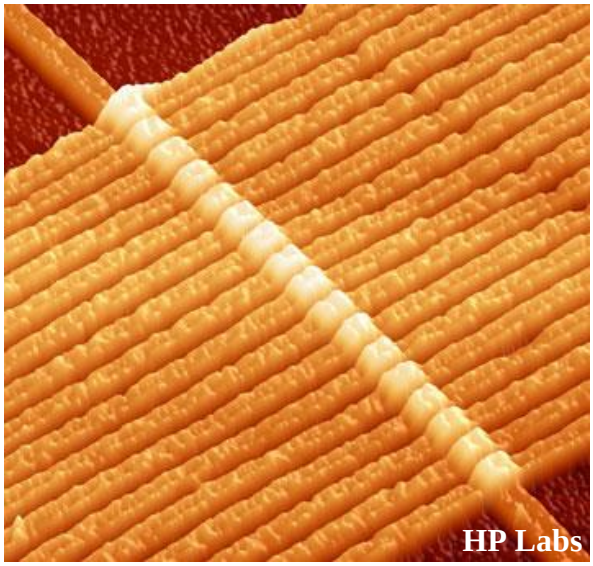
Imperial College London

Department of Electrical and Electronic Engineering

Final Year Project Report 2016



Project Title:	**Practical Analysis and Comparison of Memresistor Based Circuits**
Student:	**V.S. Gami**
CID:	**00739377**
Course:	**4T**
Project Supervisor:	**Dr Christos Papavassiliou**
Second Marker:	**Dr Esther Rodriguez-Villegas**

Abstract

This project involves the analysis and comparison of various memristor based logic families via simulations. Numerous mathematical memristor models are investigated and a modified version of the Voltage ThrEshold Adaptive Memristor model (VTEAM) is implemented in PSpice. This model is fitted to practical TiO_2 memristor data and used for simulations of three promising memristor based logic families: Memristor Ratioed Logic (MRL), Implication Logic and Memristor Aided LoGIC (MAGIC). Advantages, disadvantages and limitations are presented based on simulation results and research analysis. Simulations reveal significant limitations and restrictions of some logic families for the case of devices with voltage thresholds. Finally, design and test of an instrumentation circuit intended for investigating memristor switching behaviour is presented.

Acknowledgments

I would like to express my sincere appreciation to Dr. Christos Papavassiliou for his invaluable support and expertise throughout this project. He truly is an inspiring source of motivation whose passion and enthusiasm for memristor technologies is unparalleled, something which sparked my interest in these unique devices. I would also like to thank Radu Berdan for providing experimental memristor data and advice. Last but not least, I express my appreciation for my family and friends for their unwavering support throughout my studies.

Table of Contents

Symbols and Abbreviations

$CMOS$	Complementary Metal Oxide Semiconductor
MRL	Memristor Ratioed Logic
$MAGIC$	Memristor Aided LoGIC
PLMA	Programmable Logic Memristor Array
$VTEAM$	Voltage ThrEshold Adaptive Memristor model
$TEAM$	ThrEshold Adaptive Memristor model
BCM	Boundary Condition Memristor model
DAQ	Data Acquisition Device
VI	Virtual Instrument

R_{on}	Memristor 'on' state (low) resistance. Represents logic level 1.
R_{off}	Memristor 'off' state (high) resistance. Represents logic level 0.
v_{on}	Voltage threshold for 'on' switching
v_{off}	Voltage threshold for 'off' switching
i_{on}	Current threshold for 'on' switching
i_{off}	Current threshold for 'off' switching
v_z	Denotes high impedance output mode of voltage driver used for Implication and MAGIC
v_{set}	The negative voltage applied across the memristor to set the state to logic 1.
v_{clear}	The positive voltage applied across the memristor to clear the state to logic 0.
v_{cond}	Voltage level used by the Implication family during the $IMPLY$ operation. These are all related as follows: $v_{set} < v_{on} < v_{cond} < 0 < v_{off} < v_{clear}$
v_{high}	Denotes the voltage level used for logic high for the MRL gates and PLA. $0V$ is used for logic 0.
v_0	The positive voltage applied to MAGIC gates during operation. The subscript '0' can be replaced by the gate type to denote the voltage for a particular gate, e.g. v_{nor}, v_{nand}.

1 Introduction

1.1 Motivation

Over the past 50 years or so, there have been vast improvements in computing in terms of processing power, memory, power efficiency and cost. This has been a substantial driving factor for worldwide advancements in a wide range of fields such as mobile communications, entertainment, education, research, industry and healthcare to name a few. From an economic perspective, it is difficult to overlook the strong correlation between past levels of investment in information and technology and in increases in human welfare and total factor productivity. Intuitively most people would expect there to be a deeper connection rather than a simple correlation. In fact studies have concluded that it is almost irrefutable fact that improved computing technology causes increasing productivity. [1]

These improvements in computing are direct consequence of the steady miniaturization of devices as stated Moore's law back in 1965. This stated that the number of transistors per affordable chip would double approximately every 2 years for the next decade or so. Although Moore's law was initially made as an observation and a forecast, it eventually transformed into a target for industry and hence a self-fulfilling prophecy. However, this miniaturization cannot be sustained indefinitely and the general consensus is that CMOS transistors will approach fundamental physical limits in the coming decade. [2] As a consequence, other technologies such as memristors must be investigated for implementation of logic and memory.

An ideal memristor, as originally hypothesized by Chua in 1971, is a two terminal electronic device whose memristance (instantaneous resistance) depends on the time integral of the current that passed through it in the entire past. Another interpretation is that the memristance depends on the time integral of the voltage, i.e. flux. [3] Chua proved that the memristor could not be synthesized by any combination of resistance, capacitance or inductance, hence justified the claim that the memristor is truly fundamental.

In 2008, a research group at HP Labs lead by Stanley Williams successfully fabricated a nanoscale memristive device consisting of titanium dioxide sandwiched between two platinum contacts. [4] [5] This particular practical realisation of the fabled memristor reignited the interest of the electronics industry and scientific community. [6] As a result, there has been a flurry of proposed applications over the past few years, some of which are illustrated by Figure 1 below. [7]

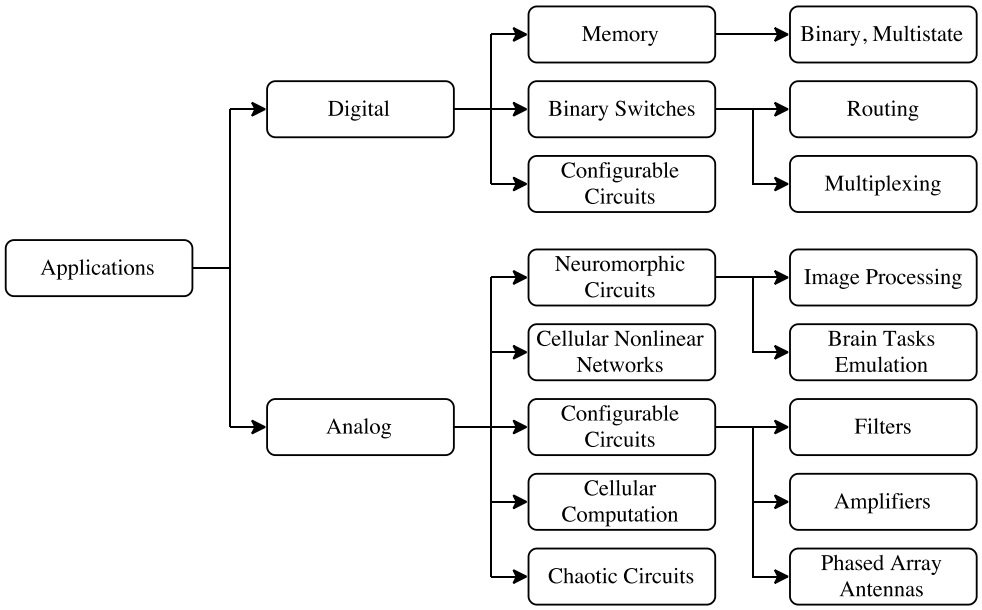


Figure 1: Possible memristor applications. Analog applications utilise the continuous range of memristances whereas digital applications utilise a discrete set.

This project focuses on binary memristor based logic. The motivation for use of memristors in logic and memory stems from the fundamental advantages they have the potential to offer:

1. Memristors offer good scalability with some current devices already down to a few nanometers. [8] [9] In contrast, scaling down transistors further is getting increasingly difficult. Functional $7nm$ transistors have been built and constructed in 2015, but are not yet commercially viable. [10] As of 2016, Intel and other fabs are currently producing $14nm$ devices with $10nm$ still under commercial development. Beyond $10nm$, the sensitivity of various parameters such as voltage threshold to fabrication spreads increases exponentially. This means that the gate dimensions need to accurate to a few angstroms and this is beyond the projections of industry according to the International Technology Roadmap for Semiconductors. [11] [12]
2. As transistors scale down their leakage current increases. A large portion of power consumption in a modern commercial processor is can be due to leakage current. Parts of the processor are actually powered down when not in use to counteract this. In contrast a memristors performance improves with scaling and there is possibility for zero static power consumption for some logic families.
3. Some memristors can switch states with low energy (picojoules or less). [13] [14] [15] [16]
4. Some devices have been reported with nanosecond or even lower $(100ps)$ switching times [17] [16] [18] [19]
5. They are non-volatile thus providing greater resilience to power interruptions, and possibility of almost zero boot times for computers. [20] [21] [22] Note that this is not characteristic of all practical memristors. For some devices, the state decays in a time frame which ranges from milliseconds to weeks depending on the type.
6. Some can be compatible with CMOS technology both in terms of voltage levels and the production process. They can be produced stacked above the CMOS layer, since they are formed between two adjacent metal layers as illustrated by Figure 2 below.

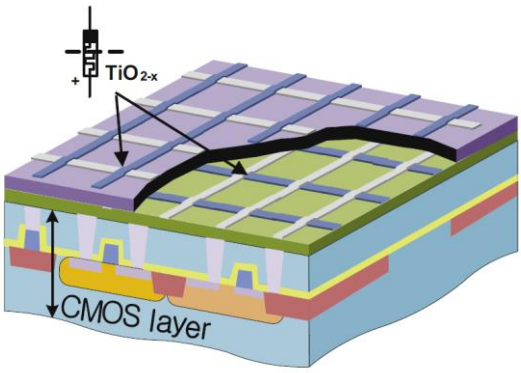


Figure 2: Illustrating a hybrid CMOS TiO_2 memristor structure. [23]

There are of course numerous challenges with memristor based digital logic. Even under the assumption that devices can be produced according to specification; there are still fundamental obstacle. Firstly, they are two terminal passive devices and so they differ greatly from transistors. This means new logic architectures must be explored. Secondly, since they are resistive, there will be associated I^2R losses. Finally, memristor based gates cannot power subsequent circuitry since they are passive. This is why hybrid CMOS architectures are of interest.

1.2 Project Aims and Objectives

The aim of this project is to investigate the ability for memristors to perform binary digital logic. To date, various logic families have been investigated through both theoretical analysis and simulation. However, there exist significant gaps in research.

For example, models with voltage thresholds have not fully been investigated even though many practical devices exhibit voltage threshold like behaviour. Additionally, little research has been published on logic simulations which use memristor models which have been fitted to practical data. Although many papers show promising simulation results for some logic families in terms of speed, power and chip area, it is difficult to accept them due to lack of physical meaning, i.e. the models parameters used could represent a device which may not exist now or even in the future. [24] [25] [26] [27] Note that this is not always the case, for example, Kyoungrok et al. use a fitted model to investigate hybrid CMOS architectures. However, there is scope to expand on this by investigating other families and logic circuits. [23] Finally, little to no research has been published on practical implementations of non-trivial logic, and this type of investigation is one of the main goals of this project. This is because although simulations are useful, they are limited for three main reasons:

1. No perfectly accurate model exists since the precise physics and dynamics of memristors are not yet fully understood.
2. The most accurate models are computationally expensive meaning accurate and long simulations of complex designs are not feasible.
3. Simulation can never fully capture real world physics due to assumptions and simplifications. For example variability between devices, or parasitic capacitances due to neighbouring tracks, or temperature effects.

The ultimate goal for this project is to design and implement a non-trivial logic circuit using multiple gates such as an half adder; something which has never been done before to the writers best knowledge. To achieve this, various logic families first need to be investigated, and basic logic gates need to be analysed. Simulation will be used to broaden understanding and help with the practical implementation so various models will be also investigated and used for simulation based analysis. The simulations in this project use a new model fitted to practical TiO_2 memristor data. This process will help develop an understanding of the advantages, disadvantages and limitations of various memristor based logic families.

1.3 Challenges and Limitations

The main limitation which resulted in the failure of the ultimate goal of this project was the unavailability of the devices. Although many titanium dioxide memristive devices were available on a wafer, inability to construct the probing card and unsuccessful bonding attempts meant the practical investigation and implementation of this project was limited. Nevertheless, the practical work done in anticipation of devices (e.g. preparation and test of instrumentation circuits) is presented section 7 and this can be useful for future work if this project is to be continued.

1.4 Contributions

The most interesting outcomes of this project are listed below.

1. Developed design guidelines for the MAGIC family for current threshold devices and identified fundamental limitations which limit the feasibility of such logic. (Section 3.5)
2. Spice implementation and modification of VTEAM [28] and TEAM [26] models. (Section 5)
3. An optimised hybrid CMOS XOR gate is proposed with reduced power consumption and device count. This is then used to successfully simulate a full adder. (Section 6.2.3.6)

4. A comprehensive comparison of MRL, MAGIC and Implication Logic is compiled based on research and simulation results. Simulations revealed limitations of each family for devices with voltage thresholds. (Section 6)
5. An optimised Implication Logic full adder sequence is proposed. This has reduced memristor and step count compared to the next best alternative as proposed by Teimoory et al. [29] Section 6.3.2

1.5 Report Outline

This report opens with the background section which more formally introduces memristors and memristive systems. A brief background of practical devices, in particular HP's popular TiO_2 device, is then presented.

The next section introduces and analyses various logic families. Three of these are the most interesting and are selected for further investigation via simulations. Design guidelines are developed based on analysis of the fundamental operating principles and some inherent shortcomings of the MAGIC family for current threshold devices are identified.

Sections 4 and 5 concern the modelling aspect of memristors. Various mathematical models are investigated and the VTEAM one is selected for PSpice implementation. Slight modifications are made to increase fitting accuracy based on observations of practical memristor data. Fitting and evaluation is then performed and the model is used for simulations in the next section.

Section 6 attempts design of a full adder circuit for each of the logic families in order to quantitatively compare power, speed and area. However, it turns out MAGIC and Implication Logic families are unsuccessful with the fitted model since they require memristors with different properties to operate. For example, Implication cannot work with a voltage threshold device since there is inadequate switching. Therefore an in depth qualitative comparison and evaluation is performed instead based on information gathered during research and simulation. Additionally, a list of preferred memristors for each family is compiled.

Section 7 presents the practical work done in anticipation of TiO_2 devices. A simple instrumentation circuit and two LabVIEW VI's are designed and tested. The final sections consist of the project evaluation and conclusion along with future work which would improve the project.

2 Memristor Background

2.1 Ideal Memristor Origin and Definition

There are four fundamental circuit variables according to elementary circuit theory: voltage (v), flux linkage (φ), current (i), and charge (q). [30] There are ${}^4C_2 = 6$ binary relations between these variables as graphically illustrated by Figure 3 below. Two of these relations are described by the physics definitions as specified by equations 2.1 below:

$$i(t) = \frac{d}{dt}q(t), \qquad v(t) = \frac{d}{dt}\varphi(t) \tag{2.1}$$

Equations 2.2 to 2.4 below show three more relations in their most general implicit form. From each of these relations, the three well-known fundamental circuit elements can be defined as derivatives; namely resistance (R), capacitance (C), and inductance (L). Note that to a first approximation these relations are practically linear, therefore R, C and L are constant.

$$f_R(v,i) = 0, \qquad for\ which\ we\ define \qquad R = \frac{dv}{di} \tag{2.2}$$

$$f_C(q,v) = 0, \qquad for\ which\ we\ define \qquad C = \frac{dq}{dv} \tag{2.3}$$

$$f_L(\varphi,i) = 0, \qquad for\ which\ we\ define \qquad L = \frac{d\varphi}{di} \tag{2.4}$$

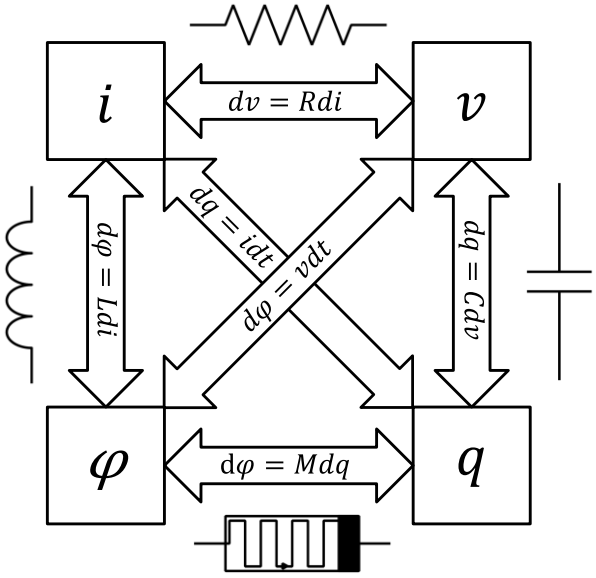


Figure 3: Four fundamental circuit variables and their 6 binary relations.
The bottommost symbol denotes the memristor. Concept modified from [4]

Based on the symmetry of Figure 3, Chua [3] postulated the existence of another fundamental circuit element which relates charge and flux linkage as shown by equation 2.5 below.

$$f_M(\varphi,q) = 0, \qquad for\ which\ we\ define \qquad M = \frac{d\varphi}{dq} \tag{2.5}$$

Before developing a more relatable physical interpretation of equation 2.5 which involves voltages and currents rather than flux and charge, it is useful to first define what is meant by an '*ideal'* memristor. A memristor is considered ideal if its flux-charge curve (the plot of flux on y-axis and charge on x-axis) satisfies the following requirements: [3]

1. *Single valued*
2. *Nonlinear*
3. *Strictly monotonically increasing*
4. *Is of class C^1 (Differentiable with a continuous derivative)*

2.6

The flux-charge relation (eqn 2.5) is of upmost importance in characterising a device. If the flux-charge curve for a particular (ideal) memristor is accurately known, then there would be no modelling issues and a completely accurate model could be created even with no knowledge of the underlying physics. Much like a nonlinear capacitor can be fully characterised by its charge-voltage curve, a nonlinear ideal memristor can be uniquely and fully described by its flux-charge curve. However, in reality a practical device is unlikely to be ideal.

A memristor is said to be flux controlled or charge controlled if the implicit relation presented in 2.5 can be written explicitly as a function of flux $\left(q = f_q(\varphi)\right)$ or charge $\left(\varphi = f_\varphi(q)\right)$ respectively. [3] For an ideal memristor, these are just inverses of each other. In the following analysis, only charge controlled memristors are considered for simplicity, but the analysis is similar for flux controlled memristors. Differentiating $\varphi = f_\varphi(q)$ with respect to time via the chain rule results in:

$$\frac{d\varphi}{dt} = \left[\frac{d}{dq} f_\varphi(q)\right]\frac{dq}{dt} \tag{2.7}$$

Substituting in the physics definitions from 2.1 into 2.7 results in 2.8:

$$v(t) = \left[\frac{d}{dq} f_\varphi(q)\right] i(t), \qquad or \qquad v(t) = [\mathcal{M}(q)]i(t) \tag{2.8}$$

In equation 2.8, $\mathcal{M}(q)$ denotes the memristance, or instantaneous resistance, and it is a function of the time integral of the current through the memristor (or the total charge that has passed through the memristor in the past). An interesting observation is that for a useful memristor the flux-charge relation must be nonlinear. If the flux-charge curve is linear, then $\mathcal{M}(q)$ is a constant and there is no difference between memristance and resistance. This is the reason for the nonlinear flux-charge relation requirement for an ideal memristor as mentioned earlier in 2.6. A quadratic relation would be nice since this means memristance is directly proportional to the time integral of the current.

Although this representation of a memristor is friendlier since current and voltages are quantities which can be more easily measured as compared to charge and flux, it is worth noting that a nonlinear memristor cannot be fully characterised by a single IV curve since its shape depends on the input waveform and frequency. This is analogous to a capacitor or inductor. To demonstrate this concept an ideal memristor is simulated in MATLAB with its characteristic flux-charge relation given by $f_\varphi(q) = q^2$. Figure 4 blow shows four IV curves generated by sinusoidal current input of different frequencies. It shows that as the driving frequency increases, the pinched hysteresis shape reduces and the IV curve tends to that of an ideal normal resistor.

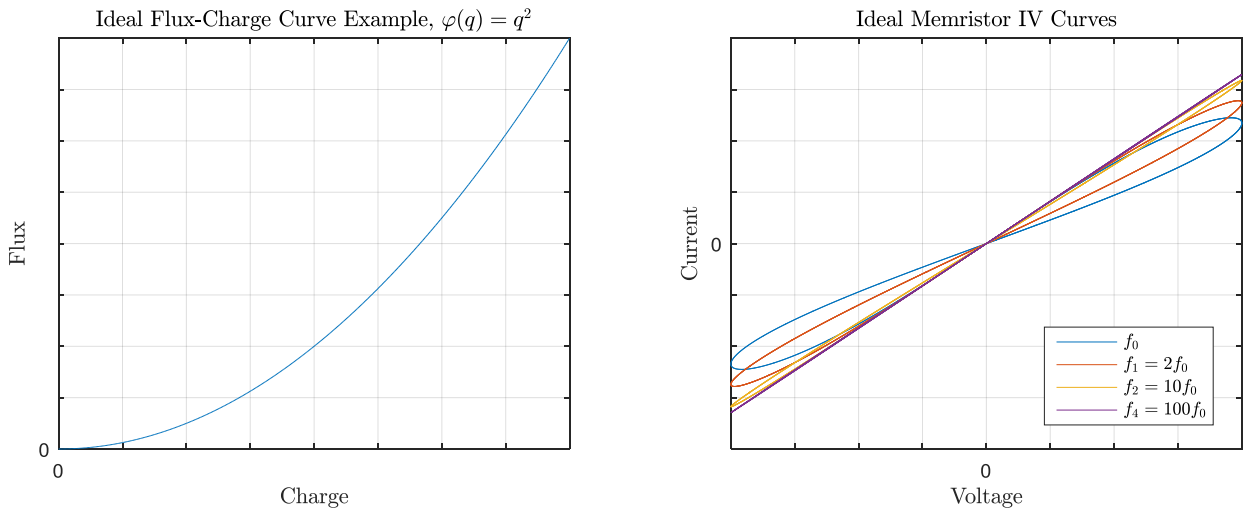


Figure 4: *IV* curves obtained with sinusoidal current input for an ideal memristor as described by $\varphi(q) = q^2$. As the frequency increases, the memristor begins to look like a normal resistor.

Having translated the original memristor definition (eqn 2.5) into more relatable quantities involving current and voltage (eqn 2.8), it makes sense to do the same with the definitions in laid out by 2.6. The following points translate the properties which define an ideal memristor into more relatable properties regarding the *IV* curves. [3]

1) If the flux-charge curve is single valued, then the *IV* curve is symmetric for a symmetric input waveform. [31]
2) If the flux-charge curve is nonlinear, then the memristor is not the same as a constant resistor and the *IV* curve has a pinched hysteresis shape.
3) If the flux-charge curve is strictly monotonically increasing, then $M(q) > 0$. This means the memristor is passive meaning it only dissipates energy since power is given by $p(t) = i^2(t)M(q(t))$. The *IV* curve is then restricted to the top right and bottom left quadrants of the *IV* plane for any excitation.
4) If the flux-charge curve is of class C^1, then $M(q)$ is well defined and finite and hence $i = 0 \iff v = 0$, i.e. the *IV* curve always crosses the origin. This implies memristance to be non-volatile and that no energy is stored. A storage device like a capacitor or inductor would have a circular shaped *IV* curve which encloses the origin.

The first point mentioned above is particularly interesting since most fabricated devices do not have a symmetric *IV* curve for a symmetric input. Figure 5 below examples the effect of a non-single valued flux-charge curve.

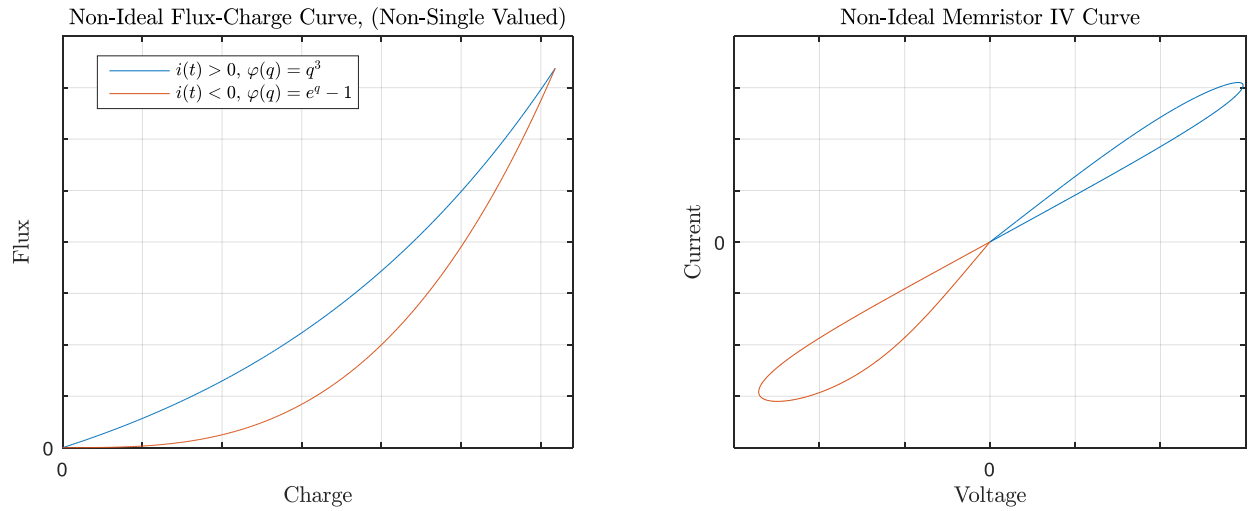


Figure 5: Non-ideal flux-charge curve and corresponding memristor *IV* curve obtained with sinusoidal current input. The orange parts of the curves apply to negative currents, and the blue parts for positive currents.

In Figure 5, the orange parts of the curves apply to negative currents, and the blue parts for positive currents. The orange flux-charge curve has a faster changing gradient (changing with q), and so the corresponding part of the IV curve is 'fatter' since a wider range of memristances are visited (for a given range of q). Another important observation is that the derivative of the flux-charge curve is not defined at all transition points when the current changes direction meaning that the memristance is discontinuous.

2.2 Memristive Systems

A physical device which satisfies the requirements of an ideal memristor is yet to be discovered and likely hard to fabricate. Chua and his student Kang later generalised the concept of memristors to memristive devices and memristive systems. [32] This concept extends beyond electrical devices [33], but the focus here is only on time invariant electrical devices. A memristive system (or device) is defined as any two terminal system (or device) which has a frequency dependant zero centred pinched hysteresis loop. This can be more formally defined by the time invariant state space equations 2.9 and 2.10 below. [32]

$$\frac{dx}{dt} = f(x, i) \qquad (The\ State\ Equation) \tag{2.9}$$

$$v(t) = R_M(x, i)i(t) \qquad (The\ Current - Voltage\ Relation) \tag{2.10}$$

The distinction between a memristive device and a memristor is in how the memristance changes. For the former, the memristance is some function of the state variable $x \in \mathbb{R}^n$, which in turn is some function of the history of current (or voltage). In contrast, a memristors memristance is determined directly from the flux-charge relation and the charge (or flux). Memristive systems and ideal memristors share some properties. For example, since all the possible IV curves always cross the origin, there is no ability for such a system to store any energy. [34] Another common property is that for high frequency and finite amplitude inputs, a memristive device behaves just like a resistor since the state variable has no time to change. Finally, it is worth noting that the memristor as described earlier in section 2.1 is essentially a memristive system with the state variable x being the charge q which has passed through the device in the past, and with $f(x, i) = i$, and $R_M(x, i) = \frac{d\varphi}{dx}$. From now this report uses the terms memristors and memristive devices loosely with both referring to a memristive system as described by 2.9 and 2.10 since this is of more practical interest.

2.3 Practical Memristors

The term 'memristor' in a practical context describes an extensive class of resistance changing devices. Such a change occurs through a wide range of physical mechanisms, for example, Redox, Phase Change, Spin Based and Magnetic, Electrostatic etc. Memristors have actually been around for a long time with over 100 published papers since the 1960's reporting the characteristic pinched hysteresis IV loops. However, these observations were not associated with memristance until HP Labs made the connection with their TiO_2 device in 2008. [35] Figure 6 below shows the so called *'bow ties'* observed by HP and Figure 7 compares this with Chua's theoretical pinched hysteresis loops.

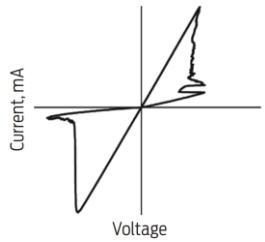


Figure 6: HP's bow ties [36]

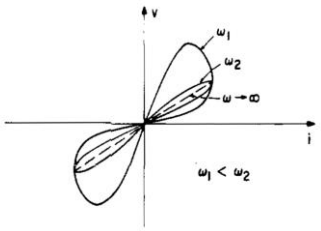


Figure 7: Chua's Pinched hysteresis loops [32]

In general, typical memristors (Redox based devices at least) have a metal-insulator-metal structure, where the 'insulator' is a transition metal oxide. The insulator is the resistive switching material, and the metals form the electrical contacts. May different materials of varying thickness $(2 - 130nm)$ are known to be used, for example TiO_2, TiO_x, TaO_x, HfO_x, CuO_x, ZnO_2, ZnO, NiO_2, Al_2O_3 to name a few. [37]

The main methods for implementing such devices are Nano Imprint Lithography (NIL) [38] and Atomic Layer Deposition (ALD). [39] [5] Both of these methods are complex and require multiple stages including annealing and electroforming. [40] [5] [39] Annealing provides energy for some of the oxygen from one side of the oxide to escape leaving behind oxygen vacancies and resulting in a higher conductance region. Electroforming causes permanent changes in the oxide layer. It causes conductive filaments to be formed via electro-reduction. After these steps, the device has the memristive property, and the basic mechanism of this is described in section 4 where models are reviewed. [41]

HP's device structure remains as one of the most popular even today. It is constructed of a thin layer of titanium dioxide sandwiched in between two platinum contacts. [4] Pure titanium dioxide alone is an intrinsic n-type semiconductor, and has a band gap of $\sim 3.0eV$ for the Routile phase, $\sim 3.2eV$ for the Anatase phase, and $\sim 3.4eV$ for the Brookite phase. [42] This metal-semiconductor structure could possibly help explain the back to back diode like IV curves for these devices as exampled in section 5.3. However, this is not fully investigated as part of this project since it is merely an interesting observation and not the focus.

3 Logic Families

3.1 Introduction

This section aims to introduce and begin analysis of four memristor based logic families: Material Implication Logic, Memristor Ratioed Logic (MRL), Memristor Aided Logic (MAGIC), and Programmable Logic Memristor Array. Firstly, the symbols and conventions used are laid out. Next, the fundamental logical operations of each family are discussed, and the ability to perform more advanced logic based on combining the basic operations is investigated. Design requirements are also derived and developed, and this section ends with a brief conclusion on feasibility of all families researched. The main comprehensive comparison is presented later in section 6.5, after more information is gathered from the simulation process.

3.2 Symbols and Conventions

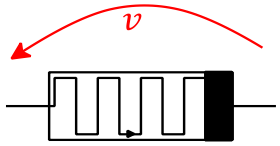


Figure 8: Memristor circuit symbol

Symbol:	*Resistance level*:	*Logic level*:
R_{off}	High	0
R_{on}	Low	1

Table 1: List of symbols and conventions used in this section

The black arrow in Figure 8 indicates the direction of the current and the voltage is measured according to the passive convention. If the current is positive, the memristance increases, whereas if the current is negative, the memristance decreases. A high memristance corresponds to logic 0, and a low memristance to logic 1. The high memristance is denoted as R_{off}, and the low as R_{on}. For convenience, this is summarised in Table 1. Finally, lower case letters are used to refer to particular memristors, and the corresponding uppercase letters refer to the logic state encoded by that memristors resistance level.

A memristor may have a voltage and/or current threshold for both R_{on} and R_{off} switching. Table 1 below lists the notation for such thresholds used throughout this report.

Symbol:	*Meaning*:
v_{on}	Voltage threshold to switch from R_{off} to R_{on}.
v_{off}	Voltage threshold to switch from R_{on} to R_{off}.
i_{on}	Current threshold to switch from R_{off} to R_{on}.
i_{off}	Current threshold to switch from R_{on} to R_{off}.
v_{set}	The negative voltage applied across the memristor to set the state to logic 1.
v_{clear}	The positive voltage applied across the memristor to clear the state to logic 0.
v_{cond}	Voltage level used by the Implication family during the $IMPLY$ operation. These are all related as follows: $v_{set} < v_{on} < v_{cond} < 0 < v_{off} < v_{clear}$
v_{high}	Denotes the voltage level used for logic high for the MRL gates and the PLMA. $0V$ is used for logic 0.
v_0	The positive voltage applied to MAGIC gates during operation. The subscript '0' can be replaced by the gate name to denote the voltage for a particular gate, e.g. v_{nor}, v_{nand}.

Table 2: List of symbols.

3.3 Material Implication Logic

This logic family uses the memristors resistance level to represent the logic level. The $IMPLY$, $(S = P \rightarrow Q)$, and $FALSE$, $(S = 0)$, operations constitute a complete logic basis meaning any arbitrary Boolean equation can be implemented based on these two fundamental operations. The main interest in this family stems from the possibility of significant area reduction and novel computer architectures in which logic is directly performed in the memory.

3.3.1 Fundamental Operations – *IMPLY, FALSE*

Table 3 below shows the truth table for logical implication and Figure 9 shows a design for a memristor based $IMPLY$ gate arranged in crossbar form. [27]

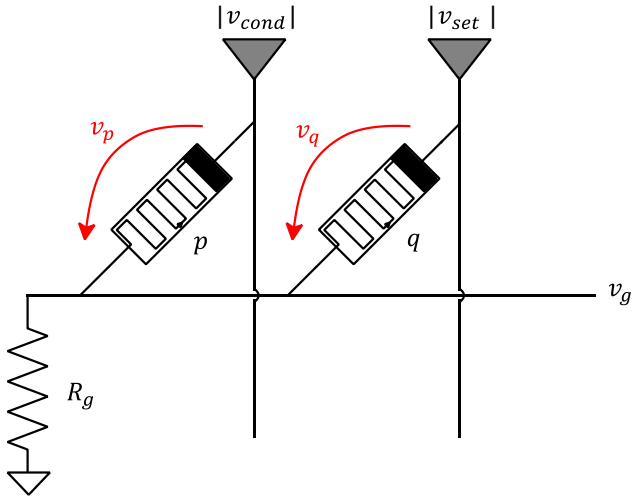
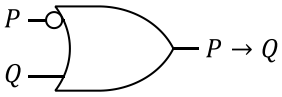


Figure 9: Material Implication gate in crossbar structure.

$Case$:	P:	Q:	$P \rightarrow Q$:
1	0	0	1
2	0	1	1
3	1	0	0
4	1	1	1

Table 3: Truth table for $IMPLY$.



$(P \rightarrow Q) = \overline{P} \vee Q$

Figure 10: Gate representation of $IMPLY$.

The principle is that the logical inputs P and Q are represented by the resistance level of the memristors p and q respectively. Voltages $|v_{cond}|$ and $|v_{set}|$ are applied for a short interval to the appropriate memristor terminals as shown by Figure 9 above. Here the absolute value is necessary due to the polarity of the memristors and the conventions set out in section 3.2. The voltages are chosen such that $|v_{cond}| < |v_{set}|$. After this pulse, the $IMPLY$ output is written to memristor q, meaning that the logic input Q is overwritten by the result. The ground resistor, R_g, is chosen such that $R_{on} \ll R_g \ll R_{off}$. [43] [27]

The operation of this gate can be studied by using basic circuit theory and by separately considering the four input cases as shown in Table 3 above. Table 4 below summarises the analysis performed.

Case:	Explanation: (Assuming the memristors have only voltage thresholds)
1	For this case, the memristance of q needs to decrease in order to produce the correct logical result. Assuming $(R_{off}\|\|R_{off}) \gg R_g$, then v_g is approximately $0v$. Hence v_p is approximately v_{set} (which is negative). v_{set} must be sufficiently negative ($\|v_{set}\| > \|v_{on}\|$) to ensure the resistance of q decreases so that it switches to logic 1. Under these conditions, v_p (the voltage across memristor p) is approximately equal to v_{cond} and this is also negative. In order to ensure the input logic state P is not destroyed v_{cond} must be sufficiently small, i.e. $\|v_{cond}\| < \|v_{on}\|$.
2	For this case, the memristance of q must remain low in order to produce the correct logical result. Since $\|v_{cond}\| < \|v_{on}\| < \|v_{set}\|$, the current can only flow downward through memristor q, meaning its resistance can only reduce and therefore the logic state $Q = 1$ can only be reinforced. It is not possible for a positive v_q so there is no need to worry about exceeding the v_{on} threshold and invalidating the operation. Furthermore, for memristor p, its logic state of 0 can only be reinforced since v_g is approximately v_{set} which means v_p is positive. As a result, case 2 is the least problematic.

3	For a correct logical result, the resistance of device q must remain high. Because $R_{on} \ll R_g$, v_g is approximately $\lvert v_{cond}\rvert$. Therefore $v_q \approx \lvert v_{cond}\rvert - \lvert v_{set}\rvert < 0$. This is required to be sufficiently small such to not reduce the memristance of device q. As for device p, v_p is negligible so memristor p retains its initial state.
4	For this final case, the memristance of q must remain low in order to produce the correct logical result. Assuming $R_{off} \ll R_g$, v_g is approximately $0.5 \times (\lvert v_{cond}\rvert + \lvert v_{set}\rvert)$. This means $v_q = 0.5 \times (\lvert v_{cond}\rvert - \lvert v_{set}\rvert)$ is negative and so the logic state 1 of q can only be reinforced. For device p, the voltage $v_p = 0.5 \times (\lvert v_{set}\rvert - \lvert v_{cond}\rvert)$ is positive, and this is required to be sufficiently small such that logic state 1 is maintained.

Table 4: Explanation of the $IMPLY$ gate shown in Figure 9 derived from basic circuit theory concepts. The 'cases' mentioned refer to Table 3.

In summary, assuming a device with a voltage threshold, the design requirements necessary for correct logical operation are given by 3.1 below. Additionally, the requirements necessary to preserve input state P are given by 3.2. Note that these do not guarantee correct operation since various assumptions are made during the derivation. The key assumption is the one that only the initial memristance state is considered; however, in reality the resistance will change during the operation. This is investigated in more detail with the help of simulations in section 6.3.

$$\lvert v_{on}\rvert < \lvert v_{set}\rvert, \qquad \lvert v_{cond}\rvert - \lvert v_{set}\rvert < \lvert v_{on}\rvert, \qquad R_{on} \ll R_g \ll \frac{R_{off}}{2} \tag{3.1}$$

$$\lvert v_{set}\rvert - \lvert v_{cond}\rvert < 2v_{off}, \qquad \lvert v_{cond}\rvert < \lvert v_{on}\rvert \tag{3.2}$$

In order to perform more complicated logic, a complete logic basis is needed. The operations $IMPLY$ and $FALSE$ form such a basis. [13] A $FALSE$ operation simply has the output equal to 0, and this can be achieved by applying a sufficiently negative voltage ($-v_{clear}$) to the crossbar structure (Figure 9). The minus sign occurs due to the polarity of the memristor in the crossbar structure. More advanced logic implementation is explained in the later subsections, but first the issue of speed verses robustness is discussed.

3.3.2 Speed and Robustness trade-off

The state of q only changes for case 1 which means this is the case which determines the minimum amount of time $\lvert v_{set}\rvert$ and $\lvert v_{cond}\rvert$ need to be pulsed on for until q reaches the required resistance level. [**43**] This pulse duration can be considered as the propagation delay or write time. The reason for pulsing rather than continuous voltage application is twofold; firstly, resistors inherently face I^2R losses, and secondly, there is a possibility for the state of q to tend towards logical 1 (R_{on}, low resistance). This is a problem especially for case 3 since the voltage applied across q is negative and the required outcome is for the logic level to not change and remain at logic 0. The tendency for the state to drift towards logical 1 is referred to as '*state drift*', and this has been observed in simulations using current threshold devices. [27] The term *'state drift'* refers to any unwanted drifting of any logic state. As expected, a longer pulse means more unwanted drifting occurs.

Note that this state drift was not predicted in the analysis of the $IMPLY$ gate (3.3.1) since the operation was explained assuming voltage thresholds as these are more convenient and useful for later simulations which use a voltage controlled model. The possibility of state drift for voltage controlled devices is investigated later in section 6.3.

The important point to convey is that if the switching speed is optimized through control of the applied voltages, there will be an increase in state drift (for current controlled devices), and hence there is a design trade-off between robustness and speed. [27]

3.3.3 Advanced Logic –*NAND*

The $NAND$ gate is universal in the sense that it can implement any Boolean logic. Such an operation can be realized by 3 memristors configured as shown in Figure 11 below. To perform $S = \overline{P \wedge Q}$, a sequence of $IMPLY$ and $FALSE$ operations are performed as shown by Table 5 below. This example illustrates the importance of the so called crossbar structure; it shows how the output of one operation can easily be used as an input of another one. [13] [27] [**43**]

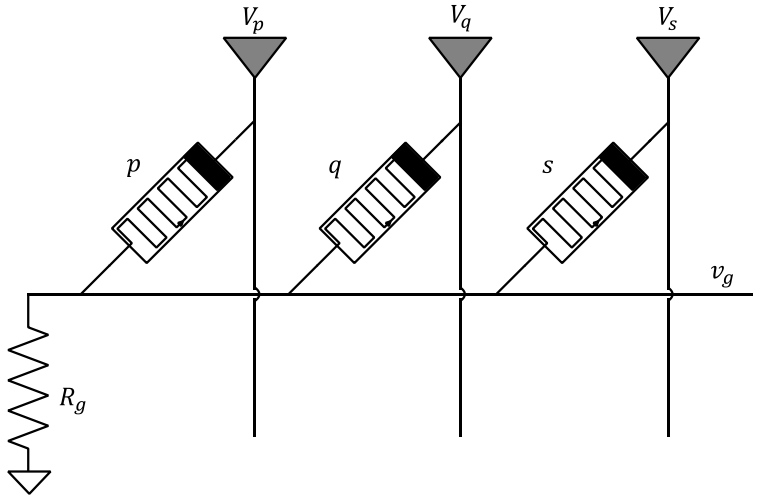


Figure 11: Crossbar structure which allows $NAND$ operation via sequential $IMPLY$ and $FALSE$ operations

$Step$:	$Logical\ Operation$:	$Voltages\ Applied$:
1	$FALSE$: $S = 0$	$V_s = -v_{clear}$
2	$IMPLY$: $S = P \rightarrow S$	$V_p = \lvert v_{cond} \rvert$, $V_s = \lvert v_{set} \rvert$
3	$IMPLY$: $S = Q \rightarrow S$	$V_q = \lvert v_{cond} \rvert$, $V_s = \lvert v_{set} \rvert$

Table 5: The sequence of operations required to realise the $NAND$ operation. (The left hand side of the expression $S = P \rightarrow S$ referrers to the updated value of S, and the right hand side refers to the previous values.)

The table below shows how the state of each memristor changes after each step during a $NAND$ operation. The red colours indicate which memristors are affected by the current step, and the 'x' signifies that the fact that the input may be destroyed during the $IMPLY$ operations. This is definitely the case if the requirements as specified by 3.2 are not satisfied. By the end of the $NAND$, both of the inputs P and Q may be destroyed. However, this is not an issue for correct $NAND$ operation since each input is only used once to generate the output.

Initial States			After Step 1: $(S_1 = 0)$			After step 2: $(S_2 = P_1 \rightarrow S_1)$			After step 3: $(S_3 = Q_2 \rightarrow S_2)$		
P_0	Q_0	S_0	P_1	Q_1	S_1	P_2	Q_2	S_2	P_3	Q_3	S_3
0	0	0	0	0	0	x	0	1	x	x	1
0	0	1	0	0	0	x	0	1	x	x	1
0	1	0	0	1	0	x	1	1	x	x	1
0	1	1	0	1	0	x	1	1	x	x	1
1	0	0	1	0	0	x	0	0	x	x	1
1	0	1	1	0	0	x	0	0	x	x	1
1	1	0	1	1	0	x	1	0	x	x	0
1	1	1	1	1	0	x	1	0	x	x	0

Table 6: Evolution of the logic state of each memristor over the $NAND$ sequence as shown by Table 5.

This operation can be represented pictorially by two cascaded '$IMPLY$' gates as shown in Figure 12 below.

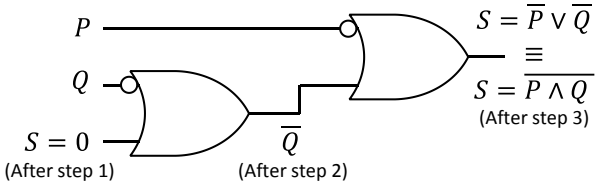


Figure 12: Alternate representation of the $NAND$ gate.

3.3.4 Copy Operation

In general, implementing advanced logic in this manor will generally require the same input to be used more than once. This is an issue since one of the inputs is always destroyed when it is overwritten with the output. If the input is to be used again elsewhere it must first be copied. This copy operation requires a sequence of 4 steps and an additional work memristor. Referring to Figure 11 again, the sequence for copying the logic state of memristor p to memristor q using a spare work memristor s is shown by Table 7 below.

$Step$:	$Logical\ Operation$:	$Voltages\ Applied$:
1	$FALSE$: $S = 0$	$V_s = -v_{clear}$
2	$IMPLY$: $S = P \rightarrow S$	$V_p = \lvert v_{cond} \rvert$, $V_s = \lvert v_{set} \rvert$
3	$FALSE$: $Q = 0$	$V_q = -v_{clear}$
4	$IMPLY$: $Q = S \rightarrow Q$	$V_s = \lvert v_{cond} \rvert$, $V_q = \lvert v_{set} \rvert$

Table 7: Sequence of steps required to copy p into q using a spare work memristor s

One observation is that for this operation to be useful the input state P is must not destroyed during step 2 otherwise the copy operation is pointless.

3.3.5 Advanced Logic – Reduced Memristor Method

Lethonen Poikonen and Laiho [44] show that any Boolean function $f: B^N \rightarrow B$ can be implemented with only $N + 2$ memristors provided the states of N memristors which store the inputs are not altered. Their analysis is explained and summarised here.

Let $\hat{\mathrm{P}} = \{p_1,\ p_2,\ p_3, .., p_N\}$ be the set of input memristors and $\hat{\mathrm{M}} = \{m_1,\ m_2\}$ be the set of the 2 computing memristors. The input memristors are initialised to the inputs, and the computing memristors are initialised to logic 0. When the imply function $X \rightarrow M_i$ is implemented as explained in 3.3, the logical result is overwritten to memristor m_i, and therefore $M_i = X \rightarrow M_i$. This expression is equivalent to $M_i = \overline{X} \vee M_i$. The only operations necessary are $M_i = 0$ and $M_i = X \rightarrow M_i$, where $x \in \hat{P} \cup \hat{M} \backslash \{M_i\}$. This means that the inputs, P_i, are not destroyed provided no state drift occurs.

Any Boolean equation can be written in a Recursive Conjunctive Form as described by equation 3.3 below. Here, α_k denotes a product of input terms, e.g. $\alpha_1 = p_1 \wedge\ p_4 \wedge\ p_2$. The notation of $\beta_k \in \{0, 1\}$ is used to signify inversion; for example ${\alpha_k}^0$ represents $\overline{\alpha_k}$, whereas ${\alpha_k}^1$ just represents α_k.

$$f = \Big(\big(\ \ldots\ \big((\alpha_k)^{\beta_k} \vee\ \alpha_{k-1} \big)^{\beta_{k-1}} \ \ldots\ \vee \alpha_2 \big)^{\beta_2} \vee \alpha_1 \Big)^{\beta_1} \qquad 3.3$$

Sub-sequence 1:	$[(M_1 = P \rightarrow M_1] \quad for\ any\ p \in \hat{P}$
Sub-sequence 2:	$[\{M_2 = M_1 \rightarrow M_2\},\ \{M_1 = 0\}]$
Sub-sequence 2:	$[\{M_2 = M_1 \rightarrow M_2\},\ \{M_1 = 0\},\ \{M_1 = M_2 \rightarrow M_1\},\ \{M_2 = 0\}]$ *Then swap memristor indexes* 1 *and* 2 so m_2 contains what was stored in m_1 and vice versa (M_i *still denotes logic state of memristor* m_i).

Table 8: Sub-sequences which can be combined to compute any Boolean equation in the form of equation 3.3.

Table 8 above defines three sub-sequences which can be combined to implement any Boolean equation in the form of equation 3.3. Applying sub-sequence 1 to an input P_1 gives $M_1 = \overline{P}_1 \vee M_1$, and since M_i is initialised to 0, the result is $M_1 = \overline{P}_1$. Applying this sequence again with another input P_2 gives $M_1 = \overline{P}_2 \vee (\overline{P}_1)$, and this is equivalent to $M_1 = \overline{P_2 \wedge P_1}$. This sequence can be repeated as required on any of the input memristors $p_i \in \hat{P}$ to obtain a function of the form:

$$f = \overline{P_{i_1} \wedge P_{i_2} \ \ldots \wedge P_{i_j}} \tag{3.4}$$

Applying Sequence 2 results in $M_2 = \overline{M}_1 \vee M_2$, where M_1 is of the general form of equation 3.4. Because $M_2 = 0$ for the first time the sequence is used, the result is $M_2 = \overline{M}_1$. Next, M_1 is cleared to 0 so that it can be used again to generate another function in the form of equation 3.4 via Sequence 1. This means using just sequences 1 and 2, any equation of the general form described by 3.5 can be implemented.

$$f = \left(P_{i_1} \wedge P_{i_2} \ \ldots \wedge p_{i_j}\right) \vee \left(P_{i_{j+1}} \wedge P_{i_{j+2}} \ \ldots \wedge P_{i_{j+k}}\right) \ldots \vee \left(P_{i_{j+k+1}} \wedge P_{i_{j+k+2}} \ \ldots \wedge P_{i_{j+k+m}}\right) \tag{3.5}$$

This is essentially the general form as in equation 3.3 but with all $\beta_k = 1$. To deal with the case $\beta_k = 0$, sequence 3 is necessary, and this is simply an extension of sequence 2. The additional task that this sequence performs is to invert the whole result in M_2; this inversion takes 2 extra steps.

In this analysis the inputs are assumed to not be destroyed. However, implementing the $IMPLY$ as in 3.3 can possibly destroy the inputs, especially if each input is used multiple times since the input state can be cumulatively weakened by state drift. A final comment on this particular process is that although the number of memristors (and hence chip area) is minimal, the computation process requires a large number of steps. Typically the number of steps can be traded off with the number of memristors to a certain extent.

3.4 Memristor Ratioed Logic

The MRL family represents the logic levels using voltages and hence it has potential to be compatible with standard CMOS logic. It is inspired from Diode logic [45] and as a result, they both share some properties, i.e. the logic is non-inverting, and non-restoring. [46] This section shows how memristors can implement CMOS compatible AND, OR logic gates. A NOT operation cannot be realized by memristors alone (since they are two terminal passive devices), and therefore CMOS inverters are required to complete the logic basis. Using CMOS has the additional benefit of providing current isolation between consecutive gates, while also restoring the digital signal and increasing fan out.

Practical memristor devices are compatible with standard CMOS processes. [47] They can be fabricated above the CMOS devices in the metal layers of the IC. Furthermore, the size of a typical memristor is comparatively small since the process is similar to that of making a via between the metal layers. [26] In short, logic circuits which are a hybrid of CMOS and memristors can be smaller than CMOS only implementations, and case studies of adders demonstrate/estimate an area reduction of 50% to 75%. [48] [23] [24] This is the major significant advantage and forms the motivation for study of this logic family. The concepts and figures used in this section are based on those presented in [23] [24] [48].

3.4.1 MRL Gates:

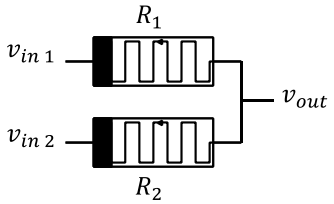


Figure 13: A 2-input OR gate, adapted from [24].

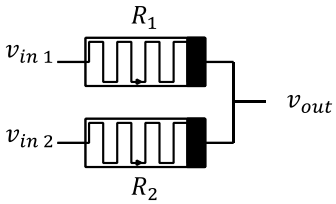


Figure 14: A 2-input AND gate, adapted from [24].

Considering the 2-input OR gate as shown in Figure 13, when both the inputs are the same, i.e. $\{v_{high}, v_{high}\}$, or $\{0v, 0v\}$, there are no current flows and the input is the same as the output. This assumes no load on the output. However, when only one input is high, the current flows from the high input to the low input. This causes the resistance of the device connected to the high input to decrease to R_{on}, and the resistance of the device connected to the low input to increase to R_{off}. The output is then determined by the potential divider formed:

$$v_{out} = \frac{R_{off}}{R_{on} + R_{off}} \times v_{high} \approx v_{high} \qquad assuming\ \ R_{off} \gg R_{on} \tag{3.6}$$

This completes the OR operation since the output is high only if at least one input is high. Figure 14 shows a 2-input AND. When both the inputs are the same, the behaviour is exactly the same as the OR gate since no current flows. However, when only one input is high, the current flows from the high input to the low input. Due to the different polarity of the device, this now causes the resistance of the device connected to the high input to increase to R_{off}, and the resistance of the device connected to the low input to decrease to R_{on}. Based on the potential divider formed, the output is then low:

$$v_{out} = \frac{R_{on}}{R_{off} + R_{on}} \times v_{high} \approx 0 \qquad assuming\ \ R_{off} \gg R_{on} \tag{3.7}$$

The initial resistance state of the memristors does not affect the logical output. It only affects the delay for the output to settle to the correct value for the case when the two inputs are different. This is because currents must flow to change the resistance state which takes non-zero time. Larger currents mean the resistance changes faster, hence high voltages result in faster logic operation. [26] The general concept can be extended to create N input gates as shown in the figures below.

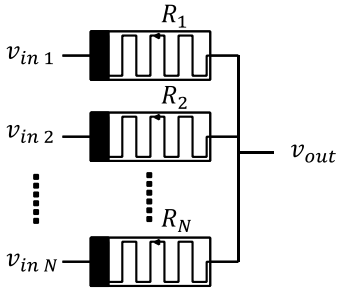


Figure 15: A N-input OR gate, adapted from [24].

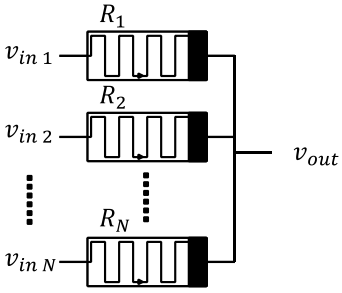


Figure 16: A N-input AND gate, , adapted from [24].

3.4.2 Advanced Logic – *NAND, NOR*

As previously mentioned, a complete logic basis can be formed with the addition of a CMOS inverter. Figure 17 and Figure 18 below show hybrid $NAND$ and NOR gates respectively. One added benefit of incorporating CMOS with memristors is that the amplification of the inverter/buffer can restore logic levels; something which memristors cannot achieve since they are passive. Without some sort of buffering, the quality of the output voltage logic value would eventually decay as more gates are cascaded. This can result in incorrect operation especially if the devices have a switching threshold.

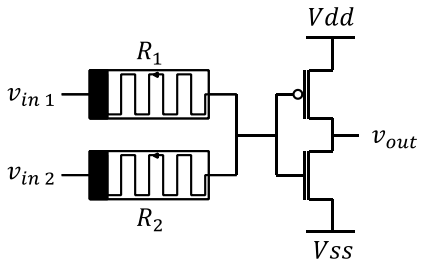


Figure 17: A hybrid CMOS 2-input NOR gate. Adapted from [24].

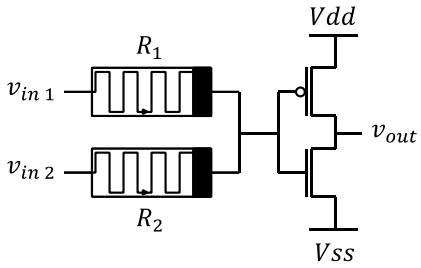


Figure 18: A hybrid CMOS 2-input $NAND$ gate. Adapted from [24].

Any logic can be implemented using only these $NAND$ gates (or only NOR gates), but this is inefficient in terms of area and power. This is because every $NAND$ gate requires 2 connections between the CMOS and memristive layers in the IC; one at the input of the CMOS inverter, and one at the output. A more optimised design would use the inverter only when inversion is needed, or when signal restoration is needed. [26] [24]

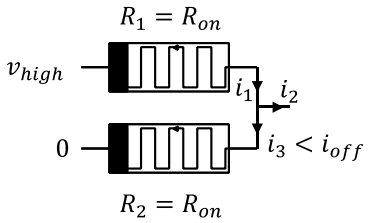


Figure 19: Effects of currents between gates

The problem with optimising a design in this manor is due to the current which can flow from the output of one gate to the input of the following gate as shown by i_2 in Figure 19. The issue arises not only during transients, but during steady state since there can be steady state input currents for some cases.

This current may cause the current through one of the memristors of the previous gate to not exceed the threshold, hence resulting in the gate not fully switching. In the example of Figure 19, R_2 is supposed to switch to R_{off}, but the current, i_3 is not sufficient. As a result the output voltage is not as high as it should be and this may cause failure for

subsequent gates. Memristors with thresholds are more vulnerable to this voltage degradation. One option to counter act this is to use higher voltages, but this affects power consumption and is limited by the CMOS process as high voltages can break down the transistors.

3.4.3 Hazards

MRL gates are known to be prone to dynamic hazards. [26] A hazard can occur whenever the input changes to logic $\{1,0\}$ or $\{0,1\}$. This is because these two sets of inputs require memristance to change for the output to settle to the correct value, and depending on the initial memristance, there can be a dynamic hazard. For a current controlled device, the longest hazard occurs for the case when both memristors are initially at a high resistance since the current is smallest meaning it takes longest for the resistance to change to the correct value.

Hazards are investigated for the OR and AND gates as shown in Figure 13 and Figure 14 respectively. The figures show how the voltage output and the resistance level of each memristor changes over time with different inputs. They have been built assuming no load on the output node, and a linear memristor with no threshold and with the following specifications:

$$R_{off} = 10{,}000\Omega, \quad R_{on} = 100\Omega, \quad and \quad \dot{r}(t) = [2{,}500{,}000 \times i(t)]\Omega s^{-1} \tag{3.8}$$

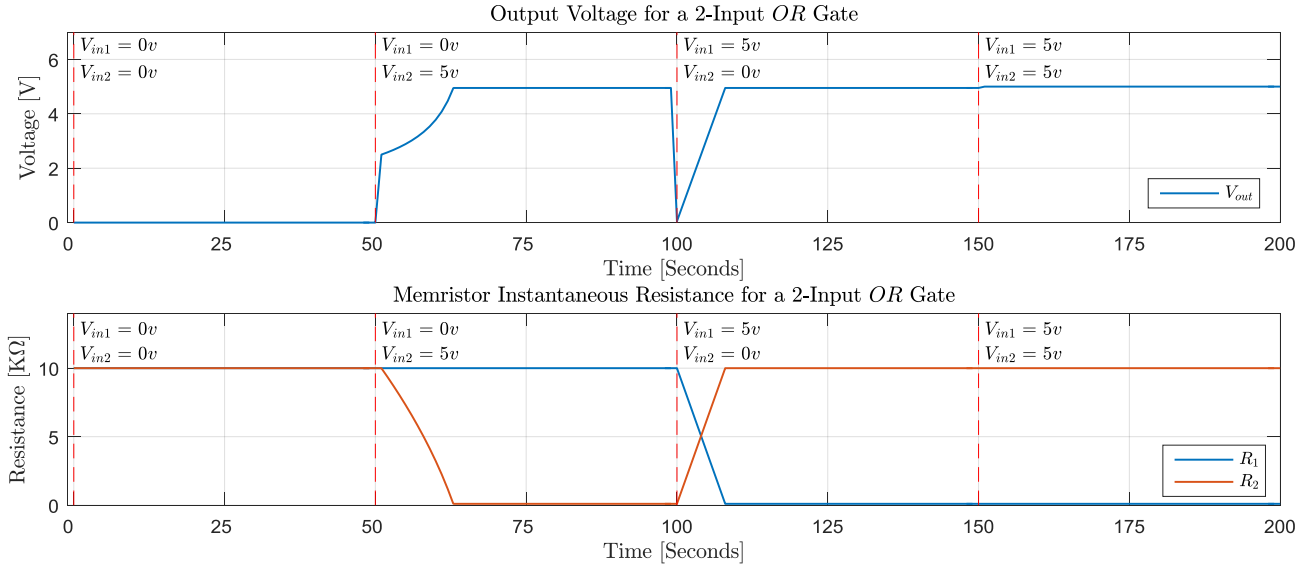


Figure 20: The two cases of dynamic hazards for the 2-input OR gate

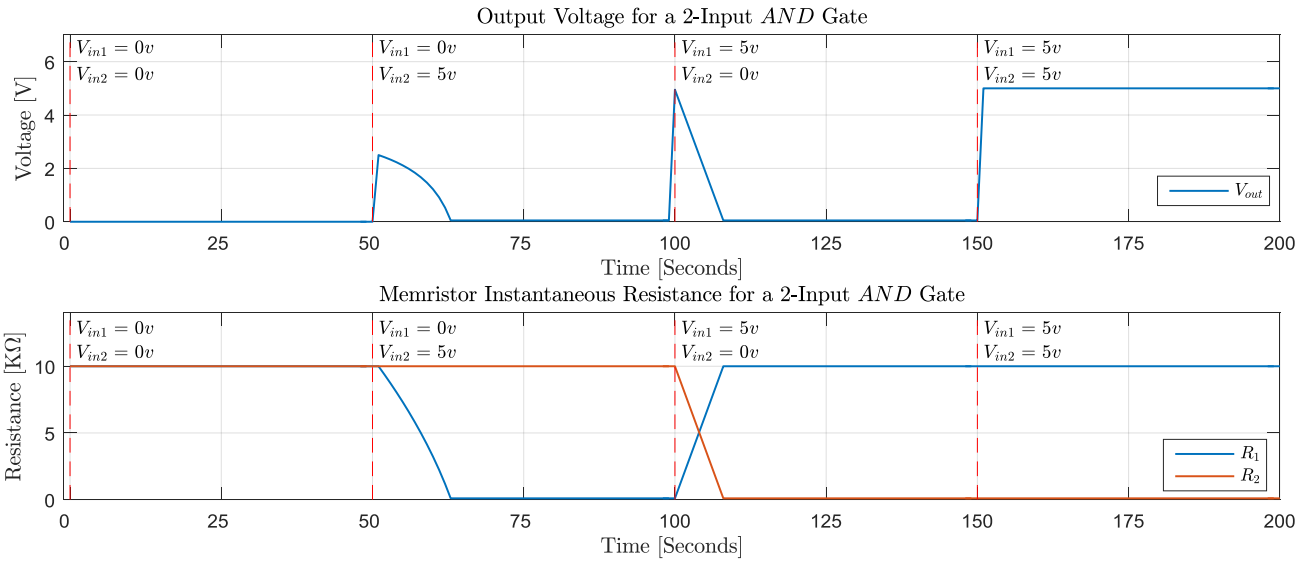


Figure 21: The two cases of dynamic hazards for the 2-input AND gate

3.5 Memristor Aided Logic

This logic family is similar to the Implication family in that the logic values are stored as the memristance. The motivation for study is also the same, i.e. the possibility for significant area reduction and novel computer architectures. It is possible to implement AND, OR, NOT, NOR, and $NAND$ gates; each of these gates requires one memristor for each input, and one for the output. For correct operation, the output memristor must be correctly initialised, and this initial state depends on the gate type. Unlike the $IMPLY$ gate, the output is stored in a separate memristor, and only a single voltage level is required for operation. The design guideline analysis which follows assumes the devices to have a current threshold, and this is because analysis of voltage threshold devices is already presented by Kvatinsky et al. [25] Kvatinsky's results are summarised in Table 9 later.

3.5.1 MAGIC Gates

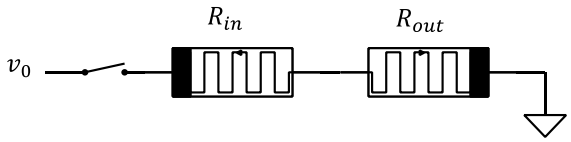


Figure 22: A NOT gate. Adapted from [25].

Figure 22 shows a NOT gate. For correct operation, the output must be initialised to logic 1 (low resistance, R_{on}). When v_0 is applied, R_{out} can switch to logic 0 provided the current through the device exceeds its thresholds. For correct NOT operation, this should only happen when R_{in} is 1, and must not happen when R_{in} is 0. This can be formalised by design guidelines for v_0 as shown below:

$$i_{off}(2R_{on}) < v_0 < i_{off}(R_{on} + R_{off}) \tag{3.9}$$

Due to the direction of the current flow, the input memristor can change from 0 to 1. If the inputs need to be preserved, then this change can be is prevented if $v_0 < |i_{on}|(R_{on} + R_{off})$. Combining these constraints gives:

$$i_{off}(2R_{on}) < v_0 < min(i_{off}, |i_{on}|) \times (R_{on} + R_{off}) \tag{3.10}$$

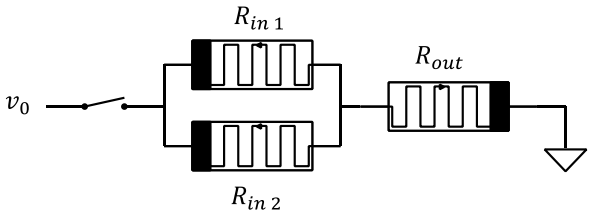


Figure 23: A 2-input NOR gate. Adapted from [25].

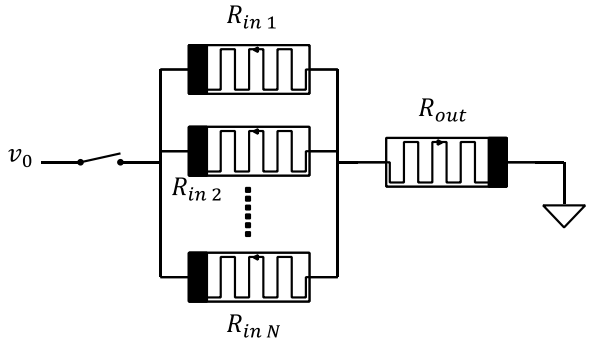


Figure 24: A N-input NOR gate. Adapted from [25].

Figure 23 shows a 2-input NOR gate. The output must be initialised to logic 1. When v_0 is applied, the output may change from 1 to 0 provided the thresholds are exceeded. This needs to be true if any input is 1, and must be false if all inputs are 0. By considering the worst cases, this can be formalised as:

$$i_{off}(R_{off}||R_{on} + R_{on}) < v_0 < i_{off}(0.5R_{off} + R_{on}) \tag{3.11}$$

Due to the polarity of the input memristors, they can only change from 0 to 1. Hence there is no need to worry about trying to preserve the inputs for the case they are both at 1. Furthermore, the case where one input is 0 is not an issue

either. This is because almost all of the current will flow through the memristor with the low resistance (logic 1) and hence not alter the memristor with logic state 0. If both the inputs are 0, then it is required that $v_0 < |2 \times i_{on}| \times \left(0.5 \times R_{off} + R_{on}\right)$. Merging these two results in:

$$i_{off}\left(R_{off}||R_{on} + R_{on}\right) < v_0 < \min\left(i_{off}, |2 \times i_{on}|\right) \times \left(0.5R_{off} + R_{on}\right) \quad 3.12$$

Figure 24 above shows a N-input NOR gate. Its operation is similar to the 2-input gate, and its design constrains can be derived with similar considerations as previously used:

$$i_{off}\left(\frac{R_{off}}{N-1}||R_{on} + R_{on}\right) < v_0 < \min\left(i_{off}, |N \times i_{on}|\right)\left(\frac{R_{off}}{N} + R_{on}\right) \quad 3.13$$

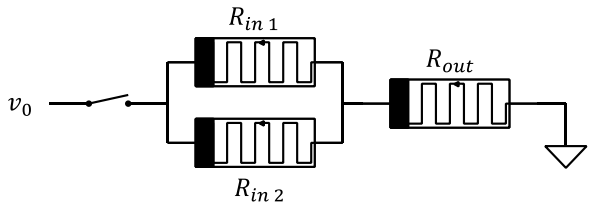


Figure 25: A 2-input OR gate. Adapted from [25].

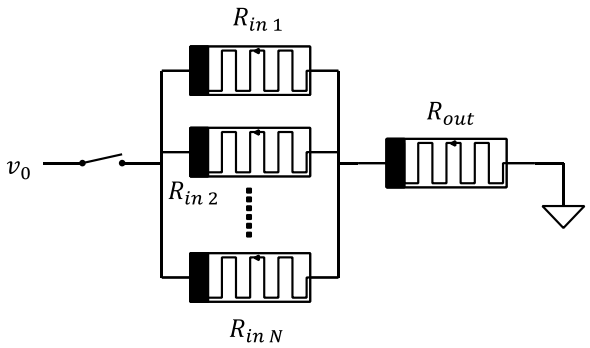


Figure 26: A N-input OR gate. Adapted from [25].

Figure 25 above shows a 2-input OR gate. The operation is similar to the NOR gate except this time R_{out} is initialized to 0, and due to its polarity, it can switch to 1 provided the current through it is exceeds its thresholds. Figure 26 shows an N-input OR gate which is just an extension of Figure 25. The design constraints (eqns 3.14) are determined with the same considerations as before. If these are satisfied, the inputs are also preserved.

$$|i_{on}|\left(\frac{R_{off}}{N-1}||R_{on} + R_{off}\right) < v_0 < |i_{on}|\left(\frac{R_{off}}{N} + R_{off}\right) \quad 3.14$$

As the number of inputs increase, the set of admissible values of v_0 decreases (assuming the other parameters are fixed). This is illustrated by Figure 27 below which is created by assuming $R_{off}/R_{on} = 100$ and $i_{off} = |i_{on}|$. Note that a NOR gate with a single input can be considered to be a NOT gate.

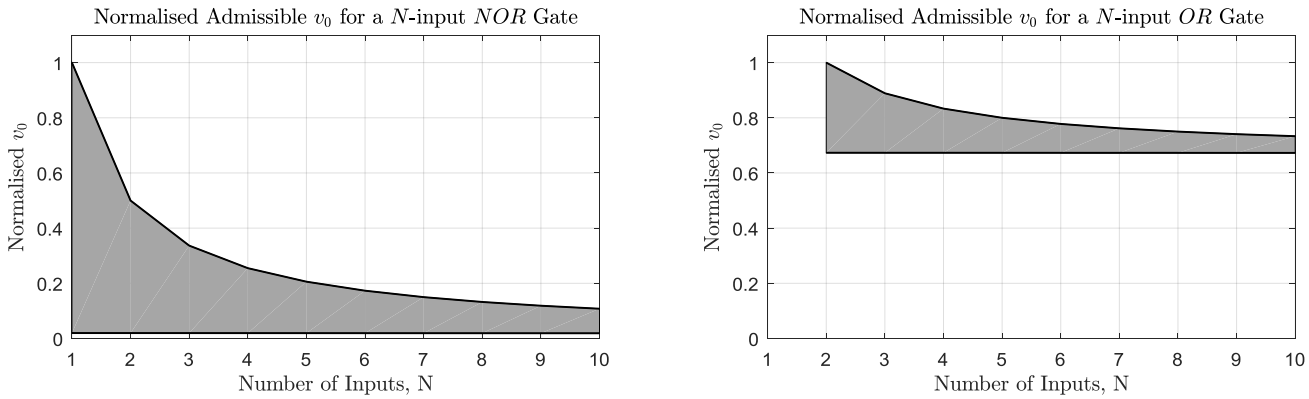


Figure 27: The left plot shows the normalised admissible values for v_0 as a function of the number of inputs for a NOR gate. The right plot shows the same for a OR gate. Derived assuming $R_{off}/R_{on} = 100$ and $i_{off} = |i_{on}|$.

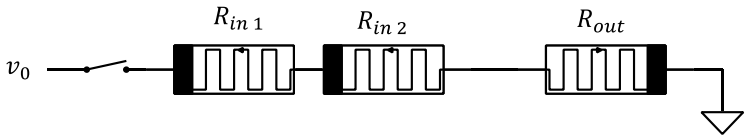


Figure 28: A 2-input $NAND$ gate. Adapted from [25].

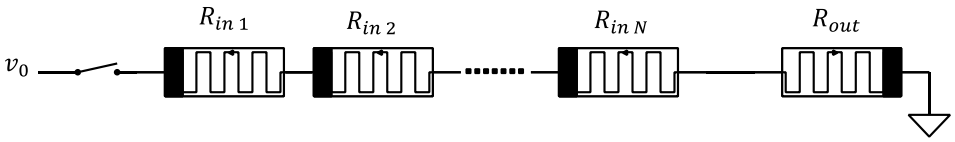


Figure 29: A N-input $NAND$ gate. Adapted from [25].

Figure 28 shows a 2-input $NAND$ gate. The output is initialised to logic 1 (R_{on}, low resistance). When v_0 is applied, the output may switch to 0 provided the threshold of R_{out} is exceeded. The output should only switch if both inputs are 1, and not switch if any input is 0. Considering the worst case scenario, this can be formalised by the equation below:

$$i_{off}(3R_{on}) < v_0 < i_{off}(2R_{on} + R_{off}) \tag{3.15}$$

The inputs can only change from 0 to 1 when v_0 is applied due to their polarity. Therefore the most troublesome case for input preservation is when only one input is logic 0. Under this case it is required that $v_0 < |i_{on}|(2R_{on} + R_{off})$. Combining this with equation 3.15 gives:

$$i_{off}(3R_{on}) < v_0 < min(i_{off}, |i_{on}|) \times (2R_{on} + R_{off}) \tag{3.16}$$

These constraints can be generalised to an N input gate as shown in Figure 29:

$$i_{off}((N+1) \times R_{on}) < v_0 < min(i_{off}, |i_{on}|) \times (N \times R_{on} + R_{off}) \tag{3.17}$$

Figure 30 shows a 2-input AND gate. Its operation is similar to the $NAND$ gate except R_{out} is now initialised to logic 0 (R_{off}, high resistance). When v_0 is applied, the current flow can cause R_{out} to change to logic 1. Figure 31 shows the extended version with N inputs. The design constraints are shown in equation 3.18, and if these are satisfied, the inputs are also preserved after the operation.

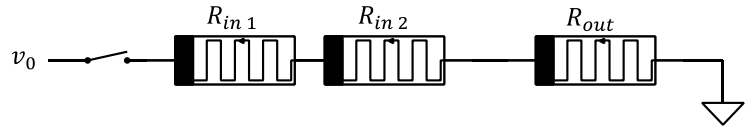


Figure 30: A 2-input AND gate. Adapted from [25].

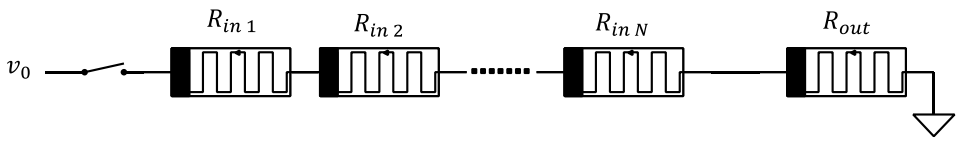


Figure 31: A N-input AND gate. Adapted from [25].

$$|i_{on}|(N \times R_{on} + R_{off}) < v_0 < |i_{on}|\left((N-1)R_{on} + 2R_{off}\right) \tag{3.18}$$

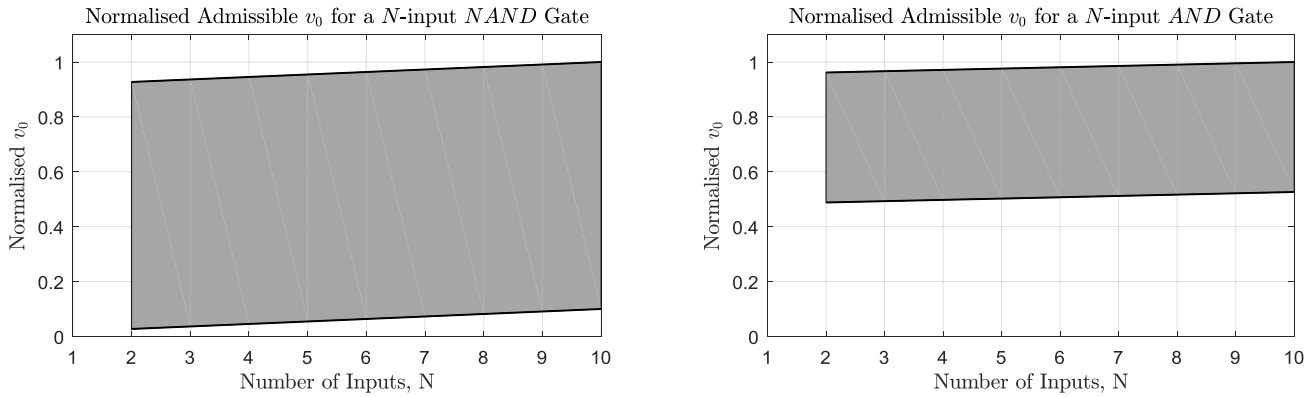


Figure 32: The left plot shows the normalised admissible values for v_0 as a function of the number of inputs for a $NAND$ gate. The right plot shows the same for a AND gate. Derived assuming $R_{off}/R_{on} = 100$ and $i_{off} = |i_{on}|$.

Figure 32 above shows how the admissible set of v_0 varies with the number of inputs for the $NAND$ and AND gates. Unlike the OR and NOR gates, the range of the set is not reducing with increasing N. Shahar Kvatinsky and Guy Satat [25] present a similar analysis on the design guidelines for the various gates, but under the assumption that the memristors have only voltage thresholds. Their results are summarised in Table 9 below.

Gate:	*N-Input Design Requirements*:
NOR	$\frac{v_{off}}{R_{on}} \times \left(R_{on} + \left(\frac{R_{off}}{N-1}\right) \| R_{on}\right) < v_0 < \min\left[\left(v_{off}\left(1 + \frac{R_{off}}{N \times R_{on}}\right)\right),\ \left(\|v_{on}\|\left(1 + \frac{N \times R_{on}}{R_{off}}\right)\right)\right]$
$NAND$	$v_{off}(N+1) < v_0 < \min\left[\left(v_{off}\left(N + \frac{R_{off}}{R_{on}}\right)\right),\ \left(\|v_{on}\|\left(1 + \frac{N \times R_{on}}{R_{off}}\right)\right)\right]$
OR	$v_{on} < v_0 < v_{on}\left(1 + \frac{1}{N}\right)$
AND	$v_{on}\left(1 + \frac{N \times R_{on}}{R_{off}}\right) < v_0 < v_{on}\left(2 + \frac{(N-1) \times R_{on}}{R_{off}}\right)$
NOT $(N = 1)$	$2v_{off} < v_0 < \min(\|v_{on}\|,\ v_{off}) \times \frac{R_{off}}{R_{on}}$

Table 9: Summary of design constraints for the various MAGIC gates assuming voltage thresholds for both '*on*' and '*off*' switching. [25]

3.5.2 Advanced Logic – Combining Gates:

NOR and NOT gates can be organised in the crossbar form as shown in Figure 33. [49] Due to this symmetric structure, the output of one operation can easily be the input for the next operation. Table 10 below shows a sequence which performs $C = \overline{A \vee B}$ and then $D = \overline{C}$ in order to illustrate this concept. Here it is assumed each voltage source (V_i) is high impedance (open circuit) unless a voltage value is specified.

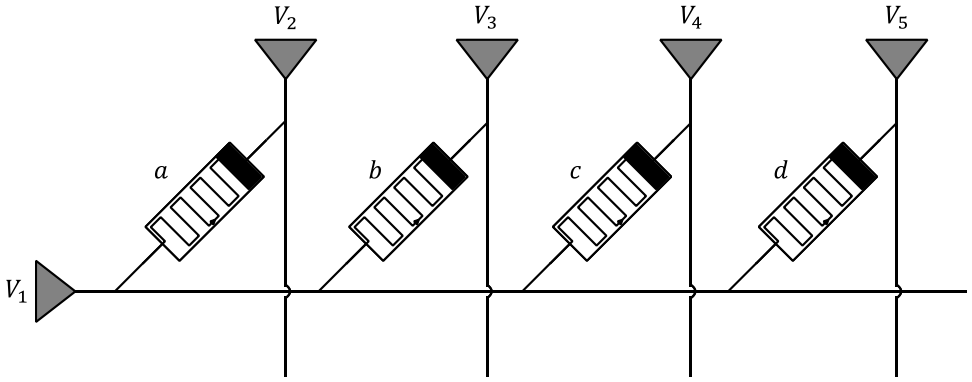


Figure 33: Section of a crossbar structure.

$Step$:	$Operation\ Performed$:	$Voltages\ applied$:	
1	$Initialise\ C = 1$	$V_4 = \lvert v_{set} \rvert,$	$V_1 = GND$
2	$C = \overline{A \vee B}$	$V_2 = V_3 = v_{nor},$	$V_4 = GND$
3	$Initialise\ D = 1$	$V_5 = \lvert v_{set} \rvert,$	$V_1 = GND$
4	$D = \overline{C}$	$V_4 = v_{not},$	$V_5 = GND$

Table 10: Sequence of operations to perform $C = \overline{A \vee B}$ and then $D = \overline{C}$.

In Table 10, v_{nor} is chosen to satisfy equation 3.12, and v_{not} to satisfy equation 3.10. Like the Implication logic family, a voltage driver capable of providing multiple voltage values as well as having a high impedance mode is necessary. However, for a given crossbar size, the MAGIC family requires more of these drivers since at least one per column is also needed.

3.5.3 Limitations

In investigating the MAGIC family, several limitations were identified. Firstly, it is not trivial to use the output of the other gates ($NAND$, AND, OR) as inputs for following stages in the circuit. This is due to the complex interconnection required. This is illustrated in the simple example below which connects the output of a 2-input AND gate into a 2-input OR gate. In this case, 4 switches are required just to isolate the two gates while computation takes place. This complication is exacerbated if the same memristor is needed as an input for multiple gates.

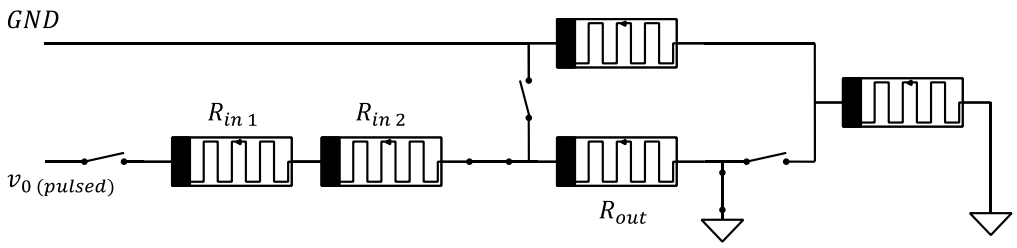


Figure 34: AND gate with its output feeding into an OR gate. The OR gate is isolated by the switches while the AND operation takes place.

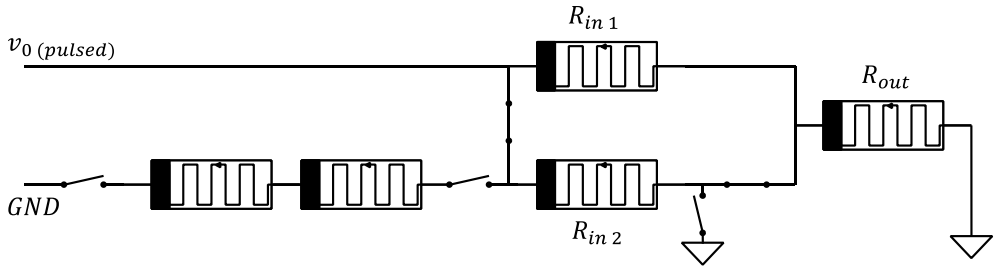


Figure 35: OR operation performed on the output of the AND gate, with the AND gate isolated by the switches.

Furthermore, this simple example does not even consider the circuitry necessary to initialise the output stages to logic 0 as required for correct gate operation. In summary, it is not really feasible to construct complicated logic circuits using MAGIC AND, OR and $NAND$ gates. Although NOR and NOT gates can be arranged in the crossbar structure, they too have an inherent limitation for the case of current threshold devices.

Only the initial values of the memristors were considered when deriving the design requirements, and therefore satisfying these requirements does not guarantee correct operation. During operation, R_{out} is intended to change for some input cases. Recall that the output of the NOR gate is initialised to logic 1 (low resistance, R_{on}), and if any of the inputs are logic 1, the output switches to logic 0. The problem is that this is never a complete switch to 0 under the assumption that the memristors have a current threshold. Essentially, this is because during switching the output memristance will increase and the current will reduce towards the threshold which prevents complete switching.

To be more precise, the output memristance can never switch beyond R_{off}/N. This can be seen from the design constraint equation below, in which $R_{out}(0) = R_{on}$.

$$i_{off}\left(\frac{R_{off}}{N-1}||R_{on} + R_{out}(t)\right) < v_0 < \min(i_{off}, |N \times i_{on}|)\left(\frac{R_{off}}{N} + R_{on}\right) \quad 3.19$$

During operation, $\mathrm{R_{out}(t)}$ increases with time provided that the lower inequality is satisfied and at least one of the inputs is logic 1. However, this increase in resistance is upper-bounded since v_0 is upper bounded. The bound on $R_{out}(t)$ is clarified by 3.20 below.

$$R_{out}(t) < \left(\frac{R_{off}}{N} + R_{on} - \frac{R_{off}}{N-1}||R_{on}\right) \approx \frac{R_{off}}{N} \quad 3.20$$

This issue does not arise if the device is assumed to have only a voltage threshold. In this case, when switching occurs it is accelerated because the output memristor gets a larger proportion v_0 because its resistance increases.

The $NAND$ gate has a similar issue except in this case the output memristance can approach R_{off} under ideal cases as compared to R_{off}/N for the NOR. This is the same for the NOT gate. One way to overcome this limitation is to sense the output resistance after each NOR operation and reinforce the logic state after the operation before using it as an input to the next stage. This requires more computation steps and complicated sense circuitry which mostly defeats the main objective of using memristors; namely a reduction in area.

3.6 Programmable Logic Memristor Array

Memristors can also be used to implement a Programmable Logic Array (PLA) as shown by Figure 36 below. [23] Here the connections are made by programing the memristors to be either high or low resistance in order to make the required connections. The logic is then performed using sub-threshold voltages/currents such that the programming remains unchanged. The gates used would most likely be CMOS due to the large fan-out required, but can be also hybrid depending on if the memristor properties are adequate, for example if the R_{off}/R_{on} ratio is sufficiently high.

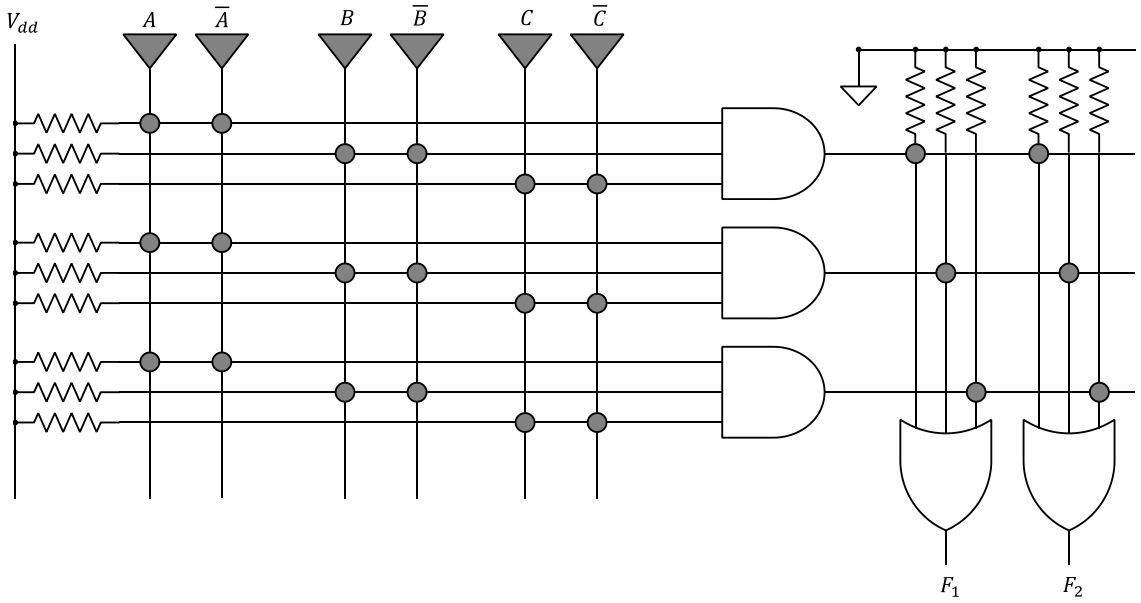


Figure 36: PLMA example. The grey circles represent memristors which can be programmed as either high or low resistance in order to make the desired connections. Based on [23].

It is required that the memristances do not change during normal operation. Therefore it is necessary that equation 3.21 is true for devices with a current threshold, and 3.22 for devices with a voltage threshold. The voltage also needs to be CMOS compatible so the requirements can be tricky to achieve unless the thresholds are relatively high. These requirements are the same regardless of the polarity of the memristors in Figure 36.

$$\frac{v_{high}}{R_{on} + R_{off}} < min(i_{off}, |i_{on}|) \tag{3.21}$$

$$\frac{v_{high} \times R_{off}}{R_{on} + R_{off}} < min(v_{off}, |v_{on}|) \tag{3.22}$$

This design has the same advantages and disadvantages as a standard PLA. The main advantages include simple implementation and flexibility in programming. Furthermore, the PLMA is expected to be smaller than the PLA due to the PLMA since memristors can be smaller and have higher packing density.

However, area utilisation is not efficient since the logic implemented is usually in canonical form (sum of misterms). Although the pull-up and pull-down resistors enable non-canonical form logic to be implemented to a certain extent, they are usually not always present and have the drawback of additional power consumption in both static and dynamic operating conditions. Compared to a traditional PLA, the PLMA has additional static power consumption which is dominated by R_{off}.

The main issue of this circuit involves determining the working voltages. It is required that the voltage be compatible with the CMOS circuitry but also satisfy 3.21 or 3.22. Therefore, a device with a high voltage or current threshold is also desired. Additionally, since like MRL it operated on the principle of potential division, a high R_{off}/R_{on} ratio is

preferred. Ideally R_{off} would be massive such that static currents are minimised, but such a device could be difficult to program, especially if it is current controlled.

3.7 Conclusion

This background research shows there are many interesting logic families to consider. In general, they can be categorised based on how they represent the logic level; it can either be represented by the resistance level of the memristor, or the voltage level as in standard CMOS logic. The MRL family falls in to the latter category, and this is one of its greatest advantages, i.e. it is easier to integrate into existing CMOS designs since it does not require additional read/write circuitry to interface with the memristors.

However, representing the logic level by the memristance has its own advantages. For example, the Implication and MAGIC families can fully take advantage of the high packing density of the memristors in the crossbar form allowing for significant reduction in size. Furthermore, this crossbar structure is the same as memristor based memory. Therefore there is possibility for advanced architectures beyond the classical Von-Neuman and Harvard (for these the data storage and logical processing is separated).

Based on the research, the MRL, Implication Logic and MAGIC families are selected for further investigation. These are the most interesting out of the three and there remain many unanswered questions due to lack of published simulations of these families which use voltage controlled models. A comprehensive comparison based on research and simulations results presented later in section 6.5.

The memristor based PLA is not investigated further as its operation is relatively straight forward since it operates on the principle that the memristors only change state during the programming phase. The advantages and disadvantages and memristor requirements have already been presented in section 3.6.

4 Memristor Modelling

4.1 Introduction

Due to the variety of practical memristive devices and their minuscule dimensions, a perfect model based on the fundamental physics is difficult to develop. [50] [51] Even if such a model were to exist, it would likely be computationally inefficient and hence unsuitable for practical design simulations. Since there is no widely accepted single model to describe all memristors, and there is no perfect model to describe even a single type of memristor, this section covers a range of popular mathematical models and their variants.

The models are based on the fundamental operation of HP's TiO_2 memristor, but some models are flexible enough to fit different types of memristors. For example, the VTEAM model has been accurately fitted [28] to a $Pt\text{-}Hf\text{-}Ti$ memristor [52], a ferroelectric memristor [53], and a single component metallic nanowire memristor. [54]

In general, a memristor model should ideally satisfy the following criteria:

1) It should be adaptable to fit a range of practical memristors (of the same type at least)
2) It should be sufficiently accurate
3) It should be computationally efficient

4.2 Linear Ion-Drift Model

The linear ion-drift model introduced by HP labs [4] is based on the physical structure of their TiO_2 resistive switching device. In this model it is assumed the device of width D is split into 2 sections as illustrated by Figure 37 below. One section of width x is highly doped with oxygen vacancies (TiO_{2-x}), which are positively charged ions, and the other part is pure un-doped titanium dioxide. The doped section has a much lower resistance compared to the un-doped section, and the total resistance can be modelled as a weighted sum of R_{on} and R_{off} as specified byequation 4.1 below. [4] [36] [55] The variable x has been chosen to represent the doped section width since this width can be considered to be the state variable of the memristive system as described in section 2.2. The current voltage relation of this model is then given by equation 4.1.

$$v(t) = \left(R_{on}\frac{x(t)}{D} + R_{off}\frac{D - x(t)}{D}\right) i(t) \tag{4.1}$$

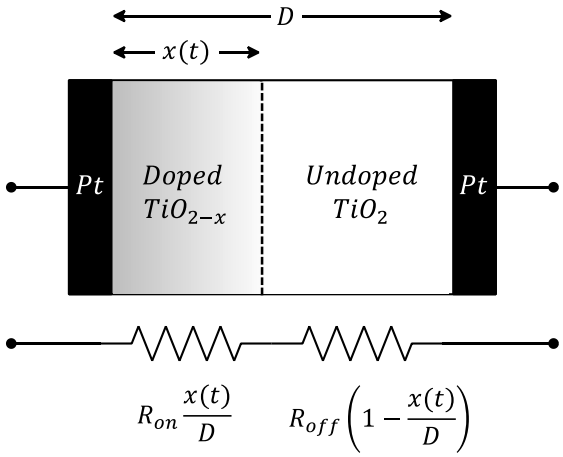


Figure 37: HP's initial model of their TiO_2 memristive device. Modified from [4] [36]

Since the width D is only a few nanometres, even a small applied voltage results in a strong electric field. Hence, when a positive voltage is applied to the side with the oxygen deficiencies, the electric field will repel the positive oxygen vacancies into the TiO_2 section, essentially changing the state x. When a negative voltage is applied, the opposite occurs, and when no voltage is applied there is no change. [36] Assuming the ion drift to be linear, and a uniform electric field, and a constant uniform oxygen vacancy concentration, and ohmic conductance, the state equation is given by: [4]

$$\frac{dx}{dt} = \mu_v E = \mu_v \frac{R_{on} i(t)}{D}, \qquad or\ equivilently \qquad x(t) = \mu_v \frac{R_{on}}{D} q(t) \tag{4.2}$$

Here μ_v is the average mobility of the ions, and E is the electric field that appears across the TiO_{2-x} section in association with the current $i(t)$. This model actually replicates the flux-charge relation of $f_\varphi(q) = q^2$ as described earlier in section 2.1.

Since the state variable x represents the physical width of the doped section, it must be bounded, i.e. $x \in [0, D]$. This bounding can be achieved by modulating the derivative in equation 4.14 by an ideal rectangular window function which essentially zeros the derivate as it reaches the boundary; thus preventing any further change in state.

However, it can be advantageous to use a nonlinear window since there is strong evidence which shows the drift of ion vacancies to be highly nonlinear. This is particularly relevant when close to the boundaries. [56] [**57**] [58] Here it has been observed that the speed of movement of the width of the doped section (i.e. $\frac{dx}{dt}$) gradually decreases to zero. This called nonlinear dopant drift and can be modelled by a nonlinear window. [50]

One such window proposed by Joglekar and Wolf [59] is described by equation 4.3 and illustrated by Figure 38. Here p is an integer parameter which controls the linearity of the window and as $p \to \infty$, the window tends to an ideal rectangular window.

$$f(x) = 1 - \left(\frac{2x}{D} - 1\right)^{2p} \tag{4.3}$$

A serious issue with using this window is that the model can suffer from a '*state freeze'* effect. Once the state reaches the bounds, its derivative is forced to zero regardless of the current. This means the state is frozen and cannot change even if the current changes direction. According to equation 4.3, it is not actually possible to reach the boundaries in finite time with finite currents and finite p. However, simulators have finite precision and are discrete in time and in this case it is possible to reach $x = 0\ or\ x = D$. To solve this, Biolek [50] proposes a window which utilises the memristor current, $i(t)$, in the window equation:

$$f(x) = 1 - \left(\frac{x}{D} - H(-i)\right)^{2p} \qquad where \qquad H(i) = \begin{cases} 1, & i \geq 0 \\ 0, & i < 0 \end{cases} \tag{4.4}$$

For clarity this is illustrated by Figure 39 for $p = 2$. This window essentially means that it is increasingly difficult for x approach a boundary, but easy to back away once the current is reversed.

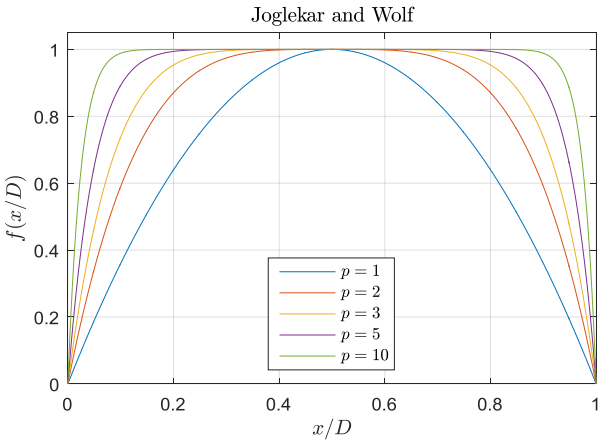


Figure 38: Joglekar and Wolf window for various p.

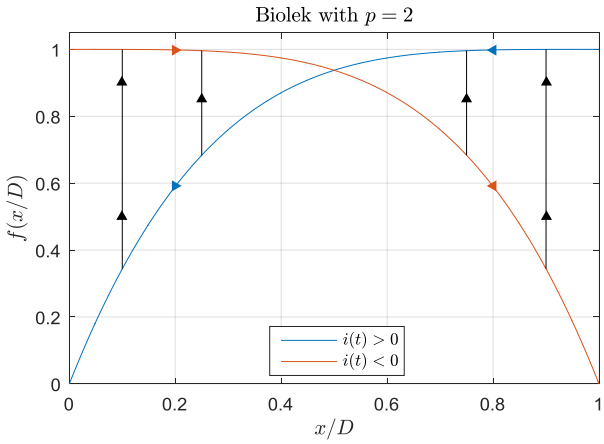


Figure 39: Biolek window with $p = 2$

Prodromakis [60] proposes a more flexible window with more control parameters $p, j \in \mathbb{R}^+$ as described by equation 4.5 and illustrated by Figure 40. Here, parameter p controls the linearity and j the scaling. In [60] Prodromakis shows that for all practical purposes this window does not suffer from the state freezing problem even though it does not introduce a discontinuity like the Biolek window.

$$f(x) = j\left(1 - \left(\left(\frac{x}{D} - 0.5\right)^2 + 0.75\right)^p\right) \tag{4.5}$$

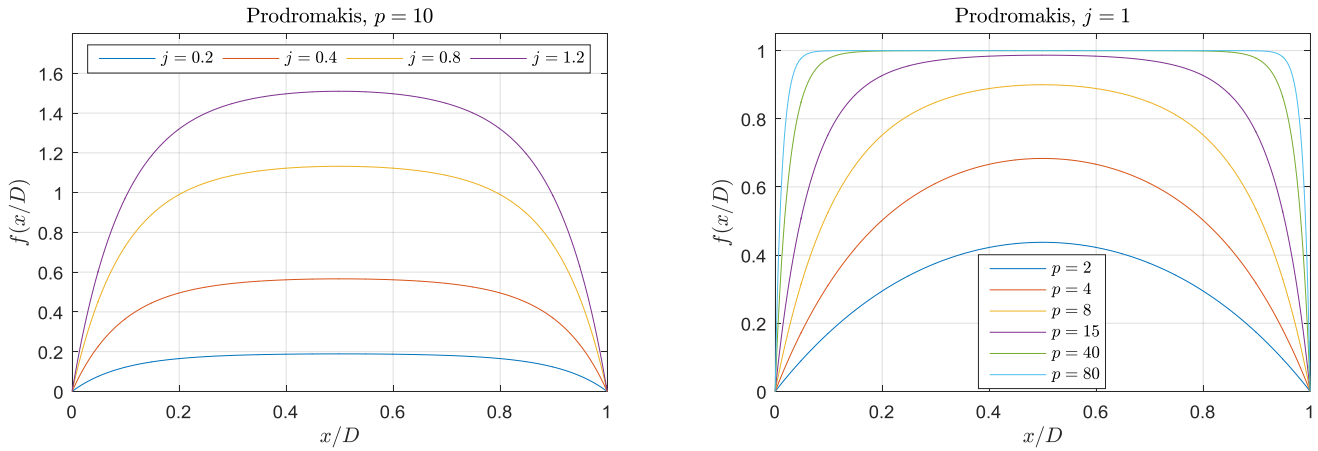


Figure 40: Prodromakis window for various parameters p, j.

A summary of some common window functions is presented in the conclusion (section 4.8) since more windows specific to some models are introduced in later sub-sections. Although the windows presented so far restrict x to the range $[0, D]$, and introduce some nonlinearity, they do not fully capture the nonlinearity of the IV relation as measured in practical devices. This leads to the improved nonlinear ion-drift models presented next.

4.3 Nonlinear Ion-Drift Model

Although the linear ion-drift model is simple and efficient and it satisfies the equations of memristive systems as described in section 2.2, it is inaccurate. Experiments have shown that practical memristors are significantly nonlinear and the error of the linear model is considerably high. [61] [57] The linear models have over simplified the electrodynamics. Even small voltages can produce a huge electric field due to the nanometer dimensions; this results in nonlinearities in ionic transport. [50] [62] This project focuses on digital logic applications, and as explained previously, nonlinear devices are preferred in some cases. Hence it is important to develop a more accurate nonlinear model to aid in the design process.

Lehtonen and Laiho [63] propose a model developed from experimental results from [61]. In this model, both the current-voltage relation (4.6) and the state equation (4.7) are nonlinear functions of voltage.

$$i(t) = \big(x(t)\big)^n \beta sinh\big(\alpha v(t)\big) + \chi\big(e^{\gamma v(t)} - 1\big) \quad 4.6$$

$$\frac{dx}{dt} = \alpha\, f(x)\, \big(v(t)\big)^m \quad 4.7$$

Here the state variable is normalised and bounded in the interval $[0, 1]$, and so $f(x)$ in equation 4.7 is a continuous window function such that $f(x) = 0\ \forall\ x \notin [0,1]$. In the state equation 4.7, m is an odd integer and α is a constant. All other unmentioned parameters are additional constants chosen to allow close fitting to experimental data.

One observation is that switching in this model is asymmetric. When $x \approx 1$, the device is close to the R_{on} state, and the current is dominated by the first term in 4.6. This term represents a tunnelling phenomenon. When $x \approx 0$, the second term dominates, and this is of the form of the ideal diode equation. The final observation is that the state is determined by voltage (or flux) rather than current as in the previous models. [64]

4.4 Simmons Tunnel Barrier Model

A more physically accurate model is proposed by Pickett et al. where the memristor is modelled by an electron tunnel barrier in series with a resistance R_s. [65] Here, the state variable x represents the width of the Simmons tunnel barrier as shown by Figure 41 below.

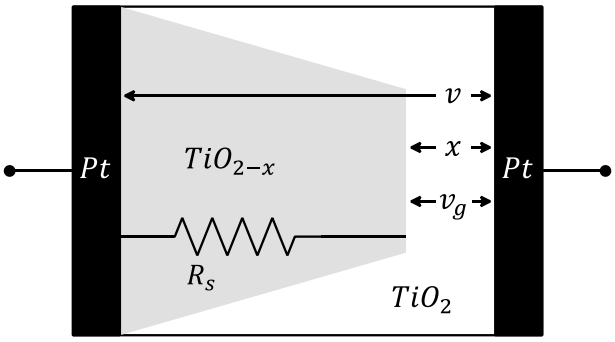


Figure 41: Physical model of a Simmons tunnel barrier memristor. Modified from [65] [64]

The state equation is shown below, and the time derivative of x is still interpreted as the drift velocity of the oxygen vacancies. [65]

$$\frac{dx}{dt} = \begin{cases} c_{off} \times \sinh\left(\dfrac{i}{i_{off}}\right) \exp\left(-\exp\left(\dfrac{x - a_{off}}{w_c} - \dfrac{|i|}{b}\right) - \dfrac{x}{w_c}\right), & for\ i > 0 \\ c_{on} \times \sinh\left(\dfrac{i}{i_{on}}\right) \exp\left(-\exp\left(-\dfrac{x - a_{on}}{w_c} - \dfrac{|i|}{b}\right) - \dfrac{x}{w_c}\right), & for\ i < 0 \end{cases} \quad 4.8$$

All the unmentioned parameters are constants used for fitting with experimental data. The physical phenomena explaining equation 4.8 are not yet fully understood, but it is possibly some mixture of the two following effects: nonlinear drift due to high electric fields, and local joule heating of the junction which enhances the thermally activated drift of the oxygen vacancies. [65]

For practical memristors switching from R_{off} to R_{on} is faster than switching from R_{on} to R_{off}. This is because when the voltage applied is positive (increasing x, and increasing resistactnce) the drift and diffusion of oxygen vacancies are in the same direction. For the opposite case, drift and diffusion are in the opposite direction meaning switching from R_{on} to R_{off} is slower. [66]. As a consequence the parameter c_{on} (which sets how fast the state variable x changes with current) is an order of magnitude higher than c_{off}. The parameters i_{on} and i_{off} are essentially current thresholds since below these values the change in state is negligible. The parameters a_{on} and a_{off} form lower and

upper bounds on the state variable and eliminate the need for windowing functions since the exponential function ensures the change in state is negligible when x is far out of range. [64]

The current voltage relation given by an implicit equation based on Simmons tunnelling. [67] A more functional form is shown below. [68]

$$i(t) = \frac{eA}{2\pi h \Delta w^2}\left[\begin{array}{c}\phi_1\left(x, v_g(t)\right) exp\left(-B\left(x, v_g(t)\right) \times \left(\phi_1\left(x, v_g(t)\right)\right)^{0.5}\right) - \\ \left(\phi_1\left(x, v_g(t)\right) + e\left|v_g(t)\right|\right) \times exp\left(-B\left(x, v_g(t)\right) \times \left(\phi_1\left(x, v_g(t)\right) + e v_g(t)\right)^{0.5}\right)\end{array}\right] \quad 4.9$$

Where,

$$\phi_1\left(x, v_g(t)\right) = \phi_0 - e\left|v_g(t)\right|\left(\frac{w_1 + w_2}{2x}\right) - \left(\frac{1.15\lambda}{\Delta w}\right) ln\left(\frac{w_2(x - w_1)}{w_1(x - w_2)}\right), \qquad B\left(x, v_g(t)\right) = \frac{4\pi \Delta w \sqrt{2m}}{h} \quad 4.10$$

$$\Delta w = w_2 - w_1, \qquad w_1 = \frac{1.2\lambda x}{\phi_0}, \qquad w_2 = w_1 + x\left(1 - \frac{9.2\lambda}{3\phi_0 + 4\lambda - 2e\left|v_g(t)\right|}\right), \qquad \lambda = \frac{e^2 \ln(2)}{8\pi\kappa\varepsilon_0 x} \quad 4.11$$

$v_g(t) = v(t) - i(t)R_s$	Voltage across tunnel barrier (see Figure 41)
x	State variable and width of the tunnel barrier (see Figure 41)
A	Channel area of memristor
h	Plank constant
ϕ_0	Barrier height in electron volts
m	Mass of electron
e	Charge of electron
ε_0	permittivity of free space
κ	Dielectric constant
$w_1,\ w_2$	Represents the limits of the potential barrier at the Fermi level. They are given by the two real solutions to the equation: [69] $\phi(\chi) = \phi_0 - \frac{e v_g(t)}{x}\chi - \frac{1.15\lambda x^2}{\chi(x - \chi)} = 0, \quad where \quad \chi\epsilon[0, x]$

Table 11: Summary of variables and constants in the Simmons Tunnel Barrier model. .Equation 4.9 is implicit since $v_g(t)$ is a function of $i(t)$

Although this model is considered to be the most physically accurate, it is computationally expensive to simulate. Furthermore, Biolek et al. discuss some more practical issues with simulation of this model, namely numerical errors, convergence issues and nonphysical solutions. This is mostly due to the ambiguity of the current voltage relation; the equations for $w_{1,2}$ in 4.11 are only approximate solutions for the equation in Table 11. As a consequence this gives rise to an abrupt nonlinearity for $v_g(t) > \phi_0$ which causes simulation difficulties. Biolek et al. propose the following approximation to replace equation 4.9. [69]

$$i(t) = sgn\left(v_g(t)\right) k_1 k_2^{(1-x)}\left[sinh\left((k_3 + k_4 x)\left|v_g(t)\right|\right) + k_5\left(exp\left(k_6\left|v_g(t)\right|\right) - 1\right)\right] \quad 4.12$$

In equation 4.12 , k_{1-6} are fitting parameters for fitting to experimental data, and x is the state variable which represents the Simmons Tunnel Barrier width normalized to the range $x \in [0,1]$. Although this is still relatively

computationally expensive, it solves the convergence errors that arise when 4.9 is used. One further disadvantage is that this model is very specific in that it only fits a specific type of memristive device. These disadvantages lead to the TEAM model discussed next.

4.5 Threshold Adaptive Memristor (TEAM) Model

The TEAM model developed by Kvatinsky et al. [64] is a simplification of the Simmons tunnel barrier model. It is intended to be more computationally efficient and flexible whilst maintaining sufficient accuracy. The state equation (eqn 4.13) is derived from a Taylor series expansion of the Simmons tunnel model's state equation (eqn 4.9), and explicit current thresholds (i_{on}, i_{off}) are introduced. Due to the simplifications, there is now a polynomial dependence on current of the state change rather than an exponential one. An explicit threshold is introduced to help mimic the exponential dependence in the Simmons model, i.e. in the Simmons model the state change is negligible for current smaller than the 'thresholds'. This is analogous to the Metal Oxide Semiconductor (MOS) Transistor voltage threshold simplification. [64]

$$\frac{dx}{dt} = \begin{cases} k_{off}\left(\frac{i(t)}{i_{off}} - 1\right)^{\alpha_{off}} \times f_{off}(x), & for\ 0 < i_{off} < i \\ 0, & for\ i_{on} < i < i_{off} \\ k_{on}\left(\frac{i(t)}{i_{on}} - 1\right)^{\alpha_{on}} \times f_{on}(x), & for\ i < i_{on} < 0 \end{cases} \quad 4.13$$

In equation 4.13, x is the state variable which still represents the effective Simmons tunnel barrier width, and k_{off}, k_{on} are positive and negative fitting parameters respectively. $f_{on}(x)$, and $f_{off}(x)$ are window functions which constrain the state such that $x \in (x_{on},\ x_{off})$. Alternatively, since the Simmons tunnel model equation 4.8 is a function of two variables (i, x), and the TEAM model is intended to just be a simplified version of this model, the optimal $f_{on}(x)$ and $f_{off}(x)$ can be found based on the separation of variables of equation 4.8. The results are shown below:

$$f_{off}(x) = \exp\left(-\exp\left(\frac{x - a_{off}}{w_c}\right)\right) \quad 4.14$$

$$f_{on}(x) = \exp\left(-\exp\left(-\frac{x - a_{on}}{w_c}\right)\right) \quad 4.15$$

Any unmentioned parameters are constants for fitting with experimental data or other models. This state equation can be fitted to a range of memristive device types through choice of the various parameters, and it can also be asymmetric just like the Simmons tunnel barrier model. The current voltage relation can be selected to be a linear or exponential based on which fits the device the best:

$$v(t) = \left[R_{on} + \frac{R_{on} - R_{off}}{x_{off} - x_{on}}(x(t) - x_{on})\right] i(t) \qquad (linear\ current - voltage\ relation) \quad 4.16$$

$$v(t) = \left[R_{on} \times exp\left(\frac{\lambda \times (x(t) - x_{on})}{x_{off} - x_{on}}\right)\right] i(t) \qquad (exponential\ current - voltage\ relation) \quad 4.17$$

From a physics perspective, the exponential current voltage relation is more suitable since this better represents the tunnelling effect which is extremely nonlinear. [65] Note that for this relation we require the parameter λ to satisfy $R_{off}/R_{on} = e^{\lambda}$, where R_{off} and R_{on} are the resistances when the state x is at the boundaries x_{on} and x_{off}. Results from [64] show the linear model can be sufficiently accurate so it is still useful for when simulation speed is of importance.

The TEAM model can be fitted to replicate the Simmons tunnel barrier model with a small mean error of only 0.2%, but with a significant 47.5% reduction in runtime. (These figures have been determined by MATLAB simulations in [64], and the performance results are parameter and implementation dependant). Another advantage of the TEAM model is that through careful selection of the parameters, it can be fitted to (exactly) replicate other models such as the linear ion-drift model, and this is demonstrated in [64]. Some intuition is provided for the most of the model parameters by Table 12 below.

$Parameter$:	$Meaning$:
k_{on}, k_{off}	Controls the magnitude of dependence on current of the state change
$\alpha_{on}, \alpha_{off}$	Determines the linearity of the state equation
i_{on}, i_{off}	Current thresholds for on and off switching respectively
R_{off}, R_{on}, λ	'On' and 'off' state resistances
x_{on}, x_{off}	Minimum and maximum value of state variable. Can be selected represent 'distance' if chosen appropriately but does not have.

Table 12: Intuition for some of the fitting parameters of the TEAM model.

4.6 Voltage Threshold Adaptive Memristor (VTEAM) Model

Kvatinsky et al. extend their TEAM model to the VTEAM model which is essentially a voltage controlled equivalent model with an a explicit voltage threshold rather than a current threshold. [28] This is useful since experiments have shown that for several types of memristors there exists some sort of threshold voltage behaviour. [4] [70] [71] For this model, the state equation is given by equation 4.18 below, which is similar to its TEAM counterpart (equation 4.13). The current voltage relation is of the exact same form as the TEAM model and is given by either equation 4.16 or 4.17 depending on if a linear or exponential relation is required.

$$\frac{dx}{dt} = \begin{cases} k_{off}\left(\frac{v(t)}{v_{off}} - 1\right)^{\alpha_{off}} \times f_{off}(x), & for\ 0 < v_{off} < v \\ 0, & for\ v_{on} < i < v_{off} \\ k_{on}\left(\frac{v(t)}{v_{on}} - 1\right)^{\alpha_{on}} \times f_{on}(x), & for\ v < v_{on} < 0 \end{cases} \quad 4.18$$

Here v_{on} and v_{off} are threshold voltages, and k_{off} and k_{on} are positive and negative numbers respectively. The window functions $f_{off}(x)\ and\ f_{on}(x)$ constrain the state x within the bounds x_{off} and x_{on} respectively which in turn correspond to $R_{off}$and R_{on}. Any other unmentioned parameters are constant fitting parameters determined from experimental data.

4.7 Thresholds

A difficult question to answer is if a certain device has a voltage or current threshold. This is also difficult to determine experimentally since there can be varying behaviour at different states. Furthermore, there can be some mixture of thresholds especially if there are multiple physical mechanisms occurring, for example, some chalcogenide based memristors have both metal ion conduction and phase change modes. [72] Another source of complexity arises from the observed time dependency of thresholds for some thin film devices. [73]

Figure 42 below shows the different characteristic *IV* curves for memristors with explicit current and voltage thresholds when driven by current and voltage sources. Comparing these with practical *IV* curves can help give a better idea on the threshold behaviour of a device. The top left plot shows a device with a current threshold being driven by a relatively low frequency sinusoidal current source. This has horizontal *IV* lines for when switching occurs.

In contrast, the voltage threshold device driven with a voltage source (bottom right plot) has vertical *IV* lines when switching occurs. The top right and bottom left plots are interesting because one of the switching *IV* lines is horizontal and the other is vertical. Considering first the current threshold voltage driven device switching from R_{on} to R_{off}. Once i_{off} is exceeded, the memristance starts to increase. Therefore a higher voltage is required in order to satisfy the current threshold such to continue the switching process and hence the switching *IV* line is horizontal. Now considering the switching from R_{off} to R_{on}. As soon as $i(t) < i_{on}$, the memristance starts to decrease and the current increases for a given voltage. Note that the voltage is relatively constant compared to the change in memristance since the voltage sinewave frequency is relatively low. Hence the switching *IV* line is vertical. These types of discussions can help explain the observed negative differential resistance as illustrated by Figure 43 below.

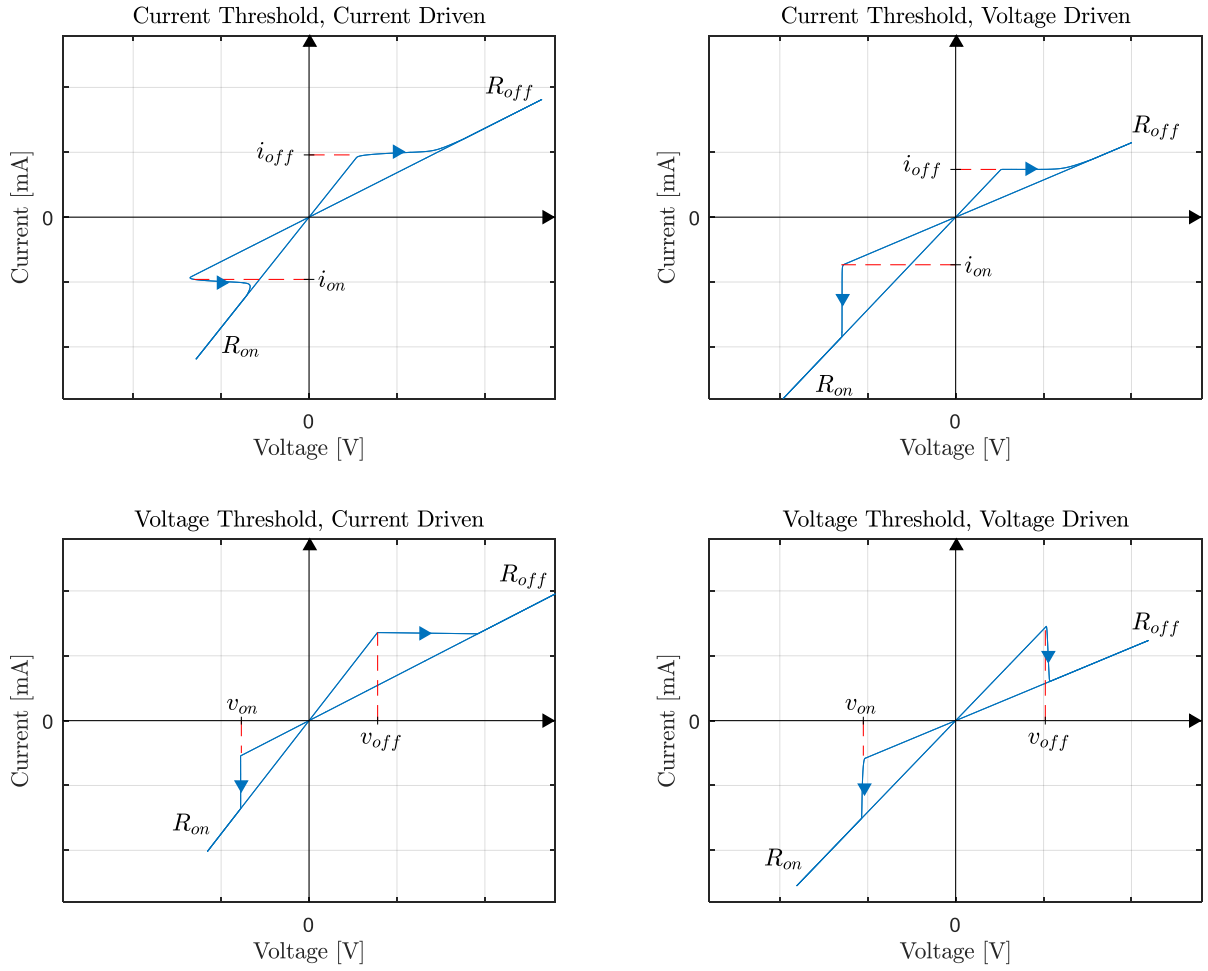


Figure 42: *IV* curves for a memristor with an explicit voltage and current threshold when excited by a relatively low frequency AC sinewave voltage or current source. These characteristic shapes can help identify the types of thresholds in practical devices. These have been obtained from the VTEAM and TEAM models as implemented in section 5.

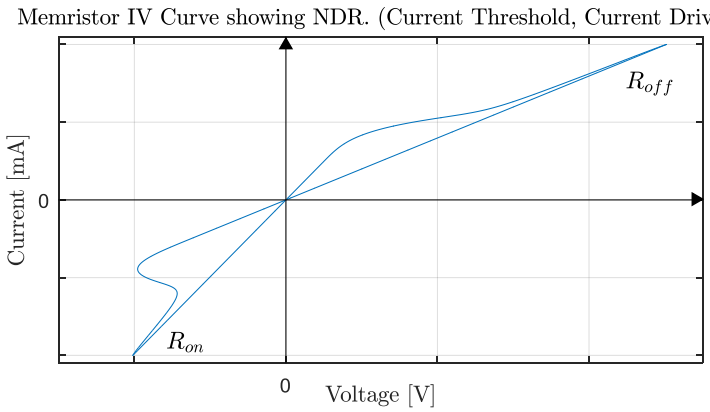


Figure 43: *IV* showing NDR. A frequency 100 times higher than that in Figure 42 is used.

4.8 Conclusion

Out of all the models discussed, the VTEAM model is selected for implementation, fitting and simulation. Although it is less accurate than the Simmons Tunnel Barrier model, it is more computationally efficient and has sufficient accuracy for the purpose of this project. Additionally, it also incorporates an explicit voltage threshold which better reflects the observed characteristics of the memristors available. Furthermore, it is more flexible compared to the Simmons Model in terms of fitting parameters and therefore should allow close fit to experimental data. Finally, logic family simulations using a model with an explicit voltage threshold will help fill in the gaps in published research.

Table 13 below summarises the discussion on various window functions. Since the VTEAM model is chosen, and the Simmons Tunnel Barrier model is considered most accurate, it makes sense to use the TEAM window for simulations. This is because the TEAM window is derived from a simplification of the Simmons Tunnel Barrier model.

Window:	*Window Function,* $f(x)$:	*Symmetric*:	*Scalable*:	*Suffers from State freeze*:	*Compatible with*:
Strukov [4]	$\frac{x(D-x)}{D^2}$	*Yes*	*No*	*yes*	*All*
Joglekar and Wolf [59]	$1-\left(\frac{2}{D}-1\right)^{2p}$	*Yes*	*No*	*yes*	*All*
Biolek [50]	$1-\left(\frac{x}{D}-H(-i)\right)^{2p}$	*Yes*	*No*	*No, but it is discontinuous*	*All*
Prodromakis [60]	$j\left(1-\left(\left(\frac{x}{D}-0.5\right)^2+0.75\right)^p\right)$	*Yes*	*Yes*	*Practically no*	*All*
TEAM [64] (*For fitting to Simmons Tunnel barrier model*)	$\exp\left(-\exp\left(\frac{\pm(x-a_{on,off})}{w_c}\right)\right)$	*Not necessarily*	*no*	*Practically no*	*TEAM VTEAM*

Table 13: Summary of common window functions. Extended from [64]

5 Spice Model Implementation

5.1 Introduction

The simulation tool used for this project is PSpice due to availability on the departmental machines. PSpice models for the linear ion-drift, nonlinear ion-drift, BCM and Simmons tunnel barrier model already exist and have been documented in Table 14 below. Verilog-A models exist for the TEAM and VTEAM models. Although Verilog-A is Spice compatible (e.g. with HSpice), PSpice does not support it, so PSpice macro-models are developed in this section. It is worth noting that Verilog-A models are generally more efficient and robust compared to macro-models.

Model Type	PSpice macro-model	Verilog-A
Linear Ion-Drift	[50] [74]	[75]
Nonlinear Ion-Drift	[63]	[75]
Simmons Tunnel Barrier	[68]	[75]
BCM	[76]	-
TEAM	Appendix A	[75]
VTEAM	Appendix A	[75]

Table 14: Sources of various simulation models.

5.2 VTEAM & TEAM PSpice Macro-models

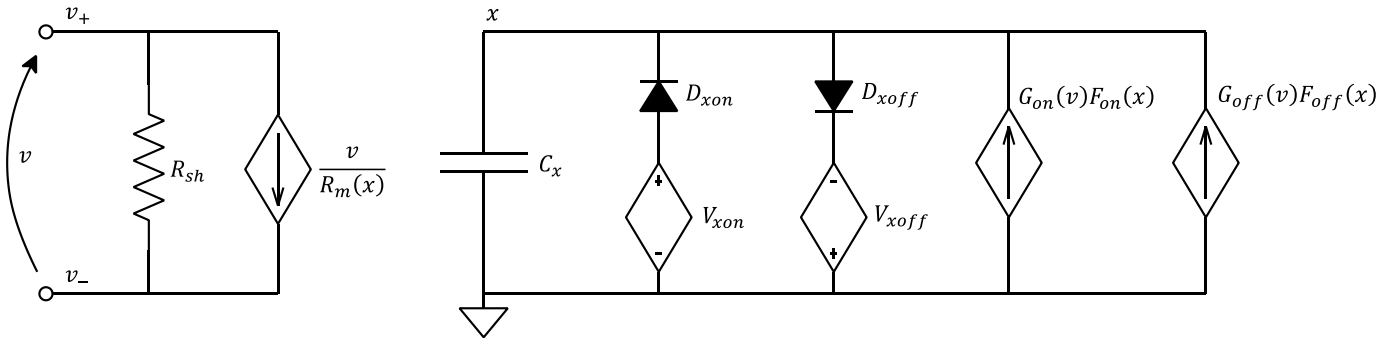


Figure 44: VTEAM PSpice macro-model.

The VTEAM macro-model is shown by Figure 44 above. Here, the capacitor section implements the state equation (eqn 4.18), with its voltage being the state variable, and the remainder implements the current-voltage relation (4.16 or 4.17). The equations for the dependant current sources are listed below. Note that the exponentials implemented are limited to prevent numeric overflows. However, since IEEE 64 bit floating point allows up to 1.79×10^{308}, overflow is unlikely and the limit is just precautionary.

$$G_{off}(v) = C_x \times k_{off}\left(\frac{v(t)}{v_{off}} - 1\right)^{\alpha_{off}} \quad for \quad v > v_{off}, \qquad G_{on}(v) = C_x \times k_{on}\left(\frac{v(t)}{v_{on}} - 1\right)^{\alpha_{on}} \quad for \quad v < v_{on} \tag{5.1}$$

$$f_{off}(x) = exp\left(-exp\left(\frac{x(t) - a_{off}}{w_c}\right)\right), \qquad f_{on}(x) = exp\left(-exp\left(\frac{-(x(t) - a_{on})}{w_c}\right)\right) \tag{5.2}$$

$$R_M(x) = R_{on} ex\, p\left(\ln\left(\frac{R_{off}}{R_{on}}\right)\left(\frac{x - x_{on}}{x_{off} - x_{on}}\right)\right), \qquad or \qquad R_M(x) = R_{on} + \frac{R_{on} - R_{off}}{x_{off} - x_{on}}(x(t) - x_{on}) \tag{5.3}$$

The concept to use a capacitor to store the state and to implement the state equation is taken from [51]. The diodes are the standard Dbreak model from the PSpice library, but they are modified to be as ideal as possible without causing convergence issues (See Appendix A). Their role (along with the sources V_{xon}, V_{xoff}) is to help constrain the state $x \in [x_{on},\ x_{off}]$. Although the window functions $f_{on}(x)$, $f_{off}(x)$ should mostly do this, the addition of the diodes results in more effective state containment. During model implementation, the following limitations of the simulation tool and design were identified:

1. Currents and voltages are limited to $\pm 10^{10} A, V$
2. Derivatives must be smaller than 10^{14}
3. The diode cannot be 100% ideal; there will be reverse leakage and non-zero forward bias voltage drop.

As a consequence of these, the following constraints for parameter selection are recommended when fitting the model to a practical device. These do not limit the fitting capability of this model since these constraints simply mean that the functions have to be appropriately scaled.

$Constraint$:	$Explanation$:
$x \in [0,1]$	The diodes have a small forward bias voltage drop. If the range of x is small, this diode drop (of the order of μV) cannot be neglected. If the range of x is too large then large currents will be required to switch the state especially if C_x is large. Therefore the dynamic range of x is conveniently chosen to be $1V$.
$10nF < C_x < 10^{10}\mathrm{t_s}$	A small capacitor reduces the need for large capacitor currents for high speed memristors. Therefore the upper bound of C_x is determined by maximum current $(10^{10} A)$ and the desired switching time (t_s) along with the range of x. If the capacitor is small, then the reverse leakage currents of the diodes $(< 1pA)$ cannot be neglected. Note that diodes have to be used to constrain the state as opposed to ideal switches because otherwise node x is 'floating' according to PSpice. A value of $10nF$ can be considered suitable for the lower limit guideline since it ensures the state x will drift by less than 1% even after 100 seconds.
$R_{sh} \gg R_{off}$	This shunt resistor forms a Norton circuit and allows this memristor model to be driven by a current source. R_{sh} should be much larger than R_{off} such to not affect the memristance $R_m(x)$

Table 15: Recommended fitting parameter ranges.

The TEAM PSpice macro model is shown by Figure 45 below. This is in very similar to the VTEAM model except that it is current controlled and has explicit current thresholds. The voltage source v_{is} is set to $0V$ and its purpose is simply to allow access to the memristor current. The small resistor R_s is added in series to prevent any voltage loops and it allows the model to be driven by a voltage source. This value should be selected to be much smaller than R_{on} such to not affect the memristive behaviour. The equations for the sources are listed below:

$$G_{off}(i) = C_x \times k_{off}\left(\frac{i(t)}{i_{off}} - 1\right)^{\alpha_{off}} \quad for \quad i > i_{off}, \qquad G_{on}(i) = C_x \times k_{on}\left(\frac{i(t)}{i_{on}} - 1\right)^{\alpha_{on}} \quad for \quad i < i_{on} \tag{5.4}$$

$$f_{off}(x) = exp\left(-exp\left(\frac{x(t) - a_{off}}{w_c}\right)\right), \qquad f_{on}(x) = exp\left(-exp\left(\frac{-(x(t) - a_{on})}{w_c}\right)\right) \tag{5.5}$$

$$R_M(x) = R_{on} ex\,p\left(\ln\left(\frac{R_{off}}{R_{on}}\right)\left(\frac{x - x_{on}}{x_{off} - x_{on}}\right)\right), \qquad or \qquad R_M(x) = R_{on} + \frac{R_{on} - R_{off}}{x_{off} - x_{on}}(x(t) - x_{on}) \tag{5.6}$$

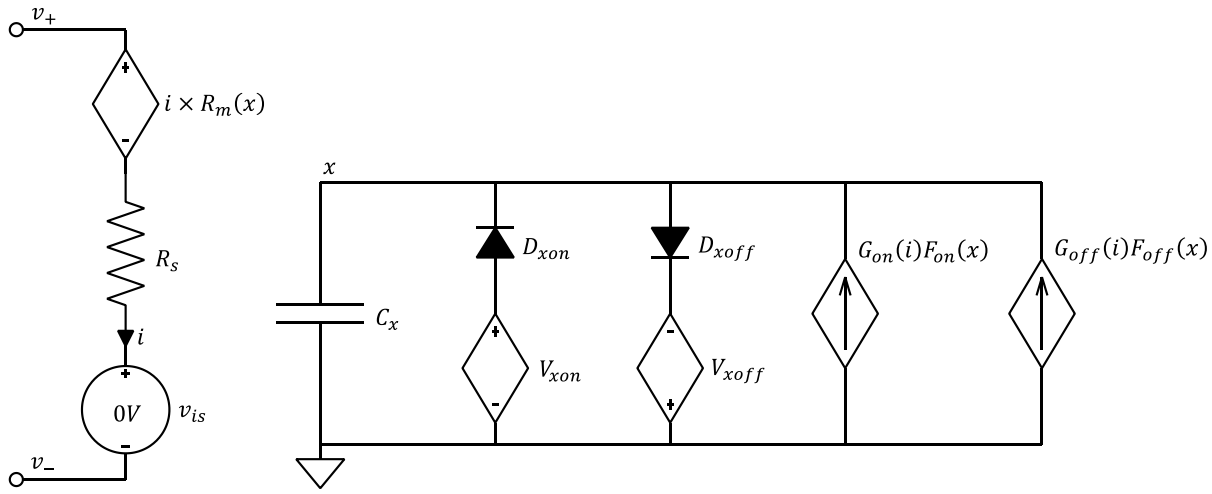


Figure 45: TEAM PSpice macro-model.

In order for the model to provide more useful simulation results, it needs to be first fitted to a practical device. In the next section the VTEAM model is fitted to IV curves obtained from a 60 µm by 60 µm $Ti0_2$ memristor with 25 nm oxide thickness.

5.3 Model Fitting, Modifications & Results:

Figure 46 below shows multiple static (no switching) IV curves for a 60 by 60 μm $Ti0_2$ memristor with 25 nm oxide thickness. The state is changed in between each run by manual pulsing. The bottom plot of Figure 46 shows the sequence of measuring pulses used to obtain each IV run. Each pulse is applied for 20 ms at different voltage levels, and the respective current is measured to form the IV curve. It is easy to see that the IV curve is in fact static because a full cycle of measuring pulses is applied and the area in-between each of the IV curves is negligible, i.e. there is negligible hysteresis.

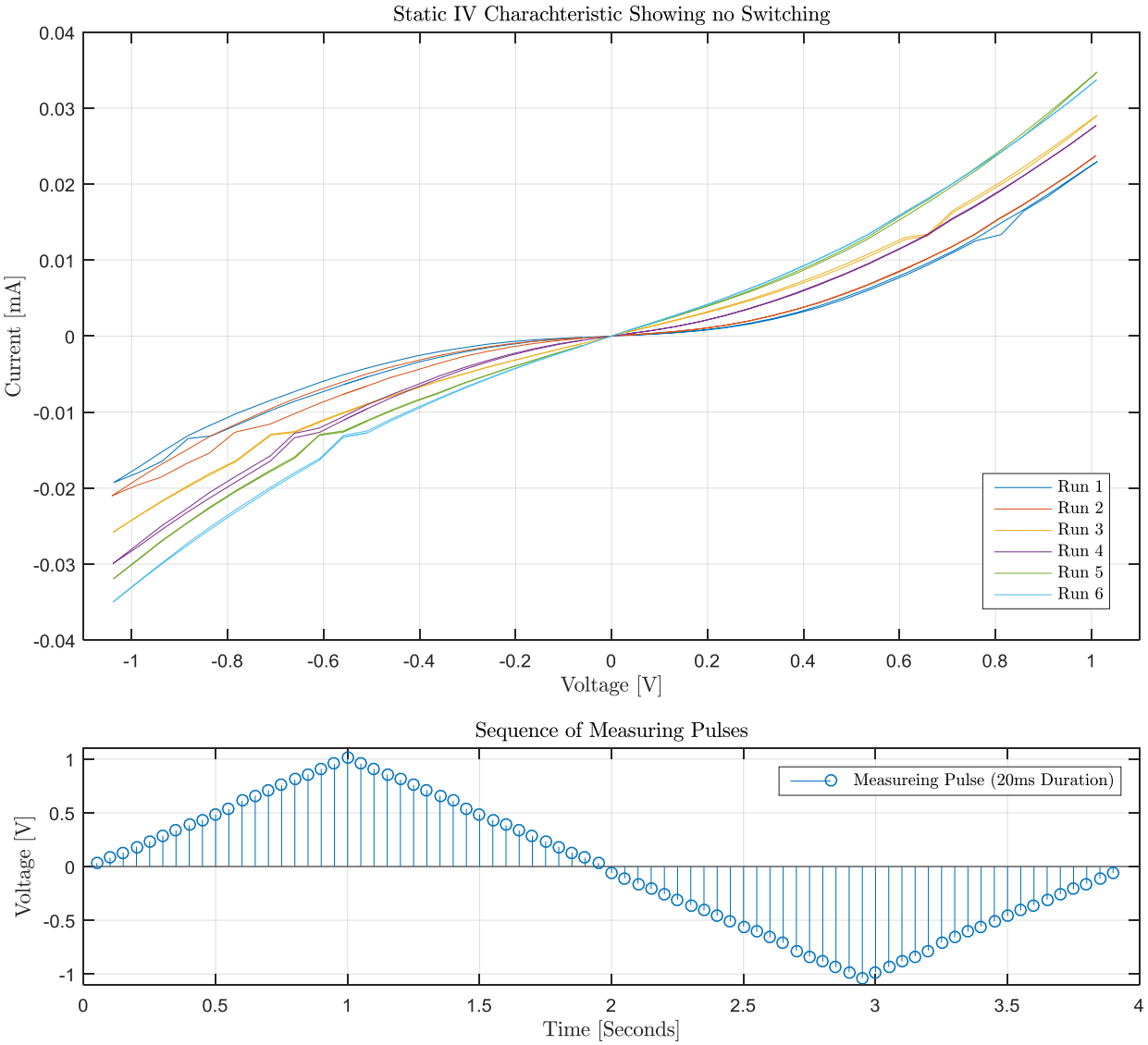


Figure 46: Bipolar static IVs taken for a 60 μm by 60 μm $Ti0_2$ device with 25 nm oxide thickness. Each IV is taken in between changing the state of the device using manual pulsing. The lower plot shows the sequence of measuring pulses applied in order to obtain each IV curve. Data courtesy of Radu Berdan, PhD student, Imperial College.

As expected changing the state results in different static IV curves. The more interesting observation is that each static curve is nonlinear; the device actually looks like two back to back diodes. This may be due to schottky junction behaviour at both the $Pt\text{-}TiO_2$ junctions. Based on this observation, the VTEAM model needs to be modified since it assumes a linear IV relationship for a fixed state (eqn 5.3). A simple modification to the model is developed next based on the experimental data.

One option is to use the nonlinear IV relation from the nonlinear ion-drift model (eqn 4.6) which is also derived based on experimental data [63]. The $sinh$ part of this relation would be able to model the shape observed in Figure 46. A simpler and more computationally efficient alternative is to use a polynomial consisting of odd powers. Even powers are not really necessary since the IV curves have a negligible even component. Through experimentation in MATLAB, it was found incorporating equation 5.7 into the VTEAM model caters for the behaviour seen in Figure 46. Here c_1, c_2 and c_3 are constants which are determined from the static IV curves.

$$i(t) = \frac{c_5 v^5(t) + c_3 v^3(t) + c_1 v(t)}{R_M(x)} \qquad (Modified\ VTEAM\ IV\ Relation) \tag{5.7}$$

Figure 47 below justifies the use of constant polynomial coefficients as opposed to coefficients which are each function of the state variable. It shows that for each *IV* run, the same coefficients result in small fitting error provided they are scaled appropriately for each state. This scaling can be achieved by $R_M(x)$ meaning that there is no need for each coefficient to be a function of the state variable which would make fitting stage more difficult. In Figure 47, they are found by first determining the polynomial for each *IV* run which minimises the mean square error, and then averaging the result. This results in $[c_3, c_2, c_1] = [-0.796, 2.1465, 1]$. Note that the VTEAM model included in Appendix A includes the aforementioned modification (eqn 5.7).

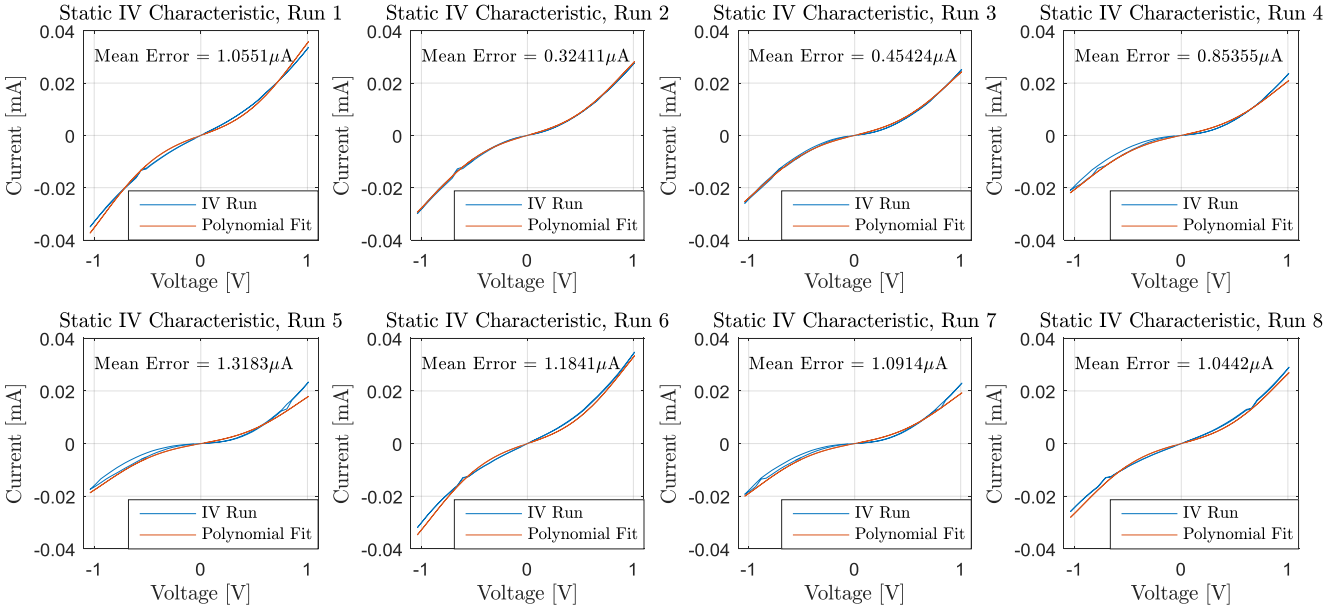


Figure 47: Justifying the use of constant coefficients $[c_3, c_2, c_1] = [-0.796, 2.1465, 1]$ in equation 5.7. These plots shows there is small error for all *IV* runs provided there is appropriate scaling of the coefficients for each run. This scaling can be done by $R_M(x)$, so there is no need for each coefficient to be a function of the state variable.

Figure 48 below shows an experimental IV curve with hysteresis for a memristor with when a triangular waveform voltage is applied. It also shows the fitted VTEAM model on the same plot. From this we can see the model is able to accurately fit the practical data; the relative mean square error (eqn 5.8) is 3.5%. The parameters determined are listed in Appendix C, and the steps below briefly summarise the steps developed for the fitting procedure:

1. Estimate the voltage thresholds by applying a series of pulses of increasing amplitude and by observing when the memristance change is non-negligible. This should be done for different initial states since this will give better idea if it is a current or voltage threshold.
2. Estimate R_{on} and R_{off} of the device by applying state changing pulses and measuring resistances.
3. Estimate the IV curve polynomial coefficients (for eqn 5.7) by minimising the MSE between multiple static IV curves for the devices. The MATLAB function polyfitcoef has been written for this very purpose (Appendix B). For this particular device it is easy to obtain static IV curves since the memristor is very slow. A relatively high frequency input waveform, say 150Hz, can be used to obtain an IV curve with negligible hysteresis.
4. The window parameters are chosen based on fitting to the Simmons tunnel barrier model as in [64].
5. The remaining parameters can be found by the applying the method of steepest decent to minimise the relative root mean square error which is defined by equation 5.8 below. Here, N is the number of sample points, and $v_{ref,k}$ and $i_{ref,k}$ are the k^{th} voltage and current samples respectively. $\|V_{ref}\|_2^2$ and $\|I_{ref}\|_2^2$ denote the Euclidean norm of the vector of voltage and current reference points respectively. [28] [77]

$$e_{rms} = \sqrt{\frac{1}{N}\left(\frac{\sum_{k=1}^{N}\left(v_{VTEAM,k} - v_{ref,k}\right)^2}{\left\|V_{ref}\right\|_2^2} + \frac{\sum_{i=1}^{N}\left(i_{VTEAM,k} - i_{ref,k}\right)^2}{\left\|I_{ref}\right\|_2^2}\right)} \quad 5.8$$

6. A simpler alternate approach to step 5 (which is what is used here) is to use the simulation tool to replicate the input voltage waveform which was used to obtain the experimental data points. Then iterative methods can be used to minimise the cost function given by 5.9. It is easiest to perform the iteration over one fitting parameter at a time, and then revisiting each parameter as necessary.

$$e_{rms} = \sqrt{\frac{1}{N}\left(\frac{\sum_{i=1}^{N}\left(i_{VTEAM,k} - i_{ref,k}\right)^2}{\left\|I_{ref}\right\|_2^2}\right)} \quad 5.9$$

In general this process is unlikely to result in a global minimum. Nevertheless the computed parameters result in a small fitting error as shown by Figure 48 below.

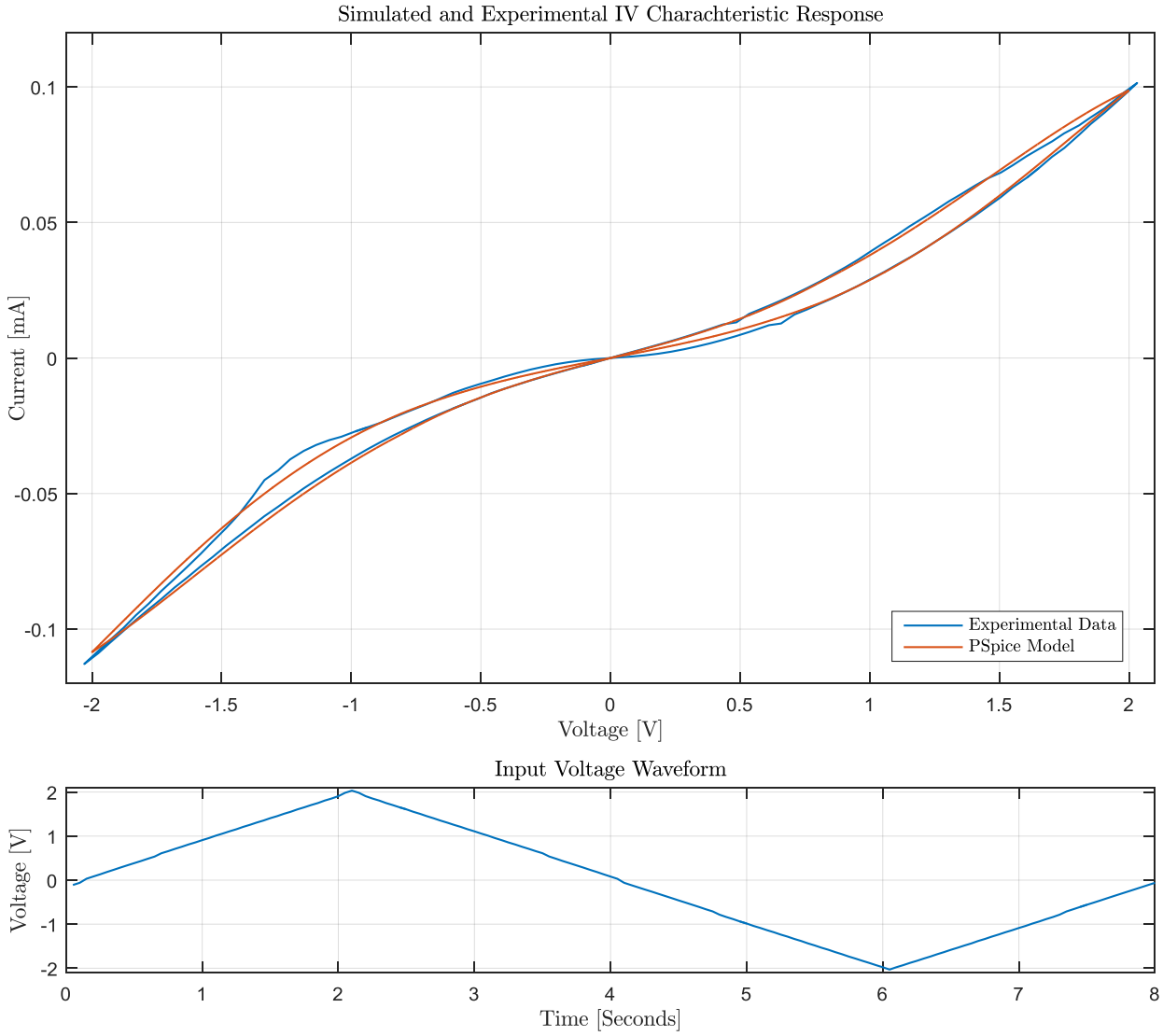


Figure 48: Model fitted to data from a 60 by $60\mu m$ $Ti0_2$ device with $25nm$ oxide thickness. Relative RMS error (eqn 5.8) is 3.5%, and the mean absolute error is 1.9×10^{-6}A. The parameters used for fitting are listed in Appendix C.

5.4 Testing

It is important to first verify the stability of the model before proceeding with simulations. This is because the model is a system with feedback in some operating cases, i.e. when it is driven by a current source, or when it is driven by a voltage source with series resistance. Under these cases the voltage across the memristor depends on the instantaneous memristance which in turn is also a function of the voltage and so a feedback path is present.

The feedback can be positive. For example, if a current source drives the model with a current in the direction such that memristance increases, then the voltage magnitude will increase and hence the rate of increase of memristance also rises and so on. However, the end result should be a stable state since the window functions and the diodes should eventually constrain the state variable.

The VTEAM model as described in section 4.6 is a continuous time nonlinear state space system. One approach to study the stability of this mathematical model could be to linearize the state space system at different operating regions and investigate the nature of the eigenvalues. For stability, they should all be in the left hand plane (or inside the unit circle for a discretized system). Note that asymptotic stability of the linearized system only guarantees local asymptotic stability of the nonlinear system and so all operating regions would have to be linearized and studied.

However, this method is not as simple as fist envisaged and has severe limitations. Most importantly, it only considers stability of the theoretical model but the main interest is the stability of the PSpice implementation. Additionally, it is more relevant to small signal analysis where the linearization is actually valid. This is defiantly not the case here since the memristor is expected to visit the extreme states since the application is digital logic.

Theoretical stability analysis of the Spice implementation is more involved due to the methods the tool uses for performing transient simulations. It makes heavy use of equivalent circuits, Newton-Raphson method, and numerical integration such as the trapezoidal rule. Also, the precision is finite and the time step varies during runtime. [78] [79] Therefore a simpler empirical but less thorough empirical method is used and this involves study the step response.

The step response to both positive and negative voltage and current inputs is shown by the figures below. These show a well behaved response with no sign of oscillations. Additionally there is no change in state when the input is zero (or sub-threshold) and, the state is effectively constrained in the desired range by the combination of window and diode action. Other amplitudes and starting states are also investigated and they all show no signs of oscillations. In conclusion, the fitted model shows no signs of oscillatory instability or ill-behaved response.

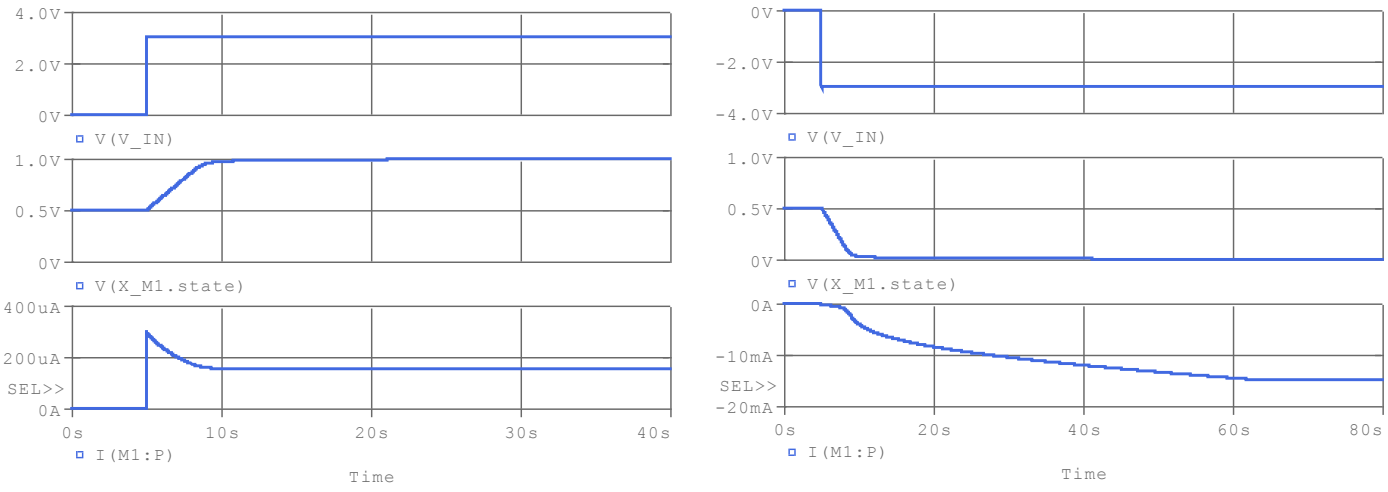


Figure 49: The upper row of plots show the voltage step input, the middle row shows the internal state variable and the lower row shows the current.

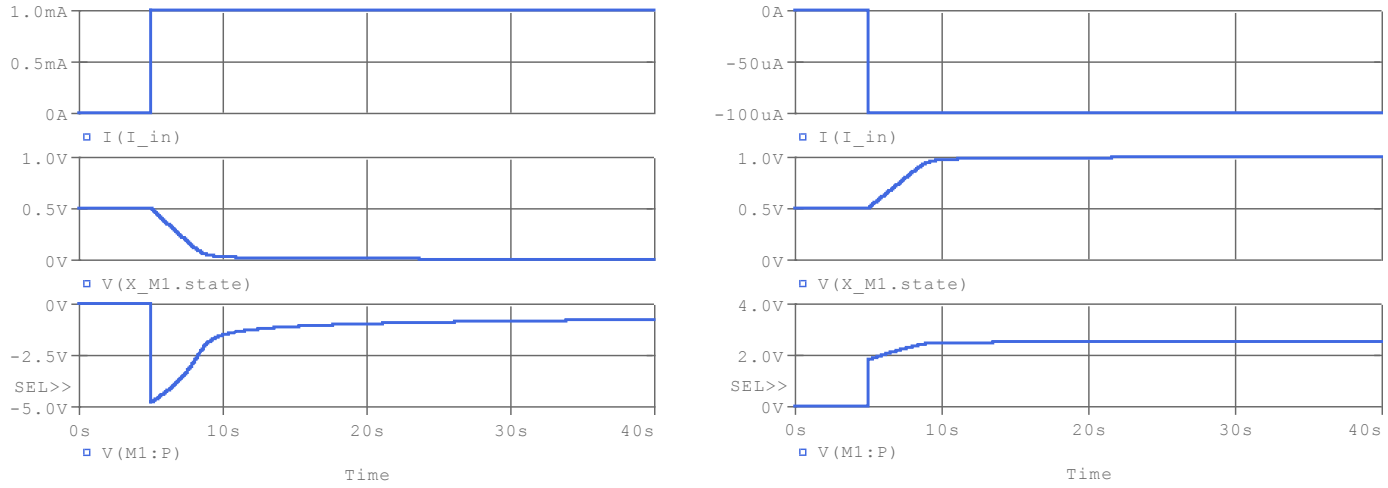


Figure 50: The upper row of plots show the current step input, the middle row shows the internal state variable and the lower row shows the voltage.

5.5 Evaluation

5.5.1 Validity and Accuracy

This model has been fitted to experimental data with sufficient accuracy; the relative RMS error is 3.5% which is comparable to fittings achieved in [28]. However, it is worth noting the limitations of this model before proceeding with simulations. This model does not fully represent the physics of any memristor since it is designed to be general such that it can fit a range of device types (also the physics of this memristor is not completely understood). As a consequence the accuracy is inherently limited.

For example fitting a model to a particular IV curve obtained by a particular input waveform does not guarantee that the model then fully represents the practical device even if the fitting error for that particular curve is close to zero. It may be that applying a different input to the fitted model and the actual device results in a larger than expected error. However, the most of the general behaviour of the device can still captured and so model fitting and simulations can still provide useful insight to memristor based logic design.

The simulations performed as part of this project are particularly useful since published research on logic family simulation results which use models fitted to practical devices is rare. Most published results use purely hypothetical model parameters [24] [26] [27] chosen simply to develop understanding on how various parameters such as linearity affect digital logic circuits. Others use models fitted to only a few data points. For example, the model parameters used in [25] are purely selected to ensure a switching time of $1ns$ with a $1V$ pulse is applied (as reported by practical devices in [80]).

5.5.2 Speed and Scaling

It is clear this particular device is not ideal for digital logic applications due to its incredibly slow switching speed and its unimpressive $R_{off}/\mathrm{R_{on}}$ ratio which is only in the region of 100. The memristor may become more useful as it scales down. For example, switching speed might increase as device area decreases because parasitic capacitance would decrease (if the oxide thickness remains the same).

However, capacitance is not considered to be the speed limiting factor for the given device. Furthermore, speed may not increase because subsequent switching after electroforming occurs in a $10 \times 10 \times 2\ nm^3$ volume close to one electrode and a formed conductive filament. What actually scales with device thickness is the voltage and time required for the electroforming process. Switching speed is a complex function of device materials, outcome of

electroforming, and temperature, and has not been investigated in depth as part of this project due to time constrains and priority of tasks. Therefore no clear conclusions are made on how scaling is expected to affect performance, and the model parameters remain un-scaled in the simulation section in order to avoid ruining the validity of the fitted model.

5.6 Conclusion

This section introduced the VTEAM and TEAM Spice implementations. The VTEAM model was fitted to practical memristor data and its stability was investigated. From the analysis and results in this section, it is concluded that the VTEAM model is suitable for use in the logic family simulations. This is because it shows reasonably good fitting accuracy and has no obvious stability issues. Although the fitted model switches very slowly, it can still give useful insight into logic family behaviour. The alternative would be to scale the model parameters based on how the devices is expected to perform once scaled to smaller sizes. However, this could invalidate the fitting procedure if not done correctly.

Although there are faster and more suitable devices, for example HfO_x and TaO_x have larger R_{off}/R_{on} ratios and switch in the region of nanoseconds [81], the fitted TiO_2 model was selected and used for simulations. This is because such devices were anticipated to be available for practical investigation and hence the simulations using this model would be most useful and relevant. Another reason was the availability of data which allowed the fitting to be performed.

6 Logic Designs, Simulations and Results

6.1 Introduction

This section builds on the research discussed in section 3 with the help of PSpice simulations which use the VTEAM model fitted to a TiO_2 memristor. The aim is to further develop the understanding of the operation of the logic families in order to find relative advantages, disadvantages and limitations. At the end of this section, a comprehensive comparison is performed based on the simulation results and the background research. The information gathered during this process will also help determine what type of memristor is preferred for each logic family.

First individual gates are investigated and if successful, various properties are discussed such as: speed, power, scalability, complexity, and device count (chip area). Next, a full adder design is attempted (based on the success of the gates). This will allow comparison to be made of more relevant and useful circuits. As it turns out, some families have limitations with the particular fitted model used which prevent the adder from being simulated. This somewhat restricts quantitative comparisons. Nevertheless, the experience gained during design and simulation allows detailed qualitative comparisons to be made.

6.2 MRL

First, individual MRL and hybrid CMOS gates are investigated under ideal circumstances (no load). The effect of loading and the need for buffering is then examined. Next, various XOR gates are constructed and compared in order to develop a full adder with the highest performance in terms of speed, power and area. A novel architecture for a hybrid XOR gate is proposed, and finally, the full adder is simulated and its performance inspected with a brief comparison with CMOS circuits.

6.2.1 Input Sequences

Section 6.2 includes many simulations of MRL gates and designs. These simulations run through all possible input combinations and so it makes sense to use the same input sequence when comparing the gates and designs. The purpose of this section is simply to define the input sequences used in section 6.2 in order to avoid repetition.

For any 2-input design, the logic input sequence given by 6.1 below is used. For any 3-input design, such as a full adder, the sequence given by 6.2 below is used. These simply run through all binary combinations in ascending order.

$$\{V_{in2}, V_{in1}\} = [\{0,0\},\{0,1\},\{1,0\},\{1,1\}] \quad 6.1$$

$$\{V_{in3}, V_{in2}, V_{in1}\} = [\{0,0,0\},\{0,0,1\},\{0,1,0\},\{0,1,1\},\{1,0,0\},\{1,0,1\},\{1,1,0\},\{1,1,1\}] \quad 6.2$$

The duration of each input set is 20 seconds, and $5V$ is used for logical 1 and $0V$ for logical 0. The points below justify the choice of a $5V$ amplitude:

1. $5V$ is a common CMOS voltage level, and as discussed in section 3.4, MRL relies on CMOS inverters to complete the logic basis so compatible voltages must be chosen.
2. Higher voltages result in faster switching, but also increase static and dynamic power consumption due to higher currents as discussed later.
3. If the voltage is too low, then the steady state output error due to the threshold is proportionally larger as discussed later.

Figure 51 below graphically summarises this information. A convenient MATLAB script to generate any desired input pulse sequence is included in Appendix B. This generates the text file which is used by the source 'V_PWL_FILE' in PSpice.

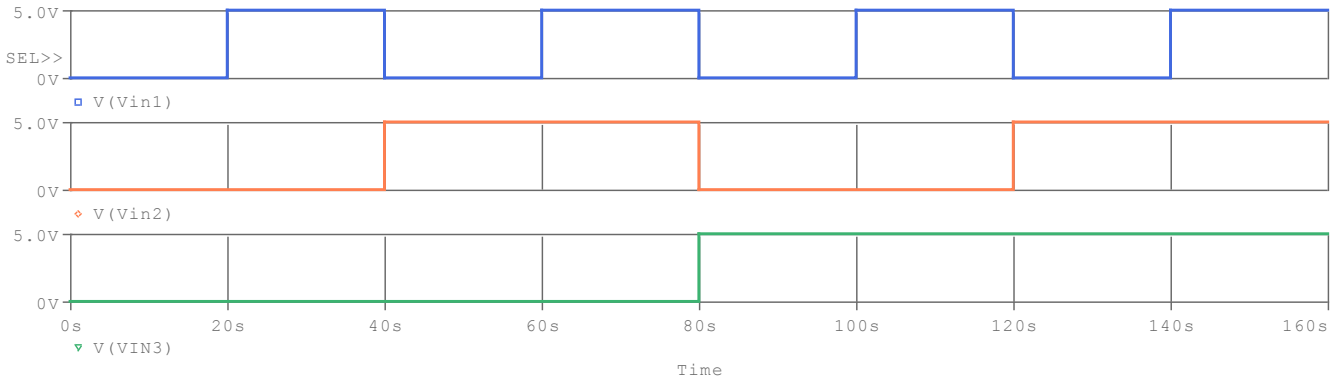


Figure 51: The input sequence of voltages applied to all gates and designs in section 6.2. The amplitude is $5V$.

6.2.2 Gates

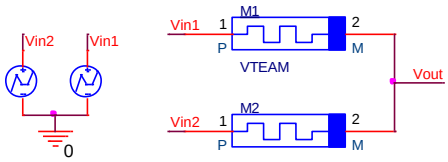


Figure 52: MRL AND gate with no load.

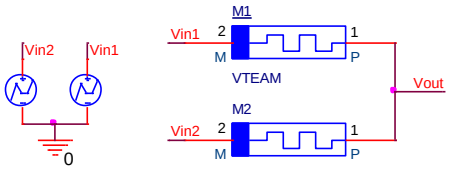


Figure 53: MRL OR gate with no load.

Figure 52 shows a 2-input MRL AND gate with no load, and Figure 53 shows a 2-input OR gate. The 'P' on the memristor symbol stands for the plus terminal, and 'M' for minus. This is consistent with conventions laid out in section 3.2 in that applying a positive voltage to the plus terminal will increase the memristance.

First the AND gate with no load is investigated. Figure 54 below shows the output voltage when the input voltage sequence is as described in section 6.2.1. From this it can be see that the response is far from ideal even in the ideal no load case. For a perfect gate, the output should be $0V$ between 0 and 60 seconds since during this time at least one input is zero.

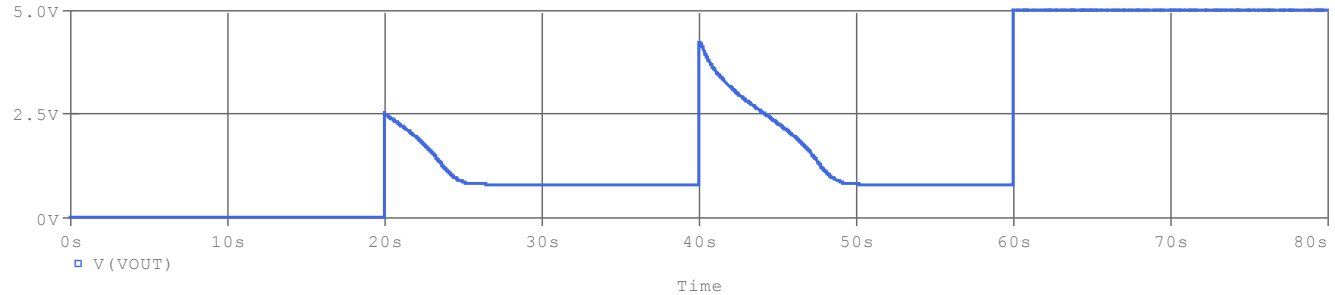


Figure 54: Output Voltage of the AND gate (Figure 52). This gate suffers from steady state error and dynamic hazards.

The first issue is the dynamic hazards which occur when transitioning the input logic values from $\{0,0\}$ to $\{0,1\}$ at 20 seconds, and from $\{0,1\}$ to $\{1,0\}$ at 40 seconds. This issue is expected based on the background research and is already explained in section 3.4.3; it arises because the memristors require non-zero time in order to change their resistance state which is necessary to produce the correct output based on potential division.

The second issue is the high steady state output error for both the input combinations $\{0,1\}$ and $\{1,0\}$. The model used has a R_{off}/R_{on} ratio of 100, and under the assumption of complete switching, the output is expected to settle to $5V \times R_{on}/(R_{off} + R_{on}) = 5/101 = 0.0495V$. However, Figure 54 shows the steady state output voltage to be just

under $0.8V$. This is much larger than expected and unsatisfactory according to CMOS standards; for a Vdd of $5V$, the output must be greater than $4.95V$ for logical 1 and less than $0.05V$ for logical 0. [82]

Figure 55 below shows the voltage across each memristor and it helps to explain this observation. It shows that whenever switching occurs, one of the memristors is always taken below the explicit threshold voltage. As a result, there can be no further switching and hence the ratio of R_{off}/R_{on} becomes less important in determining the output voltage. Instead the $|v_{on}|$ switching threshold voltage becomes the dominating factor. In this case $v_{on} = -0.8V$. A device with a current threshold can also suffer from a similar problem, but it is much less severe and full switching can occur provided $v_{high}/(R_{off} + R_{on}) > \min(i_{on}, i_{off})$ assuming a two input gate.

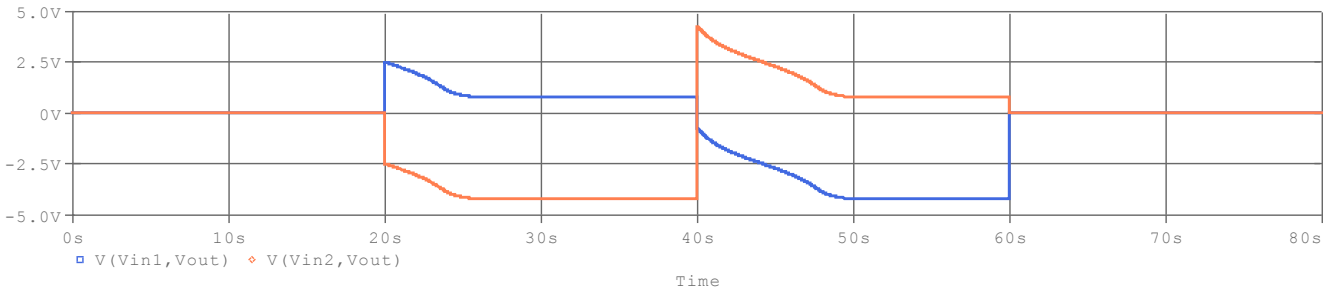


Figure 55: The voltage across each memristor for the AND gate. The blue line is the voltage across $M1$, and the orange line is the voltage across $M2$. Full switching of cannot occur since the voltage drops below the threshold before switching is complete

Figure 56 below shows the evolution of the model state variable for both memristors. Due to the voltage threshold, the lowest state value (0) can never be reached. This is because the low state value corresponds to a low resistance and as resistance decreases, the voltage across the device deceases until it falls below the threshold. The other memristor continues to fully switch to R_{off} and it is only restricted by the window function and the state constraining diode of the model rather than the circuit.

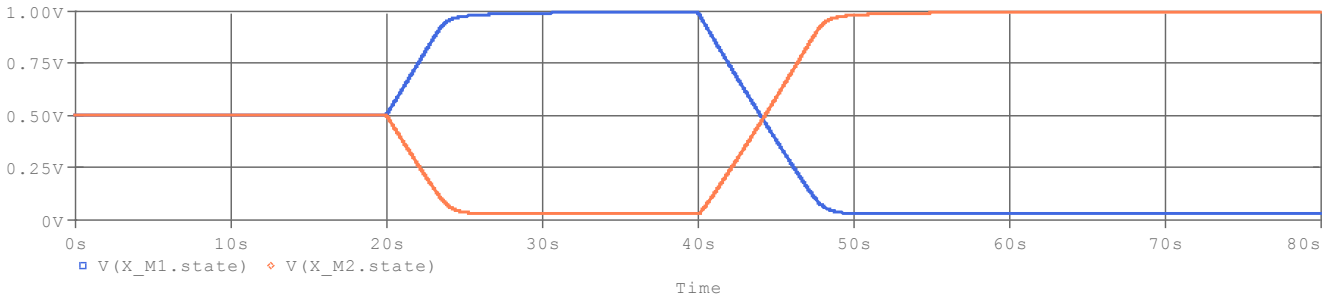


Figure 56: Shows the evolution of the state variable for both memristors for the AND gate. The blue plot is for $M1$, and orange for $M2$. State zero can never be reached in this logic gate due to the explicit voltage threshold in the VTEAM model.

Another issue is the high static power consumption. Figure 57 below shows the current flowing through the memristors from $M1$ to $M2$. The important point to note is that unlike CMOS logic, there is relatively high current flow even when the output is at a steady state. This is the case when the inputs are logic $\{0,1\}$ or $\{1,0\}$. Under these conditions, the sum of the memristor voltages is $5V$, and therefore the power consumption is directly proportional to this current. An alternate expression for power is given by equations 6.3 below.

$$P_{static} = \frac{{v_{high}}^2}{R_{on} + R_{off}} \qquad P_{dynamic}(t) = \frac{{v_{high}}^2}{R_1(t) + R_2(t)} \qquad 6.3$$

When the inputs are {0,0}, or {1,1}, then no current is drawn provided the output is unloaded. Figure 57 also shows the power consumption of the two memristors. If the intermediate results are stored (pipelining), then there is no need to have the circuit continuously active. The inputs only need to be powered during the computation stage, i.e. the time for the output to settle to the correct value, so the static losses can be minimised.

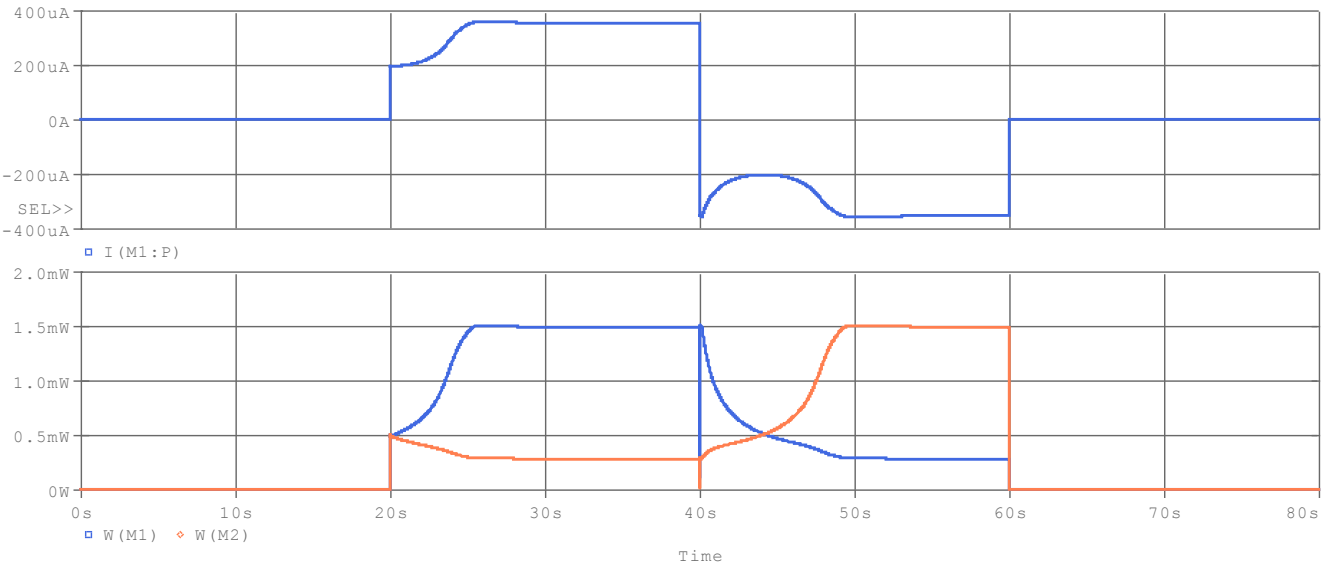


Figure 57: The top plot shows the shows the current from $M1$ to $M2$ for the AND gate. The bottom plot shows the power dissipation in the two memristors. Zero power is consumed when both inputs are the same. When the inputs are different, power is consumed even in the steady state.

Next the OR gate as shown in Figure 53 is investigated. Figure 58 below shows the output voltage for the pre-defined input sequence. This OR gate also suffers from dynamic hazards and steady state error, and this is due to the exact same reasons as explained for the AND gate. Ideally the output should be $5V$ between 20 and 80 seconds since this is the time period when at least one input is logical high. An interesting observation is that the for both the AND and OR gates, the steady state value is only influenced by the threshold v_{on} , and v_{off} is less important.

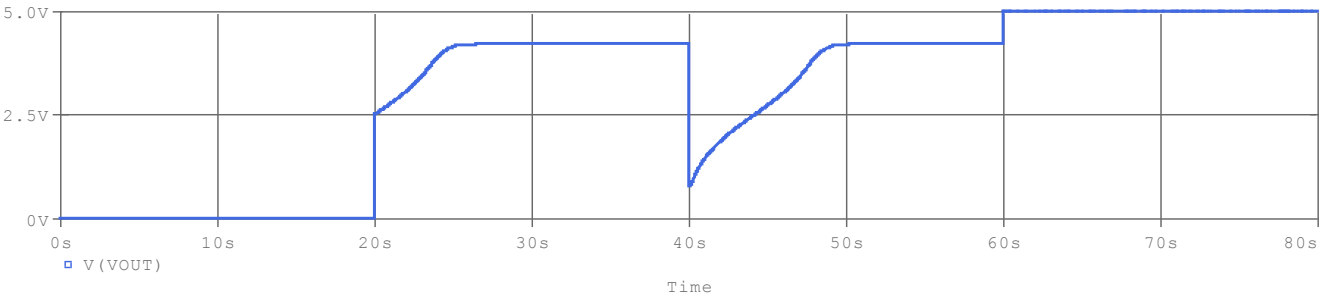


Figure 58: Voltage output for the OR gate (Figure 53). This gate also suffers from steady state error and dynamic hazards.

For completeness, the waveforms for memristor voltage, internal state variable, current and power are included below in Figure 59, Figure 60, Figure 61 respectively. Due to the symmetry of the two gates, the waveforms are the same shape but simply with the role of $M1$ and $M2$ interchanged. The current and overall power consumption is exactly the same for both gates.

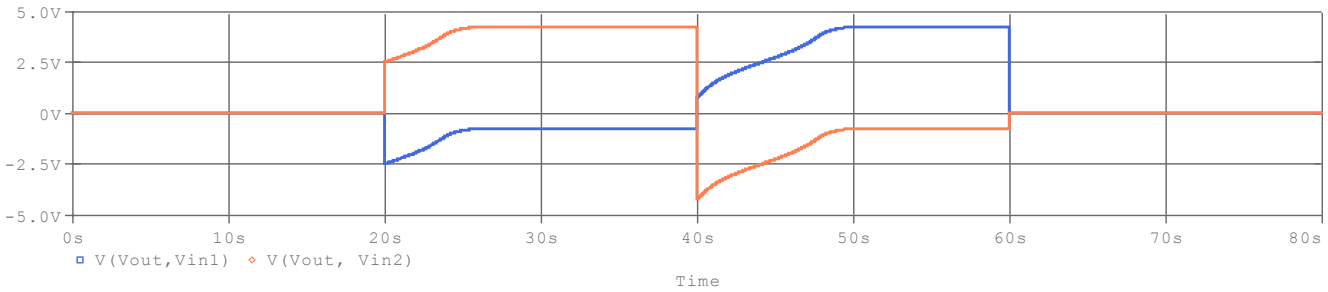


Figure 59: The voltage across each memristor for the OR gate. The blue line is the voltage across $M1$, and the orange line is the voltage across $M2$.

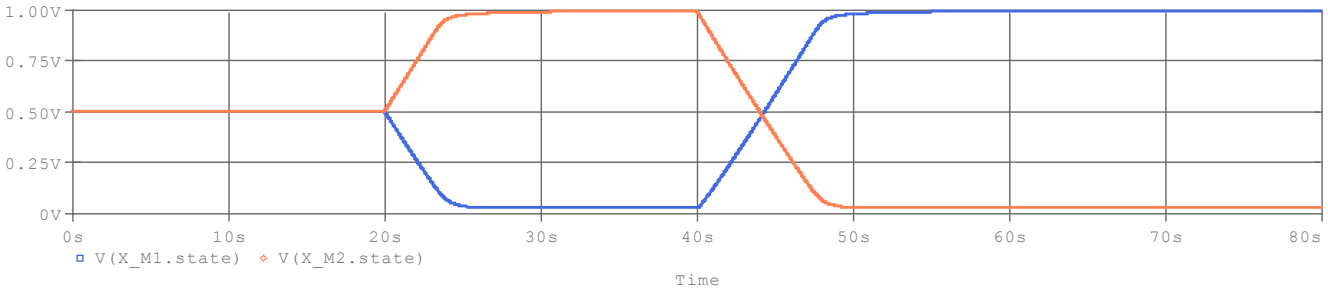


Figure 60: The state variable for both memristors for the OR gate

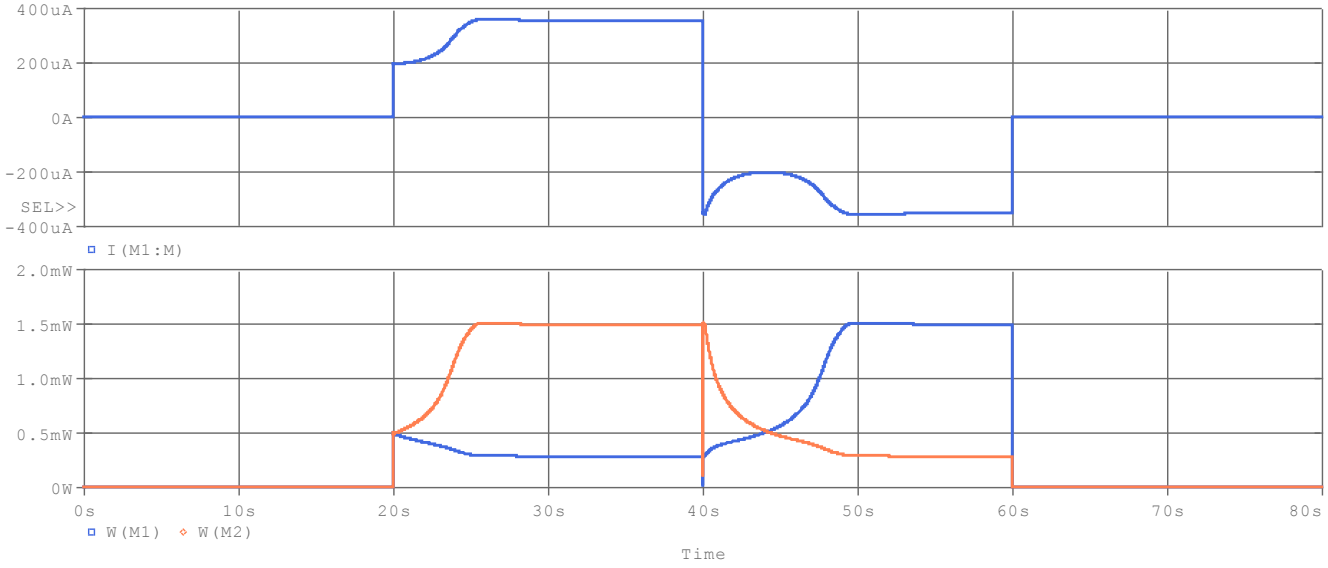


Figure 61: The top plot shows the shows the current from $M1$ to $M2$ for the OR gate. The bottom plot shows the power dissipation in the two memristors. The power curve is identical to that of the AND gate, but with the role of $M1$ and $M2$ interchanged.

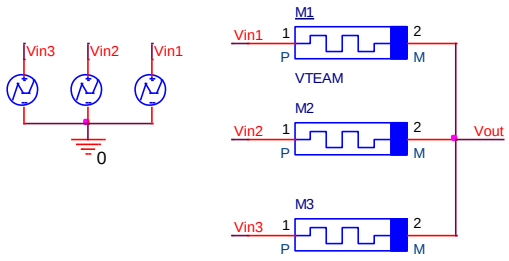


Figure 62: 3-input MRL AND gate with no load

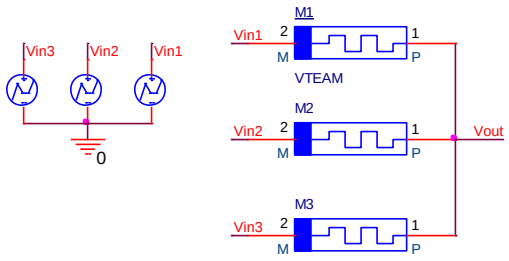


Figure 63: 3-input MRL OR gate with no load

Figure 62 above shows a 3-input MRL AND gate with no load and Figure 64 below shows its output voltage waveform for the input sequence as described in section 6.2.1. Figure 63 shows a 3-input MRL OR gate with no load and Figure 65 below shows the corresponding output voltage. The purpose of including these plots is to show that as the number of inputs increases, the duration and amplitude of some of the dynamic hazards can significantly reduce. Once a buffer is applied to the output, only 2 of the 6 hazards present in the figures will show on the output. The hazards are also generally shorter in duration.

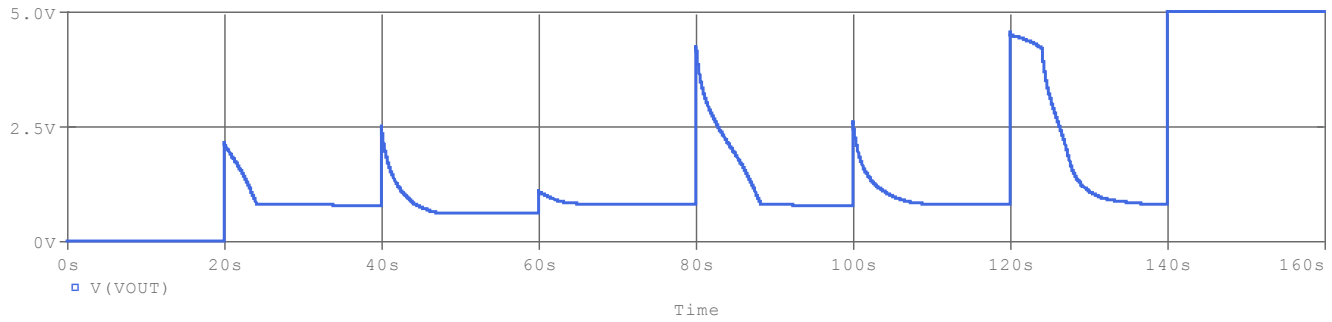


Figure 64: Output voltage of a 3-input MRL AND gate with no load and no output buffer (Figure 62). Ideally the output should be $0V$ at all times except between 140 and 160 seconds for which it should be $5V$.

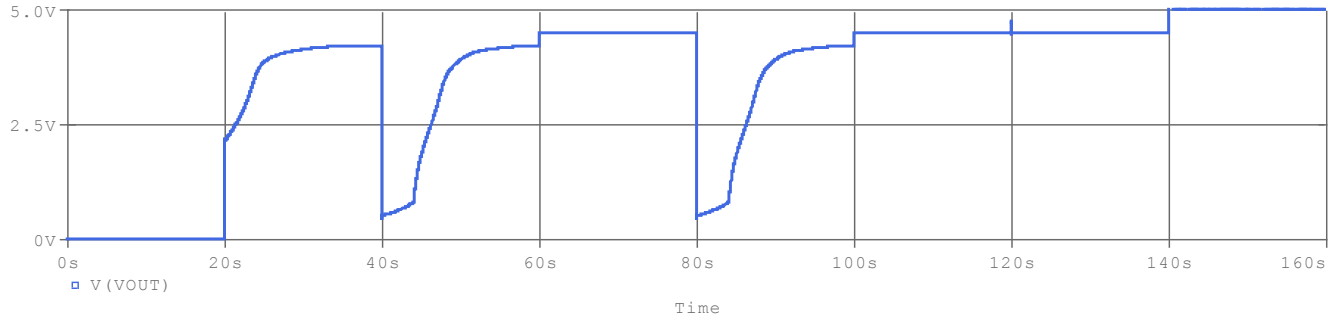


Figure 65: Output voltage of a 3-input MRL OR gate with no load and no output buffer (Figure 64). Ideally the output should be $5V$ at all times except between 0 and 20 seconds for which it should be $0V$.

Figure 67 and Figure 66 below show the effect of loading the 2-input AND and OR gates and confirm the need for buffering in-between gates. These show that drawing more current from the output results in the lower output voltage quality. This is simply because loading a potential divider interferes with its operation. Furthermore, power consumption also increases since the effective overall impedance is reduced and hence currents are greater.

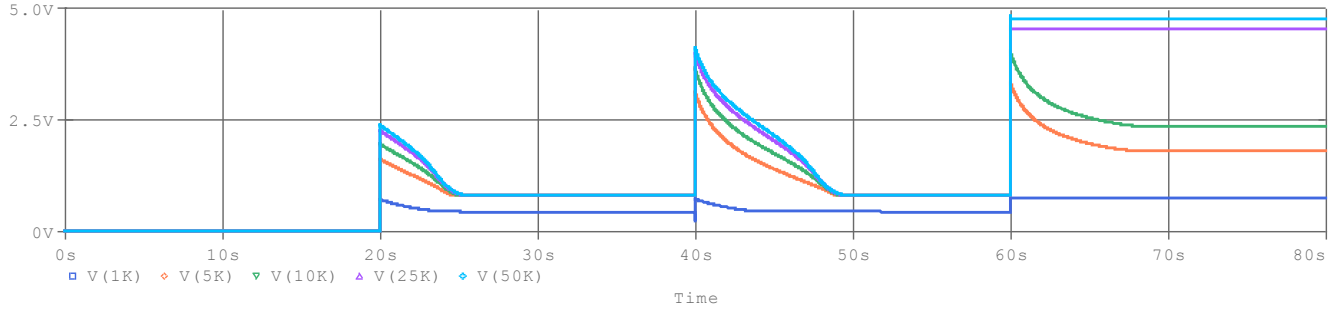


Figure 66: Showing how the output voltage of the 2-input AND gate varies when different load resistors ($1K, 5K, 10K, 25K$ and $50K$) are connected to the output. This plot is intended to show the need for output buffering with CMOS buffers or inverters.

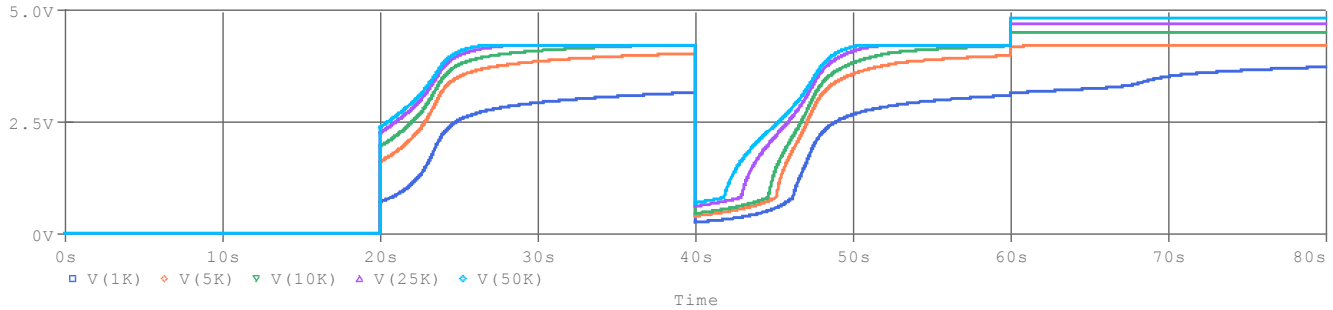


Figure 67: Showing how the output voltage of the 2-input OR gate varies when different load resistors ($1K, 5K, 10K, 25K$ and $50K$) are connected to the output.

The amount of buffering necessary depends on the memristor characteristics. For example, memristors with no threshold can more easily achieve complete switching. This coupled with a high R_{off}/R_{on} ratio means buffers or inverters can be used more sparingly.

One observation from Figure 67 and Figure 66 is that a high load current can cause switching to occur even when there is no need. Considering the input {1,1}, under no load there are no currents and hence no switching. When a sufficiently large current is drawn, the voltage of the device exceeds the threshold and it begins to switch. For the AND gate, the orientation is such that the memristance increases for all devices, whereas the OR gate is the opposite. Both of these cases are undesired. If all the memristances become low then there will be excessively high currents for some future inputs, i.e. {1,0}, or {0,1}. If both memristances become high, then the switching speed is reduced for the next set of inputs since the current would be low, (this is only applicable to a charge controlled memristor and the VTEAM model would not show this since it is voltage controlled). In the case when there is no load, one memristor will always be high and one will be low, and this is the preferred state due to the reasons just discussed.

The discussion so far has highlighted the importance of CMOS buffering due to its ability to reduce power consumption, increase fan out, maintain consistency in performance, and restore poor quality output voltage. Furthermore, CMOS inverters are necessary in order to complete the logic basis. Figure 69 shows a CMOS inverter and Figure 70 shows a non-inverting buffer. The process technology is $25\mu m$, and the devices are almost exactly matched ($K_p \approx K_n = 0.25\, mA/V^2$). This mismatch is deliberate because for some reason if the devices are exactly matched, then PSpice sometimes reports a divide by zero error. The models used are described by Figure 68.

```
.model Mbreakp PMOS VTO=-1.5 KP=0.02m W=6.3u L=0.25u
.model Mbreakn NMOS VTO=1.5  KP=0.05m W=2.5u L=0.25u
```

Figure 68: The PMOS and NMOS models used for the gates in this section.

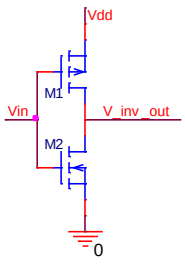


Figure 69: CMOS inverter. $Vdd = 5V$.

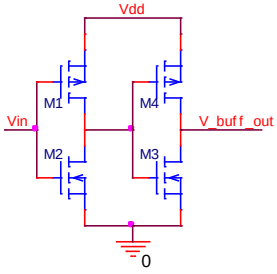


Figure 70: CMOS non-inverting buffer. $Vdd = 5V$.

Figure 71 below shows the DC transfer characteristic for the inverter and non-inverting buffer, and Figure 72 shows the steady state power consumption for the inverter as a function of input voltage. The steady state power consumption for the inverter and the non-inverting buffer is the almost exactly the same (since leakage current is negligible), and this is why only one plot is shown. From these results it is clear that the output of the un-buffered MRL gates must either be below $1.5V$, or above $3.5V$, otherwise there will be unnecessary steady state wastage in the transistors. Since the steady state output voltage error of the gates can be around $0.8V$, it is clear that each gate requires its own buffer.

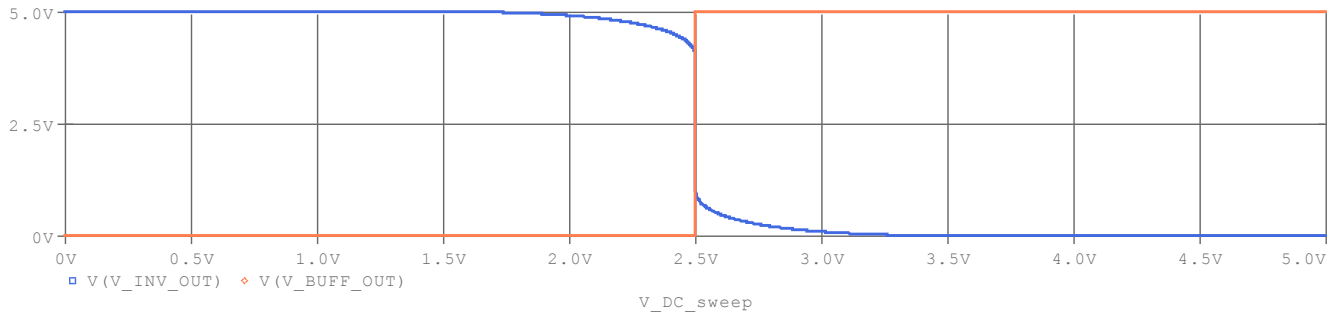


Figure 71: The blue plot shows the DC transfer characteristic for the CMOS inverter (Figure 69). The orange plot shows the DC transfer characteristic for the CMOS non-inverting buffer (Figure 70).

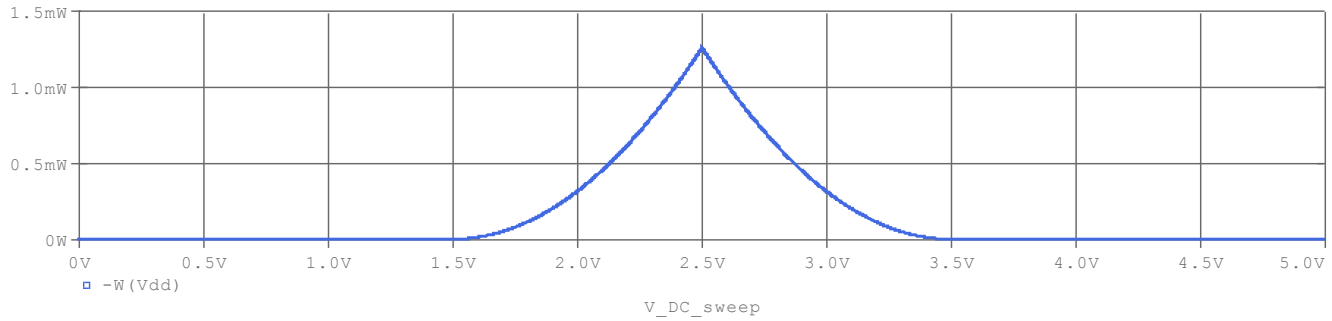


Figure 72: Steady state power consumption for the CMOS inverter as a function of input voltage. Negligible power is consumed when at least one transistor is off, and this is true when the input voltage is below $1.5V$ or above $3.5V$. Therefore the steady state power consumed by the non-inverting buffer is almost the same (since the second stage consumes negligible steady state power), which is why it is not included in this plot.

This result is disappointing because ideally the aim is to use as minimal CMOS logic as possible in order to fully take advantage of the memristors smaller sizes and higher packing densities. The interconnection between the (metal) memristive level and the (silicon) CMOS level occurs through vias, and these are space consuming and reduce the packing density. However, it is still possible to achieve a reduction in chip area even with these restrictions. [23] [24] [48] Furthermore, there exist other types of memristors which are more suitable for this particular logic family. For example some chalcogenide based memristors (which use layers of chalcogenide materials rather than oxygen vacancy migration in order to change the memristance) have smaller v_{on} thresholds in the range of $0.1v$ to $0.25v$, and much larger R_{off}/R_{on} ratios in ranges of 10^3 to 10^7. These are also compatible with the CMOS process. [83] [72]

Figure 73 below shows a 2-input OR gate with a CMOS buffer on the output. A NOR gate is the same, but with the last inverter stage removed. Figure 74 below shows the OR and NOR outputs. The addition of the buffer results in well-defined steady state logic values, but one of the dynamic hazards is still present.

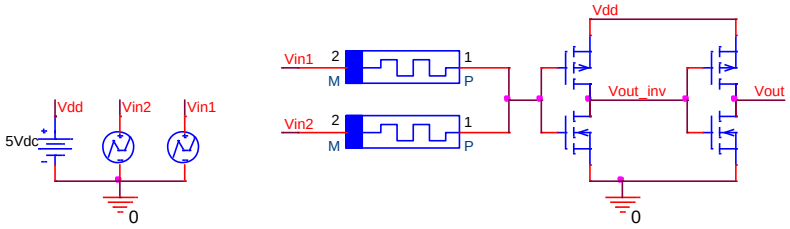


Figure 73: Hybrid 2-input OR gate with CMOS buffering. A NOR gate is the same, but with the last inverter stage removed.

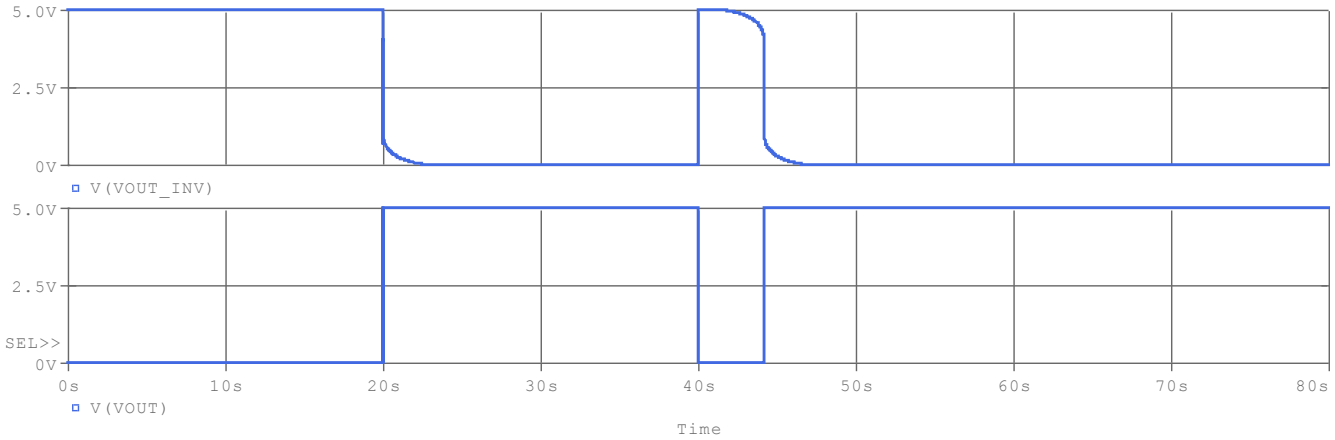


Figure 74 Output of NOR (upper plot) and OR (lower plot) gate with CMOS buffer. The steady state logic levels are now well defined, but there is still a glitch at the 40 second interval for duration of around 7.5 seconds.

Figure 75 below shows a 2-input AND gate with a CMOS buffer on the output. A $NAND$ gate is the same, but with the last inverter stage removed. Figure 76 below shows the OR and NOR outputs, and once again, the addition of the buffer results in well-defined steady state logic values, but one of the dynamic hazards (at the $40s$ interval) still exists.

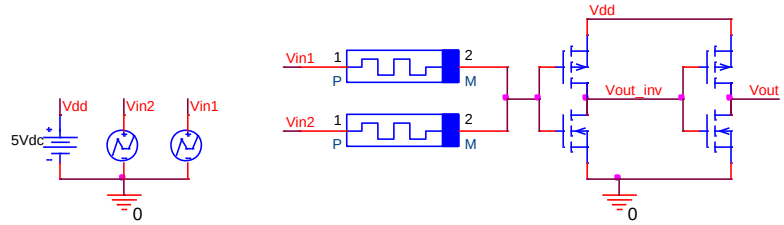


Figure 75: Hybrid 2-input AND gate with CMOS buffering. A $NAND$ gate is the same, but with the last inverter stage removed

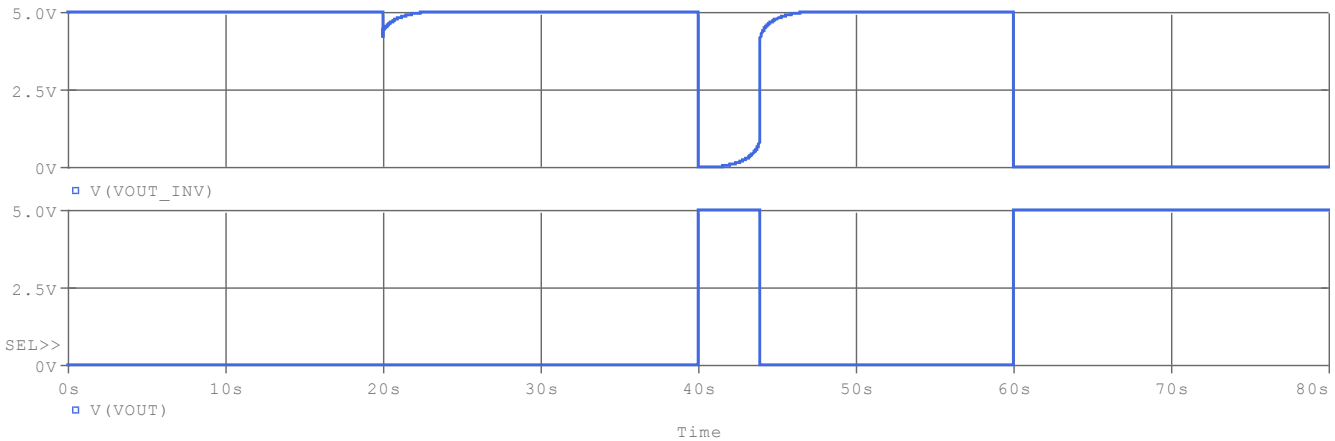


Figure 76 Output of $NAND$ (upper plot) and AND (lower plot) gate with CMOS buffer. The Steady state logic levels are now well defined, but there is still a glitch at the 40 second interval for duration of around 7.5 seconds.

Table 16 below compares the device count for N-input gates based on CMOS and hybrid MRL technology. This actually shows a reduced device count for the hybrid gates for the case when the number of inputs exceeds 2. Two input gates use the same number of devices, but area can still be less as previously mentioned. It is worth mentioning that although buffering increases the fan out, it is not as high as CMOS fan out (which is typically 100) because memristor based gates can draw more current from inputs compare to transistor gates. They even draw steady state currents.

$N\text{-}input\ Gate$:	$Memristor\ Count$:	$Transistor\ count$:	$Total\ device\ count$:
$CMOS\ AND$	0	$2N+2$	$2N+2$
$CMOS\ OR$	0	$2N+2$	$2N+2$
$CMOS\ NAND$	0	$2N$	$2N$
$CMOS\ NOR$	0	$2N$	$2N$
$MRL\ (Hybrid)\ AND$	N	4	$N+4$
$MRL\ (Hybrid)\ OR$	N	4	$N+4$
$MRL\ (Hybrid)\ NAND$	N	2	$N+2$
$MRL\ (Hybrid)\ NOR$	N	2	$N+2$

Table 16: Comparing N-input CMOS gates with hybrid CMOS gates in terms of device count. This table assumes each MRL gate is buffered at the output by CMOS buffers or inverters.

6.2.3 Full Adder

In this sub-section, a full adder circuit is simulated. The process and results of this will help allow deeper understanding and help facilitate a detailed comparison between MRL and the other logic families. The first task is to construct an XOR gate which is fundamental for the half adder. To this end, various designs are simulated and compared for in terms of the following metrics:

1. Speed: The propagation delay is defined here as the time taken for the output to settle within $0.05V$ of the correct value, i.e. the CMOS acceptable value. This is determined by simulating all possible input transitions. There are $2 \times {}^4C_2 = 12$ of these.
2. Power: The total energy consumed when the inputs run through all possible input transitions is computed.
3. Area: The individual device count along with the number of CMOS to memristor level transitions is counted for each design. This is important since it determines the number of area consuming vias necessary.

The results for each XOR design are summarised by Table 17 towards the end of this subsection. A novel hybrid XOR design is also proposed and compared with existing implementations. This section then ends with the simulation and analysis of a full adder.

6.2.3.1 XOR Gate – No Buffering:

Figure 77 shows one of the most simple and intuitive XOR designs which makes use of three mixed gates: AND, OR and $NAND$. The Boolean expression directly implemented is $V_{out} = \left(\overline{V_{in1} \wedge V_{in2}}\right) \wedge (V_{in1} \vee V_{in2})$. In this design, there is no intermediate buffering in-between subsequent gates; the only intermediate CMOS logic present is necessary for the $NAND$ gate.

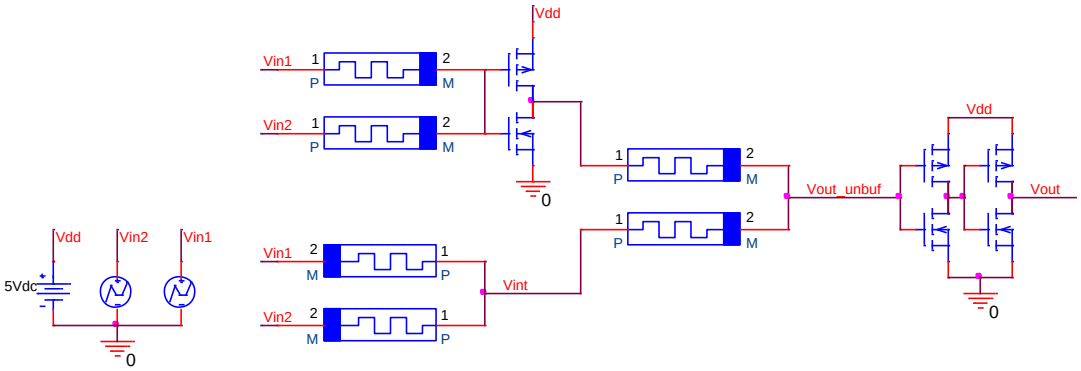


Figure 77: XOR gate with no intermediate buffering in-between gates. The only CMOS logic used is the inverter necessary to implement the $NAND$, and the output buffer.

The no-load output voltage of this XOR implementation is shown by Figure 78 below. This plot also includes the intermediate signal (v_{int}) in order to demonstrate how loading a gate with another gate affects its output. Also the signal prior to the output buffer (v_{out_unbuf}) is shown in order to demonstrate how quickly the output can deteriorate without buffering for the case of devices with relatively high switching thresholds.

Figure 78 shows the output to be incorrect for the input combination $\{0,0\}$. This is due to inadequate quality of the signal v_{int}. Loading this OR gate with the AND gate has deteriorated the steady state output values, especially for input case $\{0,0\}$ where the output is now $1.71V$ as opposed to $0.8V$ for the unloaded case. The next design attempts to remedy this without the use of additional CMOS buffers.

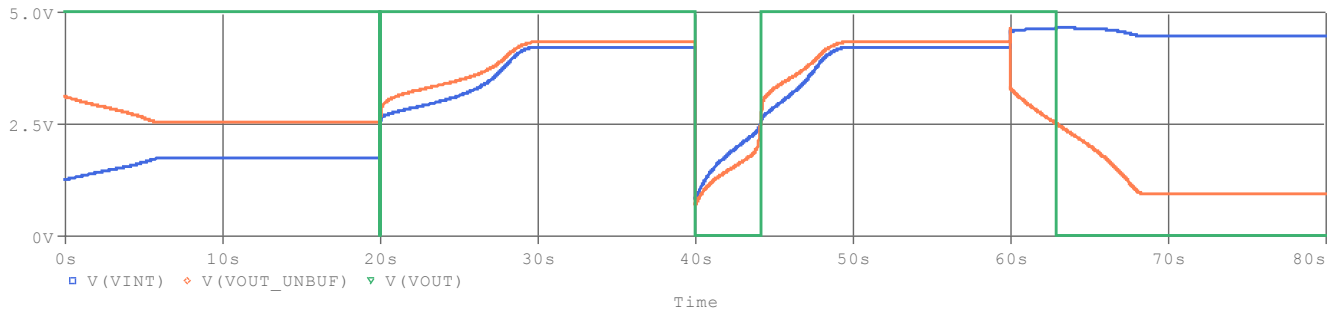


Figure 78: The buffered and un-buffered output voltage when the input sequence is as described in section 6.2.1. The intermediate signal (blue) is also included to show how loading a gate with another actual gate deteriorates the steady state output.

Figure 79 below shows non-zero steady state power consumption. For the case of unloaded single gates, there was zero consumption when all the input were the same. This is no longer the case because even if the input signals are the same, the intermediate signals may not be.

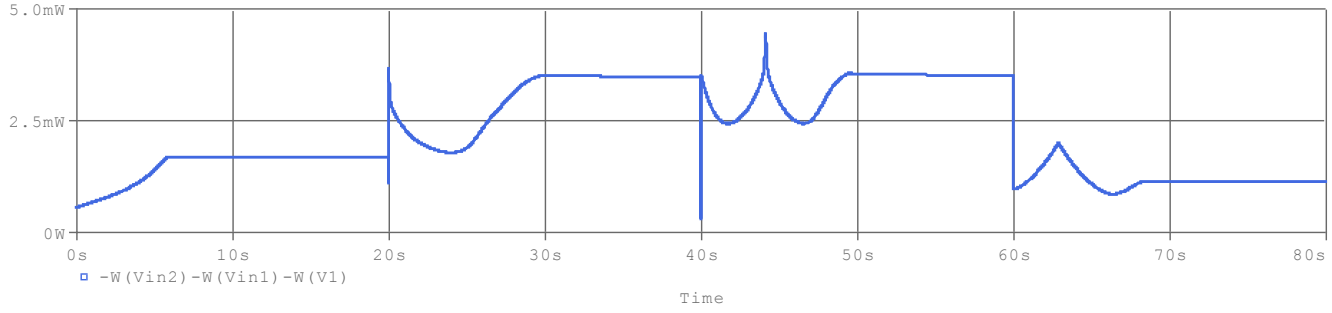


Figure 79: Power consumption. Total energy consumed over this period is $177mJ$.

6.2.3.2 *XOR Gate – Parallel Memristors*

The following circuit attempts to remedy the issue with the previous implementation without introducing additional CMOS logic. The basic principle is to increase the size of memristor gates which are driving other gates. A larger memristor can be achieved by a parallel combination as shown by Figure 80 below.

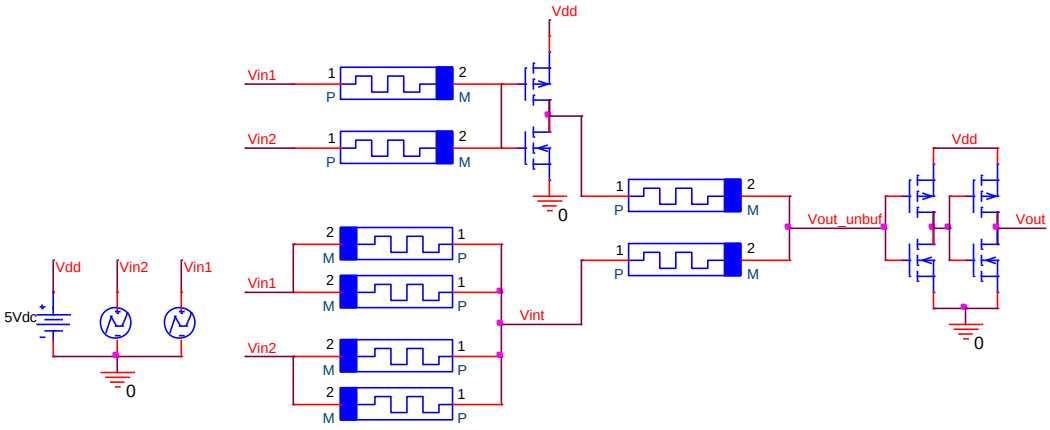


Figure 80: XOR design with parallel memristor gates driving subsequent gates.

Figure 81 shows a slight improvement in the intermediate voltage signals and the output of the gate is now logically correct. However, this improvement is limited and there are still some issues; the steady state error of the v_{out_unbuf} and v_{int} signals during the period 0 to 20s are still poor. They should both be much closer to $0V$, but instead they are close to the midpoint, meaning that the design is not robust. This agrees with the conclusions made in 6.2.2 which stated that each gate needs to have its own buffer in order to counteract the poor steady state error due to the voltage threshold and/or inadequate R_{off}/R_{on} ratios. This lack of robustness increases the chance of error due to natural variation of the memristors. One issue with these metal oxide based memristors is that they are difficult to produce consistently, and this is because fabrication techniques are complex and it is difficult to control the oxygen concentration within a film. Also, it can be tricky to keep oxygen out of the device after fabrication. [72] Another issue is that the steady state v_{out_unbuf} signal is that it is such that both p-channel and n-channel MOSFETs in the output buffer are both conducting and thus wasting power.

Furthermore, this design requires more chip area, and intuitively, the memristors are also expected to consume more power due to lower input resistance meaning greater currents. Total energy consumed over this 80 second period is $231mJ$ which is 30% more than the previous design.

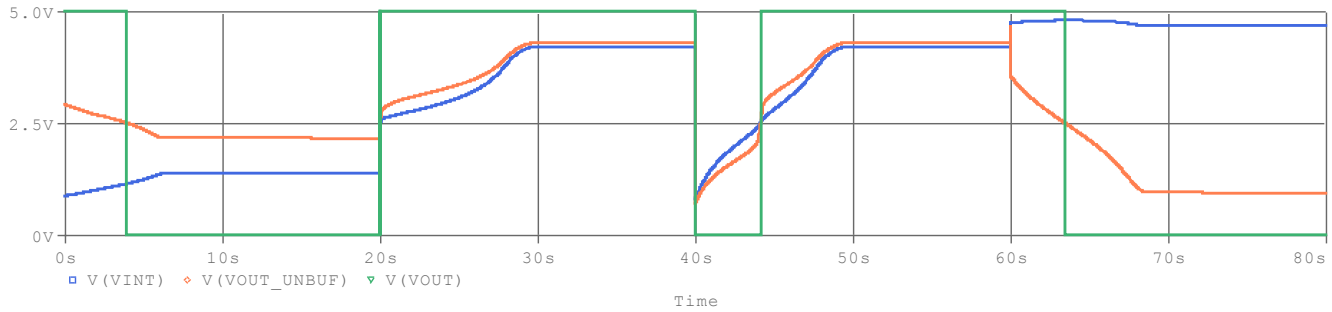


Figure 81: The buffered and un-buffered output voltage when the input sequence is as described in section 6.2.1. The intermediate signal (blue) is also included to show how loading a gate with another actual gate deteriorates the steady state output. The worst case propagation delay is $4.7s$.

6.2.3.3 *XOR gate – Full Buffering:*

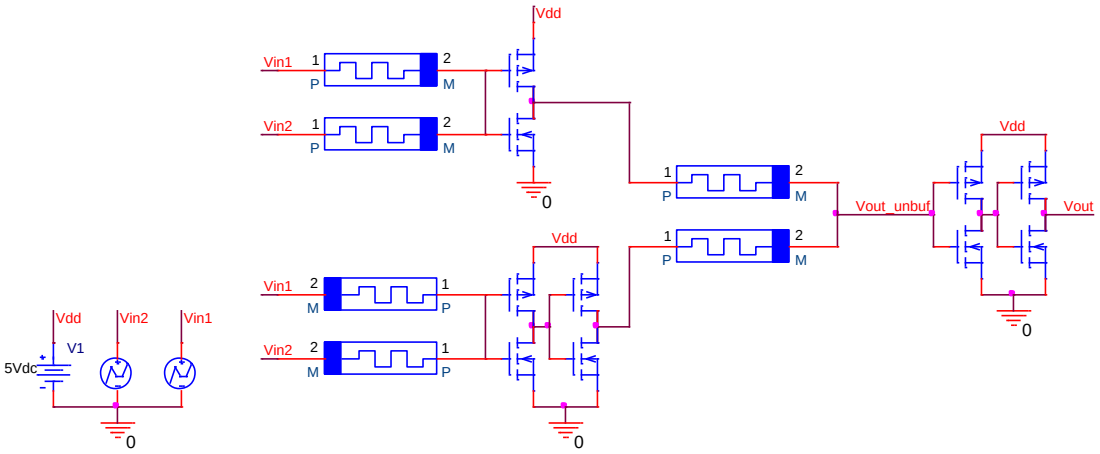


Figure 82: XOR design with a buffer for each internal gate.

Figure 82 above shows the next design which involves a buffering all intermediate gates. This uses more transistors compared to the previous XOR gate, but the operation is more robust illustrated by Figure 83. From this it can be seen that there is no glitch at the 0 second mark unlike the previous design. Furthermore, the total energy consumed over this period is $185mJ$ which is lower than the previous design and this is because the CMOS transistors in this design are operated in the correct complimentary manor, i.e. only one is conducting in the steady state. It is clear that chip area can be traded off with robustness of operation.

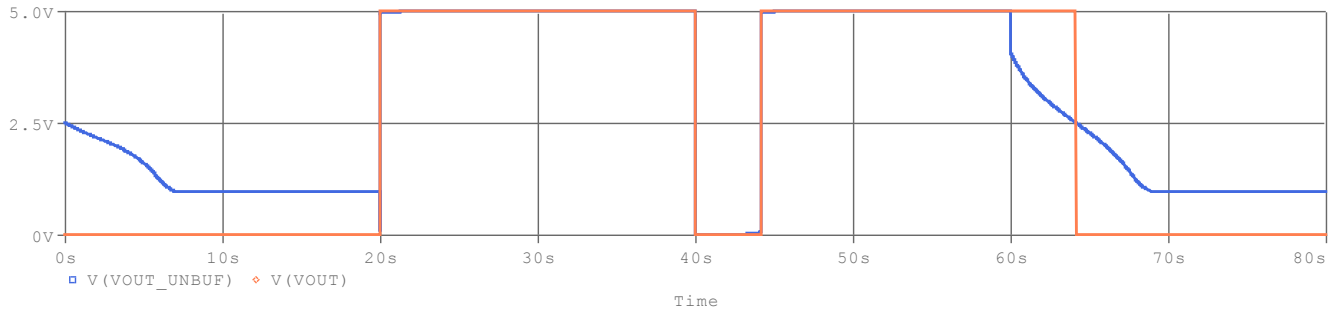


Figure 83: Output voltage before and after output buffer. The un-buffered output signal is significantly improved. The worst case propagation delay/dynamic hazard is slightly reduced to $4.3s$.

6.2.3.4 *XOR Gate – NAND Implementation:*

Figure 84 below shows a XOR gate constructed from hybrid $NAND$ gates. This design uses two more memristors and has more connections between the CMOS and memristive layers compared to the previous one meaning it required more chip area. Since each memristor gate is followed by a CMOS inverter, the intermediate logic voltage levels are well defined and operation is robust. This is demonstrated by Figure 85 below which shows the output and internal signals of interest. However, the greater number of gates results more power consumption: total energy consumed over this 80 second sequence is $325mJ$ which is 75% more than the previous one.

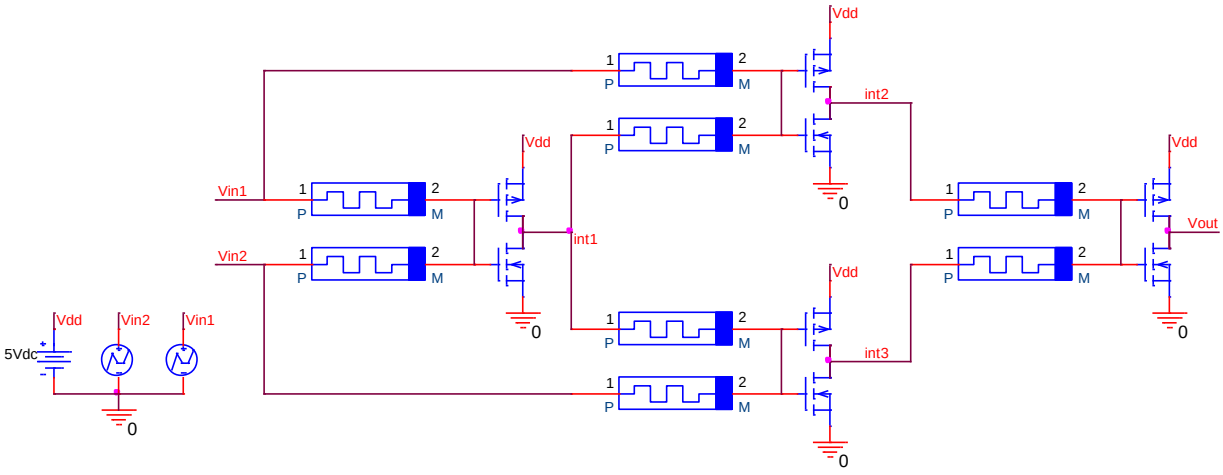


Figure 84: XOR gate built from $NAND$ gates.

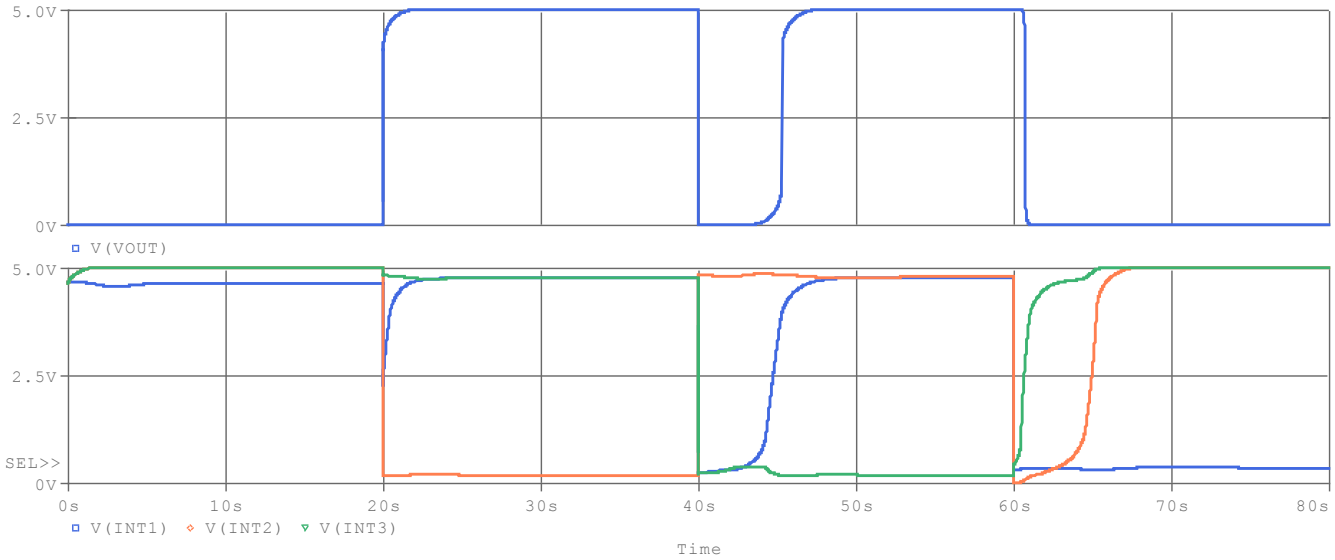


Figure 85: Output and internal voltages for the $NAND$ based XOR gate.

Another disadvantage of this gate is the large propagation delay, and this occurs because there can be 3 hybrid gates in the signal path compared to 2 for the previous designs. The worst case propagation delay is $15.6s$ which is around 3 times as long as previous designs. Figure 86 below shows an example of one of these delays (along with the glitch) when the input changes form logic {1,1} to {1,0} at the 160 second mark.

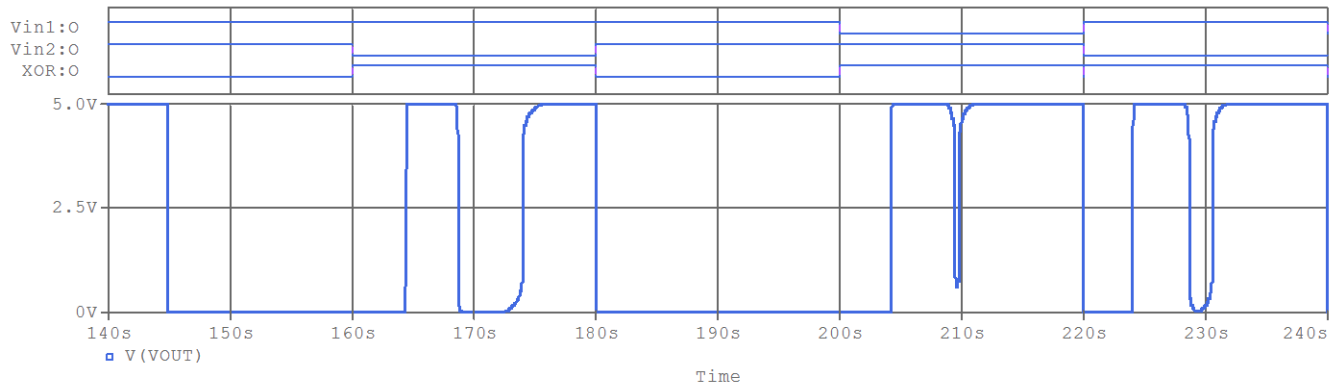


Figure 86: Demonstrating the worst case propagation delay. The top plot shows the input logical sequence and the ideal XOR output. The bottom plot shows the voltage output of the $NAND$ based XOR gate. A $15.6s$ delay and glitch occurs at $160s$.

6.2.3.5 XOR Gate – NOR Implementation:

The final common variation of the XOR gate is shown by Figure 87 below, and the signals of interest are shown by Figure 88. Only hybrid NOR gates and CMOS inverters are used in this implementation. This design is worse in terms of energy $(200mJ)$, propagation delay $(7.2s)$, and area and compared to the fully buffered design in section 6.2.3.3 which is optimal so far.

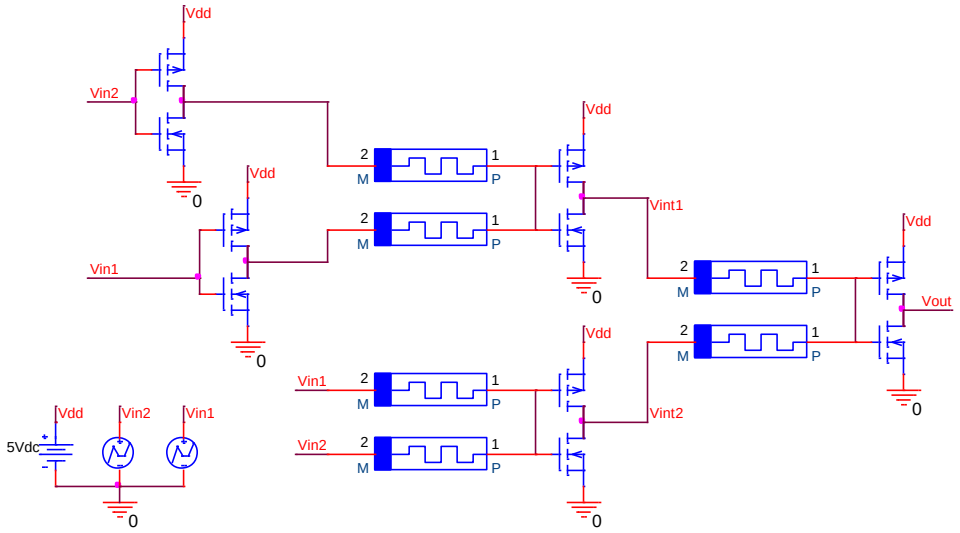


Figure 87: XOR gate built from hybrid NOR gates and CMOS inverters.

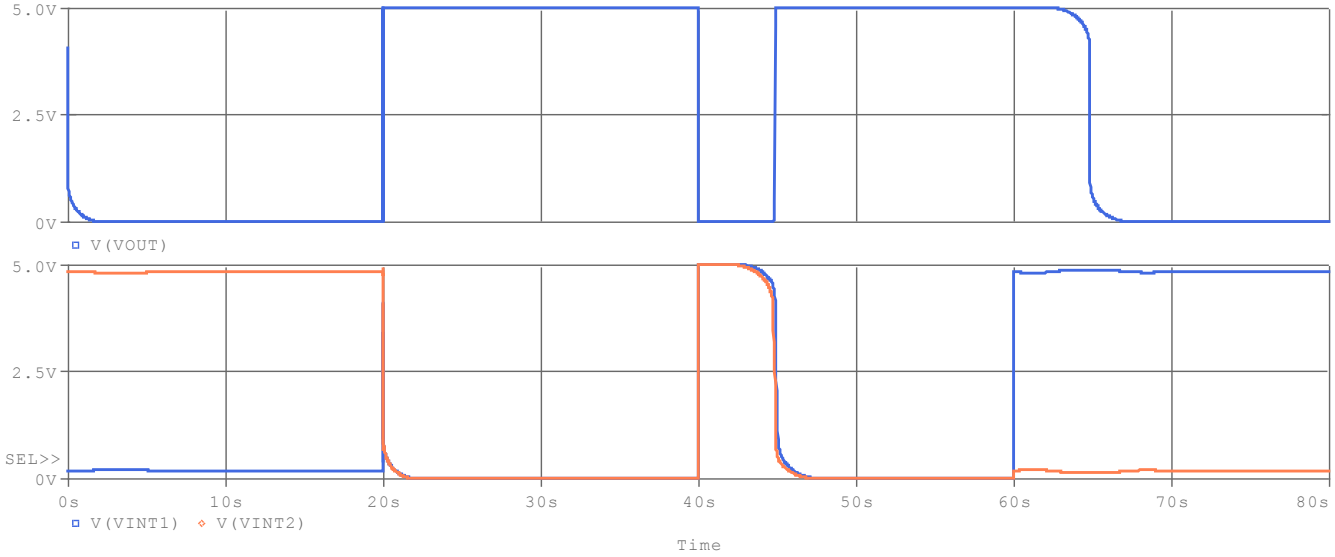


Figure 88: Output voltage and intermediate signals.

6.2.3.6 Proposed Hybrid XOR Design:

Figure 89 below shows a novel hybrid XOR gate which uses only 2 memristors and 4 transistors, and Figure 90 shows the output and internal voltages. The operation can be explained by considering the different stages of the circuit separately. The first part is the MRL AND gate and the output of this is connected to the NMOS source potential for the following CMOS NOR gate. These two stages combined perform $XNOR$ because when both the inputs are high, then the NMOS source potential is high so the output is also high. The final stage is a standard inverter which also performs the task of restoring the logic level.

This design requires considerably less space compared to the others since fewer devices are used and only 2 connections between the memristor and CMOS layers are required. Another advantage is reduced power consumption as shown by Figure 91 below. Total energy consumed over the full cycle is $73mJ$ which is a reduction of 60% compared to the next best alternative. Note that the steady state power consumption is negligible for the case when the inputs are logic $\{0,0\}$ or $\{1,1\}$. This is because the CMOS circuit does not actually load the MRL AND gate in steady state condition because of its complementary operation. One disadvantage is that the propagation delay is $6.5s$ which is 50% greater than the $4.3s$ delay for the design in section 6.2.3.3.

When the inputs are logic $\{1,1\}$ then v_{int2} is approximately equal to $5V$ minus the NMOS threshold voltage. This means that the operation of the subsequent inverter is barely complementary and not really robust. This can be remedied if the NMOS threshold is designed to be less than the PMOS one, and this can be achieved during the fabrication process since the properties can be changed by doping.

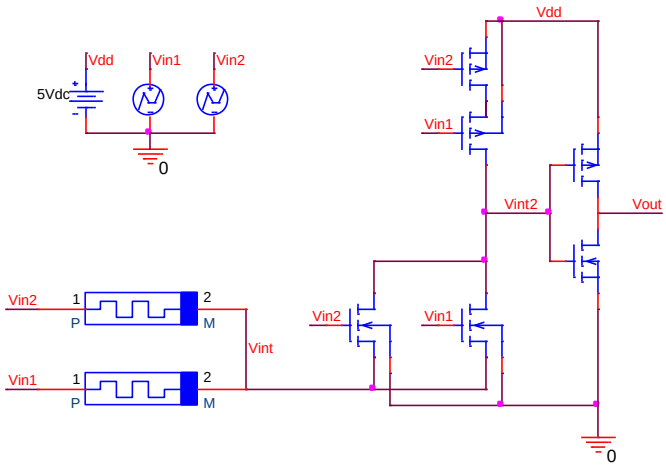


Figure 89: Proposed hybrid XOR gate. The two memristors on the left perform AND operation, and the output of this is used as the NMOS source potential for the following NOR gate. The final stage is a standard inverter.

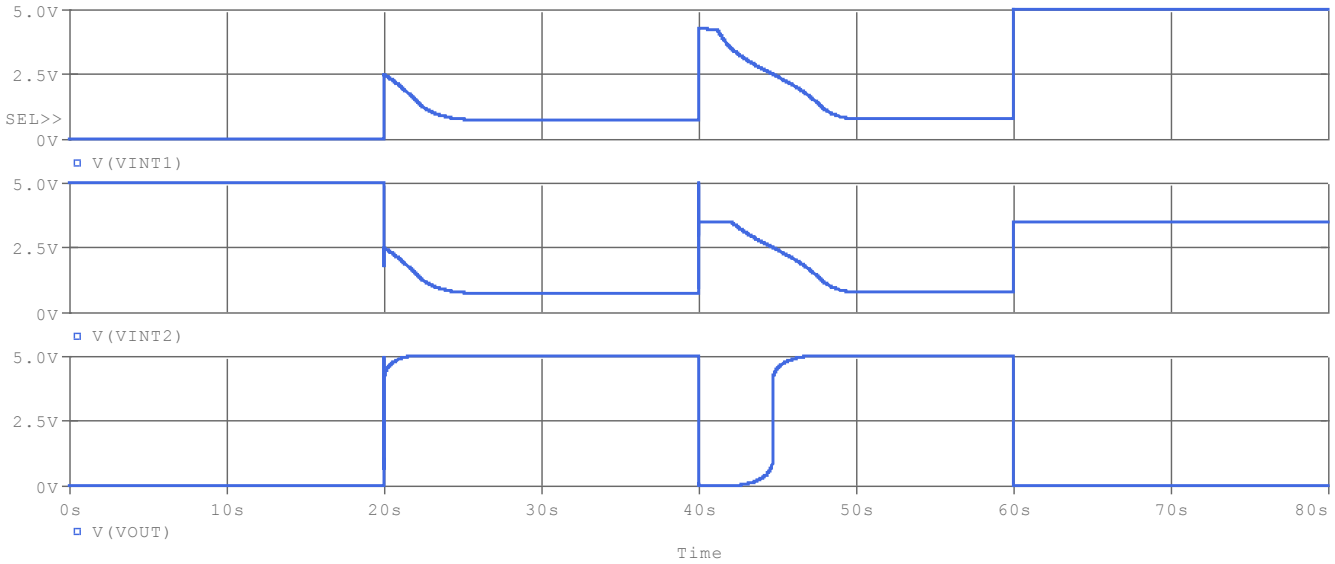


Figure 90: The output and internal voltages when the input is as described in section 6.2.1.

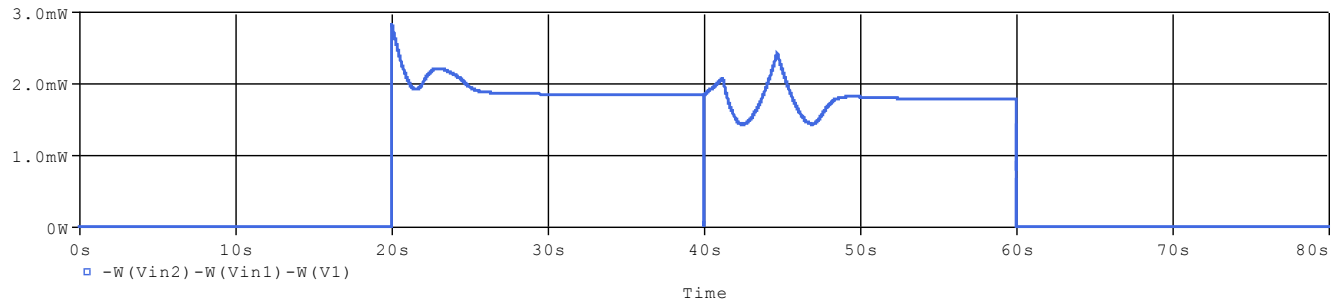


Figure 91: Power consumption. Total energy consumed over this period is $73mJ$.

Figure 92 below shows the initial idea which led to the final design. This operates on a similar principle, except here the output of a MRL OR gate is used as the drain potential for the CMOS $NAND$ gate. This uses two less transistors, but the disadvantage is poor quality output since there is no final CMOS stage with its drain and source connected to V_{dd} and V_{ss} respectively. The output internal voltages are shown by Figure 93 below. This design may be preferred if the memristors had large ratio and no thresholds.

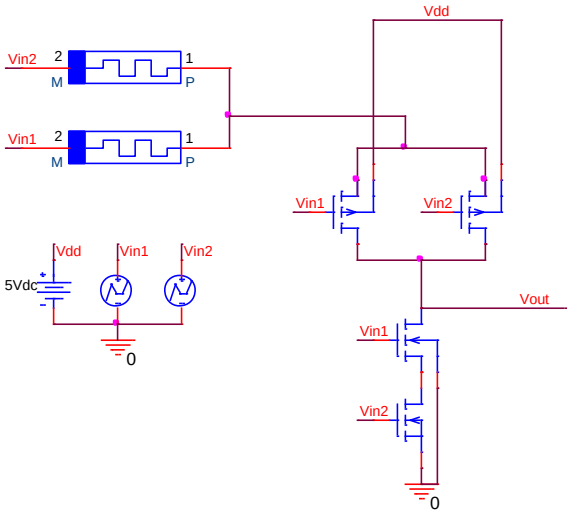


Figure 92: The initial XOR design which was then modified to form the one in Figure 89.

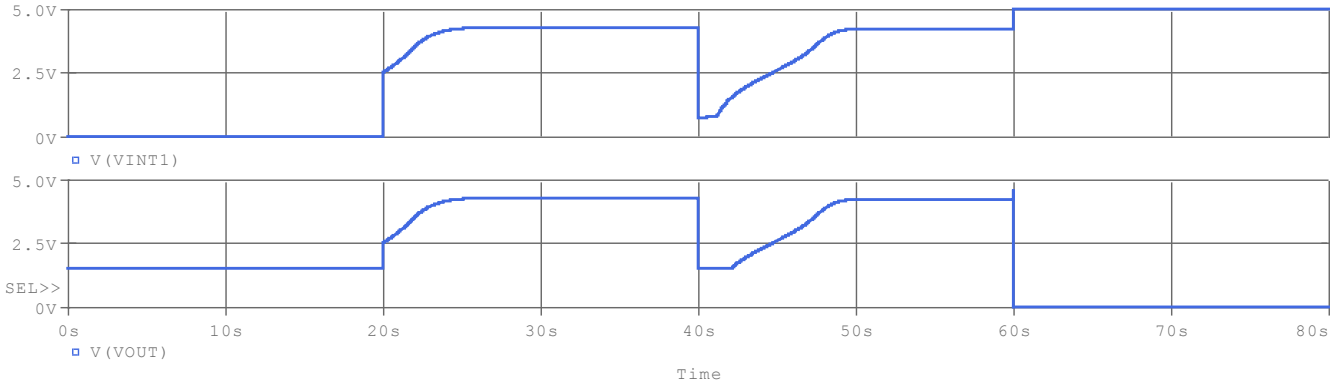


Figure 93: The output and internal voltages. The output is unsatisfactory for the logic input cases {00}, {01} and {1,0}. Total power consumed over the interval is $70.6mJ$

6.2.3.7 Summary of XOR Performance:

The table below summarises the performance of the previously discussed XOR gates. The energy consumption metric is determined by integrating the total power over a switching sequence which runs through all possible input transitions. This sequence is $240s$ long and covers all 12 transitions.

Design:	*Transistor Count*:	*Memristor Count*:	*Memristor to CMOS transitions*:	*Energy Consumed/mJ*:	*Worst Propagation case*:	*Worst propagation time/s*:
Un-buffered (6.2.3.1)	6	6	4	(*Incorrect operation*)		
Parallel (6.2.3.2)	6	8	4	760	$\{1,1\} \rightarrow \{0,0\}$ $\{0,1\} \rightarrow \{1,0\}$ $\{1,1\} \rightarrow \{0,1\}$	4.7
Fully buffered (6.2.3.3)	10	6	6	601	$\{1,1\} \rightarrow \{0,0\}$	4.3
NAND (6.2.3.4)	8	8	8	993	$\{1,0\} \rightarrow \{1,1\}$ $\{1,1\} \rightarrow \{0,0\}$ $\{0,0\} \rightarrow \{1,1\}$ $\{1,1\} \rightarrow \{0,1\}$	15.6
NOR (6.2.3.5)	10	6	10	599	$\{1,0\} \rightarrow \{1,1\}$ $\{1,1\} \rightarrow \{0,0\}$ $\{0,0\} \rightarrow \{1,1\}$ $\{0,1\} \rightarrow \{0,0\}$	7.2
Proposed (6.2.3.6)	6	2	2	210	$\{0,1\} \rightarrow \{1,0\}$ $\{1,1\} \rightarrow \{0,1\}$ $\{1,1\} \rightarrow \{1,0\}$ $\{1,0\} \rightarrow \{0,1\}$	6.5

Table 17: Summary of performance for various XOR designs.

6.2.3.8 Full Adder

The full adder is constructed based on the proposed XOR design from section 6.2.3.6 and is shown by Figure 94 below. This XOR design is selected due to its superior performance in terms of chip area and power. Another advantage is that the AND operation of the inputs is already performed by the memristor part of XOR gates and so there is opportunity to optimise the adder. This allows a reduction of 4 memristors, along with the associated CMOS buffers. This design uses 18 MOSFETs and 4 memristors.

In comparison, a CMOS only full adder requires 28 MOSFETs, [84] and a pass transistor logic full adder requires 23 MOSFETs. [85] It is actually possible to achieve a higher memristor to transistor ratio when devices with a higher R_{off}/R_{on} ratio and small thresholds are considered. This is because the less restoring of the signal is necessary. For example, the design proposed by Tejinder Singh requires 18 memristors and 8 transistors (or 16 transistors if outputs are buffered). [48]

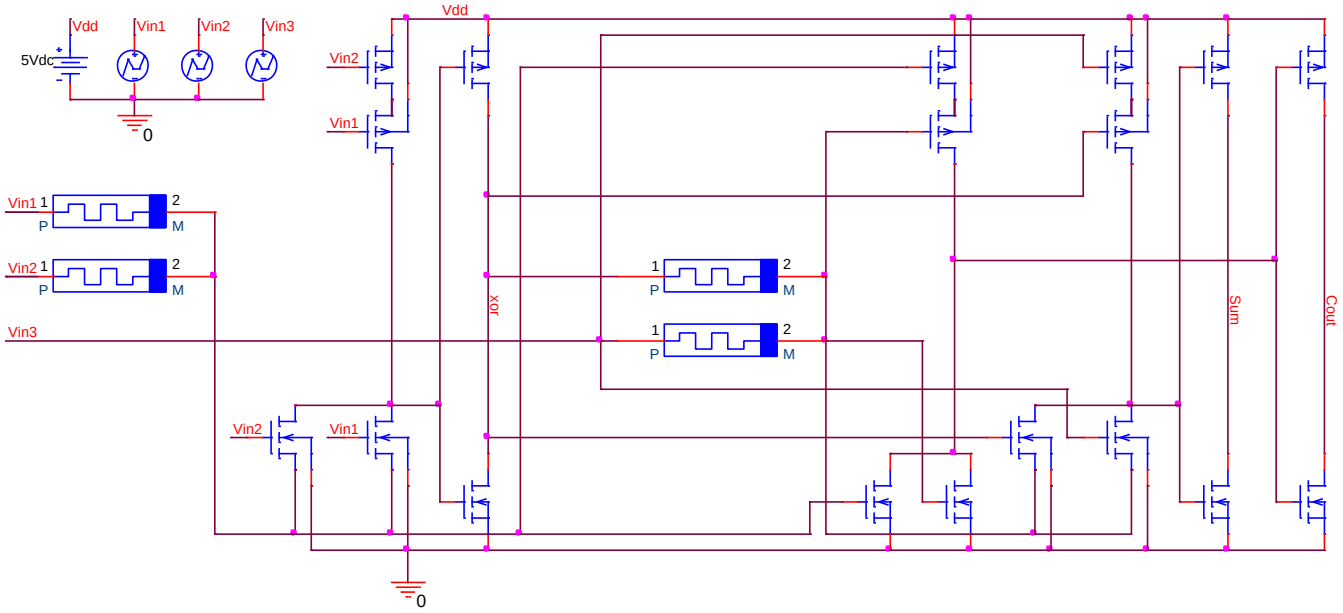


Figure 94: Hybrid Full Adder consisting of 18 transistors and 4 memristors. The NMOS voltage threshold has been modified from $1.5V$ to $1.25V$, and this is to increase robustness of the XOR gates as explained in section 6.2.3.6.

Figure 95 below shows sum and carry outputs when the input sequence is as described in section 6.2.1. This confirms correct operation, and also shows glitches on the outputs at 20, 40 and 120 seconds. The worst case propagation delay is $6.5s$ for the sum output when the input changes from logic {0,1,1} to {1,0,0}. Figure 96 below shows total power consumption and from this it is clear that there is non-zero static consumption. However, this could be solved by some input pulsing technique. The total energy consumed over this interval is $288mJ$.

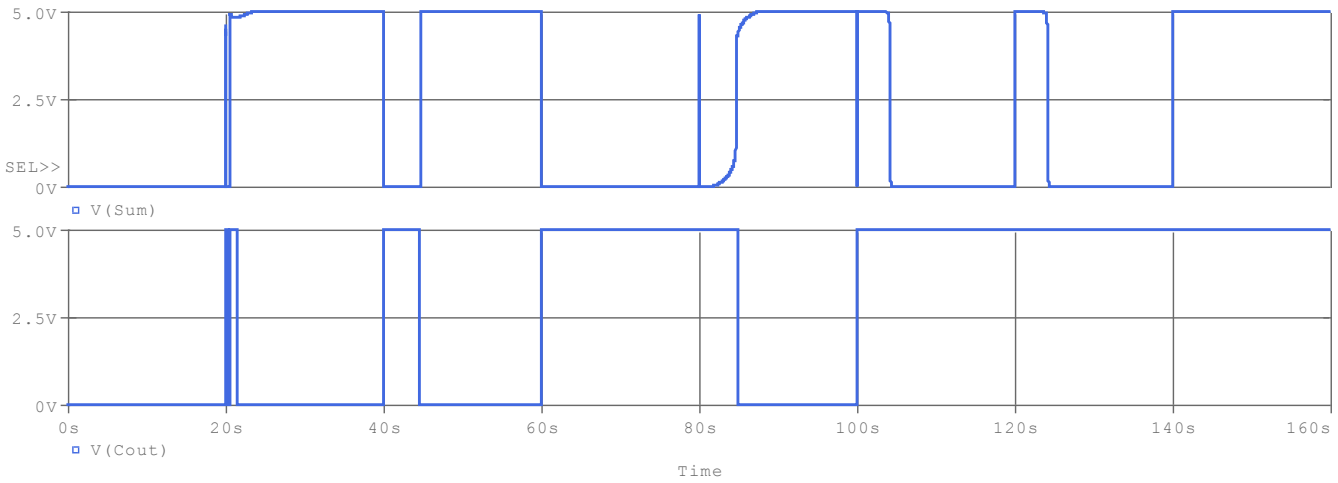


Figure 95: The sum (top plot) and carry (bottom plot) outputs when the input sequence is as described in section 6.2.1.

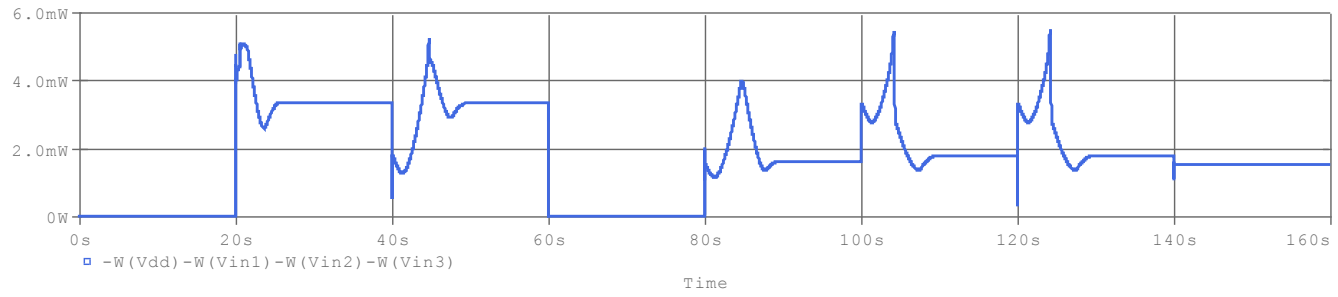


Figure 96: Power consumption. The total energy consumed over this interval is $288mJ$.

6.3 IMPLY

This section further studies the Implication Logic family as introduced in section 3.3. As mentioned earlier, this logic family requires a complex driver capable of pulsing various voltage levels as well having a as a high impedance mode. Such a driver is implemented in PSpice using the SBreak component and is shown by Figure 97 below. As before, the '*V_PWL_FILE*' voltage source is controlled by a text file generated by a MATLAB script (Appendix B), and this easily allows the user to generate the desired sequence of voltage pulses. The additional feature now is that when the voltage specified exceeds a certain value ($5V$ in this case), the switch turns off and high impedance mode is activated.

Figure 99 describes this SBreak component, and Figure 98 verifies overall operation of the driver. The top plot shows the voltage sequence as specified in the text file (v_{sin1}), as well as the output voltage of the driver (v_{in1}). These two are the same except between 3 to 4 seconds, which is when high impedance mode is entered since v_{sin1} exceeds the $5V$ threshold. The impedance of the driver is indicated by the lower plot of Figure 98. The reason for using voltages greater than $5V$ to trigger the high impedance mode is convenient since it allows only one voltage source and file to control the whole driver. If for some reason a higher voltage needs to be outputted, then the threshold can simply be increased by modification of the SBreak component.

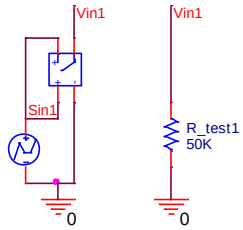


Figure 97: Driver for *IMPLY* and *MAGIC* logic.

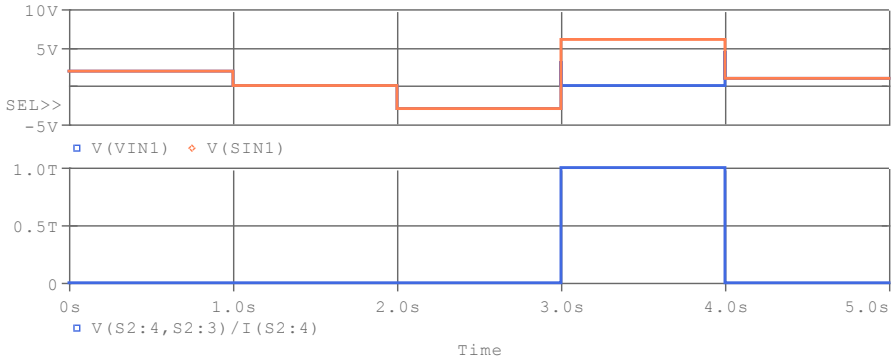


Figure 98: Verifies correct operation of the driver circuit. The top plot shows the driver output (V_{in1}) and source (S_{in1}) voltage, and the lower plot shows the drivers impedance.

```
.model Sbreak VSWITCH Roff=1e12 Ron=1e-3 Voff=5 Von=4.9
```

Figure 99: The parameters for the SBreak component. R_{off} is set to $1/GMIN = 1.0\,T\Omega$.

6.3.1 Gates – *IMPLY*

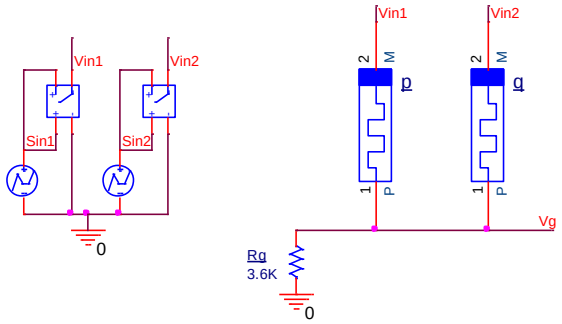


Figure 100: *IMPLY* gate in PSpice. The left memristor is denoted by p, and the right by q.

Before attempting compound/sequential logic, the fundamental *IMPLY* operation must first be verified. Figure 100 above shows the *IMPLY* gate in question, and Table 18 below shows the test sequence applied, with Table 19 below listing and justifying the parameters used. To summarise, the inputs are first set up and then the *IMPLY* operation is performed. This process is repeated for all possible binary input combinations in ascending order. The advantage of

this test method is that each case can be independently investigated without the output of the previous result affecting the results of the next case; thus it allows for easy identification of problem cases. Table 18 also summarises the steady state values of the simulation results which are shown by Figure 101 below.

Time (seconds):	$0-20$	$20-40$	$40-80$	$80-100$	$100-120$	$120-160$
Action:	*Clear P*	*Clear Q*	$Q = P \rightarrow Q$	*Clear P*	*Set Q*	$Q = P \rightarrow Q$
V_{in1}:	$-v_{clear}$	v_z	$\lvert v_{cond}\rvert$	$-v_{clear}$	v_z	$\lvert v_{cond}\rvert$
V_{in2}:	v_z	$-v_{clear}$	$\lvert v_{set}\rvert$	v_z	$\lvert v_{set\_h}\rvert$	$\lvert v_{set}\rvert$
Desired result:	$P = 0$	$Q = 0$	$P = 0$ $Q = 1$	$P = 0$	$Q = 1$	$P = 0$ $Q = 1$
Actual result:	$P = 0.006$	$Q = 0.007$	$P = 0.006$ $Q = 0.905$	$P = 0.006$	$Q = 0.990$	$P = 0.002$ $Q = 0.990$

Time (seconds):	$160-180$	$160-200$	$200-240$	$240-260$	$260-280$	$280-320$
Action:	*Set P*	*Clear Q*	$Q = P \rightarrow Q$	*Set P*	*Set Q*	$Q = P \rightarrow Q$
V_{in1}:	$\lvert v_{set\_h}\rvert$	v_z	$\lvert v_{cond}\rvert$	$\lvert v_{set\_h}\rvert$	v_z	$\lvert v_{cond}\rvert$
V_{in2}:	v_z	$-v_{clear}$	$\lvert v_{set}\rvert$	v_z	$\lvert v_{set\_h}\rvert$	$\lvert v_{set}\rvert$
Desired result:	$P = 1$	$Q = 0$	$P = 1$ $Q = 0$	$P = 1$	$Q = 1$	$P = 1$ $Q = 1$
Actual result:	$P = 0.990$	$Q = 0.007$	$P = 0.990$ $Q = 0.007$	$P = 0.990$	$Q = 0.990$	$P = 0.990$ $Q = 0.990$

Table 18: Sequence of voltages applied in order to test the *IMPLY* gate of Figure 100. The parameters used are explained in Table 19 below. The most interesting result is highlighted in red.

Parameter:	*Value*:	*Justification*:
R_g	$3.6K$	Assuming that full switching takes place, R_g is desired such that $R_{on} \ll R_g \ll \frac{R_{off}}{2}$. Hence the geometric mean of R_{on} and $\frac{R_{off}}{2}$ is suitable.
v_{set}	$-1.3V$	Both of these are used during the *IMPLY* operation. A large value for v_{set} is desired in order to aid maximum switching, but the guidelines given by 3.1 and 3.2 must also be satisfied. These particular values were chosen by starting with the guidelines, and then using iterative trial and error in order to achieve the best possible result.
v_{cond}	$-0.7V$	
v_{clear}	$3V$	This is used for the *FALSE* operation, i.e. setting the output to logic zero. A large value (within reason) is desired for fast and full clearing.
v_{set_h}	$-2.9V$	This value is only used for testing purposes in the input setup stage before each *IMPLY* operation takes place. A high value is chosen so the setup stage is fast and (almost) complete switching is achieved. Complete switching is not possible since the memristance can only decrease until the potential divider formed between the memristor and R_g is such that the voltage falls below the threshold. One consequence of this is that v_{clear} must be greater than $\lvert v_{set\_h}\rvert$, otherwise a clear operation will fail after a set operation because the threshold is not exceeded.
v_z	$6V$	Denotes high impedance mode of the driver. $6V$ is sufficient to trigger this since the 'SBREAK' switch is configured to turn off at $5V$.

Table 19: Parameters selected by considering the guidelines given by 3.1 and 3.2, and by iterative trial and error.

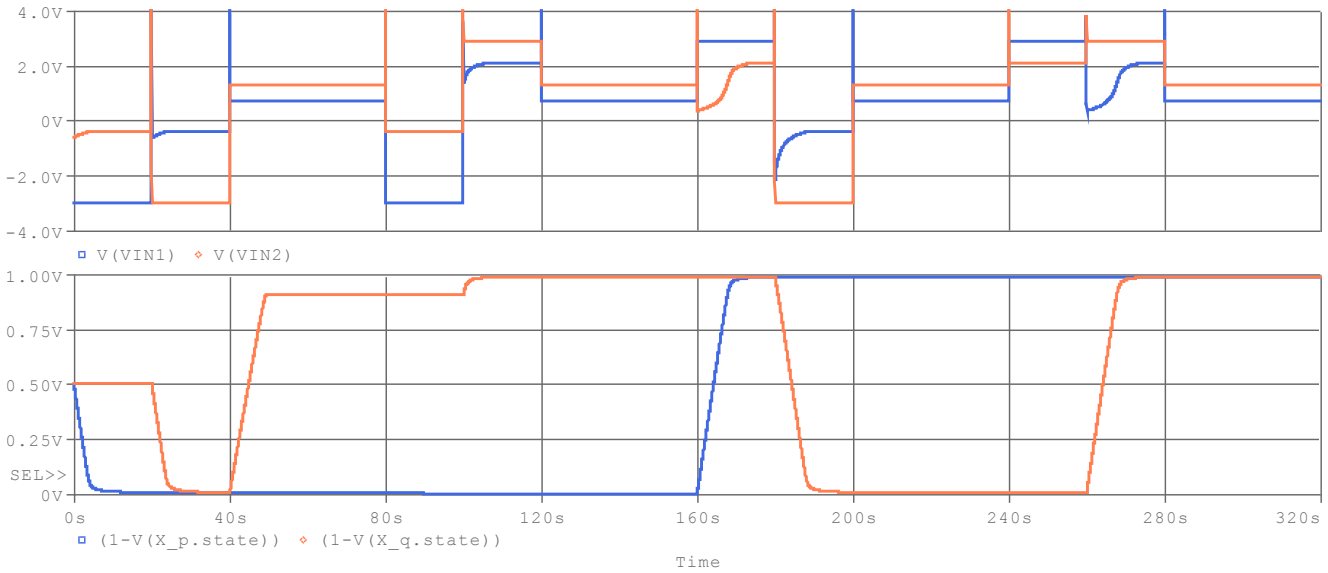


Figure 101: The top plot shows the voltage sequence applied to the gate as described by Table 18 and Table 19 above. The bottom plot shows the logic state of the two memristors p and q. Note that the logic value is given by $(1 - model\ state\ variable)$, and this is because the state variable was chosen to represent a normalised distance rather than the logic value.

The results as presented above present both good and bad news. The good news is that unlike a current threshold device, there is no state drift phenomenon meaning that the logic input P will remain intact regardless of the number of consecutive $IMPLY$ operation. The bad news is that there is inadequate switching of memristor q for the input logic case $\{P, Q\} = \{0,0\}$ as shown by the time interval $40 - 80s$. This occurs because as Q moves towards logic 1, its memristance decreases and the voltage across it falls until it is taken below its switching threshold. A larger $|v_{set}|$ can improve the situation, but then the other constraints are invalidated and the other cases start to fail.

Since all the other cases work as desired, the next step is to investigate is how the inadequate switching of input case $\{P, Q\} = \{0,0\}$ affects subsequent operations. Table 20 below shows the test sequence which tests all 4 possible operations when $\mathrm{Q} = 0.905$, and Figure 102 below shows the graphical traces.

Time (seconds):	$0 - 20$	$20 - 40$	$40 - 80$	$80 - 100$	$100 - 120$	$120 - 160$
Action:	*Clear P*	*Load Test Q*	$Q = P \rightarrow Q$	*Set P*	*Load Test Q*	$Q = P \rightarrow Q$
V_{in1}:	$-v_{clear}$	v_z	$\lvert v_{cond}\rvert$	$\lvert v_{set\_h}\rvert$	v_z	$\lvert v_{cond}\rvert$
V_{in2}:	v_z	$\lvert v_{set}\rvert$	$\lvert v_{set}\rvert$	v_z	$\lvert v_{set}\rvert$	$\lvert v_{set}\rvert$
Desired result:	$P = 0$	$Q = 0.905$	$P = 0$ $Q = 1$	$p = 1$	$Q = 0.905$	$P = 1$ $Q = 1$
Actual result:	$P = 0.006$	$Q = 0.905$	$P = 0.006$ $Q = 0.905$	$P = 0.985$	$Q = 0.905$	$P = 0.985$ $Q = 0.905$

Time (seconds):	$160 - 180$	$160 - 200$	$200 - 240$	$240 - 260$	$260 - 280$	$280 - 320$
Action:	*Clear P*	*Load Test Q*	$P = Q \rightarrow P$	*Set P*	*Load Test Q*	$P = Q \rightarrow P$
V_{in1}:	$-v_{clear}$	v_z	$\lvert v_{set}\rvert$	$\lvert v_{set\_h}\rvert$	v_z	$\lvert v_{set}\rvert$
V_{in2}:	v_z	$\lvert v_{set}\rvert$	$\lvert v_{cond}\rvert$	v_z	$\lvert v_{set}\rvert$	$\lvert v_{cond}\rvert$
Desired result:	$P = 0$	$Q = 0.905$	$P = 0$ $Q = 1$	$P = 1$	$Q = 0.905$	$P = 1$ $Q = 1$
Actual result:	$P = 0.006$	$Q = 0.905$	$P = 0.878$ $Q = 0.905$	$P = 0.985$	$Q = 0.905$	$P = 0.985$ $Q = 0.905$

Table 20: Sequence of voltages applied in order to test the $IMPLY$ gate of Figure 100, testing all 4 possible cases with Q starting from logic state 0.905. The same parameters as in Table 19 are used. The most interesting result is shown in red.

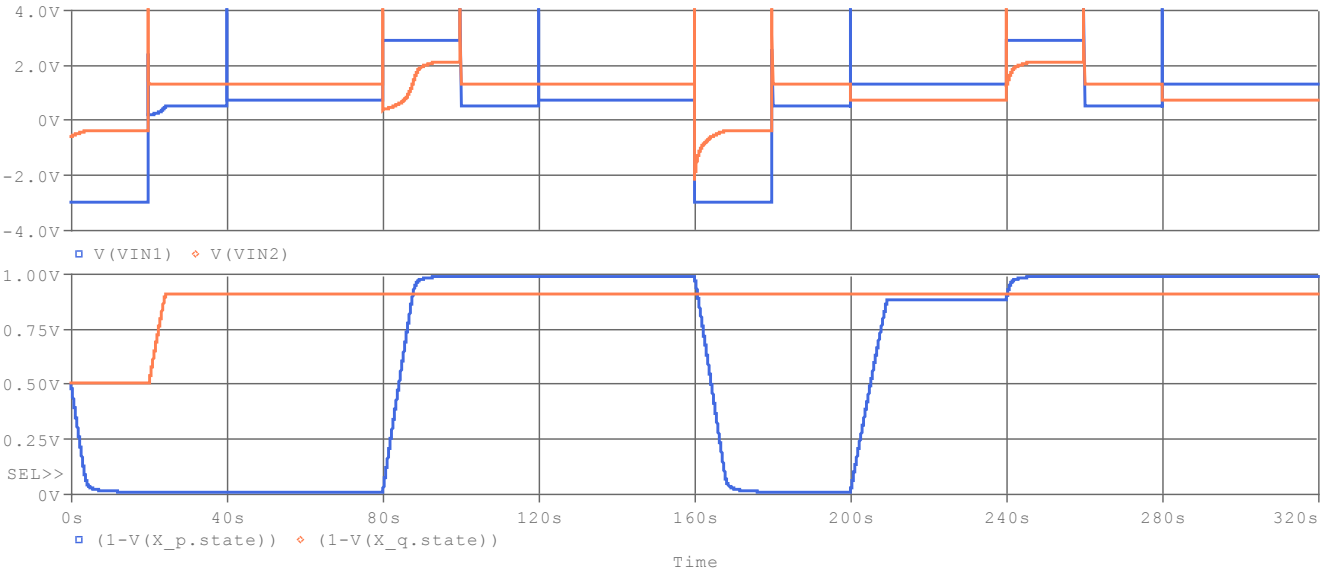


Figure 102: The test sequence as described by Table 20, and the logic state of memristors p and q.

From the above results, it is clear that the inadequate switching case causes failure for some subsequent operations. More specifically, the operation $P = Q \rightarrow P$ fails when $P \approx 0, Q = 0.905$. Failure occurs because the logic gate is developed under the assumption that $R_{on} \ll R_g \ll R_{off}$. If the devices have a voltage threshold to switch to R_{on}, then the on state is never fully reached and the lowest achievable memristance (R_{min}) is not much smaller than R_g, hence invalidating the assumption. Equation 6.4 below shows the value of R_{min} for a given $|v_{set}|$.

$$R_{min} = \frac{|v_{on}|}{\frac{|v_{set}| - |v_{cond}| - |v_{on}|}{R_{off}} + \frac{|v_{set}| - |v_{on}|}{R_g}} \approx \frac{|v_{on}| \times R_g}{|v_{set}| - |v_{on}|}, \qquad where \quad R_g \neq 0, \quad and \quad |v_{on}| < |v_{set}| \qquad 6.4$$

Since the ratio of R_g to R_{min} is a only function of v_{on} and v_{set}, this problem cannot be solved by a modification of R_g. The only option is to ensure $|v_{on}| \ll |v_{set}|$, but the scope of this is limited due to the other constrains (3.1 and 3.2).

In summary, in order to have a correctly functioning gate, the resistance of the memristor must be measured after every *IMPLY* operation, and it must be restored to either 0 or 1 based on what its state is closest to. It is not possible to achieve this by a normal *IMPLY* operation, for example the operation $Q = 1 \rightarrow Q$ should result in $Q = Q$, so this might be expected to restore the logic level. However, this is not the case and so additional circuitry and control is required. This additional complexity reduces the feasibility drastically, especially considering the already complex controller. Due to this reason, this gate is no longer further investigated as part of this project. Since the fundamental operation failed, it is meaningless to try to quantitatively compare power and speed with other families.

6.3.2 Optimised Full Adder

In anticipation of a working *IMPLY* gate, an enhanced full adder sequence was been designed and is shown by Table 21 below. This design shows improvement in both speed and memristor count compared to existing optimised sequences as compared in Table 22 later.

Based on the results in the previous section, a simulation of this sequence using the VTEAM model would be of no use. Nevertheless, this sequence is included in this report since the *IMPLY* gate has been shown to be viable for current controlled devices/models and so it is still potentially useful. [27] [43] [86]

Step:	*Operation*:	*Memristor Logic State*:				
		$M1$:	$M2$:	$M3$:	$M4$:	$M5$:
0	*Start*	A	B	C_{in}	X	X
1	$FALSE(M4)$	A	B	C_{in}	0	X
2	$FALSE(M5)$	A	B	C_{in}	0	0
3	$M4 = M1 \rightarrow M4$	A	B	C_{in}	$\overline{A}$	0
4	$M5 = M2 \rightarrow M5$	A	B	C_{in}	$\overline{A}$	$\overline{B}$
5	$M5 = M1 \rightarrow M5$	A	B	C_{in}	$\overline{A}$	$\overline{A} + \overline{B} \equiv \overline{AB}$
6	$M2 = M4 \rightarrow M2$	A	$A + B$	C_{in}	$\overline{A}$	$\overline{AB}$
7	$FALSE(M1)$	0	$A + B$	C_{in}	$\overline{A}$	$\overline{AB}$
8	$M1 = M5 \rightarrow M1$	$\overline{\overline{AB}} \equiv AB$	$A + B$	C_{in}	$\overline{A}$	$\overline{AB}$
9	$M1 = M2 \rightarrow M1$	$\overline{A + B} + AB \equiv \overline{A \oplus B}$	$A + B$	C_{in}	$\overline{A}$	$\overline{AB}$
10	$FALSE(M4)$	$\overline{A \oplus B}$	$A + B$	C_{in}	0	$\overline{AB}$
11	$M4 = M5 \rightarrow M4$	$\overline{A \oplus B}$	$A + B$	C_{in}	$\overline{\overline{AB}} \equiv AB$	$\overline{AB}$
12	$FALSE(M2)$	$\overline{A \oplus B}$	0	C_{in}	AB	$\overline{AB}$
13	$M2 = M1 \rightarrow M2$	$\overline{A \oplus B}$	$\overline{\overline{A \oplus B}} \equiv A \oplus B$	C_{in}	AB	$\overline{AB}$
14	$FALSE\ (M5)$	$\overline{A \oplus B}$	$A \oplus B$	C_{in}	AB	0
15	$M5 = M1 \rightarrow M5$	$\overline{A \oplus B}$	$A \oplus B$	C_{in}	AB	$\overline{\overline{A \oplus B}} \equiv A \oplus B$
16	$M1 = M3 \rightarrow M1$	$\overline{A \oplus B} + \overline{C_{in}}$	$A \oplus B$	C_{in}	AB	$A \oplus B$
17	$M4 = M1 \rightarrow M4$	$\overline{A \oplus B} + \overline{C_{in}}$	$A \oplus B$	C_{in}	$AB + \overline{\overline{A \oplus B} + \overline{C_{in}}} \equiv C_{out}$	$A \oplus B$
18	$M5 = M3 \rightarrow M5$	$\overline{A \oplus B} + \overline{C_{in}}$	$A \oplus B$	C_{in}	C_{out}	$A \oplus B + \overline{C_{in}}$
19	$M3 = M2 \rightarrow M3$	$\overline{A \oplus B} + \overline{C_{in}}$	$A \oplus B$	$\overline{A \oplus B} + C_{in}$	C_{out}	$A \oplus B + \overline{C_{in}}$
20	$FALSE(M2)$	$\overline{A \oplus B} + \overline{C_{in}}$	0	$\overline{A \oplus B} + C_{in}$	C_{out}	$A \oplus B + \overline{C_{in}}$
21	$M2 = M5 \rightarrow M2$	$\overline{A \oplus B} + \overline{C_{in}}$	$\overline{A \oplus B + \overline{C}_{in}}$	$\overline{A \oplus B} + C_{in}$	C_{out}	$A \oplus B + \overline{C_{in}}$
22	$M2 = M3 \rightarrow M2$	$\overline{A \oplus B} + \overline{C_{in}}$	$A \oplus B \oplus C_{in} \equiv \boldsymbol{Sum}$	$\overline{A \oplus B} + C_{in}$	$\boldsymbol{C_{out}}$	$A \oplus B + \overline{C_{in}}$

Table 21: Proposed sequence for a full adder. A and B denote the binary inputs, and C_{in} the carry input. C_{out} denotes the carry output.

	Memristors:	*Steps*:
Proposed sequence (Table 21)	$2N + 3$	$22N$
Teimoory et al. [29]	$3N + 3$	$23N$
Kvatinsky et al. [43]	$3N + 3$	$29N$
Lehtonen and Laiho [87]	$3N + 5$	$89N$

Table 22: Performance of N-bit adder sequences. Kvatinsky et al. also present a parallel approach which trades of the step count with number of memristors and crossbar/driver complexity. [43]

The proposed sequence requires 5 memristors and 22 steps in order to implement a full adder. This is one less memristor and one less step then the next best alternative as proposed by Teimoory et al. [29] By the end of the sequence, the sum result is written to $M2$ and the carry out (C_{out}) to $M4$. If desired for some reason, the Sum result can be moved to $M5$ simply by swapping the role of M5 and $M2$ after step 10. However, the version as presented by Table 21 is deliberately chosen since it is slightly more convenient for multiple bit adders.

For implementing an N-bit full adder, the previous input memristors can be reused in order to aid the calculation of the next bit. Therefore, only $2N + 3$ memristors are required where the additional 2 devices per bit store the additional 2 bits to be added together. In this case, the sum result can be written to memristors $M2, M6, M8, M10, M12 \dots etc$. The carry bits can be written to any unused memristor; for simplicity it could just toggle between $M4$ and $M1$ as each additional bit is computed.

6.4 MAGIC

This section further investigates the MAGIC logic family as introduced in 3.5. The focus is on the NOR and NOT gates since these are the only gates which can be made in the convenient crossbar structure which allows more advanced logic. Note that the NOT gate is just a single input NOR gate. The aim in this section is to verify the operation of this gate, and highlight limitations. As it turns out, this gate is not feasible with the VTEAM model with the parameters as used previously; however, operation is successful with modified threshold parameters. This statement is justified next.

Equation 6.5 below repeats the design requirements for a N-input NOR gate assuming voltage threshold devices. Equation 6.6 simplifies this expression by substituting the some of VTEAM model parameters and with $N = 2$. From this it is clear that this gate is not actually feasible unless $|v_{on}| > 1.951 \times v_{off}$ since otherwise all the inequalities cannot be simultaneously satisfied. If the lower inequality is breached, then the output fails to change from 1 to 0 when any one (or both) of the input is high. If the upper inequalities are breached, then the inputs start to drift during operation and the output can also drift from 1 to 0 for the case when both inputs are low. Both of these violate the fundamental NOR operation.

$$\frac{v_{off}}{R_{on}} \times \left(R_{on} + \left(\frac{R_{off}}{N-1}\right) || R_{on}\right) < v_{nor} < \min\left[\left(v_{off}\left(1 + \frac{R_{off}}{N \times R_{on}}\right)\right),\ \left(|v_{on}|\left(1 + \frac{N \times R_{on}}{R_{off}}\right)\right)\right] \quad 6.5$$

$$\left(1.99 \times v_{off}\right) < v_{nor} < \min\left[\left(51 \times v_{off}\right),\ \left(1.02 \times |v_{on}|\right)\right] \quad 6.6$$

In order to prove correct operation under the assumption that $|v_{on}| > 1.951 \times v_{off}$ is true, the parameter v_{on} in the VTEAM model is temporarily changed from $-0.8V$ to -2V. The analysis is as follows.

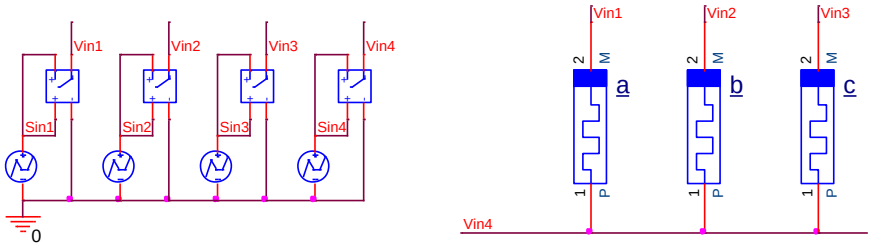


Figure 103: NOR gate in crossbar form as introduced in section 3.5. Memristors a and b are used as inputs and c as the output. The NOR operation is only successful if $|v_{on}| > 1.951 \times v_{off}$.

Figure 103 above shows the NOR gate in crossbar form. For the purpose of these tests, memristors a and b are used as inputs and c as the output. The test sequence and steady state results are summarised by Table 23 below. This sequence is similar to previous ones except now there is an additional setup stage since the output memristor must be initialised to logic 1. The simulation results are presented by Figure 104 below.

$Time\ (sec)$:	$0-20$	$20-40$	$40-60$	$60-80$	$80-100$	$100-120$	$120-140$	$140-160$
$Action$:	$Init.o/p$	$Init.i/p$	$Init.i/p$	NOR	$Init.o/p$	$Init.i/p$	$Init.i/p$	NOR
V_{in1}:	v_z	$-v_{clear}$	v_z	v_{nor}	v_z	$-v_{clear}$	v_z	v_{nor}
V_{in2}:	v_z	v_z	$-v_{clear}$	v_{nor}	v_z	v_z	$\lvert v_{set}\rvert$	v_{nor}
V_{in3}:	$\lvert v_{set}\rvert$	v_z	v_z	gnd	$\lvert v_{set}\rvert$	v_z	v_z	gnd
V_{in4}:	gnd	gnd	gnd	v_z	gnd	gnd	gnd	v_z
$Desired\ result$:	$C=1$	$A=0$	$B=0$	$A=0$ $B=0$ $C=1$	$C=1$	$A=0$	$B=1$	$A=0$ $B=1$ $C=0$
$Actual\ result$:	$C=0.993$	$A=0.006$	$B=0.006$	$A=0.006$ $B=0.006$ $C=0.993$	$C=0.997$	$A=0.004$	$B=0.991$	$A=0.002$ $B=0.991$ $C=0.006$

$Time\ (sec)$:	$160-180$	$180-200$	$200-220$	$220-240$	$240-260$	$260-280$	$280-300$	$300-320$
$Action$:	$Init.o/p$	$Init.i/p$	$Init.i/p$	NOR	$Init.o/p$	$Init.i/p$	$Init.i/p$	NOR
V_{in1}:	v_z	$\lvert v_{set}\rvert$	v_z	v_{nor}	v_z	$\lvert v_{set}\rvert$	v_z	v_{nor}
V_{in2}:	v_z	v_z	$-v_{clear}$	v_{nor}	v_z	v_z	$\lvert v_{set}\rvert$	v_{nor}
V_{in3}:	$\lvert v_{set}\rvert$	v_z	v_z	gnd	$\lvert v_{set}\rvert$	v_z	v_z	gnd
V_{in4}:	gnd	gnd	gnd	v_z	gnd	gnd	gnd	v_z
$Desired\ result$:	$C=1$	$A=1$	$B=0$	$A=1$ $B=0$ $C=0$	$C=1$	$A=1$	$B=1$	$A=1$ $B=1$ $C=0$
$Actual\ result$:	$C=0.991$	$A=0.991$	$B=0.008$	$A=0.991$ $B=0.007$ $C=0.008$	C=0.991	A=0.997	C=0.991	$A=0.997$ $B=0.991$ $C=0.008$

Table 23: Test sequence and steady state results for the NOR gate of Figure 103 with the modified VTEAM parameter (v_{on} is temporarily changed from $-0.8V$ to $-2V$). $v_{set}=-2.9V, v_{clear}=3V, v_{nor}=1.9V$, and v_z denotes high impedance mode.

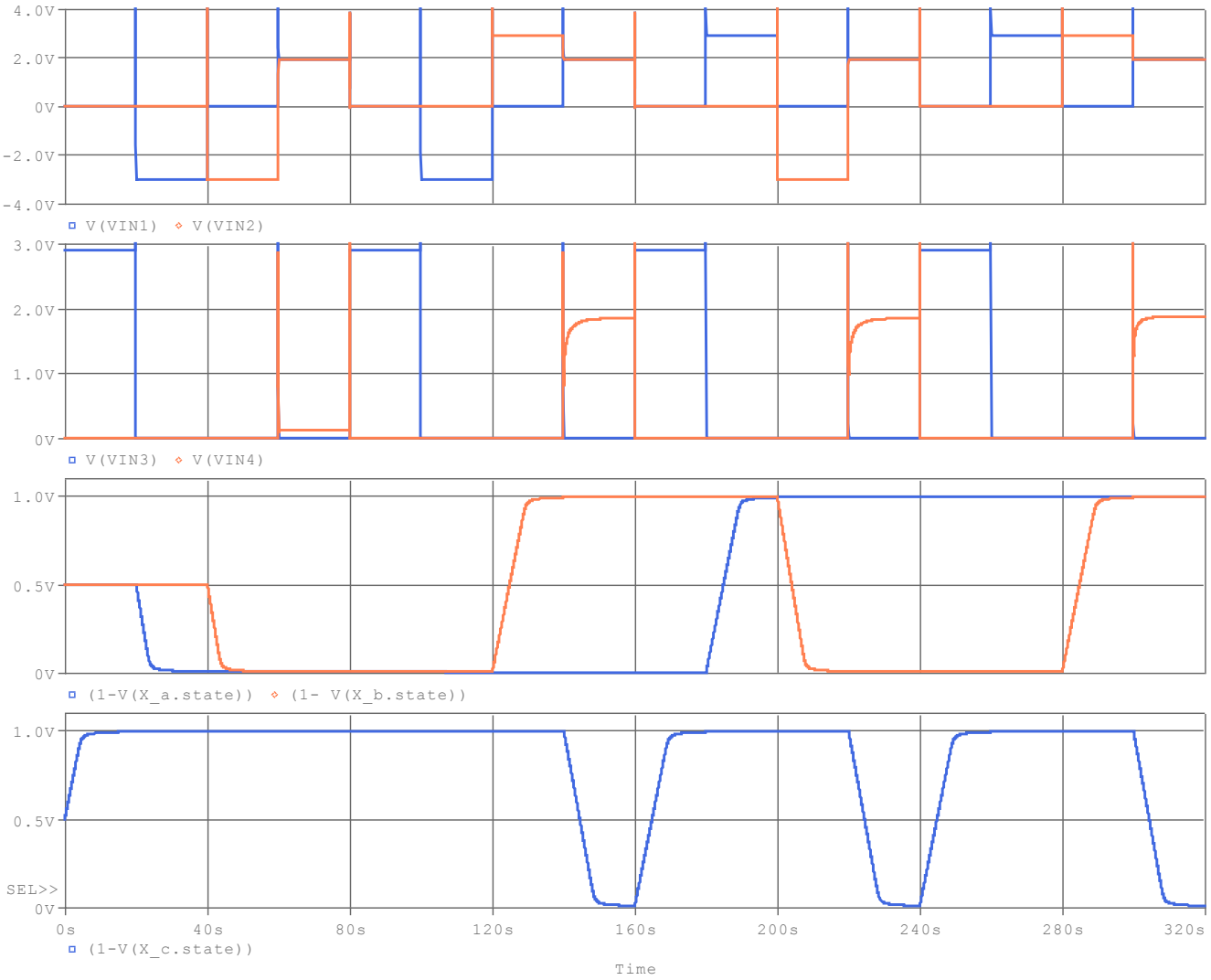


Figure 104: The simulation results showing the input sequences, input logic states and output logic state. The intervals of interest are 60-80s, 140-160s, 220-240s and 300-320s since this is when the NOR is performed. The other parts are for initialising the inputs and output.

These results verify correct operation. There is also no state drift and the output is self-restoring. For example if one of the input states is 0.015, the output is restored to 0.008. This means there is no issue in sequential operations and more advanced logic. However, A full adder is not simulated as part of this project due to time constraints, and because it the results would serve no purpose for comparison with other logic families since the model parameter is different to the one previously used.

6.5 Evaluation and Comparison

Based on the background research and the information gathered from simulations, the families are qualitatively compared with each other. Unfortunately it is not possible perform quantitative comparison since the particular memristor model used resulted in unsuccessful operation of the MAGIC and Implication Logic families. This does not automatically mean MRL is superior, it simply means MRL is most suitable compared to the rest based on the current model fitted to the data at hand. Each family has its own preferred memristor type which it works best with, and this is also discussed below.

6.5.1 Implication logic:

Simulation results concluded a device with a voltage threshold is not viable for the $IMPLY$ gate since full switching of the output memristor cannot be achieved unless the voltages used are much higher than the threshold, but this result in the destruction of inputs. One option to solve this issue is to include additional CMOS circuitry which measures the resistance state of the output after each $IMPLY$ operation and restores the state to the closest match. This mostly defeats the objective of the Implication family which aims to reduce area and transistors required.

A device with a well-defined current threshold is preferred for correct logical operation. However, for such a device the output tends to drift to logic 1 when it should not i.e. for the logic input case $\{1,0\}$. A nonlinear memristor with a well-defined current threshold is desired to minimise this state observed drift effect. [43] [27]

Advantages:

1. The $IMPLY$ operation can be performed on memristors in a crossbar structure which means high packing density. Additionally, the crossbar structure enables the output of one operation to easily be used as the input of another operation on the same row in the crossbar meaning a sequence of operations ($IMPLY$ and $FALSE$) can be performed to produce the desired logic function. Since the minimum number of devices required for any N-input Boolean function is $N + 2$, this family can require the least amount of area compared to traditional CMOS and all other memristor based families. However, this statement does not consider the area for the controller. Factoring the controller in is tricky since no such controller designs have been proposed as of yet.
2. The same crossbar can perform any logic; only the controller sequence needs to be changed.
3. The crossbar structure used for $IMPLY$ is the same as what is used for memristor based memory. This coupled with the fact that the logic level is stored as the resistance means there is possibility for advanced architectures beyond the classical Von-Neuman and Harvard (for these the data storage and logical processing is separated).
4. Non-volatile logic values and memory means possibility of high resilience to power outages and instant start up times.
5. In implementing a given logic function there is great flexibility in trading off speed of operation (number of steps) and number of memristors required. There is even scope of utilising multiple rows of the crossbar simultaneously to perform parallel computations. For example, for multi-bit adders, each bit can be computed on a separate row. The downside to this is that additional complexity introduced since the carry results need to be transferred between rows and this is not as trivial as transferring values in within the same row. Kvatinsky et al. propose such a parallel multibit adder sequence. [32]

Disadvantages:

1. A long sequence of steps is necessary to implement non-trivial logic which means a complex controller is required and the overall operation is significantly slower compared to CMOS or hybrid CMOS which is highly parallel in comparison. This controller adds complexity, increases power consumption and increases chip area.

2. Unless there are isolating switches which segregate the rows in the crossbar, it is impossible to perform more than one $IMPLY$ operation at the same time. This eliminates the opportunity for pipelining of the sequence of steps. Simultaneous $FALSE$ operations on the same row are possible however.
3. One input (Q) is always destroyed by the $IMPLY$ operation since it is overwritten with the output of $P \rightarrow Q$. There is possibility for the other input (P) to also be destroyed after repeated operations due to state drift. This is because the state drift of Q for input case $\{P, Q\} = \{0,1\}$ (as introduced in section 3.3.2) can actually be traded off with state drift of P for the input case $\{1,1\}$ as demonstrated by Statat et al. [49] This state drift discussion applies only to devices with a current threshold. For devices with a voltage threshold, state drift can be eliminated provided the design guidelines given by 3.1 and 3.2 are satisfied.
4. Since at least one input is always destroyed, it must first be copied if it needs to be used multiple times. This takes a sequence of 4 steps and requires a spare work memristor. This copy operation is useful only if there is negligible state drift, otherwise the value to be copied is destroyed during the operation.
5. There is a trade-off between speed of operation and robustness due to the state drift phenomenon. This is a characteristic of a particular device and depends on the switching characteristic as well as the thresholds (which restrict input voltages). [43]
6. Another source of state drift is sneak paths due to the crossbar structure. [88] This introduces further design guidelines on selecting voltage values, though this is not simple to quantify. This is particularly an issue for current threshold devices.
7. The logic level is stored as the resistance level meaning some sort of interface is needed to connect with standard CMOS logic.
8. Design of optimised logic sequences is non-trivial since many factors need to be considered and balanced. For example there is the possibility to choose equivalent Boolean operations such that the input is not destroyed so that it can be re-used later (a copy is expensive). There is also opportunity to design sub-sequences that produce by-products which are useful for later calculations. These are the optimisations used in the optimised adder sequence proposed in section 6.3.2.

6.5.2 MRL

Since MRL operates on the basis of a potential divider, a high R_{off}/R_{on} ratio is preferred. Furthermore, the simulation results show that a device with a low $|v_{on}|$ is also desired in order to allow complete switching. If the on-switching voltage is not negligible compared to the logic high voltage (v_{high}), then complete switching cannot occur and the steady state output error is dominated by the threshold meaning it is close to $|v_{on}|$ rather than $v_{high}R_{on}/(R_{on} + R_{off})$. The threshold v_{off} is less critical, but it also needs less than $v_{high}/2$ to allow recovery from the R_{on} state in the worst case scenario. Also, a large v_{off} is also undesired because it means higher dynamic power consumption because the memristor remains in a low resistance state for a longer time compared to if $v_{off} \approx 0$.

Devices with current thresholds can achieve complete switching provided $v_{high}/(R_{off} + R_{on}) > \min(i_{on}, i_{off})$, (valid for a two input gate). Experimentation with MRL gates using both VTEAM and TEAM models show linear and low threshold devices result in lowest power consumption and faster switching for a given v_{high}. In general, devices with a high thresholds require larger voltages in order to ensure the steady state error is small relative to v_{high}. However, this results in greater static and dynamic power consumption, and the scope of this is also limited by the voltage range which is compatible with CMOS logic.

Advantages:

1. Logic circuits which are a hybrid of CMOS and memristors can use less area than CMOS only implementations. Case studies on adders demonstrate/estimate a reduction of 50% to 75%. [48] [23] [24]
2. CMOS buffers can be used to provide isolation between memristor gates which can prevent logic faults and increase fan out and reduce power consumption.
3. It is faster than MAGIC and Implication Logic since the operation is performed in a single step compared to the others which require a sequence. Parallelisation is simpler and more flexible compared to the MAGIC and Implication families which are inherently sequential.

Disadvantages:

1. There are dynamic hazards for both AND and OR gates whenever the input changes to logic $\{1, 0\}$ or $\{0, 1\}$. As a result, compound logic is prone to many glitches.
2. Static power can be consumed even once the output has settled to the correct logic state. This can be fixed by powering down the signal to the inputs once the output has settled. A high R_{off} is desired to minimise this consumption.
3. More chip area is used compared to $IMPLY$ since more memristors are required and CMOS inverters and/or buffers may be required.
4. If no buffers or inverters are used, the outputs of cascaded gates deteriorate due to the potential dividers formed. This can cause logic failure especially if the memristors have voltage or current thresholds.
5. The need for CMOS inverters/buffers means the logic voltage levels are restricted to CMOS compatible ones.
6. Power consumption is expected to be higher than Implication logic. Implication is efficient since most of its consumption is associated the the switching stage and there is no static consumption.

6.5.3 MAGIC

Only the MAGIC NOT and NOR gates are of interest since only these can be arranged in the standard crossbar structure. Analysis in section 3.5.3 showed that memristors with current thresholds are not suitable since the NOR output cannot fully switch to R_{off} when required; in fact it can only approach R_{off}/N where N is the number of inputs. The simulation results revealed further restrictions for devices with a voltage threshold. In this case the design guidelines given by equation 6.5 require $|v_{on}|$ to be greater than approximately $2 \times v_{off}$. This is possible and has been observed in some memristors, e.g. a ferroelectric memristor, [28] or some chalcogenide based memristors. [72] Note that the memristor direction conventions used in [72] are opposite to the ones used here and *'Forward adaption threshold'* in [72] refers to v_{on}.

Advantages:

1. The Implication family requires 3 steps to perform $NAND$ or NOR operation, whereas the MAGIC family requires only 2 steps. Unlike the $IMPLY$ gate, the output is not overwritten to one of the inputs. Therefore no copy operations are required. The combination of these two advantages means MAGIC sequences can be shorter than Implication sequences. However, Implication Logic is not restricted to only executing logic by a sequence of $NAND$ operations and so there is scope for optimisation.
2. Memristors can be in the standard crossbar form whose advantages have already been discussed
3. If the device has voltage thresholds which satisfy the design guidelines, then there is no state drift effect and operation is highly robust regardless of the linearity or non-linearity of the model. The output is even self-restoring.

4. Each operation for the MAGIC gate is faster than the Implication case since no series R_g is present. There is only $R_{on} \ll R_g$ (or less due to parallel input memristors) in series. This can allow lower voltages to be used and lower power consumption compared to Implication Logic.

Disadvantages:

1. Only NOR and NOT gates can be in the crossbar structure, so only these operations are feasible. Other gates have a complex interconnection with many isolating switches and intricate control.
2. Logic is implemented by a sequence of operations just like the Implication case
3. A sequence of operations and a complex driver for the crossbar is required. No operations can be performed in parallel in the same row of the crossbar unless there are isolating switches.

6.6 Conclusion

This section investigated the three main logic families with the VTEAM model fitted to data from a TiO_2 device. The results revealed interesting limitations of all the families and this is mostly due to the explicit voltage thresholds present in the model. These results allowed several gaps in research to be filled, and thus allowed a more comprehensive comparison of the families as discussed in the previous section. The process also gave additional insight into the specific requirements of each logic family. The specific requirements are not an issue for the future of memristor based logic since memristors are still in their early days and there already exist a wide range of devices with different properties suitable for different applications.

Based on the results, the MRL family seems most feasible to work with the TiO_2 devices of interest. In general, this family has the least tight requirements and simplest operation, making it an attractive option for further study. The next section presents preparations for practical investigations in anticipation of TiO_2 devices, and briefly describes the first circuit which was planned to be implemented in order to demonstrate the basic MRL working principles. The main reason for prioritising MRL (for the purpose of this project) is that it does not require a complex sequence of operations and it more easily allows the same input to be used for multiple gates. Furthermore, the fan out and voltage degradation problems have simple solutions, i.e. buffers.

7 Practical Preparation

7.1 Introduction

During the project, it was unknown that devices would be unavailable until the very last few days. Therefore in preparation, an instrumentation circuit was designed and tested in order help investigate memristor behaviour and aid the practical implementation. A MRL based circuit was also designed with the intention to demonstrate the basic operation. Based on the success of this, the plan was to implement the full adder designed in section 6.2.3.8. the work done in preparation is presented in this section.

7.2 Instrumentation Design

In order to investigate the memristors switching behaviour, it is necessary to be able to apply precise state changing pulses of desired voltage and duration. Additionally, a means of measuring resistance without changing the state is required. A digital multimeter cannot be used for this purpose since they can affect the state due to the long duration of measurement as well as relatively high measuring voltage. Furthermore, the high open circuit voltages of multimeters can permanently damage the devices. It is also desirable to be able to produce IV curves of devices.

These objectives can actually all be achieved simply by using a signal generator, an oscilloscope and a current sense resistor. However, a National Instruments Data Acquisition Device (NI-USB 6009 DAQ) was chosen instead. This is due to the great flexibility of Virtual Instrument (VI) and system design and ability for the DAQ to handle multiple analog/digital inputs/outputs simultaneously with high data rate and accuracy. The additional flexibility is preferred since other logic families such as Implication logic can be explored more easily using a DAQ. Even though a virtual instrument can be initially tricky to design and implement, once completed, the graphical front panel allows simple and efficient control and therefore fast data collection. Another advantage is its high portability and convenience. Some disadvantages of using the DAQ compared to a digital oscilloscope and signal generator setup are listed below:

1. The analog output range of the DAQ is $0 - 5v$, i.e. no negative voltages are possible.
2. The output rate is $150Hz$. However, this is not a big issue since the TiO_2 devices are slow.
3. The analog inputs have relatively low accuracy for low input voltages in the range of -0.1 to $0.1V$ (In single ended input mode).
4. The input impedance is relatively low at $144\ K\Omega$.
5. The analog input range is $\pm 10V$ or $\pm 20V$ for differential input with $48Khz$ sample rate.

The instrumentation circuit shown by Figure 105 below was designed to overcome some of these issues. This essentially provides the required buffering, offsets, and gain. The gain is useful since the amplifier noise is less than the DAQ measurement noise so amplifying, measuring, and then scaling down results in less noisy and more accurate voltage readings. The current sense amplifier has a digitally controlled gain, and this is useful since the memristance can take on a wide range of values which results in a large range of current sense voltages. The gain can be adapted to bring the sense voltages within the operating range of the DAQ for best accuracy. The amplifiers were specifically chosen to be low voltage such that they can be powered by batteries for portability ($8\ AA$, or AAA batteries, $6V$ dual supply). Portability was desired during the time of design since it was anticipated that the memristor investigation would be performed under a probing station where there may or may not be easy access to power supplies. Higher supplies up to $15V$ can be used as well.

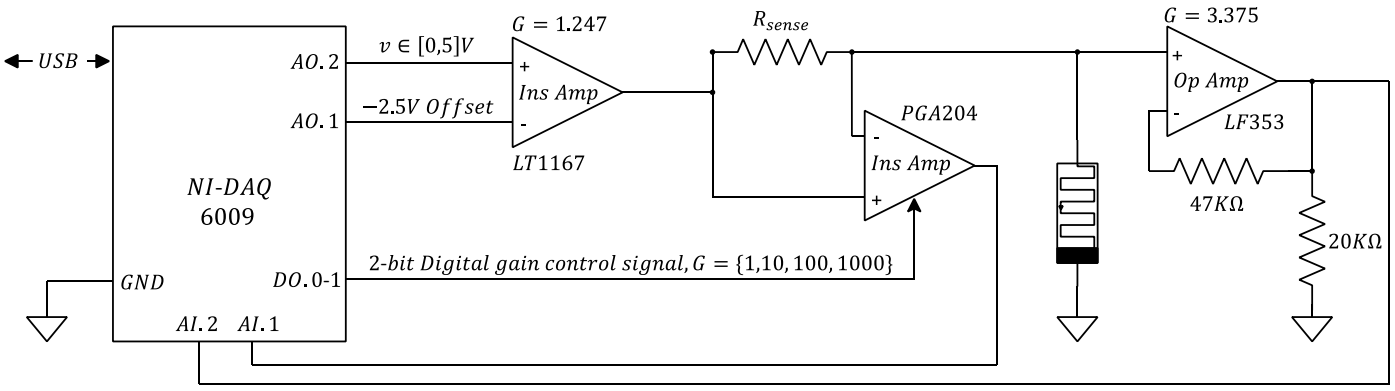


Figure 105: Instrumentation circuit which forms the interface between the DAQ and the memristor or other device under test. It is used in conjunction with the LabVIEW Virtual Instruments as described next. (G denotes closed loop)

The front panel for the VI which generates configurable state changing voltage pulses and measures memristance is shown by Figure 106 below. It allows easy change of settings and displays all voltages and intermediate calculations of interest in order to help monitor possible malfunctions.

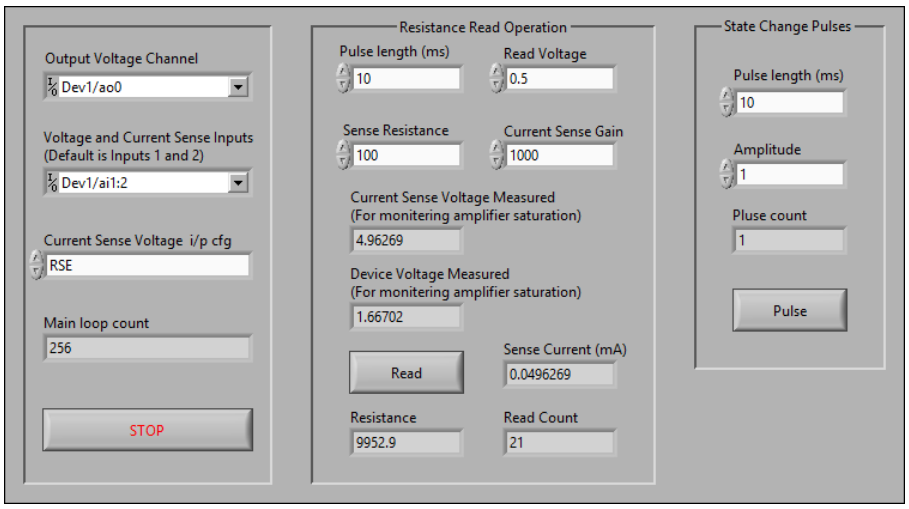


Figure 106: Front panel for the VI which generates configurable state changing voltage pulses and measures memristance.

Figure 107 below shows the VI for collecting IV data. The main features are listed below:

1. Minimum output rate is $1ms$ which is interesting since the maximum output rate of the DAQ is supposed to be $150Hz$ which corresponds to $6.66ms$.
2. The voltage output range is $\pm 2.5 \times 1.247 = \pm 3.12V$.
3. The device voltage and current measured are displayed live along with a live IV curve.
4. There is a button to save the collected data to a CSV file.
5. Warnings are displayed if there is any voltage clipping is detected at any stage in the circuit of Figure 105. This is useful when powered from batteries, since the working ranges are smaller and clipping can easily occur.
6. Sinewave and square wave outputs with desired offsets and frequencies are supported.

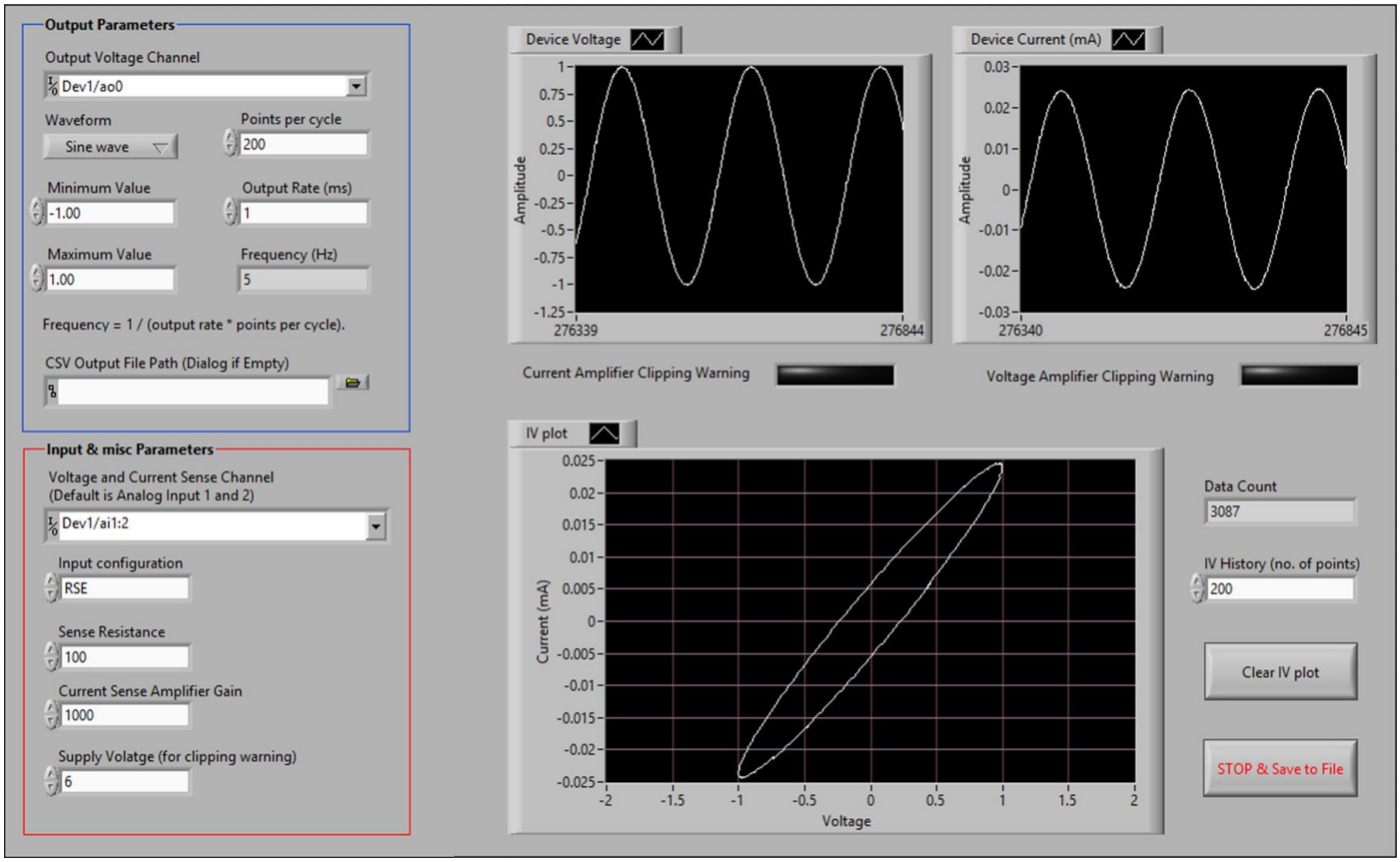


Figure 107: The font panel for the VI which allows *IV* data to be collected. The *IV* curve of a *'test device'* consisting of a $30K\Omega$ resistor in series with a $220nF$ capacitor is shown. This test device is chosen since it gives a more interesting *IV* shape compared to just a resistor which is results in just a diagonal line.

The figures below show the block diagrams for the two VI's as described above.

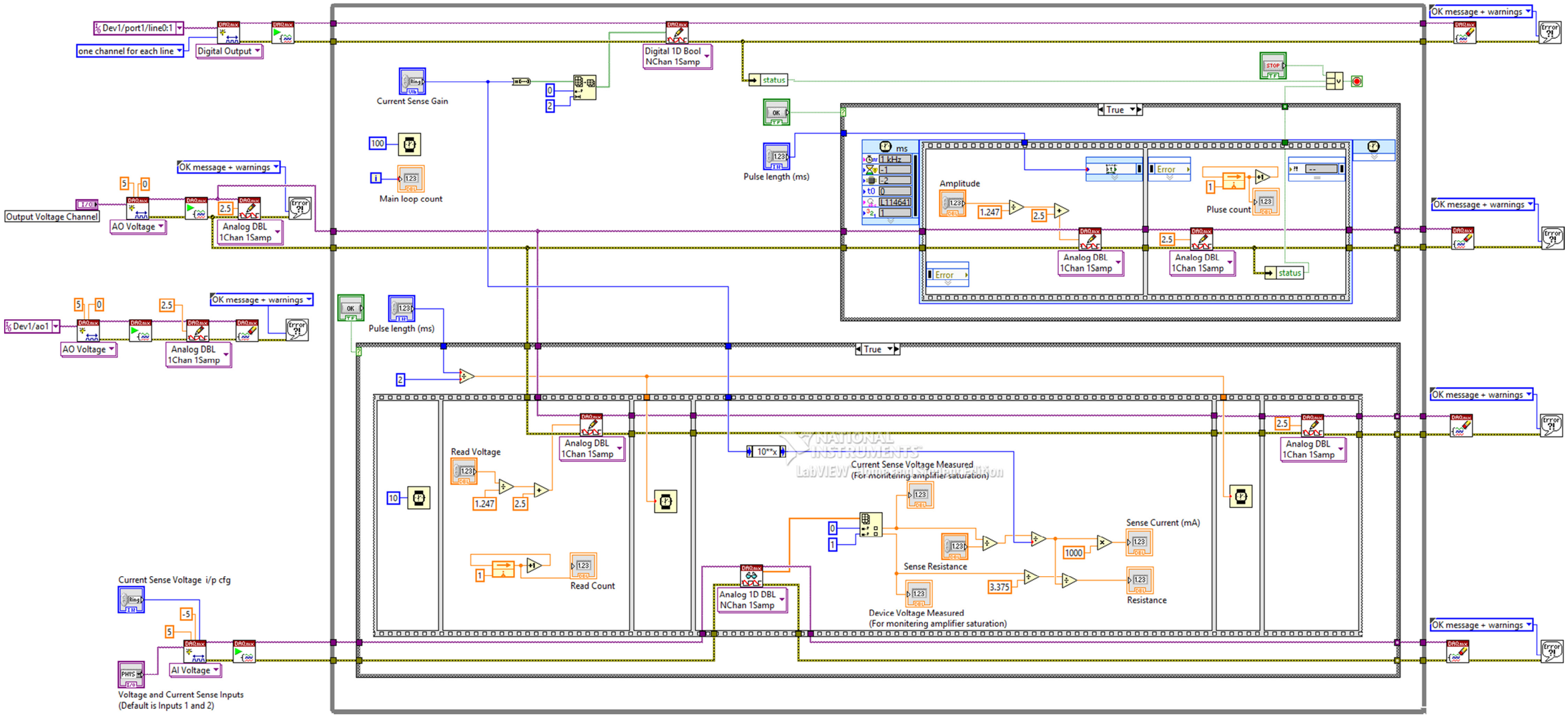


Figure 108: Block Diagram for the Virtual Instrument which generates pulses and measures memristance.

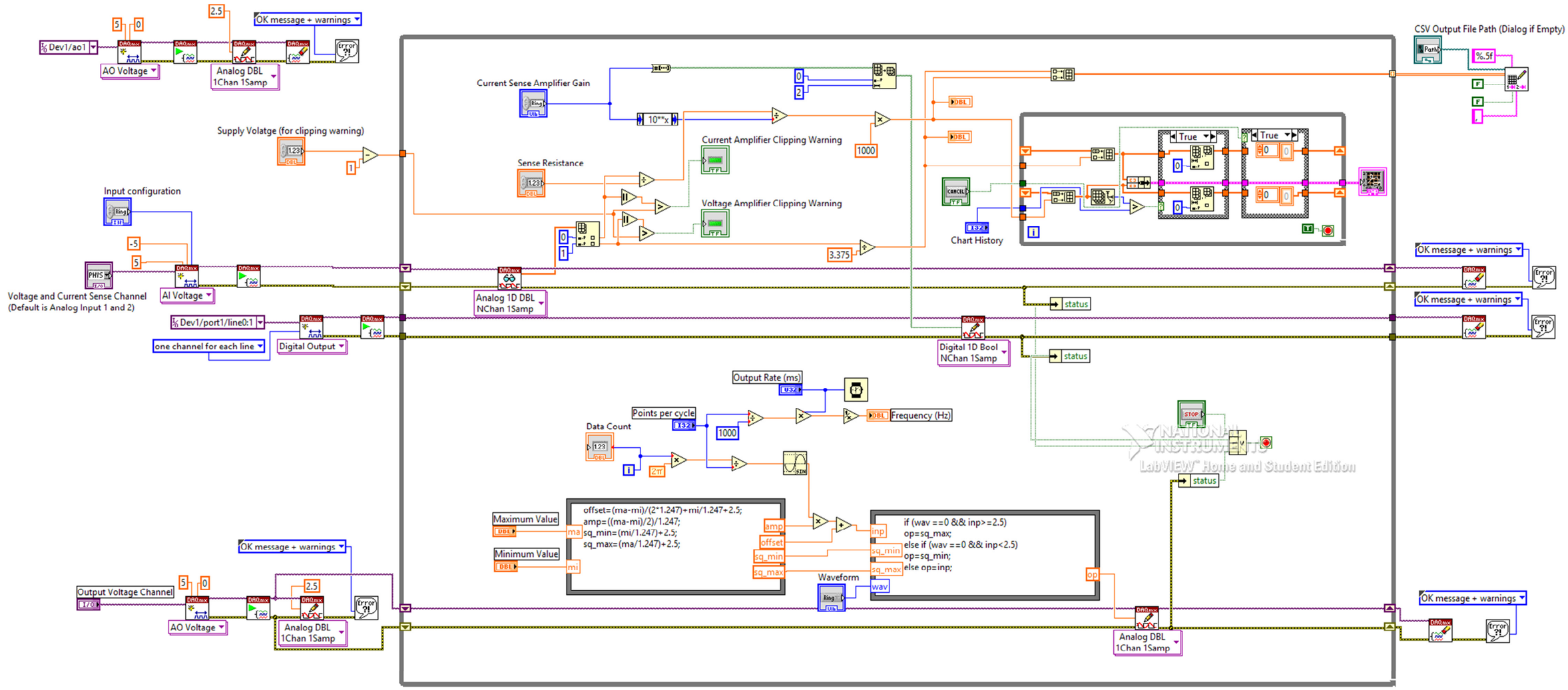


Figure 109: Block Diagram for the Virtual Instrument which collects *IV* data.

7.3 Instrumentation Test

The VI which generates the precise pulses and measures memristance is tested first. Figure 110 below shows some example positive and negative pulses applied to a $10K\Omega$ test resistor (with $R_{sense} = 100$) and they confirm correct operation with error less than 1% in terms of both voltage and time. This is sufficient for all intents and purposes. The minimum pulse with possible whilst maintaining timing accuracy is $6ms$, and this roughly agrees with the $150Hz$ output rate in the DAQ specification.

Figure 110: Verifying operation of the pulse generating VI. The left shows a pulse of amplitude $1V$ with a duration of $10ms$. The right shows a $-0.6V$ pulse of $6ms$ duration. Accuracy for amplitude and time is within 1% as shown by the cursor measurements.

Table 24 below summarises the results for the accuracy of the resistance measurements. The 'True' resistance is measured using a multimeter accurate to ~0.1% (HAMEG HM8011-3). The voltage pulse applied during the measuring process is $0.5V$ and $10ms$ in duration, and an average of 5 readings are taken. This is something expected to be suitable for the TiO_2 memristors since the change in state should be negligible after this pulse. The results show percentage error to be less than 0.7% which is more than sufficient for the application in question.

Table 24 shows accuracy and relative variance to generally worsen as the test resistance increases. This is mostly due to the voltage measurement accuracy of the DAQ which is $\pm 14mv$ for a single ended input in the range of $\pm 10V$. Since the error is independent of the signal amplitude, the relative accuracy reduces at high resistance measurements since the current sense voltage value is relatively smaller. In hindsight, these results should have been taken using differential ended input mode instead of single ended input. Under this mode of operation, the accuracy is greater, and actually dependant on the range of the input signal, meaning relative errors should be more consistent and smaller. For example, for signals in the range of $\pm 10V$, the error is $\pm 7.73mV$, whereas for signals in the range of $\pm 1V$, the error is only to $1.53mV$.

'True' Resistance: Ω	*Current Sense Resistor*: Ω	*Current Sense Gain*:	*Measured Resistance*: Ω	*Relative Variance*: (σ/μ)	*Percentage Error*:
$1.022K$	99.31	10	$1.019K$	0.225	-0.294
$5.061K$	99.31	100	$5.051K$	0.255	-0.198
$9.982K$	99.31	100	$10.01K$	0.420	0.281
$51.09K$	99.31	1000	$51.19K$	0.195	0.196
$99.76K$	99.31	1000	$99.51K$	0.481	-0.251
$511.4K$	99.31	1000	$514.9K$	0.615	0.684

Table 24: Summary of test resistance measurements. All measurements are taken at the same approximate temperature and time.

Correct operation of the VI which produces IV curves is demonstrated by Figure 107 in the previous section. Here the test device is a $30K\Omega$ resistor in series with a $220nF$. This is chosen simply because it gives a produces an interesting IV curve compared to a standard resistor. Surprisingly, this VI is able to accurately output samples with a minimum period of $1ms$ $(1000Hz)$ and this disagrees with the DAQ specification which states a max output rate is $150Hz$.

7.4 Implementation Plan

A hybrid XOR gate based on 3 MRL gates and 7 op-amps was the first circuit planned to be attempted. It would directly implement the Boolean equation $\left(A \wedge \overline{B}\right) \vee \left(B \wedge \overline{A}\right)$. The purpose of the op-amps is to act as buffer or inverter for the inputs, output and intermediate signals. The advantage of an op-amp is that it can be used as a comparator or inverter with an adjustable threshold which can be set by a potential divider. Furthermore, a rail to rail single supply low voltage op-amp was considered (MCP6001). This allows the logic high value to be adjusted by controlling the supply voltage to all amplifiers, and this can be achieved with an adjustable voltage regulator, (e.g. LM317T). The DAQ can be used to provide all input signals as well as measure all voltages of interest. Overall, this circuit was intended to simple and highly flexible in terms of adjustable thresholds and logic values, therefore giving it the largest change of success. Based on the success, the plan was to then attempt the adder design as proposed in section 6.2.3.8.

8 Evaluation

The main objective of this project was to attempt to implement a memristor based logic circuit such as a full adder. Due to the unavailability of devices, the focus remained on simulation based design and analysis instead. The VTEAM model was implemented in PSpice with a slight modification to increase fitting accuracy, and was then fitted to practical TiO_2 memristor data. The models suitability along with the fitting result is evaluated in section 5.5.

The fitted model was then used to investigate the three logic families. MRL showed success to a certain extent, but there is scope for improvement (in terms of reduced buffering) if memristors with the preferred properties are used instead. MAGIC and Implication Logic showed little success with this particular fitted model. However, they are operationally feasible given suitable memristor properties. The in-depth evaluation and comparison of these families is presented in section 6.5.

Although the MRL simulations showed some success and some useful observations, there is room for improvement and further work. The main issue is that the TiO_2 memristors that were used for fitting are so slow that the effects of the CMOS transistors are negligible in terms of propagation delay and power consumption. This somewhat limits the conclusions that can be drawn from the simulation results in terms of which design is best. For example, the proposed XOR design in section 6.2.3.6 may only be most efficient since the ratio of memristors to transistors is low, and the transistors in this particular case are more suited for digital applications as compared to the memristors.

Where applicable, the work done as part of this project has already been compared to existing work in the relevant sections of the report. For example, the performance of the proposed hybrid CMOS Full adder is compared with alternatives in section 6.2.3.6, and the proposed Implication logic full adder sequences is compared with alternate proposals in section 6.3.2.

In summary, although the main objective was not even attempted, the various fallbacks were all achieved and interesting observations and results were obtained.

9 Conclusions and Further Work

This project investigated the ability for TiO_2 memristors to implement digital logic based on three popular logic families. Although practical results were not obtained, the simulation results led to interesting observations and allowed a detailed qualitative comparison to be made between the three families. The most useful part of this project is the fact that a model with an explicit voltage threshold which is fitted to experimental memristor data was used for simulations. The use of the voltage threshold revealed some unpredicted and (possibly unknown) limitations for all three logic families. The simulations in conjunction with the background research allowed sufficient insight of operation in order to develop a list of properties of a preferred device type most suitable for each family. Additionally, during the design and simulation process, some optimised circuits (sections 6.2.3.6, and 6.3.2) were developed with marginally better performance compared to the next best alternatives.

In terms of the TiO_2 device, the MRL family seems most promising in terms of functional correctness. This is likely true for most devices since this family has the least strict requirements for correct operation. However, the TiO_2 devices (at least the ones used by the research group) appear to be too slow for any useful digital applications. This is not an issue for the future of memristor based logic since there are a wide range of much faster devices available.

There remains a vast amount of further work which could have improved this project. Firstly, the purpose of using the VTEAM model fitted to TiO_2 device to simulate all families was purely in anticipation of having these devices for practical experimentation. The simulations were intended to give insight into which family would be most likely to succeed with these devices in order to aid the practical implementation. However, performing a comparison of these families via simulation with this fitted model is not truly fair and not quantifiable. This is because each family has its own preferred memristor type. Using the TiO_2 fitted model, the MAGIC and Implication families were unsuccessful. This meant quantitative comparison of power, speed and area of an example circuit such as an adder could not be completed. Given more time it would be useful to find experimental data for an optimal device for each family, and use this for fitting and then simulation based quantitative and qualitative comparison.

Furthermore, given TiO_2 devices and more time, it would be desirable to implement and investigate the adder circuit presented in section 6.2.3.6 and compare with simulations. This would allow the fitted model to be assed in terms of accuracy in a more realistic setting compared to just a *IV* curve. It would also give additional insight into the family's operation.

Bibliography

[1] Lorin M. Hitt Erik Brynjolfsson, "Computing Productivity: Firm Level Evidence," MIT, 2002.

[2] S. Parthasarathy S. E. Thompson, "Moore's Law: the Future of Si Microelectronics," *Materials Today*, vol. 9, no. 6, pp. 20-25, June 2006.

[3] Leon O. Chua, "Memristor - The Missing Circuit Element," *IEEE TRANSACTIONS ON CIRCUIT THEORY*, vol. 18, no. 5, pp. 507 - 519, 1971.

[4] G. S. Snider, D. R. Stewart and R. S. Williams D. B. Strukov, "The Missing Memristor Found," *Nature*, vol. 453, pp. 80-83, May 2008.

[5] Matthew D. Pickett, Xuema Li, Douglas A. A. Ohlberg, Duncan R. Stewart, and Williams R. Stanley J. Joshua Yang, "Memristive switching mechanism for metal/oxide/metal nanodevices," *Nature Nanotechnology*, vol. 3, no. 7, pp. 429-433, July 2008.

[6] Google. (2016) Google Trends. [Online]. https://www.google.co.uk/trends/explore#q=memristor

[7] Sung Mo Kang, Rainer Waser Pinaki Mazumde, "Memristors: Devices, Models, and Applications," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1911-1919, 2012.

[8] R. Stanley Williams. (2010) Finding the Missing Memristor - R. Stanley Williams. Presentation.

[9] Regina Dittmann, Georgi Staikov, Kristof Szot Rainer Waser, "Redox-Based Resistive Switching Memories – Nanoionic Mechanisms, Prospects, and Challenges," *Advanced Materials*, vol. 21, pp. 2632–2663, 2009.

[10] Larry Dignan, IBM Research builds functional 7nm processor, 2015.

[11] D. R. Stewart , J. Borghetti, X. Li, M. Pickett, G. Medeiros Ribeiro, W. Robinett, G. Snider, J.P. Strachan, W. Wu, Q. Xia, J. Joshua Yang, R.S.Williams D. B. Strukov, "Hybrid CMOS/Memristor Circuits ," *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 1967-1970, 2010.

[12] ITRS, International Technology Roadmap for Semiconductors, http://www.itrs2.net/itrs-reports.html.

[13] Gregory S. Snider, Philip J. Kuekes, J. Joshua Yang, Duncan R. Stewart & R. Stanley Williams Julien Borghetti, "'Memristive' switches enable 'stateful' logic operations via material implication," *Nature*, vol. 464, pp. 873-876, April 2010.

[14] M. Weides, M. N. Kozicki, and R. Waser C. Schindler, "Low current resistive switching in Cu-SiO2 cells," *Applied Physics Letters*, vol. 92, no. 12, 2008.

[15] G. Staikov, and R. Waser C. Schindler, "Electrode kinetics of Cu-SiO2-based resistive switching cells: Overcoming the voltage-time dilemma of electrochemical metallization memories," *Applied Physics Letters*, 2009.

[16] John Paul Strachan, Qiangfei Xia, Douglas A. A. Ohlberg Philip J. Kuekes, Ronald D. Kelley, William F. Stickle, Duncan R. Stewart Gilberto Medeiros-Ribeiro, and Williams R. Stanley J. Joshua Yang, "Diffusion of adhesion layer metals controls nanoscale memristive switching," *Advanced Materials*, vol. 22, no. 36, pp. 4034-4038,

2010.

[17] John William Strachan, Gilberto Medeiros-Ribeiro, Stan Williams Antonio C Torrezan, "Sub-nanosecond switching of a tantalum oxide memristor," *Nanotechnology*, December 2011.

[18] John Paul Strachan, Gilberto Medeiros-Ribeiro, and R Stanley Williams Antonio C Torrezan, "Sub-nanosecond switching of a tantalum oxide memristor," *Nanotechnology*, vol. 22, no. 48, 2011.

[19] Sung Hyun Jo and Wei Lu switching memory, "CMOS compatible nanoscale nonvolatile resistance ," *Nano Letters*, vol. 8, no. 2, pp. 392-397, 2008.

[20] Kuk-Hwan Kim, Ting Chang, S. Gaba, and Wei Lu. Si Sung Hyun Jo, "memristive devices applied to memory and neuromorphic circuits," *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 13-26, June 2010.

[21] Garng M. Huang, Peng Li Yenpo Ho, "Nonvolatile Memristor Memory: Device Characteristics and Design Implications ," *Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009*, pp. 485-490, November 2009.

[22] Kuk-Hwan Kim, and Wei Lu Sung Hyun Jo, "High-density crossbar arrays based on a Si memristive system," *Nano Letters*, vol. 9, no. 2, pp. 870-874, 2009.

[23] Sang-Jin Lee, Kamran Eshraghian Kyoungrok Cho, "Memristor - CMOS logic and digital computational components," *MicroelectronicsJournal*, vol. 46, pp. 214-220, January 2015.

[24] Nimrod Wald, Guy Satat, Eby G. Friedman, Avinoam Kolodny, and Uri C. Weiser Shahar Kvatinsky, "MRL - Memristor Ratioed Logic for Hybrid CMOS-Memristor Circuits," *IEEE TRANSACTIONS ON NANOTECHNOLOGY*, vol. XXX, no. XXX, p. XXX, 2013.

[25] Dmitry Belousov, Slavik Liman, Guy Satat, Nimrod Wald, Eby G. Friedman, Avinoam Kolodny, and Uri C. Weiser Shahar Kvatinsky, "MAGIC—Memristor-Aided Logic," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS*, vol. 61, no. 11, pp. 895-899, November 2014.

[26] Nimrod Wald, Guy Satat, Avinoam Kolodny, and Uri C. Weiser Shahar Kvatinsky, "MRL – Memristor Ratioed Logic," *Proc. Int. Cellular*, pp. 1-6, 2012.

[27] Avinoam Kolodny, Uri C. Weiser Shahar Kvatinsky, "Memristor-based IMPLY Logic Design Procedure," *IEEE 29th International Conference on Computer Design*, 2011.

[28] Misbah Ramadan, Eby G. Friedman , Avinoam Kolodny Shahar Kvatinsky, "VTEAM – A General Model for Voltage Controlled Memristors," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS*, vol. 62, no. 8, pp. 786-790, August 2015.

[29] Amirali Amirsoleimani, Jafar Shamsi, Arash Ahmadi, Shahpour Alirezaee, Majid Ahmadi Mehri Teimoory, "Optimized Implementation of Memristor-Based Full Adder by Material Implication Logic," *International Conference on Electronics Circuits and Systems (ICECS)*, January 2015.

[30] Leon O. Chua, *Introduction to Nonlinear Network Theory*.: McGraw-Hill, 1969.

[31] Panayiotis S. Georgiou, "A Mathematical Framework for the Analysis and Modelling of Memristor Nanodevices," Imperial College London, PhD Thesis 2013.

[32] Leon O. Chua and Sung Mo Kang, "Memristive devices and systems," *Proceedings of the IEEE*, vol. 64, no. 2, pp. 209-223, Feb 1976.

[33] Christofer Toumazou, and Leon O. Chua Themistoklis Prodromakis, "Two centuries of memristors," *Nature Materials*, vol. 11, no. 6, pp. 478-481, June 2012.

[34] Siu K. Ho, "Memristor as amplifier," Imperial College London, MSc Thesis 2015.

[35] HP Labs, The Missing Memristor Found- FAQ, 2009, http://www.hpl.hp.com/news/2008/apr-jun/memristor_faq.html.

[36] R. S. Williams, "How we found the missing Memristor," *IEEE Spectrum*, vol. 45, no. 12, pp. 28-35, 2008.

[37] H. Shima H. Akinaga, "Resistive Random Access memory (ReRAM) Based on Metal Oxides," *Proceedings of the IEEE* , vol. 98, no. 12, pp. 2237–2251, 2010.

[38] Warren Robinett, Michael W. Cumbie, Neel Banerjee, Thomas J. Cardinali, J. Joshua Yang, Wei Wu, Xuema Li, William M. Tong, Dmitri B. Strukov, Gregory S. Snider, Gilberto Medeiros-Ribeiro, R. Stanley Williams Qiangfei Xia, "Memristor CMOS Hybrid Integrated Circuits for Reconfigurable Logic," *Nano Letters*, vol. 9, no. 10, pp. 3640-3645, 2009.

[39] D. S. Jeong, S. K. Kim, C. Rohde, S. Choi, J. H. Oh, H. J. Kim, C. S. Hwang, K. Szot, R. Waser, B. Reichenberg, S. Tiedke B. J. Choi, "Resistive switching mechanism of TiO2 thin films grown by atomic layer deposition," *Journal of Applied Physics*, 2005.

[40] Wei Lu Sung Hyun Jo, "CMOS Compatible Nanoscale Nonvolatile Resistance Switching Memory," *Nano Letters*, vol. 8, no. 2, pp. 392-397, 2008.

[41] Kyle James Miller, "Fabrication and modeling of thin-film anodic titania memristors," Iowa State University, Graduate Theses 2010.

[42] ResearchGate. [Online]. https://www.researchgate.net/post/Is_Titanium_Dioxide_an_n-type_or_a_p-type_semiconductor

[43] Guy Satat, Nimrod Wald, Eby G. Friedman, Avinoam Kolodny, and Uri C. Weiser, Shahar Kvatinsky, "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, October 2014.

[44] J.H. Poikonen and M. Laiho E. Lehtonen, "Two memristors suffice to compute all Boolean functions," *Electronics Letters*, vol. 46, no. 3, February 2010.

[45] R. H. Wilkinson, "A Method of Generating Functions of Several Variables Using Analog Diode Logic," *IEEE Transactions on Electronic Computers*, vol. EC-12, no. 2, pp. 112 - 129, April 1963.

[46] G. G. Langdon Jr, "Logic Design - A Review of Theory and Practice," *Academic Press*, 1974.

[47] Z. Li, J. Strasnicky, X. Li, D. A. A. Ohlberg, W. Wu, D.R. Stewart, and R. S. Williams J. Borghetti, "A Hybrid Nanomemristor/Transistor Logic Circuit Capable of Self-Programming," *Proceedings of the National Academy of Sciences*, vol. 106, no. 6, pp. 1699-1703, February 2009.

[48] Tejinder Singh, "Hybrid Memristor-CMOS (MeMOS) based Logic Gates and Adder Circuits," *Emerging Technologies in Computer Science* , june 2015.

[49] Nimrod Wald, Shahar Kvatinsky Guy Satat, "Logic Design with Memristors," Israel Institute of Technology, 2011-2012.

[50] Dalibor Biolek, and Viera Biolkova Zdenek Biolek, "SPICE model of memristor with nonlinear dopant drift," *Radioengineering*, vol. 18, no. 2, pp. 210-214, June 2009.

[51] Kyungmin Kim, and Sung-Mo Kang Sangho Shin, "Compact models for memristors based on charge-flux constitutive relationships," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 4, pp. 590-598, April 2010.

[52] A. Gavrilov, S. Cohen, D. Mistele, B. Meyler, J. Salzman, D. Ritter E. Yalon, "Resistive Switching in Probed by a Metal–Insulator–Semiconductor Bipolar Transistor," *Electron Device Letters*, vol. 33, no. 1, pp. 11-13, January 2012.

[53] Vincent Garcia, Ryan O. Cherifi, Karim Bouzehouane, Stéphane Fusil, Xavier Moya, Stéphane Xavier, Hiroyuki Yamada, Cyrile Deranlot, Neil D. Mathur, Manuel Bibes, Agnès Barthélémy, Julie Grollier André Chanthbouala, "A Ferroelectric Memristor," *Nature Materials*, vol. 11, no. 10, pp. 860-864, July 2012.

[54] A. Sundararajan, D. P. Hunley, D. R. Strachan S. L. Johnson, "Memristive Switching of Single-Component Metallic Nanowires," *Nanotechnology*, vol. 21, no. 12, March 2010.

[55] A. Iqabal, Y. S. Kimi, K. Eshraghian, and S. F. Al-Sarawi, O. Kavehei, "The Fourth Element: Characteristics, Modeling and Electromagnetic Theory of the Memristor," *Proceedings of the Royal Society*, vol. 486, no. 2120, pp. 2175–2202, 2010.

[56] Julien L. Borghetti, and Williams R. Stanley Dmitri B. Strukov, "Coupled ionic and electronic transport model of thin-," *Small*, vol. 5, no. 9, pp. 1058-1063, May 2009.

[57] Dmitri B. Strukov and Williams R. Stanley, "Exponential ionic drift: fast switching and low volatility of thin-film memristors," *Applied Physics A: Materials Science & Processing*, vol. 94, no. 3, pp. 515-519, March 2009.

[58] Sung-Hyun Jo, Kuk-Hwan Kim, Patrick Sheridan, Siddharth Gaba, Wei Lu Ting Chang, "Synaptic behaviors and modeling of a metal oxide memristive device," *Applied Physics A: Materials Science & Processing*, vol. 102, pp. 857-863, 2011.

[59] Y. N. Joglekar and S. J. Wolf, "The elusive memristor: Properties of basic electrical circuits," *Eur. J. Phys*, vol. 30, no. 4, pp. 661–675, July 2009.

[60] B. P. Peh, C. Papavassiliou, and C. Toumazou T. Prodromakis, "A versatile memristor model with non-linear dopant kinetics," *IEEE Trans. Electron Devices*, vol. 58, no. 9, pp. 3099–3105, Sept 2011.

[61] M.D. Pickett,X. Li,D.A. A. Ohlberg, D. R. Stewart, R.S Williams J. J.Yang, "Memristive switching mechanism

for metal/oxide/metal nanodevices," *Nature Nanotechnol*, vol. 3, pp. 429-433, 2008 July.

[62] Robinson E. Pino, Peter J. Rozwood, and Bryant T. Wysocki Nathan R. McDonald, "Analysis of Dynamic Linear and Non-linear Memristor Device Models for Emerging Neuromorphic Computing Hardware Design," *Air Force Research Laboratory*, January 2010.

[63] Mika Laiho Eero Lehtonen, "CNN Using Memristors for Neighborhood Connections ," *International Workshop on Cellular Nanoscale Networks and their Applications*, 2010.

[64] Eby G. Friedman, Avinoam Kolodny, Uri C. Weiser Shahar Kvatinsky, "TEAM: ThrEshold Adaptive Memristor Model," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS*, vol. 60, no. 1, pp. 211 - 223, January 2013.

[65] Dmitri B. Strukov, Julien L. Borghetti, J. Joshua Yang, Gregory S. Snider, Duncan R. Stewart, R. Stanley Williams Matthew D. Pickett, "Switching dynamics in titanium dioxide memristive devices," *Applied Physics*, vol. 106, no. 7, pp. 1-6, October 2009.

[66] J. L. Borghetti, and R. S. Williams D. B. Strukov, "Coupled ionic and electronic transport model of thin-film semiconductor memristive behavior," *Small*, vol. 5, no. 9, pp. 1058-1063, May 2009.

[67] John G. Simmons, "Generalized Formula for the Electric Tunnel Effect between Similar Electrodes Separated by a Thin Insulating Film," *Journal of Applied Physics* , vol. 34, no. 6, pp. 1793–1803, January 1963.

[68] Matthew D. Pickett Hisham Abdalla, "SPICE Modeling of Memristors," *IEEE International Symposium on Circuits and Systems*, pp. 1832-1835, 2011.

[69] Dalibor Biolek, Viera Biolkova Zdenek Kolka, "Improved Model of TiO2 Memristor," *Radioengineering*, vol. 24, no. 2, pp. 378-383, June 2015.

[70] Haifeng Cheng, Xuan Zhu, Guang Wang, Nannan Wang Dongqing Liu, "Analog Memristors Based on Thickening/Thinning of Ag Nanofilaments in Amorphous Manganite Thin Films," *Applied Materials and Interfaces*, vol. 5, no. 21, pp. 11258-11264, October 2013.

[71] Vincent Garcia, Ryan O. Cherifi, Karim Bouzehouane, Stéphane Fusil, Xavier Moya, Stéphane Xavier, Hiroyuki Yamada, Cyrile Deranlot, Neil D. Mathur, Manuel Bibes, Agnès Barthélémy, Julie Grollier André Chanthbouala, "A ferroelectric memristor," *NATURE MATERIALS*, vol. 11, no. 10, pp. 860–864, July 2012.

[72] Kris Campbell. (2015) Knowm org. [Online]. http://knowm.org/memristors/

[73] Jussi Poikonen, Mika Laiho, Wei lu Eero Lehtonen, "Time-Dependency of the Threshold Voltage in Memristive Devices," *IEEE International Symposium of Circuits and Systems (ISCAS)*, pp. 2245-2248, 2011.

[74] Zdenek Biolek, Viera Biolkova Dalibor Biolek, "SPICE Modeling of Memristive, Memcapacitative and Meminductive Systems," *European Conference on Circuit Theory and Design Conference Program*, pp. 249-252, 2009.

[75] Shahar Kvatinsky. Memristor Models. [Online]. http://webee.technion.ac.il/people/skva/Memristor%20Models.htm

[76] Ronald Tetzlaff, Fernando Corinto, Marco Gilli Alon Ascoli, "PSpice switch-based versatile memristor model,"

IEEE International Symposium on Circuits and Systems Conference, pp. 205 - 208, May 2013.

[77] Jan Snyman, *Practical Mathematical Optimization; An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*.: Springer, 2005.

[78] SIMetrix Technologies Ltd. (2003) Technical Note: How Spice Works. [Online]. http://www.simetrix.co.uk/site/support/kb/How%20Spice%20Works.pdf

[79] Paul D. Mitcheson, Semiconductor Modelling in SPICE: SPICE algorithms and internals , http://www3.imperial.ac.uk/pls/portallive/docs/1/7292571.PDF.

[80] Dmitri B. Strukov, Duncan R. Stewart J. Joshua Yang, "Memristive devices for computing," *Nature Nanotechnology*, vol. 8, January 2013.

[81] J. P. Strachan, G. Medeiros-Ribeiro, and R. S. Williams A. C. Torrezan, "Sub-Nanosecond Switching of a Tantalum Oxide Memristor," *Nanotechnology*, vol. 22, no. 48, pp. 1-7, November 2011.

[82] Tony R. Kuphaldt, *Lessons in Electric Circuits, Vol.IV- Digital, Chapter 3 - Logic Gates*. [Online]. http://www.allaboutcircuits.com/textbook/digital/chpt-3/logic-signal-voltage-levels/

[83] Achyut Timilsina, David Moore, and Kristy A. Campbell Antonio S. Oblea, "Silver Chalcogenide Based Memristor Devices," *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-3, 2010.

[84] David Harris, Introduction to CMOS VLSI Design, 2004, Harvey Mudd College.

[85] Dr.R.Anita , M.K.Anandkumar R.P.Meenaakshi Sundari, "Implementation of Low Power CMOS Full Adders Using Pass Transistor Logic," *IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)*, pp. 38-43, June 2013.

[86] Gregory S. Snider, Philip J. Kuekes, J. Joshua Yang, Duncan R. Stewart, R. Stanley Williams Julien Borghetti, "'Memristive' switches enable 'stateful' logic - suplimentary information," *nature*, vol. 464, no. 8, pp. 873-876, April 2010.

[87] E. Lehtonen and M. Laiho, "Stateful Implication Logic with memristors," *Proceedings of the IEEE/ACM International Symposium on Nanoscale Architectures*, pp. 33-36, July 2009.

[88] Shahar Kvatinsky, Eitan Yaakobi Yuval Cassutor, "Sneak-Path Constraints in Memristor Crossbar Arrays," *IEEE International Symposium on Information Theory*, pp. 156-160, 2013.

[89] S. E. Thompson and S. Parthasarathy, "Moore's Law: the Future of Si Microelectronics," *Materials Today*, vol. 9, no. 6, June 2006.

[90] David Schor. ZigWap. [Online]. http://www.zigwap.com/digital/gates

Appendices

Appendix A

Almost Ideal Diode Model:

```
.model Dbreak D Is=1e-14 Cjo=.1pF Rs=.0001 n=0.0001
```

VTEAM PSpice Macro-model:

```
* PSpice Model Editor - Version 16.6.0
.SUBCKT VTEAM p m
+PARAMS: initial_x = 0.5
+x_off = 1
+x_on = 0
+v_on = -0.8
+v_off = 0.8
+k_on = -0.036363636
+k_off = 0.036363636
+alpha_on = 0.1
+alpha_off = 0.1
+r_off = 50000
+r_on = 500
* Manually calculate lamda = ln(r_off/r_on) *
+lamda = 4.60517
+omega = 25e-3
+a_on = 0.05
+a_off = 0.95
****** Memristance dependance on state variable ********
*Linear*
.func mem_rel(x)={r_on+((r_off-r_on)*(x-x_on))/(x_off-x_on)}
*Exponential*
*.func mem_rel(x)={r_on*exp((lamda*(x-x_on))/(x_off-x_on))}
********* Functions **********
.func expL(x)= {if((x<=10), exp(x), exp(10))}
.func gfon(x)= {if( (v_on > x),(MIN((1*k_on*(PWR(((x/v_on)-1), alpha_on))), 0.99e10)),
0)}
.func  gfoff(x)=  {if((v_off  <  x),(MIN((1*k_off*(PWR(((x/v_off)-1),  alpha_off))),
0.99e10)), 0)}
.func winon(x)={ expL(-expL(-((x-a_on)/omega ))) }
.func winoff(x)={ expL(-expL((x-a_off)/omega )) }
.func poly_iv(x)={(0.25*sgn(x)*PWR(x, 3) + x)*0.76}
********* State Equation ********
Cstate state 0 330mF IC={initial_x}
Gon 0 state VALUE={(gfon(V(p, m)))*winon(V(state,0))}
Goff 0 state VALUE={(gfoff(V(p, m)))*winoff(V(state,0))}
Vxon p1 0 {x_on}
Vxoff p2 0 {x_off}
Dxonlim p1 state Dbreak
Dxofflim state p2 Dbreak
****** Memristor ******
Rshunt p m 10000000
Gmemrist p m VALUE={(poly_iv(V(p, m)))/mem_rel(V(state,0))}
.ENDS
*
```

TEAM PSpice Macro-model:

```
* PSpice Model Editor - Version 16.6.0
*$
.SUBCKT TEAM plus minus
+PARAMS:
+x_off = 1
+x_on = 0
+i_on = -50e-6
+i_off = 50e-6
+k_on = -0.1
+k_off = 0.1
+alpha_on = 2
+alpha_off = 2
+r_off = 10000
+r_on = 100
+lamda = 4.60517
+omega = 100e-3
+a_on = 0.2
+a_off = 0.8
****** Memristance dependence on state variable ********
*Linear*
*.func mem_rel(x)={r_on+((r_off-r_on)*(x-x_on))/(x_off-x_on)}
*Exponential*
.func mem_rel(x)={r_on*exp((lamda*(x-x_on))/(x_off-x_on))}
********* Functions **********
.func expL(x)= {if((x<=10), exp(x), exp(10))}
.func  gfon(x)=  {if(  (i_on  >  x),  (MIN(((k_on)*  (PWR(((  x/i_on)-1),  alpha_on  ))),
0.99e10)), 0)}
.func   gfoff(x)=   {if((i_off   <   x),(MIN(((k_off)*(PWR(((x/i_off)-1),   alpha_off))),
0.99e10)), 0)}
.func winon(x)={ expL(-expL( -(x-a_on)/omega )) }
.func winoff(x)={ expL(-expL(  (x-a_off)/omega )) }
******** State Equation ********
Cstate state 0 3nF IC={0.5}
Gon 0 state VALUE={(gfon(I(Visense)))*winon(V(state,0))}
Goff 0 state VALUE={(gfoff(I(Visense)))*winoff(V(state,0))}
Vxoff p2 0 {x_off}
Vxon p1 0 {x_on}
Dxonlim p1 state Dbreak
Dxofflim state p2 Dbreak
******** Memristor ********
Visense n0 minus {0}
Rseries n0 n1 5
Ememrist plus n1 VALUE={I(Visense)*mem_rel(V(state,0))}
.ENDS
*
*$
```

Mosfet parameters used for Hybrid CMOS gates:

```
.model Mbreakp PMOS VTO=-1.5 KP=0.02m W=6.3u L=0.25u
.model Mbreakn NMOS VTO=1.5  KP=0.05m W=2.5u L=0.25u
```

Mosfet parameters for Hybrid CMOS Full Adder:

```
.model Mbreakp PMOS VTO=-1.5 KP=0.02m W=6.3u L=0.25u
.model Mbreakn NMOS VTO=1.25  KP=0.05m W=2.5u L=0.25u
```

Appendix B

Function to fit a polynomial to a set of data points using only the specified exponents:

```
function p=polyfitcoef(x,y,powers)
% Finds the optimal coefficients for the 'powers' specified. The rest are zero.
%
% Inputs:
%   powers: A vector of integers containing all the desired powers of
%   the polynomial in, e.g. [5,3,1] will fit the polynomial y = ax^5+ bx^3+cx
%   x: x axis input
%   y: y axis input
%
% Outputs:
%   p: The resulting vector of optimal polynomial coefficients. The ones not
%   included in 'powers' are zero.
%
    y=y(:);                          % ensures column vector
    x=x(:);
    powers=powers(:);
    A=bsxfun(@power,x,powers');      % Matrix of powers of x
    [Q,R]=qr(A);                     % QR decomposition
    coeffs = R\(Q'*y);               % Solve linear equations
    p=zeros(1,max(powers)+1);
    p(powers+1)=coeffs;
    p=p(end:-1:1);                   % so result is in descending order of powers

end
```

Script which generates the desired voltage sequence for the PSpice simulations:

```
% Generates the text file which controls the 'V_PWL_FILE' source in PSpice. Duplicate
% this script with different filenames for as many files/sources as necessary

filename = 'v1.txt';
rt=10e-9;                                % rise and fall times (seconds)
vol=[0 5 0 5 0 5 0 5];                   % voltage amplitude sequence
steps=[20, 20, 20, 20, 20, 20, 20, 20];  % Duration of each voltage pulse (seconds)
num_values = length(vol);

if (length(steps)~=length(vol))
   error('Number of voltage pulses specified must equal number of pulse-widths
specified.')
end

time=[];
volts=[];
step_cum=[0, cumsum(steps)];

% Generate Vectors
for i=1:num_values
    time(2*i-1) = step_cum(i);
    time(2*i) = step_cum(i+1) - rt;

    volts(2*i-1) = vol(i);
    volts(2*i) = vol(i);
end

time=time(:);
volts=volts(:);
%plot(time, volts)
```

```
% Write to file
fileID = fopen(filename,'w');
for i=1:length(time)
    fprintf(fileID,'%.13e \t %.13e\n' ,time(i), volts(i));
end
fclose(fileID);
```

Appendix C

Parameter:	*Value*:
v_{on}	$-0.8\ v$
v_{off}	$0.8\ v$
R_{on}	$500\ \Omega$
R_{off}	$50\ K\Omega$
α_{on}	0.1
α_{off}	0.1
k_{on}	$-0.0363636\ s^{-1}$
k_{off}	$0.0363636\ s^{-1}$
x_{on}	1
x_{off}	0
C_x	$0.33\ F$
a_{on}	0.05
a_{off}	0.95
ω	25×10^{-3}
Initial state	0.75
Memristance dependence on state variable	*Linear*
Static IV polynomial relation	$I = -0.05168v^5 - 0.0114\ v^4 + 0.494v^3 + 0.76v$

Table 25: The fitting parameters used with the VTEAM model (Appendix A) to generate Figure 48. The state variable represents a normalised distance and hence it has no units of dimensions.