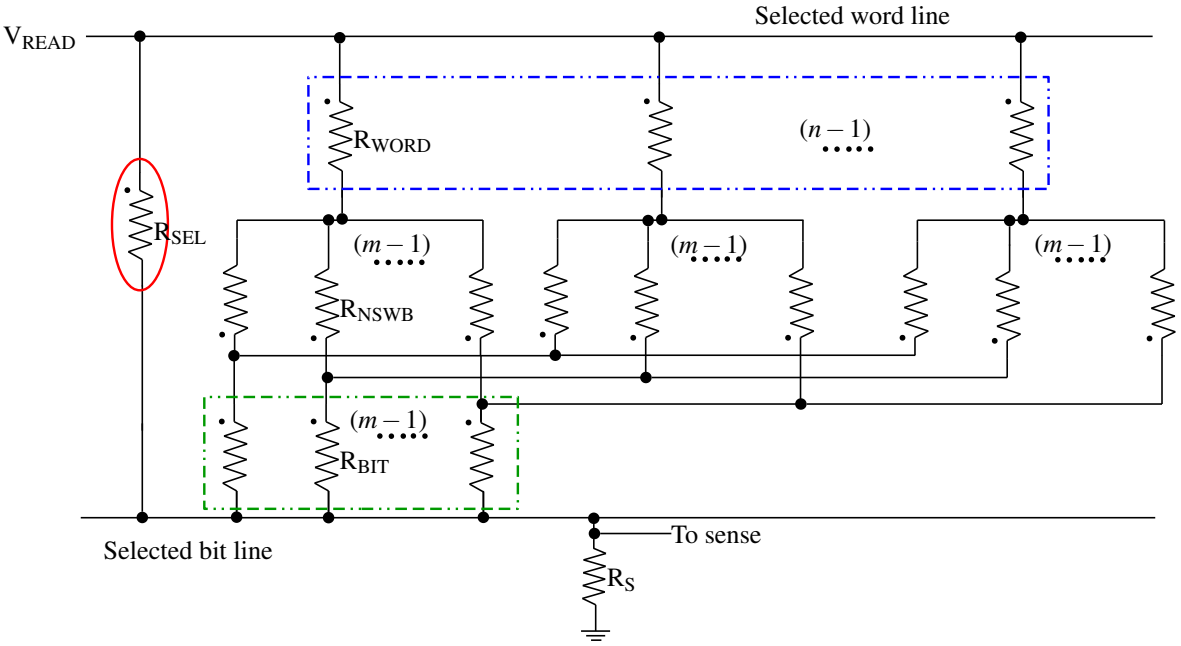


Figure 4.11. The resistive equivalent circuit for read scheme in memristive crossbar. To read the state of memristor, voltage V_{READ} is applied to its word line while the resulting voltage is sensed across R_S connected to its bit line. It is expected that when memristor is in LRS, the sensed voltage is larger. If memristor to be read is in HRS state, the sensed voltage is smaller in value. But sensed voltage also depends on the states of surrounding memristors, making it difficult to differentiate between HRS and LRS state of memristor.

Figure 4.14. With the shown polarity (using dot notations defined earlier), the memristors other than input and destination memristors namely $R_{CONDWORD}$, $R_{SETWORD}$ and R_{BIT} will change their states if they are in HRS and voltage across them cross the threshold level $V_{th1,\,M}$ while R_{NSWB} will change its state if it is in LRS and voltage across it (with polarity shown in figure) is less than $V_{th2,\,M}$. The maximum voltage across every such memristor is given below to analyze possible state change. The stateful NOR gate consists of three input memristors with resistance $R_{SELINPUT}$, one destination memristor with resistance $R_{SELDEST}$ and resistance R_G. The notations used in the analysis of 3-input NOR function implementation on memristive crossbar are described in Table 4.2. The maximum voltage across memristor other than input memristors in word lines where input memristors are present, when it is in HRS, is given by

$$V_{\mathrm{WCOND,\,max}} = -\frac{[A_1R_{\mathrm{HRS}}^2 + A_2R_{\mathrm{HRS}} + A_3]V_{\mathrm{COND}} + [B_1R_{\mathrm{HRS}} + B_2]V_{\mathrm{SET}}}{C_1R_{\mathrm{HRS}}^2 + C_2R_{\mathrm{HRS}} + C_3}, \tag{4.14}$$

where,

$A_1 = (4-m)$,

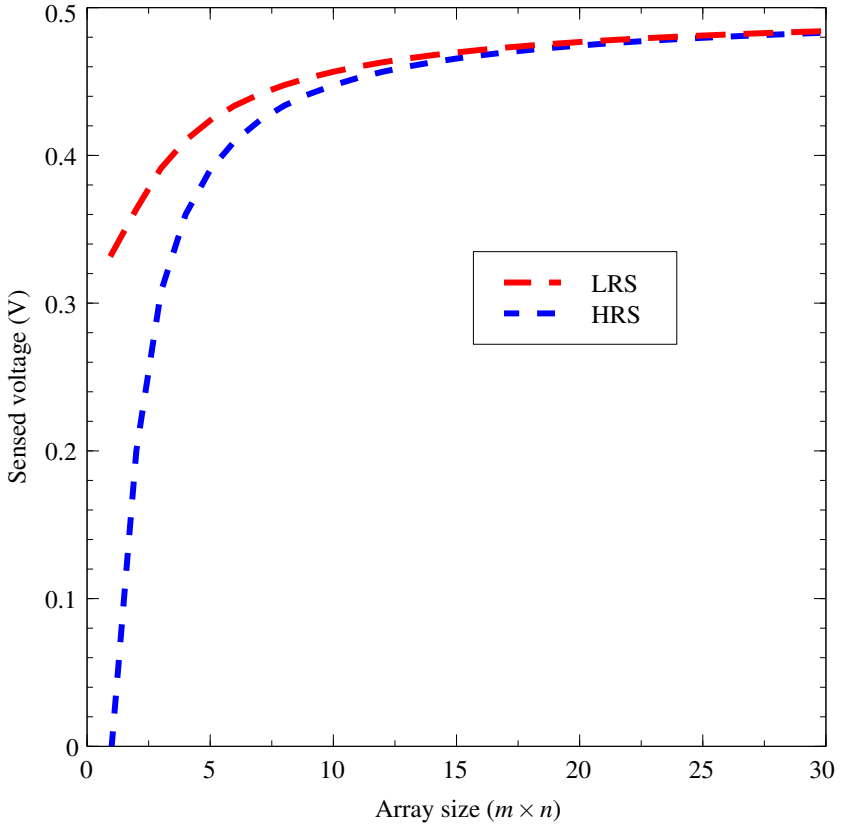


Figure 4.12. The voltage across sense resistor in memristive crossbar when memristor to be read in LRS/HRS and other surrounding memristors are in LRS as function of array size ($m \times n$) for symmetric array. In memristive read operation shown in Figure 4.10, the voltage across sense resistor R_S becomes indistinguishable for memristor to be read in either LRS or HRS when all other memristors are in LRS. This limitation due to sneak path problems restricts the size of memristive crossbar array that can be read without error.

Table 4.2. Notations used in memristive crossbar array analysis in evaluate operation for 3-input NOR implementation.

Notation	Meaning	Voltage notation
$R_{SELINPUT}$	Resistance of selected input memristors.	$V_{SELINPUT}$
$R_{SELDEST}$	Resistance of selected destination memristors.	$V_{SELDEST}$
$R_{CONDWORD}$	Resistance of each memristor on selected input word line except input memristors.	V_{WCOND}
$R_{SETWORD}$	Resistance of each memristor on selected destination word line except destination memristor.	V_{WSET}
R_{BIT}	Resistance of each memristor on selected bit line except input and destination memristor.	V_{BIT}
R_{NSWB}	Resistance of each memristor on unselected word lines and bit lines.	V_{NSWB}

$$A_2 = (4-m)nR_G - nR_{LRS},$$

$$A_3 = -4nR_G R_{LRS},$$

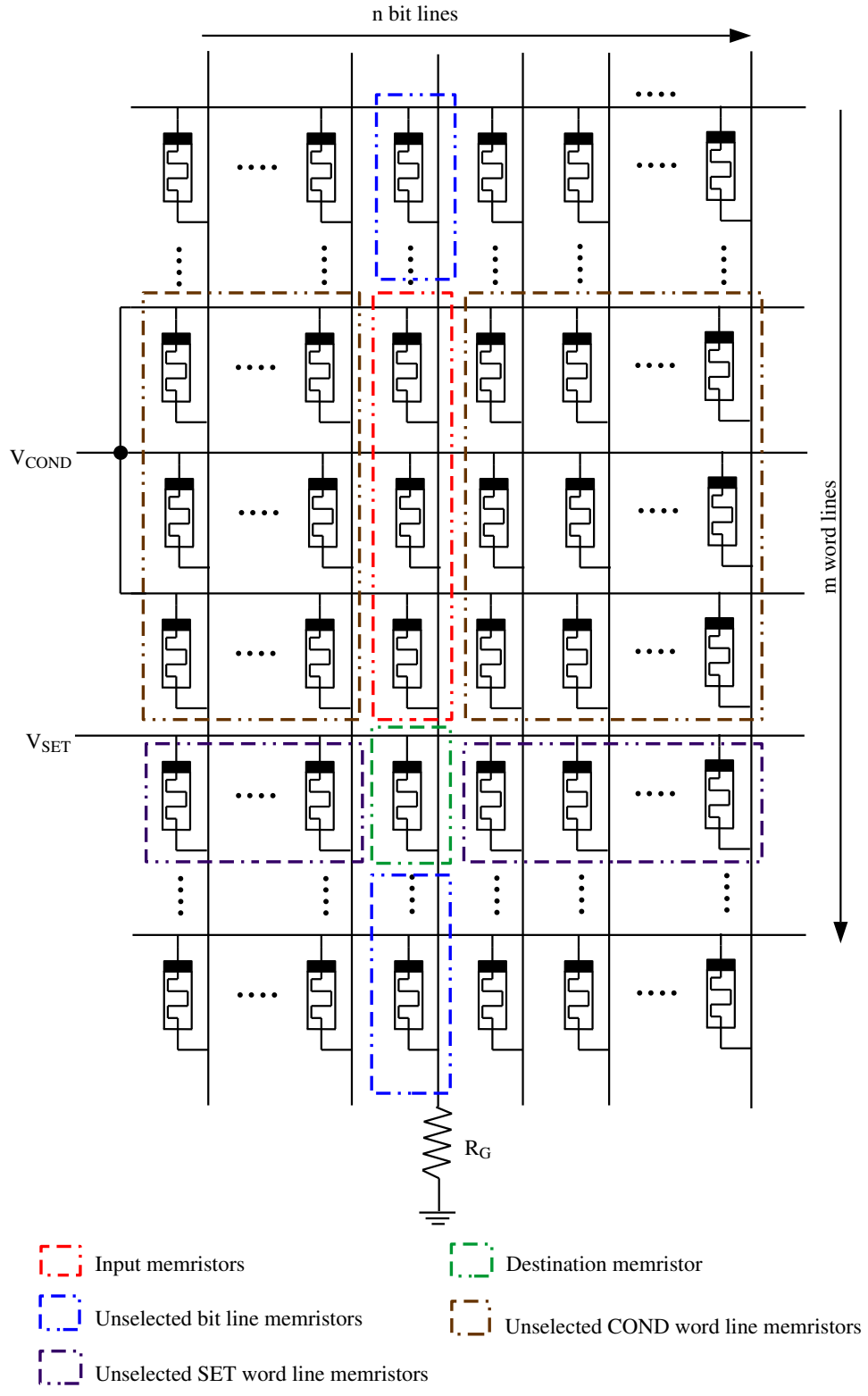


Figure 4.13. The implementation of evaluate operation for 3-input NOR logic on memristive crossbar. V_{COND} is applied to three word lines of three input memristors while V_{SET} is applied to word line of destination memristor. The common bit line of all is connected to ground through resistance R_G.

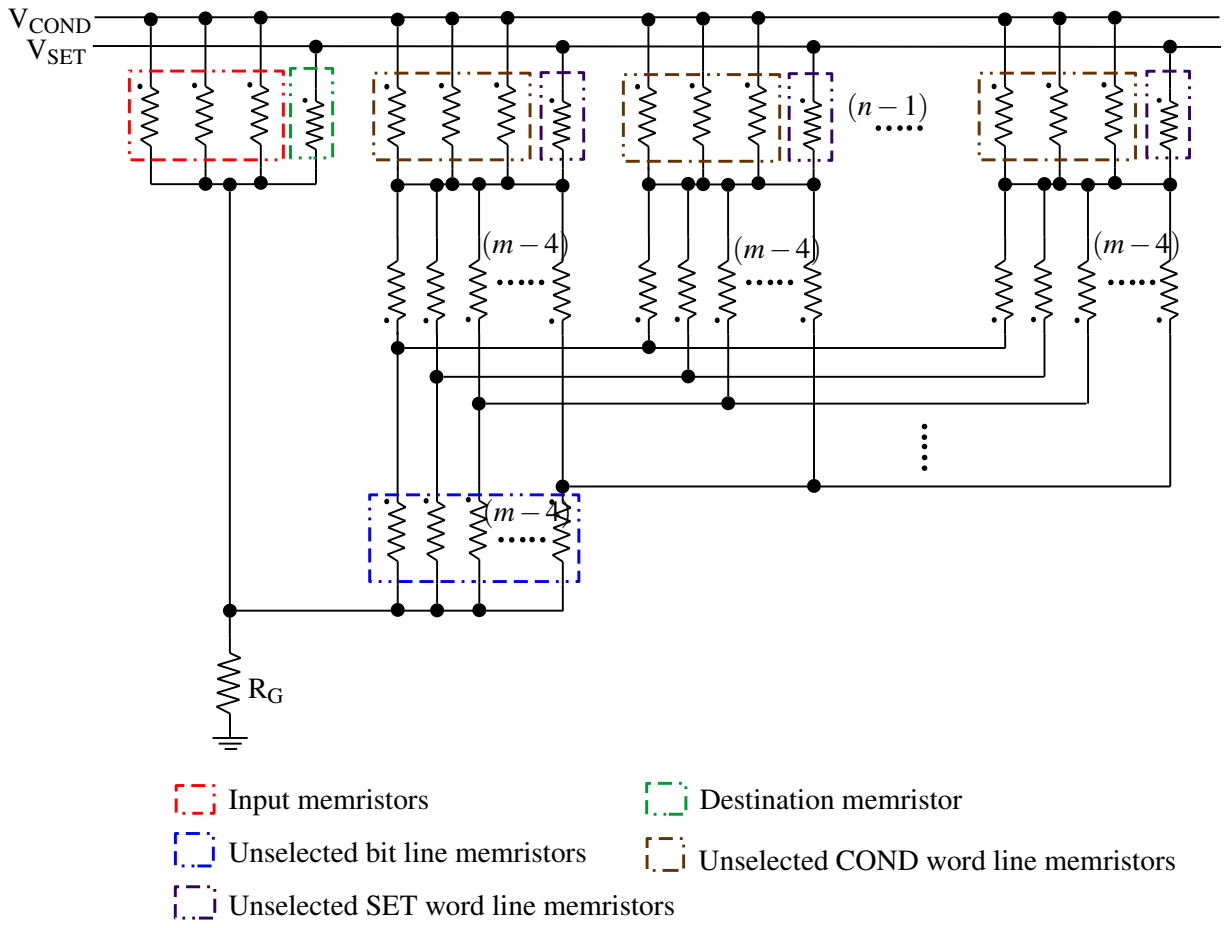


Figure 4.14. The equivalent resistive circuit (of circuit shown in Figure 4.13) for implementation of evaluate operation for 3-input NOR logic on memristive crossbar. The orientation of memristors in crossbar is shown with dot notation defined in Figure 4.3.

$B_1 = (m-4)nR_G + nR_{LRS}$,

$B_2 = 4nR_G R_{LRS}$,

$C_1 = m-4$,

$C_2 = (4m-16)nR_G + 4nR_{LRS}$,

$C_3 = 16nR_G R_{LRS}$.

The maximum voltage across memristor other than destination memristor in word lines where destination memristor is present, when it is in HRS, is given by

$$V_{WSET,\,max} = \frac{[A_4 R_{HRS}^2 + A_5 R_{HRS} + A_6]V_{SET} + [B_3 R_{HRS} + B_4]V_{COND}}{C_4 R_{HRS}^2 + C_5 R_{HRS} + C_6}, \tag{4.15}$$

where,

$A_4 = (m-4)$,

$A_5 = (3m-12)nR_{\mathrm{G}} + 3nR_{\mathrm{LRS}}$,

$A_6 = 12nR_{\mathrm{G}}R_{\mathrm{LRS}}$,

$B_3 = (12-3m)nR_{\mathrm{G}} - 3nR_{\mathrm{LRS}}$,

$B_4 = -12nR_{\mathrm{G}}R_{\mathrm{LRS}}$,

$C_4 = m-4$,

$C_5 = (4m-16)nR_{\mathrm{G}} + 4nR_{\mathrm{LRS}}$,

$C_6 = 16nR_{\mathrm{G}}R_{\mathrm{LRS}}$.

The maximum voltage across memristor other than input and destination memristors in bit line where input and destination memristors are present, when it is in HRS, is given by

$$V_{\mathrm{BIT,\,max}} = \frac{[A_1R_{\mathrm{HRS}}^3 + A_2R_{\mathrm{HRS}}^2]V_{\mathrm{SET}} + [B_1R_{\mathrm{HRS}}^3 + B_2R_{\mathrm{HRS}}^2]V_{\mathrm{COND}}}{C_1R_{\mathrm{HRS}}^3 + C_2R_{\mathrm{HRS}}^2 + C_3R_{\mathrm{HRS}} + C_4}, \tag{4.16}$$

where,

$A_1 = mn^2 - 2mn + 5$,

$A_2 = (m-5)R_{\mathrm{LRS}}$,

$B_1 = 3mn^2 - 6mn + 15$,

$B_2 = (3m-15)R_{\mathrm{LRS}}$,

$C_1 = 4mn^2 - 8mn + 20$,

$C_2 = [4m^2n^2 + (20m-12m^2)n + 8m^2 - 20m]R_{\mathrm{G}} + (m^2n - 2m^2 + 9m - 20)R_{\mathrm{LRS}}$,

$C_3 = (m^2-5m)R_{\mathrm{LRS}}^2 + [(8m^2-20m)n - 12m^2 + 40m]R_{\mathrm{G}}R_{\mathrm{LRS}}$,

$C_4 = (4m^2-20m)R_{\mathrm{G}}R_{\mathrm{LRS}}^2$.

in Figure 4.15, (4.14), (4.15) and (4.16) are plotted against array size ($m \times n$) with $m = n$. For 3-input NOR, the minimum size of array ($m \times n$) should be (4×4) for symmertic array. The unselected memristors in word lines and bit lines are not disturbed if array size is more than 50×50 for selected parameters.

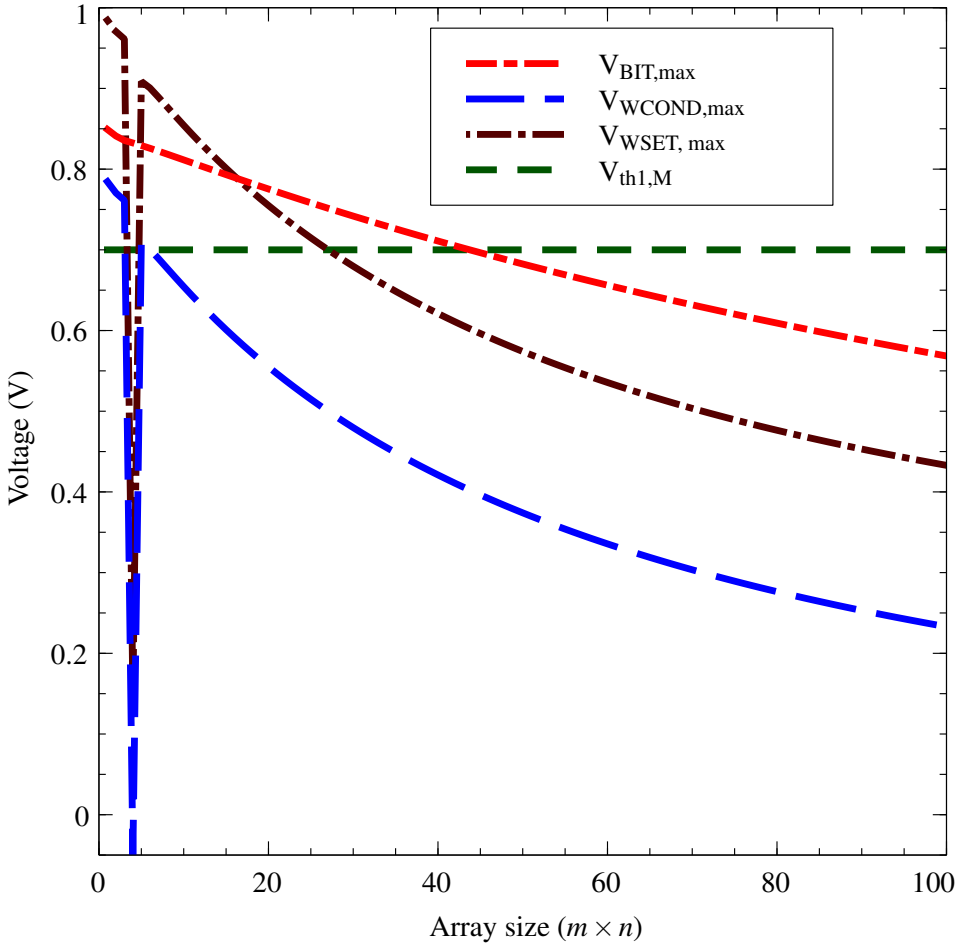


Figure 4.15. The maximum voltage variations across unselected memristors on selected word lines, bits lines, which are in HRS in memristive crossbar for 3-input NOR function implementation as function of array size $m \times n$ with $m = n$. All unselected memristors on word lines, bit lines are not disturbed if array size is more than 50×50.

The maximum voltage across unselected memristor not present on selected word lines and bit line, when it is in LRS, is given by

$$V_{\text{NSWB, max}} = \frac{[A_1 R_{\text{HRS}}^2 + A_2 R_{\text{HRS}} + A_3]V_{\text{SET}} + [B_1 R_{\text{HRS}}^2 + B_2 R_{\text{HRS}} + B_3]V_{\text{COND}}}{C_1 R_{\text{HRS}}^2 + C_2 R_{\text{HRS}} + C_3}, \tag{4.17}$$

where,

$A_1 = (3n-6)(R_G + R_{LRS})$,

$A_2 = [(3m-14)n - 6m + 28]R_G R_{LRS} + (n-2)R_{LRS}^2$,

$A_3 = [(m-5)n - 2m + 10]R_G R_{LRS}^2 + R_{LRS}^3$,

$B_1 = (6-3n)(R_G + R_{LRS})$,

$B_2 = [(14-3m)n + 6m - 28]R_G R_{LRS} + (2-n)R_{LRS}^2$,

$B_3 = [(5-m)n + 2m - 10]R_G R_{LRS}^2 + 3R_{LRS}^3$,

$C_1 = (2-n)(R_G + R_{LRS})$,

$C_2 = [(6-m)n + 2m - 12]R_G R_{LRS} + (n-2)R_{LRS}^2$,

$C_3 = [(m-5)n - 2m + 6]R_G R_{LRS}^2 + R_{LRS}^3$.

In Figure 4.16, (4.17) is plotted as a function of array size ($m \times n$) with $m = n$. It is clear that the voltage across all unselected memristors, that are not on selected word lines and bit lines, is always less than $V_{th2,\,M}$ and state change never occur for these memristors in the implementation of 3-input NOR logic function on memristive crossbar.

Voltage across destination memristor (which is initialized to R_{HRS} at the start of NOR implementation), when all input memristors are in HRS state is given by,

Case-I : When all other memristors are in HRS

$$V_{DEST} = \frac{[3mnR_G + 4nR_{HRS} + (m-4)R_{HRS}]V_{SET} - 3mnR_G V_{COND}}{(4n+m-4)R_{HRS} + 4mnR_G}. \tag{4.18}$$

Case-II: When all other memristors are in LRS

$$V_{DEST} = \frac{[A_1 R_{HRS} + A_2]V_{SET} + [B_1 R_{HRS} + B_2]V_{COND}}{C_1 R_{HRS} + C_2}, \tag{4.19}$$

where,

$A_1 = [(3m-12)n - 3m + 12]R_G + (4n+m-4)R_{LRS}$,

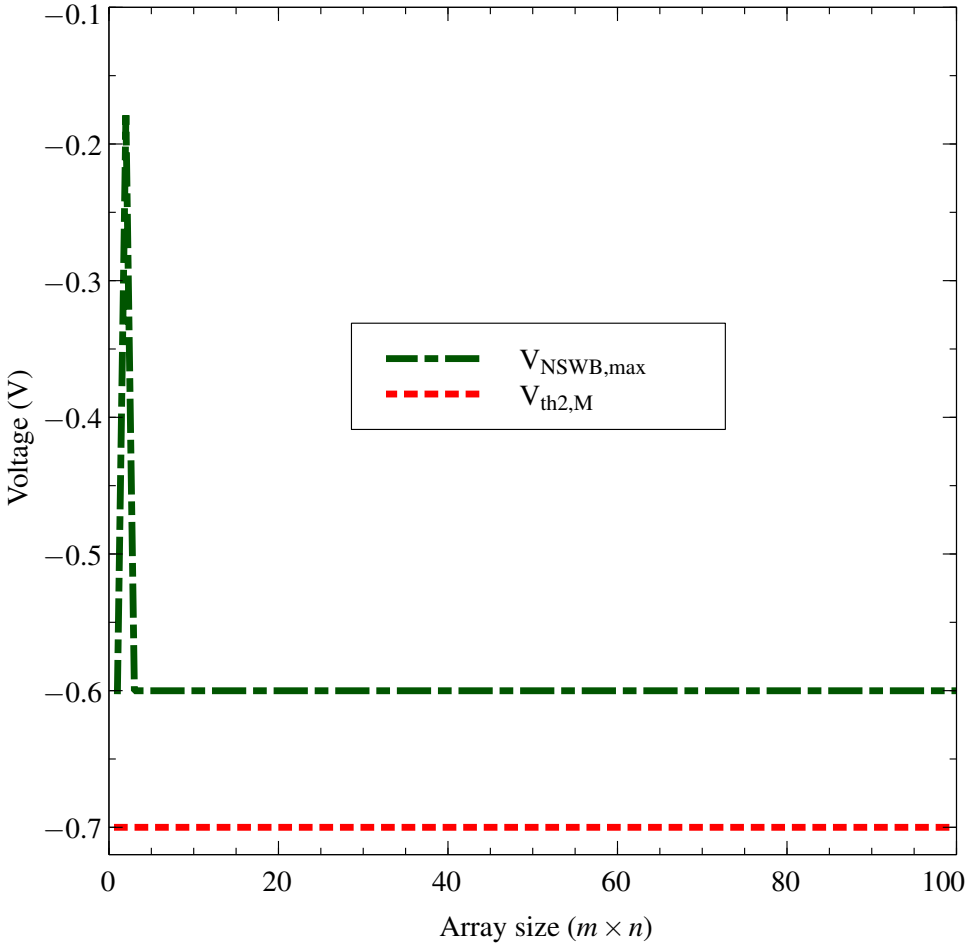


Figure 4.16. The maximum voltage variations across unselected memristors not on selected word lines and bit line in memristive crossbar as function of array size $m \times n$ with $m = n$ for implementation of 3-input NOR function. The unselected memristors not on selected word lines and bit line are not disturbed for any array size in implementation of 3-input NOR function.

$$A_2 = (12n + 3m - 12)R_G R_{LRS},$$

$$B_1 = [(12 - 3m)n + 3m - 12]R_G,$$

$$B_2 = (-12n - 3m + 12)R_G R_{LRS},$$

$$C_1 = (4m - 16)(n - 1)R_G + (4n + m - 4)R_{LRS},$$

$$C_2 = (16n + 4m - 16)R_G R_{LRS}.$$

In Figure 4.17, (4.18) and (4.19) are plotted as a function of array size $m \times n$ with $m = n$. In this case when all input memristors are in HRS (logic '0'), the expected voltage drop across destination memristor should be more than $V_{th1,\,M}$ so that destination memristor initialized to HRS before the implementation of logic operation should toggle to LRS (logic '1') for correct NOR function.

There is no problem when all other memristors are in HRS, but if in worst case, when all other memristors are in LRS then there is limit on the maximum size of array (in our case it is 10×10) for correct implementation of 3-input NOR operation.

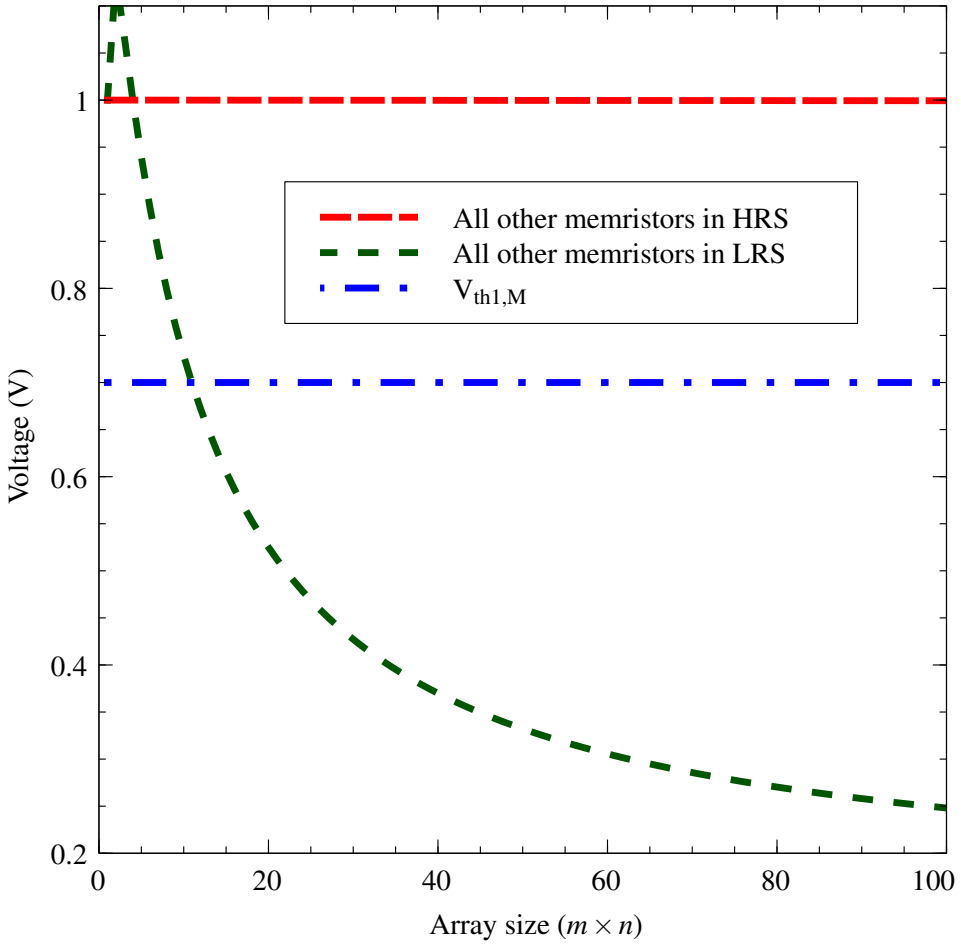


Figure 4.17. The voltage variations across destination memristor in memristive crossbar as function of array size $m \times n$ with $m = n$ during implementation of NOR logic function when all input memristors are in HRS. When all other memristors are in HRS, the voltage across destination memristor is greater than $V_{th1,\,M}$ and hence it switches to LRS state. When all other memristor are in LRS, then the voltage across destination memristor is less than $V_{th1,\,M}$ for array size greater than 10×10 and will not switch to LRS even with all input memristors are in HRS. This is another consequence of sneak path problem.

In other cases when one or more of the input memristors are in LRS, the effective voltage across the destination memristor will always be less than threshold voltage $V_{th1,\,M}$. For example when one input memristor is in LRS for 3-input NOR gate, the voltage across destination memristor will be

Case-I : When all other memristors are in HRS

$$V_{\mathrm{DEST}} = \frac{[A_1 R_{\mathrm{HRS}} + A_2]V_{\mathrm{SET}} + [B_1 R_{\mathrm{HRS}} + B_2]V_{\mathrm{COND}}}{C_1 R_{\mathrm{HRS}} + C_2}, \tag{4.20}$$

where,

$A_1 = (12n + 3m - 12)R_{\mathrm{G}} + (4n + m - 4)R_{\mathrm{LRS}}$,

$A_2 = [(3m - 12)n - 3m + 12]R_{\mathrm{G}}R_{\mathrm{LRS}}$,

$B_1 = (-12n - 3m + 12)R_{\mathrm{G}}$,

$B_2 = [(12 - 3m)n + 3m - 12]R_{\mathrm{G}}R_{\mathrm{LRS}}$,

$C_1 = (12n + 3m - 12)R_{\mathrm{G}} + (4n + m - 4)R_{\mathrm{LRS}}$,

$C_2 = [(4m - 12)n - 3m + 12]R_{\mathrm{G}}R_{\mathrm{LRS}}$.

Case-II : When all other memristors are in LRS

$$V_{\mathrm{DEST}} = \frac{[(4n + m - 4)R_{\mathrm{LRS}} + 3mnR_{\mathrm{G}}]R_{\mathrm{HRS}}V_{\mathrm{SET}} - 3mnR_{\mathrm{G}}R_{\mathrm{HRS}}V_{\mathrm{COND}}}{([(4m-4)n - m + 4]R_{\mathrm{G}} + [4n + m - 4]R_{\mathrm{LRS}})R_{\mathrm{HRS}} + (4n + m - 4)R_{\mathrm{G}}R_{\mathrm{LRS}}}. \tag{4.21}$$

In Figure 4.18, (4.20) and (4.21) are plotted as a function of array size $m \times n$ with $m = n$. Here it can be seen that the voltage across the destination memristor is always less than $V_{\mathrm{th1,\,M}}$ for any array size $m \times n$ with $m = n$ and hence destination memristor will remain in HRS for if any or all input memristor(s) is(are) in LRS. This is necessary for correct 3-input NOR operation.

The maximum power consumption for implementation of stateful NOR gate as a function of array size (m $\times$ n) for symmetric array is shown in Figure 4.19. In addition to $R_{\mathrm{HRS}}/R_{\mathrm{LRS}}$ values of memristor, power consumption is also function of V_{SET} and V_{COND} voltage values used in evaluate operation which in turn depends on threshold voltage of memristor.

In order to implement stateful NOR logic on memristive crossbar successfully, following strategy can be adopted. For writting the inputs to memristors, 1/3 write scheme is more suitable as it does not change the contents of memristors not selected for write operation. While performing material implication to implement stateful NOR logic, if size of array is more than 50×50, the logic can be successfully implemented without affecting contents of memristors not taking part in logic. But if

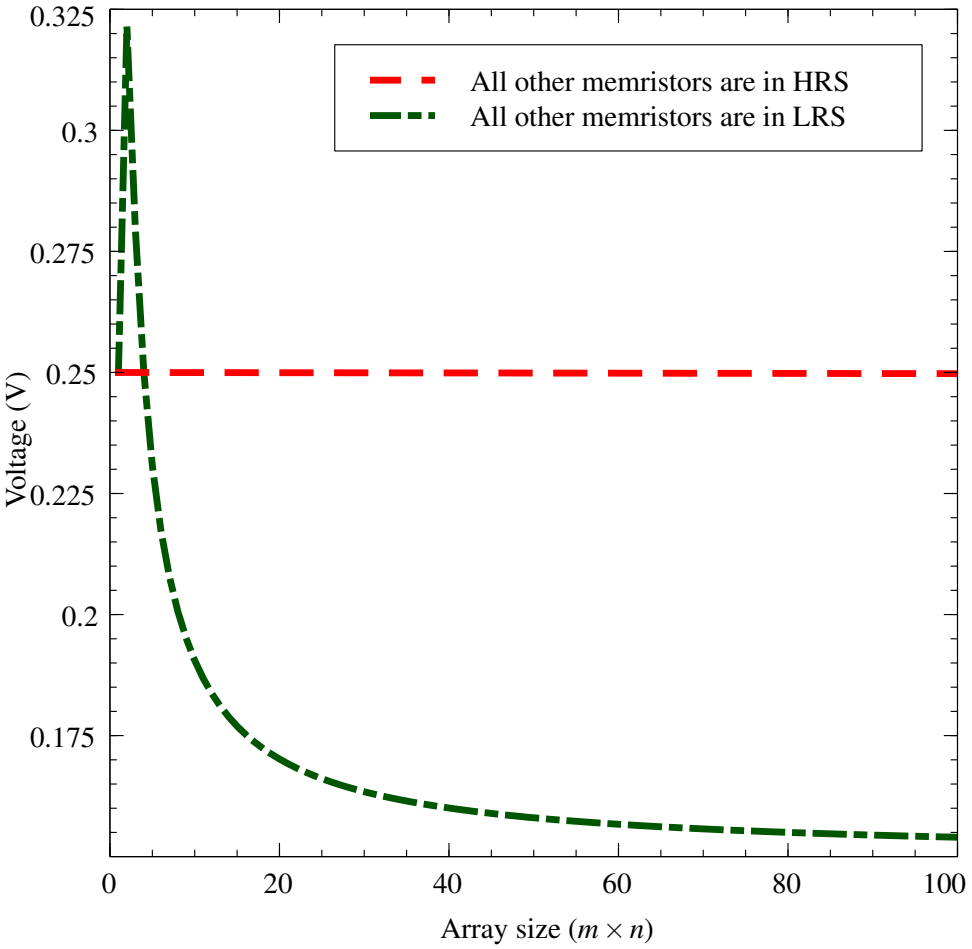


Figure 4.18. The voltage variations across destination memristor in memristive crossbar as function of array size $m \times n$ with $m = n$ during implementation of NOR logic function when any or all input memristors are in LRS. When all other memristors are in HRS or LRS, the voltage across destination memristor is less than $V_{th1,\,M}$ and hence destination memristor will retain its HRS state.

size of the array is more than 10×10, it is difficult to read the state of destination memristor (result of NOR operation) due to sneak path problem. To remove this restriction on size of crossbar while reading the destination memristor, all other memristors in the word line of destination memristor should be forced to HRS. Other alternative is to keep size of array below or equat to 10×10 and allow NOR operation to change the states of word line and bit line memristors (other memristors are not affected for any array size). Such small size array can only be used for computations and not as a memory because its content gets destroyed. Also multiple NOR operations can be implemented diagonally on large crossbar array, such that one NOR implementation is done in area not on word lines and bit lines of the other, and word lines of one NOR operation become bit lines for other NOR operation. In such case, the effective voltage across memristors not taking

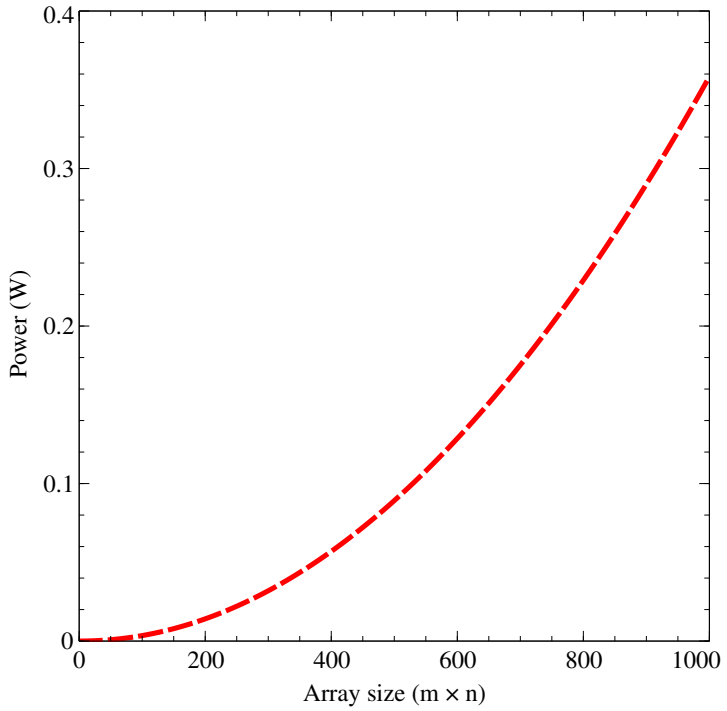


Figure 4.19. Power consumption for evaluate operation in memristive crossbar as function of array size (m $\times$ n) for symmetric crossbar.

part in logic implementation get reduced because of opposite polarity of word line and bit line voltages, evident from the analysis.

4.5 Logic Implementation on Specialized Memristive Crossbar

The analysis of implementation of stateful NOR logic on memristive crossbar has shown that the size of crossbar has to be restricted, specifically to read the state of destination memristor (the result of stateful NOR operation). One way is to use small size repeated regular array structures in order to avoid the sneak path problem. In other method, the specialized architectures such as CMOL [46] and FPNI [54] can be used where wire segments of small length are arranged such that small segmented array structure is created in large architectural area. The details of these specialized architecture are given in Appendix C .

Stateful NOR logic operation can be implemented on specialized crossbar such as CMOL as shown in Figure 4.21. In this figure only nanowires used in the implementation of stateful NOR are shown (NV is vertical nanowire with NV_{p} as its access pin while $NH_1 - NH_4$ are horizontal nanowires

with $NH_{p1} - NH_{p4}$ as their access pins). The nanowire can be accessed through pass transistor (as shown in Figure C.2 where gate of pass transistor is used to enable access and source/drain for performing operations on nanowire) or transmission gate. The conceptual diagram is shown in Figure 4.20 and its implementation on CMOL is shown in Figure 4.21. The switches are either pass transistors or transmission gates.

In write mode, the inputs are written to input memristors (M_1, M_2, M_3) and logic '0' is written to destination memristor (M_4). This is done in two steps. In first step, all logic '0' input values are written and in second step all logic '1' values are written. For example, if inputs to 3-input NOR gates are '010' then logic '0' is written to memristors M_1, M_3 and M_4 in step 1 of write mode and then logic '1' is written to memristor M_2 in second step. When effective voltage across memristor is $2V_W$, logic '1' (i.e. R_{LRS}) is written to it ($2V_W > V_{th1,M}$). When effective voltage across memristor is $-2V_W$, logic '0' (i.e. R_{HRS}) is written to it ($-2V_W < V_{th2,M}$). If effective voltage is $\pm V_W$ across memristor, it will not change its state.

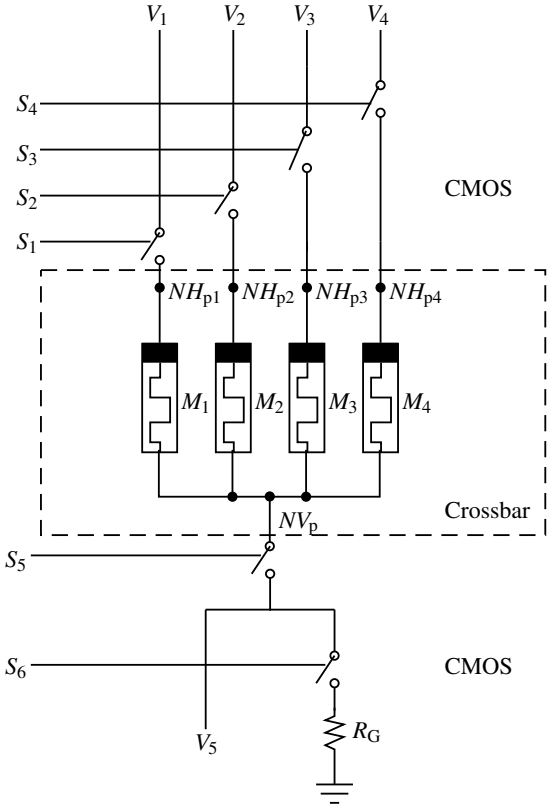


Figure 4.20. Equivalent circuit for 3-input stateful NOR gate implemented on crossbar as shown in Figure 4.21. Switches $S_1 - S_6$ are implemented in CMOS layer using pass transistors/transmission gates.

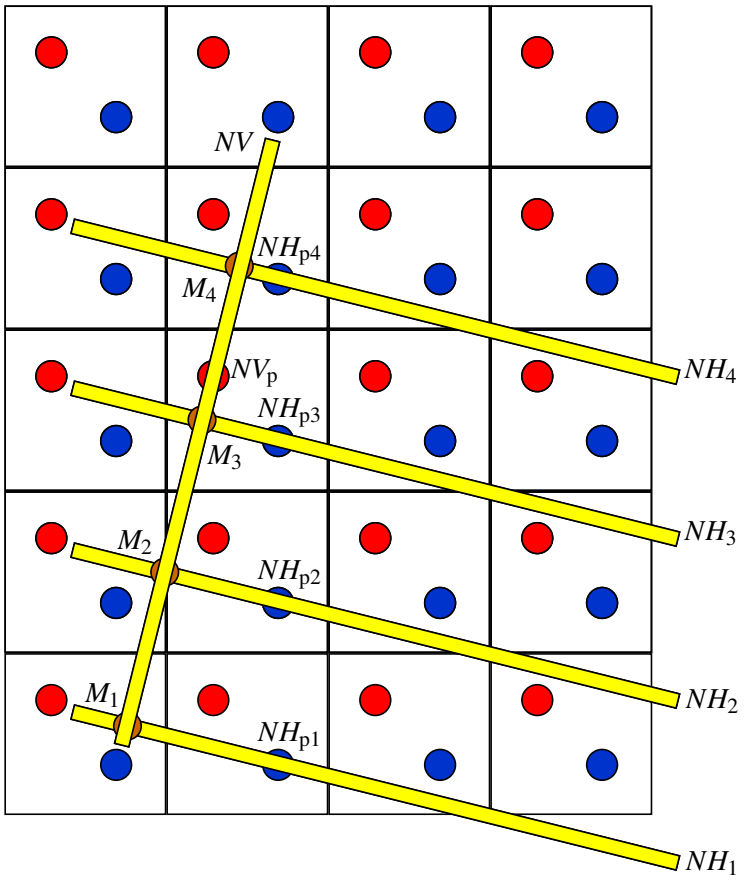


Figure 4.21. Example implementation of 3-input stateful NOR gate on crossbar. M_1, M_2, M_3 are input memristors while M_4 is destination memristor.

Table 4.3. Execution sequence for 3-input stateful NOR gate implementation on crossbar as in Figure 4.21. In write mode the inputs are written in memristors. Using full selection scheme first input logic '0' written to memristors and the destination memristor in step 1. In remaining memristors logic '1' is written in step 2. In step 3, material implication is carried out to perform NOR. Step 4 reads the status of destination memristor as result of NOR operation to be used later.

Mode	**Step**	**NV_p**	**NH_{p1}**	**NH_{p2}**	**NH_{p3}**	**NH_{p4}**
Write	1	$-V_W$	V_W where data is logic '0' , others not selected			V_W
	2	V_W	$-V_W$ where data is logic '1' , others not selected			Not selected
Evaluate	3	Connected to ground through R_G	V_{COND}	V_{COND}	V_{COND}	V_{SET}
Read	4	Voltage is sensed	Not selected			V_R

In evaluate mode, the voltage V_{COND} is applied to one terminal of all input memristors M_1, M_2, M_3 (to terminals $NH_{p1} - NH_{p3}$) and simultaneously voltage V_{SET} is applied to one terminal of

Table 4.4. Steps for implementation of 3-input NOR gate in hybrid crossbar architecture with equivalent circuit shown in Figure 4.20. In write mode, step 1 writes all logic '0' inputs to input memristors and logic '0' to destination memristor. In this table inputs to NOR gate are taken as '010' as an example, hence logic '0' is written to M_1, M_3 and M_4. Step 2 writes all logic '1' inputs to input memristors. Here logic '1' is written to M_2 memristor. In evaluate mode, voltage V_{COND} is applied to one terminal of all input memristors (M_1, M_2, M_3), V_{SET} to the destination memristor (M_4) and the common terminal of all is connected to ground through resistance R_G to perform material implication. In read mode, smaller read voltage V_R is applied to destination memristor M_4 and is sensed at other terminal of it to know the state (i.e. the result of NOR operation). All other memristors are disconnected in this step. Switches $S_1 - S_6$ open when '0' and closed when '1'.

Mode	Step	V_1	V_2	V_3	V_4	V_5	S_1	S_2	S_3	S_4	S_5	S_6
Write	1	V_W				$-V_W$	1	0	1	1	1	0
	2	$-V_W$				V_W	0	1	0	0	1	0
Evaluate	3	V_{COND}			V_{SET}		1	1	1	1	1	1
Read	4				V_R	To sense circuit	0	0	0	0	1	0

destination memristor M_4 (to terminal NH_{p4}). The other common terminal of all (terminal NV_p) is connected to ground through resistance R_G. This mode performs general material implication operation on memristors M_1, M_2, M_3 and M_4 with destination memristor initialized to logic '0'. In this operation, the destination memristor M_4 will toggle to logic '1' only if all input memristors M_1, M_2, M_3 were in logic '0' state before the operation. This is NOR logic operation.

In read mode, voltage V_R is applied to one terminal NH_{p4} of memristor M_4 and is sensed at other terminal NV_p of it to know its state (result) after NOR operation. The voltage V_R is selected such that $|V_R| < |V_{th1,\,M}|$ and $|V_R| < |V_{th2,\,M}|$ so that the state is not changed in read operation.

4.6 Summary

In order to implement stateful NOR logic on memristive crossbar, three basic steps need to be performed on it, namely write, imply and read. Analysis of these operations is carried out in this chapter to investigate the effect on other memristors in crossbar not taking part in logic implementation. Also limitations on size of crossbar to perform stateful NOR operation and to read the state of (destination) memristor correctly are investigated. These limitations are due to sneak path problem common in memristive crossbars. The observations are listed below.

- In floating write scheme, the content of word line and bit line memristors get affected while writing LRS to selected memristor and other memristors are not affected. While writing HRS to selected memristor, the content of bit line and word line memristors are not affected for array size less than 8×8, but other memristors are affected even for small array size. Hence this method is not suitable for writing to memristor.

- 1/3 write scheme is most suitable for writing the data to selected memristor as it does not change the content of other memristors in crossbar. While reading the state of a memristor in a crossbar, the states LRS and HRS become indistinguishable for array size more than 10×10 in worst case scenario where states of all surrounding memristors around it are LRS. This is due to sneak path problem. This limit on size of crossbar can be relaxed by forcing all other memristors in bit line of memristor to be read to HRS but this step will destroy their contents.

- While performing material implication (imply) to implement stateful NOR function on memristive crossbar, the contents of memristors that are on bit lines and word lines of memristors involved in logic (but not part of logic) will not be disturbed if array size is more than 50×50. Remaining memristors are not disturbed for any array size while performing material implication. Although in worst case scenario when all other memristors are in LRS, the array size should be limited to 10×10 due to sneak path problem to implement stateful NOR logic correctly, this limit can be removed by forcing all bit line memristors to HRS similar to read operation.

- If array is to be only used for logic implimentation and not as a memory, then small size array can be used, and changes in the content of bit line and word line memristors can be neglected.

- Stateful NOR function can be implemented on specialized architectures (CMOL or FPNI), where interrupted electodes are used to generate small size arrays in large architectural area.

Sneak path problem is major reason behind the limitations on size of crossbar to implement stateful NOR logic. Overall characteristic of memristive crossbar depends on data stored in it. In the next chapter, CRS crossbar is analyzed for implementation of stateful NOR logic because it is free from sneak path problem.

Chapter 5

Logic Implementation on CRS Crossbar Array

5.1 Introduction

Even though passive crossbar array made up of memristors can be used as non-volatile Random Access Memory (RAM) and in reconfigurable logic circuits [44, 118, 124, 128, 143], the problem of selecting the designated memristor and doing operations on it puts limit on the size of crossbar because of sneak path problem. The sneak path currents through neighboring LRS memristors interfere with proposed operation on designated memristor. Also the effective resistance of crossbar or its section is a function of data (states of memristors) stored in it making it difficult to analyze.

In order to overcome the sneak path problem in a memristive crossbar, Complementary Resistive Switch (CRS) was proposed by Linn et al. [135]. CRS consists of two antiserial memristors and data is stored in the form of LRS-HRS or HRS-LRS. Thus independent of data, the effective resistance of CRS is always high and there is significant reduction in energy consumption of operations in CRS crossbar. Also local fusion of logic and memory, which is an alternative to the conventional von Neumann computer architectures, will avoid the transfer of data in a chip and hence there is further reduction in power consumption [93, 113, 114].

Binary data is stored in the form of HRS-LRS and LRS-HRS making total resistance $R_{HRS}+R_{LRS}$ in both cases. Hence new method needs to be developed to implement stateful NOR using CRSs. The effect of write and logic implementation operations on CRSs not taking part in logic on CRS crossbar needs to be investigated. Also, the limitation on size of crossbar, if any, needs to be determined while reading the state of CRS. The read mechanism for CRS in CRS crossbar will be different form memristor in memristive crossbar because the effective resistance is same in both HRS-LRS and LRS-HRS states of CRS.

This chapter is organized as follows. The switching mechanism in CRS and threshold voltage notations used in the analysis of CRS crossbar are described in first section. Stateful NOR logic gate using CRS is proposed in second section. Diffrent write and read schemes found in literature, and implementation of proposed stateful NOR gate on CRS crossbar are described and analyzed in third section.

5.2 CRS Fundamentals

The concept of Complementary Resistive Switches (CRS) was introduced by Linn et al. [135] in order to solve the sneak path problem. It consists of two antiserially connected memristors A and B as shown in Figure 5.1. Revisiting the bipolar memristor operation, if its initial state is HRS and voltage across it is increased gradually, then it will change its state to LRS as voltage across it crosses the threshold voltage $V_{th1,M}$. If it is in LRS state, then it will not change its state on increasing voltage across it. This is shown in Figure 5.2. Similarly if voltage across memristor is decreased gradually, it will switch to HRS from LRS as soon as voltage across it becomes less than threshold voltage $V_{th2,M}$ ($V_{th2,M}$ is negative). If it is already in HRS then it will not change its state. This is shown in Figure 5.3. For CRS, the HRS-HRS state is possible only during fabrication. To understand the switching action of CRS, first consider Figure 5.4, where voltage across CRS is increased gradually. If it is initially in HRS-LRS state, then almost all voltage will appear across HRS memristor (memristor A). As the voltage across memristor A reaches its threshold voltage $V_{th1,M}$, it will switch to LRS forming LRS-LRS state of CRS. The positive threshold voltage at which CRS switches to LRS-LRS state from HRS-LRS state is shown as $V_{th1,C}$, where 'C' in suffix shows that this threshold voltage is for CRS to distinguish it from memristor threshold voltage.

$V_{th1,C}$ is slightly higher than $V_{th1,M}$ for larger ratio of R_{HRS}/R_{LRS}. At this point, the magnitude of voltage across each memristor is approximately equal to $V_{th1,M}/2$. (Memristor A can not switch back to HRS state). If voltage across CRS is further increased, voltage across each memristor will increase equally until threshold value $V_{th2,M}$ is reached across each memristor in CRS. At this point memristor B will switch from LRS to HRS state and CRS will be in LRS-HRS state thereafter. The positive threshold voltage at which HRS switches to LRS-HRS state from HRS-LRS state is shown as $V_{th2,C}$. The magnitude of $V_{th2,C}$ is approximately equal to $|2V_{th2,M}|$. If initial state of CRS is LRS-HRS in this exercise, it will remain unchanged. Similarly if voltage across the CRS in LRS-HRS state is reduced gradually as shown in Figure 5.5, then at sufficient negative voltage it first switches to LRS-LRS state. This negative threshold voltage is shown as $V_{th3,C}$. Further reduction in voltage causes it to attain HRS-LRS state once the voltage across memristor A crosses threshold voltage $V_{th2,M}$. This negative threshold voltage at which CRS switches from LRS-HRS state to HRS-LRS state is shown as $V_{th4,C}$. If initial state of CRS is HRS-LRS, then reduction of voltage across it does not change its state. The all possible states of CRS has been summarized in Table 5.1.

Linn et al. [135] considered state HRS-LRS of CRS as logic '0' and state LRS-HRS as logic '1'. Thus in crossbar array memory consisting of CRS, irrespective of the stored value at every CRS, the resistance of it will be approximately equal to R_{HRS}. One of the memristors will always be in HRS state similar to CMOS process in transistors. The total resistance of crossbar is independent of information stored. Also, steady state power dissipation will be quite low as in CMOS process.

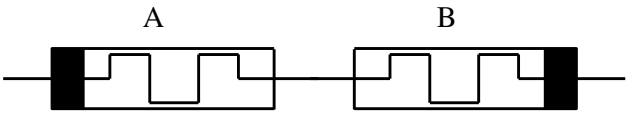


Figure 5.1. Two memristors A and B are connected antiserially to form Complementary Resistive Switch (CRS).

The ideal I-V characteristics (hysteresis curve) for CRS is shown in Figure 5.6 while the simulated I-V characteristic of CRS carried out under DC sweep using TEAM model of memristor is shown in Figure 5.7 and is perfectly antisymmetric.

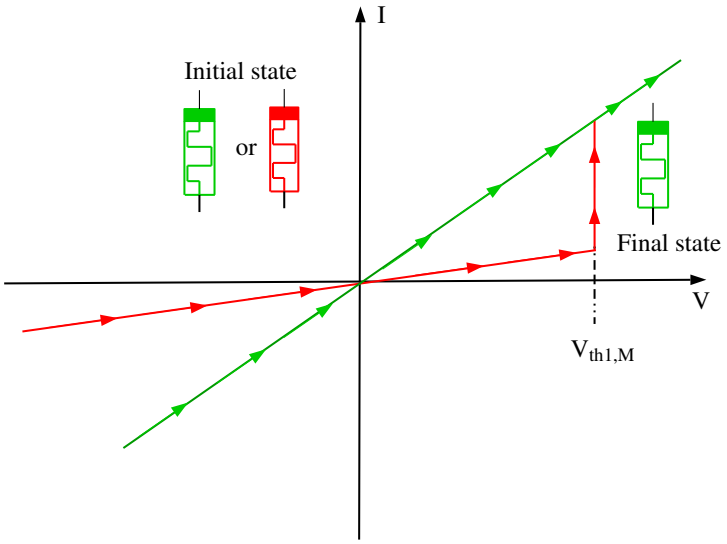


Figure 5.2. When voltage across memristor in HRS state (shown in red color) is increased gradually, it changes to LRS state (shown in green color) as voltage across it crosses positive threshold voltage $V_{th1,M}$. If it is initially in LRS state, then applying positive voltage of any value across it will not change its state.

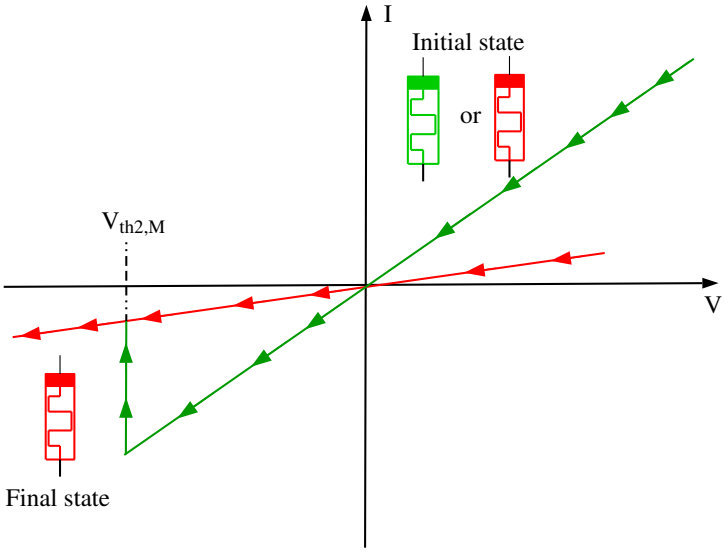


Figure 5.3. When voltage across memristor in LRS state (shown in green color) is reduced gradually, it changes to HRS state (shown in red color) as voltage across it crosses negative threshold voltage $V_{th2,M}$. If it is initially in HRS state, then applying negative voltage of any value across it will not change its state.

5.3 Stateful NOR Gate using CRSs

In CRS, binary information is stored in the form of LRS-HRS and HRS-LRS, and its resultant resistance is independent of its value. In the case of memristor, the information is stored in the

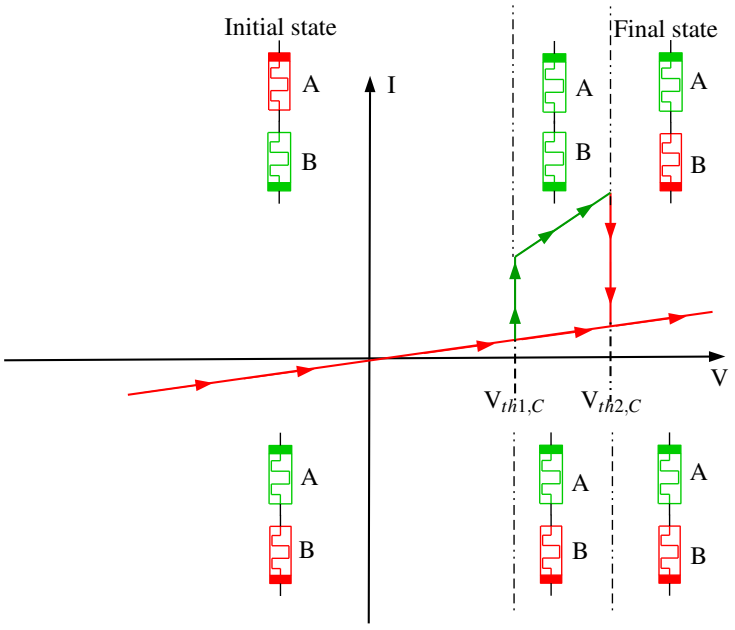


Figure 5.4. When voltage across CRS in HRS-LRS state is increased gradually, it switches first to LRS-LRS and then to LRS-HRS. If CRS is in LRS-HRS state initially, then any positive voltage across it will not changes its state. The positive threshold voltage at which CRS switches from HRS-LRS to LRS-LRS state is shown as $V_{th1,C}$ and from HRS-LRS to LRS-HRS is shown as $V_{th2,C}$, respectively.

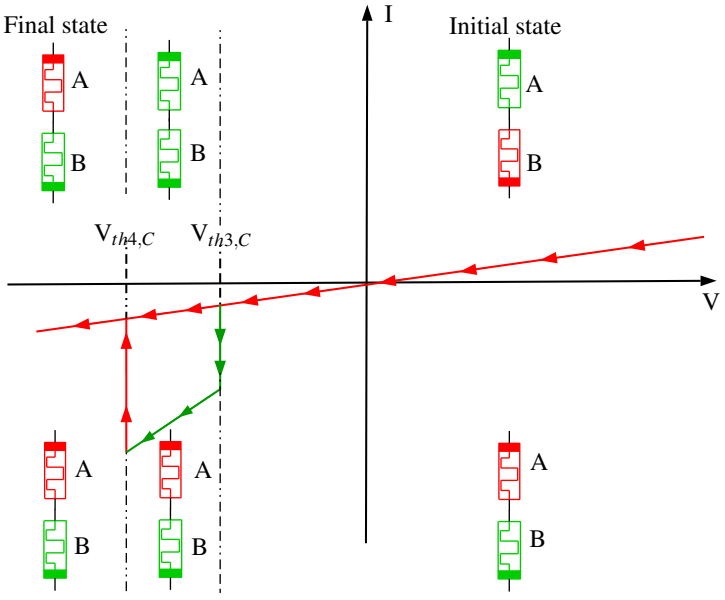


Figure 5.5. When voltage across CRS in LRS-HRS state is decreased gradually, it switches first to LRS-LRS and then to HRS-LRS. If CRS is in HRS-LRS state initially, then any negative voltage across it will not changes its state. The negative threshold voltage at which CRS switches from LRS-HRS to LRS-LRS state is shown as $V_{th3,C}$ and from LRS-HRS to HRS-LRS is shown as $V_{th4,C}$, respectively.

form of HRS and LRS having different values of resistance. Therefore imply logic involving memristors can not be used as it is in CRS. Stateful NOR gate using CRS need to be developed.

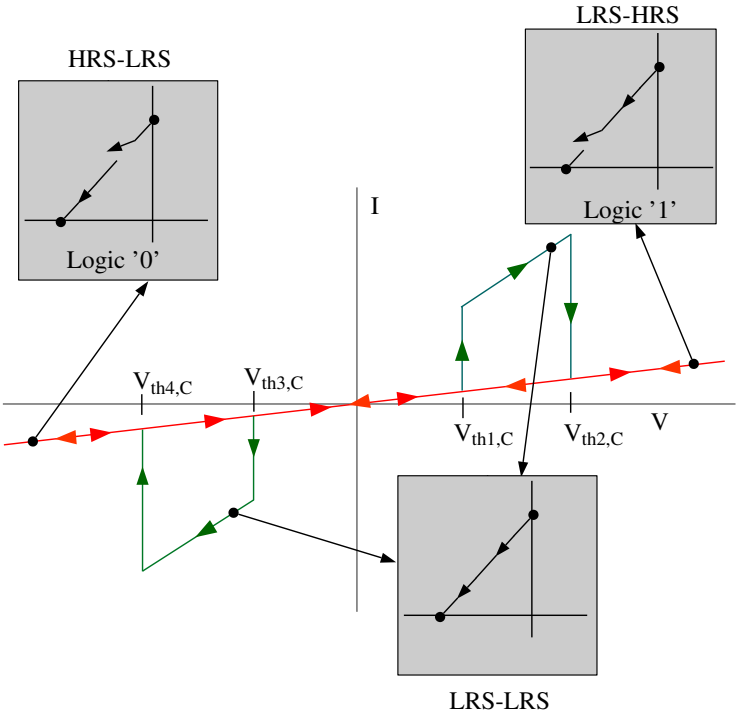


Figure 5.6. Ideal hysteretic I-V characteristic of CRS. If voltage across CRS is increased gradually, it switches to LRS-LRS state from HRS-LRS state when voltage across it equal to threshold voltage $V_{th1,C}$. When voltage across it becomes equal to $V_{th2,C}$, it switches to LRS-HRS state and remains in LRS-HRS state thereafter. When voltage across CRS is decreased gradually, it switches to LRS-LRS state from LRS-HRS state when voltage across it equal to threshold voltage $V_{th3,C}$. When voltage across it becomes equal to $V_{th4,C}$, it switches to HRS-LRS state and remains in HRS-LRS state thereafter. If voltage is less than $V_{th1,C}$ or greater than $V_{th3,C}$, it retains its previous state.

Table 5.1. Possible states of CRS. The total resistance is either R_{HRS} or $2R_{LRS}$ and helps in handling sneak path problem.

CRS state	Memristor A state	Memristor B state	Resistance of CRS
HRS-LRS	HRS	LRS	$R_{HRS} + R_{LRS} \approx R_{HRS}$
LRS-HRS	LRS	HRS	$R_{HRS} + R_{LRS} \approx R_{HRS}$
ON	LRS	LRS	$R_{LRS} + R_{LRS} = 2R_{LRS}$ see figure 5.4 and 5.5
OFF	HRS	HRS	$R_{HRS} + R_{HRS} = 2R_{HRS}$ only during fabrication

The proposed basic 2-input NOR gate with CRS is shown in Figure 5.8 whose structure is similar to imply gate. Consider the voltage applied at CRS Q (which is one of the inputs and destination CRS) is $V_Q = V_x$ and voltage applied at other input CRS P is $V_P = V_x - \Delta V$ for logic evaluation. Also resistance R_G is selected such that $R_{LRS} < R_G < R_{HRS}$. In this case voltage at point C in Figure 5.8 is given by

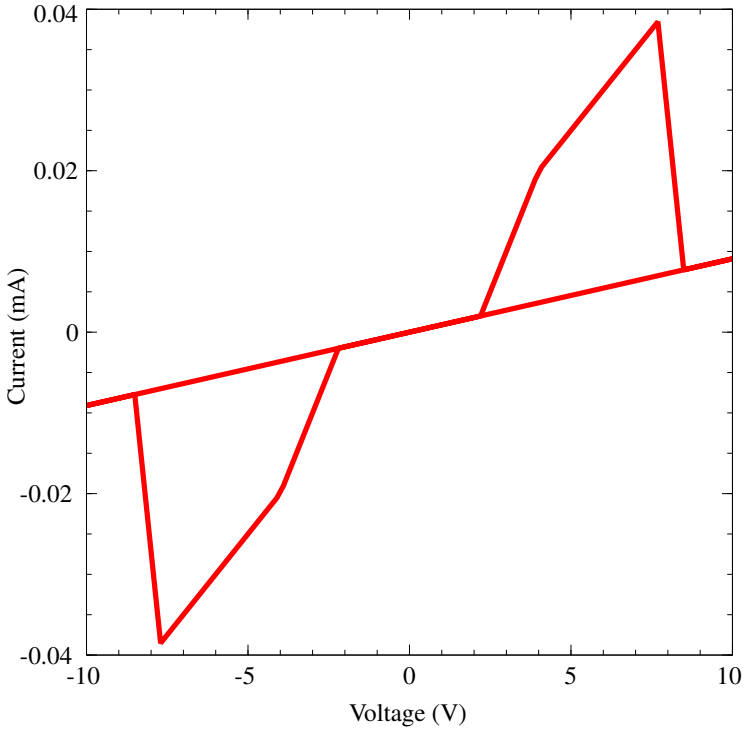


Figure 5.7. The I-V characteristics of CRS under DC sweep showing hysteresis. The simulation is carried out using TEAM model and IV characteristic shows perfect antisymmetry in its shape.

$$V_C = \frac{(V_x - \Delta V)\left(R_G || \left(R_{QM_1} + R_{QM_2}\right)\right)}{\left(R_G || \left(R_{QM_1} + R_{QM_2}\right)\right) + \left(R_{PM_1} + R_{PM_2}\right)} + \frac{V_x \left(R_G || \left(R_{PM_1} + R_{PM_2}\right)\right)}{\left(R_G || \left(R_{PM_1} + R_{PM_2}\right)\right) + \left(R_{QM_1} + R_{QM_2}\right)} \quad (5.1)$$

For each of the sixteen possible cases listed in Table 5.2, the effective voltage across each memristor in CRS can be calculated using (5.1). For example, in Case 1 where $R_{PM_1} = R_{LRS}$, $R_{PM_2} = R_{LRS}$, $R_{QM_1} = R_{LRS}$, $R_{QM_2} = R_{LRS}$, (5.1) becomes

$$V_C = \frac{(V_x - \Delta V)\left(R_G || \left(R_{LRS} + R_{LRS}\right)\right)}{\left(R_G || \left(R_{LRS} + R_{LRS}\right)\right) + \left(R_{LRS} + R_{LRS}\right)} + \frac{V_x \left(R_G || \left(R_{LRS} + R_{LRS}\right)\right)}{\left(R_G || \left(R_{LRS} + R_{LRS}\right)\right) + \left(R_{LRS} + R_{LRS}\right)}, \quad (5.2)$$

which can be approximated as,

$$V_C \approx \frac{(V_x - \Delta V)}{2} + \frac{V_x}{2} \approx V_x - \frac{\Delta V}{2}. \quad (5.3)$$

Therefore, voltage across memristor PM_1 is approximately given as,

$$V_{PM_1} = \left[(V_x - \Delta V) - \left(V_x - \frac{\Delta V}{2}\right)\right] \frac{R_{LRS}}{R_{LRS} + R_{LRS}} \approx -\frac{\Delta V}{4}. \tag{5.4}$$

In a similar manner, voltage across memristor PM_2 is

$$V_{PM_2} \approx -\frac{\Delta V}{4}, \tag{5.5}$$

voltage across QM_1 is

$$V_{QM_1} = \left[V_x - \left(V_x - \frac{\Delta V}{2}\right)\right] \frac{R_{LRS}}{R_{LRS} + R_{LRS}} \approx \frac{\Delta V}{4}, \tag{5.6}$$

and across QM_2 is

$$V_{QM_2} \approx \frac{\Delta V}{4}. \tag{5.7}$$

The complete analysis of remaining cases of stateful NOR gate using CRS is given in Appendix D and effective voltage across each memristor in CRS is summarized in Table 5.2.

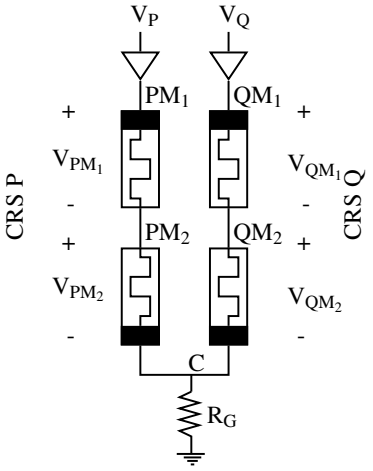


Figure 5.8. 2-input NOR gate using CRS. CRS P and CRS Q are input CRS while CRS Q is destination CRS as well. Voltage V_Q=V_x is applied to destination CRS while voltage V_P=$V_x - \Delta V$ is applied to other input CRS/s.

The input to the gate are either in the form of HRS-LRS or LRS-HRS states. We selected V_x and ΔV such that $V_x > V_{th1,M}$, $V_x > |V_{th2,M}|$, $\frac{\Delta V}{4} > V_{th1,M}$, $\left|\frac{\Delta V}{4}\right| > |V_{th2,M}|$, $(V_x - \Delta V) < V_{th1,M}$ and $(V_x - \Delta V) < |V_{th2,M}|$. One interesting observation is that even if $\Delta V = V_x$ i.e. upper terminal of CRS P is at zero potential, the circuit will work. Hence only one voltage source is sufficient

Table 5.2. Summary of voltage across each memristor in 2-input NOR operation using two CRS as shown in Figure 5.8. Voltage V_x is applied to destination CRS (which is also one of the inputs) while voltage $V_x - \Delta V$ is applied to other input CRS.

Case No.	State of memristor				Voltage across memristor			
	PM_1	**PM_2**	**QM_1**	**QM_2**	**PM_1**	**PM_2**	**QM_1**	**QM_2**
1	LRS	LRS	LRS	LRS	$-\frac{\Delta V}{4}$	$-\frac{\Delta V}{4}$	$\frac{\Delta V}{4}$	$\frac{\Delta V}{4}$
2	LRS	HRS	LRS	LRS	0	$-\Delta V$	0	0
3	HRS	LRS	LRS	LRS	$-\Delta V$	0	0	0
4	HRS	HRS	LRS	LRS	$-\frac{\Delta V}{2}$	$-\frac{\Delta V}{2}$	0	0
5	LRS	LRS	LRS	HRS	0	0	0	ΔV
6	LRS	HRS	LRS	HRS	0	$V_x - \Delta V$	0	V_x
7	HRS	LRS	LRS	HRS	$V_x - \Delta V$	0	0	V_x
8	HRS	HRS	LRS	HRS	$\frac{V_x - \Delta V}{2}$	$\frac{V_x - \Delta V}{2}$	0	V_x
9	LRS	LRS	HRS	LRS	0	0	ΔV	0
10	LRS	HRS	HRS	LRS	0	$V_x - \Delta V$	V_x	0
11	HRS	LRS	HRS	LRS	$V_x - \Delta V$	0	V_x	0
12	HRS	HRS	HRS	LRS	$\frac{V_x - \Delta V}{2}$	$\frac{V_x - \Delta V}{2}$	V_x	0
13	LRS	LRS	HRS	HRS	0	0	$\frac{\Delta V}{2}$	$\frac{\Delta V}{2}$
14	LRS	HRS	HRS	HRS	0	$V_x - \Delta V$	$\frac{V_x}{2}$	$\frac{V_x}{2}$
15	HRS	LRS	HRS	HRS	$V_x - \Delta V$	0	$\frac{V_x}{2}$	$\frac{V_x}{2}$
16	HRS	HRS	HRS	HRS	$\frac{V_x - \Delta V}{2}$	$\frac{V_x - \Delta V}{2}$	$\frac{V_x}{2}$	$\frac{V_x}{2}$

to perform NOR operation on CRSs.All input state combinations of memristors in CRS and the resultant states of memristors (by using Table 5.2) in CRS are listed in Table 5.3.

Table 5.3. All possible state transitions of memristors involved in CRS based 2-input NOR gate shown in Figure 5.8 and using Table 5.2. Voltage V_x is applied to destination CRS while input CRS/s is(are) grounded. $V_x > V_{th1,M}$, $V_x > V_{th2,M}$, $\frac{\Delta V}{4} > V_{th1,M}$, $\left|\frac{\Delta V}{4}\right| > V_{th2,M}$, $(V_x - \Delta V) < V_{th1,M}$ and $(V_x - \Delta V) < V_{th2,M}$ and $\Delta V = V_x$ (since the terminal of input CRS except destination CRS is connected to ground).

Case No.	State of memristor before operation				State of memristor after operation				Remark
	PM_1	PM_2	QM_1	QM_2	PM_1	PM_2	QM_1	QM_2	
1	LRS	LRS	LRS	LRS	HRS	LRS	LRS	HRS	stable
2	LRS	HRS	LRS	LRS	LRS	LRS	LRS	LRS	unstable switch to case 1
3	HRS	LRS	LRS	LRS	HRS	LRS	LRS	LRS	stable
4	HRS	HRS	LRS	LRS	HRS	LRS	LRS	LRS	stable
5	LRS	LRS	LRS	HRS	LRS	LRS	LRS	HRS	stable
6	LRS	HRS	LRS	HRS	LRS	HRS	LRS	HRS	stable
7	HRS	LRS	LRS	HRS	HRS	LRS	LRS	HRS	stable
8	HRS	HRS	LRS	HRS	HRS	HRS	LRS	HRS	stable
9	LRS	LRS	HRS	LRS	LRS	LRS	LRS	LRS	unstable switch to case 1
10	LRS	HRS	HRS	LRS	LRS	HRS	LRS	LRS	unstable switch to case 2
11	HRS	LRS	HRS	LRS	HRS	LRS	LRS	LRS	stable
12	HRS	HRS	HRS	LRS	HRS	HRS	LRS	LRS	unstable switch to case 4
13	LRS	LRS	HRS	HRS	LRS	LRS	LRS	HRS	stable
14	LRS	HRS	HRS	HRS	LRS	HRS	LRS	HRS	stable
15	HRS	LRS	HRS	HRS	HRS	LRS	LRS	HRS	stable
16	HRS	HRS	HRS	HRS	HRS	HRS	LRS	HRS	stable

From Table 5.4, it can be seen that, when both the input CRS states are HRS-LRS (logic '00' input), the destination CRS is in state LRS-LRS. For all other input combinations of states of input CRS, the destination CRS is in HRS-LRS (read in reverse direction). The LRS-LRS state can be converted to LRS-HRS state, if required using feedback mechanism which makes use of sense voltage across R_S, forming voltage divider with CRS to be read during read mechanism.

The concept can be extended to N-input stateful NOR gate, by connecting one terminal of all input CRSs except destination CRS to ground, one terminal of destination CRS to V_x and other common

Table 5.4. Truth table for CRS based 2-input NOR logic. Only possible input combinations are selected from Table 5.3 and stable resultant state of destination CRS is specified in the table. The destination CRS is read in reverse direction i.e. by applying read voltage to bit line of destination CRS instead of word line and the resultant voltage is sensed across resistance R_S connected to word line instead of bit line.

Sr. No.	Input states of CRS		Destination CRS after operation (result)
	CRS P	CRS Q	CRS Q
1	HRS-LRS	HRS-LRS	LRS-LRS
2	HRS-LRS	LRS-HRS	LRS-HRS
3	LRS-HRS	HRS-LRS	LRS-HRS
4	LRS-HRS	LRS-HRS	LRS-HRS

terminal of all CRSs to ground through resistance R_G. The major differences between memristor based stateful NOR gate and CRS based stateful NOR gate are as follows. For N-input stateful NOR using memristor requires N+1 memristors because destination memristor is different from input memristors. Also destination memristor needs to be initialized to HRS before performing NOR operation. CRS based N-input stateful NOR gate requires N CRSs because destination CRS is one of the input CRSs and initialization step is not required for destination CRS. Memristor based stateful NOR gate requires two voltage sources (V_{SET} and V_{COND}) for its operation while CRS based gate requires single voltage source (V_x) for its operation.

5.4 Analysis of Operations on CRS Crossbar

Structure of CRS crossbar is similar to memristive crossbar with memristors replaced by CRSs. In order to implement stateful NOR gate on CRS crossbar, first the inputs on which NOR operation has to be performed should be written to CRSs in the form of HRS-LRS and LRS-HRS. Then logic operation explained in the previous section should be performed and finally the result of NOR operation should be read. These operations are described and analyzed in this section in order to find out their effect on CRSs not involved in logic implementation and, to find out limitation on the size of CRS crossbar, if any, for correct logic implementation. The resistive equivalent symbol for CRS is shown in Figure 5.9 and is used in the analysis. The dot in resistive equivalent symbol shows the side of memristor A in CRS made up of memristors A and B.

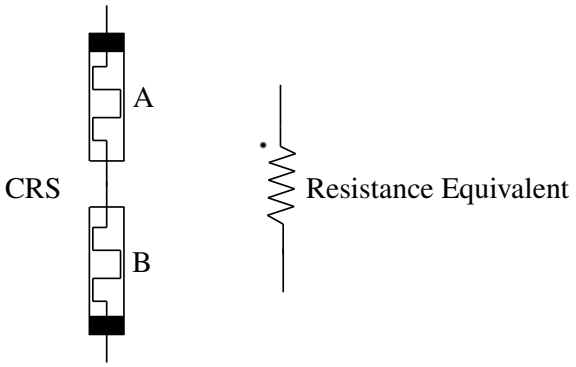


Figure 5.9. CRS resistance equivalent symbol used in crossbar equivalent circuit. The dot in equivalent resistance notation indicate the side of memristor A in CRS having memristors A and B. The state of CRS has been specified as HRS-LRS or LRS-HRS or LRS-LRS, the first being the state of memristor A and second is the state of memristor B.

5.4.1 Write Operation

In order to write LRS-HRS state in CRS, voltage $V_W > V_{th2,C}$ should be applied across selected CRS. To write HRS-LRS state to CRS, voltage $-V_W < V_{th4,C}$ should be applied across selected CRS. In both cases, the magnitude of voltage across unselected CRSs should be less than $V_{th1,C}$ and greater than $V_{th3,C}$ so that their contents are not destroyed. The different write schemes for CRS crossbar are explained below.

5.4.1.1 Floating Write Scheme

In this method write voltage V_{WRITE} is applied to the selected word/bit line while selected bit/word line is grounded to ensure the selected CRS cell has enough voltage to perform the successful write operation, while the remaining word lines and bit lines are kept floating [35, 135] similar to floating write scheme for memristive crossbar explained in the previous chapter. This scheme for CRS crossbar is shown in Figure 5.10 and its equivalent resistive circuit is shown in Figure 5.11. The CRSs are categorized as shown in Table 5.5 for the analysis along with notations for voltage across them. If the size of the crossbar array is $m \times n$ with m rows and n columns, then there will be $n-1$ CRS_{WORD} shown as Region-I, $m-1$ CRS_{BIT} shown as Region-II, $(m-1)(n-1)$ CRS_{NSWB} shown as Region-III and one CRS_{SEL} as shown in Figure 5.10.

Let I_1 be the current flowing through each CRS_{WORD} during write operation as shown in Figure 5.11. Then as equivalent resistance of each CRS is approximately equal to R_{HRS},

Table 5.5. Notations used in the analysis of operations on CRS crossbar.

CRS notation	Description	Voltage across CRS
CRS_{WORD}	CRS on selected word line except CRS selected for operation	V_{WORD}
CRS_{BIT}	CRS on selected bit line except CRS selected for operation	V_{BIT}
CRS_{NSWB}	CRS not on selected word and bit lines	V_{NSWB}
CRS_{SEL}	Selected CRS for operation	V_{SEL}

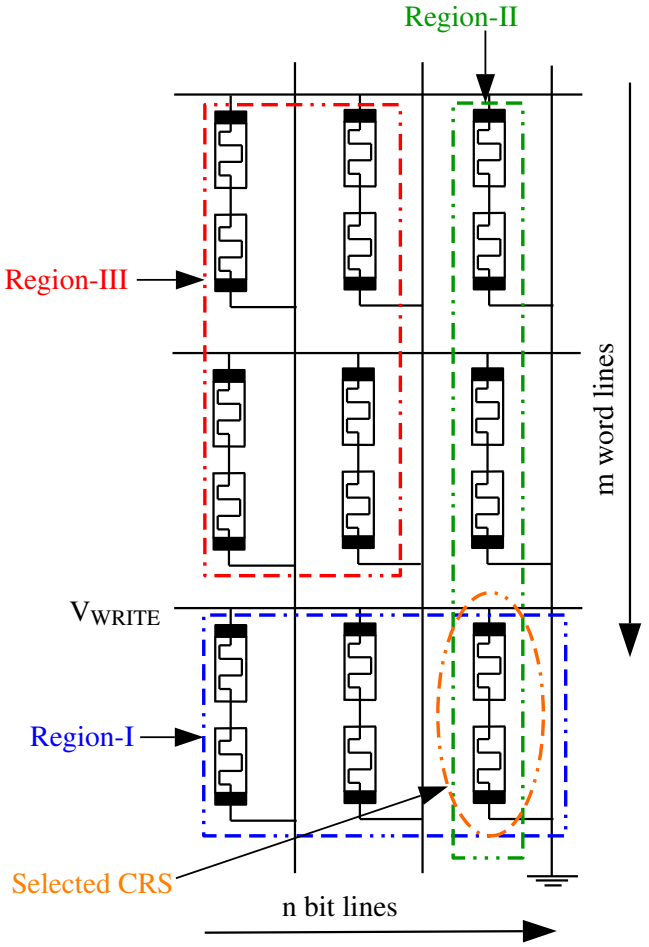


Figure 5.10. In floating write scheme of CRS crossbar, the word line of CRS to be written is driven by voltage V_{WRITE}, its bit line is grounded while unselected word lines and bit lines are kept floating [144].

$$V_{WRITE} = I_1 R_{HRS} + \frac{I_1}{(m-1)} R_{HRS} + \frac{I_1}{(m-1)} (n-1) R_{HRS}. \tag{5.8}$$

Therefore,

$$I_1 = \frac{(m-1)V_{WRITE}}{(m+n-1)R_{HRS}}. \tag{5.9}$$

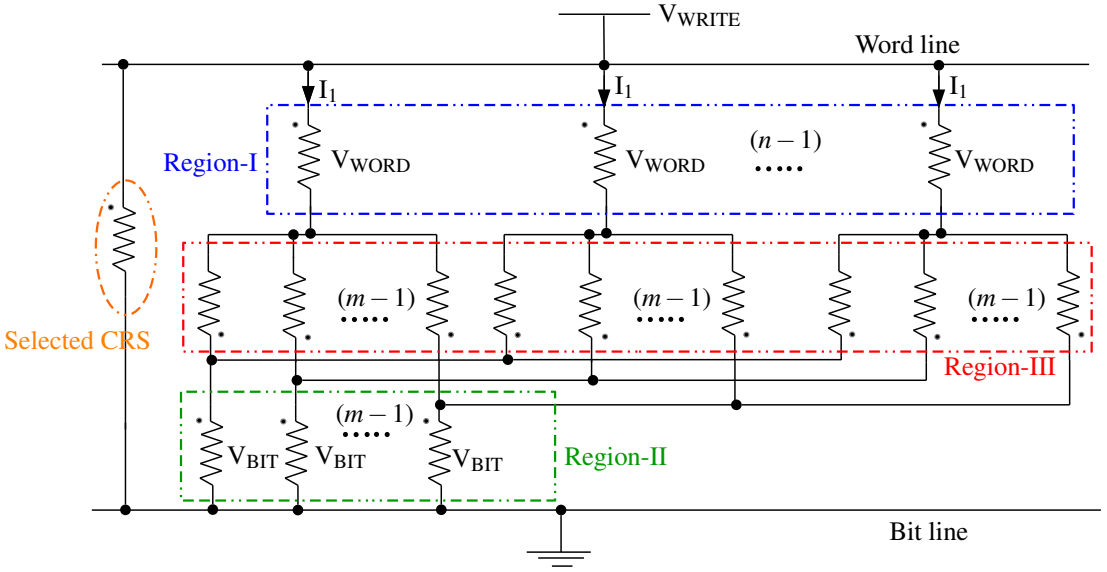


Figure 5.11. Resistive equivalent circuit for floating write scheme for CRS crossbar array shown in Figure 5.10. The size of array is $m \times n$, where m and n indicate number of rows and columns, respectively.

Voltage across unselected CRSs in Region-I will be

$$V_{\mathrm{WORD}} = I_1 R_{\mathrm{HRS}} = \frac{(m-1)V_{\mathrm{WRITE}}}{(m+n-1)}. \tag{5.10}$$

Voltage across unselected CRSs in Region-II will be

$$V_{\mathrm{BIT}} = \frac{I_1}{(m-1)}(n-1)R_{\mathrm{HRS}} = \frac{(n-1)V_{\mathrm{WRITE}}}{(m+n-1)}, \tag{5.11}$$

and across unselected CRSs in Region-III will be

$$V_{\mathrm{NSWB}} = \frac{I_1}{m-1}R_{\mathrm{HRS}} = \frac{V_{\mathrm{WRITE}}}{(m+n-1)}. \tag{5.12}$$

Resulting voltage across unselected CRSs ($\mathrm{CRS_{WORD}}$, $\mathrm{CRS_{BIT}}$, $\mathrm{CRS_{NSWB}}$) due to application of voltage $\mathrm{V_{WRITE}}$ across selected CRS ($\mathrm{CRS_{SEL}}$), should satisfy the conditions

$$V_{\mathrm{th3,C}} < V_{\mathrm{WORD}} < V_{\mathrm{th1,C}}, \tag{5.13}$$

$$V_{\mathrm{th3,C}} < V_{\mathrm{BIT}} < V_{\mathrm{th1,C}}, \tag{5.14}$$

and

$$V_{\text{th3,C}} < V_{\text{NSWB}} < V_{\text{th1,C}}, \tag{5.15}$$

in order to avoid the state change of unselected CRSs.

For writing LRS-HRS state to selected CRS, $V_{WRITE}=+V_W$ such that $V_W > V_{th2,C}$, should be applied to its word line while its bit line should be grounded. Out of unselected CRSs, CRS_{WORD} and CRS_{BIT} can change their state if they are in HRS-LRS and the resulting voltage across them is greater than $V_{th1,C}$ for applied write voltage V_W. CRS_{NSWB} can change its state if it is in LRS-HRS and resulting voltage due to application of V_W across it is less than $V_{th3,C}$ (because of its orientation in the resistive equivalent circuit shown in Figure 5.11). If CRS_{WORD} and CRS_{BIT} are in LRS-HRS or CRS_{NSWB} is in HRS-LRS state, application of V_W will not change their states. Using (5.10), (5.11) and (5.12) [144], the resulting voltage across each unselected CRS, due to application of voltage V_W across selected CRS, as function of array size $m \times n$ with $m = n$ is plotted in Figure 5.12. It can be seen that CRS_{NSWB} will not change their state for any array size as magnitude of maximum voltage across them is always less than $|V_{th3,C}|$. CRS_{WORD} and CRS_{BIT} may change their states to LRS-LRS states if they are in HRS-LRS states as the resulting voltage (due to application of voltage V_W across selected CRS) across them may exceed $V_{th1,C}$. If they are in LRS-HRS states, application of voltage V_W across CRS_{SEL} will not change their states. The change of state to LRS-LRS can be recovered back using feedback mechanism explained later in this chapter.

For writing HRS-LRS state to selected CRS, $V_{WRITE}=-V_W$ such that $-V_W < V_{th4,C}$ should be applied to its word line while its bit line should be grounded. Out of unselected CRSs, CRS_{WORD} and CRS_{BIT} can change their states if they are in LRS-HRS and the resulting voltage across them is less than $V_{th3,C}$ for applied write voltage $-V_W$ across CRS_{SEL}. CRS_{NSWB} can change its state if it is in HRS-LRS and the resulting voltage (because of application of $-V_W$ across CRS_{SEL}) across it is greater than $V_{th1,C}$. If CRS_{WORD} and CRS_{BIT} are in HRS-LRS states or CRS_{NSWB} is in LRS-HRS state, application of $-V_W$ will not change their states. The resulting voltage across each unselected CRS as function of array size $m \times n$ with $m = n$ is plotted in Figure 5.13 using (5.10), (5.11) and (5.12) [144]. It can be seen that CRS_{NSWB} will not change their state for any array size as magnitude of maximum voltage across them is always less than $|V_{th1,C}|$. CRS_{WORD} and CRS_{BIT} may change their states to LRS-LRS states if they are in LRS-HRS states as the resulting

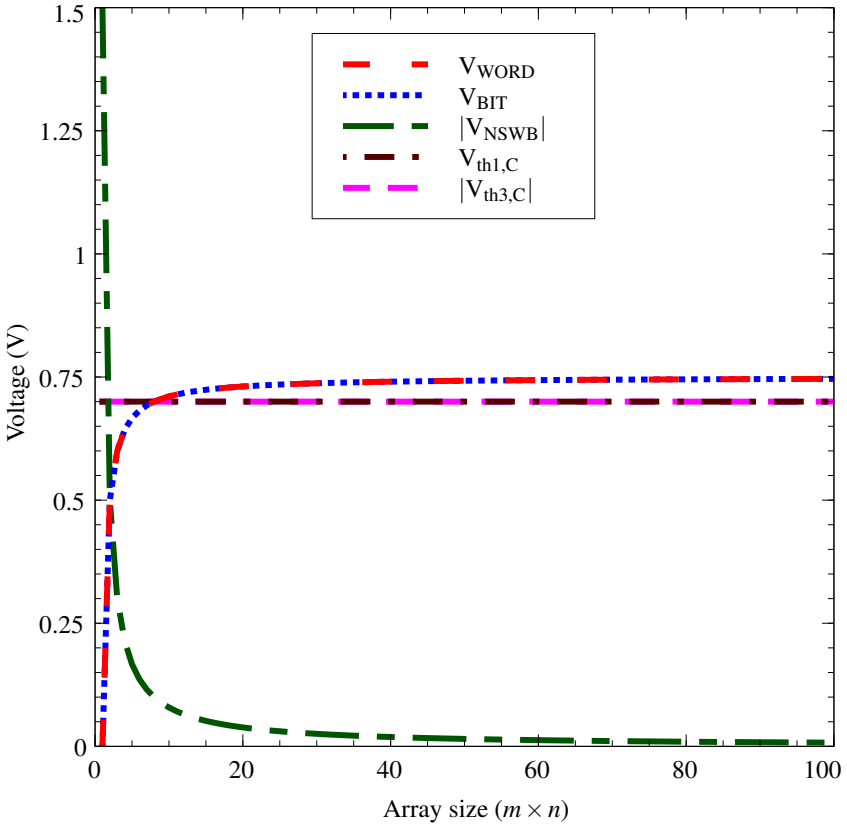


Figure 5.12. Voltage variations across unselected CRSs as function of array size ($m \times n$) with $m = n$ for floating write scheme in order to write LRS-HRS in selected CRS. CRS_{NSWB} will not change their states for any array size as magnitude of maximum voltage across them is always less than $|V_{th3,C}|$. CRS_{WORD} and CRS_{BIT} may change their states to LRS-LRS if they are in HRS-LRS states as the resulting voltage (due to application of voltage V_W across selected CRS) across them may exceed $V_{th1,C}$. If they are in LRS-HRS states, application of voltage V_W across CRS_{SEL} will not change their states. The change of state to LRS-LRS can be recovered back using feedback mechanism. In this plot $V_{th1,C}$ =0.7 V, $V_{th3,C}$=−0.7 V, V_W=1.5 V because $V_{th2,C}$= 1.4 V.

voltage (due to application of voltage V_W across selected CRS) across them may exceed $V_{th3,C}$. If they are in HRS-LRS states, application of voltage V_W across CRS_{SEL} will not change their states. Again, the change of state to LRS-LRS can be recovered back using feedback mechanism explained later in this chapter. Floating write scheme consumes a small amount of power during write operation [145].

5.4.1.2 1/3 Write Scheme

The floating write scheme explained in previous subsection may change the state of unselected CRS cells (CRS_{WORD} and CRS_{BIT}) [35, 145], evident from Figures 5.12 and 5.13 because voltage

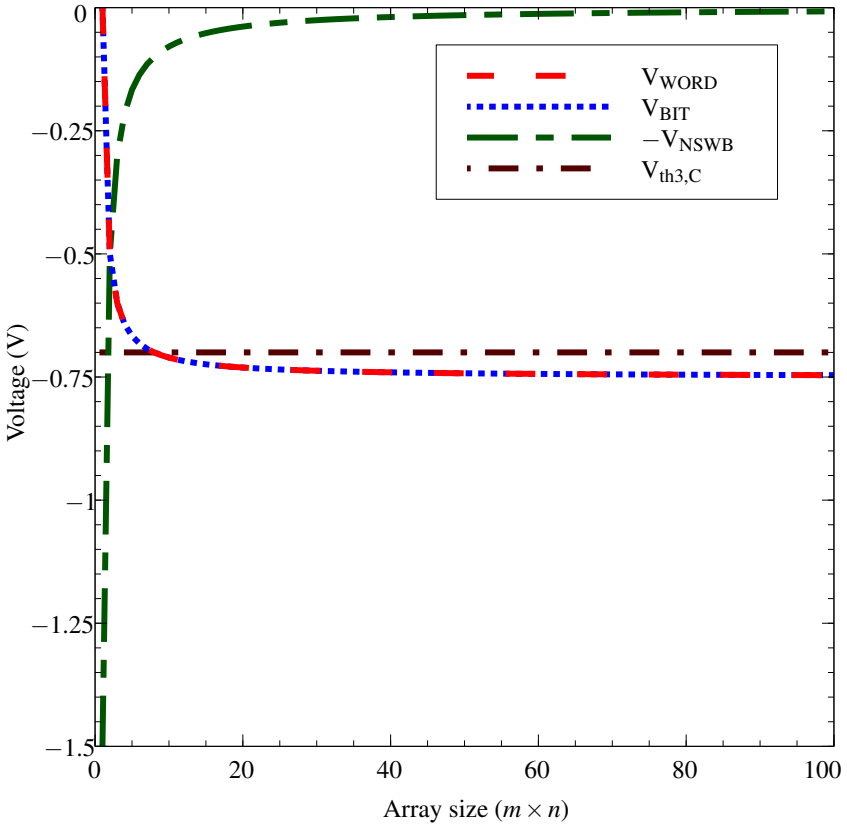


Figure 5.13. Voltage variations across unselected CRSs as function of array size ($m \times n$) with $m = n$ for floating write scheme in order to write HRS-LRS in selected CRS. CRS_{NSWB} will not change their states for any array size as magnitude of maximum voltage across them is always less than $|V_{th1,C}|$. CRS_{WORD} and CRS_{BIT} may change their states to LRS-LRS if they are in LRS-HRS states as the resulting voltage (due to application of voltage $-V_W$ across selected CRS) across them may be less than $V_{th3,C}$. If they are in HRS-LRS states, application of voltage V_W across CRS_{SEL} will not change their states. The change of state to LRS-LRS can be recovered back using feedback mechanism. In this plot $V_{th1,C}$ =0.7 V, $V_{th3,C}$=−0.7 V, $-V_W$=−1.5 V because $V_{th4,C}$= −1.4 V.

across them rises to $\pm V_W/2$ even for small array size and cross threshold values $V_{th1,C}$ or $V_{th3,C}$ ($V_{th1,C}$ is less than $V_W/2$ or $V_{th3,C}$ is greater then $-V_W/2$). In 1/3 write scheme [35], write voltage V_{WRITE} is applied to the word line of CRS to be written while its bit line is grounded. At the same time, voltage $V_{WRITE}/3$ is applied to all unselected word lines and voltage $2V_{WRITE}/3$ to all unselected bit lines as shown in Figure 5.14. The effective voltage across each unselected CRS will be $|V_{WRITE}/3|$ while across selected CRS will be V_{WRITE} . This 1/3 write scheme can avoid the unselected CRSs state changes. However, voltages are applied to every word line and bit line, which will not only consume more power [145], but also have a complex write scheme control circuitry.

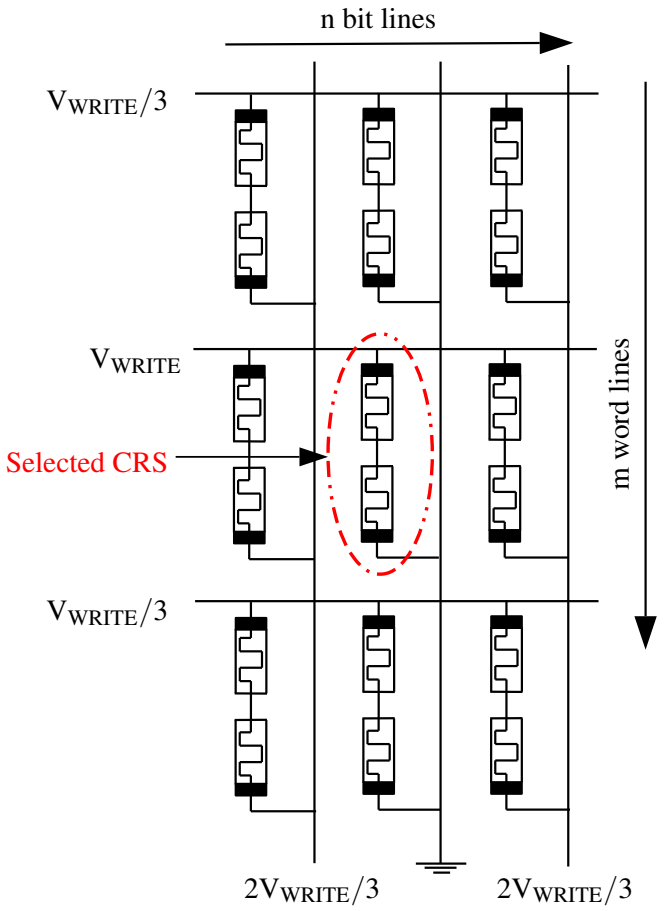


Figure 5.14. In 1/3 write scheme of CRS crossbar, the word line of CRS to be written is driven by voltage V_{WRITE}, its bit line is grounded while unselected word lines are driven with voltage $V_{WRITE}/3$ and bit lines are are driven with voltage $2V_{WRITE}/3$. The effective voltage across each unselected CRS will be $|V_{WRITE}/3|$ while across selected CRS will be V_{WRITE} [144].

5.4.1.3 Configuration Row based 1/3 Write Scheme

Configuration row based 1/3 write scheme [144] is shown in Figure 5.15, in which one of the rows of CRS is designated as configuration row. All configuration CRS cells are written with state LRS-LRS (ON). Write voltage V_{WRITE} is applied to word line of CRS to be written while its bit line is grounded. The unselected word lines except configuration word line are driven with $V_{WRITE}/3$ and the configuration word line is driven with voltage $2V_{WRITE}/3$. As configuration CRSs are in LRS-LRS during write operation, the configuration word line is effectively shorted with bit lines and thus are driven with voltage $2V_{WRITE}/3$. This is similar to conventional 1/3 write scheme with only one $2V_{WRITE}/3$ voltage source instead of $n-1$ and hence has less power consumption. Since all configuration CRS cells are in state LRS-LRS (low resistance state), compared with other CRS cells which are in either LRS-HRS or HRS-LRS states, the magnitude

of voltages across all configuration CRSs are less than $V_{th1,C}$ or greater then $V_{th3,C}$. Therefore configuration CRS cells can remain in LRS-LRS state. The extra cost is the configuration row and large driver transistors to switch all configuration CRSs to LRS-LRS states (or extra time to switch them sequentially to LRS-LRS states). The resistive equivalent circuit for this configuration is shown in Figure 5.16. The principle of superposition is applied to find out the effective voltage

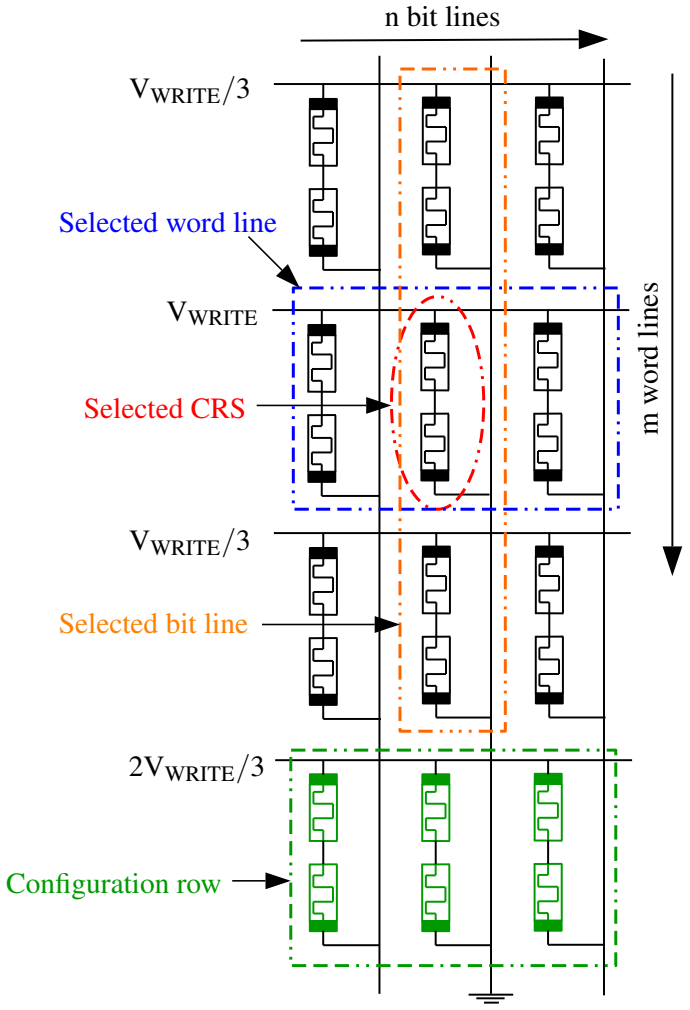


Figure 5.15. In configuration row based 1/3 write scheme of CRS crossbar, the selected word line is driven by voltage V_{WRITE}, selected bit line is grounded. Additional configuration row of CRS is added. During write operation all CRS in configuration row will be in LRS-LRS state and their word line is driven by $2V_{WRITE}/3$. Remaining unselected word lines are driven with voltage $V_{WRITE}/3$ [144].

across each CRS due to V_{WRITE}, $2V_{WRITE}/3$ and $V_{WRITE}/3$.

1. Effect of V_{WRITE}

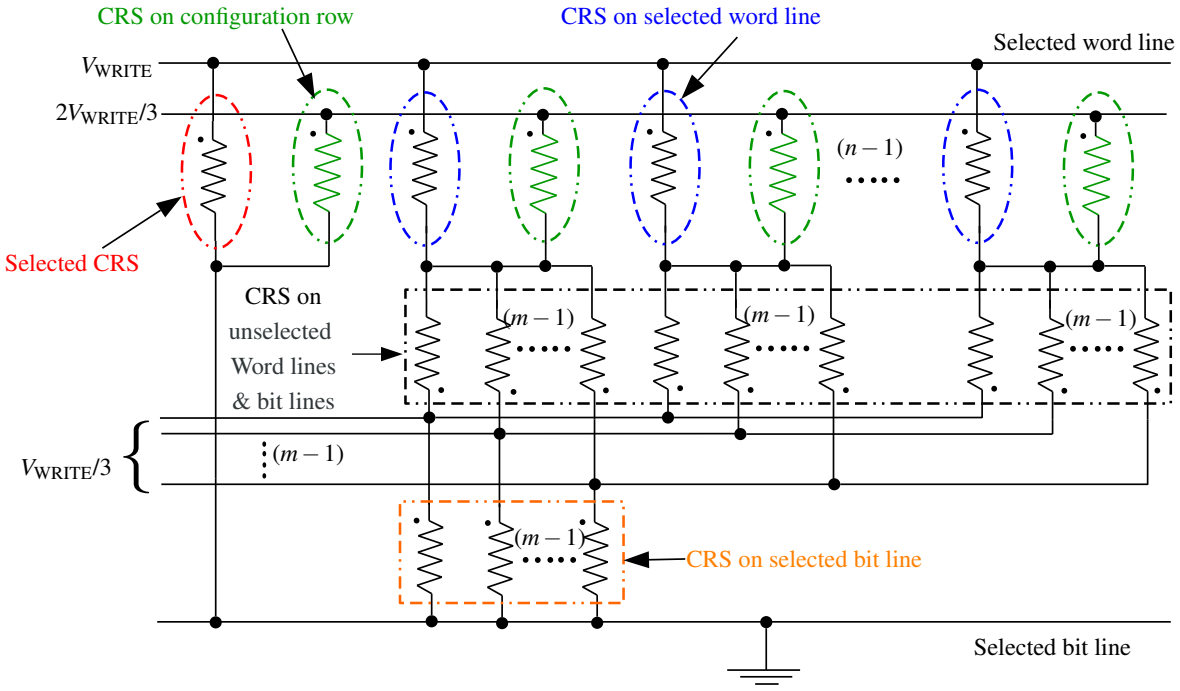


Figure 5.16. Resistive equivalent circuit for configuration row based 1/3 write scheme for CRS crossbar shown in Figure 5.23. The dot notation shown in figure 5.9 is used to draw equivalent circuit.

Due to V_{WRITE}, voltage across CRS_{SEL} is

$$V_{SEL,1} = V_{WRITE}, \tag{5.16}$$

across CRS_{WORD} is

$$V_{WORD,1} = \frac{R_{HRS} + (m-1)R_{LRS}}{R_{HRS} + mR_{LRS}} V_{WRITE}, \tag{5.17}$$

across CRS_{NSWB} is

$$V_{NSWB,1} = -\frac{R_{LRS}}{R_{HRS} + mR_{LRS}} V_{WRITE}, \tag{5.18}$$

across CRS_{BIT} is

$$V_{BIT,1} = 0, \tag{5.19}$$

and across CRS_{CONF} is

$$V_{CONF,1} = -\frac{R_{LRS}}{R_{HRS} + mR_{LRS}} V_{WRITE}. \tag{5.20}$$

2. Effect of $2V_{WRITE}/3$

Due to $2V_{WRITE}/3$, voltage across CRS_{SEL} is

$$V_{SEL,2} = 0, \tag{5.21}$$

across CRS_{WORD} is

$$V_{WORD,2} = -\frac{2R_{HRS}}{3(R_{HRS}+mR_{LRS})}V_{WRITE}, \tag{5.22}$$

across CRS_{NSWB} is

$$V_{NSWB,2} = -\frac{2R_{HRS}}{3(R_{HRS}+mR_{LRS})}V_{WRITE}, \tag{5.23}$$

across CRS_{BIT} is

$$V_{BIT,2} = 0, \tag{5.24}$$

and across CRS_{CONF} is

$$V_{CONF,2} = \frac{2mR_{LRS}}{3(R_{HRS}+mR_{LRS})}V_{WRITE}. \tag{5.25}$$

3. Effect of $V_{WRITE}/3$

   Similarly, due to $V_{WRITE}/3$, these voltages are

$$V_{SEL,3} = 0, \tag{5.26}$$

$$V_{WORD,3} = -\frac{(m-1)R_{LRS}}{3(R_{HRS}+mR_{LRS})}V_{WRITE}, \tag{5.27}$$

$$V_{NSWB,3} = \frac{R_{HRS}}{3(R_{HRS}+mR_{LRS})}V_{WRITE}, \tag{5.28}$$

$$V_{BIT,3} = \frac{1}{3}V_{WRITE}, \tag{5.29}$$

   and

$$V_{CONF,3} = -\frac{(m-1)R_{LRS}}{3(R_{HRS}+mR_{LRS})}V_{WRITE}, \tag{5.30}$$

   respectively.

The cumulative effect will be as follows. The overall voltage across CRS_{SEL} is

$$V_{SEL} = V_{WRITE}, \tag{5.31}$$

across CRS_{WORD} is

$$V_{WORD} = \frac{R_{HRS} + 2(m-1)R_{LRS}}{3(R_{HRS} + mR_{LRS})} V_{WRITE}, \tag{5.32}$$

across CRS_{NSWB} is

$$V_{NSWB} = -\frac{R_{HRS} + 3R_{LRS}}{3(R_{HRS} + mR_{LRS})} V_{WRITE}, \tag{5.33}$$

across CRS_{BIT} is

$$V_{BIT} = \frac{1}{3} V_{WRITE}, \tag{5.34}$$

and across CRS_{CONF} is

$$V_{CONF} = \frac{(m-2)R_{LRS}}{3(R_{HRS} + mR_{LRS})} V_{WRITE}. \tag{5.35}$$

Voltage variations across different CRS cells as function of array size $m \times n$ with $m = n$ is plotted using (5.31), (5.32), (5.33), (5.34) and (5.35) for writing LRS-HRS in selected CRS with $V_{WRITE} = V_W$ in Figure 5.17, and for writing HRS-LRS to selected CRS with $V_{WRITE} = -V_W$ in Figure 5.18. In writing LRS-HRS or HRS-LRS to selected CRS, voltages across CRS_{BIT}, CRS_{NSWB} and CRS_{CONF} are always less than $V_{th1,C}$ or greater than $V_{th3,C}$ and hence they retain their states for any array size in configuration row based 1/3 write scheme. Only CRS_{WORD} may switch to LRS-LRS state (if it is in HRS-LRS state while writing LRS-HRS to selected CRS or in LRS-HRS state while writing HRS-LRS state to selected CRS) for array size approximately greater than 7000 $\times$ 7000. However, the original state of it can be retained back using feedback mechanism explained later in this chapter.

The total power consumption for write operation in CRS crossbar is given by

$$P_{total,CRS} = P_{SEL} + P_{BIT} + P_{WORD} + P_{NSWB} \tag{5.36}$$

with additional component of P_{CONF} in case of configuration row based 1/3 write scheme. The power consumption is function of array size (m $\times$ n) and is plotted for different write schemes in Figure 5.19 for symmetric array. The power also depends on threshold voltages of CRS as it directly dictates the minimum value of write voltage to be used. In CRS crossbar array however, the power is independent of state of CRS as in logic '0' or logic '1' state, the effective resistance is $R_{HRS}+R_{LRS}$. The power consumption in floating write scheme is comparatively less but it does alter the states of other unselected CRS and hence not useful. 1/3 write scheme is widely used

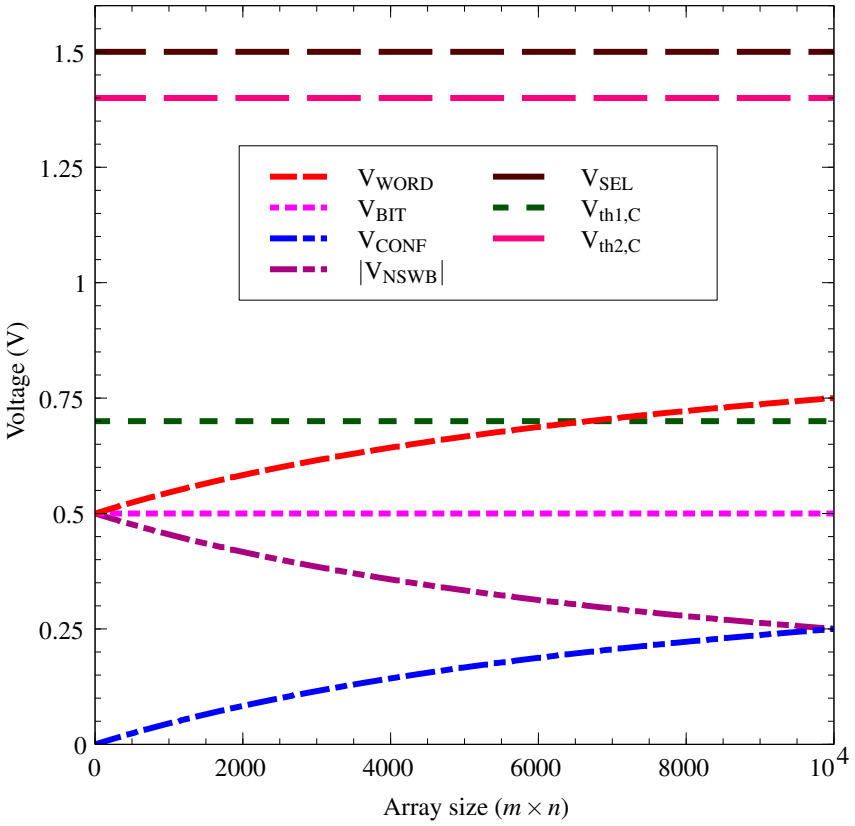


Figure 5.17. Voltage variations across different CRSs as function of array size ($m \times n$) with $m = n$ for configuration row based 1/3 write scheme in order to write LRS-HRS to selected CRS. Any of the unselected CRSs apart from CRS_{WORD} will not change its state as magnitude of voltage across each of such unselected CRS is less than $V_{th1,C}$. Configuration CRS will also remain in LRS-LRS state as voltage across them is less than $V_{th1,C}$. CRS_{WORD} can change its state to LRS-LRS if it is in HRS-LRS for array size more than around 7000 $\times$ 7000, but its HRS-LRS state can be retained back using feedback mechanism. In this plot $V_{th1,C}$ =0.7 V, V_W=1.5 V and $V_{th2,C}$= 1.4 V. Only selected CRS has voltage greater than $V_{th2,C}$ and hence LRS-HRS state will be written to it.

to write data in selected CRS/s even though the power consumption is more than floating write scheme because it does not change the states of unselected CRSs. Configuration row based 1/3 write scheme has slightly improved power consumption over 1/3 write scheme at the cost of area overhead for configuration row of CRSs.

5.4.2 Read Operation

This section describes the different read schemes used to read the state of selected CRS. The state of CRS is either LRS-HRS or HRS-LRS and hence same resistance is presented by CRS irrespective of its content.

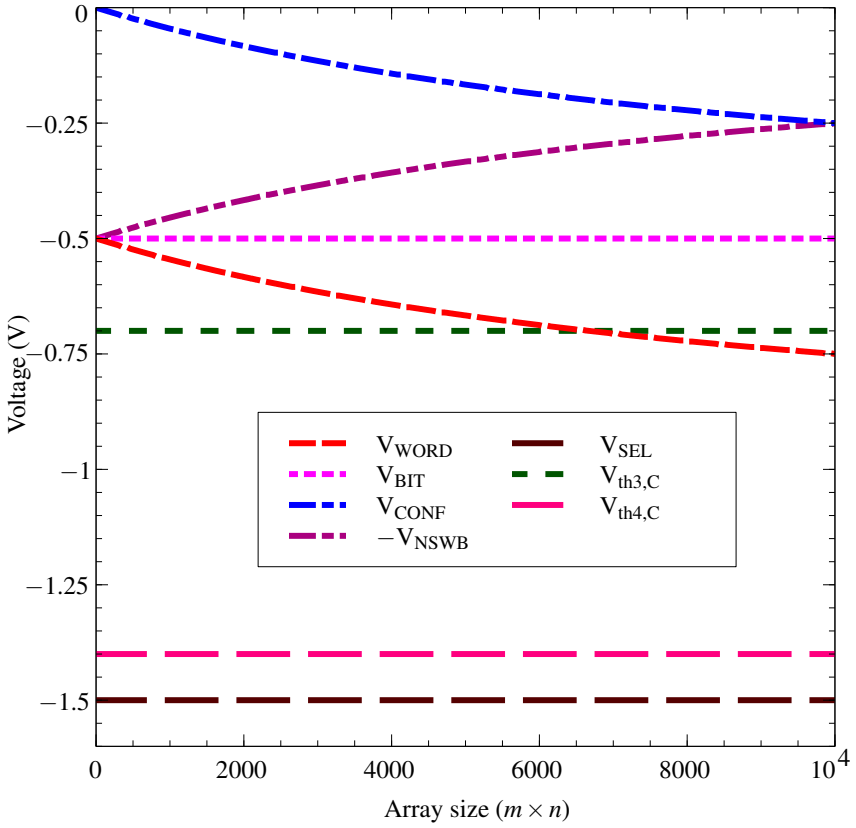


Figure 5.18. Voltage variations across different CRSs as function of array size ($m \times n$) with $m = n$ for configuration row based 1/3 write scheme in order to write HRS-LRS to selected CRS. Any of the unselected CRSs apart from CRS_{WORD} will not change its state as voltage across each of such unselected CRS is greater than $V_{th3,C}$. Configuration CRS will also remain in LRS-LRS state as voltage across them is greater than $V_{th3,C}$. CRS_{WORD} can change its state to LRS-LRS if it is in LRS-HRS for array size more than around 7000 × 7000, but its LRS-HRS state can be retained back using feedback mechanism. In this plot $V_{th3,C} = -0.7$ V, $-V_W$=−1.5 V and $V_{th4,C}$=− 1.4 V. Only selected CRS has voltage less than $V_{th4,C}$ and hence HRS-LRS state will be written to it.

5.4.2.1 Conventional Read Scheme

A conventional CRS read scheme takes three steps to read a value (resistance) stored in CRS [35, 135, 146]. In the first step, set voltage V_S is applied to the word line of CRS to be read and its bit line is connected to ground through sense resistance R_S. The resistance R_S and CRS form voltage divider. Value of voltage V_S is selected such that $V_{th1,C} < V_{CRS} < V_{th2,C}$, where V_{CRS} is voltage across CRS to be read. If the state of CRS is HRS-LRS before application of voltage V_S, then it will change to LRS-LRS after application of V_S. If it is in LRS-HRS before, it will not change its state. In second step, read voltage V_R (with condition $V_R < V_{th2,C}$ and V_R can be equal to V_S) is applied to word line of CRS to be read and its effect sensed across resistance R_S connected to bit line of CRS to be read. The resistance R_S and CRS to be read form voltage divider. If sensed

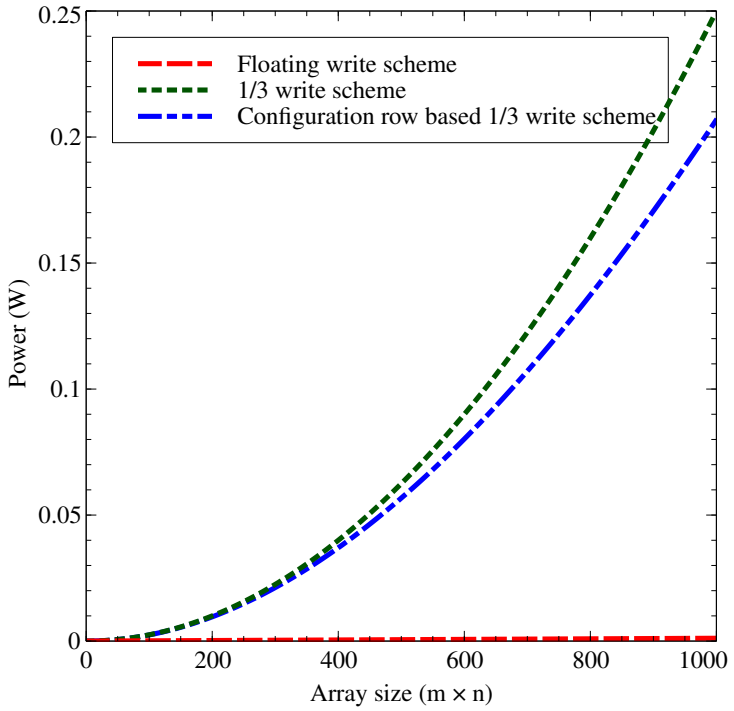


Figure 5.19. Power consumption for write operation in CRS crossbar as function of array size (m × n) for symmetric crossbar. The floating write scheme consumes less power but susceptible to change the states of other unselected CRSs and hence rarely used. 1/3 write scheme is widely used to write state in selected CRS and contents of unselected CRSs are not disturbed. Configuration row based 1/3 write scheme is modified version of 1/3 write scheme with improvement in power consumption at the cost of area overhead of configuration row of CRSs.

voltage is larger, then CRS was in HRS-LRS state (which has changed to LRS-LRS state in first step). If sensed voltage is less, the state of CRS is LRS-HRS. The read scheme is destructive as it destroys the content of it (if it was in HRS-LRS). Hence third step to write back its original content is required. The sensed voltage across R_S is used to write back the original state to CRS used in read operation. The read scheme is shown in Figure 5.20.

In the second step, when the CRS to be read is in LRS-LRS, the voltage across sense resistor R_S is

$$V_{R_S} = \frac{A_1 R_{HRS} + A_2 R_{LRS}}{2A_1 R_{HRS} R_{LRS} + 2A_1 R_{LRS}^2 + A_1 R_{HRS} R_S + A_2 R_{LRS} R_S} R_S V_R, \tag{5.37}$$

where,

$A_1 = n + m - 1,$

$A_2 = (2m-1)n - m + 1.$

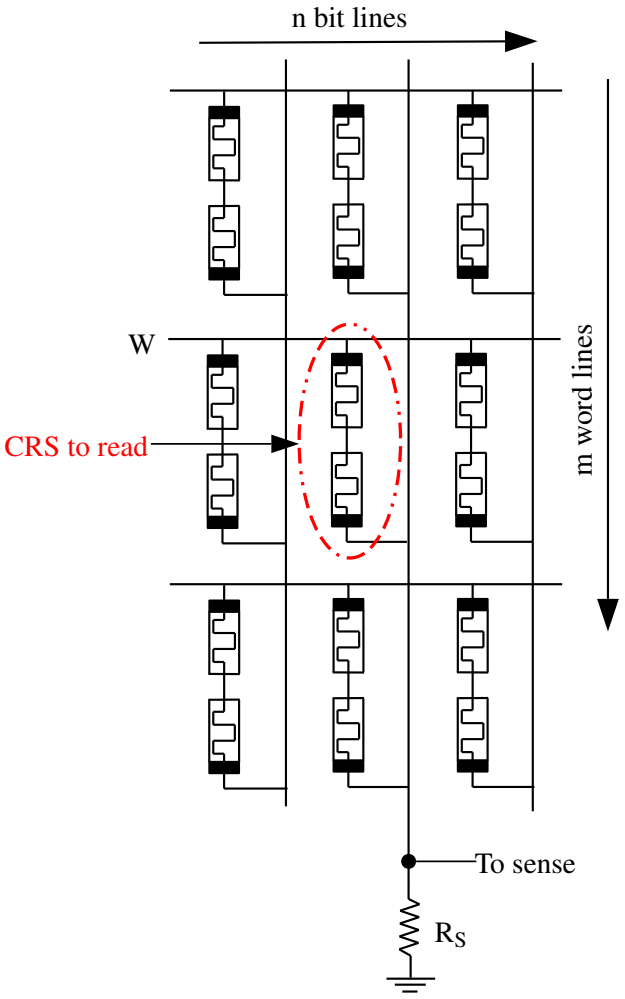


Figure 5.20. Conventional read scheme to read CRS from CRS crossbar array. In first step voltage V_S is applied to word line W of CRS to be read such that effective voltage across CRS (V_{CRS}) to be read is $V_{th1,C} < V_{CRS} < V_{th2,C}$. If CRS to be read is in HRS-LRS then it will switch to LRS-LRS in step 1 otherwise it will remain in LRS-HRS. In second step, read voltage $V_{READ} = V_R$ (with condition $V_R < V_{th2,C}$) is applied to word line W and resultant voltage is sensed across R_S. If sensed voltage is larger, the CRS was in HRS-LRS otherwise it is in LRS-HRS state. Third step is used to write back the original content of CRS using sensed voltage as read operation is destructive in nature.

If CRS to be read is in LRS-HRS state, then the voltage across sense resistor R_S is

$$V_{R_S} = \frac{mn}{(n+m-1)R_{HRS} + (n+m-1)R_{LRS} + mnR_S} R_S V_R. \tag{5.38}$$

In Figure 5.22, (5.37) and (5.38) are plotted as a function of array size $m \times n$ for symmetric array ($m = n$). Even for larger array size, it is possible to distinguish between LRS-LRS state and LRS-HRS state of CRS to be read as all other CRSs surrounding it are either in LRS-HRS or HRS-LRS states with equivalent resistance of each CRS is approximately equal to R_{HRS}.

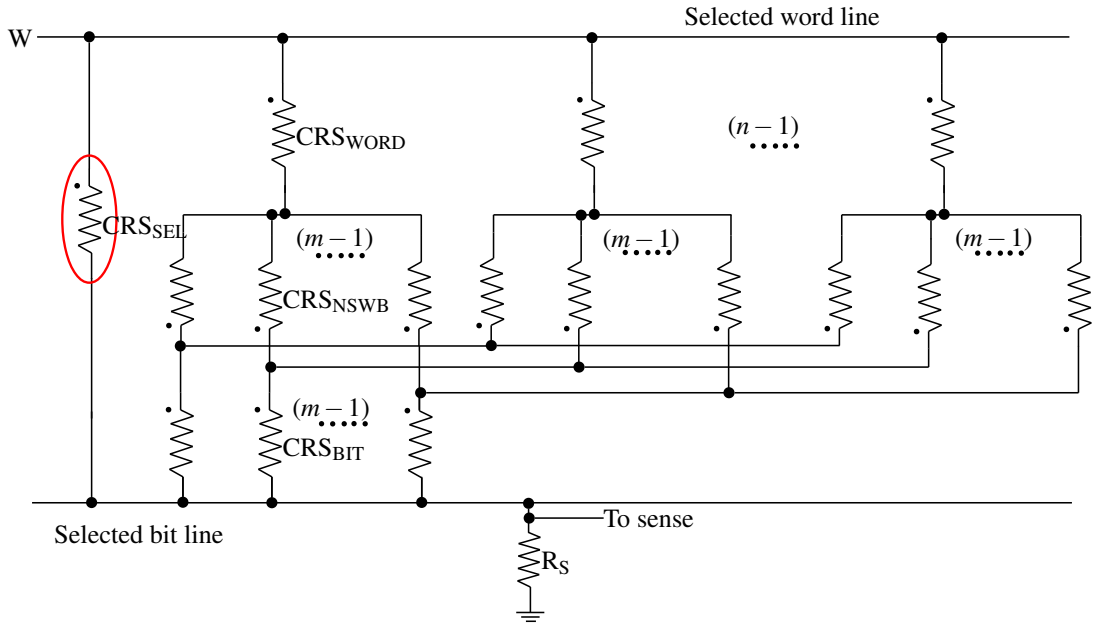


Figure 5.21. Resistive equivalent circuit for conventional read scheme to read CRS from CRS crossbar array. After applying voltage V_R to word line W in step 2, the sensed voltage across R_S should be able to distinguish between LRS-LRS and LRS-HRS state.

5.4.2.2 Self-resetting Read Scheme

The self-resetting read scheme [144] reads the state of CRS to be read like conventional read scheme. The self-resetting circuit makes use of read out voltage to control the write back voltage. The read out voltage will be low if CRS to be read is in LRS-HRS state or will be high if CRS is in LRS-LRS state. Note that HRS-LRS state of CRS is converted to LRS-LRS state in first step of conventional scheme. The read out voltage is delayed to control write back voltage.

The general self-resetting crossbar of CRS is shown in Figure 5.23. To address the CRS, row and column decoders are used. The pulse generator will provide the reading/writing pulse of appropriate magnitude during the read and write operations for selected CRS. The self-resetting circuit is shown in Figure 5.24 which is only used during self-resetting read scheme.

To read the CRS state, positive voltage pulse of duration t_{READ} and of amplitude such that $V_{th1,C} < V_{CRS} < V_{th2,C}$ is applied to word line of CRS while its bit line is connected to ground through sense resistor R_S as shown in Figure 5.24. CRS will switch to LRS-LRS state if it is in HRS-LRS state because of V_{READ}. If it is in LRS-HRS state, it will not change its state. After this possible state switching, if read out voltage (the sensed voltage across R_S within the interval t_{READ}) is high, the state of CRS is read as HRS-LRS (which is switched to LRS-LRS). If read out

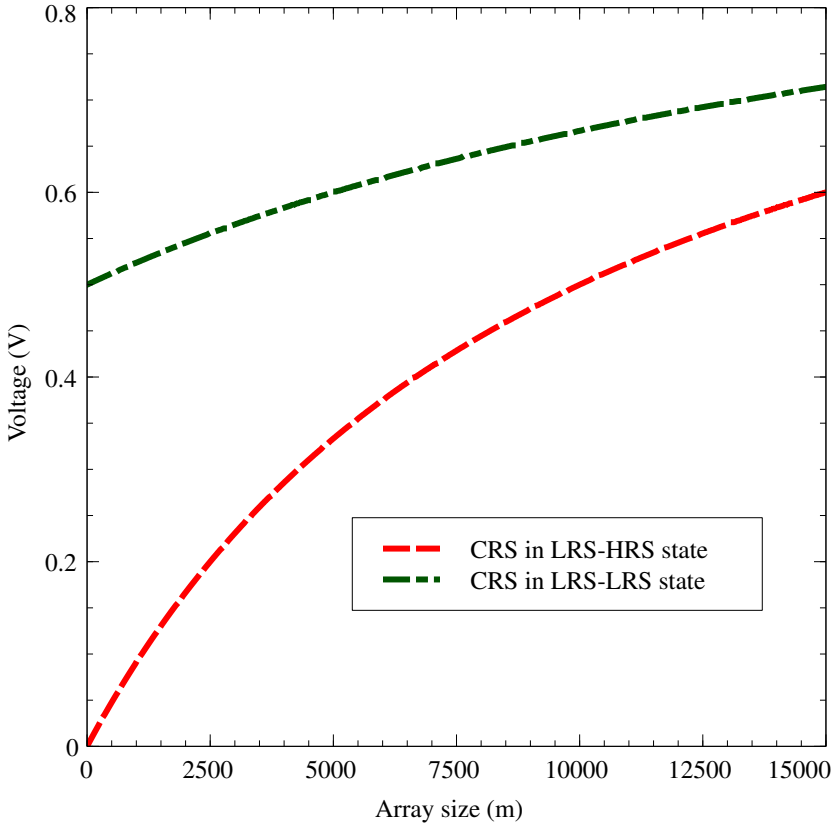


Figure 5.22. Voltage variations across sense resistance R_S when CRS to be read is in LRS-LRS or LRS-HRS (after step 1 of scheme) as function of array size ($m\times$) with $m = n$ for conventional read scheme. The equivalent resistance of every other CRS is approximately equal to R_{HRS} as they are either in LRS-HRS or HRS-LRS state. Hence it is possible to differentiate between LRS-LRS and LRS-HRS state even for larger array sizes.

voltage is small, the state of CRS is read as LRS-HRS. The timing diagram of the self-resetting read scheme is shown in Figure 5.25. If the HRS-LRS to LRS-LRS switching delay of CRS is $t_{d,HL\text{-}LL}$ then the width of read out voltage pulse V_{SENSE} is $(t_{READ} - t_{d,HL\text{-}LL})$. $V_{CONTROL}$ is inverted and delayed V_{SENSE}.

During the read operation the V_{READ} pulse is used to identify the state of the CRS to be read. The same read pulse is delayed by $t_{d,READ}$ (shown as V_{M_3} in Figure 5.25) and is used to enable reset circuit (consisting of M_1, M_2, M_3 and inverter in Figure 5.24) for CRS after identification of its state. Thus

$$t_{d,READ} > t_{READ} \tag{5.39}$$

When the state of CRS to be read is HRS-LRS, the V_{READ} pulse switches it to LRS-LRS state. If the switching time of CRS from HRS-LRS state to LRS-LRS state is $t_{d,HL\text{-}LL}$, then the duration of

V_{SENSE} pulse is

$$t_{SENSE} = t_{READ} - t_{d,HL\text{-}LL} \tag{5.40}$$

The amplitude of this V_{SENSE} pulse is high while reading HRS-LRS state and is low while reading LRS-HRS state. $V_{CONTROL}$ is inverted and delayed (by $t_{d,CONTROL}$) V_{SENSE} pulse.

While reading HRS-LRS state, V_{SENSE} pulse is high, $V_{CONTROL}$ pulse is low and hence transistor M_1 is ON, M_2 is OFF, which will apply voltage V_{WRITE} to lower end of CRS while upper terminal is connected to ground (V_{READ} =0 during this interval). V_{WRITE} is selected such that $V_{CRS} < V_{th4,C}$ and hence HRS-LRS will be written back to read CRS. On the other hand, while reading LRS-HRS state, V_{SENSE} pulse is low, $V_{CONTROL}$ pulse is high and hence transistor M_1 is OFF, M_2 is ON because of which the voltage across CRS after reading CRS in write back state will be zero. Hence read CRS will retain its state.

For successful self-resetting read, with respect to rising edge of V_{READ},

$$t_{d,HL\text{-}LL} + t_{d,CONTROL} \geqslant t_{d,READ}, \tag{5.41}$$

and

$$t_{READ} - t_{d,HL\text{-}LL} \geqslant t_{d,LL\text{-}HL} \tag{5.42}$$

where $t_{d,LL\text{-}HL}$ is time required to write HRS-LRS state to CRS which is in LRS-LRS state.

The whole read operation can be covered in $2t_{READ}+\Delta t$ time where $\Delta t = t_{d,READ} - t_{READ}$. Δt can be made very small but not equal to zero. The minimum time required to perform self-resetting read operation will be $2(t_{d,HL\text{-}LL}+t_{d,LL\text{-}HL})+\Delta t$.

This scheme can be used to reinstate the contents of other CRSs (CRSs not part of operation) if they are altered during write operation as explained previously or during stateful NOR operation. For example, while writing LRS-HRS to selected CRS using floating write scheme, CRS_{WORD} and CRS_{BIT} can change their state to LRS-LRS if they are in the state HRS-LRS. If they are in LRS-HRS state, they will not change their state. To rewrite the original contents to them, if sensed voltage across R_S is relatively larger (CRS and R_S form voltage divider and CRS is in LRS-LRS state), it can be used to activate write circuit in order to write HRS-LRS state.

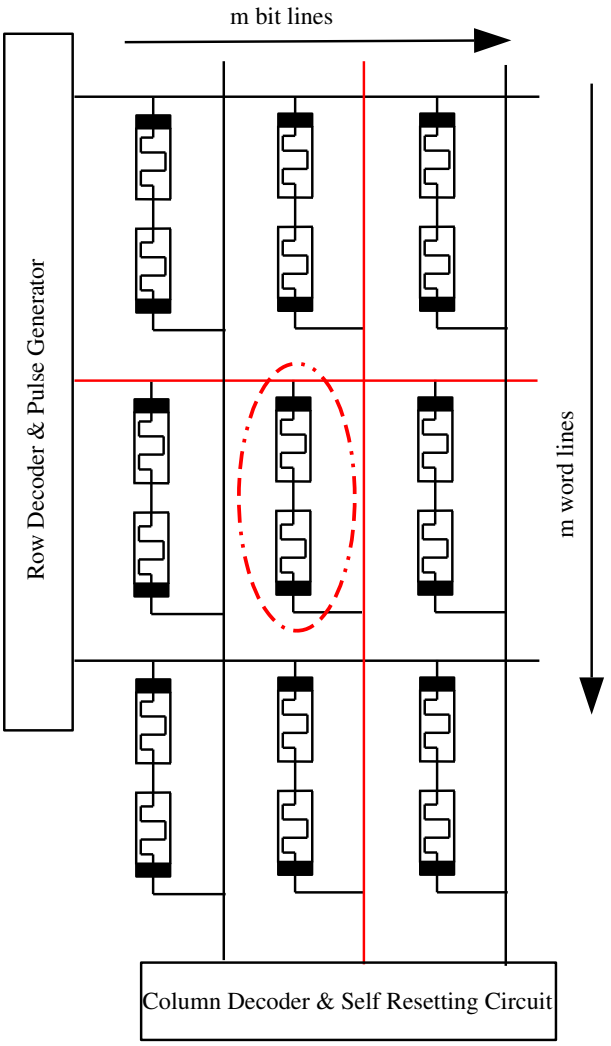


Figure 5.23. Generalized structure of CRS crossbar array with self-resetting read mechanism. Row and column decoder are used to select the CRS for operation. Pulse generator applies the pulse for read operation while self-resetting read circuit writes back the original state of CRS after read operation as read operation is destructive.

5.4.3 Stateful NOR Operation

The implementation of 3-input NOR gate on CRS crossbar is shown in Figure 5.26 and its resistive equivalent circuit is shown in Figure 5.27. If the magnitude of voltage across CRS other than input and destination CRSS is less than $V_{th1,C}$ then it will not change its state. If this magnitude is greater than $V_{th1,C}$ but less than $V_{th2,C}$, the state may change to LRS-LRS but the original state can be restored back by read and feedback write mechanism. The content will be destroyed and can not be restored if magnitude of voltage across CRS is greater than $V_{th2,C}$. In the 3-input NOR operation, the voltages across each unselected CRS due to application of V_x before possible

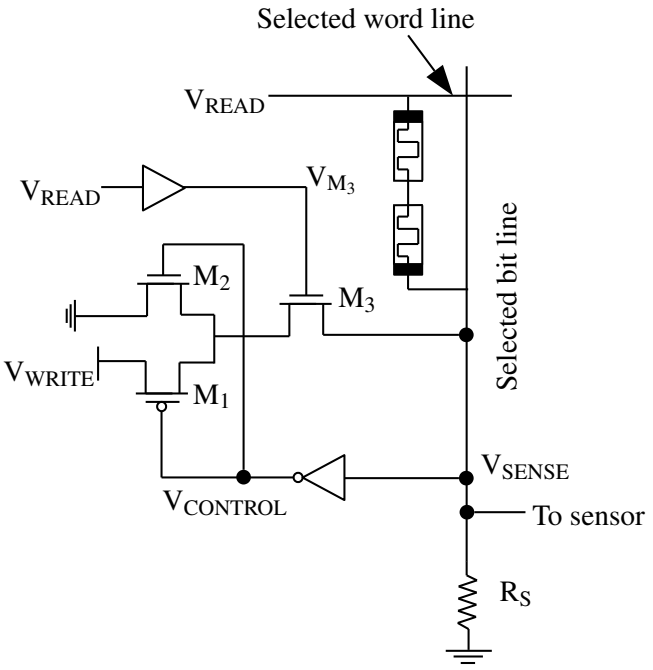


Figure 5.24. Self-resetting circuit for self-resetting read scheme. Reading the state of CRS using conventional three step mechanism destroy the content of CRS (if it is in HRS-LRS state). The self-reset circuit is activated using delayed read pulse. The sensed voltage across R_S is inverted, delayed and used as control signal to write back the initial state of CRS [144].

change of state of destination CRS to LRS-LRS state are given below. We will use notation R_H to represent resistance of CRS in either HRS-LRS or LRS-HRS state where R_H=R_{HRS}+R_{LRS} and R_L for CRS in LRS-LRS state where R_L=2R_{LRS}. The voltage across unselected CRSs in word lines where input CRSs are present (not destination CRS) is

$$V_{\text{WORD,INPUT}} = -\frac{nR_{\text{H}} + mnR_{\text{G}}}{(3n+m-3)R_{\text{H}} + 3mnR_{\text{G}}}V_{\text{x}}. \tag{5.43}$$

The voltage across unselected CRS in word line where destination CRS is present, is given by

$$V_{\text{WORD,DEST}} = \frac{(2n+m-3)R_{\text{H}} + 2mnR_{\text{G}}}{(3n+m-3)R_{\text{H}} + 3mnR_{\text{G}}.}V_{\text{x}} \tag{5.44}$$

Voltage across unselected CRS in selected bit line is given by

$$V_{\text{BIT}} = \frac{(n-1)R_{\text{H}}}{(3n+m-3)R_{\text{H}} + 3mnR_{\text{G}}}V_{\text{x}}. \tag{5.45}$$

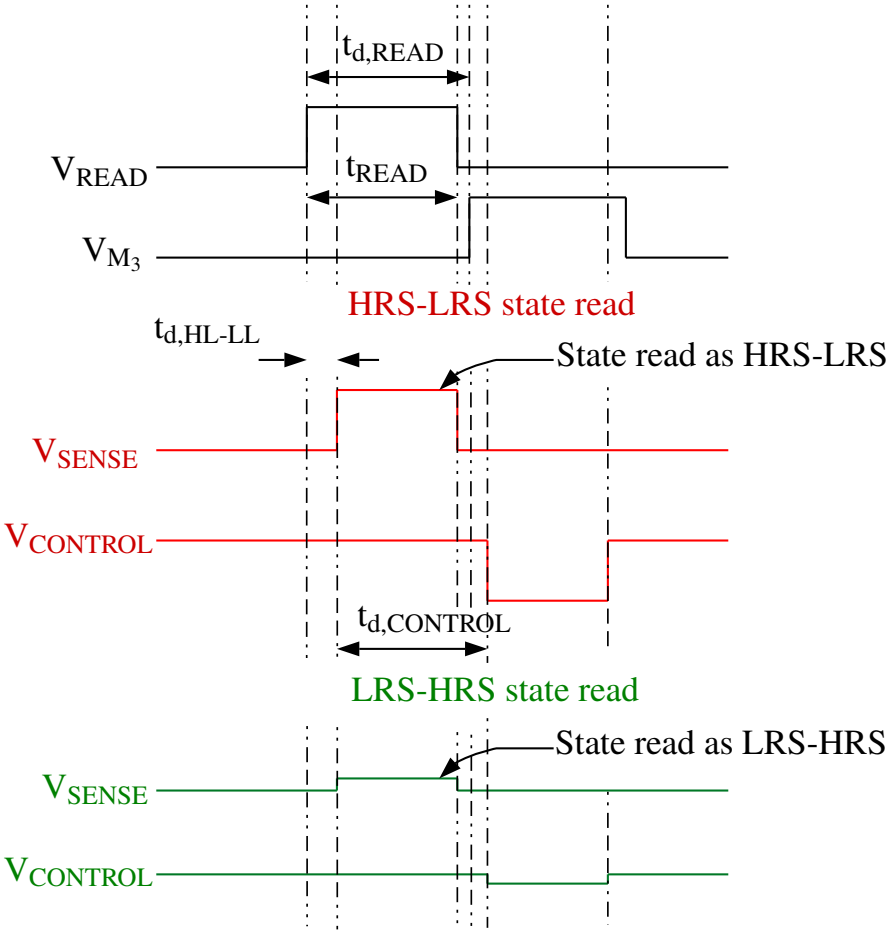


Figure 5.25. Timing diagram for self-resetting read scheme shown in Figure 5.24. The V_{READ} pulse is delayed and used to activate self reset circuit. The sense voltage pulse V_{SENSE} is inverted and delayed ($V_{CONTROL}$) and is used to write back the initial state to CRS.

Voltage across CRS not on selected word lines and bit line is given by

$$V_{NSWB} = -\frac{R_H}{(3n+m-3)R_H + 3mnR_G} V_x. \tag{5.46}$$

In Figure 5.28, (5.43), (5.44), (5.45) and (5.46) are plotted as a function of array size $m \times n$ for symmetric array (i. e. $m = n$). The dot side of resistance notation of CRS is considered as positive terminal in their voltage equations. CRSs on unselected bit lines, unselected word lines (CRS_{NSWB}) and unselected CRSs on selected bit line (CRS_{BIT}) are not affected as magnitude of voltage across them is less than $V_{th1,C}$ or $|V_{th3,C}|$. Unselected CRSs on selected input word lines ($CRS_{WORD,INPUT}$) (not on destination word line) will not change their states if they are in HRS-LRS states. But if they are in LRS-HRS states, they will switch to LRS-LRS states as voltage across them is less than $V_{th3,C}$ but greater than $V_{th4,C}$. Once they are in LRS-LRS states, the sensed

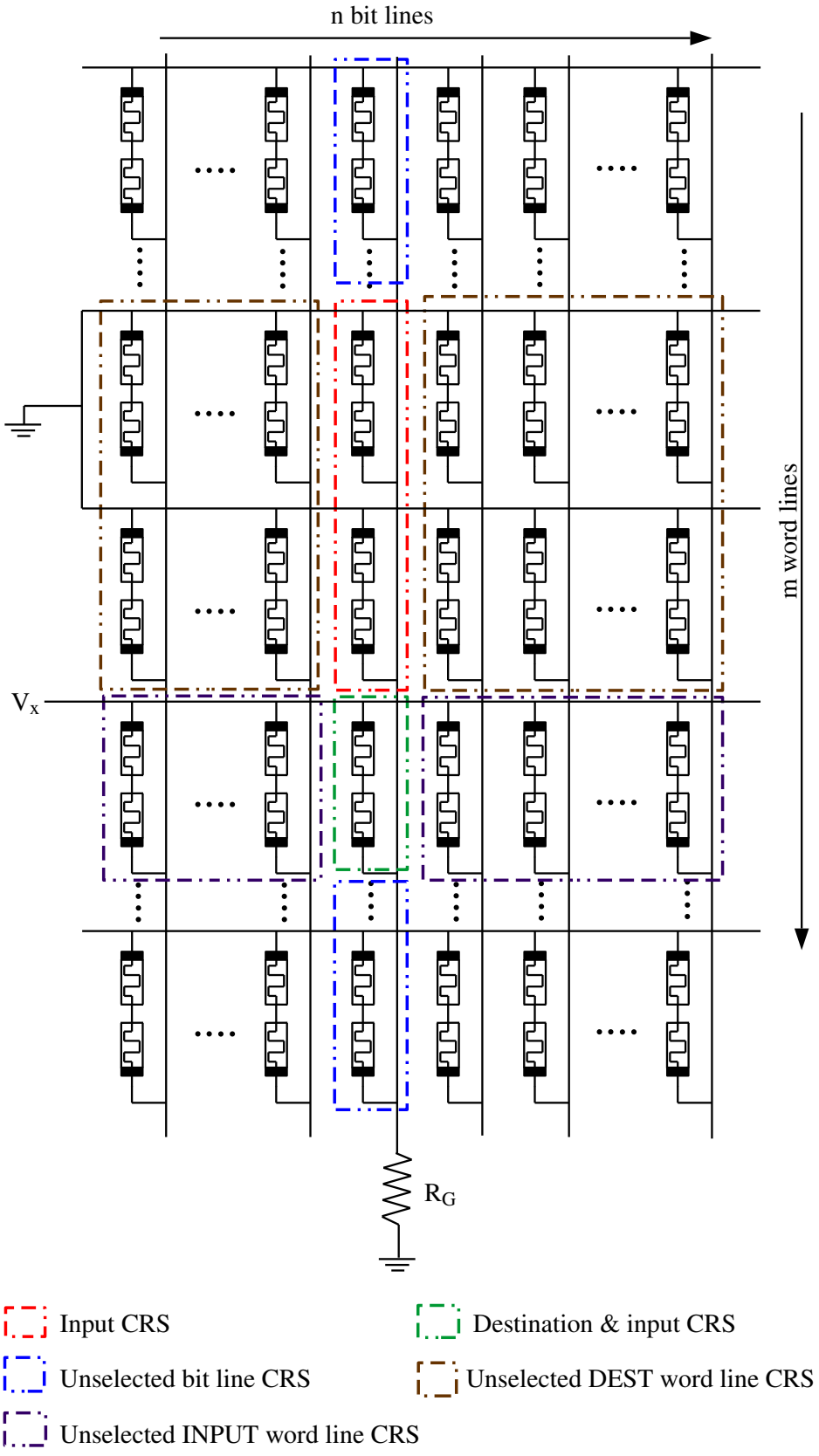


Figure 5.26. Implementation of 3-input NOR logic on CRS crossbar. The word line of destination CRS (which is also one of the inputs) is connected to voltage V_x while word lines of other input CRS are connected to ground. The common bit line of all input CRS is connected to ground through resistance R_G.

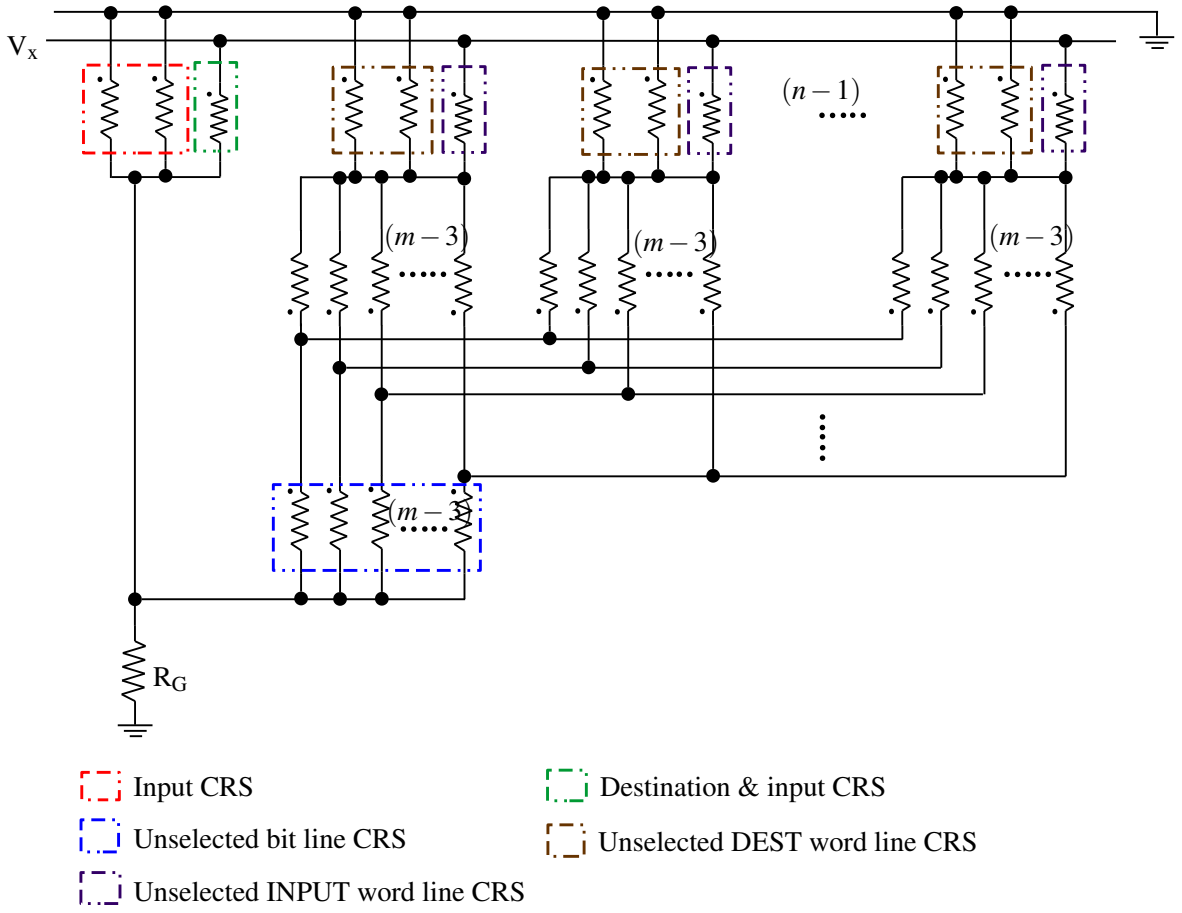


Figure 5.27. Resistive equivalent circuit for implementation of 3-input NOR logic on CRS crossbar shown in Figure 5.26. The dot notation defined in Figure 5.5 are used in equivalent circuit. The magnitude resultant voltages because of V_x across CRS other than input and destination CRS should be less than $V_{th1,C}$ or at least less than $V_{th2,C}$ so that either their contents are not destroyed or can be restored back.

voltage across sense resistor R_S in read procedure can be used to restore (or write) LRS-HRS state in those CRSs. Contents of unselected CRSs in destination word line ($CRS_{WORD,DEST}$)are destroyed if they are in HRS-LRS states as voltage across them is greater than $V_{th2,C}$.

If all the inputs are in HRS-LRS, the destination CRS will switch to LRS-LRS state. The voltage across each unselected CRS after this switching is given below. The voltage across unselected CRS in word lines where input CRS are present (not destination CRS) is

$$V_{\text{WORD,INPUT}} = -\frac{[nR_L + (n+m-3)R_G]R_H + [(m-1)n - m + 3]R_G R_L}{[(3n+m-3)R_L + (3n+m-3)R_G]R_H + [(3m-3)n - m + 3]R_G R_L} V_x \quad (5.47)$$

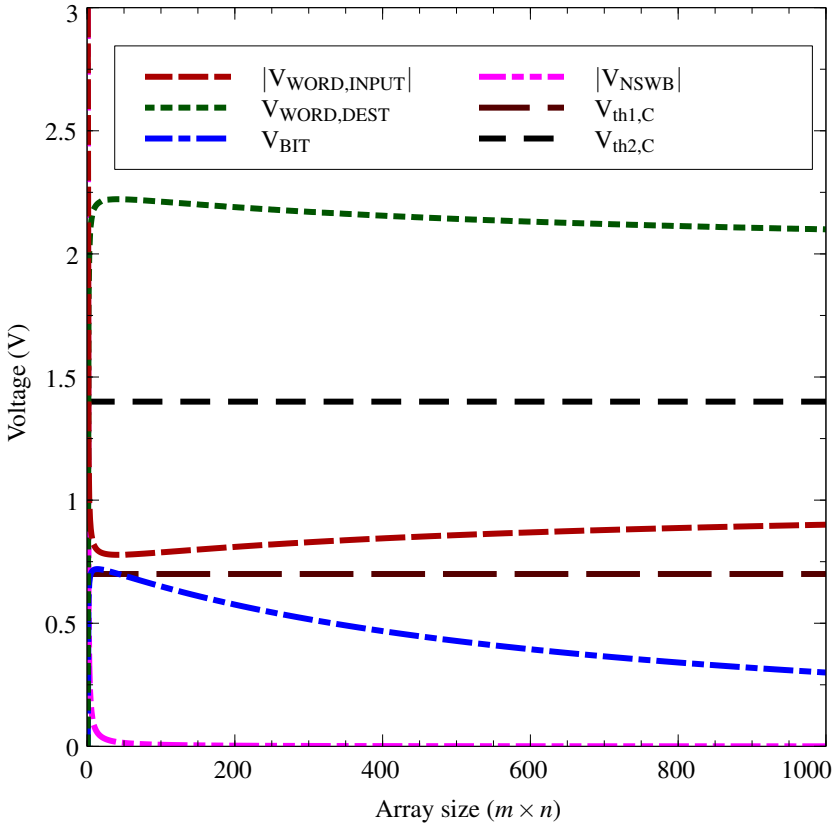


Figure 5.28. Voltage variations across unselected CRSs before destination CRS switching as function of array size ($m \times n$) with $m = n$ for implementation of 3-input NOR logic on CRS crossbar. Content of unselected CRSs on selected bit line (CRS_{BIT}), unselected bit lines and unselected word lines (CRS_{NSWB}) do not change as voltage across them is always less than $V_{th1,C}$ or $|V_{th3,C}|$. Contents of unselected CRSs on selected input word line ($CRS_{WORD,INPUT}$) (not destination word line) will change to LRS-LRS if they are in LRS-HRS as voltage across them is less than $V_{th3,C}$ but greater than $V_{th4,C}$ but they can be restored back to LRS-HRS by using feedback write mechanism where read voltage is used as feedback signal. The content of unselected CRS in destination word line ($CRS_{WORD,DEST}$) will be destroyed if they are in HRS-LRS as voltage across them is greater than $V_{th2,C}$.

The voltage across unselected CRS in word line where destination CRS is present, is given by

$$V_{WORD,DEST} = \frac{[(2n+m-3)R_L + 2nR_G]R_H + (2m-2)nR_GR_L}{[(3n+m-3)R_L + (3n+m-3)R_G]R_H + [(3m-3)n-m+3]R_GR_L}V_x \quad (5.48)$$

Voltage across unselected CRS in selected bit line is given by

$$V_{BIT} = \frac{[(n-1)R_L + (2-2n)R_G]R_H + (2n-2)R_GR_L}{[(3n+m-3)R_L + (3n+m-3)R_G]R_H + [(3m-3)n-m+3]R_GR_L}V_x \quad (5.49)$$

Voltage across CRS not on selected word lines and bit line is given by

$$V_{\mathrm{NSWB}} = -\frac{(R_{\mathrm{L}} - 2R_{\mathrm{G}})R_{\mathrm{H}} + 2R_{\mathrm{G}}R_{\mathrm{L}}}{[(3n+m-3)R_{\mathrm{L}} + (3n+m-3)R_{\mathrm{G}}]R_{\mathrm{H}} + [(3m-3)n-m+3]R_{\mathrm{G}}R_{\mathrm{L}}} V_{\mathrm{x}} \tag{5.50}$$

In order to check the effect of destination CRS switching to LRS-LRS in case where all inputs are HRS-LRS, the magnitude of effective voltages across each unselected CRS given by (5.47), (5.48), (5.49) and (5.50) are plotted as a function of array size $m \times n$ for symmetric array (i. e. $m = n$) in Figure 5.29. In this case also the contents of unselected CRSs in destination word line are destroyed. Other CRSs either don't change their states or they can be restored back.

The power consumption in evaluate (NOR) operation on CRS crossbar as function of array size (m $\times$ n) is shown in Figure 5.30 for symmetric array. The power consumption just before and after execution is shown in this figure.

5.5 Summary

Implementation of stateful NOR gate on CRS crossbar is described and analysed in this chapter. The observations are summerized below.

- In order to implement stateful NOR gate on CRS crossbar, 1/3 write scheme is better choice for writting inputs to CRSs as it does not alter the contents of other CRSs.
- While performing stateful NOR operation,
  - The contents of CRSs that are neither on word lines nor on bit lines of CRSs taking part in logic implementation are unaffected as voltage across them is almost negligible.
  - The contents of CRSs on bit and word lines of input CRSs (excluding destination CRS) taking part in logic can alter to LRS-LRS if they are in HRS-LRS and array size is less than 7000$\times$7000 but they can be restored back by using mechanism used in self-resetting read scheme. The contents of these CRSs will not be destroyed at all if they are in LRS-HRS state or array size is more than 7000$\times$7000.
  - The contents of CRSs on word line of destination CRS (excluding it) are destroyed if they are in HRS-LRS state and they can not be recovered back. In order to avoid

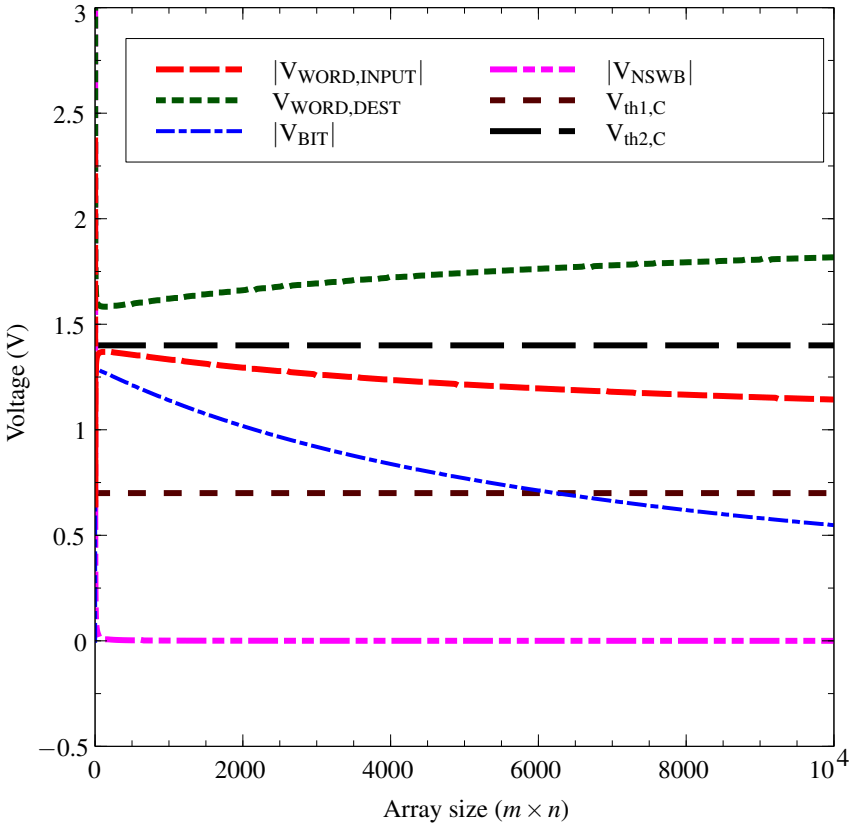


Figure 5.29. Voltage variations across unselected CRSs after possible destination CRS switching to LRS-LRS as function of array size ($m \times n$) with $m = n$ for implementation of 3-input NOR logic on CRS crossbar. Contents of CRSs on unselected bit lines and unselected word lines (CRS_{NSWB}) do not change as voltage across them is greater than $V_{th3,C}$. Also contents of unselected CRSs on selected bit line (CRS_{BIT}) will not change if array size is approximately more than 7000×7000. Contents of unselected CRSs on selected input word lines ($CRS_{WORD,INPUT}$) (not destination word line) for any array size or on selected bit line (CRS_{BIT}) for array size less than approximately 7000×7000 will switch to LRS-LRS if they are HRS-LRS as magnitude of voltage across them is greater than $V_{th1,C}$ and less than $V_{th2,C}$, but they can be restored back to HRS-LRS by using feedback write mechanism where read voltage is used as feedback signal. The content of unselected CRS in destination word line ($CRS_{WORD,DEST}$) will be destroyed if they are in HRS-LRS as voltage across them is greater than $V_{th2,C}$.

this, all such HRS-LRS states in these CRSs can be converted to LRS-LRS state by applying voltage greater than $V_{th1,C}$ and less than $V_{th2,C}$ to its word line and grounding bit line, before performing stateful NOR operation. Once stateful NOR operation is over, their contents can be restored back by using self-resetting read scheme. Other option is to keep some word lines as dedicated destination word lines so that even if their contents are changed, it will have any impact on system.

- The state of CRS can be read correctly even for larger array size as CRS crossbar array is free from sneak path problem. Self-resetting read scheme is better choice for read operation.

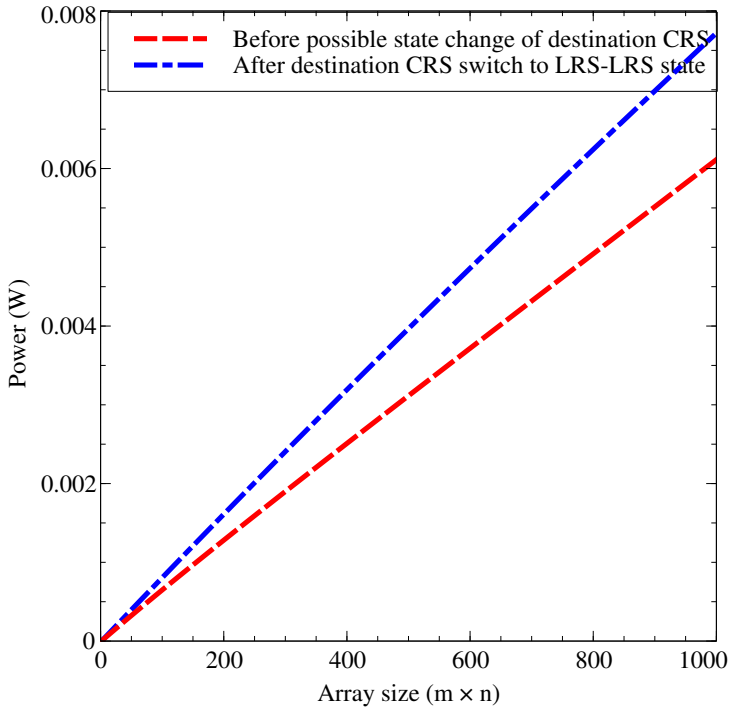


Figure 5.30. Power consumption for evaluate operation in CRS crossbar as function of array size (m $\times$ n) for symmetric crossbar. The power consumption after possible LRS-LRS switching of destination CRS while performing stateful-NOR on CRS crossbar is slightly higher than just before destination CRS switching.

In this scheme, read voltage V_R is applied to word line of CRS to be read such that $V_{th1,C} < V_R < V_{th2,C}$ and its bit line is connected to ground through resistance R_S forming voltage divider with CRS to be read. If CRS to be read is in HRS-LRS state, it will be switched to LRS-LRS state because of V_R and hence sensed voltage across R_S is larger. If CRS to be read is in LRS-HRS state, its state remains unchanged and sensed voltage across R_S is relatively smaller. The delayed sensed voltage is used as control signal to write back the original contents to CRS that were present before performing read operation.

In next chapter, reconfigurable pipelined architecture using stateful NOR gate as basic building block is proposed that can be implemented on memristive/CRS crossbar. The automation algorithm for proposed architecture and supporting CMOS circuits are also given.

Chapter 6

Reconfigurable Architecture

6.1 Introduction

The implementation of stateful NOR logic operation with memristors (on passive memristive crossbar or on specialized crossbars like CMOL and FPNI) is explained in Chapter 4 and on CRS CRS crossbar array is explained in Chapter 5. NOR is universal operation and hence any other logic function can be implemented with the help of NOR logic gate. Also the NOR operation with memristors/CRSs is stateful i.e. the result of NOR is stored in the form of resistance state (HRS or LRS in memristor, HRS-LRS or LRS-HRS in CRS) in destination memristor or CRS. Hence combinational as well as sequential operations can be carried out with stateful NOR gate.

In this chapter, generalized reconfigurable logic architecture is proposed using stateful NOR as basic building block. Common CMOS circuit blocks required in the architecture are explained. Automation algorithm for implementation of 3-input logic function, where truth table of logic function to be implemented as input and netlist as output for function implementation, is described. If the logic function has more than 3 inputs, then multiplexer based implementation of such logic function using proposed 3-input architecture and its automation algorithm are also explained.

6.2 Common Circuit Blocks in CMOS Layer

To implement the architecture on memristive crossbar or on CMOL/FPNI structure or on CRS crossbar, various common circuit blocks are required in CMOS layer as memristor and therefore CRS is passive element. Such circuits are explained in this section.

6.2.1 Write Circuit

The circuit shown in Figure 6.1 can be used to write LRS/ HRS state in memristor or HRS-LRS/LRS-HRS state in CRS using 1/3 write scheme. LRS/LRS-HRS is considered as logic '1' while HRS/HRS-LRS as logic '0' in memristive/CRS logic. WRC is write control signal to enable write operation. Terminal ST_1 is connected to word line (bit line) of memristor/CRS to be written while ST_2 is connected to bit line (word line) of it. The terminal UT_1 is connected to unselected word lines (bit lines) and UT_2 is connected to unselected bit lines (word lines). When data to be written is logic '1' (logic '0'), voltage V_W will be at ST_1, ST_2 is connected to ground, voltage at UT_1 will be $V_W/3$ and voltage at UT_2 will be $2V_W/3$. When data to be written is logic '0' (logic '1'), the voltages at these terminals will be negative with same magnitude. The value of voltage V_W is selected such that $V_W > V_{th1,M}$ for memristor and $V_W > V_{th2,C}$ for CRS, $V_W/3 < V_{th1,M}$ for memristor and $V_W/3 < V_{th1,C}$ for CRS, $-V_W < V_{th2,M}$ for memristor and $-V_W < V_{th4,C}$ for CRS, $-V_W/3 > V_{th2,M}$ for memristor and $-V_W/3 > V_{th3,C}$.

6.2.2 Evaluate Circuit

To implement stateful NOR on memristive crossbar or CRS crossbar, evalute logic circuit is shown in Figure 6.2, where EV is evaluate control signal to enable operation. For memristor based stateful NOR, voltage V_{SET} is applied to terminal P while voltage V_{COND} is applied to terminal Q. For CRS based stateful NOR, voltage V_x is applied to terminal P while terminal Q is connected to ground. Terminal T_D is connected to word line of destination memristor or CRS, terminal T_I is connected to word lines of input memristors or CRSs. Terminal T_C connects common bit line of destination and input memristors or CRSs to ground through resistance R_G.

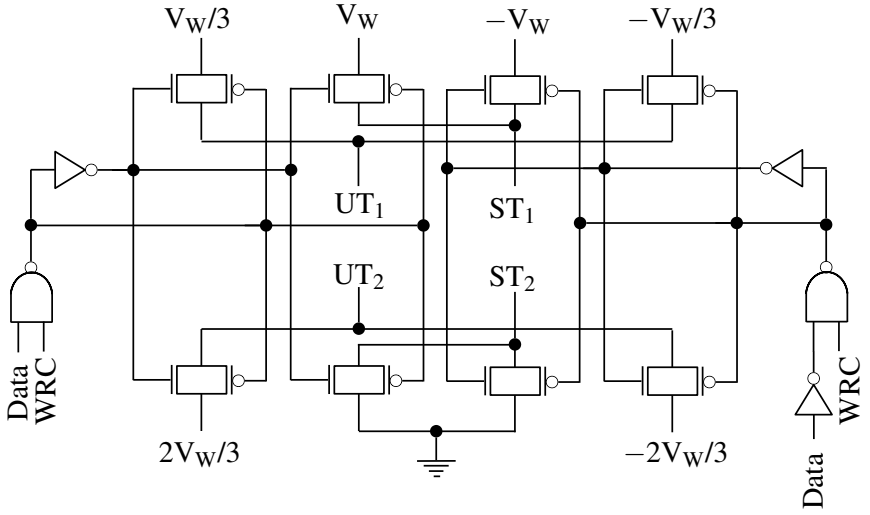


Figure 6.1. Write circuit to write state in selected memristor or CRS using 1/3 write scheme. WRC is write control signal to enable write operation. Based on logic value of data (logic '1' or logic '0'), appropriate voltage (V_W or $-V_W$) is applied to one terminal of memristor or CRS selected while its other terminal is connected to ground. Unselected word lines are driven with voltage $V_W/3$ or $-V_W/3$ while unselected bit lines are driven with voltage $2V_W/3$ or $-2V_W/3$. Proper value of V_W is chosen for memristive and CRS crossbar write operation.

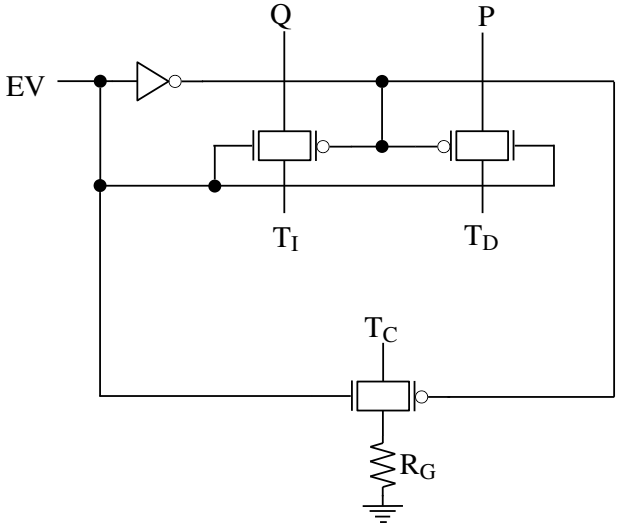


Figure 6.2. Evaluate logic circuit to implement stateful NOR in memristive crossbar or in CRS crossbar. EV is evaluate control signal to enable operation. Terminal P is connected to voltage V_{SET} and terminal Q is connected to voltage V_{COND} for memristor based stateful NOR. Terminal P is connected to voltage V_x while terminal Q is grounded for CRS based stateful NOR.

6.2.3 Read Circuit

The read circuit is used to detect state of destination memristor or CRS after evaluation and is shown in Figure 6.3. The voltage V_R is such that it does not alter the state of memristor during read operation in memristive crossbar while in CRS crossbar, voltage V_R will be such that if CRS

to be read is in HRS-LRS state, it will switch to LRS-LRS state. If CRS is in LRS-HRS state, it will not change its state. Resistance R_S form voltage divider with memristor or CRS, and voltage drop across it will be greater than threshold voltage of nMOS when destination memristor is in LRS or CRS is in LRS-LRS state. This will produce V_{RESULT} = logic '1'. When destination memristor is in HRS or CRS is in LRS-HRS, the nMOS will be off and in turn will produce V_{RESULT} = logic '0'. The read control signal RDC=logic '1' will enable this read circuit. For CRS crossbar additional write back step is required to retain back the contents of CRS. Sense amplifier can also be used to read the state by sensing the voltage across R_S. The read circuit using sense amplifier and precharge circuit is shown in Figure 6.4 [147]. It consists of precharge circuit, where EQ signal is used to force equal charge across capacitor C_1 and C_2 in the initialization phase. The sampling signals ϕ_1 and ϕ_2 are used to convert charge to voltage across C_1 and C_2. The difference between charge on C_1 and C_2 will vary based on the voltage across R_S which in turn depends on state of memristor/CRS to be read. The voltage V_1 and V_2 is applied to sense amplifier and sensing is enabled by applying NS and PS signals sequentially.

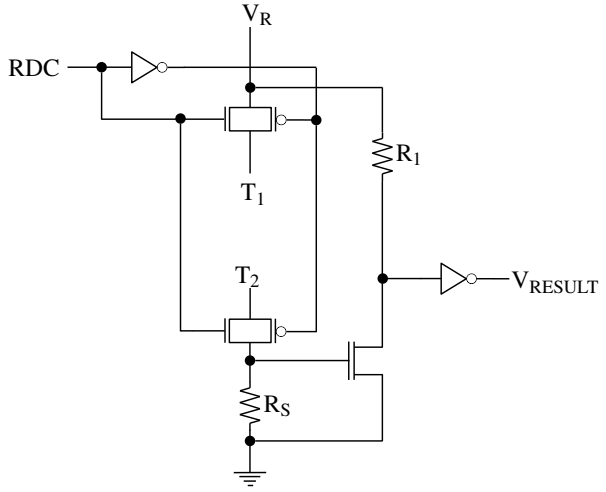


Figure 6.3. Read Circuit to detect state of memristor or CRS after evaluate phase. When read control signal RD='1', voltage V_R is applied across voltage divider circuit consisting of destination memristor or CRS and resistance R_S. When memristor is at logic '1' (LRS) or CRS is in LRS-LRS state, voltage drop across R_S is above threshold of nMOS which turns on to give V_{RESULT} =logic '1'. Otherwise it is logic '0'.

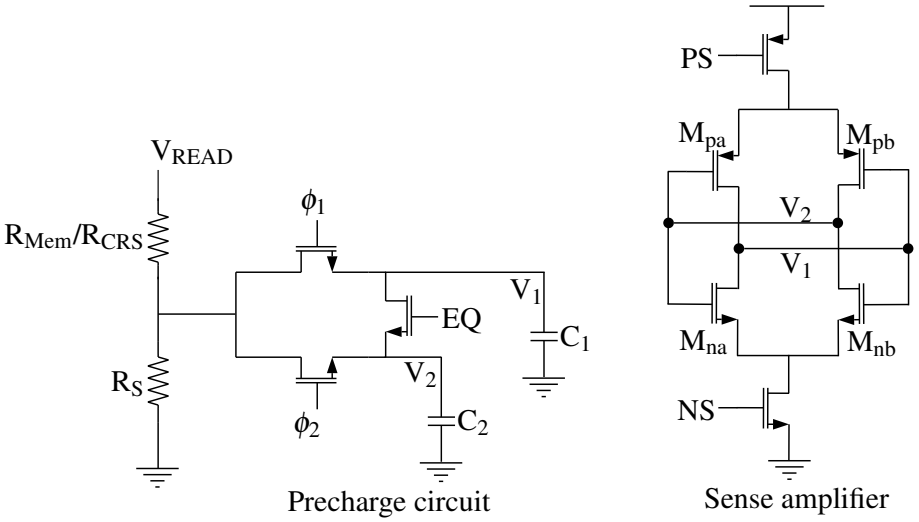


Figure 6.4. Sense amplifier used in reading the state of memristor/CRS. It consists of precharge circuit, where EQ signal is used to force equal charge across capacitor C_1 and C_2 in the initialization phase. The sampling signals ϕ_1 and ϕ_2 are used to convert charge to voltage across C_1 and C_2. The difference between charge on C_1 and C_2 will vary based on the voltage across R_S which in turn depends on state of memristor/CRS to be read. The voltage V_1 and V_2 is applied to sense amplifier and sensing is enabled by applying NS and PS signals sequentially [147].

6.2.4 Priority Logic

The priority logic circuit is shown in Figure 6.5 having $I_1 - I_8$ inputs and $O_1 - O_8$ outputs with I_1 having highest priority and I_8 having lowest priority. If, out of all inputs to priority logic, some or all are at logic '1', then out of it, only highest priority input is kept at logic '1' at output and other outputs are forced to logic '0' using priority logic circuit. The exact use of this with respect to architecture will be explained in description of architecture.

6.3 Reconfigurable Architecture using Stateful NOR

Stateful NOR gate implemented in memristive/CRS crossbar and the CMOS circuit blocks explained in previous section are used in the proposed architecture. The basic NOR block is given in Figure 6.6 which will be used to describe the architecture. It consists of '*N*' input stateful NOR gate implemented on crossbar, '*N*' bits of data, control signals (WRC, EV and RDC) and one timing control signal LH used in write operation as it is not possible to write logic '0' and '1' simultaneously in memristors or CRSs. When LH=logic '0', logic '0' is written to memristors or CRS in addition to destination memristor (not required in CRS) in step 1 of write mode and then to

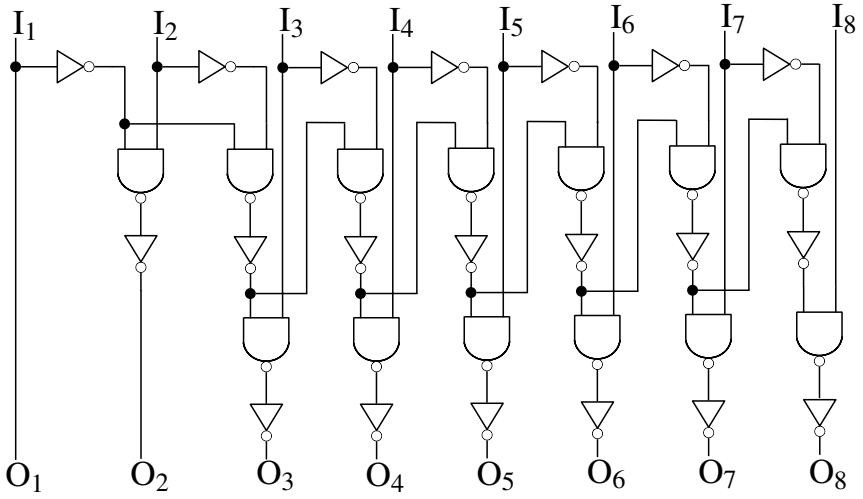


Figure 6.5. Priority Circuit with $I_1 - I_8$ as inputs and $O_1 - O_8$ as outputs (I_1-highest priority, I_8-lowest priority). When some or all inputs are at logic '1' then the one with highest priority is passed as it is to output while others are forced to logic '0' at output.

write logic '1' to memristors or CRSs in step 2 of write mode, LH=logic '1'. The voltages $\pm V_W$, V_{SET}, V_{COND} for memristive NOR or V_x for CRS based NOR and V_R are not shown explicitly in this symbol and are applied through CMOS layer as described below.

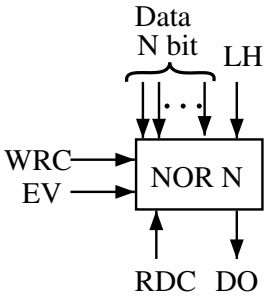


Figure 6.6. NOR logic block symbol with '*N*' data inputs. WRC, EV and RDC are write, evaluate and read control signals, respectively to enable these modes. LH is to write logic '0' in step 1 with LH='0' and logic '1' in step 2 with LH='1' of write mode. DO is the output of read operation which is result of NOR logic.

In write mode, WRC='1'. In step 1 of write mode LH='0' is combined with data bits to access those memristors or CRSs for which data is '0' along with destination memristor for memristive NOR. Using write circuit given in Figure 6.1, voltage V_W is applied to one terminal of all such memristors or CRS and common terminals of all are connected to ground so that effective voltage across them is V_W and logic '0' is written to them. In step 2 of write mode LH='1' is combined with data bits to access those memristors or CRSs for which data is '1'. Voltage $-V_W$ is applied to one terminal of all such memristors or CRSs and common terminal of all are connected to ground so that effective voltage across them is $-V_W$ and logic '1' is written to them.

In evaluation mode, EV='1'. One terminal of all input memristors is connected to voltage V_{COND}, one terminal of destination memristor is connected to voltage V_{SET} and common terminal of all memristors of NOR gate is connected to ground through resistance R_G for memristive NOR. For CRS based NOR, terminal of destination memristor is connected to voltage V_x, one terminal of all input CRS is connected to ground while common terminal of all CRS is connected to ground through resistance R_G. Evaluation circuit shown in Figure 6.2 is used for this operation.. The result of stateful NOR logic is stored in destination memristor or CRS.

In read mode, RDC='1' and only destination memristor or CRS is accessed. Voltage V_R is applied to one terminal of memristor or CRS while other terminal of it is connected to ground through R_S to form voltage divider. If destination memristor is in HRS state or CRS is in LRS-HRS, voltage drop across R_S is small. If destination memristor is in LRS or CRS is in LRS-LRS state, voltage drop across R_S is larger. Thus drop across R_S can be used to identify state of destination memristor or CRS (i.e. the result of NOR operation).

6.3.1 Architecture Description

The proposed architecture can be used to implement any 3-input logic function. The block diagram representation of architecture is shown in figure 6.7. The first level consists of one 3-input NOR block, three 2-input NOR blocks and three 1-input NOR blocks (essentially an inverter). If A, B and C are the inputs to logic function, then inputs (A, B, C) are applied to 3-input NOR block, inputs (A, B), (B, C) and (A, C) are applied to each of the three 2-input NOR blocks while inputs (A), (B), (C) are applied to each of three 1-input NOR blocks. In addition to this the output(s) corresponding to all inputs at logic '1' is/are written to destination memristor(s) or CRS(s).

Following the write and evaluate operation explained previously on all NOR blocks simultaneously in level 1, the destination memristors or CRS in those logic block will toggle to logic state '1' (i. e. LRS state for memristor or LRS-LRS state for CRS) for which all inputs are logic '0'. The remaining NOR blocks' destination memristors or CRSs will remain in logic state '0'. All destination memristors or CRSs are read simultaneously in level 1 and read voltage is applied to priority logic.

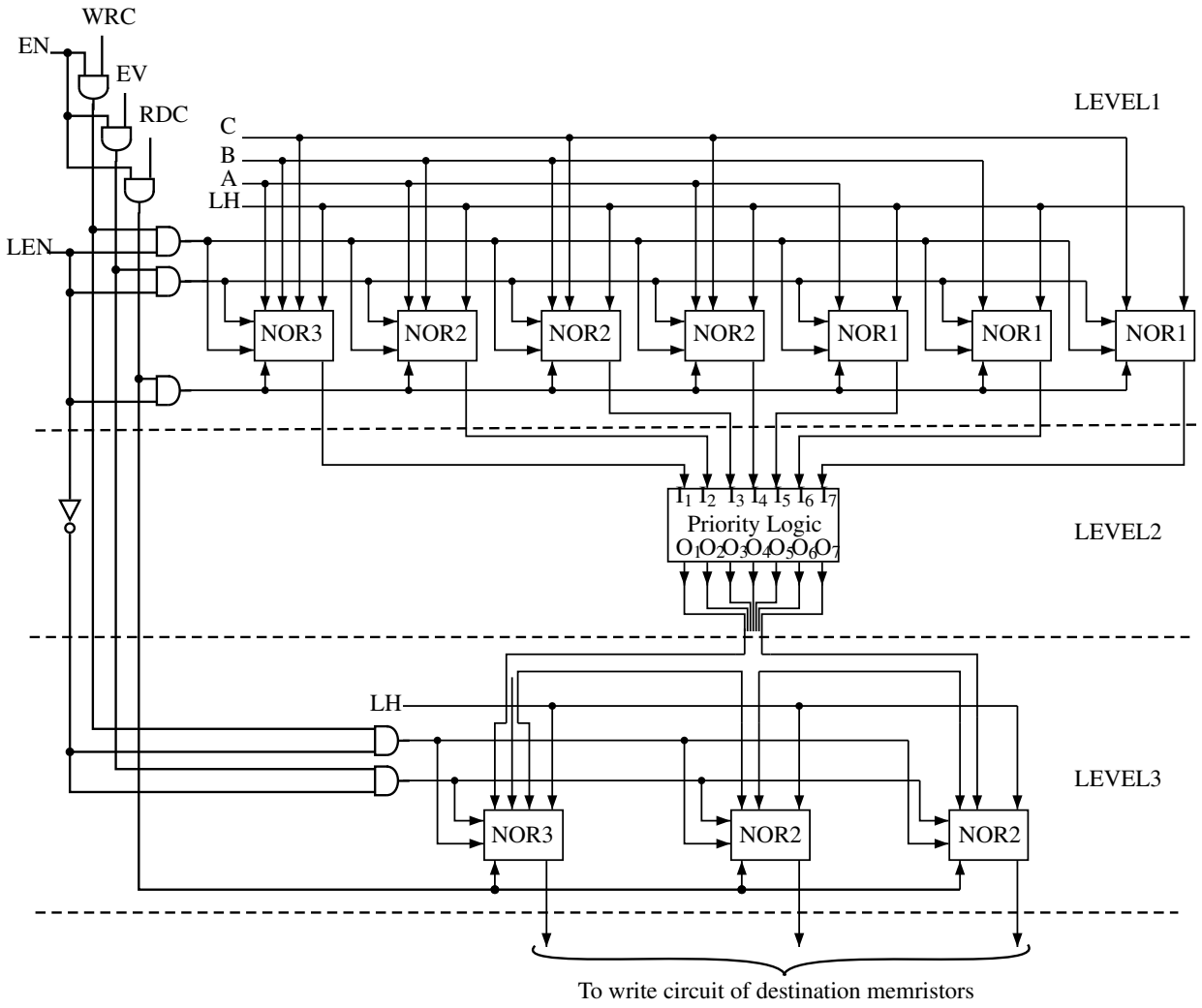


Figure 6.7. Recofigurable, pipelined 3-input logic block (LB3) architecture using stateful NOR operation.

The priority logic is used to identify the minterm. As an example, for a function with three inputs A, B, C, suppose inputs are ABC=‘000’. All destination memristors or CRSs will toggle in level 1 of architecture. The priority logic will keep the read voltage of destination memristor or CRS of only first three input NOR block (NOR block with inputs A, B, C) to level ‘1’ and will force the read voltage of other destination memristor or CRS to logic ‘0’. If inputs are ABC=‘010’, then the destination memristor or CRS of NOR block with inputs A and C, of NOR block with input A and of NOR block with input C will toggle. The priority logic will keep the read voltage of destination memristor or CRS of NOR block with inputs A and C to logic ‘1’ force the read voltage of destination memristors or CRSs of NOR block with A input and of NOR block with C input to logic ‘0’.

Those input combinations creating logic ‘0’ output are grouped from the outputs of priority logic

Table 6.1. The stepwise implementation of 3-input logic function using proposed architecture. The 'EN' should be logic '1' to enable the 3-input logic block. The write circuits for destination memristors or CRSs should be enabled only during step 8. 'x' represent don't care condition.

Step No.	LEN	WRC	LH	EV	RDC	Input
1	1	1	0	0	0	C, B, A
2	1	1	1	0	0	C, B, A
3	1	0	x	1	0	x
4	1	0	x	0	1	x
5	0	1	0	0	1	Priority logic outputs
6	0	1	1	0	1	Priority logic outputs
7	0	0	x	1	0	x
8	0	0	x	0	1	x

and written to NOR block in level 2 and remaining outputs are written to other NOR block and then sequence of evaluate and read operation is repeated. The output corresponding to NOR bock whose destination memristor or CRS does not toggle in second level (in evaluation operation) is written to destination memristor or CRS. The steps involved in 3-input logic function implementation using proposed architecture are summarized in Table 6.1. Note that EN is enable control signal to activate LB3 block (Figure 6.14). There are two different levels of implementation, LEVEL1 and LEVEL2. LEN signal is the enable (disable) signal for LEVEL1 (LEVEL2).

One key consideration here is that the minterm where all the inputs are logic high can not be identified using the above scheme. Due to this reason, default value(s) (depending upon the number of inputs) are written to the destination memristor(s) or CRS (Table **??**). The default value(s) are the output(s) of the logic function to be implemented, when all the inputs are high.

The architecture can be modified in order to avoid the writing the result(s) corresponding to all inputs high to destination memristors or CRSs, by permanently connecting the 8^{th} input of priority circuit to logic '1' and grouping the outputs in usual manner. This will remove the initialization step required for destination memristors or CRSs.

Irrespective of 3-input logic function, LEVEL1 contains the fixed logic and does not change with the function to be implemented. The LEVEL2 is programmable, which is dependent on the function to be implemented. The port mapping algorithm to implement 3-input logic function is explained in the Subsection 6.3.2.

6.3.2 Automation Algorithm for 3-input Logic Block Architecture

The automation algorithm for implementation of any y input and z output logic function is given in Algorithm 1. This algorithm gives the connection pattern between various blocks (port mapping) to implement the given logic function according to the proposed architecture.

In this algorithm, notation $ABC1x.I/Onm$ is used to denote NOR logic blocks in LEVEL1, where ABC is name of block or gate, x denotes the number of inputs to the block, I/O represents either input (I) or output (O) pin, n represents pin number and m represents the serial number of the NOR block. In LEVEL3, notation $ABC3x.I/On$ is used to denote NOR logic blocks, where ABC is name of block or gate, x denotes the number of inputs to the block, I/O represents either input (I) or output (O) pin and n represents pin number. Moreover, in LEVEL3, notation $ABC3.I/On$ is used to denote NOT blocks, where ABC is name of block or gate, 3 is the level number, I/O represents either input (I) or output (O) pin and n represents pin number.

Inputs: Array $I(n,m)$: where, n:1 to 2^y, y is the number of inputs, m:1 to z, z is the number of outputs and $I(n,m)$ are the minterms; NOR blocks; priority block.

Outputs: Number of NOR blocks with number of inputs at each level, 'netlist'- like connection pattern.

The overview of this algorithm is given below:

1. Write the default values into the destination memristors or CRSs.
2. Connect all the inputs, in the combinations of 'a', to 'a'- input NOR blocks. The 'a'- input implication-based NOR blocks produce minterms with 'a' number of complemented inputs.
3. All the a input NOR blocks are grouped together to produce ${}^nC_\mathrm{a}$ outputs and these outputs are connected to consecutive inputs of the priority block *in opposition to fetching minterms consecutively*. This prioritizes the a-input stateful NOR blocks over $a-1$, $a-2$, ...- input implication-based NOR blocks.
4. Count the number of minterms that produce a different output than the output produced by the minterm with all standard inputs (i.e. all inputs are logic high).

5. Call NOR blocks with the above calculated count of inputs, each NOR block associated with an output of the function to be implemented, and connect the inputs to the outputs of the priority block corresponding to the minterms that produce a different output than the output produced by the minterm with all standard inputs.

6. The outputs of all the NOR blocks are inverted.

7. The outputs of the inverters go to the write signals of the write blocks (to enable/disable the write block) of the corresponding destination memristors or CRS, where the outputs of the function are stored. Here, the data signal to the write blocks is the negation of the default value (the destination memristor or CRS will toggle only if the output(s) of a particular input combination is(are) different than the default value(s), which is the optimal way of implementing any function).

6.3.3 Simulation of 3-input Function using Proposed Architecture

1-bit full adder using the stateful NOR based architecture is implemented and simulated using current threshold based TEAM Model and Kvatinsky Window [81]. The values of the various parameters used in the TEAM model are listed in Table 3.2. The proposed design has been simulated in Cadence Analog Design Environment (ADE) with Spectre simulator. The results are shown in Figures 6.8, 6.9, 6.10 and 6.11. To illustrate the pipelined nature of architecture, 1-bit pipelined full adder is also implemented and simulated using proposed architecture. The inputs and outputs for 1-bit pipelined full adder are given in Figures 6.12 and 6.13, respectively. The adder design given in [148, 149] use basic logic equations for sum and carry output and number of basic steps vary from function to function. However, the number of basic steps remain the same for any function in the proposed architecture.

For stateful NOR logic circuit implemented using memristors or CRSs to work correctly, the value of resistance R_G should be scaled down as the number of inputs n increases. All input memristors or CRSs are in parallel with each other while calculating effective voltage across them, and hence effective resistance decreases with number of inputs. In order to retain same voltage required for NOR operation, R_G should be also scaled down. For fixed value of R_G, the maximum number of inputs for successful NOR operation is given by

Algorithm 1: Stateful NOR based 3-input Logic Block design

Input: Array $I(n,m)$ where n:1 to 2^y, y is the number of inputs; m:1 to z, z is the number of outputs and $I(n,m)$ are the minterm; NOR blocks; PIORITY block

Output: Number of NOR blocks with number of inputs at each level,'netlist'-like connection pattern

$memristor(k) \leftarrow I(2^y,k)$ ▷ writing default values into the destination memristors or CRS

$NOR1a.I(x)(1\ to\ {}^yC_a) \leftarrow I(combinations\ of\ a\ from\ y\ inputs)$ ▷ a:1 to y, x:1 to a

$PRIORITY2((2^y)-1).I(u) \leftarrow NOR1(m).O(x)(1\ to\ {}^nC_m)$ ▷ u:1 to $((2^n)-1)$ ▷ All the m input NOR blocks are grouped together and these ${}^nC_a$ outputs are connected to consecutive inputs of the PRIORITY block in *opposition to fetching minterms consecutively*

for $i \leftarrow 1$ ***to*** z **do**
  for $j \leftarrow 1$ ***to*** 2^y-1 **do**
    if $I(j,i) != I(2^y,i)$ **then**
      Increment *number of inputs to* i^{th} LEVEL3 NOR block

for $i \leftarrow 1$ ***to*** z **do**
  for $j \leftarrow 1$ ***to*** 2^y-1 **do**
    if $I(j,i) != I(2^y,i)$ **then**
      $NOR3$(*number of inputs to* i^{th} LEVEL3 NOR block).I(pin_count) $\leftarrow PRIORITY2(2^y-1).Oj$
      Increment *pin_count*

for $i \leftarrow 1$ ***to*** z **do**
  $NOT3.Ii \leftarrow NOR3$(*number of inputs to* i^{th} *LEVEL3 NOR block*).Oi

for $i \leftarrow 1$ ***to*** z **do**
  $memristor(i) \leftarrow NOT3.Ii$

return *connection pattern*

1. $NOR1a.I(x)(1\ to\ {}^yC_m) \leftarrow I(combinations\ of\ a\ from\ y\ inputs)$ ▷ *connection pattern between inputs and NOR inputs*
2. $PRIORITY2(2^y-1).Id \leftarrow NOR1a.Obs$ ▷ *connection pattern between outputs of NOR blocks at LEVEL1 and PRIORITY block inputs*
3. $NOR3f.I(w)(k) \leftarrow PRIORITY2(2^y-1).Id$ ▷ *connection pattern between outputs of PRIORITY at LEVEL2 and NOR block inputs*

$$n \leq \frac{R_{\mathrm{LRS}}}{\frac{V_{\mathrm{COND}}R_{\mathrm{G}}}{V_{\mathrm{SET}}-I_{\mathrm{LRS}}R_{\mathrm{HRS}}}-R_{\mathrm{G}}} \tag{6.1}$$

Let us assume that the fan-in on the NOR block is n. Thus, the similar structure, as shown in the Figure 6.7 can be extended for n-input function. LEVEL1 consists of NOR blocks of each combination possible out of n-inputs. The total number of NOR blocks in LEVEL1 will be equal

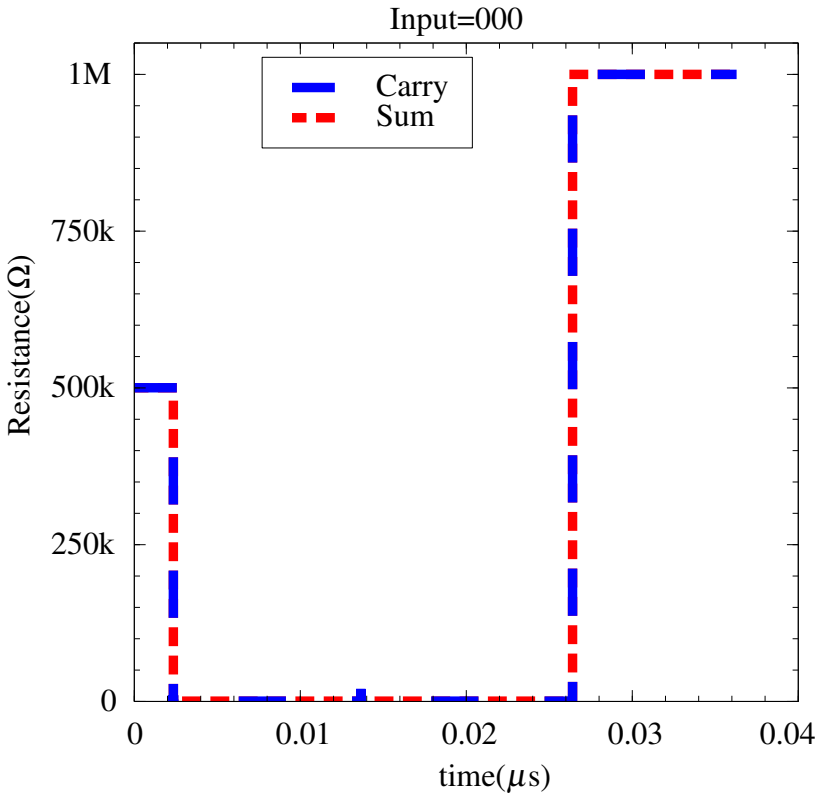


Figure 6.8. Simulation results for 1-bit full adder with inputs='000'; outputs:Sum='0'(1 MΩ), Carry='0'(1 MΩ).

to $2^n - 1$ ($n \leq$ fan-in of NOR gate). They are used to identify the input minterm (output is logic high only if all the inputs are logic low). Even though it looks huge for large input logic function, but it greatly reduces the further functionality of the circuit once the input minterm has been identified.

Here 3-input logic block has been designed because of constrain on fan-in (otherwise, R_G specified in Figure 4.20 has to be scaled accordingly). Logic function having more than 3-inputs can be realized by integrating 3-input blocks as is done in conventional FPGA.

6.4 *n*-Input Function Implementation

The disadvantage of the above architecture is that the fan-in is restricted according to the fan-in of NOR block (given by (6.1)). The 3-input logic blocks [LB3] can be multiplexed to convert it into a generalized architecture. The LB3 symbol is shown in Figure 6.14 and generalized architecture in Figure 6.15. Here, many such (LB3) blocks are used and the output is produced according to

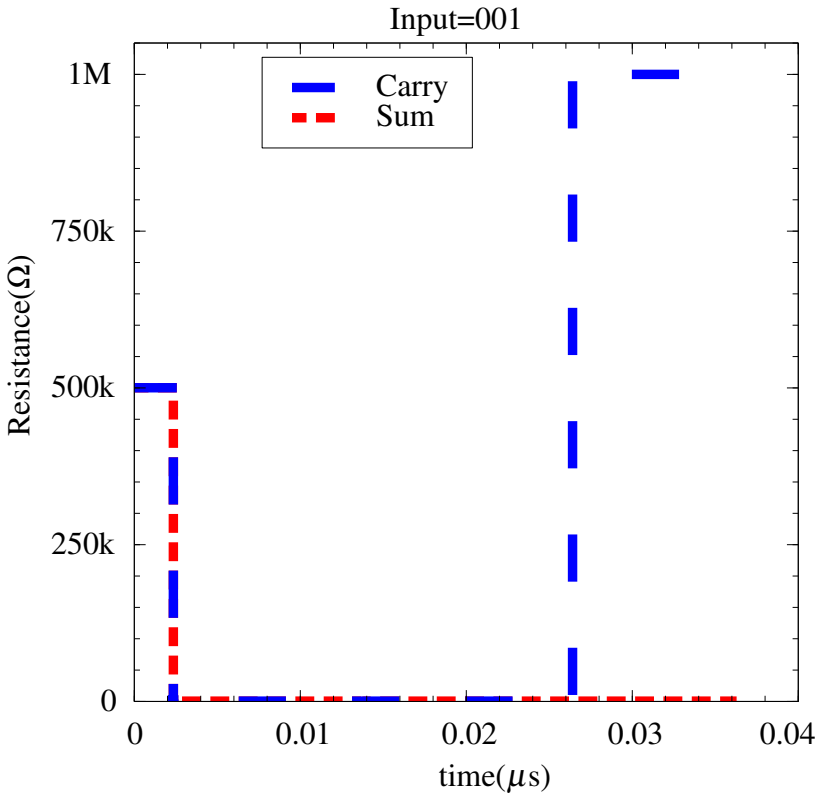


Figure 6.9. Simulation results for 1-bit full adder with inputs='001'; outputs:Sum='1'(100 Ω), Carry='0'(1 MΩ).

the inputs. Out of n-inputs lower three are connected to three inputs of each 3-input logic block (LB3) and remaining $n-3$ are decoded to use as selection line for one of many LB3s based on inputs. The automation algorithm for this is explained in the Section 6.4.1.

Even though the functions with large number of inputs seem to use huge number of memristors or CRS in this architecture, the block active at a time will be only single logic block (LB3) and rest of the area can be utilized for other functions. This greatly reduces the device count. Also note that the architecture is pipelined. Hence, the actual throughput of circuit is quite high. Also, the number of outputs will not impact the size of circuit to be configured in this architecture, which is against the conventional FPGA as single LUT can provide only single output in its basic form.

Consider an example of the logic function shown in Table 6.2, with an input combination having I_4=1, I_3=0, I_2=0, and I_1=1. Since the fan-in (n given by (6.1)) is 3 for proposed architecture, multiplexing of logic blocks is required as in Figure 6.15 for 4-input function. The working of the circuit for selected logic function is explained in Figure 6.16. It shows NOR M2, which is the minterm of interest out of all input combinations given in Table 6.2. I_4 is used for selection of LB3 block. The input data I_3, I_2, and I_1 is written on the input memristors or CRS of each

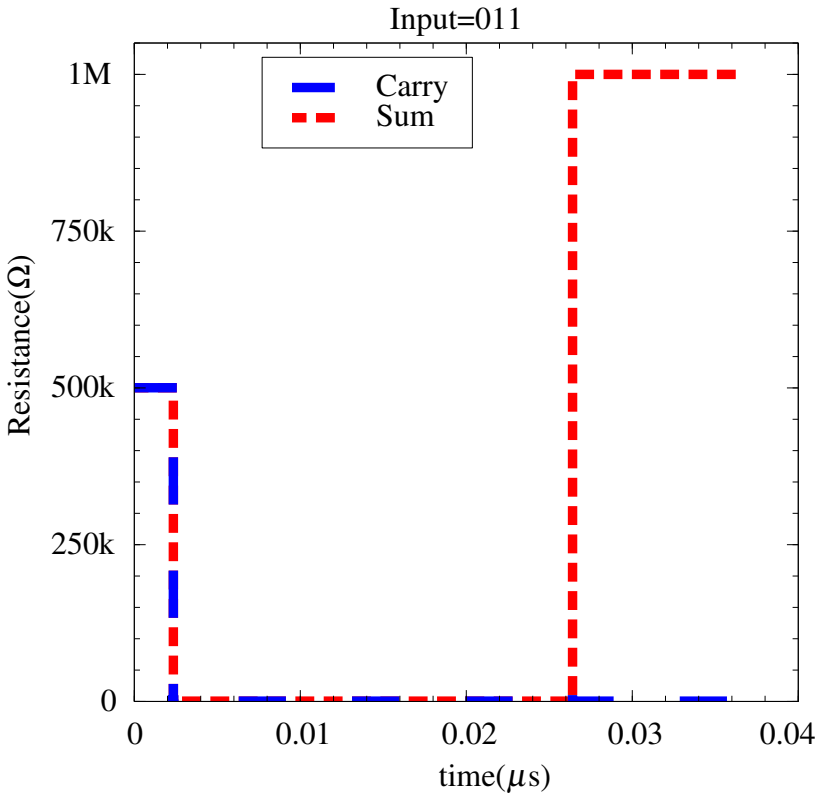


Figure 6.10. Simulation results for 1-bit full adder with inputs='011'; outputs:Sum='0'(1 MΩ), Carry='1'(100 Ω).

LB3 evaluation memristors or CRSs are cleared. Default values are written to the destination memristors or CRS (O1, O2, and O3). Then the appropriate voltages are applied to the input memristors or CRS and evaluate memristors or CRS in the evaluate phase. The output of each NOR block will be available in the read phase. In this case, the outputs D_1-D_7 (Figure 6.16) will be: D_1=0, D_2=1, D_3=0, D_4=0, D_5=1, D_6=1, D_7=0.

Here, the minterm is 001 and three of the NOR gates are giving high output. To identify the minterm, we would expect only one of the outputs to be high. To remove the insignificant bits of data (D_5 and D_6), priority circuit is used. Output of each NOR gate is the input to this circuit. The highest priority is given to D_1 and least to D_7. For the above example, only D_2 remains high and thus the minterm 001 has been identified.

Since the default data is written on the destination memristors or CRSs, it is expected to toggle them, if and only if the input minterm gives other output than the default data and are called decision minterms. Among the output minterms generated at LEVEL1, only decision minterms are given as an inputs to the NOR blocks in LEVEL2. If a particular NOR output is low, then it is concluded that the input minterm is the decision minterm. This can be inverted and given to the

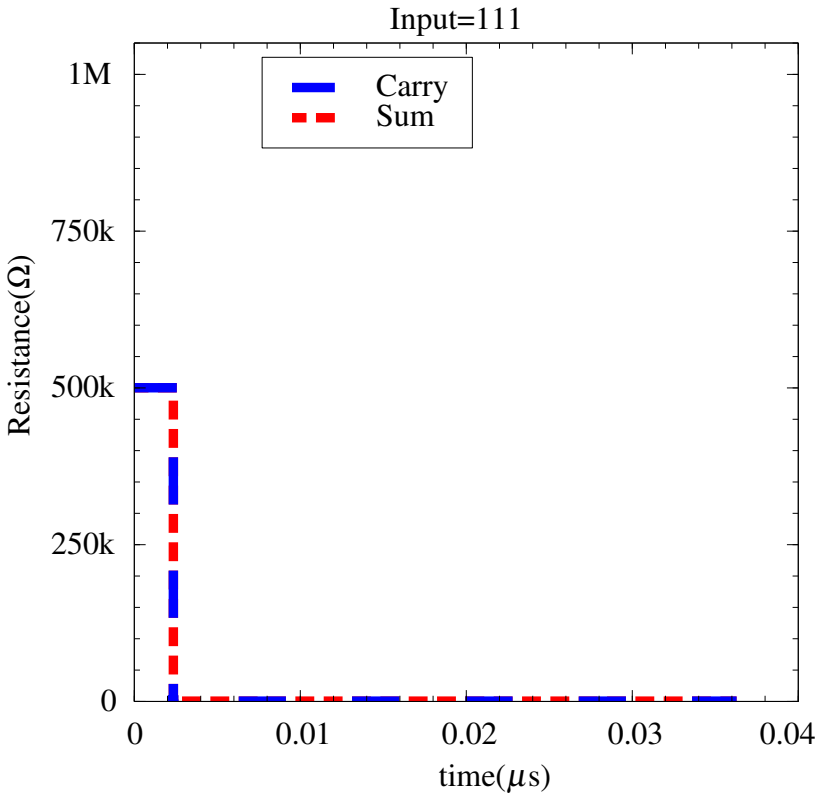


Figure 6.11. Simulation results for 1-bit full adder with inputs='111'; outputs:Sum='1'(100 Ω), Carry='1'(100 Ω).

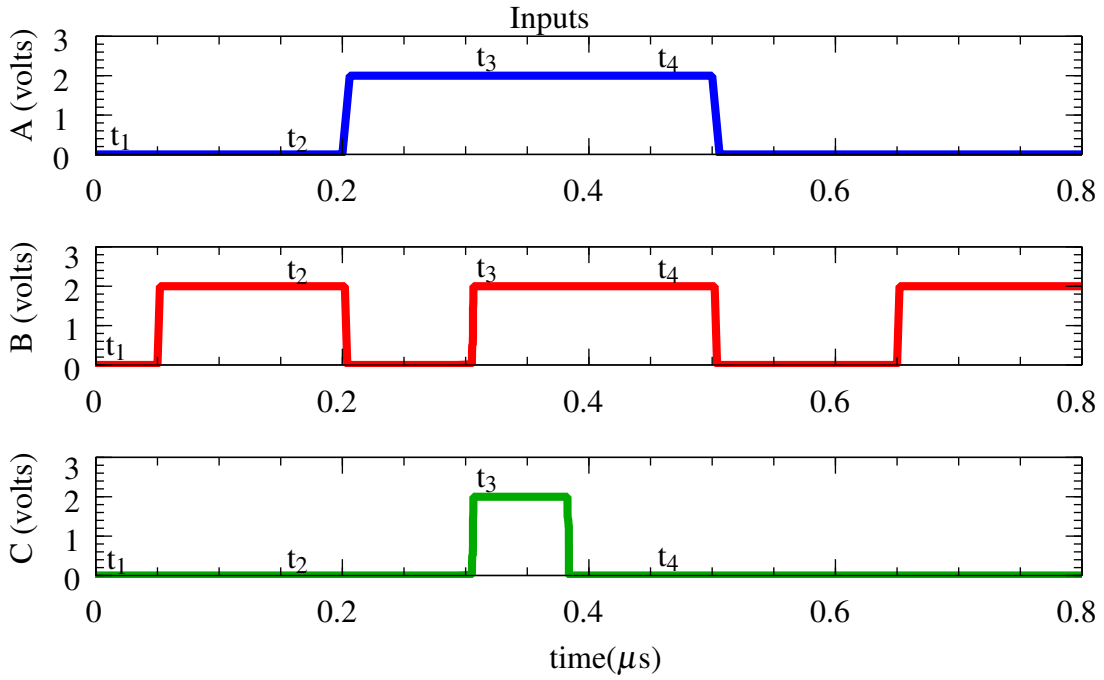


Figure 6.12. Input applied to 1-bit pipelined full adder implemented with proposed 3-input logic block architecture.

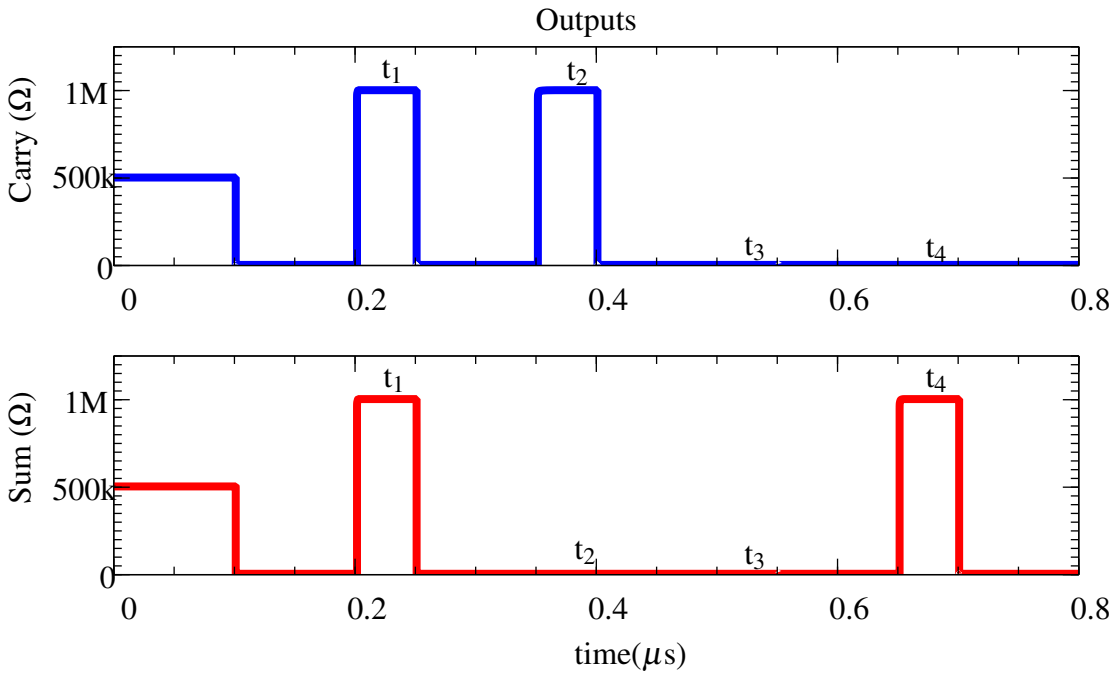


Figure 6.13. Output of 1-bit pipelined full adder for different inputs Output(t_1) (Sum='0',Carry='0') for inputs ABC='000', Output(t_2) (Sum='1',Carry='0') for inputs ABC='010', Output(t_3) (Sum='1',Carry='1') for inputs ABC='111', Output(t_4) (Sum='0',Carry='1') for inputs ABC='110'.

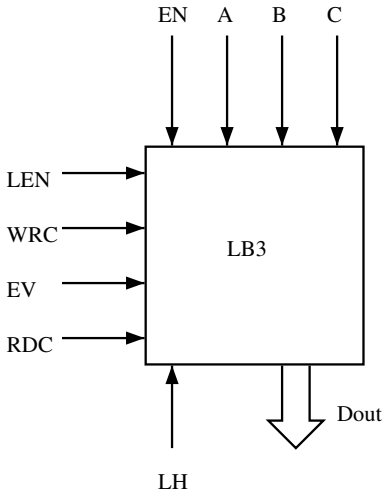


Figure 6.14. 3-input Logic Block (LB3) symbol for architecture shown in Figure 6.7.

enable signal to the write block of the destination memristor or CRS, which will toggle the state from the default value.

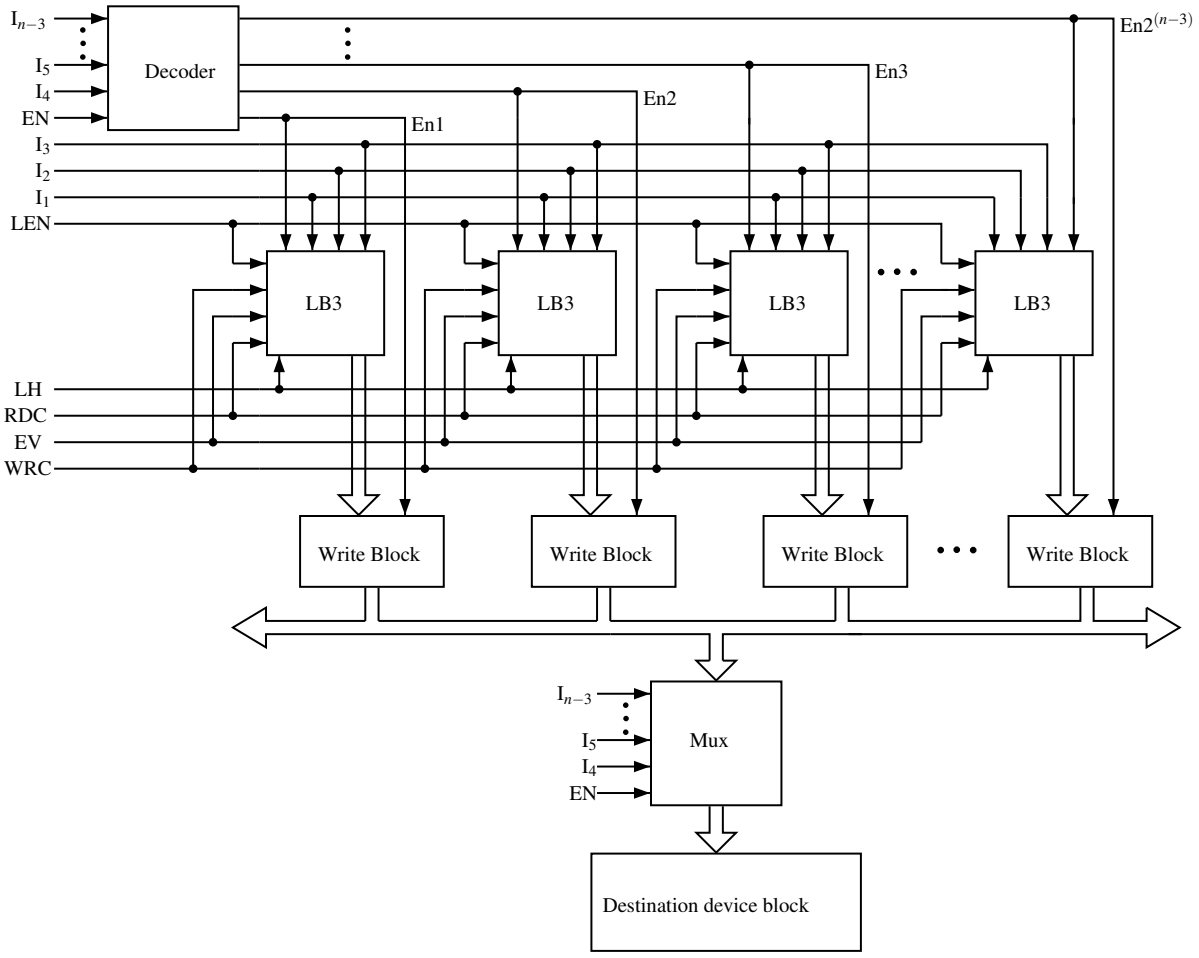


Figure 6.15. Generalized architecture to implement *n*-input logic function constructed from proposed reconfigurable 3-input logic block architecture using multiplexers.

6.4.1 Automation Algorithm for Generalized Architecture for *n*-Input Function Implementation

The automation algorithm for generating a netlist-like pattern for a y input and z output function has been described in Algorithm 2. In this algorithm the nomenclature used for the decoder is DECODER(I/O)p, where I/O indicates whether it is an input or an output pin and p represents the pin number. A similar representation is used for the multiplexer used at the end of the architecture-MUXt(I/O/sel)s, where t is the type of signal being multiplexed. If t=1, the write signal is being multiplexed and if t=2, the default data (result corresponding to all '1' logic) is being multiplexed. Further, I/O/sel denotes whether the pin is an input, an output pin or a select pin and s represents the serial number of the multiplexer.

Logic blocks are represented as logic BLOCKs(en/wr/default value)(w) where s is the serial

Table 6.2. Truth table of a random logic function.

Inputs				**Outputs**		
$\mathbf{I_4}$	$\mathbf{I_3}$	$\mathbf{I_2}$	$\mathbf{I_1}$	$\mathbf{O_1}$	$\mathbf{O_2}$	$\mathbf{O_3}$
0	0	0	0	1	0	0
0	0	0	1	1	0	1
0	0	1	0	0	1	0
0	0	1	1	1	0	1
0	1	0	0	0	0	0
0	1	0	1	0	1	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	0	1	1
1	0	0	1	0	0	1
1	0	1	0	1	1	1
1	0	1	1	0	1	1
1	1	0	0	1	1	0
1	1	0	1	0	0	0
1	1	1	0	0	0	1
1	1	1	1	0	1	1

number of the logic block, en/wr/default value is the type of signal transmitted on that category of pins and w is the serial number of that pin among the particular cateory of pins that transmit the same type of signal. The representation used for the destination write block is Destination Write Block(wr/default value/O)(w) where wr/default value/O is the type of signal transmitted on that wire and w is the serial number of that pin among the particular cateory of pins that transmit the same type of signal. The destination memristors or CRSs are denoted by destination memristors(m) or CRS(m) where m denotes the serial number of the output of the entire function that the architecture resolves. Moreover, when generating the netlist between destination write block and the destination memristors or CRS, a nomenclature ABC(HIGH,LOW) has been used where ABC is the standard representation and HIGH,LOW represents the two terminals of the memristor or CRS.

Inputs: Array $I(n,m)$: where, n:1 to 2^y, y is the number of inputs, m:1 to z, z is the number of outputs and $I(n,m)$ are the minterms; decoder; multiplexers; logic block; destination write block; destination memristors or CRS.

Outputs: 'netlist'- like connection pattern.

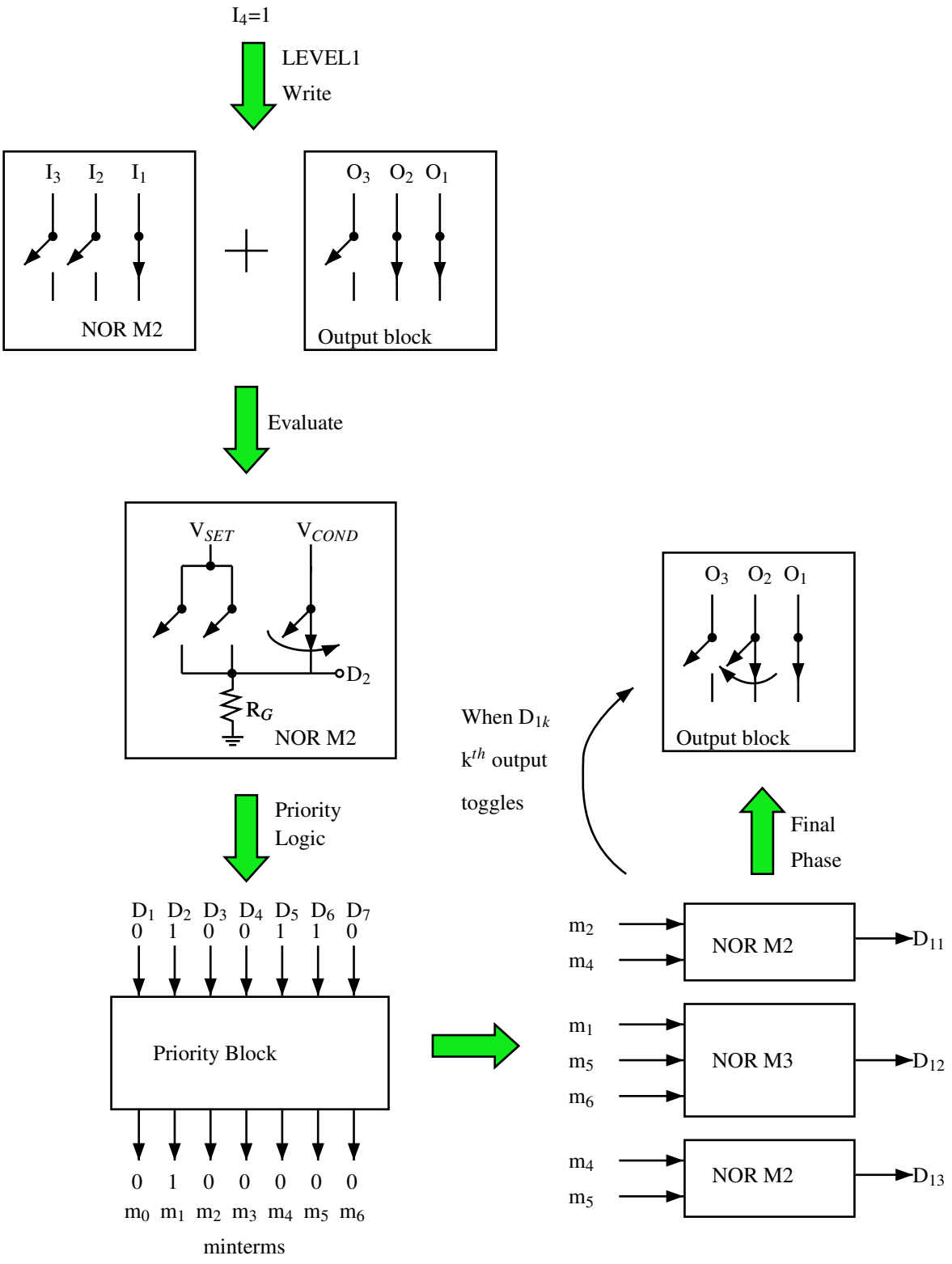
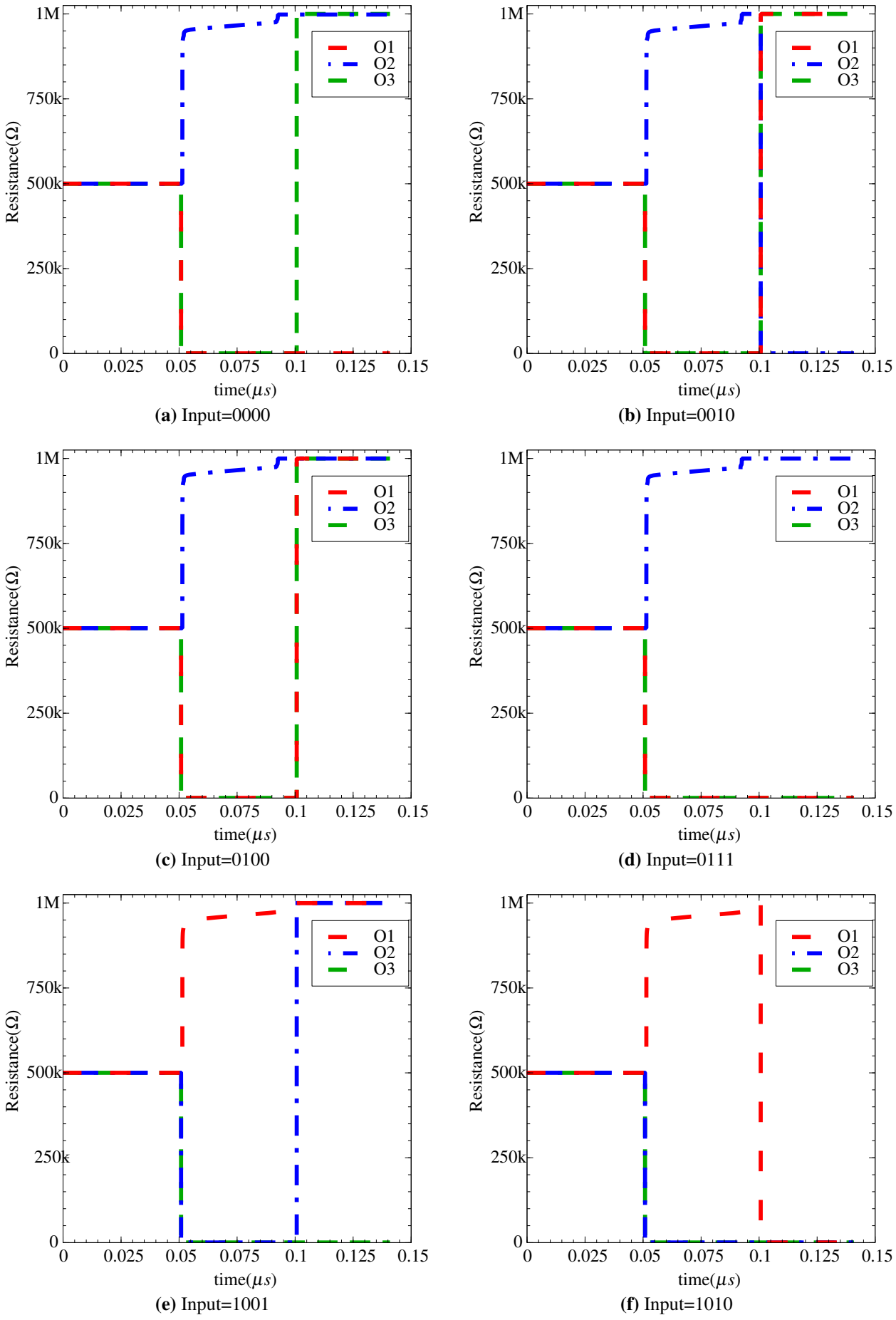


Figure 6.16. Flow of the functionality of architecture to implement function shown in Table 6.2 for inputs '1001'.



(a) Input=0000 **(b)** Input=0010

(c) Input=0100 **(d)** Input=0111

(e) Input=1001 **(f)** Input=1010

Figure 6.17. The simulation results for random logic function given in Table 6.2 implemented on *n*-input generalized architecture.

Algorithm 2: Automation Algorithm for n-input logic function using 3-input Logic Block architecture

Input: Array $I(n,m)$ where n:1 to 2^y, y is the number of inputs; m:1 to z, z is the number of outputs and $I(n,m)$ are the minterms; x-input,z-output logic BLOCK; $y-x$ input decoder

Output: 'netlist'-like connection pattern

for $n \leftarrow 1$ ***to*** $y-x$ **do**
  $DecoderI(i) \leftarrow Input(x+i)$
for $j \leftarrow 1$ ***to*** 2^{y-x} **do**
  $logicBLOCK(j)(en) \leftarrow DecoderO(j)$
logic BLOCK$(1\ to\ 2^{y-x})(data(1\ to\ x)) \leftarrow$**Input**$(1\ to\ x)$
Mux1$(1\ to\ z)I(1\ to\ 2^{y-x}) \leftarrow$**logic BLOCK**$(1\ to\ 2^{y-x})(Wr(1\ to\ z))$
Mux1$(1\ to\ z)(En)(1\ to\ 2^{y-x}) \leftarrow$**Decoder**$O(1\ to\ 2^{y-x})$
Mux2$(1\ to\ z)I(1\ to\ 2^{y-x}) \leftarrow$**logic BLOCK**$(1\ to\ 2^{y-x})(Defaultvalue(1\ to\ z))$
Mux2$(1\ to\ z)(En)(1\ to\ 2^{y-x}) \leftarrow$ **Decoder**$O(1\ to\ 2^{y-x})$
Destination Write BLOCK$(Wr(1\ to\ z)) \leftarrow$ **Mux1**$(1\ to\ z)O(1\ to\ 2^{y-x})$
Destination Write BLOCK$(Defaultvalue(1\ to\ z))$
  $\leftarrow$**Mux2**$(1\ to\ z)O(1\ to\ 2^{y-x})$
Destination memristor(m) or **Destination CRS**$(m)(HIGH,LOW) \leftarrow$ **Destination Write BLOCK**$O(m)(HIGH,LOW)$
return *connection pattern;*

1. $DecoderI(i)(1\ to\ y-x) \leftarrow Input(x\ to\ y)$ ▷ *connection pattern between inputs and Decoder inputs;*
2. $logic\ BLOCK(1\ to\ 2^{y-x})(en) \leftarrow DecoderO(i)(1\ to\ 2^{y-x})$ ▷ *connection pattern between outputs of Decoder and logic block inputs;*
3. $logic\ BLOCK(1\ to\ 2^{y-x})(data) \leftarrow Input(1\ to\ x)$ ▷ *connection pattern between input lines and logic BLOCK inputs;*
4. $Destination\ memristor(1\ to\ z)(HIGH,LOW) \leftarrow$ $logic\ BLOCK(1\ to\ 2^{y-x})O(1\ to\ z)(HIGH,LOW)$ ▷ *connection pattern between outputs of logic BLOCKS and destination memristors or CRS;*

6.5 Summary

As memristor is passive circuit element (and hence CRS as it is made up of antiserial memristors), supporting CMOS circuits are required to implement any operation on memristive or CRS crossbar. All such common blocks are discussed in the initial section. Using the stateful NOR as basic operation in memristive crossbar (Chapter 4) or in CRS crossbar (Chapter 5), pipelined reconfigurable architecture for implementing 3-input logic function is developed. Automation algorithm for this architecture, with truth table as input and netlisting of different blocks used in

the architecture as output, is presented. The architecture and automation algorithm for any *n*-input function, using 3-input stateful NOR logic block, are described. Simulation results of 1-bit full adder, 1-bit pipelined full adder and 4-input random function are also presented.

Next chapter evaluates the performance of proposed reconfigurable pipelined architecture in terms of time (delay), power dissipation (energy) and area, and compares with LUT based CLB used in most commercial FPGAs.

Chapter 7

Performance Analysis

7.1 Introduction

In this chapter, performance analysis of the proposed 3-input 1-output reconfigurable architecture implemented in memristive as well as CRS crossbar is carried out in terms of delay (timing), energy (power) and area, and compared with existing LUT based 3-input CLB architectures used in conventional FPGA.

The memristive crossbar array (Chapter 4) and CRS crossbar array (Chapter 5) are generally part of ReRAM. The general organization of memristive/CRS based memory (ReRAM) is shown in Figure 7.1 [150]. It consists memory banks, each bank being subdivided into mats. Mats are again divided into subarrays which are basic elements of memory. In the analysis, the subarray size is restricted to 128×128. For analysis purpose, memristor is modeled as state resistance in parallel with capacitor of value 0.12 pF while CRS is modeled as state resistance in parallel with capacitor of value 0.24 pF [76].

7.2 Timing Analysis

The switching time of memristor from LRS to HRS and from HRS to LRS is important in timing analysis. The switching time for memristor is given as 120 ps [151]. The switching delays based

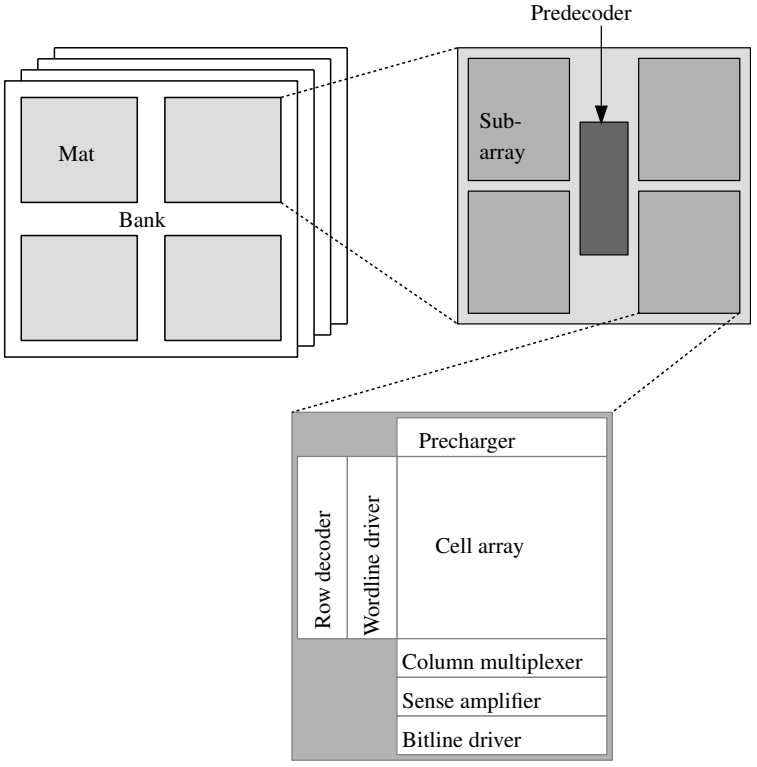


Figure 7.1. The generic ReRAM structure consists of banks made up of small sections called mats. Mats are further divided into subarrays which is the smallest unit of memory. The supporting circuit components such as multiplexers, drivers, sense amplifiers and prechargers are designed in CMOS layer while memory elements are in crossbar array situated above CMOS layer.

on the physical properties of material and process involved in working of a memristor are given by L. Zhang et al. [152] and are given below. Turn-on time (HRS to LRS) is given by

$$t_{\text{on}} = \frac{{C_0}^2}{2\gamma_t \beta V \lambda}, \tag{7.1}$$

where t_{on} = time for R_{HRS} to R_{LRS} transition

and turn-off time (LRS to HRS) is given by

$$t_{\text{off}} = \frac{\left[\left(1-\frac{n_{\text{c}}}{n_{\text{i}}}\right)D + 2C_0\right]\left(1-\frac{n_{\text{c}}}{n_{\text{i}}}\right)D}{2\gamma_t \beta \lambda V}, \tag{7.2}$$

where t_{off} = time for R_{LRS} to R_{HRS} transition,

V = voltage across memristor.

The parameter C_0 used in (7.1) and (7.2) is given by

$$C_0 = \left(\frac{\lambda}{2}\right) + \left(\frac{n_c}{n_i}\right)\left(D - \frac{\lambda}{2}\right), \tag{7.3}$$

where λ = transition/conduction/insulation region length,

n_c, n_t, n_i = conduction, transition and insulation region concentrations, respectively,

D = thickness of insulating material (TiO_2) in memristor,

γ_t = electron generating coefficient in the transition region.

The rate of generation of electrons in transition region is given by

$$\frac{dn_t}{dt} = \gamma_t \left(n_c - n_i(t)\right) E_t(t) \tag{7.4}$$

where $E_t(t)$ is the electric field in the transition region.
The parameter β used in (7.1) and (7.2) is given by

$$\beta = \frac{1}{4}\left(\frac{n_c}{n_i} - 1\right)^3 + \frac{2}{3}\left(\frac{n_c}{n_i} - 1\right)^2 + \frac{1}{2}\left(\frac{n_c}{n_i} - 1\right). \tag{7.5}$$

The parameter β has no physical significance.

The voltage across memristor in the write cycle is fixed. However, the voltage in the evaluate cycle varies with the number of inputs to the block as given below:

$$V = V_{SET} - V_{COND}\left(\frac{R_G}{R_G + \left(\frac{R_{HRS}}{n-k} || \frac{R_{LRS}}{k}\right)}\right), \tag{7.6}$$

where k is the number of memristors in LRS state and n= the number of inputs satisfying (6.1). For maximum value of V=V_{max}, $k = 0$. Therefore the minimum switching times for memristor are given by

$$t_{on_{min}} = \frac{{C_0}^2}{2\gamma_t \beta V_{max} \lambda}, \tag{7.7}$$

and

$$t_{\text{off}_{\min}} = \frac{\left[\left(1-\frac{n_c}{n_i}\right)D+2C_0\right]\left(1-\frac{n_c}{n_i}\right)D}{2\gamma_t\beta\lambda V_{\max}} \tag{7.8}$$

Using (7.7) and (7.8) with physical parameters given in [152] calculated the turn-on delay is 200 *ps* and turn-off delay is 60 *ps* even for intrinsic concentration of TiO_2 as 10^{14} (the standard value of n_i for TiO_2 is 1 m^{-3} whereas in [152], it is given as function of electric field with condition $n_i \ll n_c$, n_c being equal to 8.75×10^{22} m^{-3}). The graphs for turn-on and turn-off time versus n_i are given in Figures 7.2 and 7.3, respectively using above equations. In [151] the reported value of memristor switching is 120 ps which is close to average value calculated here. The set and reset time for tantalum oxide based memristors are 105 and 120 ps, respectively [76, 153].

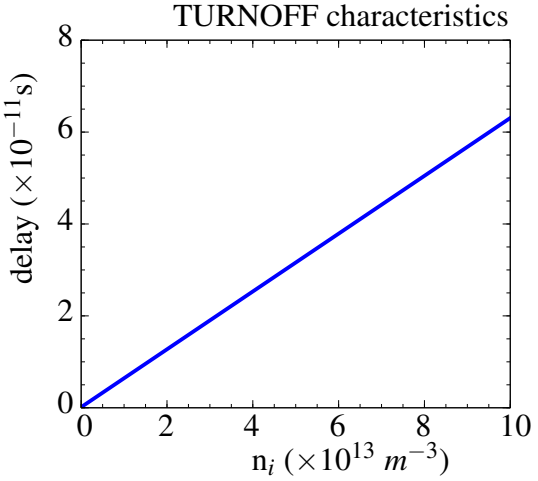


Figure 7.2. Turn-off delay as function of intrinsic concentration of TiO_2.

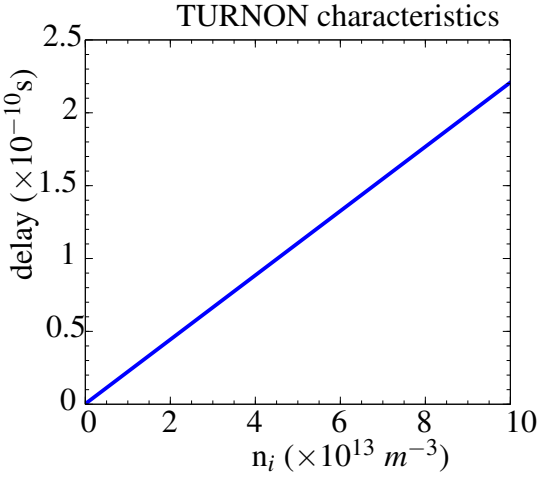


Figure 7.3. Turn-on delay as function of intrinsic concentration of TiO_2.

The timing diagram for the proposed architecture is shown in Figure 7.4. The architecture consists of 3 levels, the first and third is built from stateful NOR gates while second level is made up

of priority logic. We know that logic '0' and logic '1' can not be written simultaneously in memristive/CRS crossbar. Hence when WRC (write control signal) is active, data bits at logic '0' are written to memristors/CRSs by forcing LH control signal at logic '0' and then data bits at logic '1' are written to other memristors/CRSs by forcing LH control signal at logic '1'. In the timing diagram, the complete processing of one set of data are shown with same background color. The input data to each level is shown as 'DataIn N' , where N is level number. Control signals are shown as 'WRC N' (write control signal), 'EV N' (evaluate control signal), 'RDC N' (read control signal) and 'LH N' (low/high write control). The data read from first level is used as write data for third level after passing it through priority logic. The time taken by a single set of data for execution of an operation is given by

$$t_{\text{total}} = t_{\text{write\_LEVEL1}} + t_{\text{eval\_LEVEL1}} + t_{\text{read\_LEVEL1}} + t_{\text{eval\_LEVEL3}} + t_{\text{read\_LEVEL3}}, \tag{7.9}$$

with the condition that

$$t_{\text{read\_LEVEL1}} \geqslant t_{\text{priority\_block}} + t_{\text{write}}, \tag{7.10}$$

and

$$t_{\text{read\_LEVEL3}} \geqslant t_{\text{write}} \tag{7.11}$$

where,

$$t_{\text{write}} = t_{\text{write\_CMOS\_circuit}} + t_{\text{memristors/CRSs\_switching\_logic'0'}} + t_{\text{memristors/CRSs\_switching\_logic'1'}}, \tag{7.12}$$

$$t_{\text{eval}} = t_{\text{eval\_CMOS\_circuit}} + t_{\text{Memristor/CRS\_switching}} = t_{\text{memristive/CRS\_stateful\_NOR}}, \tag{7.13}$$

$$t_{\text{read}} = t_{\text{read\_CMOS\_circuit}}. \tag{7.14}$$

The time delay of write CMOS circuit, evaluation CMOS circuit and read CMOS circuit described in previous chapter are shown as $t_{\text{write\_CMOS\_circuit}}$, $t_{\text{eval\_CMOS\_circuit}}$ and $t_{\text{read\_CMOS\_circuit}}$ and measured with respect to activation of control signals WRC, EV and RDC, respectively. The read data of first level is written into third level of proposed reconfigurable architecture after passing it through priority logic (second level) and hence read control signal of first level 'RDC1' should be kept active until write operation of third level is complete. The read control signal of third level RDC3 should be kept active until final result of operation is written to destination

memristor(s)/CRS(s).

Further, when it is required to apply the same configured function on n sets of data, the time taken need not be $n \times t_{\text{total}}$. Since the operation can be pipelined, the next data can be manipulated in the $(k-1)^{th}$ step when the previous data is being processed in k^{th} step. Thus, the total timing for the operation would be:

$$t_{\text{pipelined}} = t_{\text{total}} + (n-1) \times t_{\text{LEVEL1}}, \tag{7.15}$$

where $t_{\text{LEVEL1}} = t_{\text{write}} + t_{\text{eval}} + t_{\text{read}}$. The overlapping of execution can be further extended by separating destination memristor/CRS from NOR block (and adding another destination memristor/CRS in place of it) after evaluate operation, so that read operation of NOR result and writing of new data for next NOR operation can be done simultaneously. This will reduce the execution time but at the cost of complex CMOS circuit.

The delay for various operations performed in memristive and CRS crossbar is given in the Table 7.1. The results are obtained using methods used in [150] for array size of 128 $\times$ 128.

Table 7.1. Delay in implementation of write, read and evaluate operation implemented in resistive crossbar (memristive and CRS crossbar) of size 128 $\times$ 128.

Sr. No.	**Operation**	**Delay** (*ns*)	
		Memristive crossbar	CRS crossbar
1	Write	3.12	3.68
2	Read	0.95	1.31
3	Evaluate	3.73	4.02

The delay in various components of LUT based 3-input CLB is given in Table 7.2 using 65 *nm* CMOS technology.

Table 7.2. Delay in various components of LUT based 3-input CLB using CMOS 65 *nm* technology

Sr. No.	**Unit**	**Delay** (*ns*)
1	MUX 2:1	0.29
2	MUX 4:1	0.43
3	MUX 8:1	0.60
4	SRAM	3.25
5	DFF	2.68
6	BUF	0.40

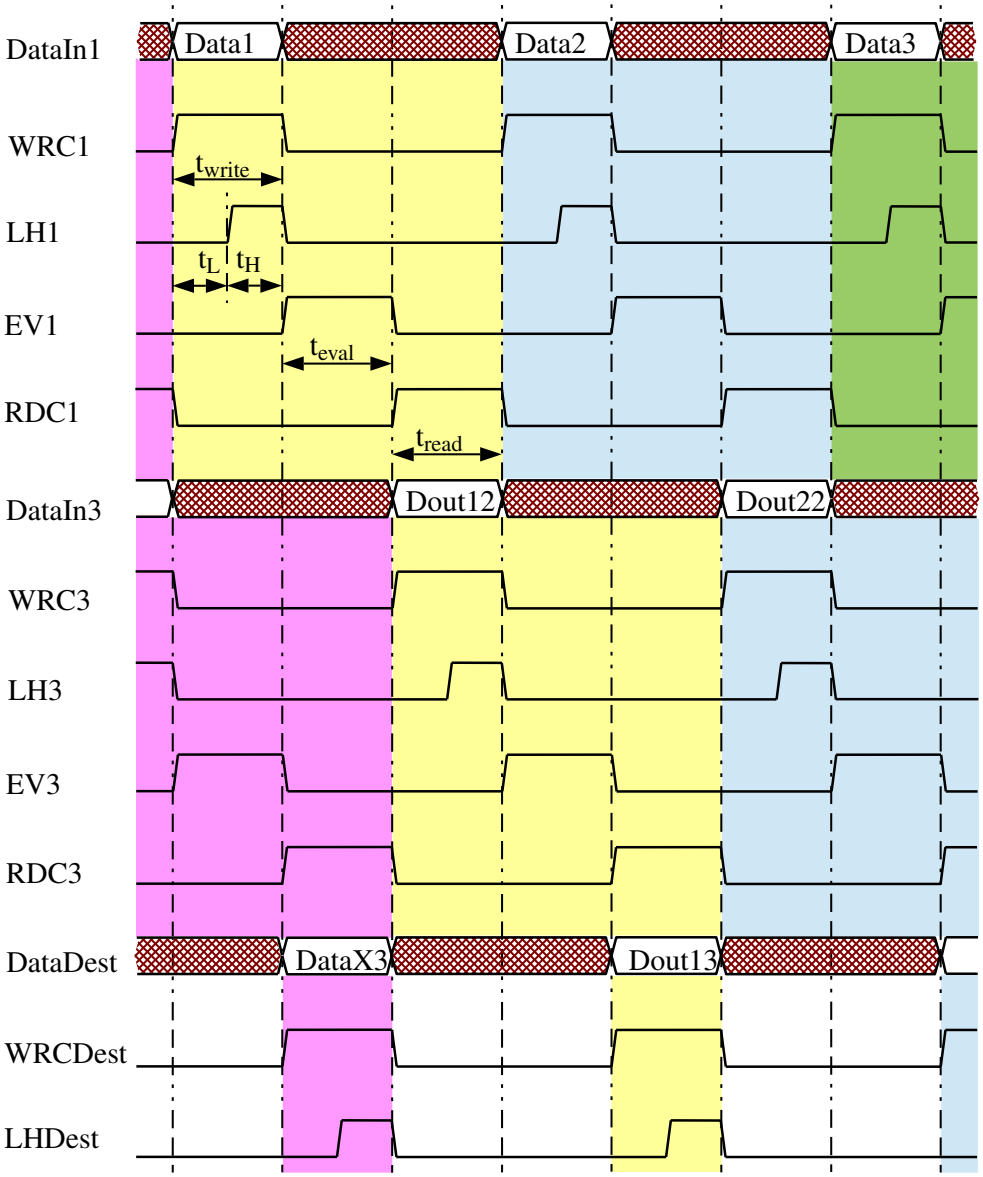


Figure 7.4. Timing diagram for proposed pipelined architecture. The complete processing of one set of data are shown with same background color. Data input to each level is shown by 'DataIn N', while control signals are shown as 'WRC N' (write control signal), 'EV N' (evaluate contol signal), 'RDC N' (read control signal) and 'LH N' (low/high write control), where N is level number. When LH is logic low, all data bits at logic '0' will be written to memristors/CRSs while when LH is logic high, all data bits at logic '1' will be written to memristors/CRSs simultaneously. The final result to be written to destination memristor(s)/CRS(s) along with WRC and LH control signal are also shown (with N=Dest). Level 2 is not shown in timing diagram as it consists of only priority logic and its delay is considered as part of read logic.

Overall delay in the implementation of 3-input logic function on conventional LUT based FPGA and with proposed architecture on memristive and CRS crossbar is given in Table 7.3. For implementation on LUT based FPGA, the routing delay is also taken into account.

It can be seen from Table 7.3 that LUT based 3-input CLB used in conventional FPGA is better

Table 7.3. The delay in LUT based 3-input CLB, proposed 3-input architecture implemented in memristive and CRS crossbar. In LUT based CLB implementation, routing delay is taken into account.

Delay in LUT based 3-input CLB used in FPGA (*ns*)	**Delay in proposed 3-input 1-output reconfigurable architecture implemented in memristive crossbar (*ns*)**	**Delay in proposed 3-input 1-output reconfigurable architecture implemented in CRS crossbar (*ns*)**
14.5	16.82	19.08

for delay than the proposed architecture implemented both in memristive and CRS crossbar. But as the length of data set to be operated on becomes bigger, the proposed architecture becomes better because of its pipelined nature. The graph of delay verses length of data set (n) is shown in Figure 7.5 for LUT based CLB and for proposed architecture implemented both in memristive and CRS crossbar. For larger data set this improvement is $1.28\times$ for CRS crossbar and $1.45\times$ for memristive crossbar over LUT based CLB architecture. Also if function is to be reconfigured over LUT based CLB, loading configuration data to configuration memory takes lot of time as writing to configuration memory is done sequentially.

7.3 Power Analysis

The power consumed by different operations in crossbar is summarized in Table 7.4. For write operation, 1/3 write scheme is chosen as it maintains the integrity of other memristors/CRSs. The reading of CRS is done with the help of self resetting read scheme explained in Chapter 5. The power consumed for evaluate (NOR) and write operations in crossbar is taken from results in Chapter 4 and 5 by considering the array size to be 128×128.

The power consumed by different elements in conventional LUT based 3-input CLB is given in Table 7.5.

The power consumed by LUT based 3-input CLB and proposed 3-input 1-output reconfigurable architecture implemented in memristive and CRS crossbar is given in Table 7.6. CMOS 65 *nm* technology is used in calculation of power dissipation for LUT based 3-input CLB used in conventional SRAM based FPGA. For proposed 3-input 1-output architecture, power dissipation in all common CMOS blocks with 65 *nm* technology node is calculated along with power dissipation

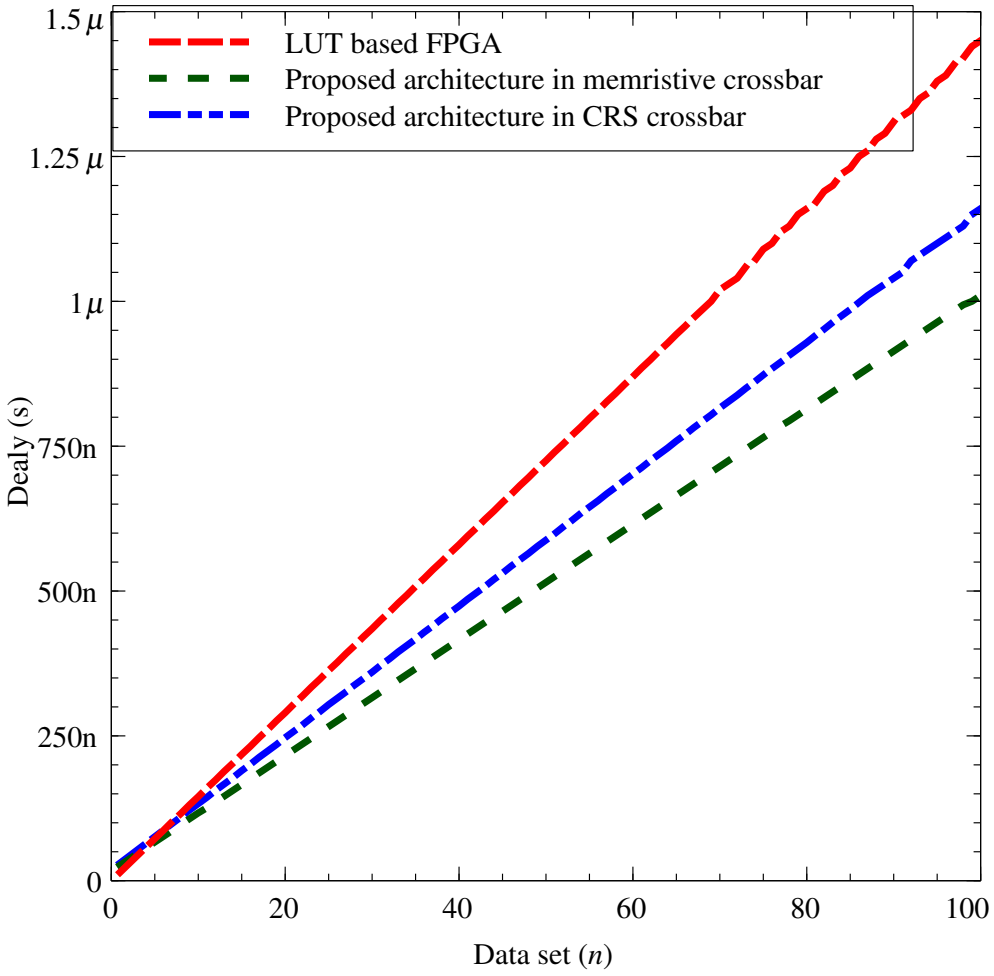


Figure 7.5. Delay verses length of data set(n) for LUT based CLB used in FPGA, proposed architecture implemented in memristive and CRS crossbar. As length of data set increases, proposed pipelined architecture shows improvement in delay over LUT based FPGA. Also the proposed architecture is better if function changes because CLB needs to be reconfigured and the configuration data is written sequentially to configuration memory.

in crossbar. The power dissipation for implementation of proposed architecture in CRS crossbar is 84.04 mW against that of implementation in memristive crossbar, where it is 64.02 mW. But still CRS crossbar implementation is better as the voltages used in write operation are double for CRS crossbar in comparison to memristive crossbar, and for evaluate operation the voltage used is three times higher in CRS crossbar to that used in memristive crossbar. Also, the sneak path problem limits the size of memristive crossbar for performing read and evaluate operations.

The proposed architecture provides 2.68× improvement in power consumption for implementation in memristive crossbar and 2.04× for implementation in CRS crossbar over LUT based CLB architecture.

Table 7.4. Maximum power consumption in resistive crossbar(memristive and CRS crossbar) for different operations to implement logic functions for array size of 128 × 128. For write operation, V_{WRITE}= ±1 V for memristive crossbar and V_{WRITE}=±1.5 V for CRS crossbar. Read operation for memristor applies voltage less than $V_{th,M}$=0.7 V while for CRS, $V_{th1,C} < V_{READ} < V_{th2,C}$ where $V_{th1,C}$=0.7 V and $V_{th2,C}$=1.4 V. For evaluate operation, V_{SET}= 1V and V_{COND}= 0.8 V for memristive crossbar while for CRS, V_x=3 V. All notations for voltages are defined in Chapters 4 and 5.

Sr. No.	**Operation**	**Power dissipation** (*mW*)				
		Resistive crossbar			**CMOS block**	**Total**
1	Write	Memristive crossbar	Floating write scheme	2.03	1.03	4.93
			1/3 write scheme	3.9		
		CRS crossbar	Floating write scheme	0.142	1.4	5.49
			1/3 write scheme	4.09		
			configuration row 1/3 write scheme	3.9		
2	Read	Memristor		1.08	0.8	1.88
		CRS	3-step reading scheme	1.64	1.03	2.67
			Self-resetting scheme	1.15	0.9	2.05
3	Evaluate	Memristive crossbar		0.57	1.06	1.63
		CRS crossbar		1.03	1.4	2.43

Table 7.5. Power consumed by different components in conventional LUT based 3-input CLB and programmable interconnection switch using CMOS 65 *nm* technology. The structure of CLB is shown in Figure 1.3 while that of programmable interconnection switch is shown in Figure 1.5

Sr. No.	**Unit**	**Power consumption** (*mW*)
1	MUX 2:1	0.83
2	MUX 4:1	2.29
3	MUX 8:1	6.33
4	SRAM	1.43
5	DFF	4.5
6	BUF	1.03

7.4 Area Analysis

The transistor count for LUT based 3-input 1-output CLB (see Figure 1.3) used in conventional FPGAs is given in Table 7.7 while that for programmable interconnect switch (Figure 1.5) is shown in Table 7.8. In order to implement any 3-input 1-output logic function on LUT based CLB, one

Table 7.6. Power consumption in LUT based 3-input CLB, proposed 3-input 1-output reconfigurable architecture implemented in memristive and CRS crossbar. The power consumption in LUT based CLB is using 65 *nm* CMOS technology. While the power dissipation of proposed architecture implemented in CRS crossbar is slightly higher than implementation in memristive crossbar, the voltage values used in write and evaluate operations in CRS crossbar are higher than in memristive crossbar as CRS is made up of antiserially connected two memristors.

Power consumption in LUT based 3-input CLB used in FPGA (mW)	**Power consumption in proposed 3-input 1-output reconfigurable architecture implemented in memristive crossbar (mW)**	**Power consumption in proposed 3-input 1-output reconfigurable architecture implemented in CRS crossbar (mW)**
171.58	64.02	84.04

Table 7.7. Transistor count for LUT based 3-input 1-output CLB used in conventional FPGAs shown in Figure 1.3. In this design 6T SRAM cell is considered while other components are designed using CMOS process. Buffers (BUF) are bidirectional.

Unit	**No. of MOSFETs/unit**	**No. of Units**	**Total count**
SRAM	6	19	114
MUX 2:1	10	1	10
MUX 4:1	30	3	90
MUX 8:1	70	1	70
DFF	26	1	26
BUF	8	4	32

CLB and at least two programmable interconnection switches are required, one for directing input and one for sending output of logic function to other devices. In this analysis, write circuit for loading configuration data and LUTs is not taken into account.

Table 7.8. Transistor count for programmable interconnection switch used in conventional FPGAs shown in Figure 1.5. In this design 6T SRAM cell is considered while other components are designed using CMOS process. Buffers (BUF) are bidirectional.

Unit	**No. of MOSFETs/unit**	No. of Units	Total count
SRAM	6	12	72
MUX 4:1	30	4	120
BUF	8	4	32

The device count for proposed reconfigurable 3-input logic block architecture shown in Figure 6.7 is given in Table 7.9. Memristor and CRS count is different because memristive architecture requires one additional memristor in addition to input memristors for performing stateful-NOR operation, while in case of CRS based architectures, destination CRS is one of the input CRSs. The output from priority block needs to be grouped into two NOR gates, one producing output

Table 7.9. Device count for proposed reconfigurable 3-input 1-output memristive/CRS crossbar based architecture shown in Figure 6.7.

Unit	No. of MOSFETs per unit	No. of Units	Total count	Memristors	CRSs
LEVEL1					
Write circuit	30	1	30	19	15
Read circuit	9	7	63		
Evaluate circuit	8	7	56		
LEVEL2					
Priority circuit	92	1	92	-	-
LEVEL3					
Write circuit	30	1	30	12	10
Read circuit	9	4	36		
Evaluate circuit	8	4	32		
Destination					
Write circuit	30	1	30	1	1
Reconfiguration components					
Grouping units	2	8	16	8	8
Other components					
AND gates	6	8	48	-	-
Inverters	2	1	2	-	-

logic '0' and the other producing output logic '1'. The memristors/CRSs and transistors needed for it are shown as reconfiguration components in the Table 7.9.

If fan-in of stateful-NOR is not restricted to 3, the logic block with more than three inputs can be designed, but device count increases exponentially. If the logic function has n inputs, the number of NOR blocks in LEVEL1 is given by

$$ {}^{n}C_{\mathrm{n}} + {}^{n}C_{\mathrm{n\text{-}1}} + \ldots + {}^{n}C_{2} + {}^{n}C_{1} = 2^{n} - 1, \tag{7.16} $$

where ${}^{n}C_{\mathrm{k}}$ represents the binomial coefficient.

Each k-input NOR block consists of k input memristors and one evaluation memristor. Hence, the total number of memristors in LEVEL1 of the memristive architecture of an n-input logic function is

$$ (n+1)({}^{n}C_{\mathrm{n}}) + (n)({}^{n}C_{\mathrm{n\text{-}1}}) + \ldots + 2({}^{n}C_{1}) = n2^{n-1} + 2^{n} - 1. \tag{7.17} $$

For CRS based architecture, the total number of CRS in LEVEL1 is given by

$$(n)({}^{n}C_{\text{n}}) + (n-1)({}^{n}C_{\text{n-1}}) + \ldots + ({}^{n}C_{1}) = n2^{n-1}. \tag{7.18}$$

LEVEL2 consists of the priority circuit. The maximum number of NOR blocks after the priority block in LEVEL3 is equal to 4 if number of outputs is one for fan-in restriction of 3 for stateful-NOR logic. The maximum number of memristors in these NOR blocks will be 12 for memristive architecture and for CRS based architecture, maximum number of CRSs will be 10.

The memristive/CRS crossbar is fabricated above CMOS layer using CMOS Back End of Line (BEOL) process. Hence the area comparison of proposed reconfigurable architecture with conventional LUT based FPGA is done with reference to transistor count as it is deciding factor of area. The 3-input 1-output reconfigurable logic block architecture using memristive/CRS crossbar shows 1.8× improvement in the area with respect to 3-input 1-output LUT based CLB of conventional FPGA. If number of outputs are more than one, there will be further improvement in area because in case of conventional LUT based FPGA, more CLBs to be connected in parallel to produce more outputs and hence effective number of transistors increases in proportion to the number of outputs. In case of proposed architecture, components shown in LEVEL3, reconfiguration components and destination components gets added as number of outputs increases. For 3-input *n*-output function, the area improvement factor for proposed architecture with respect to conventional LUT based CLB architecture for FPGA as a function of *n* is shown in Figure 7.6.

The device count given in Table 7.7 and 7.8 is native to the specific CLB (and programmable interconnections around it) for LUT based commercial FPGAs while in case of proposed architecture, the common blocks (write circuit, read circuit, evaluate circuit and priority circuit) are not native to specific logic block and are assigned to it during reconfigurable implementation. Hence the number of common blocks to be present in integrated circuit depends on maximum number of parallel implementations of logic blocks at a given time and is dependent on applications for which the reconfigurable architecture is designed. Thus there will be further improvement in area for proposed architecture with respect to LUT based FPGAs. If logic function to be implemented has more than three inputs, then multiplexer based generalized architecture shown in Figure 6.15 will be used, which is conceptually similar to that used in FPGAs made up

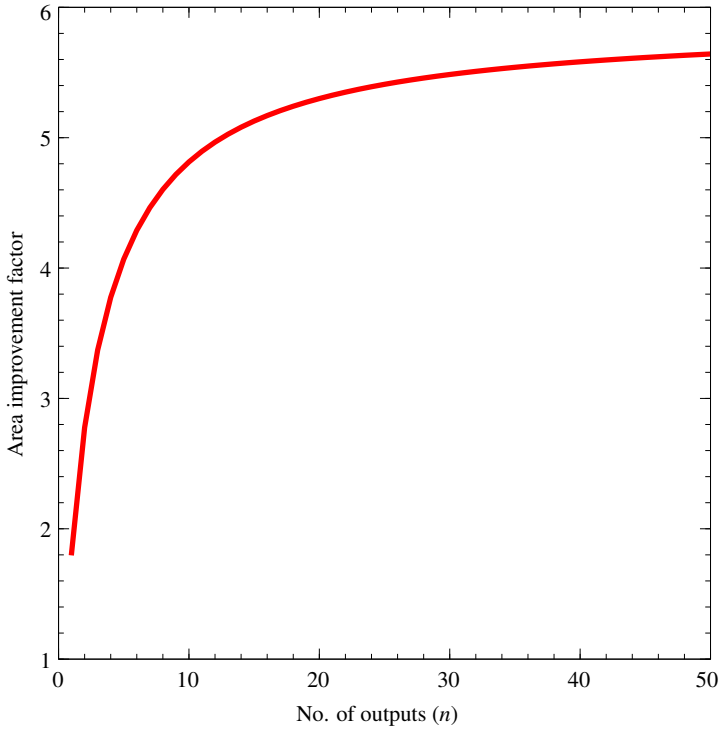


Figure 7.6. Area improvement factor (ratio of area for proposed reconfigurable 3-input logic block architecture to that of 3-input LUT based configuration logic block) for proposed reconfigurable 3-input logic block architecture against conventional LUT based 3-input configuration logic block of FPGA as a function of no. of outputs (n) of logic function to be implemented.

of LUT based 3-input CLBs. Hence similar area improvement will be there as discussed above for functions having more than 3-inputs.

7.5 Summary

The performance of proposed 3-input 1-output reconfigurable architecture is analyzed for implementation in memristive and CRS crossbar with respect to delay, power and area, and compared with LUT based CLB used in most commercial FPGAs. The subarray size in hierarchy of banks-mats-subarray organization of ReRAM is restricted to 128 $\times$ 128 for analysis purpose. The outcomes of analysis are summerized below.

- For already configured circuit function, the delay in LUT based CLB architecture is better than proposed architecture for small data set. However for new circuit function implementation, proposed architecture is better in terms of delay as reconfiguration (writing

configuration data to configuration memory) is sequential process in most commercial FPGAs. Also for larger data set size, the proposed architecture has shown 1.45× and 1.28× improvement in delay for implementation in memristive and CRS crossbar, respectively over implementation in LUT based CLB architecture.

- Implementation of proposed architecture in memristive crossbar has shown 2.68× less power dissipation while CRS crossbar implementation has shown 2.04× less power over implementation in LUT based CLB architecture.

- The area in resistive crossbar is decided mainly by supporting CMOS circuit as feature size of memristor/CRS is very small as compared to transistor. The crossbar is fabricated over CMOS layer using Back-End-Of-Line (BEOL) process. For 3-input 1-output function implementation, the proposed architecture has shown 1.8× area improvement on memristive/CRS crossbar over implementation in LUT based CLB architecture. For 3-input n-output logic function implementation this improvement increases to 5.6× for larger values of n. If number of inputs are more than 3, then similar improvement can be seen as same arrangement of multiplexers is used in proposed as well as LUT based architecture.

Chapter 8

Summary and Future Scope of Work

In this chapter, the conclusions drawn from research work carried out are summerized and scope for future work is presented.

8.1 Summary

ASICs are most efficient in terms of area, delay and power consumption but are becoming increasingly expensive with technological development and takes longest development time. The most economical and fast implementation of functions at circuit level is done with FPGAs. But FPGAs are inferior in terms of speed, area and power consumption as compared to ASICs. In order to increase the share of FPGAs in application implementation at circuit level, the performance gap between ASIC and FPGA needs to be improved.

- The main cause of inferiority in terms of area, delay, power consumption and cost of FPGAs over ASICs is extensive use of SRAM and programmable interconnections. Modifying the architecture of SRAM based FPGA has shown limited improvement in performance parameters and with technology scaling, it is becoming more and more difficult to sustain Moore's law because of leakage power and fabrication difficulty in CMOS technology. Hence new emerging devices need to be investigated as a replacement of SRAM and in developing low cost high performance reconfigurable architectures. In the work presented

in this thesis, fourth passive element called memristor is used in developing reconfiguration architecture.

- Memristor can be used as a logic element and as a nonvolatile memory cell where information is stored in the form of resistance. Universal gates like stateful NOR and stateful NAND can be implemented using memristors. In this work stateful NOR gate is used to design reconfigurable architecture.

- In order to carry out simulation of systems made up of memristors, its model is necessary so that it can be used in electronic design automation tools for design, verification and testing. Till date, universally acceptable and mature model of memristor is not available. Out of the models available in the literature, VTEAM model is used in the work carried out in this thesis as it is generic, accurate, simple, flexible and computationally efficient.

- Memristors are passive elements and need supporting CMOS circuit to implement any function. They are fabricated in the form of crossbar array over CMOS layer using nanoimprint lithography. For the realization of stateful NOR gate on memristive crossbar, write, read and evaluate (NOR) operations need to be implemented without disturbing the contents of memristors other than involved in logic operation.

- Thorough analysis is carried out of write, read and evaluate (NOR) operation implementation on memristive crossbar in order investigate the effect on memristors in a crosssbar that are not part of a logic and to find limitation on the size of a crossbar due to sneak path problem. The outcomes of this analysis are:

  - Floating write scheme is not suitable to write data in a memristor of memristive crossbar as it alters the contents of other memristors even for small array size.

  - The most useful write scheme for memristive crossbar is 1/3 write scheme as it does not change the contents of other memristors while writing data to selected memristor/s.

  - While reading the state of a memristor in a crossbar, the sneak path problem limits the size of array to distinguish between LRS and HRS state of memristor. In this work, array size is limited to 10×10. This limit on size can be relaxed by forcing memristors on bit line of memristor to be read to HRSs.

  - While performing evaluate (NOR) operation, there is limit on minimum size of array so that contents of other memristors in a crossbar not involved in logic, are not disturbed. In this work, limit on crossbar size is 50×50. At the same time, in order to implement stateful NOR operation correctly, there is limit on maximum size of array (10×10 in present work). Thus, either the size of crossbar should be kept small in order to implement NOR logic correctly, neglecting changes in the contents of memristors (memristors that are not part of logic, and present on bit and word lines of memristors selected for logic, are only disturbed) or by forcing all bit line memristors to HRSs like in read operation.
  - The size of crossbar can be logically restricted to smaller size by using specialized crossbar architectures such as CMOL and FPNI, and stateful NOR gate can be implemented on them.

- CRS crossbar arrays are free from sneak path roblem as they store information in terms HRS-LRS (logic '0') and LRS-HRS (logic '1') states presenting high equivalent resistance in both cases.

- The stateful NOR gate using CRSs is proposed and has following advantages over memristor based gate.

  - Stateful NOR gate using CRSs requires single voltage source while memristor based gate requires two voltage sources for its operation.
  - N-input stateful NOR gate using CRS requires N CRSs as one of the input CRSs is also destination. N-input stateful NOR gate using memristors requires (N+1) memristors as destination memristor is different from input memristors.
  - Destination memristor needs to be initialized to HRS before performing NOR operation in memristor based stateful NOR. This step is not required in CRS based stateful NOR as destination CRS is one of the input CRSs.

- Again, in order to implement stateful NOR gate on CRS crossbar, write, evaluate and read operations need to be performed. The outcomes of analysis of these operations on CRS crossbar are given below.

- For write operation, floating write scheme is not useful as it alters the contents of other CRSs in a crossbar. 1/3 write scheme is the best method as it maintains the integrity of other CRSs. Even though the configuration row based write scheme is slightly better than 1/3 write scheme in terms of power dissipation, but at the cost of extra area for configuration row and complex circuit in order to write LRS-LRS state to configuration row CRSs.
- While performing stateful NOR operation on CRS crossbar-
  - There is maximum limit on the size of a crossbar so that other CRSs on bit and word lines of input CRSs (excluding destination CRS) either do not change their states (if they are in LRS-HRS states) or can be restored back using self-resetting read scheme (if they are in HRS-LRS states). If size limit is exceeded, they will not change states at all. In this work this limit is 7000×7000.
  - The contents of CRSs that are not on input and destination lines are not destroyed for any array size.
  - The contents of CRSs on word line of destination CRS (excluding it) will be destroyed and can not be recovered back if they are in HRS-LRS states. To avoid this, such CRSs in HRS-LRS states should be converted to LRS-LRS states before operation and should be restored back after operation using self-resetting read scheme. Other option is to reserve some word lines as destination word lines and neglect their data in overall system operation.
- Read operation is destructive in nature as one of the states (out of HRS-LRS and LRS-HRS) is converted into LRS-LRS during read. Analysis of read operation on CRS crossbar shows that the state of CRS (HRS-LRS and LRS-HRS) can be safely read without error even for very large array size. Self-resetting read scheme is better choice for read operation as it uses the readout voltage (indicating state of CRS) as control signal (after delaying it) to reinstate the original state of CRS.

- The stateful NOR logic based reconfigurable 3-input pipelined logic block architecture has been proposed which can be implemented on memristive/CRS crossbar. It has three stages, stage one and three are made up of stateful NOR blocks while stage two is priority block. The automation algorithm for the same is presented. The input to algorithm is in the form of

truth table of logic function to be implemented, and it gives output in the form of netlist of components in the architecture. If number of inputs are more than three for a function, the proposed logic block architecture can be multiplexed to implement the function. Automation algorithm for this multiplexing is also presented.

- The proposed 3-input reconfigurable logic block architecture implemented on memristive/CRS crossbar is compared with LUT based 3-input CLB used in conventional FPGA in terms of delay, power and area. The crossbar array size is restricted to 128×128 in this work. The results are summerized below.
  - For already configured logic function, implementation in LUT based CLB is better for small data set in terms of delay, but if the function is to be newly reconfigured, then implementation using proposed architecture gives better results in terms of delay as configuration data is written sequentially to configuration memory in most commercial FPGAs. Also if data set to be operated on with logic function is large, then memristive crossbar implementation shows 1.45 times and CRS crossbar shows 1.28 times delay improvement over implementation in LUT based CLB architecture.
  - With respect to power dissipation, proposed architecture implemented on memristive crossbar has shown 2.68 times and on CRS crossbar 2.04 times improvement over implementation in LUT based CLB architecture.
  - Area of memristive/CRS crossbar is primarily decided by CMOS circuits used to implement functions. The analysis of proposed architecture for implementation of 3-input 1-output logic function has shown 1.8 times improvement in area over implementation in LUT based 3-input CLB architecture. The improvement factor increases to 5.6 times for implementation of 3-input n-output logic function for large values of n. If number of inputs are more than 3, then similar trends are seen in the area improvement for proposed architecture over LUT based CLB architecture.

8.2 Scope for future work

Following are some of the topics which can be explored further in the context of work carried out in this thesis :

- New methods/materials need to be investigated so that while performing certain operations in one section of crossbar, the data in other section is not disturbed, for any array size.
- The switching of signals between memristive/CRS crossbar and CMOS layer needs to be minimized in order to improve the performance of circuits implemented on memristive/CRS crossbar.
- The reading of results and writing back it on crossbar is redundant operation but presently it can not be eliminated. If it is eliminated, then results of one NOR operation can be used directly in next consecutive NOR operations (i.e. removing the fanout restrictions on stateful NOR) and there will be improvement in performance parameters. New methods and techniques need to be investigated.
- If fanout restriction on statful logic is removed, logic functions can be implemented with NAND/NOR networks, with minimum number of read/write steps.
- Electronics Automation Design (EDA) tool need to be developed for implementation of hybrid circuit made up of memristors and CMOS. Universally acceptable model based on physical mechanism of switching in memristor needs to be available for this purpose.

Appendix A

Memristor Models and Window Functions

Memristor model and window functions used in simulations are described in Chapter 3. In this Appendix, remaining models and window functions are described.

A.1 Memristor Models

A.1.1 Linear Ion Drift Model

According to the linear ion drift model proposed by Strukov et al. [58], if external bias voltage $v(t)$ is applied across memristor shown in Figure 3.2, the boundary between undoped and doped region will move due to drift of dopants. For ohmic electronic conduction and linear ionic drift in uniform field with average ion mobility μ_v, the ohm's law gives

$$v(t) = \left(R_{\mathrm{LRS}} \frac{w(t)}{D} + R_{\mathrm{HRS}} \left(1 - \frac{w(t)}{D} \right) \right) i(t), \tag{A.1}$$

and the state variable w is defined by

$$\frac{dw(t)}{dt} = \mu_v \frac{R_{\mathrm{LRS}}}{D} i(t). \tag{A.2}$$

This model is called a linear drift model, as the drift of the state variable (w) is linearly proportional to the current (i). Integrating (A.2) on both sides, we obtain

$$\begin{aligned} w(t) &= \mu_v \frac{R_{LRS}}{D} \int_{-\infty}^{t} i(\tau) d\tau \\ &= \mu_v \frac{R_{LRS}}{D} \int_{0}^{t} i(\tau) d\tau + \mu_v \frac{R_{LRS}}{D} \int_{-\infty}^{0} i(\tau) d\tau \\ &= \mu_v \frac{R_{LRS}}{D} \int_{0}^{t} i(\tau) d\tau + w_0. \end{aligned} \tag{A.3}$$

From (A.1), the memristance $M(t)$ is given by

$$M(t) = R_{LRS} \frac{w(t)}{D} + R_{HRS} \left(1 - \frac{w(t)}{D}\right). \tag{A.4}$$

Substituting (A.3) into (A.4), we get

$$M(t) = R_{HRS} + (R_{LRS} - R_{HRS}) \frac{w_0}{D} + (R_{LRS} - R_{HRS}) \frac{\mu_v R_{LRS}}{D^2} \int_{0}^{t} i(\tau) d\tau. \tag{A.5}$$

Using $M = \dfrac{d\phi}{dq}$, we integrate (A.5) with respect to q, we get

$$\phi(t) = \left[R_{HRS} + (R_{LRS} - R_{HRS}) \frac{w_0}{D}\right] q(t) + (R_{LRS} - R_{HRS}) \frac{\mu_v R_{LRS}}{2D^2} q^2(t). \tag{A.6}$$

From (A.6), the memristance is given by

$$M = R_{HRS} + (R_{LRS} - R_{HRS}) \frac{w_0}{D} + (R_{LRS} - R_{HRS}) \frac{\mu_v R_{LRS}}{D^2} q(t). \tag{A.7}$$

From (A.7), we see that the memristance is a linear function of the charge $q(t)$.

The assumption of linear ion drift in the above model is not correct in practical memristors, especially when state variable w approaches either $w = 0$ or $w = D$. The motion of dopants becomes highly nonlinear and this effect should be included in the model.

A.1.2 Nonlinear Ion Drift Model

To circumvent the problem of observed nonlinearity at $w = 0$ or $w = D$, the nonlinear ion drift model is given by

$$i(t) = w^n(t)\beta \sinh(\alpha v(t)) + \chi[\exp(\gamma v(t)) - 1], \tag{A.8}$$

and the derivative of state variable is given by

$$\frac{dw(t)}{dt} = av^m(t)f(w), \tag{A.9}$$

where $f(w)$ is window function. Different window functions are described later.

Here memristor is modeled as voltage controlled device and has nonlinear dependency between voltage and state derivative. Also asymmetric switching is also taken into account. $\alpha, \beta, \gamma, \chi, a$ and m are experimental fitting parameters, and the parameter n determines the effect of state variable on current. State variable w is normalized with respect to D within interval [0, 1]. During LRS state, the I-V curve in the model follows a tunneling process (sinh part) while during HRS state, it behaves like PN junction (exp part). Thus this model show asymmetric switching behavior. Low voltage can be used for read operation as switching takes very long time and hence device is stable. Higher voltages can be used for fast write operation. Another form of model is given by

$$\frac{dw(t)}{dt} = \mu_{\mathrm{v}} \frac{R_{\mathrm{LRS}}}{D} i(t) F_{\mathrm{p}}(w), \tag{A.10}$$

where $F_{\mathrm{p}}(w)$ is called window function and takes care of all nonlinearities observed in memristors. Parameter p is a positive integer. As p increases, the model tends to be more linear.

A.1.3 Simmons Tunnel Barrier Model

Simmons tunnel barrier model [154] is considered to be the most accurate model as it is based on experimental data. Memristor is modeled as a resistor in series with an electron tunnel barrier. Nonlinear and asymmetric switching behavior is assumed due to an exponential dependence of the movement of the ionized dopants. The strength of the Simmons tunnel barrier model stands in its experiment-based development and in its ability to describe some physical mechanism at the

origin of memristor dynamics. It explains the mechanisms at the origin of the complex dynamics observed in the TiO_2-based memristor by means of the Simmons tunnel barrier model [155]. The width of Simmons tunnel barrier is the state variable x. This model is given by

$$\begin{aligned} i(t) = & A(x,v_g)\phi_1(v_g,x) \times \exp(-B(v_g,x)\cdot \phi_1^{0.5}(v_g,x)) \\ & -A(x,v_g)(\phi_1(v_g,x)+e|v_g|) \times \exp(B(v_g,x)(\phi_1(v_g,x)+ev_g)^{0.5}), \end{aligned} \tag{A.11}$$

and

$$v_g = v - i(t)R_s. \tag{A.12}$$

The time derivative of state variable is given by the following equations. In the case of HRS (OFF) switching ($i > 0$)

$$\frac{dx(t)}{dt} = f_{HRS} \sinh\left(\frac{|i|}{i_{HRS}}\right) \exp\left[-\exp\left(\frac{x-a_{HRS}}{w_c} - \frac{|i|}{b}\right) - \frac{x}{w_c}\right], \tag{A.13}$$

with the fitting parameters $f_{HRS} = 3.5 \pm 1\ \mu m/s$, $i_{HRS} = 115 \pm 4\ \mu A$, $a_{HRS} = 1.2 \pm 0.02\ nm$, $b = 500 \pm 70\ \mu A$ and $w_c = 107 \pm 4\ pm$, whereas in the case of LRS (ON) switching ($i < 0$)

$$\frac{dx(t)}{dt} = f_{LRS} \sinh\left(\frac{|i|}{i_{LRS}}\right) \exp\left[-\exp\left(\frac{a_{LRS}-x}{w_c} - \frac{|i|}{b}\right) - \frac{x}{w_c}\right], \tag{A.14}$$

with the fitting parameters $f_{LRS} = 40 \pm 10\ \mu m/s$, $i_{LRS} = 8.9 \pm 0.3\ \mu A$, $a_{LRS} = 1.8 \pm 0.01\ nm$, $b = 500 \pm 90\ \mu A$ and $w_c = 107 \pm 3\ pm$.

The current i through the device has been modeled after that of a tunneling junction current [155] and its functional form is given by:

$$i = \frac{j_0 A}{\Delta x^2}\left\{\phi_I e^{-B\sqrt{\phi_I}} - \left(\phi_I + e|v_g|\right) e^{-B\sqrt{\phi_I + e|v_g|}}\right\}, \tag{A.15}$$

where

$$j_0 = \frac{e}{2\pi h}, x_1 = \frac{1.2\lambda x}{\phi_0}, \Delta x = x_2 - x_1, \tag{A.16}$$

$$\phi_I = \phi_0 - e|v_g|\left(\frac{x_1+x_2}{x}\right) - \left(\frac{1.15\lambda x}{\Delta x}\right)\ln\left(\frac{x_2\,(x-x_1)}{x_1\,(x-x_2)}\right), \tag{A.17}$$

$$B = \frac{4\pi\Delta x\sqrt{2m}}{h}, \tag{A.18}$$

$$x_2 = x_1 + x\left(1 - \frac{9.2\lambda}{\left(3\phi_0 + 4\lambda - 2e|v_g|\right)}\right), \tag{A.19}$$

$$\lambda = \frac{e^2 \ln(2)}{8\pi\kappa\varepsilon_0 w}, \tag{A.20}$$

where A is the channel area of the memristor, e is the electron charge, v_g is the voltage across the tunnel barrier, m is the mass of the electron, h is Planck's constant, κ is the dielectric constant, and ϕ_0 is the barrier height in electron volts [154]. $f_{\text{LRS}}, f_{\text{HRS}}, a_{\text{LRS}}, a_{\text{HRS}}, i_{\text{LRS}}, i_{\text{HRS}}$ and b are fitting parameters. The parameters f_{LRS} and f_{HRS} affect the magnitude of rate of change of state variable x and $f_{\text{LRS}} \gg f_{\text{HRS}}$. The parameters i_{LRS} and i_{HRS} effectively act like current thresholds. The parameters a_{HRS} and a_{LRS} put upper and lower bound on x, respectively and hence window function is not required in this model.

The model is most accurate but having some limitations : (1) The model is complicated, (2) In this model, voltage and current relationships are not explicit, (3) As it describes specific type of memristor, it is not generic model.

A.1.4 Boundary Condition Memristor (BCM) Model

As analytically demonstrated in [156], the BCM model can capture single-valued and multi-valued memductance-flux characteristics under sign-varying control voltage source offering the opportunity to tune the boundary behavior and the non-volatility degree according to the dynamics under modeling. In this model the input is assumed in voltage form, i.e. $u = v$, and the window function is expressed as

$$f(w,v) = \begin{cases} b, & \text{if } C_1 \text{ or } C_2 \text{ holds,} \\ 0, & \text{if } C_3 \text{ or } C_4 \text{ holds,} \\ a, & \text{if } C_5 \text{ holds,} \end{cases} \tag{A.21}$$

where parameters $a \in \mathbb{R}_{0,+}$ and $b \in \mathbb{R}_{+}(b > a)$ describe the degree of non-volatility of the nano-structure and tunable conditions $C_n(n = 1,2,3,4,5)$ are mathematically described by

$$C_1 = \{(w \in (0,D)\ and\ ((v > v_{t0})\, or\, (v < -v_{t1}))).\} \tag{A.22}$$

$$C_2 = \{(w = 0 \; and \; v > v_{\text{th0}}) \, or \, (w = D \; and \; v < -v_{\text{LRS}})\}. \tag{A.23}$$

$$C_3 = \{w = 0 \; and \; v \leq v_{\text{th0}}\}. \tag{A.24}$$

$$C_4 = \{w = D \; and \; v \geq -v_{\text{LRS}}\}. \tag{A.25}$$

$$C_5 = \{(w = \bar{w} \in (0, D) \; and \; ((v \leq v_{\text{t0}}) \, and \, (v \geq -v_{\text{t1}})))\}. \tag{A.26}$$

In conditions (A.23), (A.24), (A.25) voltage parameters $v_{\text{th0}} \in \mathbb{R}_{0,+}$ and $v_{\text{LRS}} \in \mathbb{R}_{0,+}$ represent the boundary threshold voltages (i.e. the threshold voltage of the input needs to exceed, after a sign reversal, for the state to be released from the lower and upper bound, respectively). Further voltage parameters $v_{\text{t0}} \in \mathbb{R}_{0,+}$ and $v_{\text{t1}} \in \mathbb{R}_{0,+}$ in (A.22) and (A.26) denote the programmability threshold voltages (i.e. the threshold voltage magnitude of a positive and negative input, respectively needs to be exceeded for the window function to exhibit a discontinuous transition from a smaller a to a larger b value).

A.1.5 Other Models

Li and Hu [157] presented a compact model of the spintronic memristor based on the magnetic-domain-wall motion mechanism for circuit design by taking into account the variations of material parameters and fabrication process. Abdalla and Pickett [158] presented a SPICE model for the titanium dioxide memristor device from its modeling equations as described in [154]. SPICE models based on physical models given by Strukov et al.[58] and Yang et al. [69] are given in [34, 159–163]. Its emulators and macro models have been proposed in [164, 165].

A.2 Window Functions

Every memristor model should have bound on region of operation. For example, the linear ion drift model should work within bound [0, D]. Thus in order to bound the state variable in certain range and to include the nonlinearity and asymmetry observed in practical memristors at bounds, window function is used. The state variable derivative is multiplied by window function to limit

the working interval and to add nonlinearity and asymmetry. Different window functions used in memristor models are described in this section.

A.2.1 Joglekar's Window

Joglekar's window function [166] is given by

$$F_{\mathrm{p}}(w) = 1 - \left(\frac{2w}{D} - 1\right)^{2p} \tag{A.27}$$

This window function for various values of p is plotted in Figure A.1, where p is a positive integer controlling the rate of decrease of the state variable as it approaches either bound.

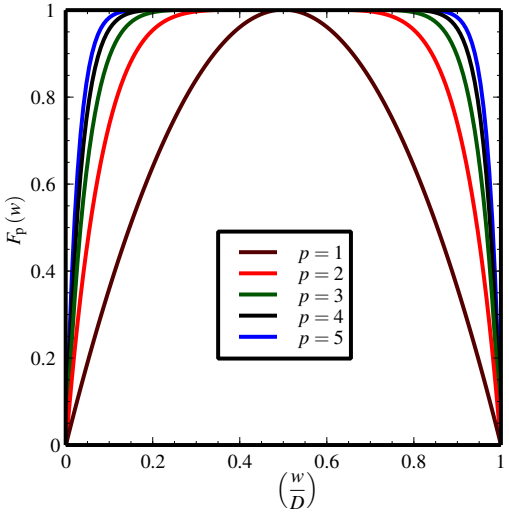


Figure A.1. The plot of normalized (with respect to width D of memristor) Joglekar's window function given by (A.27) for different values of parameter p. Parameter p is a positive integer which controls the rate of decrease of state variable w as it approaches zero or D.

The first limitation of Joglekar's window is that when state variable w achieves the extreme values ($w = 0$ or $w = D$), then from (A.27), value of window function $F_{\mathrm{p}}(w) = 0$. Thus when its terminal state of memristor in the model is either LRS or HRS, it is not possible to change its state thereafter even with application of external stimulus. It will held its state forever which is contradictory to practical memristive devices. Second limitation lies in the fact that it models the memristor as a component which exactly remembers the entire charge which is passing through (given by (A.5)). The memory effect is lost at the boundaries.

A.2.2 Biolek's Window

The stuck at boundary problem of Joglekar's window explained in previous section and discrepancy between the behavior of the model and the requirements for the operation of a real circuit element is resolved by designing a modified window function given by Biolek [161]. It models the fact that the speeds of approaching and leaving the boundary of thin film limits are different. Biolek introduced window function which is dependent on sign of input current (current controlled memristor). This window function is given by

$$F_{\mathrm{p}}(w,i) = 1 - \left[\frac{w}{D} - stp(-i)\right]^{2p}, \tag{A.28}$$

where p is a positive integer controlling the rate of decrease of the state variable from each bound to the other one. The $stp(i)$ function is given by

$$stp(i) = \begin{cases} 1, & \text{if } i \geq 0, \\ 0, & \text{if } i < 0. \end{cases} \tag{A.29}$$

The Biolek's window function is drawn in Figure A.2 for $p = 2$.

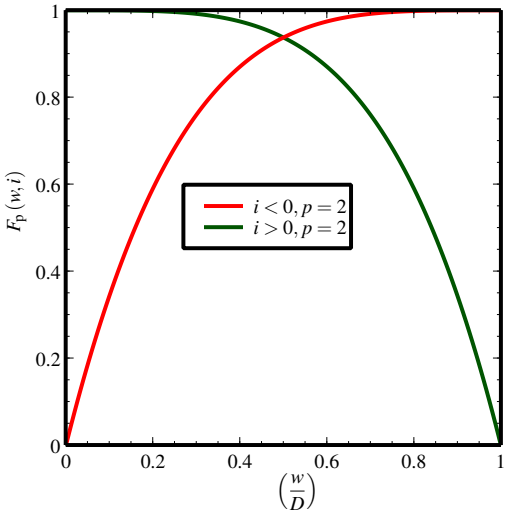


Figure A.2. The plot of normalized (with respect to width D of memristor) Biolek's window function given by (A.28) for parameter $p = 2$ and for positive/negative values of current.

The limitation of Biolek's window is the continuity condition at the boundaries. Also, Biolek's window function is a multivalued function which make it difficult in the analysis of the memristor-based circuits[167].

A.2.3 Prodomakis' Window

Prodomakis' [168] window function is independent of i and also gives maximum value of function other than unity. This is given as

$$F_{\mathrm{p}}(w) = j\left\{1 - \left[\left(\frac{w}{D} - 0.5\right)^2 + 0.75\right]^p\right\}, \tag{A.30}$$

where p and j are positive real parameters which determine both the rate of decrease of the window function as the state variable approaches any of its two bounds and the maximum value of the window function itself. Figure A.3 gives the plot for Prodomakis window for different values of parameter j with $p = 30$ while Figure A.4 gives plot for Prodomakis window for different values of parameter p with $j = 0.2$.

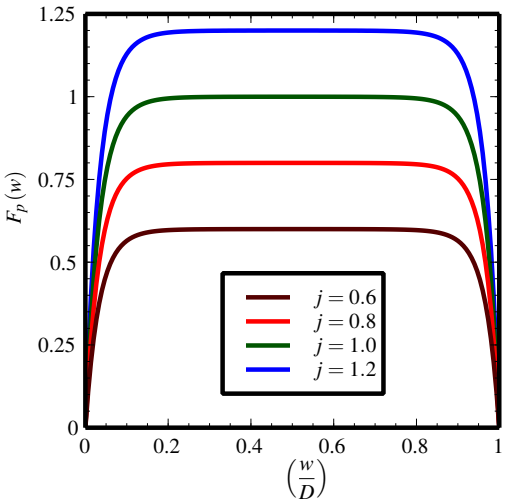


Figure A.3. The plot of normalized (with respect to width D of memristor) Prodomakis' window function given by (A.30) for parameter $p = 30$ and for different values of parameter j.

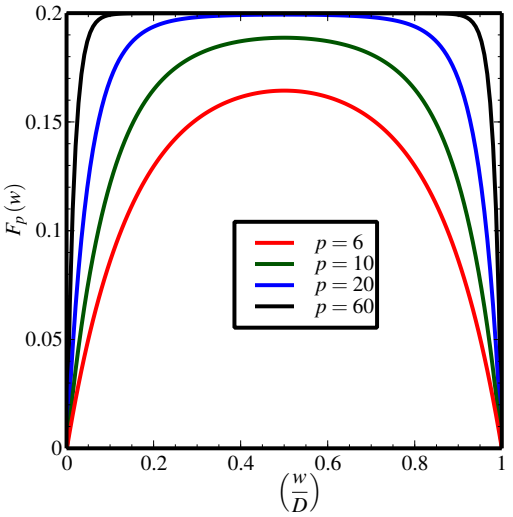


Figure A.4. The plot of normalized (with respect to width D of memristor) Prodomakis' window function given by (A.30) for parameter $j = 0.2$ and for different values of parameter p.

Appendix B

Imply Logic Analysis

In order to find the value of voltage V_{SET}, V_{COND} and R_G in Figure 3.6 to implement imply operation as shown in Table 3.3, analysis is carried out. Let R_1 and R_2 be resistance of memristors M_1 and M_2, respectively. Voltage across R_G is given by

$$V_{R_G} = \frac{V_{SET}(R_1||R_G)}{(R_1||R_G)+R_2} + \frac{V_{COND}(R_2||R_G)}{(R_2||R_G)+R_1}. \tag{B.1}$$

Voltage across R_1 and R_2 is given by

$$V_{R_1} = V_{COND} - V_{R_G}, \tag{B.2}$$

and

$$V_{R_2} = V_{SET} - V_{R_G}. \tag{B.3}$$

Four combinations of input and output from Table 3.3 are analyzed below.

- **Case 1:** $R_1 = R_{HRS}, R_2 = R_{HRS}$, **final value of** $R_2 = R_{LRS}$ For this case, (B.1) can be written as

$$V_{R_G} = \frac{V_{SET}(R_{HRS}||R_G)}{(R_{HRS}||R_G)+R_{HRS}} + \frac{V_{COND}(R_{HRS}||R_G)}{(R_{HRS}||R_G)+R_{HRS}}. \tag{B.4}$$

To change the final value of R_2 from R_{HRS} to R_{LRS}, current through R_2 must cross I_{LRS}. Hence

$$\frac{V_{\text{R}_2}}{R_2} = \frac{V_{\text{SET}} - V_{\text{R}_\text{G}}}{R_2} > I_{\text{LRS}}. \tag{B.5}$$

If $R_{\text{LRS}} << R_{\text{G}} << R_{\text{HRS}}$ (necessary to satisfy all conditions) then (B.4) can be approximated as

$$V_{\text{R}_\text{G}} \approx 0, \tag{B.6}$$

and (B.5) can be written as

$$\frac{V_{\text{R}_2}}{R_2} = \frac{V_{\text{SET}}}{R_2} > I_{\text{LRS}}. \tag{B.7}$$

After this step, circuit will follow case 4.

- **Case 2:** $R_1 = R_{\text{HRS}}, R_2 = R_{\text{LRS}}$, **final value of** $R_2 = R_{\text{LRS}}$ For this case, (B.1) can be written as

$$V_{\text{R}_\text{G}} = \frac{V_{\text{SET}}(R_{\text{HRS}}||R_{\text{G}})}{(R_{\text{HRS}}||R_{\text{G}}) + R_{\text{LRS}}} + \frac{V_{\text{COND}}(R_{\text{LRS}}||R_{\text{G}})}{(R_{\text{LRS}}||R_{\text{G}}) + R_{\text{HRS}}}. \tag{B.8}$$

To maintain the final value of R_2 as R_{LRS}, either $I_{\text{HRS}} < I_{\text{R}_2} < I_{\text{LRS}}$ or I_{R_2} may cross I_{LRS}. Both conditions are satisfied by

$$\frac{V_{\text{R}_2}}{R_2} = \frac{V_{\text{SET}} - V_{\text{R}_\text{G}}}{R_2} > I_{\text{HRS}}. \tag{B.9}$$

If $R_{\text{LRS}} << R_{\text{G}} << R_{\text{HRS}}$, (B.8) can be approximated as

$$V_{\text{R}_\text{G}} \approx V_{\text{SET}}, \tag{B.10}$$

and (B.9) can be written as

$$I_{\text{HRS}} < 0. \tag{B.11}$$

- **Case 3:** $R_1 = R_{\text{LRS}}, R_2 = R_{\text{HRS}}$, **final value of** $R_2 = R_{\text{HRS}}$ For this case, (B.1) can be written as

$$V_{\text{R}_\text{G}} = \frac{V_{\text{SET}}(R_{\text{LRS}}||R_{\text{G}})}{(R_{\text{LRS}}||R_{\text{G}}) + R_{\text{HRS}}} + \frac{V_{\text{COND}}(R_{\text{HRS}}||R_{\text{G}})}{(R_{\text{HRS}}||R_{\text{G}}) + R_{\text{LRS}}}. \tag{B.12}$$

To maintain the final value of R_2 as R_{HRS}, either $I_{HRS} < I_{R_2} < I_{LRS}$ or I_{R_2} may fall below I_{HRS}. Both conditions are satisfied by

$$\frac{V_{R_2}}{R_2} = \frac{V_{SET} - V_{R_G}}{R_2} < I_{LRS}. \tag{B.13}$$

If $R_{LRS} << R_G << R_{HRS}$ (B.12) can be approximated as

$$V_{R_G} \approx V_{COND}, \tag{B.14}$$

and (B.13) can be written as

$$\frac{V_{R_2}}{R_2} = \frac{V_{SET} - V_{COND}}{R_2} < I_{LRS}. \tag{B.15}$$

- **Case 4:** $R_1 = R_{LRS}, R_2 = R_{LRS}$, **final value of** $R_2 = R_{LRS}$ For this case, (B.1) can be written as

$$V_{R_G} = \frac{V_{SET}(R_{LRS}||R_G)}{(R_{LRS}||R_G) + R_{LRS}} + \frac{V_{COND}(R_{LRS}||R_G)}{(R_{LRS}||R_G) + R_{LRS}}. \tag{B.16}$$

To maintain the final value of R_2 as R_{LRS}, $I_{HRS} < I_{R_2} < I_{LRS}$ or I_{R_2} may cross I_{LRS}. Both conditions are satisfied by

$$\frac{V_{R_2}}{R_2} = \frac{V_{SET} - V_{R_G}}{R_2} > I_{HRS}. \tag{B.17}$$

If $R_{LRS} << R_G$ (B.16) can be approximated as

$$V_{R_G} \approx 0.5(V_{SET} + V_{COND}), \tag{B.18}$$

and (B.17) can be written as

$$\frac{V_{R_2}}{R_2} = 0.5\left(\frac{V_{SET} - V_{COND}}{R_2}\right) < I_{LRS}. \tag{B.19}$$

Note that $V_{SET} - V_{COND}$ can not be negative as $V_{SET} > V_{COND}$.

Similar analysis is given in [81, 169] but in this analysis, state drift (due to often alternate read and write) is neglected and the number of memristors in each NOR block is limited to four (like 3 input LUT) so scaling of R_G is not required. Also resultant equations are in terms of R_1, R_2,

R_G, V_{SET} and V_{COND} which can be solved for getting R_G, V_{SET} and V_{COND}. (R_1, R_2 takes values R_{LRS} or R_{HRS}).

Appendix C

Specialized Memristive Crossbar Array

Due to the sneak path problem, the size of memristive crossbar has to be restricted in order to use it for logic function implementation and as a memory. The specialized architectures such as CMOL [46] and FPNI [54] can be used where wire segments of small length are arranged such that small segmented array structure is created in large architectural area. These hybrid circuits combine CMOS technology with memristors in crossbars, called CMOS-MOLecule (CMOL) [46] and Field Programmable Nanowire Interconnect (FPNI) [54]. They are FPGA-like architectures and combine the advantages of CMOS (high yield, high gain, versatile functionality) with the reconfigurability and scalability of nanoscale crossbars.

C.1 CMOL Architecture

The transistor-based configuration memory and associated routing circuits are removed from CMOS transistors layer and replaced them with crossbar network in a layer of metal interconnect above CMOS Layer. The total area of an FPGA has been decreased by a factor of 10 or more while simultaneously it has increased the clock frequency and decreased the power consumption of the chip [46, 54].

In CMOL architecture nanowire segment is restricted to length L. CMOS layer is interfaced with nanolayer through nanopins distributed all over the circuit area. One set of pins connects CMOS with lower metal nanowire while other with upper nanowire. Nanopins in a set are arranged in square array fashion with distance between adjacent pins in a set equal to $2\beta F_{\text{CMOS}}$ where F_{CMOS} is half pitch of CMOS subsystem and β is dimensionless factor greater than one and depends on the CMOS cell complexity.

The nanocrossbar layer is rotated by an angle α with respect to CMOS pin array and nanowires in a crossbar are delimited to length L so that one pin connects to exactly one nanowire. The distance between two parallel nanowires connected by adjacent nanopins is $2rF_{\text{nano}}$ where F_{nano} is nanowire half pitch and r is dimensionless parameter greater than one. This generic architecture is shown in Figure C.1. The angle of rotation (α) is given by,

$$\sin\alpha = \frac{2F_{\text{nano}}}{2\beta F_{\text{CMOS}}} = \frac{F_{\text{nano}}}{\beta F_{\text{CMOS}}}, \tag{C.1}$$

and

$$\cos\alpha = \frac{2rF_{\text{nano}}}{2\beta F_{\text{CMOS}}} = \frac{rF_{\text{nano}}}{\beta F_{\text{CMOS}}}. \tag{C.2}$$

Hence

$$\alpha = tan^{-1}\left(\frac{1}{r}\right). \tag{C.3}$$

From Figure C.1,

$$\sin\alpha = \frac{2\beta F_{\text{CMOS}}}{L} = \frac{F_{\text{nano}}}{\beta F_{\text{CMOS}}}. \tag{C.4}$$

Therefore,

$$L = \frac{2\left(\beta F_{\text{CMOS}}\right)^2}{F_{\text{nano}}}. \tag{C.5}$$

Also,

$$\left(2\beta F_{\text{CMOS}}\right)^2 = \left(2rF_{\text{nano}}\right)^2 + \left(2F_{\text{nano}}\right)^2 \tag{C.6}$$

$$\left(\beta F_{\text{CMOS}}\right)^2 = \left(1+r^2\right)F_{\text{nano}}^2. \tag{C.7}$$

Thus,

$$L = 2\left(1+r^2\right)F_{\text{nano}}. \tag{C.8}$$

The number of memristors N on nanowire segment of length L will be,

$$N = \frac{L}{2F_{\text{nano}}} - 1 = \frac{2\left(1+r^2\right)F_{\text{nano}}}{2F_{\text{nano}}} - 1 = r^2. \tag{C.9}$$

From Figure C.1 and (C.3),

$$\tan\alpha = \frac{2\beta F_{\text{CMOS}}}{K} = \frac{1}{r}. \tag{C.10}$$

Hence,

$$K = 2\beta r F_{\text{CMOS}} \tag{C.11}$$

Therefore total number of CMOS cells (C) spanned by horizontal (vertical) nanowire of segment of length L in horizontal (vertical) direction is

$$C = \frac{K}{2\beta F_{\text{CMOS}}} = \frac{2\beta r F_{\text{CMOS}}}{2\beta F_{\text{CMOS}}} = r. \tag{C.12}$$

One pin in a cell can connect to other $(N-1)$ CMOS cells through pin - nanowire - memristor - nanowire - pin interface, where $N = r^2$ as in (C.9). Hence, through this pin, a cell is connected to cell area $(r^2-1)(2\beta F_{\text{CMOS}})^2$ around it. Remember each cell has 2 CMOS nanopins. Thus the array of size $r^2 \times r^2$ can be accessed for NOR implementation. The size of array can be restricted by controlling r which is function of angle of rotation α. The analysis of usage of CMOL architecture will be similar to memristive crossbar with $m = n = r^2$.

Hybrid CMOS/Memristor based CMOL architecture consists of nanocrossbar stacked over CMOS plane. The scheme for accessing single memristor in CMOL architecture is shown in Figure C.2. Two pairs of perpendicular CMOS lines are used to access a memristor and performing operations on it. For example, in Figure C.2 vertical metal wires V_{m} and V_{n} are used to access memristor at the crosspoint of two nanowires. To write data into memristor, horizontal lines H_r and H_p are used. One drives voltage $\pm V_{\text{W}}$ on one nanowire and the other drives $\mp V_{\text{W}}$ on other nanowire making effective voltage across memristor as $\pm 2V_{\text{W}}$. The voltage $\pm V_{\text{W}}$ is selected such that $|\pm V_{\text{W}}| < |V_{\text{th1, M}}|$ and $|\pm V_{\text{W}}| < |V_{\text{th2, M}}|$ but $|\pm 2V_{\text{W}}| > |V_{\text{th1, M}}|$ and $|\pm 2V_{\text{W}}| > |V_{\text{th2, M}}|$. Half selected memristor (out of two nanowires forming memristor at their crosspoint, one nanowire is driven while other not) will not be affected in this scheme but only fully selected will be written

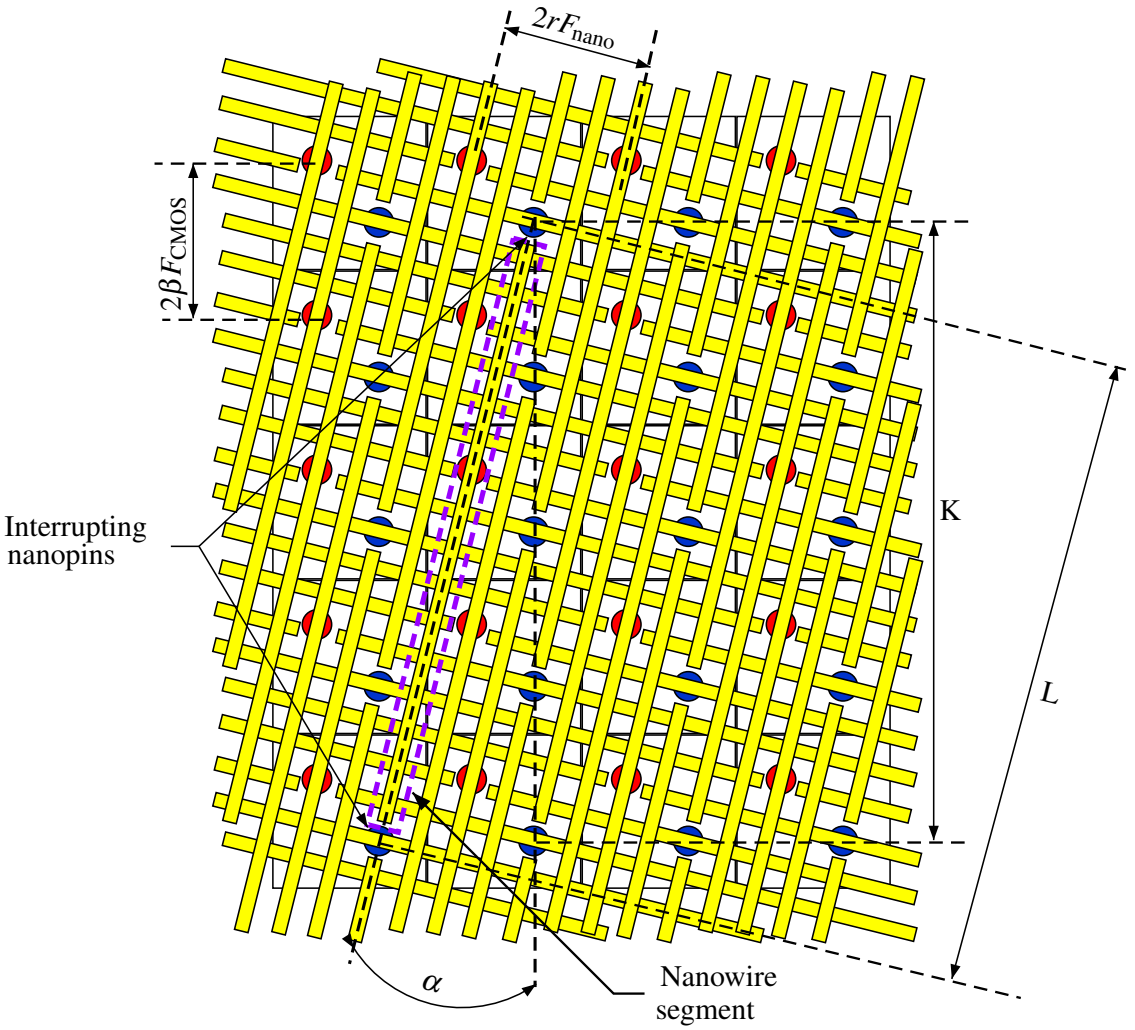


Figure C.1. Generic CMOL architecture top view. The nanocrossbar is rotated with an angle α with respect to CMOS nanopin array. Each nanopin connects CMOS cell with exactly one nanowire (red (blue) connects with vertical (horizontal) nanowire).

with state (R_{HRS} or R_{LRS}) depending on voltage polarity across memristor. In read operation, one nanowire is driven with read voltage V_R, and is sensed on the other nanowire to know the state of memristor.

The CMOS cell has inverter in addition to pass transistors to implement wired-NOR gate. The structure is shown in Figure C.3. The inverter can be disabled if required by cutting power supply of it. The implementation of wired-NOR gate is shown as an example to implement the function $C = \overline{A+B}$ on CMOL architecture in Figure C.4 and its equivalent circuit in Figure C.5.

In CMOL, nanowires are used for signal routing and performing wired-OR logic while other functions are being implemented in CMOS layer. As CMOL uses non-complementary wired-NOR logic, careful optimization is required of closed junction resistance, pass-transistor resistance

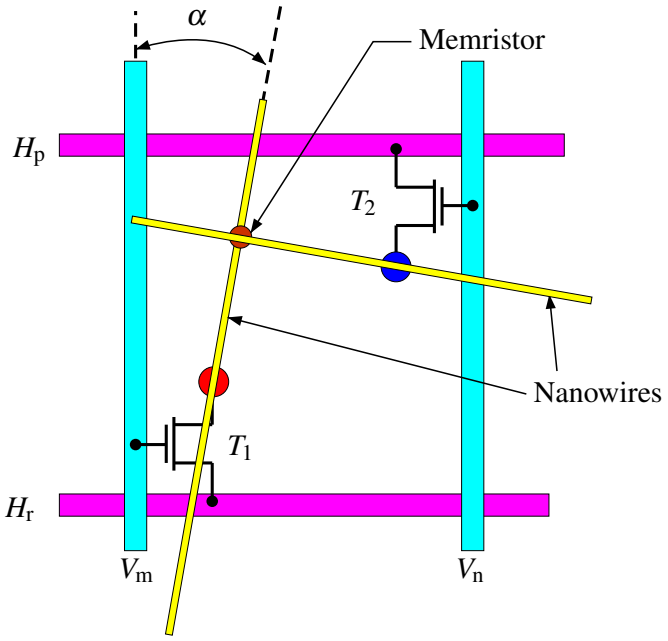


Figure C.2. Single memristor can be accessed through CMOS - nanowire - memristor - nanowire - CMOS interface. In this generic figure vertical metal wires are used to select nanowire and horizontal metal wires for driving/receiving data to/from selected nanowires.

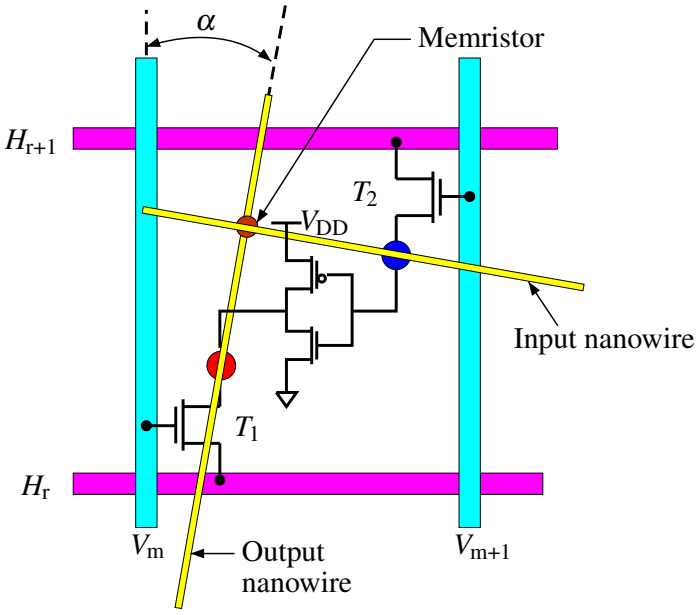


Figure C.3. CMOL cell consisting of inverter and pass transistors to implement wired-NOR gate. The inverter can be disabled in case not required by switching off the power supply.

and supply voltage. Also the configurable junctions need to be extremely nonlinear antifuses in order to implement the wired-OR function and may restrict fan-in because of device variability or insufficient nonlinearity. This in turn will reduce circuit density. Nonlinear devices are difficult to fabricate in crossbar. The fabrication of tapered nanopins with extremely small tip diameter and

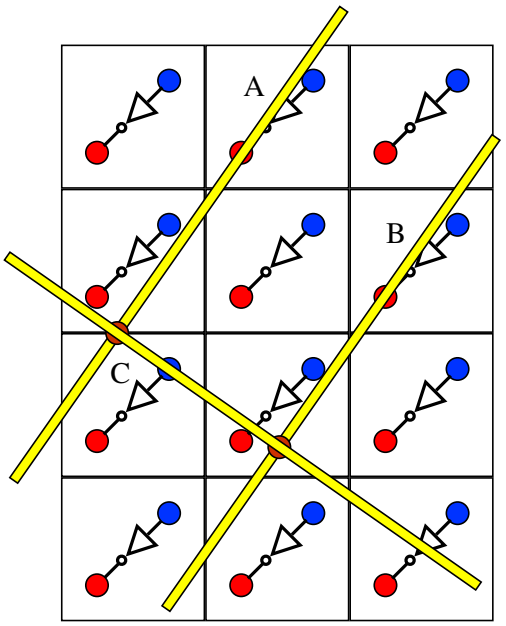


Figure C.4. Example implementation of wired-NOR gate on CMOL architecture, implementing $C = \overline{A+B}$. Each CMOS cell consists of inverter and pass transistors (not shown in this Figure) [46].

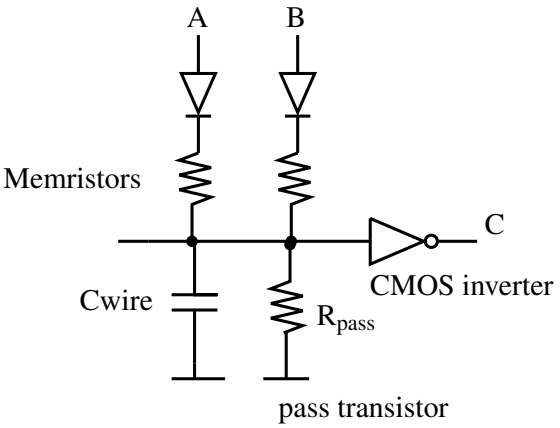


Figure C.5. The equivalent NOR gate for example implementation of wired-NOR in CMOL architecture shown in Figure C.4 [46].

their placement is difficult [54]. The modification to this architecture is FPNI architecture.

C.2 FPNI Architecture

FPNI architecture is similar to CMOL with following changes: In FPNI logic operations are done only in CMOS while signal routing is done only through nanowires and memristors. This significantly reduces static power dissipation. The FPNI routing network is buffer based (CMOL routing is inverter based) and it simplifies signal routing through available routing algorithms. The alignment of FPNI nanowire crossbar with CMOS pins require accuracy on CMOS scale. FPNI uses conventional CMOS process and voltages, and provides planar silicon surface for nanowires. In CMOL, nanopins of different height makes surface nonplanar and CMOL architecture works at very low voltage levels.

The top view of FPNI architecture is shown in Figure 2.8 and its side view is drawn in Figure 2.7. In order to simplify the nanocrossbar CMOS interface and to avoid the tapering of CMOS pins, the nanowires are flattened at the location of CMOS nanowire interface. The CMOS level resolution

is sufficient to fabricate the CMOS pins. However this arrangement makes the crossbar sparser with less circuit density. Figure C.6 shows the example implementation of simple NAND function $C = \overline{AB}$ on FPNI fabric. The signal routing is only done through nanocrossbar while logic function is executed in CMOS part.

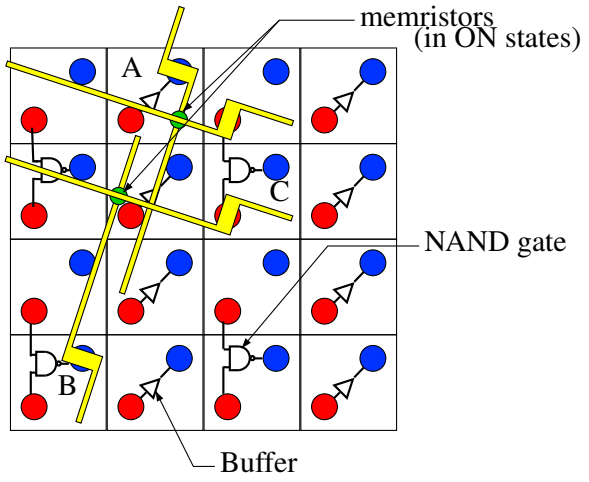


Figure C.6. Example implementation of NAND function ($C = \overline{AB}$) in FPNI fabric where the inputs are routed through nanowire and memristors while actual logic function (NAND in this case) is implemented in CMOS layer along with buffers [54].

Appendix D

CRS Logic Analysis

Basic 2-input NOR gate with CRS is shown in Figure 5.8. Consider the voltage applied at CRS Q (which is one of the inputs and destination CRS) is $V_Q = V_x$ and voltage applied at other input CRS P is $V_P = V_x - \Delta V$ for logic evaluation. Voltage at point C in Figure 5.8 is given by

$$V_C = \frac{(V_x - \Delta V)\left(R_G || \left(R_{QM_1} + R_{QM_2}\right)\right)}{\left(R_G || \left(R_{QM_1} + R_{QM_2}\right)\right) + \left(R_{PM_1} + R_{PM_2}\right)} + \frac{V_x \left(R_G || \left(R_{PM_1} + R_{PM_2}\right)\right)}{\left(R_G || \left(R_{PM_1} + R_{PM_2}\right)\right) + \left(R_{QM_1} + R_{QM_2}\right)} \quad \text{(D.1)}$$

All possible cases of states of memristors that are part of CRS based stateful NOR gate in Figure 5.8 are listed in Table 5.2. The resultant voltage across each memristor in stateful NOR gate for Case 1 is evaluated in Chapter 5 while for remaining cases, evaluation is done below.

- **Case 2 : $\mathbf{R_{PM_1} = R_{LRS}}$, $\mathbf{R_{PM_2} = R_{HRS}}$, $\mathbf{R_{QM_1} = R_{LRS}}$, $\mathbf{R_{QM_2} = R_{LRS}}$**

  For Case 2 where $R_{PM_1} = R_{LRS}$, $R_{PM_2} = R_{HRS}$, $R_{QM_1} = R_{LRS}$, $R_{QM_2} = R_{LRS}$, (D.1) becomes

  $$V_C = \frac{(V_x - \Delta V)\left(R_G || \left(R_{LRS} + R_{LRS}\right)\right)}{\left(R_G || \left(R_{LRS} + R_{LRS}\right)\right) + \left(R_{LRS} + R_{HRS}\right)} + \frac{V_x \left(R_G || \left(R_{LRS} + R_{HRS}\right)\right)}{\left(R_G || \left(R_{LRS} + R_{HRS}\right)\right) + \left(R_{LRS} + R_{LRS}\right)}, \quad \text{(D.2)}$$

  which can be approximated as,

  $$V_C \approx \frac{(V_x - \Delta V)\, 2R_{LRS}}{3R_{LRS} + R_{HRS}} + \frac{V_x R_G}{R_G + 2R_{LRS}} \approx V_x. \quad \text{(D.3)}$$

Therefore, voltage across memristor PM_1 is approximately given as,

$$V_{PM_1} = [(V_x - \Delta V) - V_x] \frac{R_{LRS}}{R_{LRS} + R_{HRS}} \approx \frac{-\Delta V R_{LRS}}{R_{HRS}} \approx 0. \tag{D.4}$$

In a similar manner, voltage across memristor PM_2 is

$$V_{PM_2} = [(V_x - \Delta V) - V_x] \frac{R_{HRS}}{R_{LRS} + R_{HRS}} \approx -\Delta V, \tag{D.5}$$

voltage across QM_1 is

$$V_{QM_1} = [V_x - V_x] \frac{R_{LRS}}{R_{LRS} + R_{LRS}} \approx 0, \tag{D.6}$$

and across QM_2 is

$$V_{QM_2} = [V_x - V_x] \frac{R_{LRS}}{R_{LRS} + R_{LRS}} \approx 0. \tag{D.7}$$

- **Case 3 : $\mathbf{R_{PM_1} = R_{HRS}, R_{PM_2} = R_{LRS}, R_{QM_1} = R_{LRS}, R_{QM_2} = R_{LRS}}$**

For Case 3 where $R_{PM_1} = R_{HRS}, R_{PM_2} = R_{LRS}, R_{QM_1} = R_{LRS}, R_{QM_2} = R_{LRS}$, (D.1) becomes

$$V_C = \frac{(V_x - \Delta V)\,(R_G || (R_{LRS} + R_{LRS}))}{(R_G || (R_{LRS} + R_{LRS})) + (R_{HRS} + R_{LRS})} + \frac{V_x\,(R_G || (R_{HRS} + R_{LRS}))}{(R_G || (R_{HRS} + R_{LRS})) + (R_{LRS} + R_{LRS})}, \tag{D.8}$$

which can be approximated as,

$$V_C \approx \frac{(V_x - \Delta V)\,2R_{LRS}}{3R_{LRS} + R_{HRS}} + \frac{V_x R_G}{R_G + 2R_{LRS}} \approx V_x. \tag{D.9}$$

Therefore, voltage across memristor PM_1 is approximately given as,

$$V_{PM_1} = [(V_x - \Delta V) - V_x] \frac{R_{HRS}}{R_{LRS} + R_{HRS}} \approx -\Delta V. \tag{D.10}$$

In a similar manner, voltage across memristor PM_2 is

$$V_{PM_2} = [(V_x - \Delta V) - V_x] \frac{R_{LRS}}{R_{LRS} + R_{HRS}} \approx \frac{-\Delta V R_{LRS}}{R_{HRS}} \approx 0, \tag{D.11}$$

voltage across QM_1 is

$$V_{QM_1} = [V_x - V_x] \frac{R_{LRS}}{R_{LRS} + R_{LRS}} \approx 0, \tag{D.12}$$

and across QM_2 is

$$V_{QM_2} = [V_x - V_x] \frac{R_{LRS}}{R_{LRS} + R_{LRS}} \approx 0. \tag{D.13}$$

- **Case 4 : $R_{PM_1} = R_{HRS}$, $R_{PM_2} = R_{HRS}$, $R_{QM_1} = R_{LRS}$, $R_{QM_2} = R_{LRS}$**

For Case 4 where $R_{PM_1} = R_{HRS}$, $R_{PM_2} = R_{HRS}$, $R_{QM_1} = R_{LRS}$, $R_{QM_2} = R_{LRS}$, (D.1) becomes

$$V_C = \frac{(V_x - \Delta V)\,(R_G || (R_{LRS} + R_{LRS}))}{(R_G || (R_{LRS} + R_{LRS})) + (R_{HRS} + R_{HRS})} + \frac{V_x\,(R_G || (R_{HRS} + R_{HRS}))}{(R_G || (R_{HRS} + R_{HRS})) + (R_{LRS} + R_{LRS})}, \tag{D.14}$$

which can be approximated as,

$$V_C \approx \frac{(V_x - \Delta V)\,2R_{LRS}}{2R_{LRS} + 2R_{HRS}} + \frac{V_x R_G}{R_G + 2R_{LRS}} \approx V_x. \tag{D.15}$$

Therefore, voltage across memristor PM_1 is approximately given as,

$$V_{PM_1} = [(V_x - \Delta V) - V_x] \frac{R_{HRS}}{R_{HRS} + R_{HRS}} \approx -\frac{\Delta V}{2}. \tag{D.16}$$

In a similar manner, voltage across memristor PM_2 is

$$V_{PM_2} = [(V_x - \Delta V) - V_x] \frac{R_{HRS}}{R_{HRS} + R_{HRS}} \approx -\frac{\Delta V}{2}, \tag{D.17}$$

voltage across QM_1 is

$$V_{QM_1} = [V_x - V_x] \frac{R_{LRS}}{R_{LRS} + R_{LRS}} \approx 0, \tag{D.18}$$

and across QM_2 is

$$V_{QM_2} = [V_x - V_x] \frac{R_{LRS}}{R_{LRS} + R_{LRS}} \approx 0. \tag{D.19}$$

- **Case 5 : $R_{PM_1} = R_{LRS}$, $R_{PM_2} = R_{LRS}$, $R_{QM_1} = R_{LRS}$, $R_{QM_2} = R_{HRS}$**

For Case 5 where $R_{PM_1} = R_{LRS}$, $R_{PM_2} = R_{LRS}$, $R_{QM_1} = R_{LRS}$, $R_{QM_2} = R_{HRS}$, (D.1) becomes

$$V_C = \frac{(V_x - \Delta V)\,(R_G || (R_{LRS} + R_{HRS}))}{(R_G || (R_{LRS} + R_{HRS})) + (R_{LRS} + R_{LRS})} + \frac{V_x\,(R_G || (R_{LRS} + R_{LRS}))}{(R_G || (R_{LRS} + R_{LRS})) + (R_{LRS} + R_{HRS})}, \tag{D.20}$$

which can be approximated as,

$$V_C \approx \frac{(V_x - \Delta V)\,R_G}{R_G + 2R_{LRS}} + \frac{V_x\,(2R_{LRS})}{3R_{LRS} + R_{HRS}} \approx V_x - \Delta V. \tag{D.21}$$

Therefore, voltage across memristor PM_1 is approximately given as,

$$V_{PM_1} = [(V_x - \Delta V) - (V_x - \Delta V)] \frac{R_{LRS}}{R_{LRS} + R_{LRS}} \approx 0. \tag{D.22}$$

In a similar manner, voltage across memristor PM_2 is

$$V_{PM_2} \approx 0, \tag{D.23}$$

voltage across QM_1 is

$$V_{QM_1} = [V_x - (V_x - \Delta V)] \frac{R_{LRS}}{R_{LRS} + R_{HRS}} \approx \frac{\Delta V R_{LRS}}{R_{HRS}} \approx 0, \tag{D.24}$$

and across QM_2 is

$$V_{QM_2} = [V_x - (V_x - \Delta V)] \frac{R_{HRS}}{R_{LRS} + R_{HRS}} \approx \Delta V. \tag{D.25}$$

- **Case 6 : $R_{PM_1} = R_{LRS}$, $R_{PM_2} = R_{HRS}$, $R_{QM_1} = R_{LRS}$, $R_{QM_2} = R_{HRS}$**

For Case 6 where $R_{PM_1} = R_{LRS}$, $R_{PM_2} = R_{HRS}$, $R_{QM_1} = R_{LRS}$, $R_{QM_2} = R_{HRS}$, (D.1) becomes

$$V_C = \frac{(V_x - \Delta V)(R_G || (R_{LRS} + R_{HRS}))}{(R_G || (R_{LRS} + R_{HRS})) + (R_{LRS} + R_{HRS})} + \frac{V_x (R_G || (R_{LRS} + R_{HRS}))}{(R_G || (R_{LRS} + R_{HRS})) + (R_{LRS} + R_{HRS})}, \tag{D.26}$$

which can be approximated as,

$$V_C \approx \frac{(V_x - \Delta V) R_G}{R_G + R_{LRS} + R_{HRS}} + \frac{V_x R_G}{R_G + R_{LRS} + R_{HRS}} \approx \frac{(2V_x - \Delta V) R_G}{R_{HRS}} \approx 0. \tag{D.27}$$

Therefore, voltage across memristor PM_1 is approximately given as,

$$V_{PM_1} = \left[(V_x - \Delta V) - \frac{(2V_x - \Delta V) R_G}{R_{HRS}} \right] \frac{R_{LRS}}{R_{LRS} + R_{HRS}} \approx 0. \tag{D.28}$$

In a similar manner, voltage across memristor PM_2 is

$$V_{PM_2} = \left[(V_x - \Delta V) - \frac{(2V_x - \Delta V) R_G}{R_{HRS}} \right] \frac{R_{HRS}}{R_{LRS} + R_{HRS}} \approx V_x - \Delta V, \tag{D.29}$$

voltage across QM_1 is

$$V_{QM_1} = \left[V_x - \frac{(2V_x - \Delta V)\,R_G}{R_{HRS}}\right] \frac{R_{LRS}}{R_{LRS} + R_{HRS}} \approx 0, \tag{D.30}$$

and across QM_2 is

$$V_{QM_2} = \left[V_x - \frac{(2V_x - \Delta V)\,R_G}{R_{HRS}}\right] \frac{R_{HRS}}{R_{LRS} + R_{HRS}} \approx V_x. \tag{D.31}$$

- **Case 7 : $\mathbf{R_{PM_1} = R_{HRS}, R_{PM_2} = R_{LRS}, R_{QM_1} = R_{LRS}, R_{QM_2} = R_{HRS}}$**

For Case 7 where $R_{PM_1} = R_{HRS}, R_{PM_2} = R_{LRS}, R_{QM_1} = R_{LRS}, R_{QM_2} = R_{HRS}$, (D.1) becomes

$$V_C = \frac{(V_x - \Delta V)\,(R_G || (R_{LRS} + R_{HRS}))}{(R_G || (R_{LRS} + R_{HRS})) + (R_{HRS} + R_{LRS})} + \frac{V_x\,(R_G || (R_{HRS} + R_{LRS}))}{(R_G || (R_{HRS} + R_{LRS})) + (R_{LRS} + R_{HRS})}, \tag{D.32}$$

which can be approximated as,

$$V_C \approx \frac{(V_x - \Delta V)\,R_G}{R_G + R_{LRS} + R_{HRS}} + \frac{V_x R_G}{R_G + R_{LRS} + R_{HRS}} \approx \frac{(2V_x - \Delta V)\,R_G}{R_{HRS}} \approx 0. \tag{D.33}$$

Therefore, voltage across memristor PM_1 is approximately given as,

$$V_{PM_1} = \left[(V_x - \Delta V) - \frac{(2V_x - \Delta V)\,R_G}{R_{HRS}}\right] \frac{R_{HRS}}{R_{LRS} + R_{HRS}} \approx V_x - \Delta V. \tag{D.34}$$

In a similar manner, voltage across memristor PM_2 is

$$V_{PM_2} = \left[(V_x - \Delta V) - \frac{(2V_x - \Delta V)\,R_G}{R_{HRS}}\right] \frac{R_{LRS}}{R_{LRS} + R_{HRS}} \approx 0, \tag{D.35}$$

voltage across QM_1 is

$$V_{QM_1} = \left[V_x - \frac{(2V_x - \Delta V)\,R_G}{R_{HRS}}\right] \frac{R_{LRS}}{R_{LRS} + R_{HRS}} \approx 0, \tag{D.36}$$

and across QM_2 is

$$V_{QM_2} = \left[V_x - \frac{(2V_x - \Delta V)\,R_G}{R_{HRS}}\right] \frac{R_{HRS}}{R_{LRS} + R_{HRS}} \approx V_x. \tag{D.37}$$

- **Case 8 : $\mathbf{R_{PM_1} = R_{HRS}, R_{PM_2} = R_{HRS}, R_{QM_1} = R_{LRS}, R_{QM_2} = R_{HRS}}$**

For Case 8 where $R_{PM_1} = R_{HRS}, R_{PM_2} = R_{HRS}, R_{QM_1} = R_{LRS}, R_{QM_2} = R_{HRS}$, (D.1) becomes

$$V_C = \frac{(V_x - \Delta V)\,(R_G || (R_{LRS} + R_{HRS}))}{(R_G || (R_{LRS} + R_{HRS})) + (R_{HRS} + R_{HRS})} + \frac{V_x\,(R_G || (R_{HRS} + R_{HRS}))}{(R_G || (R_{HRS} + R_{HRS})) + (R_{LRS} + R_{HRS})}, \tag{D.38}$$

which can be approximated as,

$$V_C \approx \frac{(V_x - \Delta V)\,R_G}{R_G + 2R_{HRS}} + \frac{V_x R_G}{R_G + R_{LRS} + R_{HRS}} \approx 0. \tag{D.39}$$

Therefore, voltage across memristor PM_1 is approximately given as,

$$V_{PM_1} = [(V_x - \Delta V) - 0]\,\frac{R_{HRS}}{R_{HRS} + R_{HRS}} \approx \frac{V_x - \Delta V}{2}. \tag{D.40}$$

In a similar manner, voltage across memristor PM_2 is

$$V_{PM_2} = [(V_x - \Delta V) - 0]\,\frac{R_{HRS}}{R_{HRS} + R_{HRS}} \approx \frac{V_x - \Delta V}{2}, \tag{D.41}$$

voltage across QM_1 is

$$V_{QM_1} = [V_x - 0]\,\frac{R_{LRS}}{R_{LRS} + R_{HRS}} \approx 0, \tag{D.42}$$

and across QM_2 is

$$V_{QM_2} = [V_x - 0]\,\frac{R_{HRS}}{R_{LRS} + R_{HRS}} \approx V_x. \tag{D.43}$$

- **Case 9 : $\mathbf{R_{PM_1} = R_{LRS}, R_{PM_2} = R_{LRS}, R_{QM_1} = R_{HRS}, R_{QM_2} = R_{LRS}}$**

For Case 9 where $R_{PM_1} = R_{LRS}, R_{PM_2} = R_{LRS}, R_{QM_1} = R_{HRS}, R_{QM_2} = R_{LRS}$, (D.1) becomes

$$V_C = \frac{(V_x - \Delta V)\,(R_G || (R_{HRS} + R_{LRS}))}{(R_G || (R_{HRS} + R_{LRS})) + (R_{LRS} + R_{LRS})} + \frac{V_x\,(R_G || (R_{LRS} + R_{LRS}))}{(R_G || (R_{LRS} + R_{LRS})) + (R_{HRS} + R_{LRS})}, \tag{D.44}$$

which can be approximated as,

$$V_C \approx \frac{(V_x - \Delta V)\,R_G}{R_G + 2R_{LRS}} + \frac{V_x\,(2R_{LRS})}{3R_{LRS} + R_{HRS}} \approx V_x - \Delta V. \tag{D.45}$$

Therefore, voltage across memristor PM_1 is approximately given as,

$$V_{PM_1} = [(V_x - \Delta V) - (V_x - \Delta V)] \frac{R_{LRS}}{R_{LRS} + R_{LRS}} \approx 0. \tag{D.46}$$

In a similar manner, voltage across memristor PM_2 is

$$V_{PM_2} = [(V_x - \Delta V) - (V_x - \Delta V)] \frac{R_{LRS}}{R_{LRS} + R_{LRS}} \approx 0, \tag{D.47}$$

voltage across QM_1 is

$$V_{QM_1} = [V_x - (V_x - \Delta V)] \frac{R_{HRS}}{R_{LRS} + R_{HRS}} \approx \Delta V, \tag{D.48}$$

and across QM_2 is

$$V_{QM_2} = [V_x - (V_x - \Delta V)] \frac{R_{LRS}}{R_{LRS} + R_{HRS}} \approx 0. \tag{D.49}$$

- **Case 10 : $R_{PM_1} = R_{LRS}$, $R_{PM_2} = R_{HRS}$, $R_{QM_1} = R_{HRS}$, $R_{QM_2} = R_{LRS}$**

For Case 10 where $R_{PM_1} = R_{LRS}$, $R_{PM_2} = R_{HRS}$, $R_{QM_1} = R_{HRS}$, $R_{QM_2} = R_{LRS}$, (D.1) becomes

$$V_C = \frac{(V_x - \Delta V)\,(R_G || (R_{HRS} + R_{LRS}))}{(R_G || (R_{HRS} + R_{LRS})) + (R_{LRS} + R_{HRS})} + \frac{V_x\,(R_G || (R_{LRS} + R_{HRS}))}{(R_G || (R_{LRS} + R_{HRS})) + (R_{HRS} + R_{LRS})}, \tag{D.50}$$

which can be approximated as,

$$V_C \approx \frac{(V_x - \Delta V)\,R_G}{R_G + R_{LRS} + R_{HRS}} + \frac{V_x R_G}{R_G + R_{LRS} + R_{HRS}} \approx 0. \tag{D.51}$$

Therefore, voltage across memristor PM_1 is approximately given as,

$$V_{PM_1} = [(V_x - \Delta V) - 0] \frac{R_{LRS}}{R_{LRS} + R_{HRS}} \approx 0. \tag{D.52}$$

In a similar manner, voltage across memristor PM_2 is

$$V_{PM_2} = [(V_x - \Delta V) - 0] \frac{R_{HRS}}{R_{LRS} + R_{HRS}} \approx V_x - \Delta V, \tag{D.53}$$

voltage across QM_1 is

$$V_{QM_1} = [V_x - 0] \frac{R_{HRS}}{R_{LRS} + R_{HRS}} \approx V_x, \tag{D.54}$$

and across QM_2 is

$$V_{QM_2} = [V_x - 0] \frac{R_{LRS}}{R_{LRS} + R_{HRS}} \approx 0. \tag{D.55}$$

- **Case 11 : $R_{PM_1} = R_{HRS}$, $R_{PM_2} = R_{LRS}$, $R_{QM_1} = R_{HRS}$, $R_{QM_2} = R_{LRS}$**

For Case 11 where $R_{PM_1} = R_{HRS}$, $R_{PM_2} = R_{LRS}$, $R_{QM_1} = R_{HRS}$, $R_{QM_2} = R_{LRS}$, (D.1) becomes

$$V_C = \frac{(V_x - \Delta V)\,(R_G || (R_{HRS} + R_{LRS}))}{(R_G || (R_{HRS} + R_{LRS})) + (R_{HRS} + R_{LRS})} + \frac{V_x\,(R_G || (R_{HRS} + R_{LRS}))}{(R_G || (R_{HRS} + R_{LRS})) + (R_{HRS} + R_{LRS})}, \tag{D.56}$$

which can be approximated as,

$$V_C \approx \frac{(V_x - \Delta V)\,R_G}{R_G + R_{LRS} + R_{HRS}} + \frac{V_x R_G}{R_G + R_{LRS} + R_{HRS}} \approx 0. \tag{D.57}$$

Therefore, voltage across memristor PM_1 is approximately given as,

$$V_{PM_1} = [(V_x - \Delta V) - 0] \frac{R_{HRS}}{R_{LRS} + R_{HRS}} \approx V_x - \Delta V. \tag{D.58}$$

In a similar manner, voltage across memristor PM_2 is

$$V_{PM_2} = [(V_x - \Delta V) - 0] \frac{R_{LRS}}{R_{LRS} + R_{HRS}} \approx 0, \tag{D.59}$$

voltage across QM_1 is

$$V_{QM_1} = [V_x - 0] \frac{R_{HRS}}{R_{LRS} + R_{HRS}} \approx V_x, \tag{D.60}$$

and across QM_2 is

$$V_{QM_2} = [V_x - 0] \frac{R_{LRS}}{R_{LRS} + R_{HRS}} \approx 0. \tag{D.61}$$

- **Case 12 : $R_{PM_1} = R_{HRS}$, $R_{PM_2} = R_{HRS}$, $R_{QM_1} = R_{HRS}$, $R_{QM_2} = R_{LRS}$**

For Case 12 where $R_{PM_1} = R_{HRS}$, $R_{PM_2} = R_{HRS}$, $R_{QM_1} = R_{HRS}$, $R_{QM_2} = R_{LRS}$, (D.1) becomes

$$V_C = \frac{(V_x - \Delta V)\,(R_G || (R_{HRS} + R_{LRS}))}{(R_G || (R_{HRS} + R_{LRS})) + (R_{HRS} + R_{HRS})} + \frac{V_x\,(R_G || (R_{HRS} + R_{HRS}))}{(R_G || (R_{HRS} + R_{HRS})) + (R_{HRS} + R_{LRS})}, \tag{D.62}$$

which can be approximated as,

$$V_C \approx \frac{(V_x - \Delta V)\,R_G}{R_G + 2R_{HRS}} + \frac{V_x R_G}{R_G + R_{LRS} + R_{HRS}} \approx 0. \tag{D.63}$$

Therefore, voltage across memristor PM_1 is approximately given as,

$$V_{PM_1} = [(V_x - \Delta V) - 0]\,\frac{R_{HRS}}{R_{HRS} + R_{HRS}} \approx \frac{V_x - \Delta V}{2}. \tag{D.64}$$

In a similar manner, voltage across memristor PM_2 is

$$V_{PM_2} = [(V_x - \Delta V) - 0]\,\frac{R_{HRS}}{R_{HRS} + R_{HRS}} \approx \frac{V_x - \Delta V}{2}, \tag{D.65}$$

voltage across QM_1 is

$$V_{QM_1} = [V_x - 0]\,\frac{R_{HRS}}{R_{LRS} + R_{HRS}} \approx V_x, \tag{D.66}$$

and across QM_2 is

$$V_{QM_2} = [V_x - 0]\,\frac{R_{LRS}}{R_{LRS} + R_{HRS}} \approx 0. \tag{D.67}$$

- **Case 13 : $\mathbf{R_{PM_1} = R_{LRS}, R_{PM_2} = R_{LRS}, R_{QM_1} = R_{HRS}, R_{QM_2} = R_{HRS}}$**

For Case 13 where $R_{PM_1} = R_{LRS}$, $R_{PM_2} = R_{LRS}$, $R_{QM_1} = R_{HRS}$, $R_{QM_2} = R_{HRS}$, (D.1) becomes

$$V_C = \frac{(V_x - \Delta V)\,(R_G || (R_{HRS} + R_{HRS}))}{(R_G || (R_{HRS} + R_{HRS})) + (R_{LRS} + R_{LRS})} + \frac{V_x\,(R_G || (R_{LRS} + R_{LRS}))}{(R_G || (R_{LRS} + R_{LRS})) + (R_{HRS} + R_{HRS})}, \tag{D.68}$$

which can be approximated as,

$$V_C \approx \frac{(V_x - \Delta V)\,R_G}{R_G + 2R_{LRS}} + \frac{V_x\,(2R_{LRS})}{2R_{LRS} + 2R_{HRS}} \approx V_x - \Delta V. \tag{D.69}$$

Therefore, voltage across memristor PM_1 is approximately given as,

$$V_{PM_1} = [(V_x - \Delta V) - (V_x - \Delta V)] \frac{R_{LRS}}{R_{LRS} + R_{LRS}} \approx 0. \tag{D.70}$$

In a similar manner, voltage across memristor PM_2 is

$$V_{PM_2} = [(V_x - \Delta V) - (V_x - \Delta V)] \frac{R_{LRS}}{R_{LRS} + R_{LRS}} \approx 0, \tag{D.71}$$

voltage across QM_1 is

$$V_{QM_1} = [V_x - (V_x - \Delta V)] \frac{R_{HRS}}{R_{HRS} + R_{HRS}} \approx \frac{\Delta V}{2}, \tag{D.72}$$

and across QM_2 is

$$V_{QM_2} = [V_x - (V_x - \Delta V)] \frac{R_{HRS}}{R_{HRS} + R_{HRS}} \approx \frac{\Delta V}{2}. \tag{D.73}$$

- **Case 14 : $R_{PM_1} = R_{LRS}$, $R_{PM_2} = R_{HRS}$, $R_{QM_1} = R_{HRS}$, $R_{QM_2} = R_{HRS}$**

For Case 14 where $R_{PM_1} = R_{LRS}$, $R_{PM_2} = R_{HRS}$, $R_{QM_1} = R_{HRS}$, $R_{QM_2} = R_{HRS}$, (D.1) becomes

$$V_C = \frac{(V_x - \Delta V)\,(R_G || (R_{HRS} + R_{HRS}))}{(R_G || (R_{HRS} + R_{HRS})) + (R_{LRS} + R_{HRS})} + \frac{V_x\,(R_G || (R_{LRS} + R_{HRS}))}{(R_G || (R_{LRS} + R_{HRS})) + (R_{HRS} + R_{HRS})}, \tag{D.74}$$

which can be approximated as,

$$V_C \approx \frac{(V_x - \Delta V)\,R_G}{R_G + R_{LRS} + R_{HRS}} + \frac{V_x R_G}{R_G + 2R_{HRS}} \approx 0. \tag{D.75}$$

Therefore, voltage across memristor PM_1 is approximately given as,

$$V_{PM_1} = [(V_x - \Delta V) - 0] \frac{R_{LRS}}{R_{LRS} + R_{HRS}} \approx 0. \tag{D.76}$$

In a similar manner, voltage across memristor PM_2 is

$$V_{PM_2} = [(V_x - \Delta V) - 0] \frac{R_{HRS}}{R_{LRS} + R_{HRS}} \approx V_x - \Delta V, \tag{D.77}$$

voltage across QM_1 is

$$V_{QM_1} = [V_x - 0]\frac{R_{HRS}}{R_{HRS}+R_{HRS}} \approx \frac{V_x}{2}, \tag{D.78}$$

and across QM_2 is

$$V_{QM_2} = [V_x - 0]\frac{R_{HRS}}{R_{HRS}+R_{HRS}} \approx \frac{V_x}{2}. \tag{D.79}$$

- **Case 15 : $R_{PM_1} = R_{HRS}$, $R_{PM_2} = R_{LRS}$, $R_{QM_1} = R_{HRS}$, $R_{QM_2} = R_{HRS}$**

For Case 15 where $R_{PM_1} = R_{HRS}$, $R_{PM_2} = R_{LRS}$, $R_{QM_1} = R_{HRS}$, $R_{QM_2} = R_{HRS}$, (D.1) becomes

$$V_C = \frac{(V_x - \Delta V)\,(R_G||\,(R_{HRS}+R_{HRS}))}{(R_G||\,(R_{HRS}+R_{HRS}))+(R_{HRS}+R_{LRS})} + \frac{V_x\,(R_G||\,(R_{HRS}+R_{LRS}))}{(R_G||\,(R_{HRS}+R_{LRS}))+(R_{HRS}+R_{HRS})}, \tag{D.80}$$

which can be approximated as,

$$V_C \approx \frac{(V_x - \Delta V)\,R_G}{R_G+R_{LRS}+R_{HRS}} + \frac{V_x R_G}{R_G+2R_{HRS}} \approx 0. \tag{D.81}$$

Therefore, voltage across memristor PM_1 is approximately given as,

$$V_{PM_1} = [(V_x - \Delta V) - 0]\frac{R_{HRS}}{R_{LRS}+R_{HRS}} \approx V_x - \Delta V. \tag{D.82}$$

In a similar manner, voltage across memristor PM_2 is

$$V_{PM_2} = [(V_x - \Delta V) - 0]\frac{R_{LRS}}{R_{LRS}+R_{HRS}} \approx 0, \tag{D.83}$$

voltage across QM_1 is

$$V_{QM_1} = [V_x - 0]\frac{R_{HRS}}{R_{HRS}+R_{HRS}} \approx \frac{V_x}{2}, \tag{D.84}$$

and across QM_2 is

$$V_{QM_2} = [V_x - 0]\frac{R_{HRS}}{R_{HRS}+R_{HRS}} \approx \frac{V_x}{2}. \tag{D.85}$$

- **Case 16 : $R_{PM_1} = R_{HRS}$, $R_{PM_2} = R_{HRS}$, $R_{QM_1} = R_{HRS}$, $R_{QM_2} = R_{HRS}$**

For Case 16 where $R_{PM_1} = R_{HRS}$, $R_{PM_2} = R_{HRS}$, $R_{QM_1} = R_{HRS}$, $R_{QM_2} = R_{HRS}$, (D.1) becomes

$$V_C = \frac{(V_x - \Delta V)\,(R_G || (R_{HRS} + R_{HRS}))}{(R_G || (R_{HRS} + R_{HRS})) + (R_{HRS} + R_{HRS})} + \frac{V_x\,(R_G || (R_{HRS} + R_{HRS}))}{(R_G || (R_{HRS} + R_{HRS})) + (R_{HRS} + R_{HRS})}, \tag{D.86}$$

which can be approximated as,

$$V_C \approx \frac{(V_x - \Delta V)\,R_G}{R_G + 2R_{HRS}} + \frac{V_x R_G}{R_G + 2R_{HRS}} \approx 0. \tag{D.87}$$

Therefore, voltage across memristor PM_1 is approximately given as,

$$V_{PM_1} = [(V_x - \Delta V) - 0]\,\frac{R_{HRS}}{R_{HRS} + R_{HRS}} \approx \frac{V_x - \Delta V}{2}. \tag{D.88}$$

In a similar manner, voltage across memristor PM_2 is

$$V_{PM_2} = [(V_x - \Delta V) - 0]\,\frac{R_{HRS}}{R_{HRS} + R_{HRS}} \approx \frac{V_x - \Delta V}{2}, \tag{D.89}$$

voltage across QM_1 is

$$V_{QM_1} = [V_x - 0]\,\frac{R_{HRS}}{R_{HRS} + R_{HRS}} \approx \frac{V_x}{2}, \tag{D.90}$$

and across QM_2 is

$$V_{QM_2} = [V_x - 0]\,\frac{R_{HRS}}{R_{HRS} + R_{HRS}} \approx \frac{V_x}{2}. \tag{D.91}$$

Appendix E

Materials and Properties

The information presented here is taken directly as it is from [170] in order to know the various materials being used in memristors and for their top and bottom electrode. Also some of the properties of memristors and their values are specified in the end.

Table E.1. Materials used in anion devices (VCM) along with top and bottom electrode material.

Sr. No.	Insulators	Bottom Electrode	Top Electrode	Switching Mode
1	MgO	Pt	Pt	Unipolar
2	TiO_x	Ru, Pt	Al, Pt	Non/Uni /Bipolar
3	ZrO_x	P^+- Si, n^+- Si	Pt, Cr	Uni/Bipolar
4	HfO_x	TiN	TiN	Bipolar
5	VO_x	N/A	N/A	Threshold
6	NbO_x	P^+- Si	Pt	Unipolar
7	TaO_x	Pt, Ta	Pt, Ta	Bipolar
8	CrO_x	TiN	Pt	Bipolar
9	MoO_x	Pt	Pt-Ir	Uni/Bipolar
10	WO_x	W, FTO	TiN, Au	Bipolar

11	MnO_x	Pt	Al, TiN	Bipolar
12	FeO_x	Pt	Pt	Non/Bipolar
13	CoO_x	Pt	Pt	Nonpolar
14	NiO_x	Pt	Pt	Nonpolar /Threshold
15	CuO_x	TiN, TaN, SRO, Pt	Pt	Bipolar
16	ZnO_x	Pt, Au	TiN, Ag	Bipolar
17	AlO_x	Ru, Pt	Pt, Ti	Unipolar /Bipolar
18	GaO_x	ITO	Pt, Ti	Bipolar
19	SiO_x	Poly-Si, TiW	Poly-Si, TiW	Unipolar
20	SiO_xN_y	W	Cu	Bipolar
21	GeO_x	ITO, TaN	Pt, Ni	Bipolar
22	SnO_x	Pt	Pt	Unipolar
23	BiO_x	Bi	W, Re, Ag, Cu	Bipolar
24	SbO_x	Pt	Sb	Unipolar /Bipolar
25	SmO_x	TiN	Pt	Bipolar
26	GdO_x	Pt	Pt	Unipolar
27	YO_x	Al	Al	Unipolar
28	CeO_x	Pt	Al	Bipolar
29	EuO_x	TaN	Ru	Unipolar /Bipolar
30	PrO_x	TaN	Ru	Bipolar
31	ErO_x	TaN	Ru	Unipolar
32	DyO_x	TaN	Ru	Unipolar
33	NdO_x	TaN	Ru	Unipolar
34	$Ba_{0.7}Sr_{0.3}TiO_3$	$SrRuO_3$	Pt, W	Bipolar

35	$SrTiO_3$	$SrRuO_3$, Au, Pt	Au, Pt	Bipolar
36	$SrZrO_3$	$SrRuO_3$	Au	Bipolar
37	$BiFeO_3$	$LaNiO_3$	Pt	Bipolar
38	$Pr_{0.7}Ca_{0.3}MnO_3$	YBCO, Pt, $LaAlO_3$	Ag	Bipolar
39	$La_{0.33}Sr_{0.67}FeO_3$	Au	Al	Bipolar
40	$Pr_yLa_{0.625-y}Ca_{0.375}MnO_3$	Ag	Ag	Bipolar
41	Nitrides (AlN)	Al, TiN, Pt	Al, TiN, Pt	Bipolar
42	Telluride (ZnTe)	Si	Au	Bipolar
43	selenide (ZnSe)	P^+-Ge	In, In-Zn	Bipolar
44	Polymers	Al, ITO, Cu	Al, ITO, Cu	Bipolar

Table E.2. Materials used in cation devices along with materials for top electrode, bottom electrode and switching modes.

Sr. No.	Electrolytes	Bottom Electrode	Top Electrode	Switching Mode
Sulfides:				
1	Ge_xS_x	W	Ag	Bipolar
2	As_2S_3	Au	Ag	Bipolar
3	Cu_2S	Cu	Pt	Bipolar
4	$Zn_xCd_{1-x}S$	Pt	Ag	Bipolar
Iodides:				
5	AgI	Pt	Ag	Bipolar
6	$RbAg_4I$	Pt	Ag	Bipolar
Selenides:				
7	Ge_xSe_y	W	Ag, Cu	Bipolar
Tellurides:				
8	Ge_xTe_y	TiW	Ag	Bipolar
Ternary chalcogenides:				
9	Ge-Sb-Te	Mo	Au, Ag	Bipolar
Oxides:				
10	Ta_2O_5	Pt	Cu	Bipolar
11	SiO_2	W	Cu	Unipolar/ Bipolar
12	HfO_2	Pt	Cu	Bipolar
13	WO_3	Pt	Cu	Bipolar
14	ZrO_2	Ag	Au	Bipolar
15	$SrTiO_3$	Pt	Ag	Bipolar
16	TiO_2	Pt	Ag	Bipolar
17	CuO_x	Cu	Al	Unipolar
18	ZnO	Pt, Al doped ZnO	Cu	Bipolar
19	Al_2O_3	Al	Cu	Bipolar
20	MoO_x	Cu	Pt	Bipolar
21	GdO_x	Pt	Cu doped MoO_x	
Others:				
22	MSQ	Pt	Ag	Bipolar
23	Doped Organic Semiconductors	Pt	Cu	Bipolar
24	Nitrides	Pt	Cu	Bipolar
25	Amorphous Si	P^+-Si	Ag	Bipolar
26	Carbon	Pt	Cu	Bipolar
27	Vacuum Gaps	$RbAg_4I_5$/Ag, Ag_2S/Ag	W, Pt	Bipolar

Table E.3. Various properties of switching materials used in memristors and their values in typical applications.

Property	Storage	Memory	Logic	Neuro	Best Reported
Reproducibility	$<$10%	$<$1 %	$<$20 %	$<$50 %	few percent
Endurance	$>10^4$	$>10^{16}$	$>$100	$>$100 - 10^6	10^{12}
Switching Energy	$<$1 pJ	$<$5 pJ	not critical	not critical	1 pJ
Switching Speed	$<10\ \mu$s	$<$1 ns	not critical	not critical	100 ps
Retention	$>$10 Years	$>$Minutes - months	$>$days	$>$seconds-days	10^{14}
ON/OFF current ratio	$>$10	$>$50	$>$100	$>$500	$>10^{11}$
OFF-state Resistance	$>$1 MΩ	$>$100 kΩ	$>$10 MΩ	$>$100 MΩ	1 kΩ -100 GΩ
Density	$>$(10 nm)$^{-2}$ and $>$4 layers	$>$(10 nm)$^{-2}$	$>$(20 nm)$^{-2}$ & multilayer	$>$(10-100 nm)$^{-2}$ & multilayer	4 layer 1/(10 nm)^2
Number of States	2-16	2	2-16	2-32	$\sim$ 100
I-V nonlinearity	$>$100 - 10000	$>$100 - 1000	$>$10	not critical	100

Bibliography

[1] I. Kuon and J. Rose, *Quantifying and exploring the gap between FPGAs and ASICs*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[2] D. B. Strukov, K. K. Likharev, and R. Waser(Eds), *Nanoelectronics and information technology*, 3rd ed. Wiley, 2012.

[3] G. Lemieux and D. Lewis, "Circuit design of routing switches," in *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. ACM Press, 2002.

[4] S. Hauck and A. DeHon, Eds., *Reconfigurable computing: The theory and practice of FPGA-based computation*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[5] V. George and J. M. Rabaey, *Low-energy FPGAs - Architecture and design*. Springer Science+Business Media, LLC, 2001.

[6] G. Lemieux and D. Lewis, *Design of interconnection networks for programmable logic*. Norwell, MA, USA: Kluwer Academic Publishers, 2004.

[7] I. Kuon, R. Tessier, and J. Rose, "FPGA architecture: Survey and challenges," *Foundations and Trends in Electronic Design Automation*, vol. 2, no. 2, pp. 135–253, 2008.

[8] K. J. Han, N. Chan, S. Kim, B. Leung, V. Hecht, B. Cronquist, D. Shum, A. Tilke, L. Pescini, M. Stiftinger, and R. Kakoschke, "A novel flash-based FPGA technology with deep trench isolation," in *2007 22nd IEEE Non-Volatile Semiconductor Memory Workshop*, 2007, pp. 32–33.

[9] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 203–215, 2007.

[10] A. DeHon, "Reconfigurable architectures for general-purpose computing," Ph.D. dissertation, MIT, 1996.

[11] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 3, pp. 288–298, 2004.

[12] F. Li, Y. Lin, L. He, D. Chen, and J. Cong, "Power modeling and characteristics of field programmable gate arrays," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 11, pp. 1712–1724, 2005.

[13] J. Rose, R. Francis, D. Lewis, and P. Chow, "Architecture of field-programmable gate arrays: the effect of logic block functionality on area efficiency," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 5, pp. 1217–1225, 1990.

[14] J. Kouloheris and A. El Gamal, "FPGA performance versus cell granularity," in *Proceedings of the IEEE 1991 Custom Integrated Circuits Conference*, 1991, pp. 6.2/1–4.

[15] S. Singh, J. Rose, P. Chow, and D. Lewis, "The effect of logic block architecture on FPGA performance," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 3, pp. 281–287, 1992.

[16] Y. Xie, G. H. Loh, B. Black, and K. Bernstein, "Design space exploration for 3D architectures," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 2, no. 2, pp. 65–103, 2006.

[17] W. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. Sule, M. Steer, and P. Franzon, "Demystifying 3D ICs: the pros and cons of going vertical," *IEEE Design Test of Computers*, vol. 22, no. 6, pp. 498–510, 2005.

[18] A. Rahman, S. Das, A. Chandrakasan, and R. Reif, "Wiring requirement and three-dimensional integration technology for field programmable gate arrays," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 1, pp. 44–54, 2003.

[19] M. Lin, A. E. Gamal, Y.-C. Lu, and S. Wong, "Performance benefits of monolithically stacked 3D-FPGA," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 216–229, 2007.

[20] Y. Chen, J. Zhao, and Y. Xie, "3D-nonFAR: Three-dimensional non-volatile FPGA architecture using phase change memory," in *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design*, ser. ISLPED '10, 2010, pp. 55–60.

[21] C. Dong, D. Chen, S. Haruehanroengra, and W. Wang, "3D nFPGA: A reconfigurable architecture for 3D CMOS/nanomaterial hybrid digital circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 11, pp. 2489–2501, 2007.

[22] C. Chen, R. Parsa, N. Patil, S. Chong, K. Akarvardar, J. Provine, D. Lewis, J. Watt, R. T. Howe, H.-S. P. Wong, and S. Mitra, "Efficient FPGAs using nanoelectromechanical relays," in *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '10, 2010, pp. 273–282.

[23] S. Seo, M. J. Lee, D. H. Seo, E. J. Jeoung, D.-S. Suh, Y. S. Joung, I. K. Yoo, I. R. Hwang, S. H. Kim, I. S. Byun, J.-S. Kim, J. S. Choi, and B. H. Park, "Reproducible resistance switching in polycrystalline NiO films," *Applied Physics Letters*, vol. 85, no. 23, pp. 5655–5657, 2004.

[24] K. Akarvardar, D. Elata, R. Parsa, G. Wan, K. Yoo, J. Provine, P. Peumans, R. Howe, and H.-S. Wong, "Design considerations for complementary nanoelectromechanical logic gates," in *IEEE International Electron Devices Meeting IEDM 2007*, 2007, pp. 299–302.

[25] R. Nathanael, V. Pott, H. Kam, J. Jeon, and T.-J. K. Liu, "4-terminal relay technology for complementary logic," in *2009 IEEE International Electron Devices Meeting (IEDM)*, 2009, pp. 1–4.

[26] F. Chen, H. Kam, D. Markovic, T.-J. K. Liu, V. Stojanovic, and E. Alon, "Integrated circuit design with NEM relays," in *IEEE/ACM International Conference on Computer-Aided Design ICCAD*, 2008, pp. 750–757.

[27] R. Bez and A. Pirovano, "Non-volatile memory technologies: emerging concepts and new materials," *Materials Science in Semiconductor Processing*, vol. 7, no. 4–6, pp. 349 – 355, 2004.

[28] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable DRAM alternative," in *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ser. ISCA '09. New York, NY, USA: ACM, 2009, pp. 2–13.

[29] X. Dong, N. Jouppi, and Y. Xie, "PCRAMsim: System-level performance, energy, and area modeling for Phase-Change RAM," in *IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers ICCAD.*, 2009, pp. 269–275.

[30] F. Bedeschi, R. Fackenthal, C. Resta, E. Donze, M. Jagasivamani, E. Buda, F. Pellizzer, D. Chow, A. Cabrini, G. Calvi, R. Faravelli, A. Fantini, G. Torelli, D. Mills, R. Gastaldi, and G. Casagrande, "A bipolar-selected phase change memory featuring multi-level cell storage," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 217–227, 2009.

[31] Y. S. Chen, H. Lee, P. Chen, P. Gu, C. Chen, W. Lin, W. Liu, Y. Hsu, S. Sheu, P.-C. Chiang, W.-S. Chen, F. Chen, C. Lien, and M. J. Tsai, "Highly scalable hafnium oxide memory with improvements of resistive distribution and read disturb immunity," in *IEEE International Electron Devices Meeting (IEDM)*, 2009, pp. 1–4.

[32] S.-S. Sheu, P.-C. Chiang, W.-P. Lin, H.-Y. Lee, P.-S. Chen, Y.-S. Chen, T.-Y. Wu, F. Chen, K.-L. Su, M.-J. Kao, K.-H. Cheng, and M.-J. Tsai, "A 5ns fast write multi-level non-volatile 1 K bits RRAM memory with advance write scheme," in *Symposium on VLSI Circuits*, 2009, pp. 82–83.

[33] C.-H. Wang, Y.-H. Tsai, K.-C. Lin, M.-F. Chang, Y.-C. King, C.-J. Lin, S.-S. Sheu, Y.-S. Chen, H.-Y. Lee, F. Chen, and M.-J. Tsai, "Three-dimensional $4F^2$ ReRAM cell with CMOS logic compatible process," in *IEEE International Electron Devices Meeting (IEDM)*, 2010, pp. 29.6.1–4.

[34] M. Mahvash and A. Parker, "A memristor SPICE model for designing memristor circuits," in *53rd IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2010, pp. 989–992.

[35] S. Yu, J. Liang, Y. Wu, and H.-S. P. Wong, "Read/write schemes analysis for novel complementary resistive switches in passive crossbar memory arrays," *Nanotechnology*, vol. 21, no. 46, pp. 465 202:1–5, 2010.

[36] M. Liu and W. Wang, "rFGA: CMOS-nano hybrid FPGA using RRAM components," in *IEEE International Symposium on Nanoscale Architectures NANOARCH*, 2008, pp. 93–98.

[37] C. Xu, X. Dong, N. Jouppi, and Y. Xie, "Design implications of memristor-based RRAM cross-point structures," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2011, pp. 1–6.

[38] P.-E. Gaillardon, M. Ben-Jamaa, G. Beneventi, F. Clermidy, and L. Perniola, "Emerging memory technologies for reconfigurable routing in FPGA architecture," in *17th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, 2010, pp. 62–65.

[39] K. Tsunoda, K. Kinoshita, H. Noshiro, Y. Yamazaki, T. Iizuka, Y. Ito, A. Takahashi, A. Okano, Y. Sato, T. Fukano, M. Aoki, and Y. Sugiyama, "Low power and high speed switching of Ti-doped NiO ReRAM under the unipolar voltage source of less than 3 V," in *IEEE International Electron Devices Meeting (IEDM)*, 2007, pp. 767–770.

[40] R. Huang, L. Zhang, D. Gao, Y. Pan, S. Qin, P. Tang, Y. Cai, and Y. Wang, "Resistive switching of silicon rich oxide featuring high compatibility with CMOS technology for 3D stackable and embedded applications," *Applied Physics A*, vol. 102, no. 4, pp. 927–931, 2011.

[41] Z. Abid, D. Homouz, B. Mohammad, and W. Wang, "Memristors-based NMOS logic circuits," in *24th International Conference on Microelectronics (ICM)*, 2012, pp. 1–4.

[42] J. Cong and B. Xiao, "mrFPGA: A novel FPGA architecture with memristor-based reconfiguration," in *IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, 2011, pp. 1–8.

[43] A. Dehon, "Array-based architecture for molecular electronics," in *Proceedings of the First Workshop on Non-Silicon Computation (NSC-1)*, 2001, pp. 1–8.

[44] M. Ziegler and M. Stan, "CMOS/nano co-design for crossbar-based molecular electronic systems," *IEEE Transactions on Nanotechnology*, vol. 2, no. 4, pp. 217–230, 2003.

[45] ——, "The CMOS/nano interface from a circuits perspective," in *Proceedings of the 2003 International Symposium on Circuits and Systems ISCAS'03*, vol. 4, 2003, pp. 904–907.

[46] D. B. Strukov and K. K. Likharev, "CMOL FPGA: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices," *Nanotechnology*, vol. 16, no. 6, pp. 888–900, 2005.

[47] P. Kuekes and R. Williams, "Demultiplexer for a molecular wire crossbar network (MWCN DEMUX)," Jul. 3 2001, US Patent 6,256,767. [Online]. Available: http://www.google.co.in/patents/US6256767

[48] Z. Zhong, D. Wang, Y. Cui, M. W. Bockrath, and C. M. Lieber, "Nanowire crossbar arrays as address decoders for integrated nanosystems," *Science*, vol. 302, no. 5649, pp. 1377–1379, 2003.

[49] P. J. Kuekes, W. Robinett, G. Seroussi, and R. S. Williams, "Defect-tolerant interconnect to nanoelectronic circuits: Internally redundant demultiplexers based on error-correcting codes," *Nanotechnology*, vol. 16, pp. 869–882, 2005.

[50] G. Snider and W. Robinett, "Crossbar demultiplexers for nanoelectronics based on n-hot codes," *IEEE Transactions on Nanotechnology*, vol. 4, no. 2, pp. 249–254, 2005.

[51] K. Gopalakrishnan, R. Shenoy, C. Rettner, R. King, Y. Zhang, B. Kurdi, L. Bozano, J. Welser, M. Rothwell, M. Jurich, M. Sanchez, M. Hernandez, P. Rice, W. Risk, and H. Wickramasinghe, "The micro to nano addressing block (MNAB)," in *IEEE International Electron Devices Meeting IEDM Technical Digest*, 2005, pp. 471–474.

[52] P. J. Kuekes, W. Robinett, R. M. Roth, G. Seroussi, G. S. Snider, and R. S. Williams, "Resistor-logic demultiplexers for nanoelectronics based on constant-weight codes," *Nanotechnology*, vol. 17, no. 4, pp. 1052–1061, 2006.

[53] D. B. Strukov and K. K. Likharev, "A reconfigurable architecture for hybrid CMOS/nanodevice circuits," in *Proceedings of the 2006 ACM/SIGDA 14th International Symposium on Field Programmable Gate Arrays*, ser. FPGA '06. New York, NY, USA: ACM, 2006, pp. 131–140.

[54] G. S. Snider and R. S. Williams, "Nano/CMOS architectures using a field-programmable nanowire interconnect," *Nanotechnology*, vol. 18, no. 3, pp. 035 204:1–11, 2007.

[55] M. C. McAlpine, R. S. Friedman, and C. M. Lieber, "Nanoimprint lithography for hybrid plastic electronics," *Nano Letters*, vol. 3, no. 4, pp. 443–445, 2003.

[56] K. Kim, S. Shin, and S.-M. Kang, "Field programmable stateful logic array," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 12, pp. 1800–1813, 2011.

[57] L. Chua, "Memristor-The missing circuit element," *IEEE Transactions on Circuit Theory*, vol. 18, no. 5, pp. 507–519, 1971.

[58] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, pp. 80–83, 2008.

[59] R. Williams, "How we found the missing memristor," *IEEE Spectrum*, vol. 45, no. 12, pp. 28–35, 2008.

[60] B. Magyari-Köpe, S. Park, H.-D. Lee, and Y. Nishi, "First principles calculations of oxygen vacancy-ordering effects in resistance change memory materials incorporating binary transition metal oxides," *Journal of Materials Science*, vol. 47, no. 21, pp. 7498–7514, 2012.

[61] M. Fujimoto, H. Koyama, M. Konagai, Y. Hosoi, K. Ishihara, S. Ohnishi, and N. Awaya, "TiO_2 anatase nanolayer on TiN thin film exhibiting high-speed bipolar resistive switching," *Applied Physics Letters*, vol. 89, no. 22, pp. 223 509:1–3, 2006.

[62] D. R. Stewart, D. A. A. Ohlberg, P. A. Beck, Y. Chen, R. S. Williams, J. O. Jeppesen, K. A. Nielsen, and J. F. Stoddart, "Molecule-independent electrical switching in Pt/organic monolayer/Ti devices," *Nano Letters*, vol. 4, no. 1, pp. 133–136, 2004.

[63] L. Courtade, C. Turquat, J. Lisoni, L. Goux, D. Wouters, D. Deleruyelle, and C. Muller, "Integration of resistive switching NiO in small via structures from localized oxidation of nickel metallic layer," in *38th European Solid-State Device Research Conference ESSDERC*, 2008, pp. 218–221.

[64] Y. Wang, Q. Liu, H. Lü, S. Long, W. Wang, Y. Li, S. Zhang, W. Lian, J. Yang, and M. Liu, "Improving the electrical performance of resistive switching memory using doping technology," *Chinese Science Bulletin*, vol. 57, no. 11, pp. 1235–1240, 2012.

[65] W. Guan, S. Long, Q. Liu, M. Liu, and W. Wang, "Nonpolar nonvolatile resistive switching in Cu doped ZrO_2," *IEEE Electron Device Letters*, vol. 29, no. 5, pp. 434–437, 2008.

[66] Y. Dong, G. Yu, M. C. McAlpine, W. Lu, and C. M. Lieber, "Si/a-Si core/shell nanowires as nonvolatile crossbar switches," *Nano Letters*, vol. 8, no. 2, pp. 386–391, 2008.

[67] J. Borghetti, D. B. Strukov, M. D. Pickett, J. J. Yang, D. R. Stewart, and R. S. Williams, "Electrical transport and thermometry of electroformed titanium dioxide memristive switches," *Journal of Applied Physics*, vol. 106, no. 12, pp. 124 504:1–5, 2009.

[68] L. Chua, "Resistance switching memories are memristors," *Applied Physics A*, vol. 102, no. 4, pp. 765–783, 2011.

[69] J. J. Yang, M. D. Pickett, X. Li, D. A. A. Ohlberg, D. R. Stewart, and R. Williams, "Memristive switching mechanism for metal/oxide/metal nanodevices," *Nature Nanotechnology*, vol. 3, no. 7, pp. 429–433, 2008.

[70] T. Chang, S.-H. Jo, K.-H. Kim, P. Sheridan, S. Gaba, and W. Lu, "Synaptic behaviors and modeling of a metal oxide memristive device," *Applied Physics A*, vol. 102, no. 4, pp. 857–863, 2011.

[71] F. Corinto, A. Ascoli, and M. Gilli, "Nonlinear dynamics of memristor oscillators," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 6, pp. 1323–1336, 2011.

[72] A. Talukdar, A. Radwan, and K. Salama, "Non linear dynamics of memristor based 3rd order oscillatory system," *Microelectronics Journal*, vol. 43, no. 3, pp. 169 – 175, 2012.

[73] D. Strukov, D. Stewart, J. Borghetti, X. Li, M. Pickett, G. Ribeiro, W. Robinett, G. Snider, J. Strachan, W. Wu, Q. Xia, J. Yang, and R. Williams, "Hybrid CMOS/memristor circuits," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2010, pp. 1967–1970.

[74] Y. V. Pershin, S. La Fontaine, and M. Di Ventra, "Memristive model of amoeba learning," *Phys. Rev. E*, vol. 80, pp. 021 926:1–6, 2009.

[75] G. K. Johnsen, "An introduction to the memristor – a valuable circuit element in bioelectricity and bioimpedance," *Journal of electrical bioimpedance*, vol. 3, pp. 20–28, 2012.

[76] A. C. Torrezan, J. P. Strachan, G. Medeiros-Ribeiro, and R. S. Williams, "Sub-nanosecond switching of a tantalum oxide memristor," *Nanotechnology*, vol. 22, no. 48, pp. 485 203:1–7, 2011.

[77] L. Chua, "The fourth element," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1920–1927, 2012.

[78] D. Biolek, Z. Biolek, V. Biolkova, and Z. Kolka, "Some fingerprints of ideal memristors," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2013, pp. 201–204.

[79] Z. Biolek, D. Biolek, and V. Biolkova, "Computation of the area of memristor pinched hysteresis loop," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 9, pp. 607–611, 2012.

[80] L. Chua and S. M. Kang, "Memristive devices and systems," *Proceedings of the IEEE*, vol. 64, no. 2, pp. 209–223, 1976.

[81] S. Kvatinsky, E. Friedman, A. Kolodny, and U. Weiser, "TEAM: ThrEshold Adaptive Memristor Model," *IEEE Transactions on Circuits and Systems I: Regular Papers,*, vol. 60, no. 1, pp. 211–221, 2013.

[82] S. Kvatinsky, M. Ramadan, E. Friedman, and A. Kolodny, "VTEAM: A general model for voltage-controlled memristors," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 62, no. 8, pp. 786–790, 2015.

[83] ITRS, "Emerging research devices," ITRS, Tech. Rep., 2009. [Online]. Available: http://www.itrs.net/Links/2009ITRS/2009Chapters_2009Tables/2009_ERD.pdf

[84] D. A. Allwood and R. P. Cowburn, *Magnetic domain wall logic*. Wiley-VCH Verlag GmbH & Co. KGaA, 2010.

[85] J. Wang, H. Meng, and J.-P. Wang, "Programmable spintronics logic device based on a magnetic tunnel junction element," *Journal of Applied Physics*, vol. 97, no. 10, pp. 10D509:1–3, 2005.

[86] H. Kimura, T. Hanyu, M. Kameyama, Y. Fujimori, T. Nakamura, and H. Takasu, "Complementary ferroelectric-capacitor logic for low-power logic-in-memory VLSI," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 6, pp. 919–926, 2004.

[87] A. N. Whitehead and B. Russell, "Principia mathematica. Vol. I." Cambridge: University Press. XV u. 666 S. 4° (1910), 1910.

[88] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. S. Williams, "Memristive switches enable stateful logic operations via material implication," *Nature*, vol. 464, pp. 873–876, 2010.

[89] Y. Chen, G.-Y. Jung, D. A. A. Ohlberg, X. Li, D. R. Stewart, J. O. Jeppesen, K. A. Nielsen, J. F. Stoddart, and R. S. Williams, "Nanoscale molecular-switch crossbar circuits," *Nanotechnology*, vol. 14, no. 4, pp. 462–468, 2003.

[90] Y. Chen, D. A. A. Ohlberg, X. Li, D. R. Stewart, R. Stanley Williams, J. O. Jeppesen, K. A. Nielsen, J. F. Stoddart, D. L. Olynick, and E. Anderson, "Nanoscale molecular-switch devices fabricated by imprint lithography," *Applied Physics Letters*, vol. 82, no. 10, pp. 1610–1612, 2003.

[91] G.-Y. Jung, E. Johnston-Halperin, W. Wu, Z. Yu, S.-Y. Wang, W. M. Tong, Z. Li, J. E. Green, B. A. Sheriff, A. Boukai, Y. Bunimovich, J. R. Heath, and R. S. Williams, "Circuit fabrication at 17 nm half-pitch by nanoimprint lithography," *Nano Letters*, vol. 6, no. 3, pp. 351–354, 2006.

[92] G. Y. Jung, S. Ganapathiappan, D. A. A. Ohlberg, D. L. Olynick, Y. Chen, W. M. Tong, and R. S. Williams, "Fabrication of a 34 $\times$ 34 crossbar structure at 50 nm half-pitch by UV-based nanoimprint lithography," *Nano Letters*, vol. 4, no. 7, pp. 1225–1229, 2004.

[93] J. E. Green, J. W. Choi, A. Boukai, Y. Bunimovich, E. Johnston-Halperin, E. DeIonno, Y. Luo, B. A. Sheriff, K. Xu, Y. S. Shin, H.-R. Tseng, J. F. Stoddart, and J. R. Heath,

"A 160-kilobit molecular electronic memory patterned at 10^{11} bits per square centimetre," *Nature*, vol. 445, no. 7126, pp. 414–417, 2007.

[94] I. Ebong and P. Mazumder, "CMOS and memristor-based neural network design for position detection," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 2050–2060, 2012.

[95] D. T., Q. J., K. S., K. H. T., K. B. J., P. Y. V., D. V. M., and B. D. N., "Memristive adaptive filters," *Applied Physics Letters*, vol. 97, no. 9, pp. 093 502:1–3, 2010.

[96] S. Shin, K. Kim, and S.-M. Kang, "Memristor applications for programmable analog ICs," *IEEE Transactions on Nanotechnology,*, vol. 10, no. 2, pp. 266–274, 2011.

[97] T. Serrano-Gotarredona, T. Masquelier, T. Prodromakis, G. Indiveri, and B. Linares-Barranco, "STDP and STDP variations with memristors for spiking neuromorphic learning systems," *Frontiers in Neuroscience*, vol. 7, no. 2, pp. 1–15, 2013.

[98] A. Gelencser, T. Prodromakis, C. Toumazou, and T. Roska, "Biomimetic model of the outer plexiform layer by incorporating memristive devices," *Phys. Rev. E*, vol. 85, pp. 041 918:1–10, 2012.

[99] G. Indiveri, B. Linares-Barranco, R. Legenstein, G. Deligeorgis, and T. Prodromakis, "Integration of nanoscale memristor synapses in neuromorphic computing architectures," *Nanotechnology*, vol. 24, no. 38, pp. 384 010:1–13, 2013.

[100] R. Waser and M. Aono, "Nanoionics-based resistive switching memories," *Nature Materials*, vol. 6, no. 11, pp. 833–840, 2007.

[101] S. H. Jo, K.-H. Kim, and W. Lu, "Programmable resistance switching in nanoscale two-terminal devices," *Nano Letters*, vol. 9, no. 1, pp. 496–500, 2009.

[102] ——, "High-density crossbar arrays based on a Si memristive system," *Nano Letters*, vol. 9, no. 2, pp. 870–874, 2009.

[103] H. Lee, P. Chen, T. Y. Wu, Y. Chen, C. Wang, P. Tzeng, C. H. Lin, F. Chen, C. Lien, and M. J. Tsai, "Low power and high speed bipolar switching with a thin reactive Ti buffer layer in robust HfO2 based RRAM," in *IEEE International Electron Devices Meeting IEDM*, 2008, pp. 1–4.

[104] S. H. Jo and W. Lu, "CMOS compatible nanoscale nonvolatile resistance switching memory," *Nano Letters*, vol. 8, no. 2, pp. 392–397, 2008.

[105] L. Cario, C. Vaju, B. Corraze, V. Guiot, and E. Janod, "Electric-field-induced resistive switching in a family of mott insulators: Towards a new class of RRAM memories," *Advanced Materials*, vol. 22, no. 45, pp. 5193–5197, 2010.

[106] H. Y. Jeong, J. Y. Kim, J. W. Kim, J. O. Hwang, J.-E. Kim, J. Y. Lee, T. H. Yoon, B. J. Cho, S. O. Kim, R. S. Ruoff, and S.-Y. Choi, "Graphene oxide thin films for flexible nonvolatile memory applications," *Nano Letters*, vol. 10, no. 11, pp. 4381–4386, 2010.

[107] C. Yoshida, K. Tsunoda, H. Noshiro, and Y. Sugiyama, "High speed resistive switching in Pt/TiO2/TiN film for nonvolatile memory application," *Applied Physics Letters*, vol. 91, no. 22, pp. 223 510:1–3, 2007.

[108] S.-Y. Wang, C.-W. Huang, D.-Y. Lee, T.-Y. Tseng, and T.-C. Chang, "Multilevel resistive switching in Ti/Cu_xO/Pt memory devices," *Journal of Applied Physics*, vol. 108, no. 11, pp. 114 110:1–6, 2010.

[109] G. Snider, "Spike-timing-dependent learning in memristive nanodevices," in *IEEE International Symposium on Nanoscale Architectures NANOARCH*, 2008, pp. 85–92.

[110] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Letters*, vol. 10, no. 4, pp. 1297–1301, 2010.

[111] F. Alibart, S. Pleutin, D. Guérin, C. Novembre, S. Lenfant, K. Lmimouni, C. Gamrat, and D. Vuillaume, "An organic nanoparticle transistor behaving as a biological spiking synapse," *Advanced Functional Materials*, vol. 20, no. 2, pp. 330–337, 2010.

[112] T. Hasegawa, T. Ohno, K. Terabe, T. Tsuruoka, T. Nakayama, J. K. Gimzewski, and M. Aono, "Learning abilities achieved by a single solid-state atomic switch," *Advanced Materials*, vol. 22, no. 16, pp. 1831–1834, 2010.

[113] R. Cavin and V. Zhirnov, "Future devices for information processing," in *Proceedings of the 31st European Solid-State Circuits Conference ESSCIRC*, 2005, pp. 7–12.

[114] R.Cavin, V. Zhirnov, D. Herr, A. Avila, and J. Hutchby, "Research directions and challenges in nanoelectronics," *Journal of Nanoparticle Research*, vol. 8, pp. 841–858, 2006.

[115] C. P. Collier, E. W. Wong, M. Belohradský, F. M. Raymo, J. F. Stoddart, P. J. Kuekes, R. S. Williams, and J. R. Heath, "Electronically configurable molecular-based logic gates," *Science*, vol. 285, no. 5426, pp. 391–394, 1999.

[116] T. Rueckes, K. Kim, E. Joselevich, G. Y. Tseng, C.-L. Cheung, and C. M. Lieber, "Carbon nanotube-based nonvolatile random access memory for molecular computing," *Science*, vol. 289, no. 5476, pp. 94–97, 2000.

[117] Y. Huang, X. Duan, Y. Cui, L. J. Lauhon, K.-H. Kim, and C. M. Lieber, "Logic gates and computation from assembled nanowire building blocks," *Science*, vol. 294, no. 5545, pp. 1313–1317, 2001.

[118] M. Stan, P. Franzon, S. Goldstein, J. Lach, and M. Ziegler, "Molecular electronics: from devices and interconnect to circuits and architecture," *Proceedings of the IEEE*, vol. 91, no. 11, pp. 1940–1957, 2003.

[119] J. R. Heath and M. A. Ratner, "Molecular electronics," *Physics Today*, vol. 56, pp. 43–49, 2003.

[120] M. A. Reed, C. Zhou, C. J. Muller, T. P. Burgin, and J. M. Tour, "Conductance of a molecular junction," *Science*, vol. 278, no. 5336, pp. 252–254, 1997.

[121] S. Kaeriyama, T. Sakamoto, H. Sunamura, M. Mizuno, H. Kawaura, T. Hasegawa, K. Terabe, T. Nakayama, and M. Aono, "A nonvolatile programmable solid-electrolyte nanometer switch," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 01, pp. 168–176, 2005.

[122] Z. J. Donhauser, B. A. Mantooth, K. F. Kelly, L. A. Bumm, J. D. Monnell, J. J. Stapleton, D. W. Price, A. M. Rawlett, D. L. Allara, J. M. Tour, and P. S. Weiss, "Conductance switching in single molecules through conformational changes," *Science*, vol. 292, no. 5525, pp. 2303–2307, 2001.

[123] C. N. Lau, D. R. Stewart, R. S. Williams, and M. Bockrath, "Direct observation of nanoscale switching centers in metal/molecule/metal structures," *Nano Letters*, vol. 4, no. 4, pp. 569–572, 2004.

[124] J. R. Heath, P. J. Kuekes, G. S. Snider, and R. S. Williams, "A defect-tolerant computer architecture: Opportunities for nanotechnology," *Science*, vol. 280, no. 5370, pp. 1716–1721, 1998.

[125] A. DeHon, "Array-based architecture for FET-based, nanoscale electronics," *IEEE Transactions on Nanotechnology*, vol. 2, no. 1, pp. 23–32, 2003.

[126] G. S. Snider, "Self-organized computation with unreliable, memristive nanodevices," *Nanotechnology*, vol. 18, no. 36, pp. 365 202:1–13, 2007.

[127] S. Y. Chou, P. R. Krauss, and P. J. Renstrom, "Imprint lithography with 25-nanometer resolution," *Science*, vol. 272, no. 5258, pp. 85–87, 1996.

[128] P. J. Kuekes, D. R. Stewart, and R. S. Williams, "The crossbar latch: Logic value storage, restoration, and inversion in crossbar circuits," *Journal of Applied Physics*, vol. 97, no. 3, pp. 034 301:1–5, 2005.

[129] J. Borghetti, Z. Li, J. Straznicky, X. Li, D. A. A. Ohlberg, W. Wu, D. R. Stewart, and R. S. Williams, "A hybrid nanomemristor/transistor logic circuit capable of self-programming," *Proceedings of the National Academy of Sciences*, vol. 106, no. 6, pp. 1699–1703, 2009.

[130] D. Strukov and K. Likharev, "Reconfigurable hybrid CMOS/nanodevice circuits for image processing," *IEEE Transactions on Nanotechnology*, vol. 6, no. 6, pp. 696–710, 2007.

[131] G. Snider, "Computing with hysteretic resistor crossbars," *Applied Physics A*, vol. 80, no. 6, pp. 1165–1172, 2005.

[132] K. K. Likharev, A. Mayr, I. Muckra, and O. Turel, "Cross nets: High-performance neuromorphic architectures for CMOL circuits," *Annals of the New York Academy of Sciences*, vol. 1006, no. 1, pp. 146–163, 2003.

[133] A. Flocke and T. Noll, "Fundamental analysis of resistive nano-crossbars for the use in hybrid Nano/CMOS-memory," in *33rd European Solid State Circuits Conference ESSCIRC*, 2007, pp. 328–331.

[134] M. Leslie and R. Jacob Baker, "Noise-shaping sense amplifier for mram cross-point arrays," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 699–704, 2006.

[135] E. Linn, R. Rosezin, C. Kügeler, and R. Waser, "Complementary resistive switches for passive nanocrossbar memories," *Nature Materials*, vol. 9, pp. 403–406, 2010.

[136] E. Katsia, N. Huby, G. Tallarida, B. Kutrzeba-Kotowska, M. Perego, S. Ferrari, F. C. Krebs, E. Guziewicz, M. Godlewski, V. Osinniy, and G. Luka, "Poly(3-hexylthiophene)/ZnO hybrid pn junctions for microelectronics applications," *Applied Physics Letters*, vol. 94, no. 14, pp. 143 501–1–3, 2009.

[137] M. Sven, P. Craig, J. Warren, T. Carl, and F. S. R., "A polymer/semiconductor write-once read-many-times memory," *Nature*, vol. 426, pp. 166–169, 2003.

[138] B. S. Kang, S.-E. Ahn, M.-J. Lee, G. Stefanovich, K. H. Kim, W. X. Xianyu, C. B. Lee, Y. Park, I. G. Baek, and B. H. Park, "High-current-density CuO_x/$InZnO_x$ thin-film diodes for cross-point memory applications," *Advanced Materials*, vol. 20, no. 16, pp. 3066–3069, 2008.

[139] C. Nauenheim, C. Kugeler, A. Rudiger, R. Waser, A. Flocke, and T. Noll, "Nano-crossbar arrays for nonvolatile resistive ram (RRAM) applications," in *8th IEEE Conference on Nanotechnology, 2008. NANO '08.*, 2008, pp. 464–467.

[140] J. J. Yang, J. Borghetti, D. Murphy, D. R. Stewart, and R. S. Williams, "A family of electronically reconfigurable nanodevices," *Advanced Materials*, vol. 21, no. 37, pp. 3754–3758, 2009.

[141] L. Emanuel, J. Ciszek, J. T. Heike, and Riel, "Reversible and controllable switching of a single-molecule junction," *Small*, vol. 2, no. 8-9, pp. 973–977, 2006.

[142] G. Csaba and P. Lugli, "Read-out design rules for molecular crossbar architectures," *IEEE Transactions on Nanotechnology*, vol. 8, no. 3, pp. 369–374, 2009.

[143] D. B. Strukov and K. K. Likharev, "Defect-tolerant architectures for nanoelectronic crossbar memories," *Journal of Nanoscience and Nanotechnology*, vol. 7, no. 1, pp. 151–167, 2007.

[144] Y. Yang, J. Mathew, M. Ottavi, S. Pontarelli, and D. Pradhan, "Novel complementary resistive switch crossbar memory write and read schemes," *IEEE Transactions on Nanotechnology*, vol. 14, no. 2, pp. 346–357, 2015.

[145] S.-J. Ham, H.-S. Mo, and K. sik Min, "Low-powerower $V_{DD}/3$ write scheme with inversion coding circuit for complementary memristor array," *IEEE Transactions on Nanotechnology*, vol. 12, no. 5, pp. 851–857, 2013.

[146] O. Kavehei, S. Al-Sarawi, K.-R. Cho, K. Eshraghian, and D. Abbott, "An analytical approach for memristive nanoarchitectures," *IEEE Transactions on Nanotechnology*, vol. 11, no. 2, pp. 374–385, 2012.

[147] I. E. Ebong and P. Mazumder, "Self-controlled writing and erasing in a memristor crossbar memory," *IEEE Transactions on Nanotechnology*, vol. 10, no. 6, pp. 1454–1463, 2011.

[148] S. Kvatinsky, G. Satat, N. Wald, E. Friedman, A. Kolodny, and U. Weiser, "Memristor-based material implication (IMPLY) logic: Design principles and methodologies," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 10, pp. 2054–2066, 2014.

[149] E. Lehtonen and M. Laiho, "Stateful implication logic with memristors," in *IEEE/ACM International Symposium on Nanoscale Architectures NANOARCH '09.*, 2009, pp. 33–36.

[150] X. Dong, C. Xu, Y. Xie, and N. Jouppi, "NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 994–1007, 2012.

[151] A. Mazady and M. Anwar, "Memristor: Part II- DC, transient, and RF analysis," *IEEE Transactions on Electron Devices*, vol. 61, no. 4, pp. 1062–1070, 2014.

[152] L. Zhang, Z. Chen, J. J. Yang, B. Wysocki, N. McDonald, and Y. Chen, "A compact modeling of TiO_2-TiO_{2-x} memristor," *Applied Physics Letters*, vol. 102, no. 15, pp. 153 503:1–4, 2013.

[153] H. Owlia, P. Keshavarzi, and A. Rezai, "A novel digital logic implementation approach on nanocrossbar arrays using memristor-based multiplexers," *Microelectronics Journal*, vol. 45, no. 6, pp. 597 – 603, 2014.

[154] M. D. Pickett, D. B. Strukov, J. L. Borghetti, J. J. Yang, G. S. Snider, D. R. Stewart, and R. S. Williams, "Switching dynamics in titanium dioxide memristive devices," *Journal of Applied Physics*, vol. 106, no. 7, pp. 074 508:1–6, 2009.

[155] J. G. Simmons, "Electric tunnel effect between dissimilar electrodes separated by a thin insulating film," *Journal of Applied Physics*, vol. 34, no. 9, pp. 2581–2590, 1963.

[156] F. Corinto, A. Ascoli, and M. Gilli, "Symmetric charge-flux nonlinearity with combined inherently-asymmetric memristors," in *20th European Conference on Circuit Theory and Design (ECCTD)*, 2011, pp. 632–635.

[157] H. (Helen) Li and M. Hu, "Compact model of memristors and its application in computing systems," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2010, pp. 673–678.

[158] H. Abdalla and M. Pickett, "SPICE modeling of memristors," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2011, pp. 1832–1835.

[159] E. Lehtonen and M. Laiho, "CNN using memristors for neighborhood connections," in *12th International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA)*, 2010, pp. 1–4.

[160] D. Batas and H. Fiedler, "A memristor SPICE implementation and a new approach for magnetic flux-controlled memristor modeling," *IEEE Transactions on Nanotechnology*, vol. 10, no. 2, pp. 250–255, 2011.

[161] Z. Biolek, D. Biolek, and V. Biolkova, "SPICE model of memristor with nonlinear dopant drift," *Radioengineering*, vol. 18, no. 2, pp. 210–214, 2009.

[162] A. Rak and G. Cserey, "Macromodeling of the memristor in SPICE," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 4, pp. 632–636, 2010.

[163] S. Benderli and T. Wey, "On SPICE macromodelling of TiO_2 memristors," *Electronics Letters*, vol. 45, no. 7, pp. 377–379, 2009.

[164] H. Kim, M. Sah, C. Yang, S. Cho, and L. Chua, "Memristor emulator for memristor circuit applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 10, pp. 2422–2431, 2012.

[165] Y. Pershin and M. Di Ventra, "Practical approach to programmable analog circuits with memristors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 8, pp. 1857–1864, 2010.

[166] Y. N. Joglekar and S. J. Wolf, "The elusive memristor: Properties of basic electrical circuits," *European Journal of Physics*, vol. 30, pp. 661–675, 2009.

[167] F. Corinto and A. Ascoli, "A boundary condition-based approach to the modeling of memristor nanostructures," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 11, pp. 2713–2726, 2012.

[168] T. Prodromakis, B. P. Peh, C. Papavassiliou, and C. Toumazou, "A versatile memristor model with nonlinear dopant kinetics," *IEEE Transactions on Electron Devices*, vol. 58, no. 9, pp. 3099–3105, 2011.

[169] K. Kim, S. Shin, and S.-M. Kang, "Stateful logic pipeline architecture," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2011, pp. 2497–2500.

[170] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nature Nanotechnology*, vol. 8, no. 1, pp. 13–24, 2013.

Publication Based on Present Work

Journal publications:

1. Pravin Mane, Nishil Talati, Ameya Riswadkar, Ramesh Raghu, and C.K. Ramesha, ”Stateful-NOR based reconfigurable architecture for logic implementation”, Microelectronics Journal, Volume 46, Issue 6, pp. 551-562, 2015.

2. Pravin Mane, Nishil Talati, Ameya Riswadkar, Ramesh Raghu, and C.K. Ramesha, ”Reconfiguration on nanocrossbar using material implication”, Sadhana - Academy Proceedings in Engineering Science (Under review- minor revision submitted).

3. Pravin Mane, Sudeep Mishra, Ravish Deliwala, and C. K. Ramesha, ”Reconfiguration in resistive switching crossbar”, Microelectronics Journal, under review.

Conference publications:

1. Pravin Mane, Nishil Talati, Ameya Riswadkar, Bhavan Jasani, and C.K. Ramesha, ”Implementation of NOR logic based on material implication on CMOL FPGA architecture,” 28th International Conference on VLSI Design, Bangalore, pp. 523-528, 2015.

2. Pravin Mane, Nishil Talati, Ameya Riswadkar, Ramesh Raghu, and C.K. Ramesha, ”Implicating logic functions with memristors,” 2014 International SoC Design Conference (ISOCC), Jeju, pp. 232-233, 2014.

3. Pravin Mane, Namita Paul, Nikhilesh Behera, Madankumar Sampath, and C. K. Ramesha, ”Hybrid CMOS - memristor based configurable logic block design,” 2014 International Conference on Electronics and Communication Systems (ICECS), Coimbatore, pp. 1-5, 2014.

4. Madankumar Sampath, Pravin Mane, and C. K. Ramesha, ”Hybrid CMOS-memristor based FPGA architecture,” 2015 International Conference on VLSI Systems, Architecture, Technology and Applications (VLSI-SATA), pp. 1-6, Bangalore, 2015.

Brief Biography of the Candidate

Mane Pravin Sakharam received Bachelor of Engineering degree in the Electronics Engineering discipline from Rajarambapu Institute of Technology, Sakharale, Maharashtra, India, in 1998. He obtained his Master of Technology degree in the System Engineering & Operations Research from Indian Institute of Technology, Roorkee, Uttarakhand, India in 2006.

He has worked as Lecturer in Electronics department of Rajaram Shinde College of Engineering, Chiplun, Maharashtra, India from August 1999 to July 2004. After completing, M. Tech. in 2006, he has worked as Assistant Professor in Mody Institute of Technology and Science, Lakshmangarh (Sikar), Rajasthan, India from August 2006 to July 2007 and in Vidyalankar Institute of Technology, Wadala, Mumbai, Maharashtra, India from July 2007 to December 2008. He is currently working as a Lecturer in Electrical & Electronics Engineering department since January 2009.

He has conducted 5 days workshop on ”Cadence IC615 Analog & Digital Flow” under faculty development program in Electronics & Telecommunication department of Sir Visvesvaraya Institute of Technology, Nashik (Affiliated to University of Pune) in June 2014. He has been carrying out research in the area of reconfigurable architectures using novel devices and in-memory calculations in Resistive RAM and published papers in international journals and conferences.

Brief Biography of the Supervisor

Ramesha C K received the M.Sc degree from Mangalore University, Konaje, Mangalore in 1990 and Ph.D degree in Electronics from University of Mysore, Mysore, India in 2007. He has published more than 11 articles in referred journals, and has been author or co-author of over 15 conference papers. He is also actively involved in Major Research Projects sponsored by UGC and DST, New Delhi, India. His current research interests are (i) Ohmic and Schottky Contacts to wide-band gap semiconductors, (ii) Surface analysis of III-V and II-VI semiconductors, Low power VLSI design, RF Antenna design and cognitive radio.