Afonso Raúl Monteiro Silva

Licenciado em Engenharia Eletrotécnica e de Computadores

Memristor Based Logic Circuits

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e de Computadores

Orientadora: Maria Helena Fino, Professora Auxiliar, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Júri

Presidente: Doutor João Almeida das Rosas
Vogais: Doutora Maria Helena Silva Fino
Doutor Pedro Miguel Ribeiro Pereira

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2019

Memristor based logic circuits

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

To all of the people that made this dissertation possible.

Acknowledgements

I would like to thank my adviser, Maria Helena Fino, for the support and advices given for the completion of this dissertation.

I would also like to thank the elelectrical engineering department, FCT-UNL and all of its teachers and workers for everything they taught me in the last 6 years.

Lastly I would like to thank all my friends and family for all the support given throughout all my academic learning.

Abstract

The increased study of electronics has lead to breakthroughs that have allowed technology such has CMOS to create smaller and faster devices. However, this scaling has slowed down due to problems such as power dissipation. Technologies such as memory devices are unable to keep up with market demand for smaller and lower powered devices.

A lot of different solutions have been proposed to solve this problem, one of which has been the use of memristors. The advantages of this technology are, no-volatility, good scalability, and compatibility with CMOS devices. Memristors can be used to improve existing technologies such as memory, logic and neuromorphic devices.

This dissertation will focus first on the study of various memristor models, then the implementation of the various models studied will be addressed, then a single model will be chosen that better fits the requirements necessary to develop logic gates. Then, using the chosen model, more complex circuits will be implemented, first circuits composed of two memristors, then logic gates circuits.

Keywords: Memristor, Logic gates.

Resumo

O estudo de eletrónica levou a avanços que permitiram que tecnologia como CMOS criar dispositivos menores e mais rápidos. No entanto, à medida que os dispositivos vão ficando mais pequenos novos desafios vão aparecendo, devido a problemas como por exemplo dissipação de energia. Tecnologias como dispositivos de memória são incapazes de acompanhar a busca do mercado por dispositivos menores e mais efecientes.

Muitas soluções diferentes foram propostas para resolver este problema, uma das quais tem sido o uso de memristores. As vantagens desta tecnologia residem no fato de serem não volateis, apresentarem boa escalabilidade e compatibilidade com dispositivos CMOS. Os Memristors podem ser usados para melhorar as tecnologias existentes em aplicações como memória, lógica e dispositivos neuromórficos.

Esta dissertação irá focar-se primeiro, no estudo de vários modelos de memristors, depois irá ser feita a imlementação dos vários modelos estudados, depois irá ser escolhido um modelo que melhor cumpre os requerimentos necessários para desenvolver porta lógicas. Depois, usando o modelo escolhido, circuitos mais complexos irão ser implementados, primeiro circuitos compostos por dois memristors, e por fim irão ser implementadas as portas lógicas.

Palavras-chave: Memristor, Portas lógicas.

Contents

List of Figures

List of Tables

Listings

Acronyms

HP	Hewlett Packard.
MAGIC	Memristor Aided LoGIC.
TEAM	ThrEshold Adaptive Memristor.
VTEAM	Voltage ThrEshold Adaptive Memristor.

Chapter 1

Introduction

The exponential growth of electronic technology has successfully created smaller and faster devices, this growth has been spear headed by the improvement of semiconductor technology such as CMOS. However, already this technology has started to hit a wall, as the technology gets smaller and smaller, new problems and challenges appear that make the continuing development of smaller technology difficult. These problems include, current leakage, high power densities and high cost of testing and manufacturing.

As such new technologies that can replace or better yet, be incorporated with CMOS technologies are desired. This dissertation will focus on the study of the memristor and its use to create logic gates. The memristor acts as a potentiometer, i.e., a resistor with variable resistance. One of the main advantages of the memristor is its compatibility with CMOS technology, when incorporated with it can be, for example, to reduce the number of transistors in a circuit.

To study the behaviour of the memristor, various mathematical models of the memristor will be studied and implemented, then, a single model will be chosen that better fits the requirements needed to build logic gates. Then, more complex circuits composed of two memristors will be implemented to better understand its behaviour. Finally, logic gates composed of memristors will be implemented and tested.

1.1 Brief introduction to Memristors

In electronics by establishing a relation between two of the four basic circuit variable i.e, the current (i), voltage (v), charge(q) and flux-linkage (φ), it can be concluded that, from a total of six possible relations, five relations are known. These relationships betwee charge and current, as well as between flux and voltage are given by

$$q(t) = \int_{-\infty}^{t} i(t)dt \qquad (1.1) \qquad\qquad \varphi(t) = \int_{-\infty}^{t} v(t)dt \qquad (1.2)$$

Three other relationships are used to characterize the three basic electronic circuits, the resistor (R), the capacitor (C) and the inductor (L), as represented by the following expressions

$$R = \frac{dv}{di} \qquad (1.3) \qquad\qquad C = \frac{dv}{dq} \qquad (1.4) \qquad\qquad L = \frac{d\varphi}{di}. \qquad (1.5)$$

In 1971 Leon Chua postulated about the existence of a fourth basic circuit element, this element would be defined by the only relationship that remains undefined, i.e, the relationship between flux-linkage and charge. Leon Chua named this element the memory resistor, or memristor (M), and its formula is [2]

$$M = \frac{d\varphi}{dq} \qquad (1.6)$$

This element would be similar to an electrical resistance but, unlike the resistor, its resistance value is not constant; its resistance value is defined by the electrical charge that has flowed through the device in the past and in which direction, this means the device will have memory in the form of its resistance value. When charge stops flowing through the memristor i.e., its power source has been turned off, the memristor will remember its last value. This makes it non-volatile as it doesn't need to be connect to a power source to keep its memory value saved.

1.2 Memristor Applications

1.2.1 Non-volatile memory

Due to its property of being able to store a value without needing a power source to save it, non-volatile memory will most likely be the main application of the memristor, reportedly HP labs have already developed memory with about one tenth of the speed of DRAM.

1.2.2 Neuromorphic circuits

The neural network in the human mind has the ability to create strong or weak connections, which is described by the neuro-plasticity. Nonlinear dependencies caused in the function of a neuron is dependent on the action potentials that it receives. The only physical device that may be close enough to mimic these nonlinearities is the memristor[7].

1.2.3 Programmable logic and signal processing

It is proven that memristors can perform logic operations. Different methods such as IMPLY[12] or MAGIC[10] logic has been developed to use memristors to build logic gates.

1.3 Structure of the document

This document also has 4 more chapters in addition to the Introduction.

State of the Art (Chapter 2) In this chapter the various models of memristor are presented and studied.

Memristor Testing (Chapter 3) In this chapter, the macromodels of the memristor models studied previously are presented. Then, to test the various models, the SPICE and VerilogA codes presented are run in LTSpice and Cadence.

Testing more complex circuits and logic gates (Chapter 4) In this chapter, using the VTEAM model, firstly more complex circuits will be tested using memristors, then logic gates will be tested using the same code.

Conclusions (Chapter 5) Finally, the last chapter presents a general conclusion of this dissertation and addresses possible future work.

1.4 Summary

As seen in the previous section, a memristor is a device that resembles a classic resistance, the main difference being that, its resistance, which is referred as memristance varies when an electrical charge passes through it. Its final value for its memristance being its last memristance value when the electrical charge is no longer present. This means the device has memory, its current memristance value, and that the device doesn't need a power source to keep its memory value stored. Applications of the memristor were also presented; non-volatile memory, neuromorphic circuits and logic and signal processing are some of the main applications being developed. Lastly, the structure of the document was presented.

Chapter 2

State of the Art

2.1 Introduction

In this chapter, different types of memristors models that are used to characterize the behavior of the device are presented. First the linear models will be studied, then the window functions, which are used to add non-linear behavior and restrict the state variable values, are presented. Then the non-linear models will be presented, first the Simmons model, then the TEAM and Vourkas models which aim to simplify the Simmons model. Lastly, the VTEAM model, which is a voltage-controlled version of the TEAM model will be presented.

2.2 Mathematical model

In this section the mathematical models describing the behavior of memristors will be presented. Models for both current controlled and voltage controlled devices will be considered.

2.2.1 HP model

In 2008, Stanley Williams and its team at the HP labs managed to successfully create the first implementation of the memristor [3]. This memristor was created by sandwiching a thin layer of titanium-dioxide (TiO_2). This thin layer would then be divided in two layers, one layer would be doped with oxygen vacancies, called the TiO_{2-x} layer, the other layer is not doped and has insulating properties, the TiO_2 layer. When a positive voltage is applied to the memristor, an electrical field is formed and forces the oxygen vacancies in the doped layer to drift to the undoped layer, causing the conductivity of the memristor to increase i.e., the resistance of the memristor will decrease. When a negative voltage

is applied, the oxygen vacancies will move back to the doped layer thus decreasing the conductivity of the device i.e., its resistance value will increase. When voltage is no longer applied to the memory resistor, the memristor will no longer have an electric field, thus the oxygen vacancies cannot move freely between TiO_2 layers, meaning its resistance cannot be altered, this is what gives the memristor its non-volatile properties [1].

In [3] a mathematical model is proposed to describe how the memristor works, this model assumes a uniform field across the device.

The resistance value of the memristor R_{mem} can be calculated by the following formula

$$R_{mem} = R_{on}x + R_{off}(1 - x) \tag{2.1}$$

where

$$x = \frac{w}{D} \tag{2.2}$$

Where R_{on} is the memristor's minimum resistance value, i.e., when the maximum number of oxygen vacancies have shifted form the doped to the undoped layer of TiO_2. R_{off} is the maximum resistance value, i.e., when the maximum number of oxygen vacancies have shifted to the doped layer of TiO_2 and x is the state variable of the device and given by the ratio between the width of the doped layer of TiO_2 (w) and the total width of TiO_2 (D) as shown in Figure 2.1 [1].

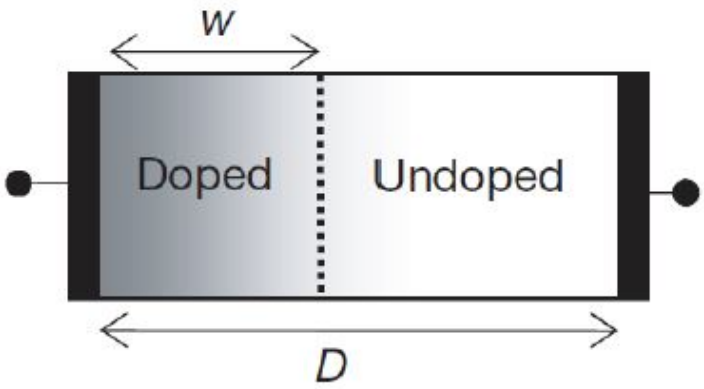


Figure 2.1: HP memristor model.

By applying Ohm's law, we obtain the relation between voltage and current in the device

$$v(t) = (R_{on}x + R_{off}(1 - x))i(t) \tag{2.3}$$

The application of a voltage $v(t)$ across the device will move the boundary between the two regions by causing the charged dopants to drift. For the simplest case of ohmic electronic conduction and linear ionic drift in a uniform field with average ion mobility μv, we obtain

$$\frac{dx(t)}{dt} = \mu v \frac{R_{on}}{D^2} i(t) \tag{2.4}$$

and

$$x(t) = \mu v \frac{R_{on}}{D} q(t) \tag{2.5}$$

By inserting equation 2.5 into equation 2.3, we get the memristance of this system which for $R_{on} << R_{off}$ simplifies to

$$M(q) = R_{off}(1 - \frac{\mu v Ron}{D^2} q(t)) \tag{2.6}$$

In the formula 2.7 and 2.8 we can see how the speed of the boundary between varies according the resistance of the doped area, current, etc,

$$\frac{dx}{dt} = ki(t)f(x) \tag{2.7}$$

$$k = \frac{\mu v R_{on}}{D^2} \tag{2.8}$$

2.2.1.1 Window Functions

As it was seen previously, the HP memristor assumes a uniform field across the device. This assumption leads to 2.3 and 2.6 where it can be concluded that, the device will have linear behavior, which means, as the width of the doped region of the memristor is increased linearly, so does the conductivity of the device.

At nanometer dimensions, by just applying a few volts, a strong electric field is formed in the device, which causes a high non-linearity in the ionic drift-diffusion. The previous formulas do not take in consideration this effect. To solve this problem, various attempts have been carried out to add nonlinear behavior to the state equation. To overcome this limitation the first proposal considered multiplying the second term of the state equation with a "window function", $f(x)$, as it can be seen in 2.9

$$\frac{dx(t)}{dt} = \mu v \frac{R_{on}}{D^2} i(t) f(x) \tag{2.9}$$

In the literature, several window functions, $f(x)$, have been proposed. Strokov proposed the following function [3]

$$f(x) = x - x^2 \tag{2.10}$$

This function lacks in flexibility, and, if x gets near to any boundary(e.g, 0 or 1), it can no longer be adjusted, this problem is known as the terminal state problem. Another window function was proposed by Joglekar and Wolf [17]

$$f(x) = 1 - (2x - 1)^{2p} \tag{2.11}$$

This window ensures zero drift at the boundaries, i.e., $f(0)$ and $f(1)$ is zero. Also nonlinear drift is imposed over the entire active region D, and for high values of p, the model resembles linear dopant drift. However, this window function does not solve the terminal state problem.

Biolek proposed a window function that solves the terminal state problem by adding the current (i) as a parameter. The memristor is brought out of the terminal state when the current reverses direction [18].

$$f(x) = 1 - (x - stp(-i))^{2p} \tag{2.12}$$

$$stp(i) = \begin{cases} 1, & \text{if } i \geqslant 0 \\ 0, & \text{if } i < 0 \end{cases} \tag{2.13}$$

Prodomakis proposed a window function, as it can be seen in formula 2.14, that allows it to scale upwards, which implies that $f_{max}(x)$ can take any value between 0 and 1. In addition, p can take any positive real number unlike the constrains of the control parameter being an integer in the Joglekar and Biolek models allowing more flexibility [15].

$$f(x) = 1 - [(x - 0.5)^2 + 0.75]^p \tag{2.14}$$

2.2.2 Simmons Tunnel Barrier Model

In 2008 D.B Strukov and M.D Pickett proposed a new solution for the modeling of memristors, this model assumes switching behavior due to an exponential dependence of the movement of the ionized dopants [14]. The schematic of the device is represented in Figure 2.2 [16].

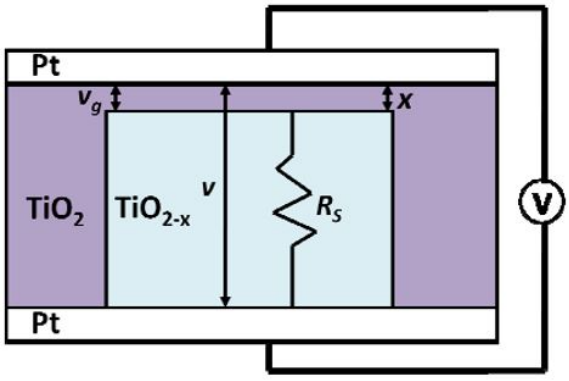


Figure 2.2: Simmons tunnel barrier memristor model.

As it can be seen in Figure 2.2, rather than having two resistors in series as in the HP model there is a resistor in series with an electron tunnel barrier. In this model the state variable x is the tunnel barrier width, and, the derivative of x can be interpreted as the oxygen vacancy drift velocity, and it is given by [16].

$$\frac{dx(t)}{dt} = \begin{cases} C_{off} sinh(\frac{i}{i_{off}}) exp(-exp(\frac{x-a_{off}}{w_c} - \frac{i}{b}) - \frac{x}{w_c}), & \text{if } i < 0 \\ C_{on} sinh(\frac{i}{i_{on}}) exp(-exp(\frac{x-a_{on}}{w_c} - \frac{i}{b}) - \frac{x}{w_c}), & \text{if } i > 0 \end{cases} \tag{2.15}$$

Where C_{off}, C_{on},i_{off},i_{on}, a_{off}, a_{on}, w_c and b are fitting parameters. In practical memristors, when a negative voltage is applied, the drift of the oxygen vacancies and the diffusion of the oxygen vacancies from the TiO_{2-x} layer to the TiO_2 layer are in the same direction, and, when a positive voltage is applied the direction of drift and diffusion are opposite. C_{off} and C_{on} influence the magnitude of the change in x. C_{on} is an order of magnitude larger than C_{off}. i_{off} and C_{on} constrain the current threshold. Bellow this currents there is no change in the derivative of x, the oxygen vacancy drift velocity. Because of the exponential dependence on $x - a_{off}$ or $x - a_{on}$, the derivative of the state variable is significantly smaller for the state variable within the permitted range, therefore there is no need for window functions [16].

In this model, the relationship between current and voltage is shown in the following formula

$$\begin{aligned} i(t) = \tilde{A}(x, v_g)\phi_1(v_g, x)exp(-B(v_g, x)\phi_1(v_g, x)^{\frac{1}{2}}) \\ - \tilde{A}(x, v_g)(\phi_1(v_g, x) + e|v_g|)exp(-B(v_g, x)(\phi_1(v_g, x) + ev_g)^{\frac{1}{2}}) \end{aligned} \quad (2.16)$$

$$v_g = v - i(t)R_s \quad (2.17)$$

Where v is the internal voltage applied across the device, this voltage is different than the external voltage applied on the device.

2.2.3 ThrEshold Adaptive Memristor (TEAM) Model

As it can be seen in the previous subsection, the Simmons memristor model is a very complex model with no explicit relationship between voltage and current.This causes the computational model to be inefficient. Also, this model fits only a specific type of memristor. Therefore a simpler model which can represent the same behavior is desired. In [16], a simpler threshold memristor is described. In order to simplify the model, it is assumed that there is no change in the state variable bellow a certain threshold, and a polynomial dependence rather than an exponential dependence is used. It was concluded, by analyzing the oxygen vacancy drift velocity when the memristor current is varied, it can be modeled as a device with threshold currents, i_{off} and i_{on}. This approximation is justified, since for small changes in the electric tunnel width, a separation of variable can be performed. The dependence of the internal state derivative on current and the state variable itself can be modeled as independently multiplying two independent functions, one which depends on the state variable x, the other is a function of the current [16]. Under this assumption, the derivative of the state variable is given by [16]

$$\frac{dx(t)}{dt} = \begin{cases} k_{off}(\frac{i(t)}{I_{off}} - 1)^{\alpha_{off}}.f_{off}(x), & \text{if } 0 < i_{off} < \text{i} \\ k_{on}(\frac{i(t)}{I_{on}} - 1)^{\alpha_{on}}.f_{on}(x), & \text{if } i < i_{on} < 0 \\ 0, & \text{otherwise} \end{cases} \quad (2.18)$$

Where k_{off}, k_{on},α_{off} and α_{on} are constants, i_{off} and i_{on} are the current thresholds, x is the state variable which represents the effective electric tunnel width, k_{off} is a positive number, k_{on} is a negative number $f_{off}(x)$ and $f_{on}(x)$ behave as window functions studied previously.

If we assume that the relation between current and voltage is like the one in the HP model, then, for this model, the relation between voltage and current is as shown bellow

$$v(t) = \left(R_{on} + \frac{R_{off} - R_{on}}{x_{off} - x_{on}}(x - x_{on})\right) i(t) \tag{2.19}$$

However, the resistance is exponentially dependent with the state variable since the memristance, in practical memristors, is dependent on a tunneling effect, which is highly nonlinear. Therefore, any changes in the tunnel barrier width changes the memristance in a exponential manner. Under these assumptions, $v(t)$ becomes [16].

$$v(t) = R_{on} e^{\frac{\lambda}{x_{off} - x_{on}}} i(t) \tag{2.20}$$

Where λ is a fitting parameter, R_{on} and R_{off} are the equivalent resistance at the bounds, and must satisfy

$$\frac{R_{off}}{R_{on}} = e^{\lambda} \tag{2.21}$$

In order to fit this model to the Simmons model, λ, k_{off}, k_{on},α_{off} and α_{on} , can be evaluated to achieve a sufficient accurate match between both models [16].

As stated previously, $f_{on}(x)$ and $f_{off}(x)$ are window functions that fit the Simmons model. As reported on [16], a sufficient approximation is

$$\begin{cases} f_{on}(x) \approx exp\left(-exp\left(\frac{x-\alpha_{on}}{w_c} - \frac{i}{b}\right) - \frac{x}{w_c}\right) \approx exp\left(-exp\left(\frac{x-\alpha_{on}}{w_c}\right)\right) \\ f_{off}(x) \approx exp\left(-exp\left(\frac{x-\alpha_{off}}{w_c} - \frac{i}{b}\right) - \frac{x}{w_c}\right) \approx exp\left(-exp\left(\frac{x-\alpha_{off}}{w_c}\right)\right) \end{cases} \tag{2.22}$$

Besides being a model that simplifies the Simmons model, another advantage of the team model is that it can also be used to characterize other memristor models. For, example, it can also be fitted to the HP model [16], where

$$k_{on} = k_{off} = \mu_v \frac{R_{on}}{D} i_{on} \tag{2.23}$$

$$\alpha_{on} = \alpha_{off} = 1 \tag{2.24}$$

$$i_{on} = i_{off} \to 0 \tag{2.25}$$

$$x_{on} = D \tag{2.26}$$

$$x_{off} = D \tag{2.27}$$

$$x = D - w \tag{2.28}$$

2.2.4 Voltage ThrEshold Adaptive Memristor (VTEAM) Model

In [8], a model is proposed that is based on the TEAM model presented in the previous section, that describes a current controlled memristor, which relies on the existence of

two current thresholds. However studies have been made that conclude the existence of voltage thresholds instead of current thresholds. Furthermore, voltage thresholds are necessary for certain memory and logical applications [8].

The VTEAM model describes the derivative of the state variable as a function of the state variable x and a voltage v as can be seen in (2.29). The relationship between voltage and current across the device is the voltage multiplied by its conductance G, as it can be seen in (2.30).

$$\frac{dx(t)}{dt} = f(x, v) \tag{2.29}$$

$$i(t) = G(x, v) * v(t) \tag{2.30}$$

The derivative of the sate variable in the VTEAM model is defined as

$$\frac{dx(t)}{dt} = \begin{cases} k_{off}(\frac{v(t)}{v_{off}} - 1)^{\alpha_{off}}.f_{off}(x), & \text{if } 0 < v_{off} < \text{v} \\ k_{on}(\frac{v(t)}{v_{on}} - 1)^{\alpha_{on}}.f_{on}(x), & \text{if } v < v_{on} < 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.31}$$

Where k_{off}, k_{on},α_{off} and α_{on} are constants, v_{off} and v_{on} are the voltage thresholds, x is the state variable, k_{off} is a positive number, k_{on} is a negative number $f_{off}(x)$ and $f_{on}(x)$ behave as window functions.

The current-voltage relationship and $G(x, v)$ are not specifically defined in the VTEAM model. The relationship used is the same as the one in the TEAM model,

$$i(t) = \left(R_{on} + \frac{R_{off} - R_{on}}{x_{off} - x_{on}}(x - x_{on})\right)^{-1} * v(t) \tag{2.32}$$

where x_{off} and x_{on} are the bounds of the state variable, and R_{off} and R_{on} are the corresponding values for the resistance of the memristor. As in the TEAM model an exponential dependence on the state variable can be assumed, as it can be seen in 2.33.

$$i(t) = \frac{e^{\frac{\lambda}{x_{off} - x_{on}}}}{R_{on}} * v(t) \tag{2.33}$$

where λ is a fitting parameter and

$$e^{\lambda} = \frac{R_{off}}{R_{on}} \tag{2.34}$$

2.2.5 Vourkas Model

In 2012 Ioannis Vourkas and Georgios Ch. Sirakoulis proposed a new solution for the modeling of memristors. This model explains the behavior of the device by investigating the occurrence of quantum tunneling [5], like the Simmons model and the TEAM model.

It is also a threshold type, voltage controlled model and it is described by the following formula

$$I(t) = G(L,t)V_M(t) \tag{2.35}$$

$$\frac{dL}{dt} = f(V_M,t) \tag{2.36}$$

In this model, L is the state variable and it accounts for the width of the undoped layer of TiO_2, G is the conductance of the device and V_M is the applied AC voltage. The derivative of the state variable L is the velocity at which the barrier between the two layers of TiO_2 moves due to the applied voltage in the device. As it can be seen in Figure 2.3 [5], R represents the ohmic resistance of the doped layer and R_t is the resistance of the undoped layer. This resistance R_t will be proportional to the width of the undoped layer. The doped layer will act as a conductor and the undoped layer acts as an insulator, meaning the resistance value of R_t will be higher than the resistance value of R, also since R_t should vary according to the value of L. Any formula of R_t should have a fitting parameter to link the effect of the device's varying geometry on the actual concentration of the oxygen vacancies in either of the sides (doped/undoped) of the TiO_2 film. R_t is also inversely proportional to the product of the voltage-dependent tunneling transmission coefficient, T_0, and the electron effective density of states, N_{eff}. R_t is also exponentially proportional to the tunnel barrier width L. With these assumptions $R_t's$ mathematical formula is given by [9]

$$R_t(V_M) = \frac{1}{N_{eff}} \frac{e^{2k_{V_M}L}}{T_{0,V_M}} \tag{2.37}$$

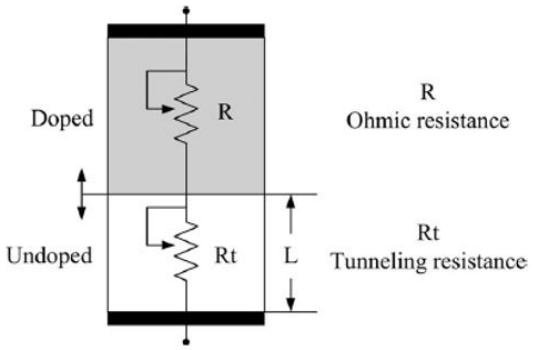


Figure 2.3: Vourkas resistor model.

This function can be simplified by replacing k and T_0 with a new voltage-dependent parameter $L_{V_{M,T}}$, since both k and T_0 are voltage-dependent parameters, which depends on the state variable L, so R_t then becomes

$$R_t(L_{V_{M,t}}) = f_0 \frac{e^{2k_{V_M}L}}{L_{V_{M,t}}} \tag{2.38}$$

Thus, we reach the formula of the device's resistance, which depends on the state variable L. f_0 is a model-fitting constant parameter which represents all the unknown material-specific and geometrical characteristics. It is expected that the state variable L should vary within a valid range. This is based on the assumption that the switching rate of L is between a threshold voltage of $-Vth$ and Vth. With these assumptions a function $L(V_{M,t})$ can be reached that gives the expected tunnel barrier width, L as a function of time and applied voltage

$$L(V_{M,t}) = L_0\left(1 - \frac{m}{r(V_{M,t})}\right) \tag{2.39}$$

L_0 is the maximum value L can attain, m is a fitting constant parameter and $r(V_{M,T})$ incorporates the assumption for the expected different switching rate of L. The time derivative of r is given by the following equation

$$\frac{dr(V_{M,t})}{dt} = \begin{cases} a.\frac{V_M+V_{th}}{c+|V_M+V_{th}|}, & V_M \in [-V_0,-V_{th}[\\ bV_M, & V_M \in [-V_{th},+V_{th}] \\ a.\frac{V_M-V_{th}}{c+|V_M-V_{th}|}, & V_M \in]+V_{th},+V_0] \end{cases} \tag{2.40}$$

The previous formula comprises one parameter sigmoid for the regions above Vth, whereas a linear relation of the applied voltage is used for the region bellow. Vth, a,b and c are fitting constants that define the slope and magnitude of the time derivative of r with $a >> b$ and $0 < c < 1$. Different values of a, b, c and m define a different set of boundaries for L. This model has the potential to describe a more general behavior of the memristance if the state variable is normalized between 0 and 1. This is achieved by dividing $L(V_{M,t})$ with $L0$, and by multiplying with L_0 the exponent and also the denominator of equation 2.37. Then, when $L \approx 0$ the device is in its most conductive state, and when $L \approx 1$ the device is in its least conductive state. This allows the device to, in a more general way, describe the behavior of various types of memristor models.

2.3 Comparing memristor models

In table 2.1, adapted from [9], a table comparing the various memristor models is presented. With this table a memristor model can be chosen that satisfies the necessary requirements for its application.

Table 2.1: Table comparing the memristor models studied previously.

Model	HP	Simmons	TEAM	VTEAM	Vourkas
Control Mechanism	Current	Current	Current	Voltage	Voltage
Current-Voltage relationship	Explicit	Ambiguous	Explicit	Explicit	Ambiguous
Generic	No	No	Yes	Yes	Yes
Threshold exists?	No	Yes	Yes	Yes	Yes
Complexity	Low	High	Low	Low	Moderate
Accuracy	Low	High	Moderate	Moderate	High

2.4 Summary

In this chapter, various types of memristor models were presented. Firstly, the more simple HP model was studied. This model uses window functions to restrict the state variable inside the memristor's dimensions. Window functions are also used to account for the non-linear behavior to the device's memristance. This is done to better emulate the behavior of real transistors.

Then, the Simmons tunnel model was analyzed. This model assumes nonlinear and asymmetric switching behavior. It gives a higher accuracy in modeling real memristor but, it offers no explicit relation between current and voltage and its very complicated model, witch results in it hard to simulate.

The TEAM model was then developed to simplify the Simmons model. It achieves this by assuming two different thresholds for the current that passes through the device. It also assumes a polynomial dependence between the memristor current and the internal state drift derivative. Another advantage is that it can be fitted to different types of memristor models.

The VTEAM model is an adaptation of the TEAM model, but is voltage controlled instead of current-controlled, it was created to take advantage of the simplicity and accuracy of the TEAM model. The main change is the presence of voltage thresholds instead of current, this change is important because voltage thresholds are necessary for certain memory and logical applications.

The Vourkas model is a voltage controlled memristor, which also tries to simplify the Simmons model. It achieves this by attributing the switching effect to an effective tunneling distance modulation. Like the TEAM model, it also can be fitted to different types of memristor models.

Chapter 3

Memristor Testing

In this section, the implementation of the previously presented models in eletrical simulators will be considered. There are two possibilities, using a SPICE code based on the marcomodels of the memristor models or using a hardware description language like VerilogA. First, the macromodels will be presented, then the Spice and Verilog codes will be tested using a single memristor. Then, the different codes will be compared and a single memristor model will be chosen to implement more complex circuits and the logic gates.

3.1 Macromodels

In this section, the macromodels of the previous types of memristors will be described, these macromodels help translating from the mathematical model to eletrical circuits, which then can be converted into a eletrical symbol, using coding languages such as Spice or verilog.

3.1.1 HP Model

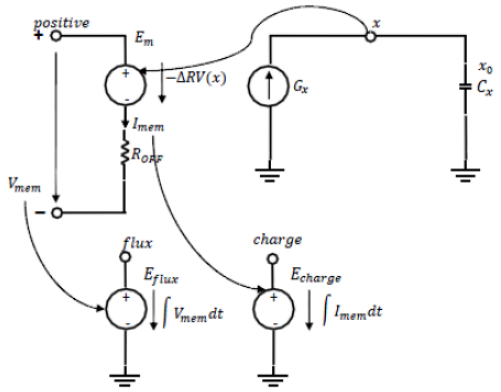


Figure 3.1: SPICE model for the HP memristor.

In Figure 3.1 [1], the SPICE model for the HP memristor can be seen, the code developed using this macro model can be seen in I.1. In it V_{MEM} is the input voltage and I_{MEM} is the current through the memristor. The flux is calculated by integrating the input voltage V_{MEM} and the charge by integrating the current I_{MEM}. E_M is a voltage that is controlled by the formula $-x\delta R$. G_x is a current source whose current is controlled by the equation $kI_{MEM}f(V(x))$ where $V(x)$ is the voltage across C_x [1].

3.1.2 Simmons Model

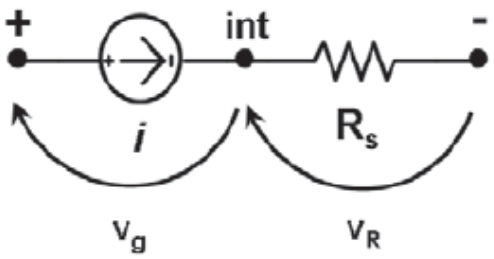


Figure 3.2: SPICE model for the Simmons memristor current equation.

Figure 3.2 [4] shows the Spice model of the Simmons model,. This model was developed in [4], the code developed using this macro model can be seen in I.2. In this model, the tunnel electron tunnel barrier is modeled as a voltage dependent current source, which is connected to a resistance R_s which represents the conducting TiO_{2-X} layer. V_g and V_R are, respectively, the voltage across the current source and the Resistance, with the total voltage across the device being $V = V_g + V_R$. To model the state equations the circuit shown in the figure 3.3 [4], the voltage w represents the width of the tunnel barrier, G_{off} models the state equation when $i < 0$, and G_{on} models the state equation when $i > 0$.

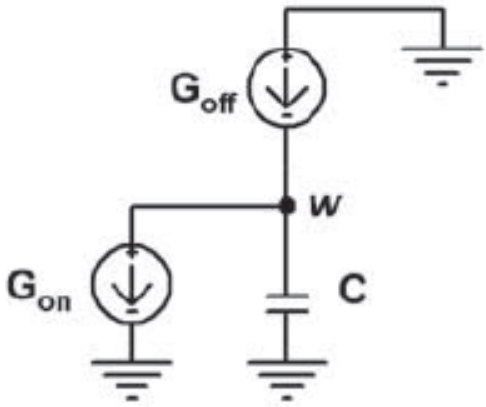


Figure 3.3: SPICE model for the Simmons memristor state equations.

3.1.3 TEAM Model

In [13] a macromodel that describes the TEAM memristor model was proposed, this model can be seen in 3.4 [13]. The code developed using this macro model can be seen in [9], in this model the state variable x is the voltage across the capacitor C, $D1$ and $D2$ constrain the bounds of the state variable to the values of the voltage sources x_{off} and x_{on}. G_{on} and G_{off} are the functions of the time derivative of x. CS is determined from the current-voltage relationship. V_P and V_N are the positive and negative ports of the memristor.

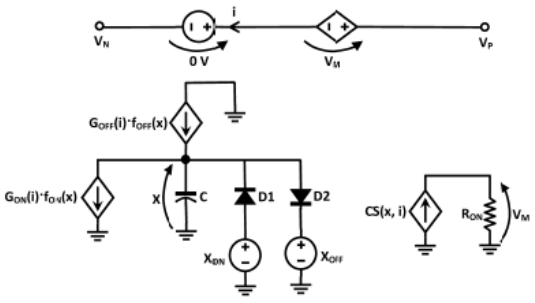


Figure 3.4: Circuit for the the TEAM memristor model.

3.1.4 Vourkas Model

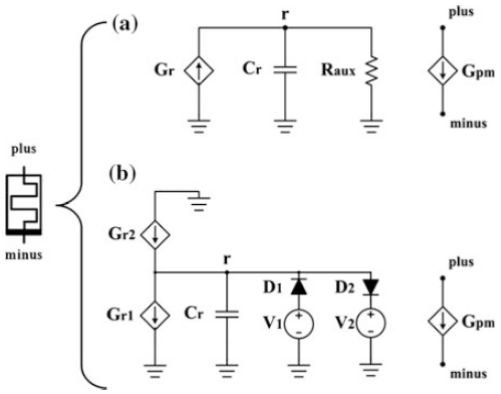


Figure 3.5: SPICE model for the memristor.

In Figure 3.5 [6] two different SPICE models are represented. The code developed using this macromodel can be seen in I.3. In figure (a), the memristor is modeled as a circuit combining two current sources, G_{pm} and G_r, with a capacitor C_r, which models the memory properties of the memristor, and a resistance R_{aux}. G_r generates a current based on 2.40. The voltage across C_x defines the value of $r(V_M,t)$. The output of G_{pm} is set using the voltage drop across the ports of the device and the memristance given by 2.38, R_{aux} is used to model the memory retention capability. In this model, $r(V_M,t)$ can step out of the valid interval, which can lead to unstable solutions, to help with this problem a smoothing function is used, which limits $r(V_M,t)$ inside the valid interval between the defined boundaries r_{max} and r_{min} [6]. In figure (b), a more in-depth way of modeling the memristor is presented as well as offering a better control of the boundary conditions. in this SPICE model G_r is replaced by two current sources G_{r1} and G_{r2}, which have opposite polarities and function in a way that, G_{r2} is in charge of charging the capacitor C_r while G_{r1} is in change on discharging it. In this model, the problem of bounding $r(V_M,t)$ is addressed by using diodes and DC voltage sources. If the voltage across the capacitor C_r, V_r, falls bellow V_1 then the diode D_1 is forward biased and V1 is maintained at V_r, this process is the same for D_2 id the voltage V_r rises above V_2. For this process to work, the voltages V_1 and V_2 have to be set manually. This version does not have an auxiliary resistor, but can be included, like in the SPICE model in (a), to model the memory retention capability of the memristor [6]. In [6] both versions were found to offer similar results.

3.2 SPICE Test Simulations

In this section, various simulations were run to see the behavior of the various types of models for the memristors. The aim is to test if the various models result in valid values for the resulting memristance. The various models implemented have a set of parameters to use in order for the simulations to work properly, so, in order to not have simulation

problems, the chosen parameters implemented are the ones used in the respective paper were the simulations are originally made.

3.2.1 HP Model

To test the HP model in LTSpice, the SPICE code used was given in [18], and it can be seen in I.1. The simulations done were the same as the ones described in [1]. The circuit used to test the behavior of the memristor is shown in Figure 3.6, the input is a sine wave with a voltage of $1.2V$ with $1Hz$ frequency. The values for the memristor parameters μv, D, R_{on}, R_{off} and $R_{initial}$ are, respectively, $10^{-}10cm^2s^{-1}V^{-1}$, $10nm$, 100Ω, $16k\Omega$ and $11k\Omega$. All models use the same window function parameter of $p = 10$.

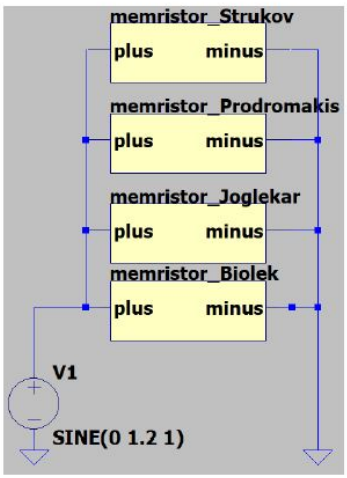


Figure 3.6: memristor testing circuit

In Figure 3.7, it can be seen that the result of the resistance with Strukov windows function. We can see that the resistance will vary between $11k\Omega$ and $12k\Omega$, which means the applied voltage will not cause large variations in the resistance value.

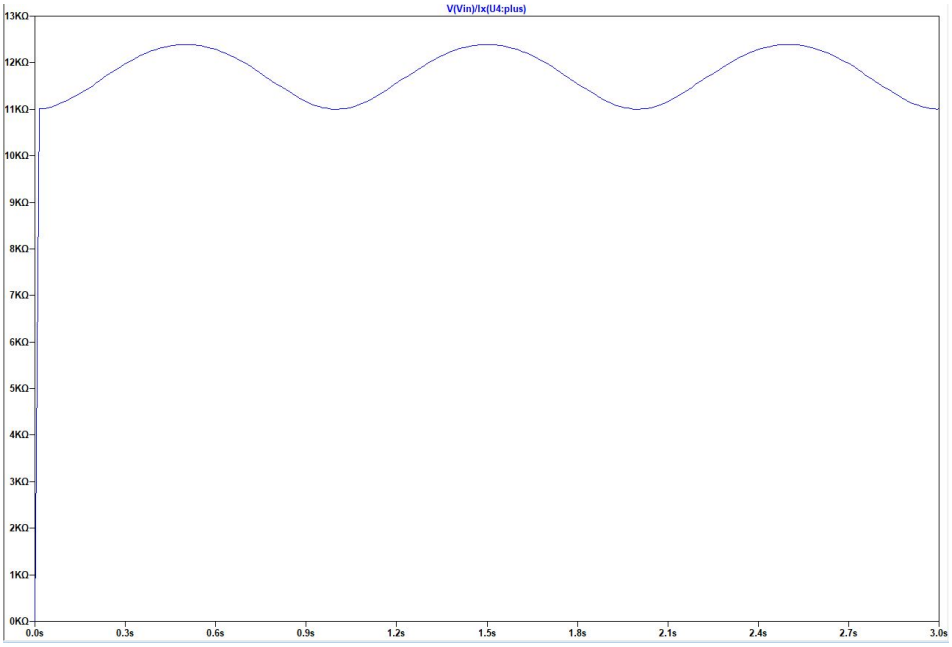


Figure 3.7: Memristor Resistance with strukov's window function

In Figure 3.8, the results of the resistance with Joglekar windows function can be seen. The resistance will vary between $11k\Omega$ and 0Ω.

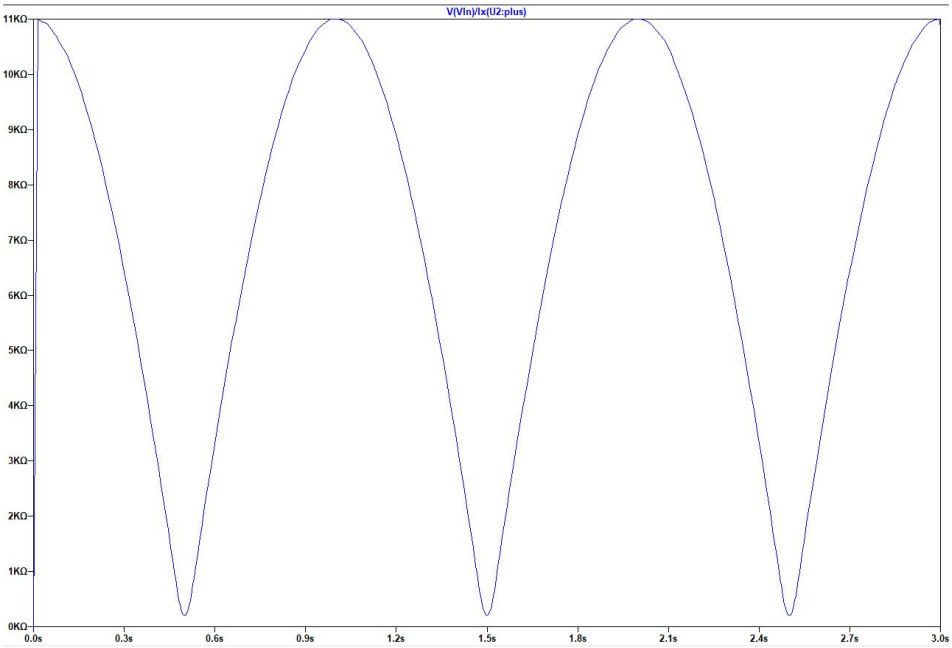


Figure 3.8: Memristor Resistance with Joglekar's window function

In Figure 3.9, the results of the resistance with Biolek windows function can be seen, the resistance will vary between $11k\Omega$ and $1k\Omega$.

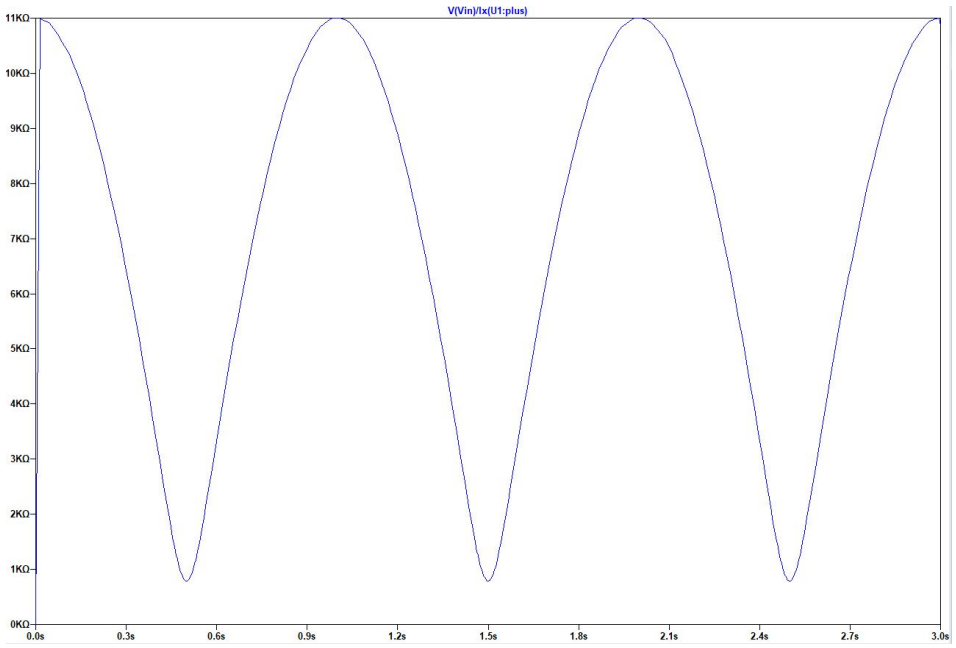


Figure 3.9: Memristor Resistance with Biolek's window function

In Figure 3.10, the results of the resistance with Prodromakis windows function, the resistance will vary between $11k\Omega$ and $3k\Omega$.

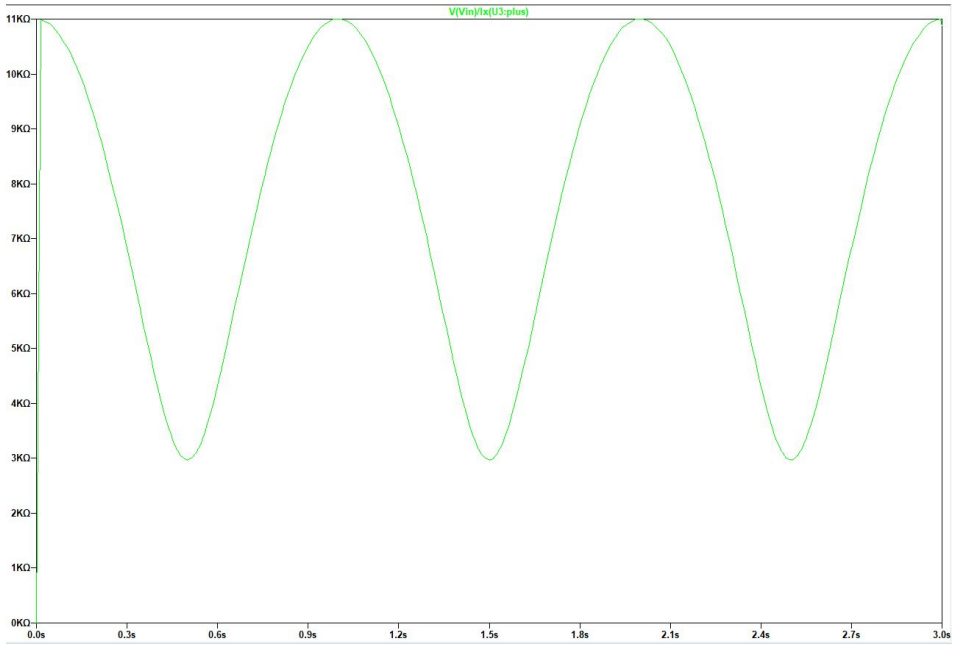


Figure 3.10: Memristor Resistance with Prodromakis's window function

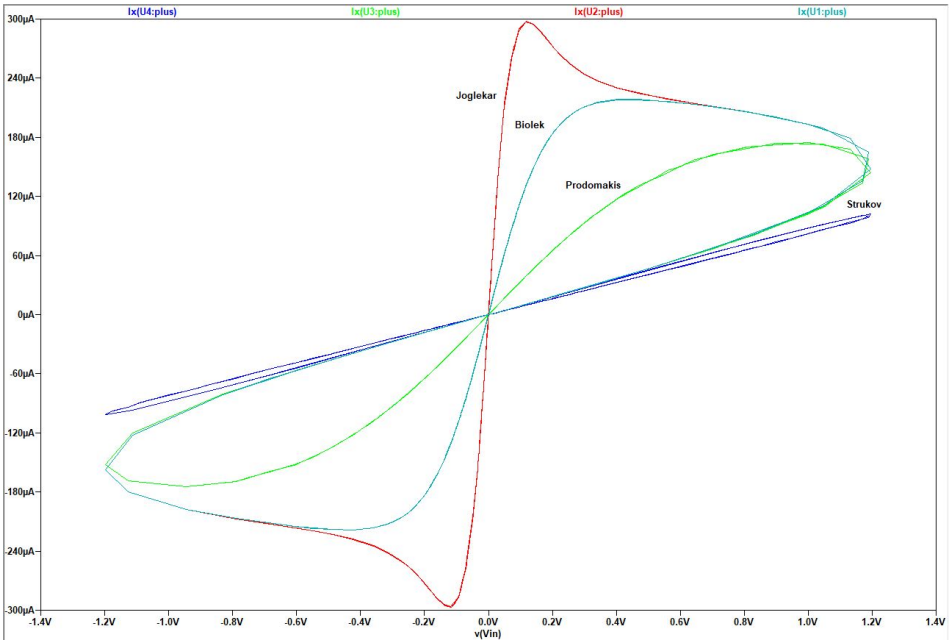


Figure 3.11: Hysteresis loop for all windows functions

In Figure 3.11, it can be seen that the hysteresis loop for all the windows functions, we can conclude the Joglekar model will result in higher currents, we can also conclude, by analyzing de previous figures that Joglerkar model will also give the biggest range in the resistance values for the memristor.

3.2.2 Simmons Model

To test the Simmons model model in LTSpice, the SPICE code used was given in [4]. Iit can be seen in I.2. In the figure 3.12, the circuit used to test the Simmons can be seen, the $2.4k\Omega$ resistor is used to account for the electrodes used in the experiments of the real memristor. The applied voltage was a sine wave with $3V$ amplitude and $100Hz$ frequency.

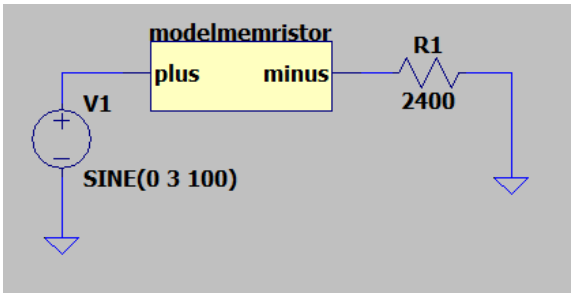


Figure 3.12: Circuit used to test the Simmons model

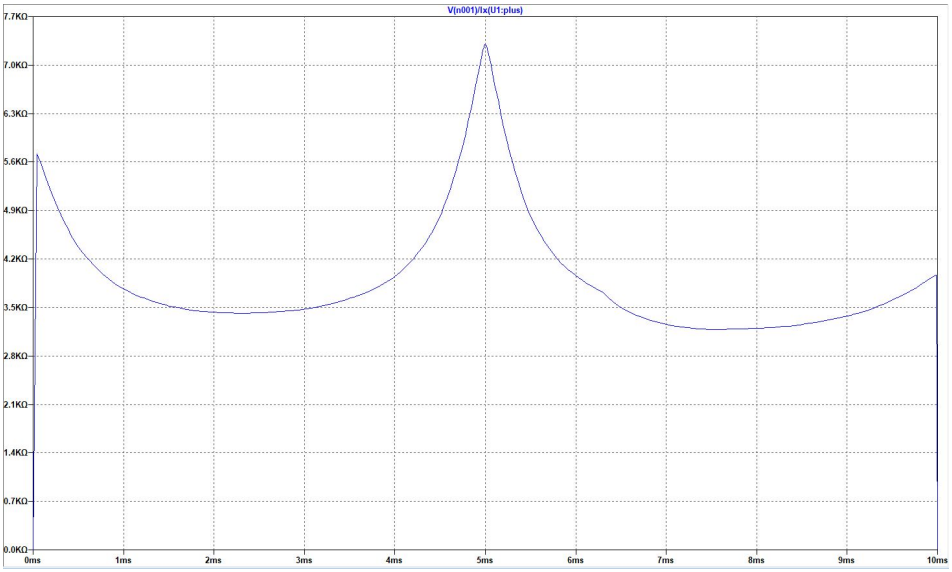


Figure 3.13: Memristance for the Simmons model

As it can be seen in figure 3.13, the resulting memristance varies between $7.2k\Omega$ and $3.1k\Omega$. In figure 3.14, the hysteresis loop can be seen. Although a pinched loop can be seen, it behaves almost in a linear behavior.

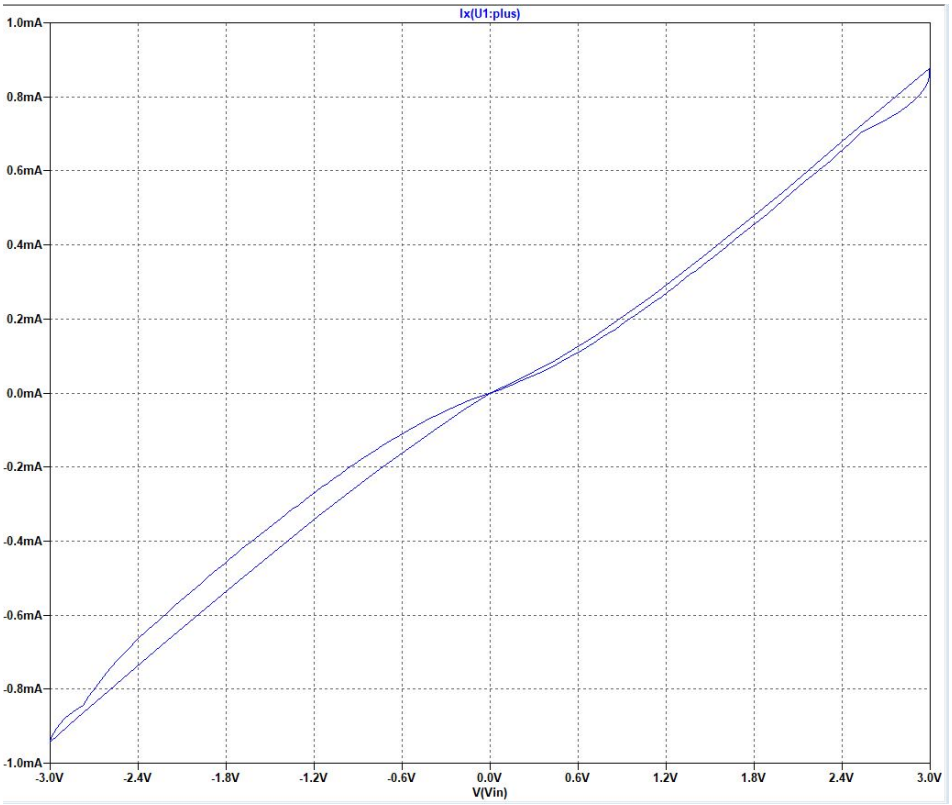


Figure 3.14: Hysteresis loop of the Simmons model

3.2.3 Vourkas Model

The SPICE implementation of model in the figure 3.5 (a) was used to test in LTSPICE. It can be seen in I.3. A 3V and 100Hz sinusoidal voltage wave was applied in the memristor which was restricted between $2k\Omega$ and $200k\Omega$.

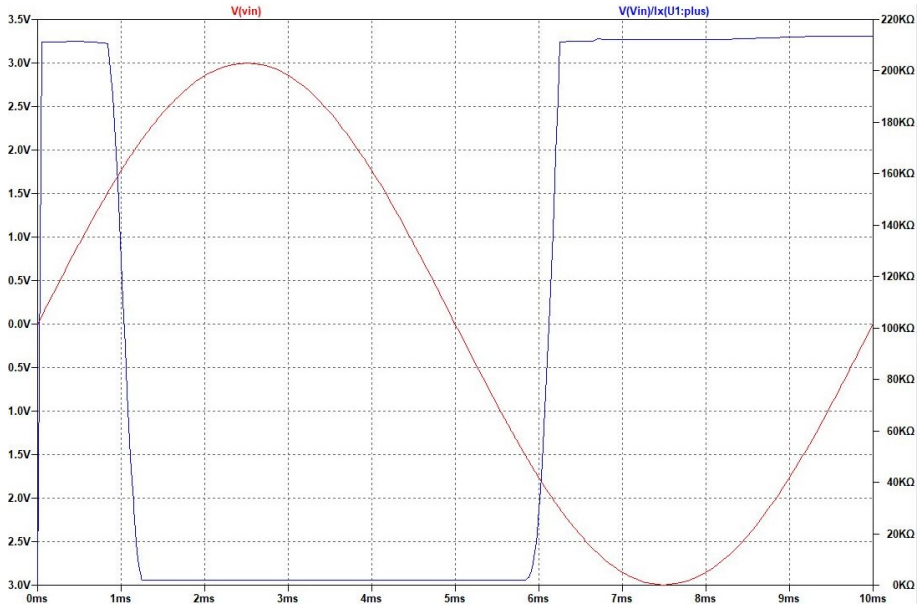


Figure 3.15: Memristor's Memristance

In figure 3.15 the memristor's meristance can be seen. Like it was expected its values are restricted between 2 and $200k\Omega$. It can also be seen the voltage thresholds, when the applied voltage exceeds $1.5V$ it switches to a low memristance value. When the applied voltage exceeds $-1.5V$ it switches to a low memristance value. In figure 3.16, the hysteresis loop can be seen.

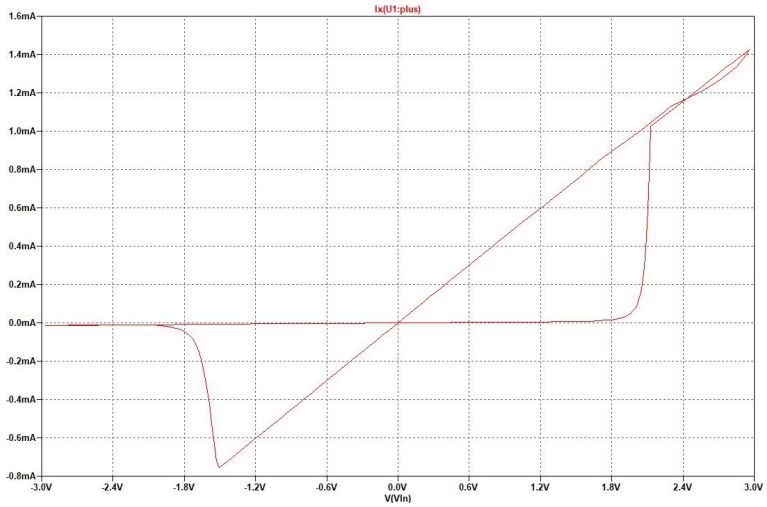


Figure 3.16: Memristor's hysteresis loop

3.3 VTEAM Model

In [10] a Verilog model was developed. This code also can be used to test the HP model, the Simmons model and the TEAM model. The parameters used are shown in table 3.1. The VTEAM model was tested using a sinusoidal wave with $2V$ amplitude and $50MHz$ frequency, with direct and reverse polarity, and a high and low starting memristance, which will be referred from now on as OFF and ON configuration.

Table 3.1: VTEAM model I.4 parameters

Parameter	Value
k_{on}	-216.2 m/sec
K_{off}	0.091 m/sec
$V_{T,ON}$	-1.5 V
$V_{T,OFF}$	0.3 V
x_{on}	0
x_{off}	3 nm
α_{on}	4
α_{off}	4
R_{ON}	1 kΩ
R_{OFF}	300 kΩ

3.3.1 Direct polarity

The circuit used to test a memristor with direct polarity and, OFF and ON configuration can be seen in figure 3.17.

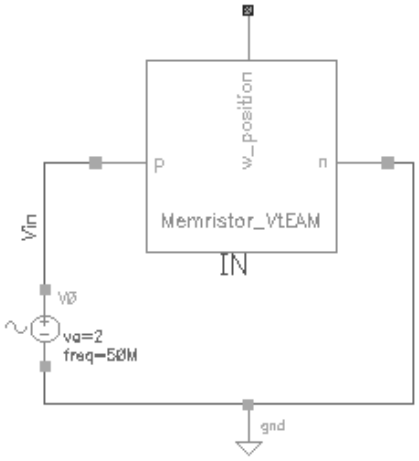
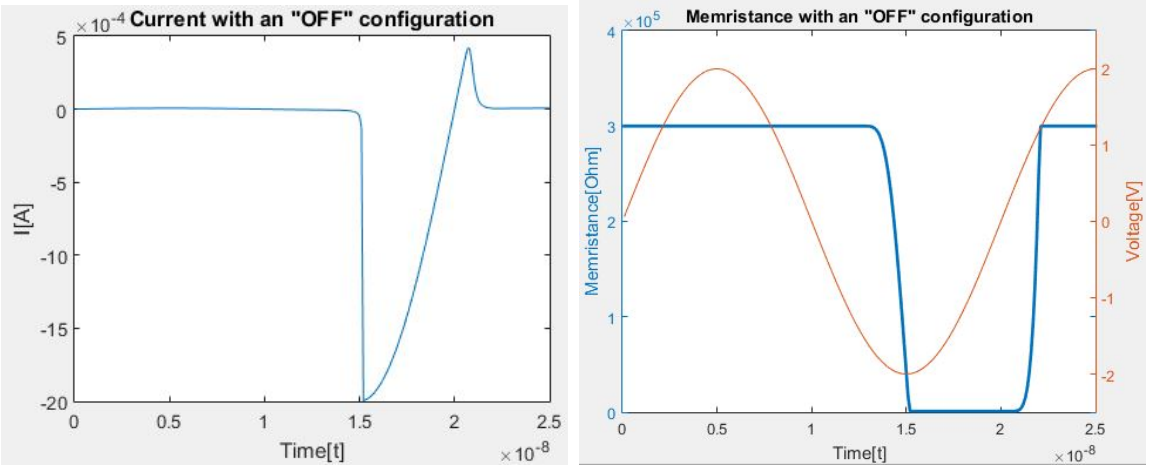


Figure 3.17: Circuit used to test a single memristor with direct polarity

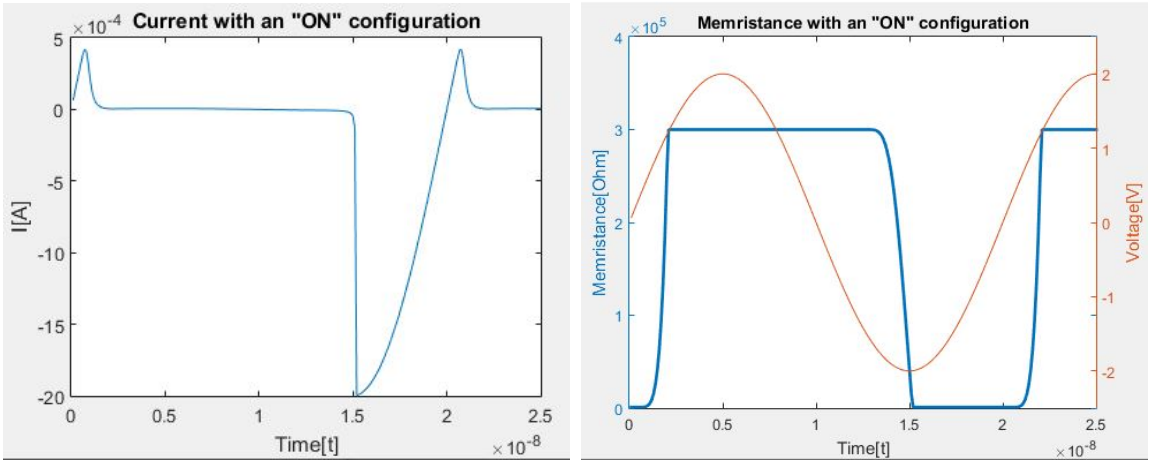
With a direct polarity, a single memristor changes from OFF to an ON configuration when a voltage of $-1.5V$. When a memristor is at an ON configuration, and a voltage of $300mV$ is applied the memristor will return to an OFF configuration. This behaviour

can be seen, with a starting configuration of OFF and ON, respectively , along with the memristors current in the figures 3.18 and 3.19.



(a) Current of the memristor (b) Memristance of the memristor

Figure 3.18: Memristance and current for a memristors in OFF configuration and direct polarity



(a) Current of the memristor (b) Memristance of the memristor

Figure 3.19: Memristance and current for a memristors in ON configuration and direct polarity

3.3.2 Reverse polarity

The circuit used to test a memristor with reverse polarity and, OFF and ON configuration can be seen in figure 3.20.

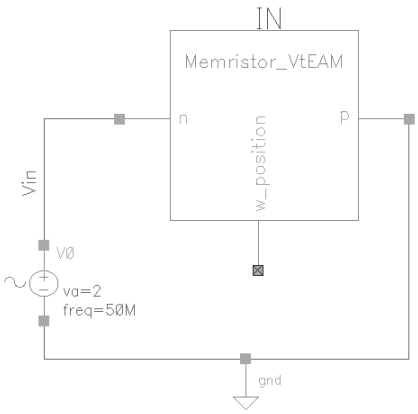
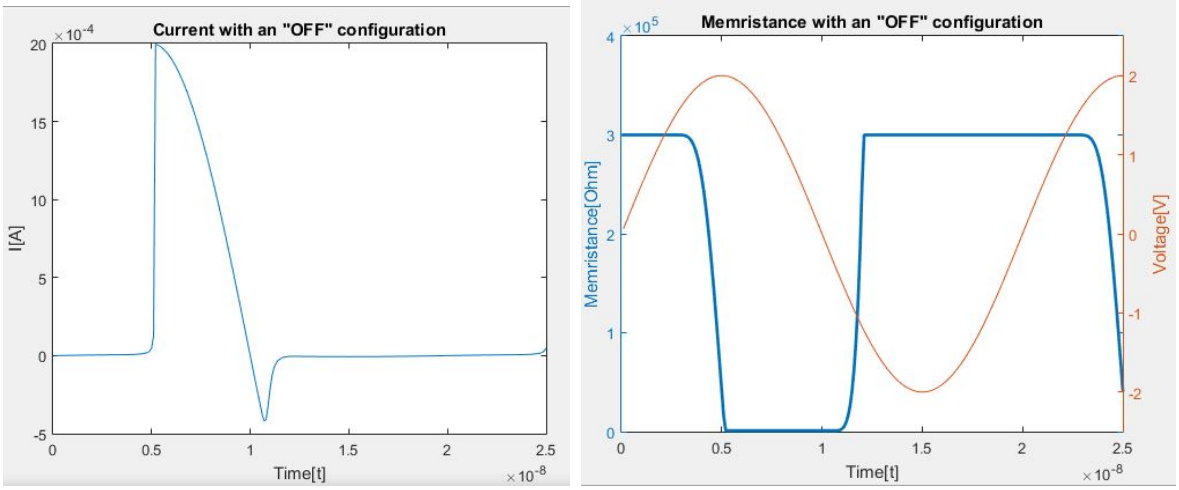


Figure 3.20: Circuit used to test a single memristor with direct polarity

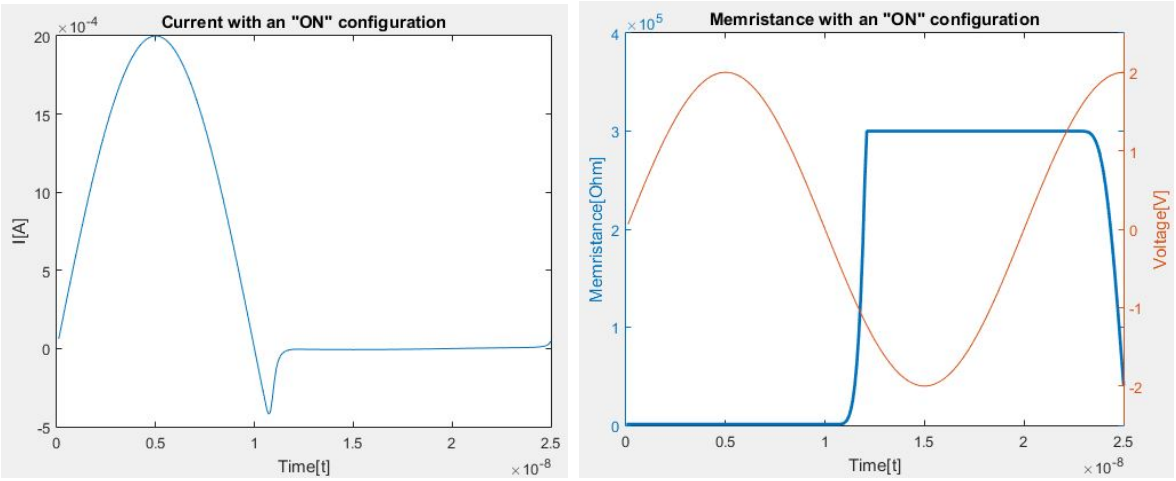
With a reverse polarity, a single memristor, since the voltage source is applied in the negative port as can be seen in the circuit in figure 3.20, the voltage necessary to switch from OFF to an ON configuration becomes $1.5V$ instead of $-1.5V$; likewise to change it from an ON to an OFF configuration becomes $-300mV$, as it can be seen in figure 3.21 and 3.22, along with the memristors current.



(a) Current of the memristor

(b) Memristance of the memristor

Figure 3.21: Memristance and current for a memristors in OFF configuration and direct polarity



(a) Current of the memristor

(b) Memristance of the memristor

Figure 3.22: Memristance and current for a memristors in ON configuration and direct polarity

3.4 Comparing SPICE to Verilog memristor codes

Having implemented and tested various memristor models, both in SPICE and in Verilog, it is now time to choose which one to use to test more complex circuits and build logic gates. In order to choose which code would be used to implement logic gates firstly the performance of a HP linear SPICE code and Verilog were used. Both codes were tested alone and then using two in series, with direct, reverse, and mixed polarity. In table 3.2 the results are shown.

Table 3.2: Comparison between SPICE and Verilog code

		Verilog					SPICE				
		One Memritor		Two Memristors			One Memritor		Two Memristors		
Polarity		Direct	Reverse	Direct	Reverse	Mixed	Direct	Reverse	Direct	Reverse	Mixed
Initial condition solution time	CPU[ms]	0	0	0	0	0	1	1	1	1.999	1
	Elapsed [us]	317.097	309.944	331.879	355.959	333.071	1025.92	990.868	1396.18	1457.93	1467.94
Intrinsic tran analysis time	CPU[ms]	3.999	4	4.998	5.999	5	4.999	2.999	2.999	4	3
	Elapsed [ms]	72.8819	81.0189	64.873	73.364	72.5942	63.364	71.357	79.5841	71.0289	71.275
Total time required for tran analysis	CPU[ms]	5.999	5.99	6.998	6.999	7	6.999	4.999	6.999	6.999	6
	Elapsed [ms]	74.9748	83.132	66.8988	75.5749	74.753	66.062	73.956	82.696	74.2409	74.419
Time accumulated	CPU[ms]	86.986	86.986	87.985	88.986	86.986	88.985	88.985	89.985	89.986	89.986
	Elapsed [ms]	375.262	376.344	401.93	385.851	385.03	392.879	408.977	375.968	401.337	350.471
Peak resident memory used [Mbytes]		47.9	47.9	48	48	48	49.2	49.2	49.2	49.2	51.2

As it can be seen in table 3.2, when comparing two simple models, the only considerable difference is in the initial condition solution time, with the Spice code having a larger initial condition solution time. With this it could be concluded that both codes have a similiar performance when used in one or two memristors in series. However, due to problems importing SPICE code to cadence environment, in the case of the Simmons code, functions such as $stp(x)$ are not recognized in the cadence Spice compiler; and, in the case of the Vourkas model, convergence problems, made it impossible to compare more complex SPICE to Verilog codes.

One other issue to consider is, as it can be seen in figure 3.23, where two examples of the simulation info of the two codes are shown, the SPICE code does not allow for step control, i.e., the user can't control the simulation steps, unlike the Verilog model.

```
tran: time = 55.83 ms    (2.79 %), step = 25.83 ms    (1.29 %)
tran: time = 175.8 ms    (8.79 %), step = 40 ms         (2 %)
tran: time = 255.8 ms    (12.8 %), step = 40 ms         (2 %)
tran: time = 375.8 ms    (18.8 %), step = 40 ms         (2 %)
tran: time = 455.8 ms    (22.8 %), step = 40 ms         (2 %)
tran: time = 565.9 ms    (28.3 %), step = 25.77 ms    (1.29 %)
tran: time = 675.9 ms    (33.8 %), step = 40 ms         (2 %)
tran: time = 755.9 ms    (37.8 %), step = 40 ms         (2 %)
tran: time = 875.9 ms    (43.8 %), step = 40 ms         (2 %)
tran: time = 955.9 ms    (47.8 %), step = 40 ms         (2 %)
tran: time = 1.076 s     (53.8 %), step = 40 ms         (2 %)
tran: time = 1.156 s     (57.8 %), step = 40 ms         (2 %)
tran: time = 1.276 s     (63.8 %), step = 40 ms         (2 %)
tran: time = 1.356 s     (67.8 %), step = 40 ms         (2 %)
tran: time = 1.476 s     (73.8 %), step = 40 ms         (2 %)
tran: time = 1.564 s     (78.2 %), step = 22.09 ms     (1.1 %)
tran: time = 1.676 s     (83.8 %), step = 40 ms         (2 %)
tran: time = 1.756 s     (87.8 %), step = 40 ms         (2 %)
tran: time = 1.876 s     (93.8 %), step = 40 ms         (2 %)
tran: time = 1.956 s     (97.8 %), step = 40 ms         (2 %)
Number of accepted tran steps =           64
```

(a) Spice code

```
tran: time = 60 ms         (3 %), step = 20 ms          (1 %)
tran: time = 160 ms        (8 %), step = 20 ms          (1 %)
tran: time = 260 ms       (13 %), step = 20 ms          (1 %)
tran: time = 360 ms       (18 %), step = 20 ms          (1 %)
tran: time = 460 ms       (23 %), step = 20 ms          (1 %)
tran: time = 560 ms       (28 %), step = 20 ms          (1 %)
tran: time = 660 ms       (33 %), step = 20 ms          (1 %)
tran: time = 760 ms       (38 %), step = 20 ms          (1 %)
tran: time = 860 ms       (43 %), step = 20 ms          (1 %)
tran: time = 960 ms       (48 %), step = 20 ms          (1 %)
tran: time = 1.06 s       (53 %), step = 20 ms          (1 %)
tran: time = 1.16 s       (58 %), step = 20 ms          (1 %)
tran: time = 1.26 s       (63 %), step = 20 ms          (1 %)
tran: time = 1.36 s       (68 %), step = 20 ms          (1 %)
tran: time = 1.46 s       (73 %), step = 20 ms          (1 %)
tran: time = 1.56 s       (78 %), step = 20 ms          (1 %)
tran: time = 1.66 s       (83 %), step = 20 ms          (1 %)
tran: time = 1.76 s       (88 %), step = 20 ms          (1 %)
tran: time = 1.86 s       (93 %), step = 20 ms          (1 %)
tran: time = 1.96 s       (98 %), step = 20 ms          (1 %)
Number of accepted tran steps =           100
```

(b) Verilog code

Figure 3.23: Simulation info of Spice and Verilog simulations

During the testing of the various models, the biggest problem that arose was when two memristors were connected in series. In the case of all the spice models, except the HP model, they use current sources to model the behaviour of the memristor, what happens then is, when two memristors are connected in series, two current sources are connected in series, which cannot occur. To solve this, one of the solutions is to add a resistor with a high value (compared to highest memristance of the memristor) in series with each memristor.

The last point to consider pertains to which logic will be used. So far two different methods of implementing logic gates using memristors have been developed, IMPLY logic and MAGIC logic.

An example of a circuit using IMPLY logic can be seen in figure 3.24 [11]. In this case a resistor R_G, with a value between R_{ON} andR_{OFF} of each memristor, R_G is connected to two memristors p an q, both act as inputs, and the result is written in q. The way it works is to first apply a voltage V_{SET} higher than V_{COND}. If p is 1, i.e, if it has a low resistance, the voltage on the common terminal of both memristor is approximately V_{COND}, and the voltage on q is $V_{SET} - V_{COND}$, which is sufficiently small to maintain the logic state of q. In the case where p is 0 and q is 0, i.e, both have a high resistance, the applied voltage

on q is approximately V_{SET} and q is switched ON. In the case where p is 0 and q is 1, the logic state of q is maintained [11].

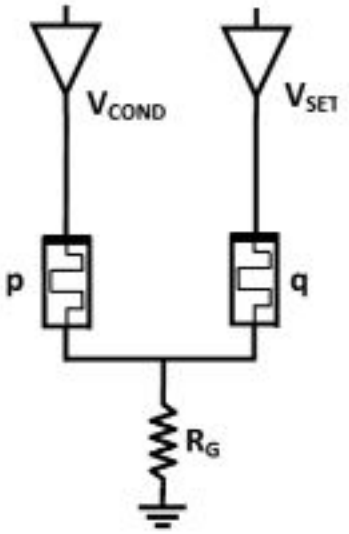


Figure 3.24: IMPLY logic example circuit.

An example of MAGIC logic can be seen in figure 3.25 [10], more specifically, a NOR. In this logic, two memristors act as inputs and a single memristor acts as an output. Like in the IMPLY logic, the logic states are saved as the devices memristance. The operation of the MAGIC gate consists in applying a voltage across the logic gate, and seeing if the resulting voltage across the output memristor is enough to change its configuration.

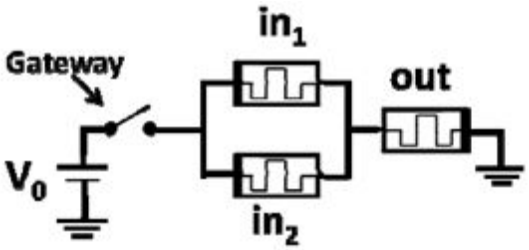


Figure 3.25: MAGIC logic example circuit.

Comparing both logics, it can be seen that the MAGIC logic is a more simple way to build logic gates. It only requires one voltage to be applied, it doesn't require a sequential voltage application in different locations, which would require a dedicated controller. It doesn't require an additional resistor, which means less dissipated power. However, it should be noted that due to the problem specified earlier, where the spice models of the Simmons and Vourkas memristors can't be connected in series, only by adding resistors in parallel, means they can't be used to build MAGIC logic gates.

With all of the considerations being presented, the model chosen is the VTEAM model, with a Veriloga code. Although both codes were shown to have similar performance, to

take advantage of the simulator tools and, to use MAGIC logic to implement logic gates, the VTEAM model is the better suited.

3.5 Summary

In this chapter, the various macromodels of the memristor models were presented and tested. In the HP model, the various window functions were tested. It can be concluded that for the same applied voltage results in larger or smaller resistance ranges, for example, with a strukov window the memristance varied between 11k and 12k ohm and with he Joglekar window it varies between 0 and 11k ohm. Then, the Simmons model was tested, like the previous model, a sinusoidal wave was applied in the memristor and its results were shown. The Vourkas model was tested next, it is a voltage-controlled model with voltage thresholds. When a sinusoidal wave was applied, its behavior was shown, i.e, when the voltage thresholds are exceed, it quickly switches configuration. The VTEAM model was then tested using verilog code, since the VTEAM model changes the TEAM models current thresholds to voltage thresholds, it was chosen to be the one used in the further chapter, since it better fits to the objective of building logic circuits, also due to the advantages of the VTEAM model seen in table 2.1 and in the section 3.4.

Chapter 4

Complex Memristor circuits

In this chapter, more complex circuits were tested, using the VTEAM model. First simple circuits composed of two memristors were tested, then, logic gates circuits will be developed and tested.

4.1 Circuits composed of two memristors

In this section, circuits composed of two memristors will be tested. Firstly, in a parallel configuration then in series. Both memristors will have the same parameters specified in table 3.1, i.e., both will have the same resistance value of R_{ON} and R_{OFF}, $1k\Omega$ and $300k\Omega$ respectively. Both will also have the same values for thresholds voltages, i.e., the same value of VT_{ON} and VT_{OFF}, $-1.5V$ and $0.3V$ respectively.

4.1.1 Parallel Configuration

In a parallel configuration, the same voltage is applied in both memristors, meaning the resulting memristance of each memristor will be the same as when tested in a single memristor. Since they're connected in parallel R_{on},witch is much smaller than R_{off} will dominate the total memristance of each circuit, meaning one of the most important aspect of the circuit will be when a memristor switches to a ON configuration.

4.1.1.1 Direct Polarity

OFF-OFF configuration

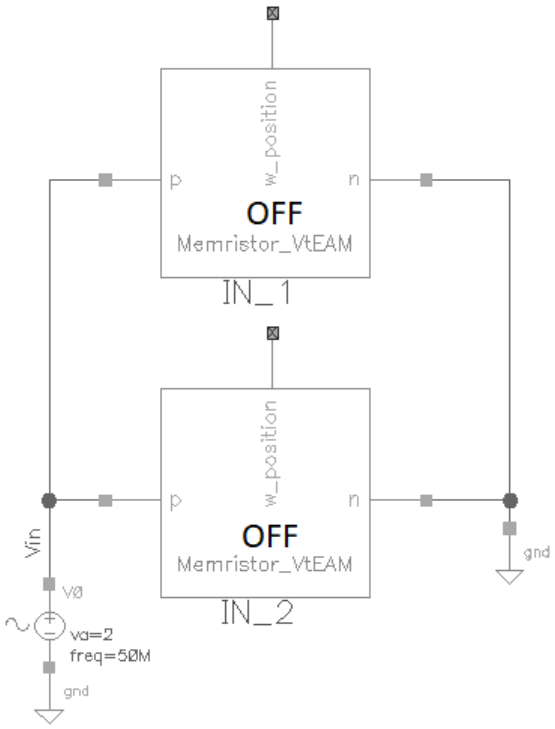
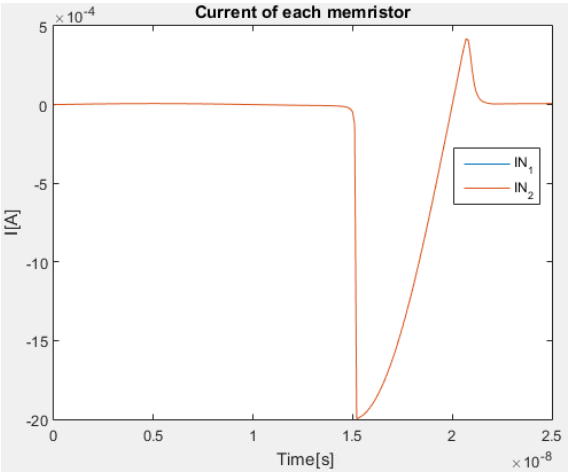
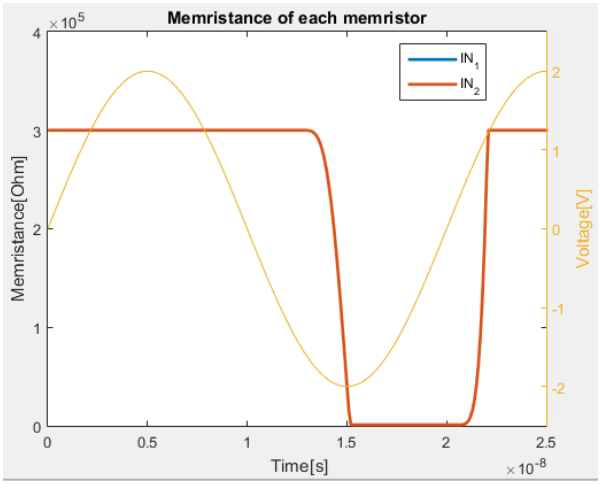


Figure 4.1: Circuit to test two parallel memristors with direct polarity and OFF-ON configuration

When two memristors are connected in parallel, both with direct polarity and the same initial configuration of OFF. Since they have the same parameters, both will switch to an ON configuration when $-1.5V$ are applied across both devices, and back to an OFF configuration when $0.3V$ is applied across both memristors. This means that both memristors switch configuration at the same instant. Since they are connected in parallel and they have the same memristance values, the resulting total memristance of the circuit will be half of a single memristor. This behavior, along with the current of each memristor and current of the circuit can be seen in figures 4.2 and 4.2.

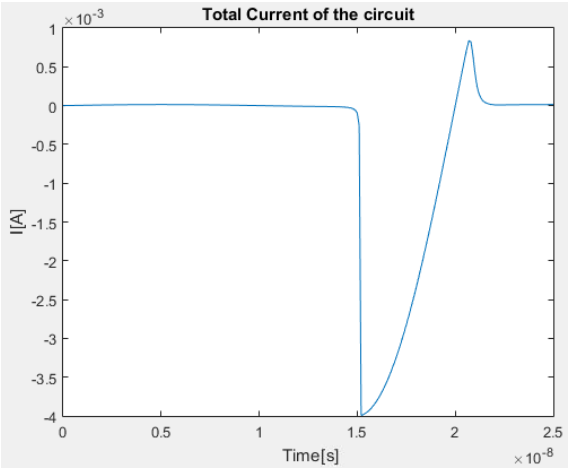


(a) Current of both memristors

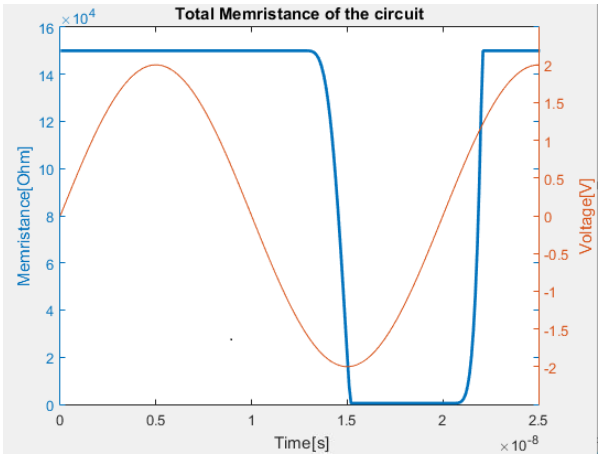


(b) Memristance of both memristors

Figure 4.2: Memristance and current for two parallel memristors with direct polarity and OFF-OFF configuration



(a) Total current of the circuit



(b) Total memristance of the circuit

Figure 4.3: Total memristance and current for two parallel memristors with direct polarity and OFF-OFF configuration

OFF-ON configuration

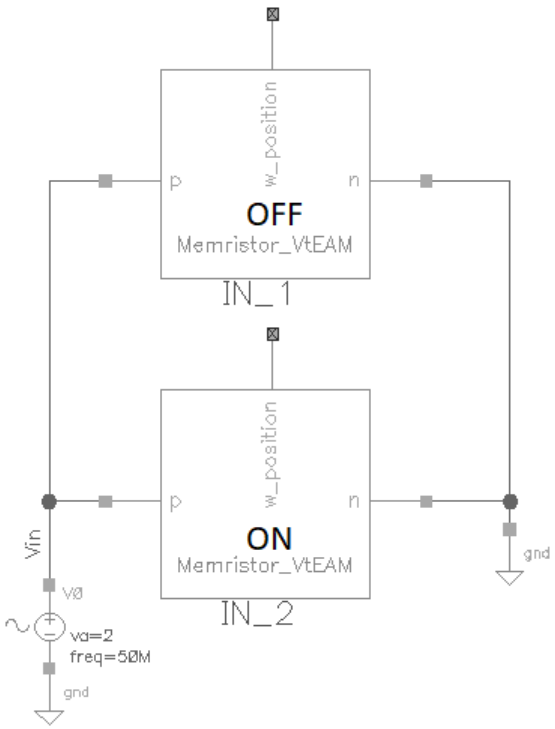
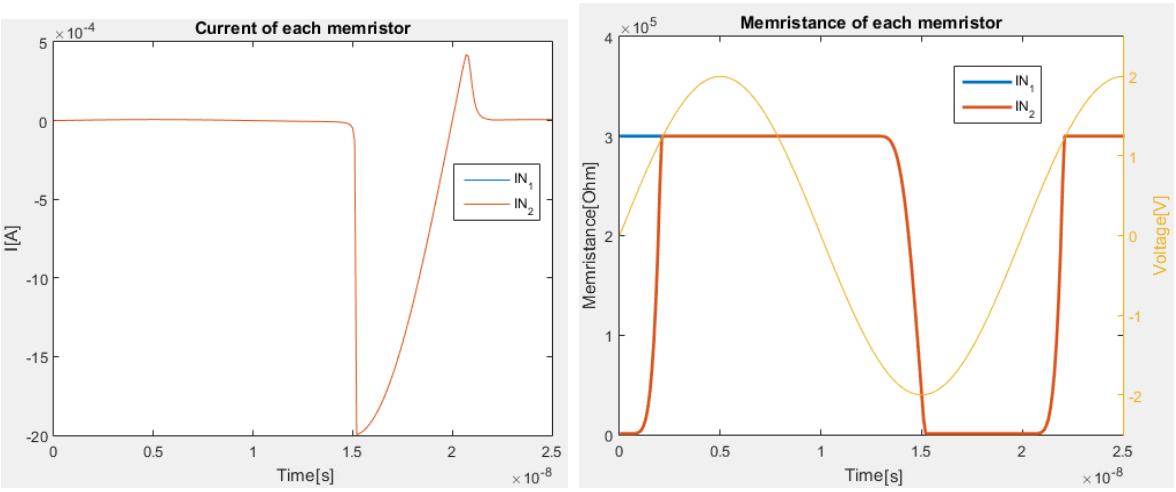


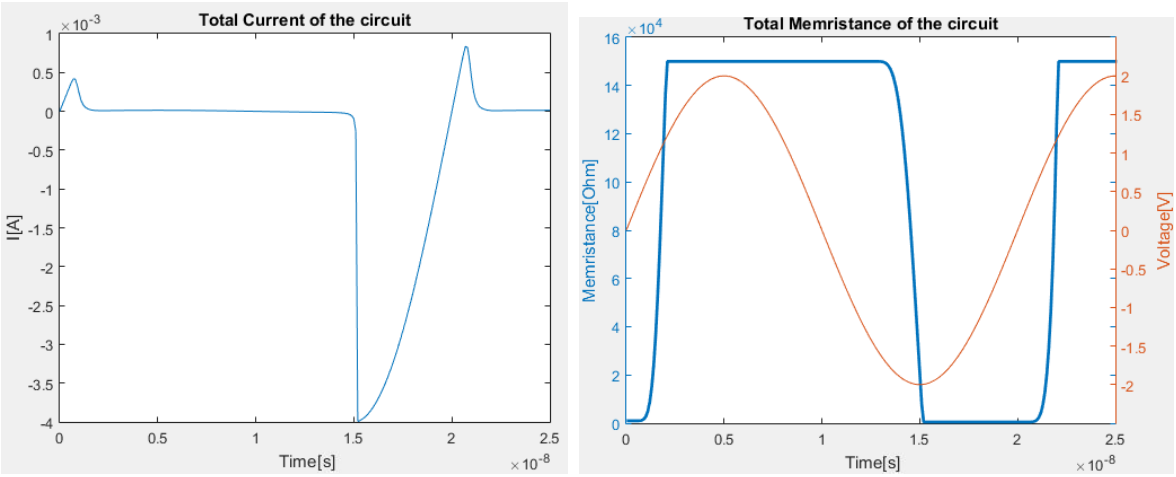
Figure 4.4: Circuit to test two parallel memristors with direct polarity and OFF-ON configuration

In figures 4.5 and 4.6 the effect of having different initial configurations on both memristors can be seen, since R_{IN2} has a much smaller memristance than R_{IN1}, it will result in a dip in the total memristance of the circuit as it can be seen in the figure 4.6. Only when $0.3V$ are applied, will the memristor R_{IN2} switch to an OFF configuration and behave the same as in the previous setup, i.e, the resulting memristance of the circuit will be half of a single memristor.



(a) Current of both memristors (b) Memristance of both memristors

Figure 4.5: Memristance and current for two parallel memristors with direct polarity and OFF-ON configuration



(a) Total current of the circuit (b) Total memristance of the circuit

Figure 4.6: Total memristance and current for two parallel memristors with direct polarity and OFF-ON configuration

ON-ON configuration

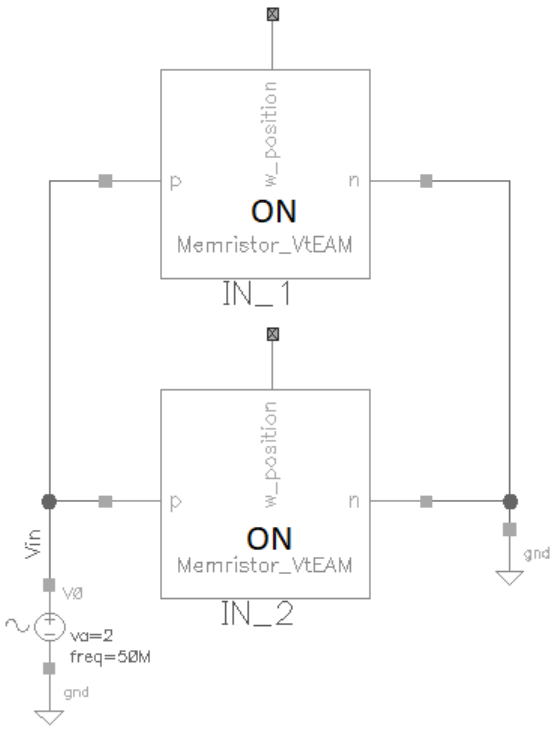
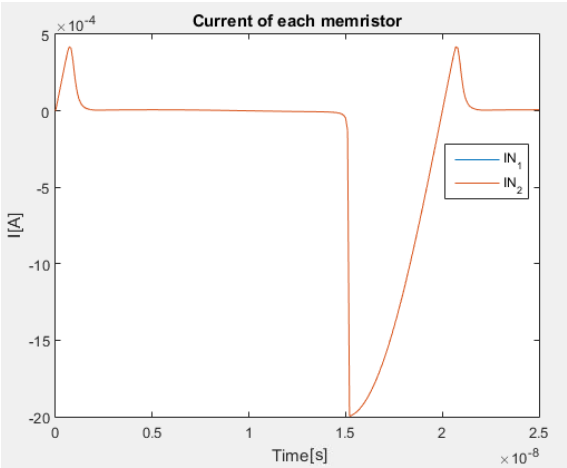
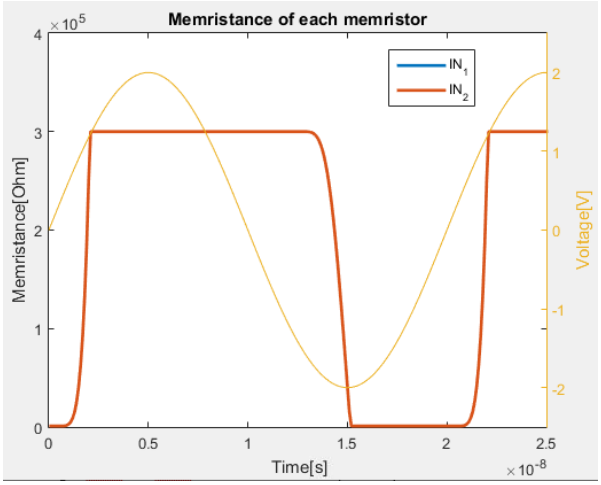


Figure 4.7: Circuit to test two parallel memristors with direct polarity and ON-ON configuration

When both memristors are connected in a ON configuration, the circuit will behave similarly as when they are both connected in an OFF configuration, i.e., they will always have the same memristance value and threshold voltages, which will result in a total memristance of the circuit of half of a single memristor. This behaviour, along with the current of each memristor and total current of the circuit can be seen in figures 4.8 and 4.9.

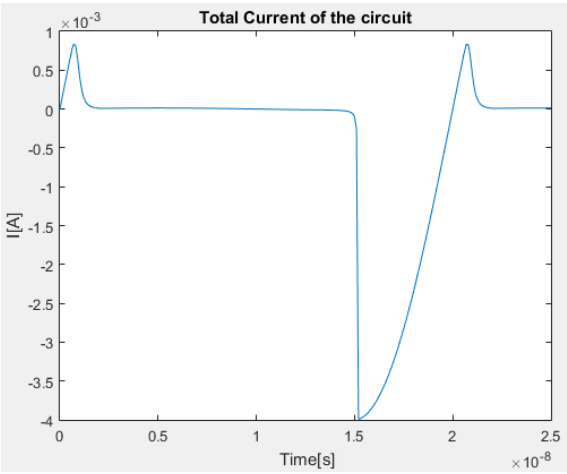


(a) Current of both memristors

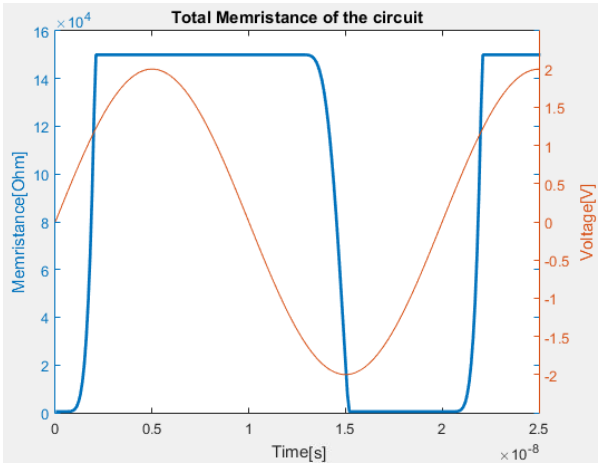


(b) Memristance of both memristors

Figure 4.8: Memristance and current for two parallel memristors with direct polarity and ON-ON configuration



(a) Total current of the circuit



(b) Total memristance of the circuit

Figure 4.9: Total memristance and current for two parallel memristors with direct polarity and ON-ON configuration

Conclusions

Having tested all the configuration combinations possible, it can be concluded that, when two memristors are connected in parallel with direct polarity; if they have the same configuration and parameters the total circuit will behave as a single memristor with half the value of R_{OFF} and R_{ON} as the memristors used, with the same threshold voltages; in this case it will have a R_{OFF} of $150k\Omega$ and R_{ON} of 500Ω, and threshold voltages VT_{ON} and VT_{OFF} of $-1.5V$ and $300mV$.

4.1.1.2 Reverse Polarity

As it was seen in section 3.3.2 when a memristor is connected with reverse polarity, its thresholds can be seen as having changed to its symmetrical values. As such, to change a memristor from OFF to ON configuration it will require 1.5V and, to change it back to OFF configuration $-0.3V$ will be required.

OFF-OFF configuration

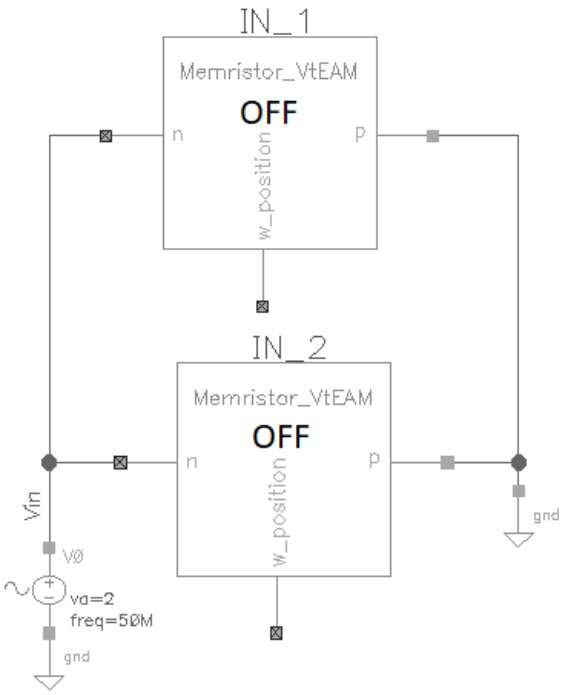
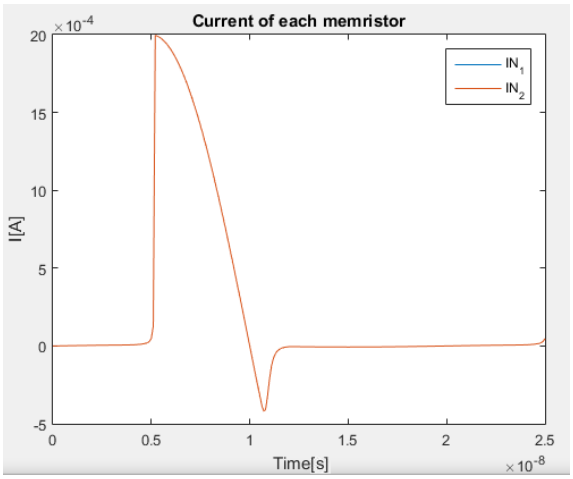
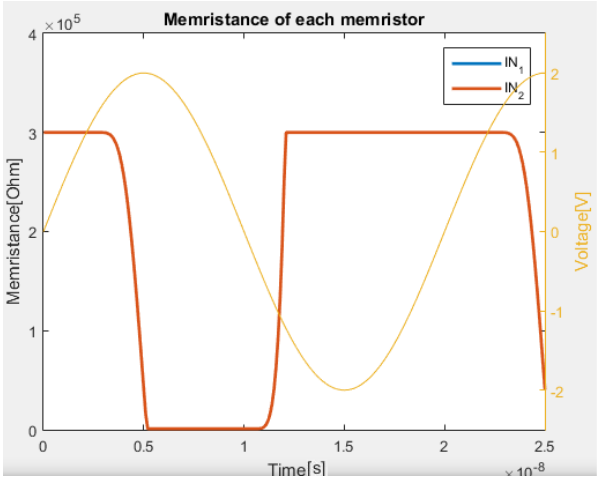


Figure 4.10: Circuit to test two parallel memristors with reverse polarity and OFF-OFF configuration

Much like in the direct polarity case, since both memristors have the same parameters and starting configurations, they have the exact same behaviour. This means the total circuit will behave like a single memristor with half the memristance values as a single memristor, threshold voltages VT_{ON} and VT_{OFF} of $1.5V$ and $-300mV$. This behavior, along with the current of each memristor and total current of the circuit can be seen in figures 4.11 and 4.12.

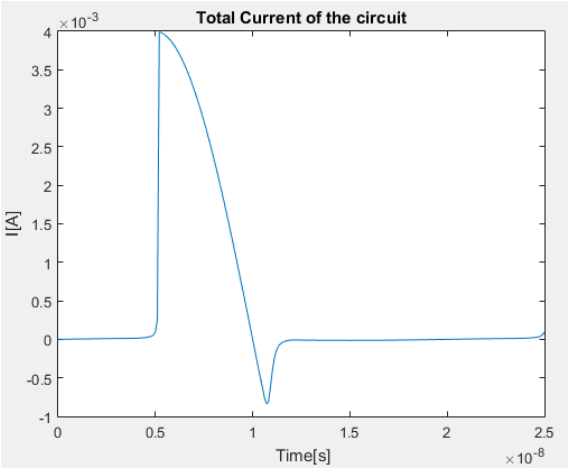


(a) Current of both memristors

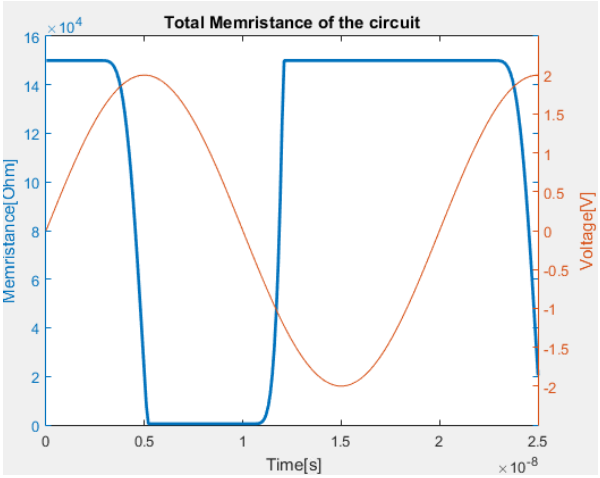


(b) Memristance of both memristors

Figure 4.11: Memristance and current for two parallel memristors with reverse polarity and OFF-OFF configuration



(a) Total current of the circuit



(b) Total memristance of the circuit

Figure 4.12: Total memristance and current for two parallel memristors with reverse polarity and OFF-OFF configuration

OFF-ON configuration

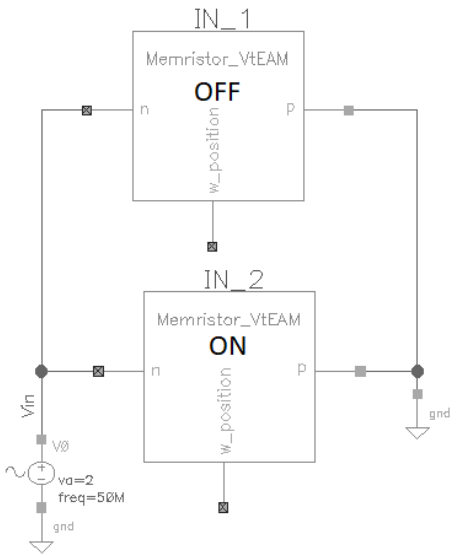
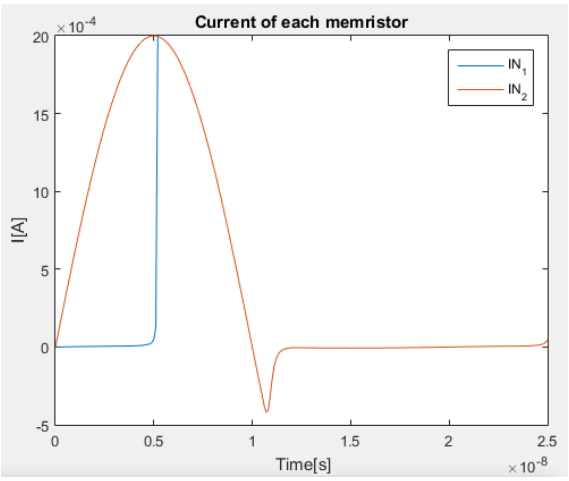
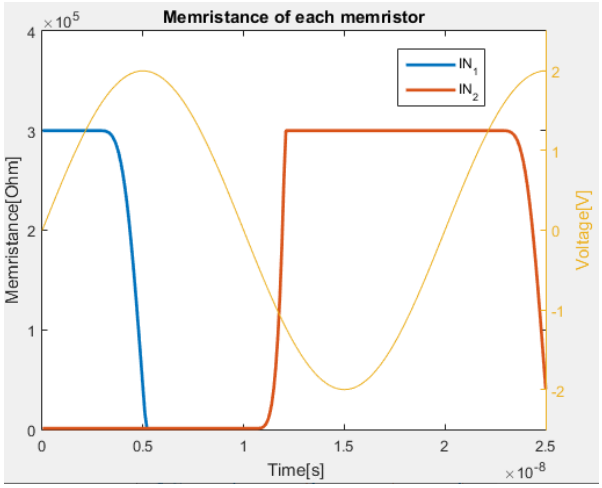


Figure 4.13: Circuit to test two parallel memristors with reverse polarity and OFF-ON configuration

Similarly to the direct polarity *OFF-ON* setup, when one of the two memristors connected in parallel has a much smaller memristance than the other one it will result in a small total memristance. In this case, since the memristor R_{IN2} stays in a *ON* configuration during the positive voltage, it will result in a small total memristance. Then when $1.5V$ is applied in the memristor R_{IN1}, it will switch to an *ON* configuration. From this point on, both memristors will have the same memristance value, and once again the total memristance will be half of a single memristor. This behavior, along with the current of each memristor and total current of the circuit can be seen in figures 4.14 and 4.15.

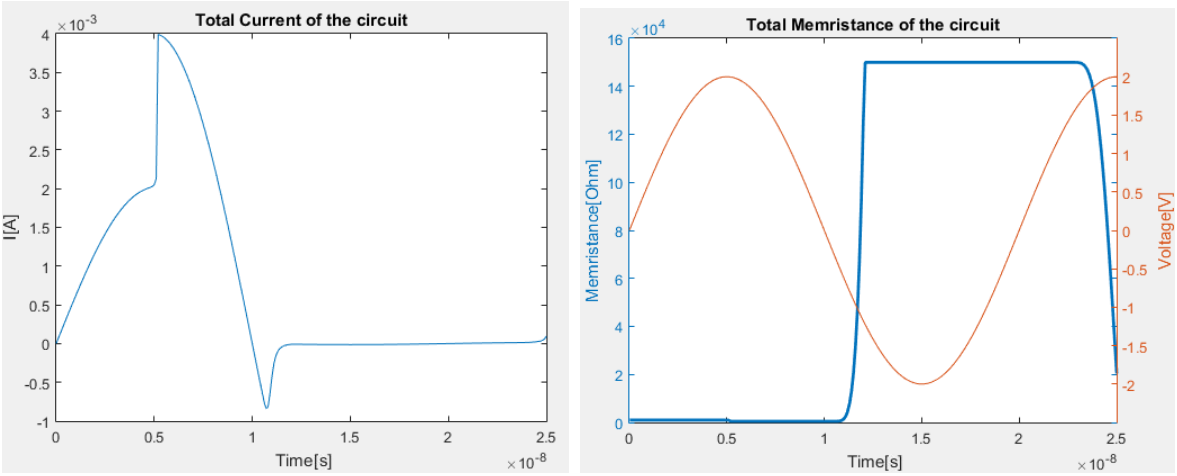


(a) Current of both memristors



(b) Memristance of both memristors

Figure 4.14: Memristance and current for two parallel memristors with reverse polarity and OFF-ON configuration



(a) Total current of the circuit

(b) Total memristance of the circuit

Figure 4.15: Total memristance and current for two parallel memristors with reverse polarity and OFF-ON configuration

ON-ON configuration

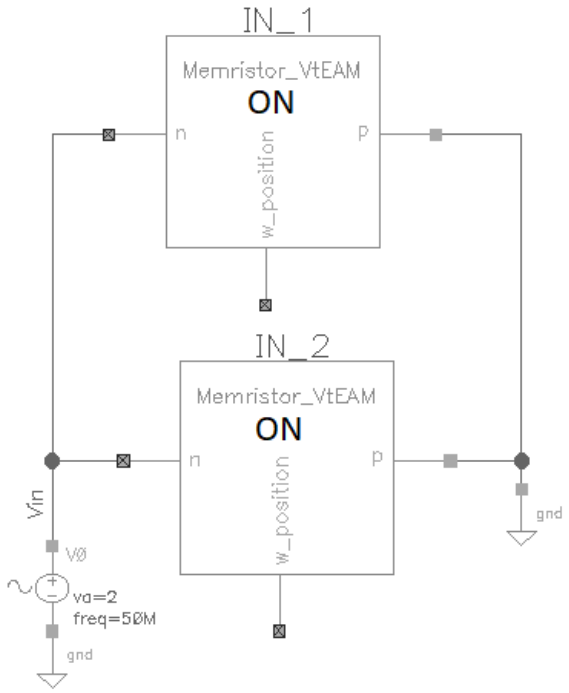
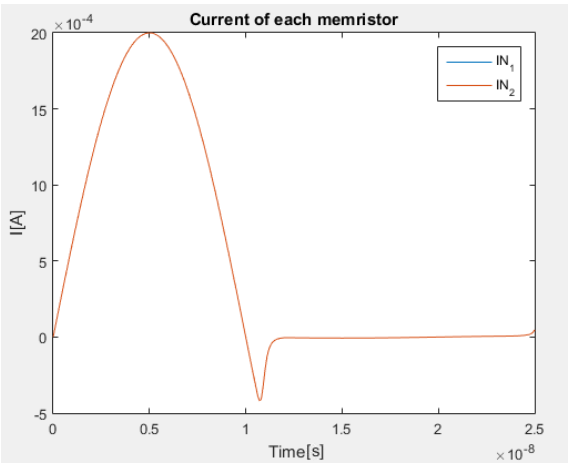
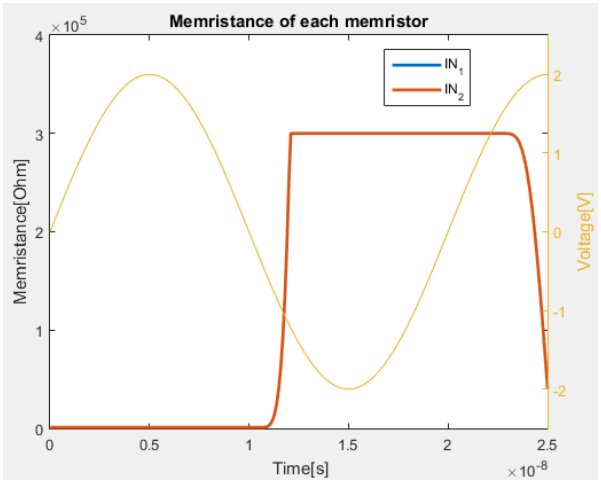


Figure 4.16: Circuit to test two parallel memristors with reverse polarity and ON-ON configuration

Once again as it can be seen in the figures 4.17 and 4.18, both memristors have the same starting configuration, which will result in both memristors always having the same memristance values, meaning, the total memristor of the circuit will have the same thresholds and half the memristance of a single memristor.

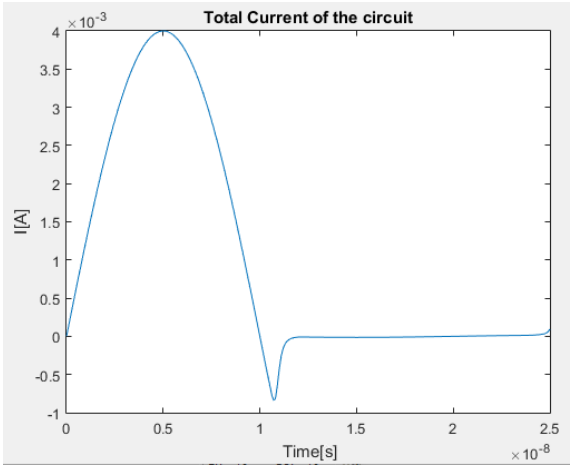


(a) Current of both memristors

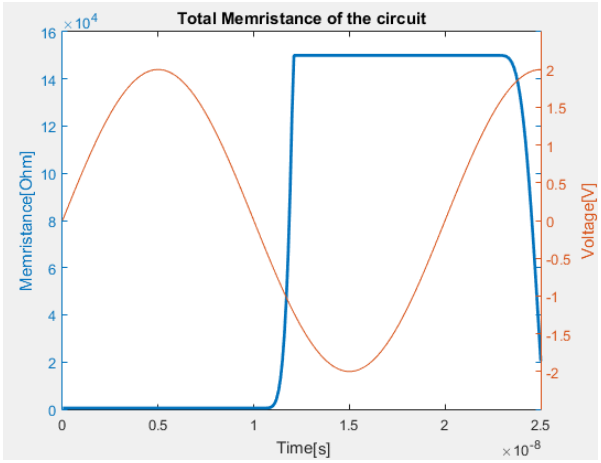


(b) Memristance of both memristors

Figure 4.17: Memristance and current for two parallel memristors with reverse polarity and OFF-ON configuration



(a) Total current of the circuit



(b) Total memristance of the circuit

Figure 4.18: Total memristance and current for two parallel memristors with reverse polarity and ON-ON configuration

Conclusions

Having tested all the configuration combinations possible, it can be concluded that, when two memristors are connected in parallel with reverse polarity; if they have the same starting configuration and parameters the total circuit will behave as a single memristor with half the value of R_{OFF} and R_{ON} as the memristors used, and with the same threshold voltages. In this case it, can also be seen the effect of having one memristor with an OFF configuration and another with ON for a relatively larger amount of time. Since they are connected in parallel the total memristance will stay at a small value until both memristors go back to an OFF configuration.

4.1.1.3 Mixed Polarity

As it was seen in the previous sections, when two memristors are connected in parallel, regardless if they are setup in a direct or with reverse polarity, and their initial configuration, after a small delay they will have the same memristance. This results in a total memristance of half of a single memristor and unaltered thresholds. Now the behavior when two memristors are connected with different connections, i.e., one with direct and the other with reverse polarity will be examined. Having different polarities means the memristors will have different thresholds, in this case R_{IN1} will have a VT_{ON} and VT_{OFF} of $-1.5V$ and $300mV$ respectively, and, R_{IN2} will have a VT_{ON} and VT_{OFF} of $1.5V$ and $-300mV$.

OFF-OFF configuration

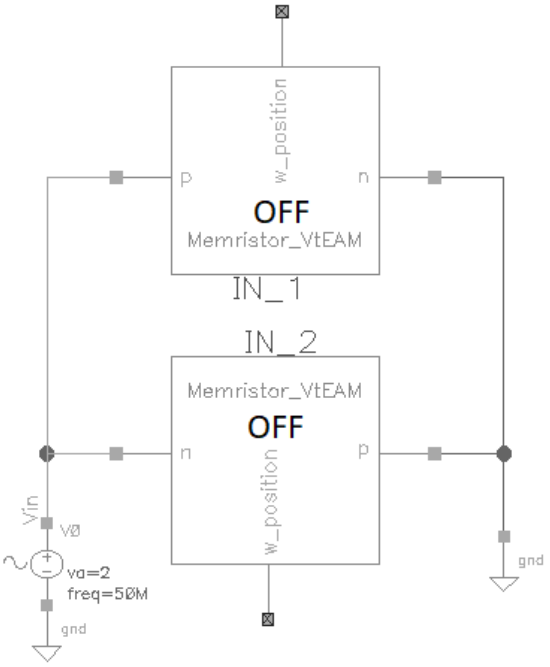
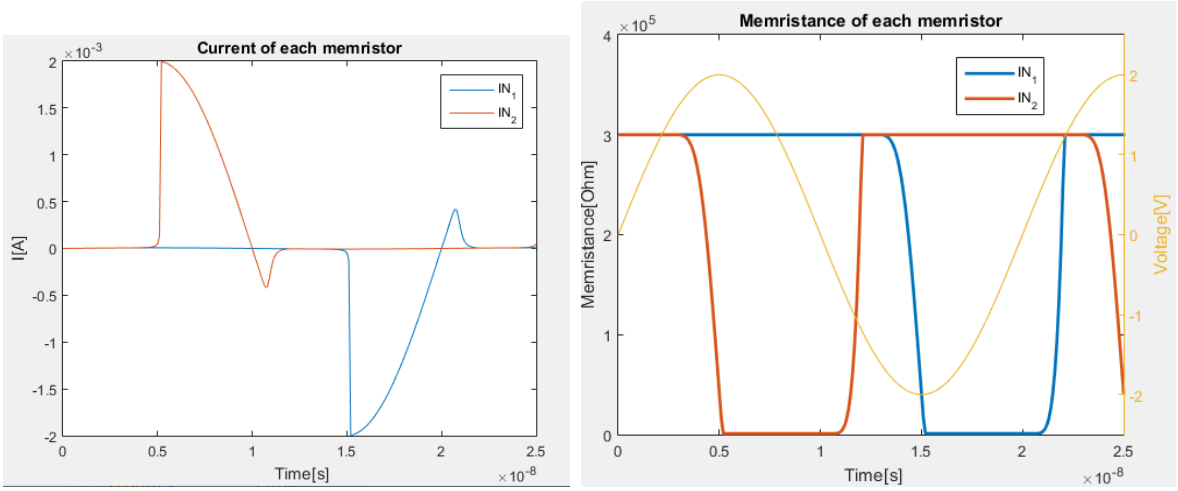


Figure 4.19: Circuit to test two parallel memristors with mixed polarity and OFF-OFF configuration

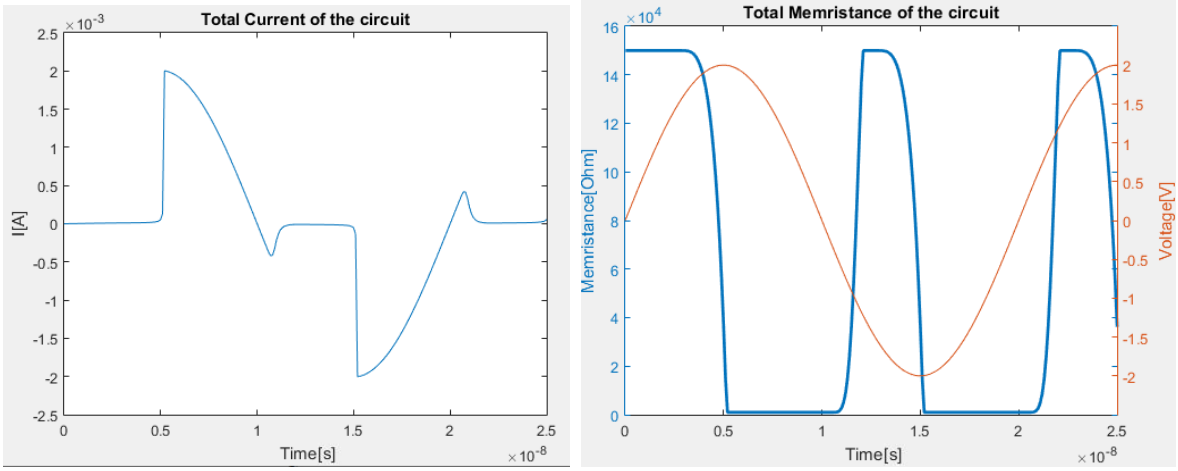
In figures 4.20 and 4.21, the memristance of both memristors, the total memristance as well as the current can be seen. Since both start with the same memristance, the total memristance will be half, i.e.,$150k\Omega$, then, when $1.5V$ is applied in R_{IN2}, it will cause it to switch to an ON configuration, which will cause a drop in the total memristance of the circuit to a small value. When $-0.3V$ are then applied, R_{IN2} will go back to an OFF configuration, which will cause an increase back to $150k\Omega$ in the total memristance because R_{IN1} stayed at an OFF configuration. Then, since $-1.5V$ are applied in R_{IN1}, it will switch to an ON configuration, which will again decrease the total memristance to a very small value. When the applied voltage goes up to $0.3V$, R_{IN1} will go back to an OFF configuration which will increase the total memristance because R_{IN2} has stayed at

an *OFF* configuration. The difference in the voltage thresholds is what causes the spikes in memristance in the total memristance of the circuit.



(a) Current of both memristors (b) Memristance of both memristors

Figure 4.20: Memristance and current for two parallel memristors with mixed polarity and OFF-OFF configuration



(a) Total current of the circuit (b) Total memristance of the circuit

Figure 4.21: Total memristance and current for two parallel memristors with mixed polarity and OFF-OFF configuration

OFF-ON configuration

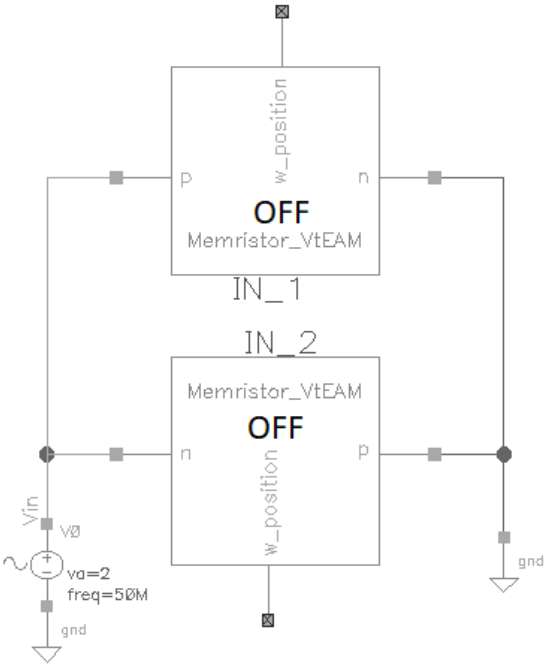
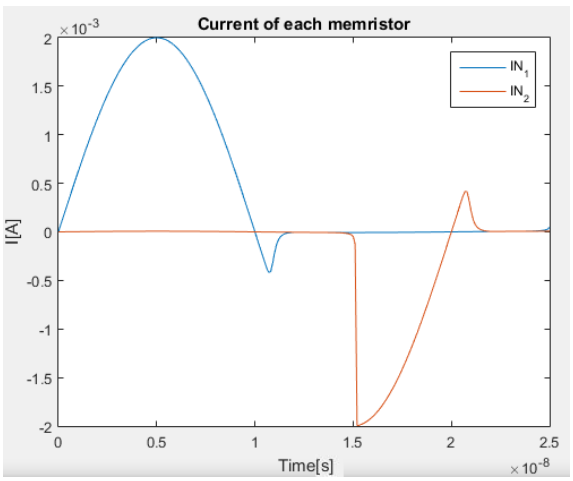
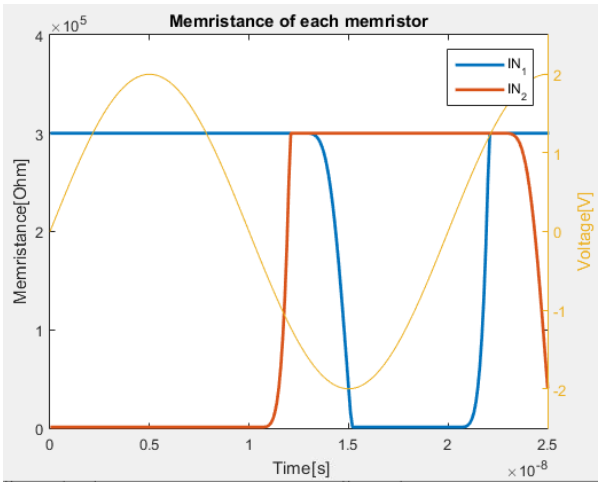


Figure 4.22: Circuit to test two parallel memristors with mixed polarity and OFF-ON configuration

In this case, since R_{IN2} starts at an *ON* configuration, the total memristance will stay at a small memristance value until R_{IN2} switches to an *OFF* configuration, that only occurs when $-0.3V$ is applied. Then, since $-1.5V$ is applied at R_{IN1}, it will switch to an *ON* configuration which will cause a drop in the total memristance of the circuit.

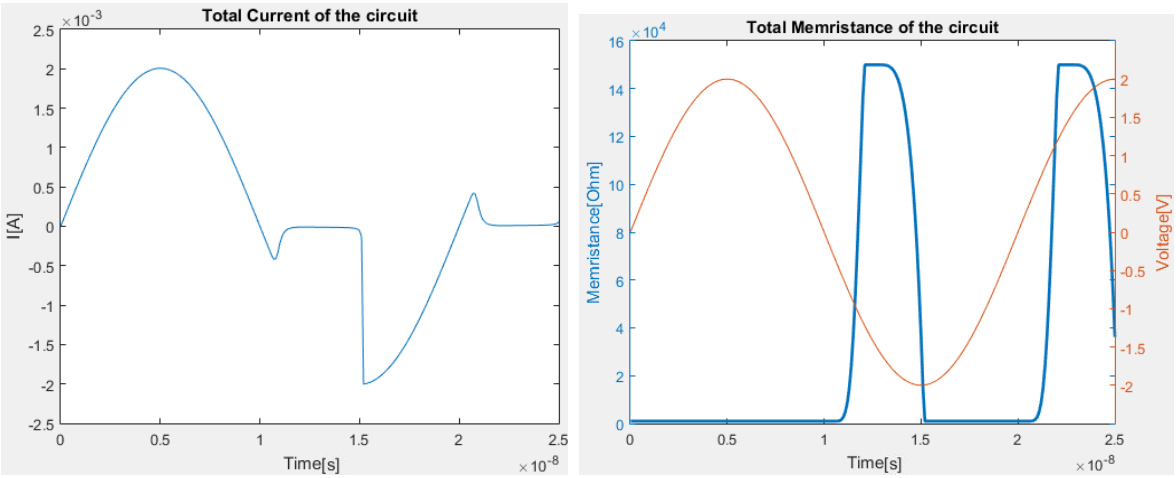


(a) Current of both memristors



(b) Memristance of both memristors

Figure 4.23: Memristance and current for two parallel memristors with mixed polarity and OFF-ON configuration



(a) Total current of the circuit

(b) Total memristance of the circuit

Figure 4.24: Total memristance and current for two parallel memristors with mixed polarity and OFF-ON configuration

ON-ON configuration

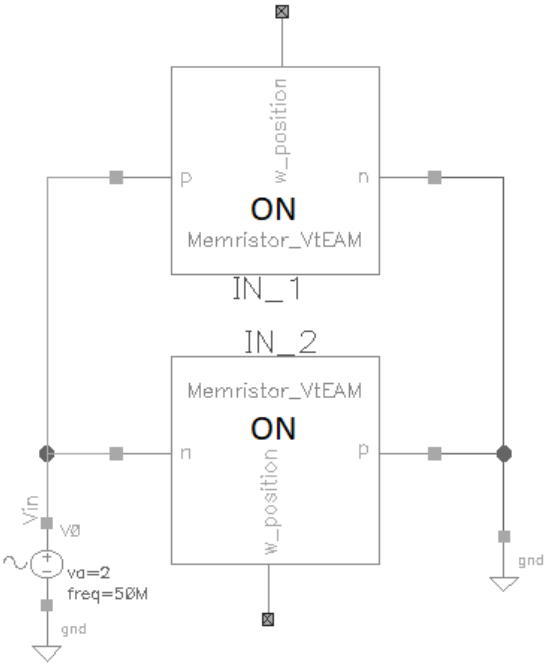
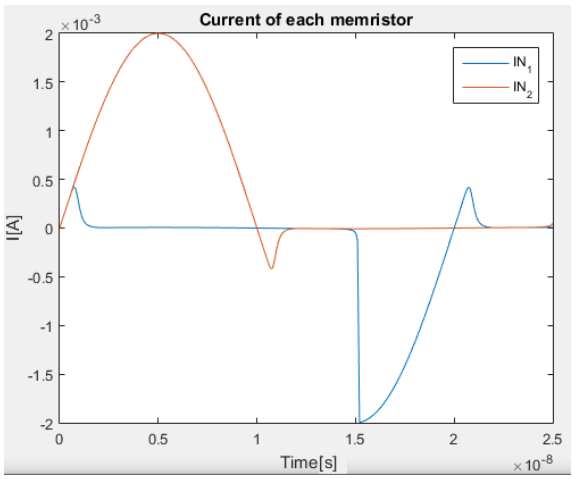


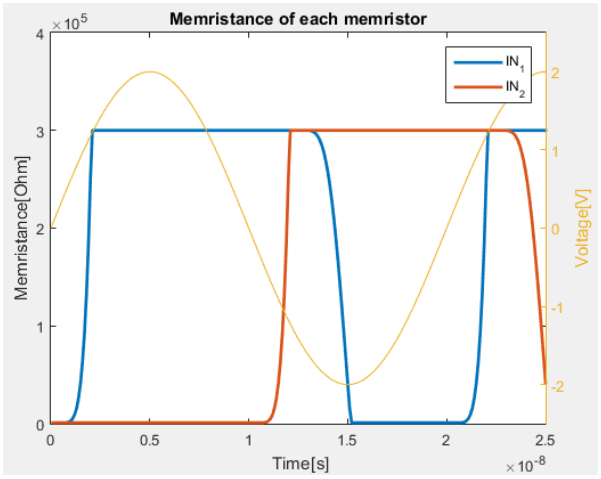
Figure 4.25: Circuit to test two parallel memristors with mixed polarity and ON-ON configuration

In this case, since they both start at an ON configuration, when $0.3V$ is applied, R_{IN1} will switch to an OFF configuration, but, since R_{IN2} remains at an ON configuration, the total memristance of the circuit will remain at a low value. Only when R_{IN2} switches to an OFF configuration, and R_{IN1} stays at an OFF configuration will an increase in the total memristance occur. Then, like in the previous cases, R_{IN1} will switch back to an

ON configuration and cause the spike in the total memristance.

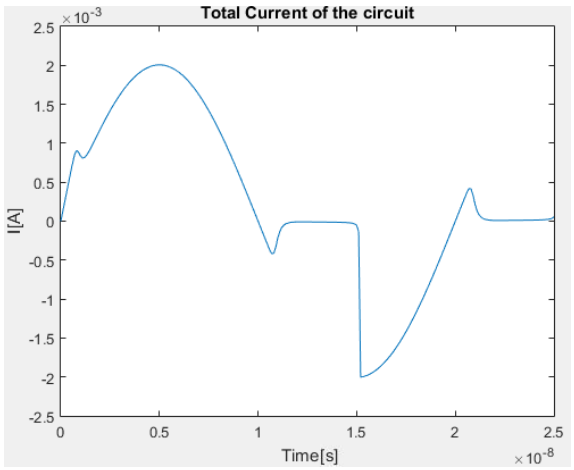


(a) Current of both memristors

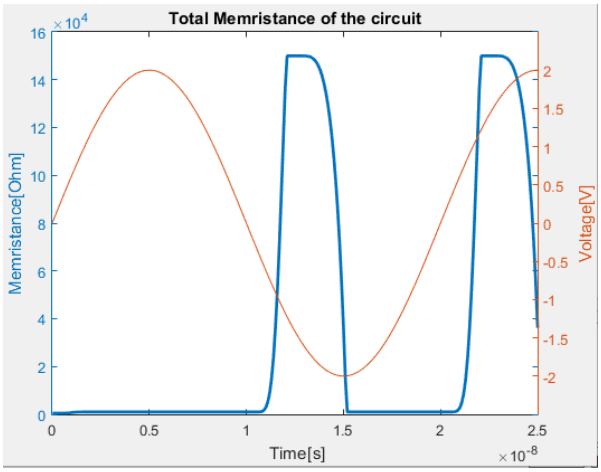


(b) Memristance of both memristors

Figure 4.26: Memristance and current for two parallel memristors with mixed polarity and ON-ON configuration



(a) Total current of the circuit



(b) Total memristance of the circuit

Figure 4.27: Total memristance and current for two parallel memristors with mixed polarity and ON-ON configuration

Conclusions

Having tested all the configuration combinations possible, it can be concluded that, when two memristors are connected in parallel with mixed polarity, since they have different voltage thresholds, they will switch from *OFF* to *ON* and from *ON* to *OFF* at different voltages, it will result in spikes in the total memristance.

4.1.2 Series Configuration

When two memristors are connected in series, the voltage divider between both is what is going to decide if the memristors switch configurations. To be able to see this behavior more clearly, the amplitude of the voltage source was increased to 5V, this value was chosen because it was bigger then $2 \times VT_{OFF}$, which allows the voltage between both memristors to be 2.5, which allows both memristors to switch easily from an OFF to an ON configuration. Due to convergence problems, R_{OFF} and R_{ON} were lowered to $1k\Omega$ and 50Ω respectively and the frequency of the voltage source was lowered to $10GHz$.

4.1.2.1 Direct Polarity

OFF-OFF configuration

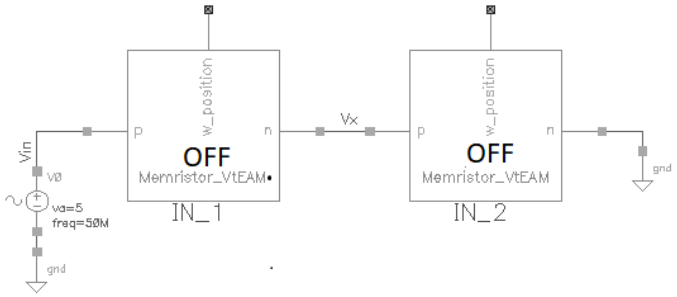
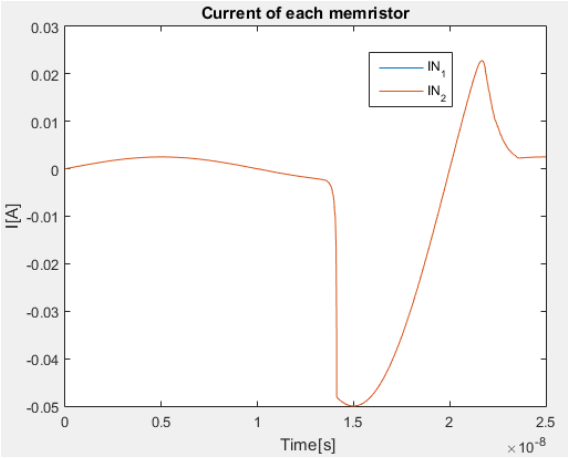
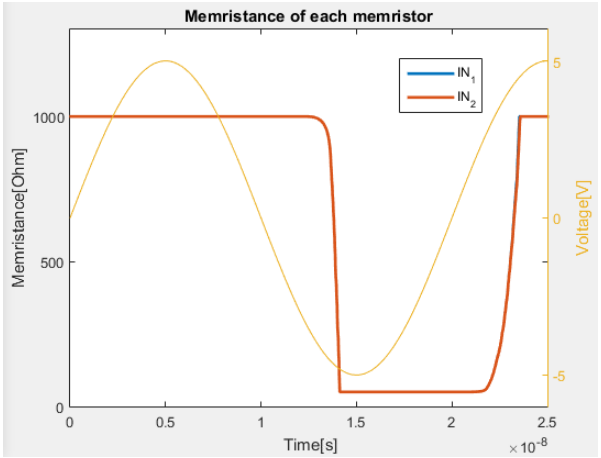


Figure 4.28: Circuit to test two memristors in series with direct polarity and OFF-OFF configuration

When two memristors are connected in series, both with direct polarity and the same initial configuration of OFF. Since they have the same parameters both will switch to an ON configuration when the voltage across both devices is $-1.5V$. Since both memristors have the same starting configuration, i.e., the same memristance value, the voltage between both of them will be half the applied voltage, this means that only when the applied voltage is lower than $-3V$ will the voltage between both memristors be lower than $-1.5V$, which will cause both memristors to switch to an ON configuration. Again, since they have the same memristace, a voltage of at least $600mV$ will be required to be applied in the first memristor to switch both memristors back to an ON configuration.This behavior, along with the current of each memristor and total current of the circuit can be seen in figures 4.29 and 4.30.

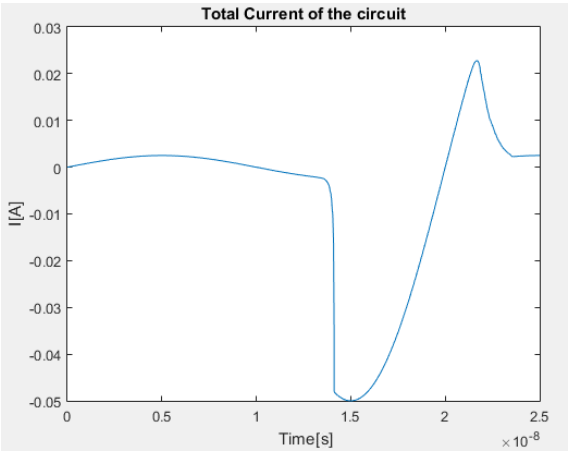


(a) Current of both memristors

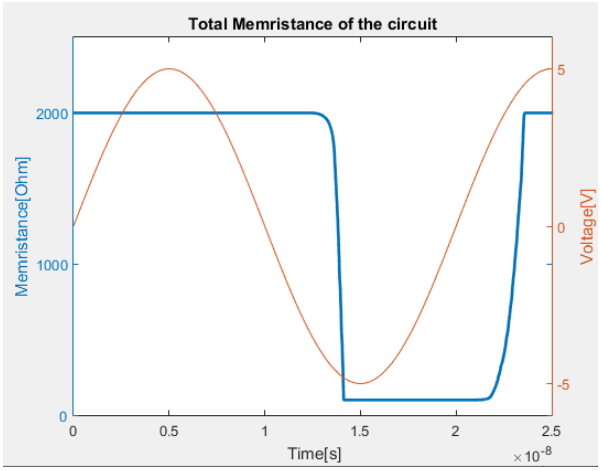


(b) Memristance of both memristors

Figure 4.29: Memristance and current for two memristors in series with direct polarity and OFF-OFF configuration



(a) Total current of the circuit



(b) Total memristance of the circuit

Figure 4.30: Total memristance and current for two memristors in series with direct polarity and OFF-OFF configuration

OFF-ON configuration

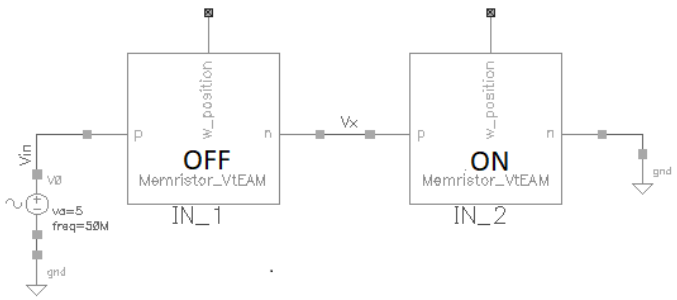
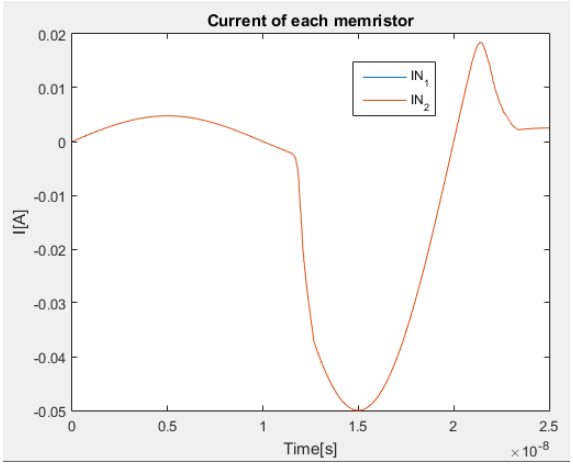
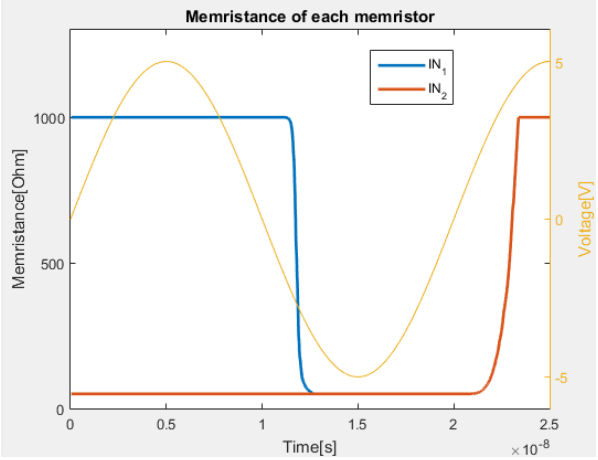


Figure 4.31: Circuit to test two memristors in series with direct polarity and OFF-ON configuration

When two memristors connected in series, both with direct polarity and,with different staring configurations, in this case, since R_{IN2} in at an ON configuration, the voltage divider will have a value of $0.047 \times Vo$. This means the voltage across the first memristor, R_{IN1}, will be almost the same as the voltage source, V_o. In this case when the applied voltage is lower than $-1.5V$, it will switch R_{IN1} to an ON configuration. After this, the voltage between both memristors becomes half of the applied voltage and only when $600mV$ are applied will both memristors switch to an OFF configuration. This behavior, along with the current of each memristor and total current of the circuit can be seen in figures 4.32 and 4.33.

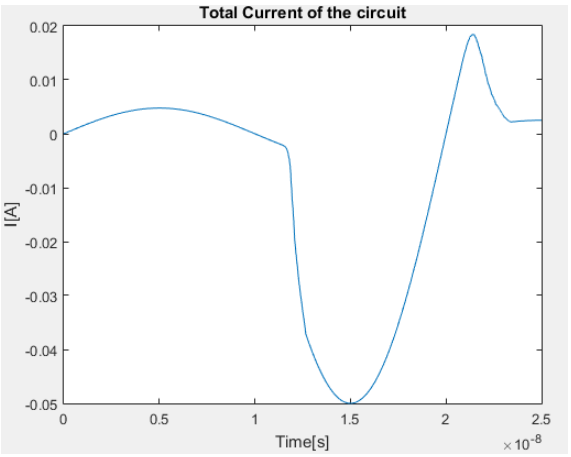


(a) Current of both memristors

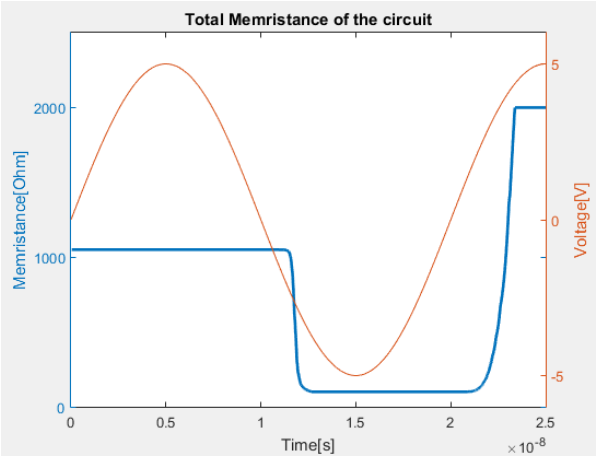


(b) Memristance of both memristors

Figure 4.32: Memristance and current for two memristors in series with direct polarity and OFF-ON configuration



(a) Total current of the circuit



(b) Total memristance of the circuit

Figure 4.33: Total memristance and current for two memristors in series with direct polarity and OFF-OFF configuration

ON-ON configuration

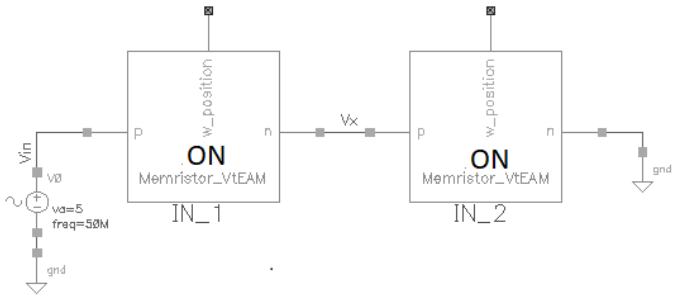
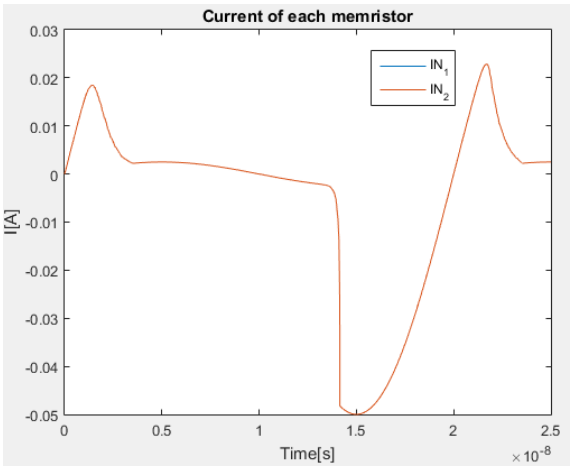
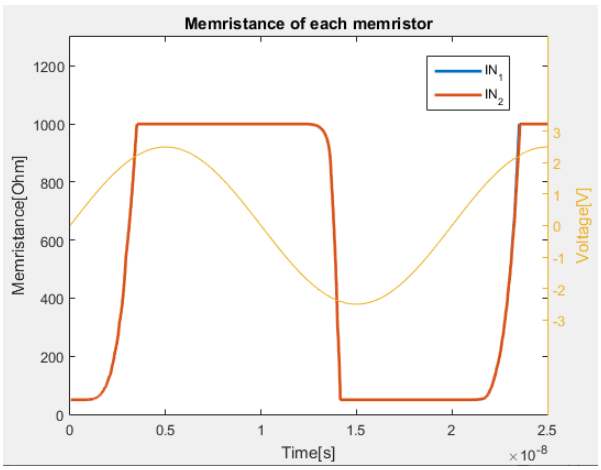


Figure 4.34: Circuit to test two memristors in series with direct polarity and ON-ON configuration

When two memristors connected in series, both with direct polarity and the same initial configuration of *ON*, same as in the *OFF-OFF* case the voltage between both memristors will be half of the applied voltage. This means the voltage necessary for them to switch configurations will be twice as its voltage thresholds, in this case $-3V$ and $600mV$. This behavior, along with the current of each memristor and total current of the circuit can be seen in figures 4.35 and 4.36.

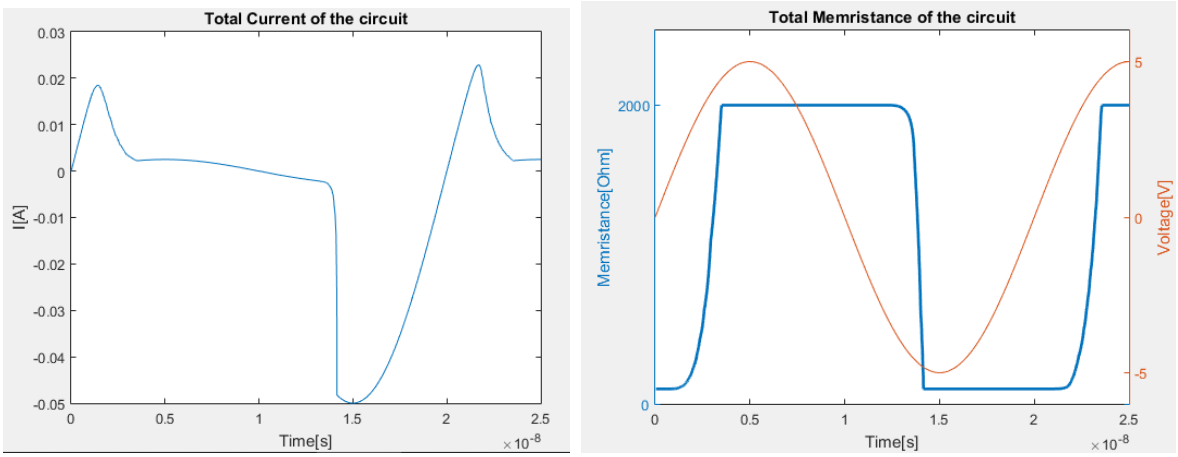


(a) Current of both memristors



(b) Memristance of both memristors

Figure 4.35: Memristance and current for two memristors in series with direct polarity and ON-ON configuration



(a) Total current of the circuit

(b) Total memristance of the circuit

Figure 4.36: Total memristance and current for two memristors in series with direct polarity and ON-ON configuration

Conclusions

Having tested all the configuration combinations possible, it can be concluded that, when two memristors are connected in series with direct polarity, if they have the same starting configuration, the circuit will behave as an memristor with double memristance, an double the thresholds VT_{OFF} and VT_{ON}. If they have different starting configurations, the voltage divider will cause one of them to switch configurations, and from there on both will have the same memristance.

4.1.2.2 Reverse Polarity

As it was seen in section 3.3.2 when a memristor is connected with reverse polarity, its thresholds can be seen as having changed to its symmetrical values. As such, to change a memristor from OFF to ON configuration it will require 1.5V and, to change it back to OFF configuration $-0.3V$ will be required.

OFF-OFF configuration

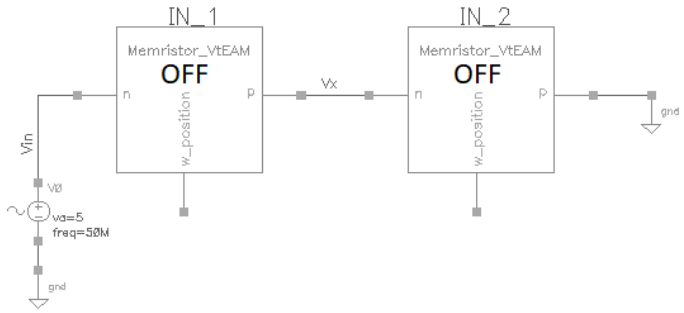
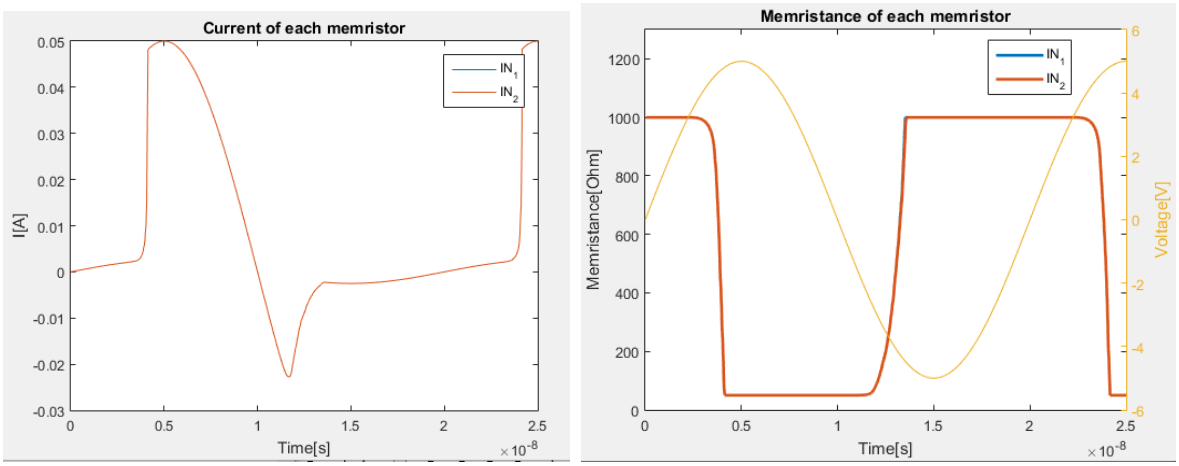


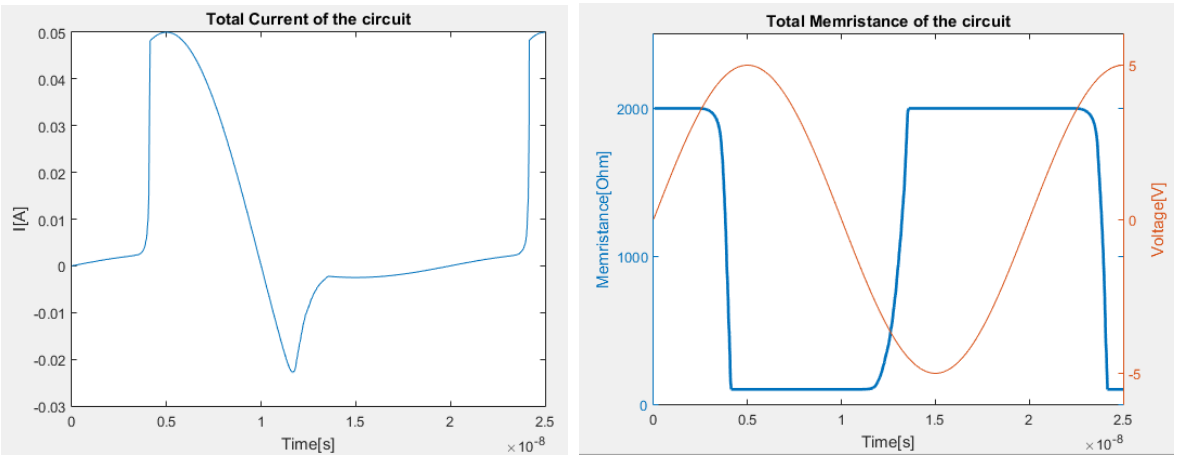
Figure 4.37: Circuit to test two memristors in series with reverse polarity and OFF-OFF configuration

This case is similar to the $OFF-OFF$ case in the direct polarity, since both memristors have the same starting configuration, the voltage between both memristors will be half of the applied voltage, V_O, this means twice as much as the voltage threshold will need to be applied to switch configurations, in this case it will be $3V$ to switch to an ON configuration, and $-600mv$to switch to an OFF configuration. This behavior, along with the current of each memristor and total current of the circuit can be seen in figures 4.38 and 4.39.



(a) Current of both memristors (b) Memristance of both memristors

Figure 4.38: Memristance and current for two memristors in series with reverse polarity and OFF-OFF configuration



(a) Total current of the circuit (b) Total memristance of the circuit

Figure 4.39: Total memristance and current for two memristors in series with reverse polarity and OFF-OFF configuration

OFF-ON configuration

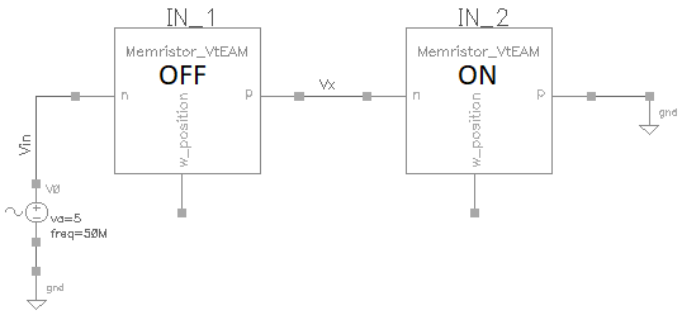
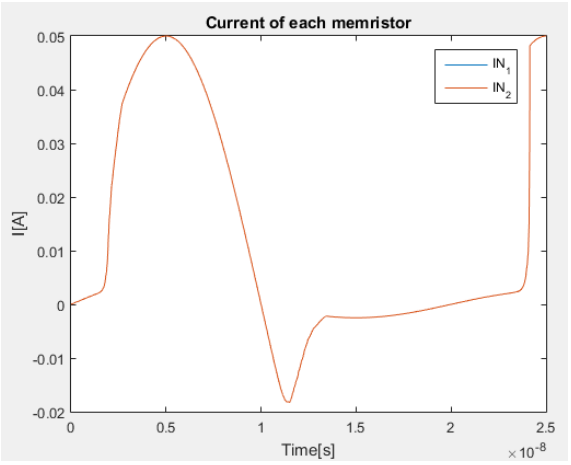
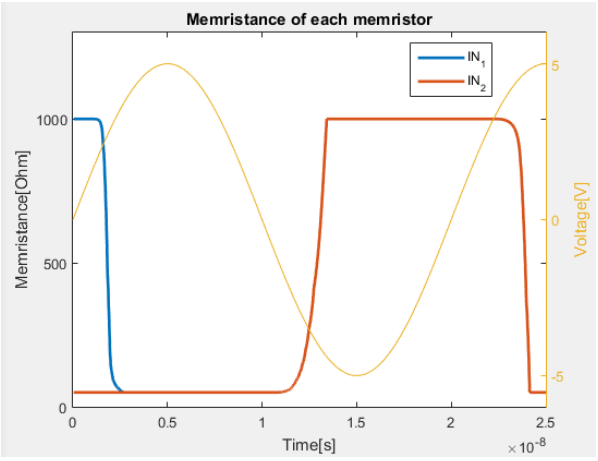


Figure 4.40: Circuit to test two memristors in series with reverse polarity and OFF-ON configuration

This case is similar to the $OFF-ON$ case in the direct polarity where, since the second memristor $R_{IN''}$ is started at an ON configuration, the voltage divider will have a value close o zero. This means the first memristor will switch to an on configuration on a voltage of $3V$ is applied. Tshen, like in the direct polarity case, since both memristors have the same memristance, the voltage in between both will be half and the voltage applied necessary to switch configurations will be twice as much. This behavior, along with the current of each memristor and total current of the circuit can be seen in figures 4.41 and 4.42.

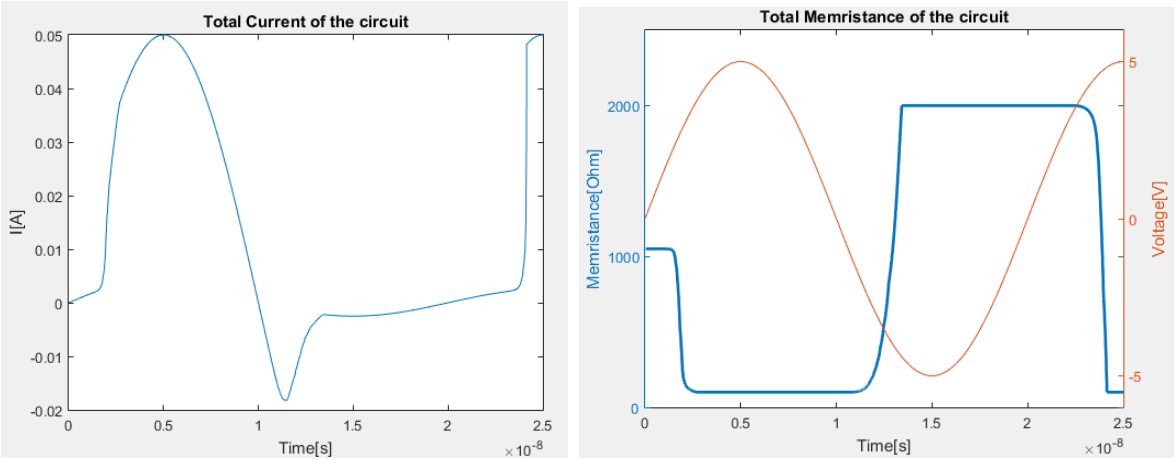


(a) Current of both memristors



(b) Memristance of both memristors

Figure 4.41: Memristance and current for two memristors in series with reverse polarity and OFF-ON configuration



(a) Total current of the circuit (b) Total memristance of the circuit

Figure 4.42: Total memristance and current for two memristors in series with reverse polarity and OFF-ON configuration

ON-ON configuration

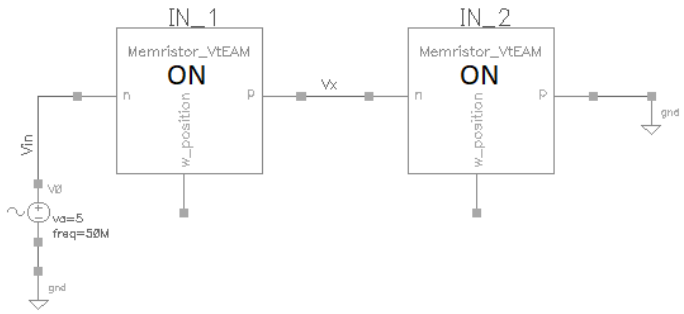
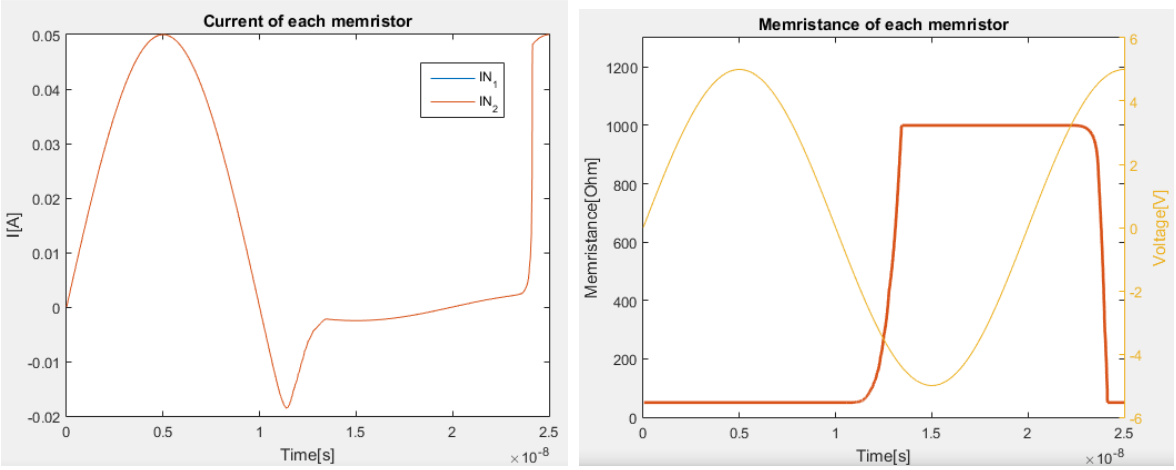


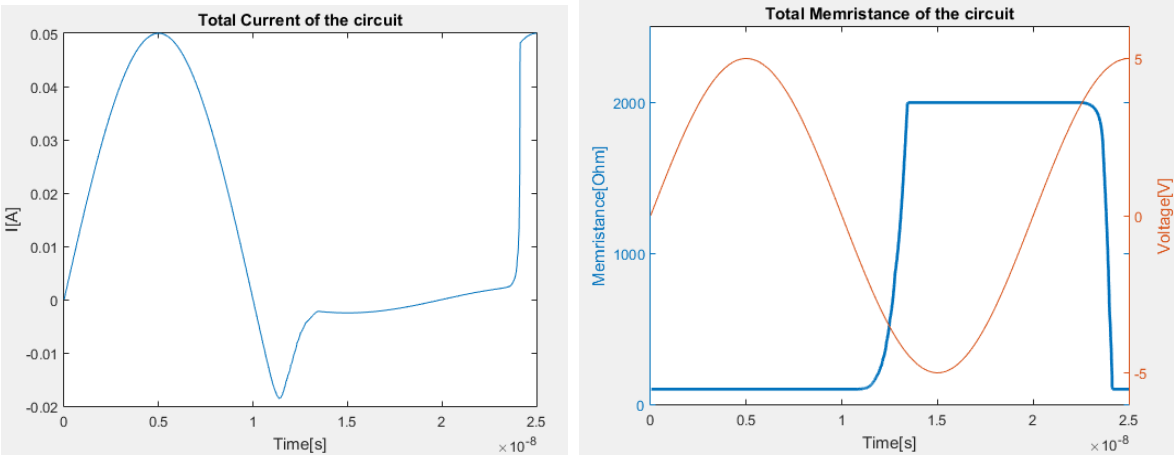
Figure 4.43: Circuit to test two memristors in series with reverse polarity and ON-ON configuration

This case is similar to the previous cases where both memristors have the same configuration, i.e., it will require twice as much applied voltage to switch configurations. This behavior, along with the current of each memristor and total current of the circuit can be seen in figures 4.44 and 4.45.



(a) Current of both memristors (b) Memristance of both memristors

Figure 4.44: Memristance and current for two memristors in series with reverse polarity and ON-ON configuration



(a) Total current of the circuit (b) Total memristance of the circuit

Figure 4.45: Total memristance and current for two memristors in series with reverse polarity and ON-ON configuration

Conclusions

Having tested with both direct and reverse polarity it can be concluded that, when two memristors in series, both with the same polarity, have different starting configurations, the voltage divider between them will decide which will switch configurations. After that since both have the same memristance, the circuit will behave like a single memristor with twice as much memristance and voltage thresholds as a single memristor.

4.1.2.3 Mixed Polarity

As it was concluded before, when two memristors with different thresholds are connected, it is expected that spikes occur. In this case, R_{IN1} will have voltage thresholds VT_{OFF} and

VT_{ON} of $-1.5V$ and $300mV$ respectively, and R_{IN2} will have voltage thresholds VT_{OFF} and VT_{ON} of $1.5V$ and $-300mV$ respectively.

OFF-OFF configuration

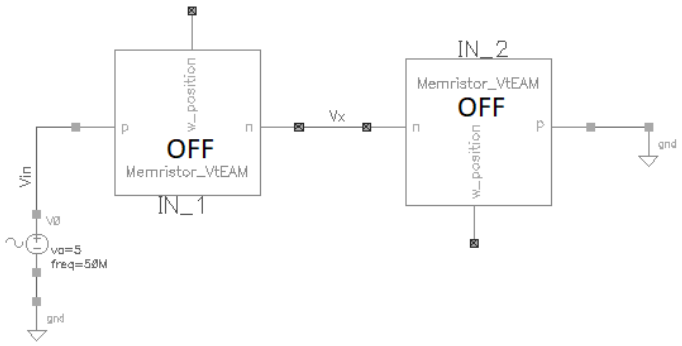
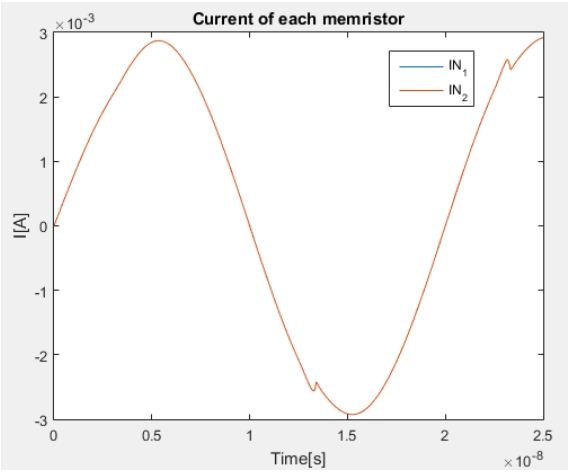
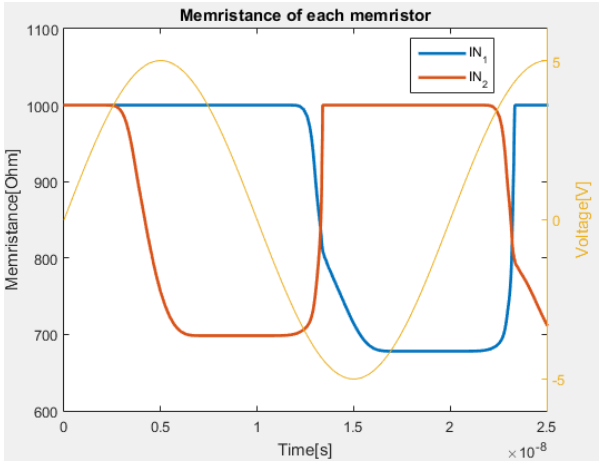


Figure 4.46: Circuit to test two memristors in series with mixed polarity and OFF-OFF configuration

When two memristors connected in series, with mixed polarity and the same initial configuration of OFF, the total memristance will be the sum of both, and the voltage in between both memristors will be half of the applied voltage, V_O, therefore when $3V$ are applied R_{IN2} will switch to an ON configuration, then, during the switching procedure, since the lower R_{IN2} will be the smaller voltage drop across it will be, meaning it wont stay at a value larger then its threshold long enough to completely change its configuration to ON. Then, during the negative sweep, the applied voltage will switch the memristor R_{IN1} to an ON configuration, and R_{IN2} back to an OFF configuration, which will cause the spike seen in the total memristance. This behavior, along with the current of each memristor and total current of the circuit can be seen in figures 4.47 and 4.48.

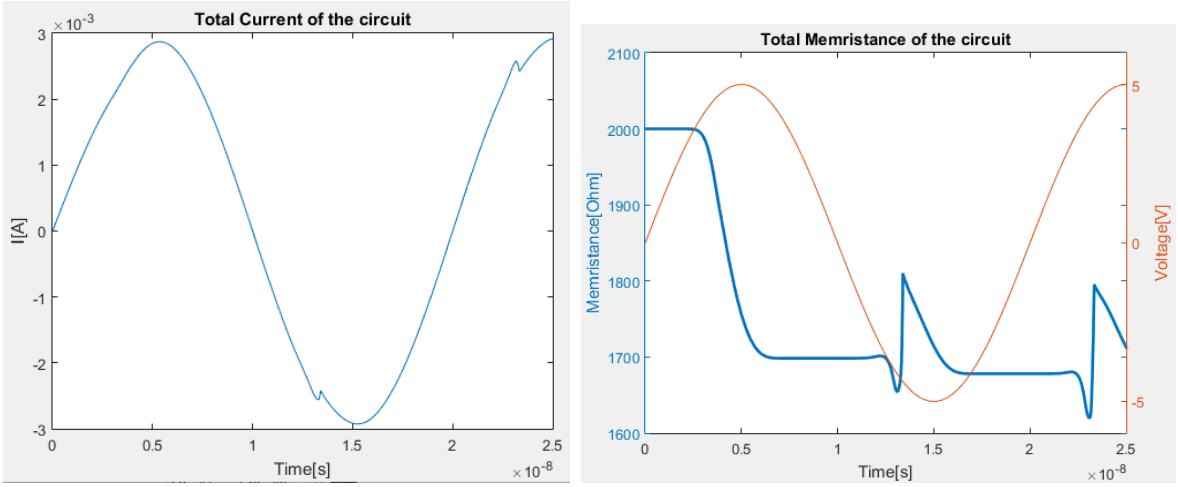


(a) Current of both memristors



(b) Memristance of both memristors

Figure 4.47: Memristance and current for two memristors in series with mixed polarity and OFF-OFF configuration



(a) Total current of the circuit

(b) Total memristance of the circuit

Figure 4.48: Total memristance and current for two memristors in series with mixed polarity and OFF-OFF configuration

OFF-ON configuration

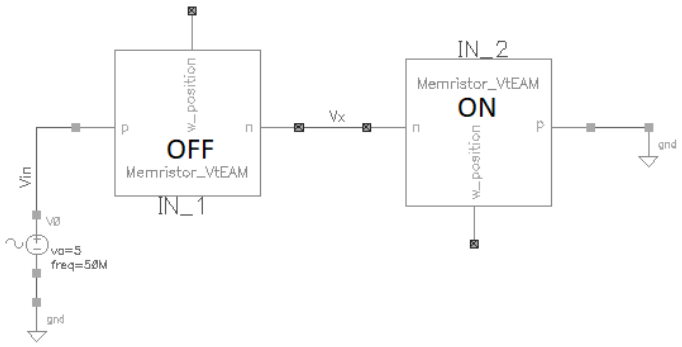
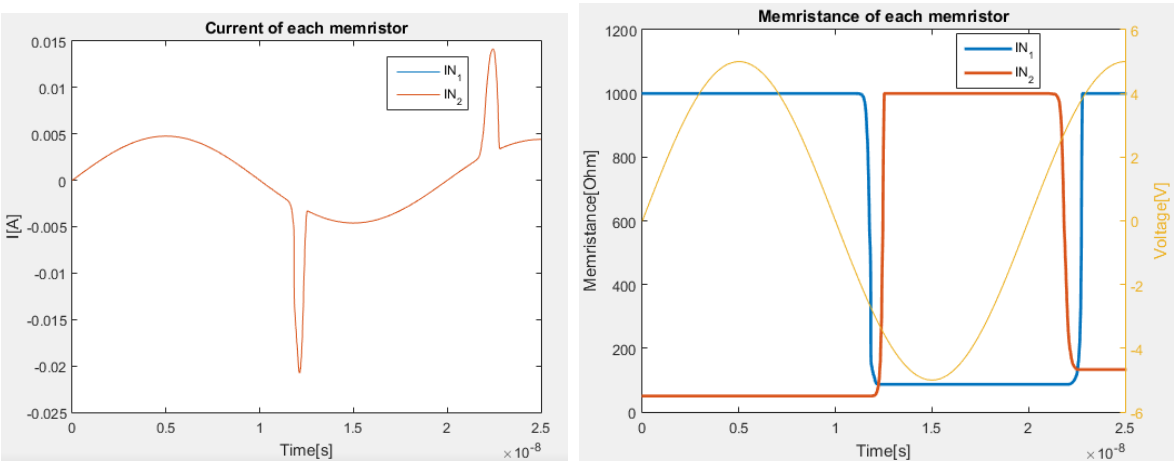


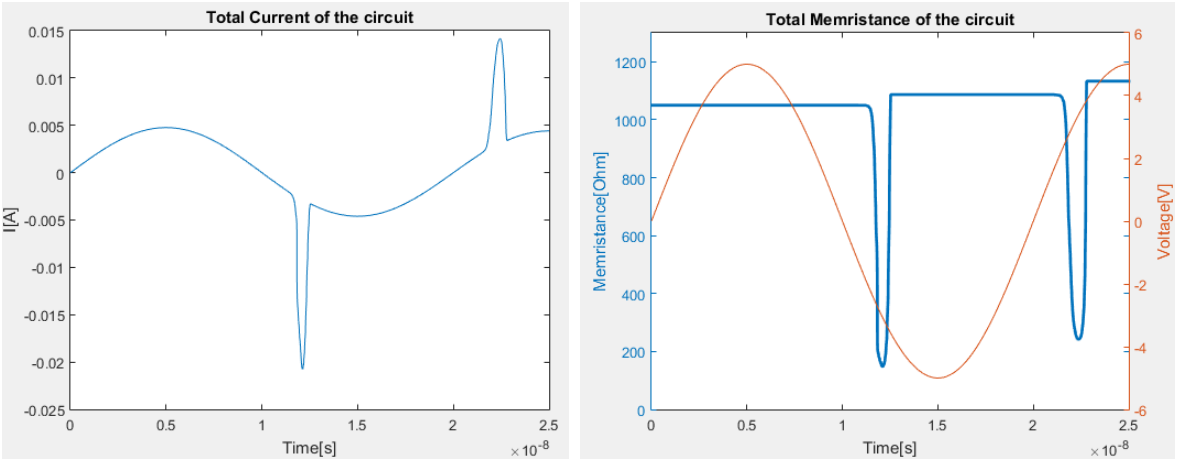
Figure 4.49: Circuit to test two memristors in series with mixed polarity and OFF-ON configuration

When the memristors are connected with a starting *OFF-ON* configuration, the voltage across the first memristor R_{IN1} will be much higher than in R_{IN2}, which means that, only when R_{IN1} switches to an *ON* configuration, will the voltage across R_{IN2} be high enough to switch its configuration to an *OFF* configuration. Then, since R_{IN2} will have a much higher memristance than R_{IN1}, the voltage across R_{IN1} will only be high enough to switch its configuration when R_{IN2} switches back to an *ON* configuration. This symmetrical behavior will give the total memristance of the circuit dips. This behavior, along with the current of each memristor and total current of the circuit can be seen in figures 4.50 and 4.51.



(a) Current of both memristors (b) Memristance of both memristors

Figure 4.50: Memristance and current for two memristors in series with mixed polarity and OFF-ON configuration



(a) Total current of the circuit (b) Total memristance of the circuit

Figure 4.51: Total memristance and current for two memristors in series with mixed polarity and OFF-ON configuration

ON-ON configuration

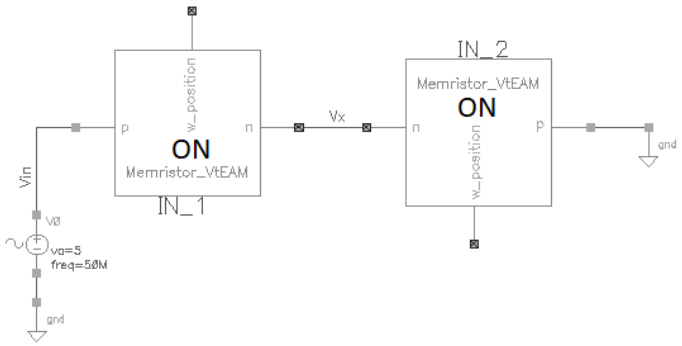
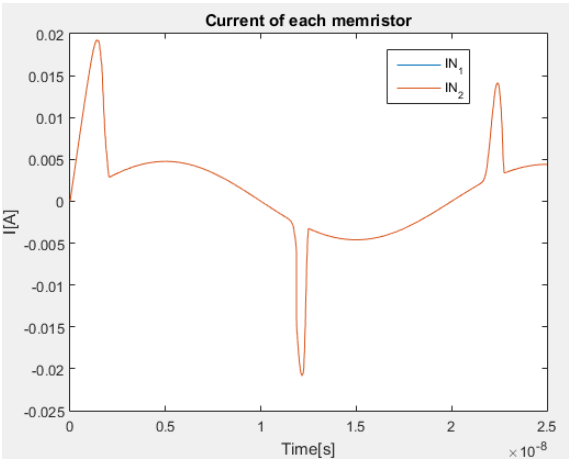
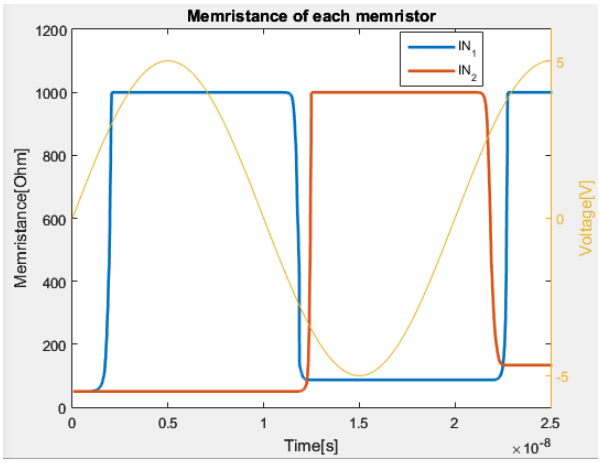


Figure 4.52: Circuit to test two memristors in series with mixed polarity and ON-ON configuration

When the memristors are connected with a starting *ON-ON* configuration, the positive voltage sweep will change R_{IN1} to an *OFF* configuration, after this, the circuit will shw the same behavior as in the *OFF-ON* configuration. This behavior, along with the current of each memristor and total current of the circuit can be seen in figures 4.53 and 4.54.

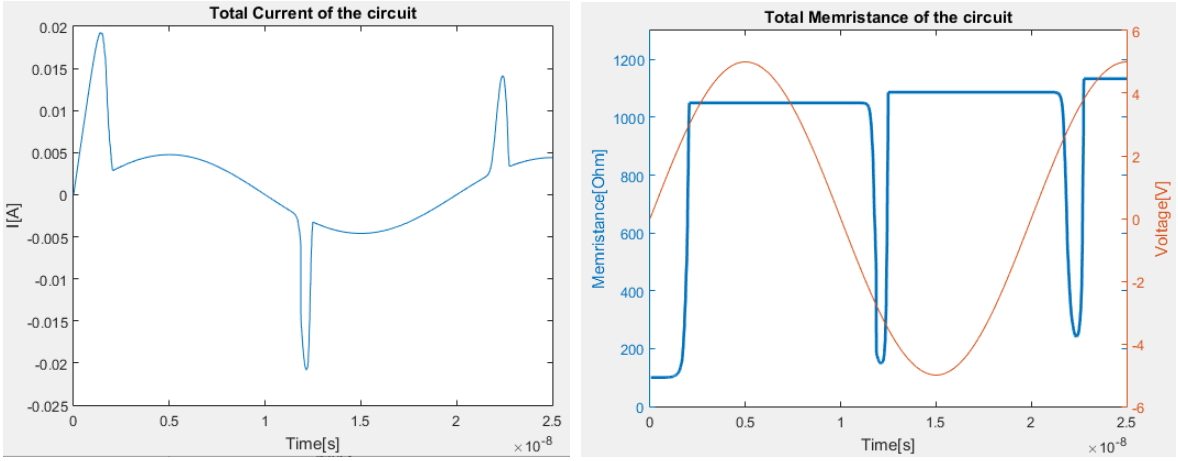


(a) Current of both memristors



(b) Memristance of both memristors

Figure 4.53: Memristance and current for two memristors in series with mixed polarity and ON-ON configuration



(a) Total current of the circuit

(b) Total memristance of the circuit

Figure 4.54: Total memristance and current for two memristors in series with mixed polarity and ON-ON configuration

Conclusions

Having tested two memristors with mixed polarity, it can be concluded that, if they have the same parameters and thresholds, it will result in a symmetrical behavior, which in return, will cause either spikes or dips in the circuits total memristance.

4.2 MAGIC logic gates

In [10] gates are described using MAGIC (Memristor Aided LoGIC). A MAGIC gate consists of two memristors, in a parallel or series configuration, which will act as inputs. Then a single output memristor connected in series which will save the final value of the logic operation. The logic state will be recorded as the high or low resistance of the output memristor, a high resistance will be the logical zero, and low resistance will be a logical one.

The initial stage of a MAGIC gate operation consists of, first, initializing the output to a known logical operator, using a high or low resistance, then a voltage is applied at the entrance of the of the gate, while this voltage is applied the resistance of the output memristor is going to depend on the logical inputs (resistance) of the two input memristors.

What happens then is, for different gates and inputs, the resulting voltage across the output memristor will not be enough to switch from the current logical output, i.e. the voltage will not be enough to switch configurations, in this case lower than $-1.5V$ to switch from 0 to 1, and $0.3V$ to switch from 1 to 0.

In figure 4.55 the voltage applied, V_O, can be seen. It should be noted that, the low voltage was set to a value close to 0, and, for certain gates the voltage needs to be increased.

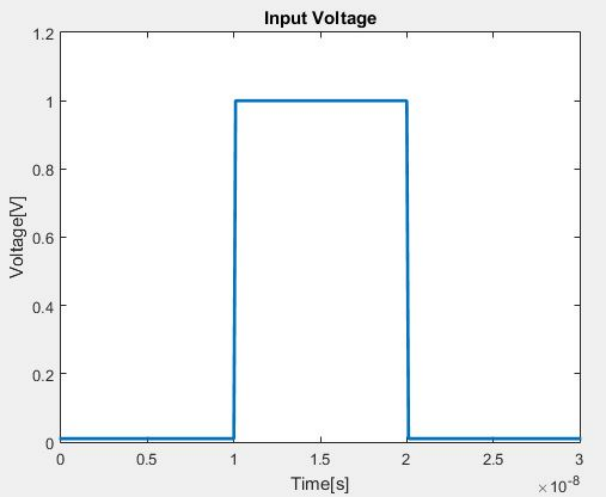


Figure 4.55: V_{in} of the logic gates

4.2.1 NOR gate

In this configuration two input memristor are connected in parallel, with a reverse polarity, in series to an output memristor with direct polarity as seen in figure 4.56 .

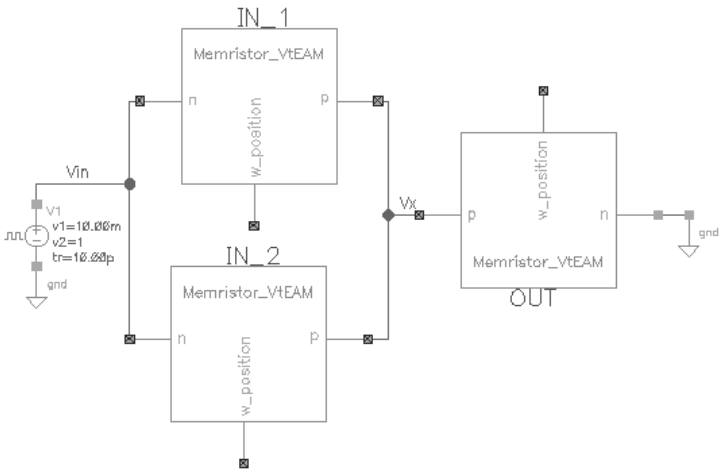


Figure 4.56: Circuit for the NOR gate

Firstly, a low resistance (logical 1) is written in the output memristor, then the input values are written in the input memristors, then a voltage V_{in} is applied. When both inputs are a logical 0 (high resistance) the voltage across the two input memristors is going to be much higher then the one across the output memristor, meaning the voltage

across it wont be high enough to switch to 0. When at least one of the input memristors is initialized at a logical 1(low resistance), the equivalent memristance between both input memristors will be small. Therefore, the voltage across the output memristor will be high enough to change its configuration to 0.

The figures 4.57 and4.58 and 4.59 and 4.60 show the behavior of the NOR logic gate.

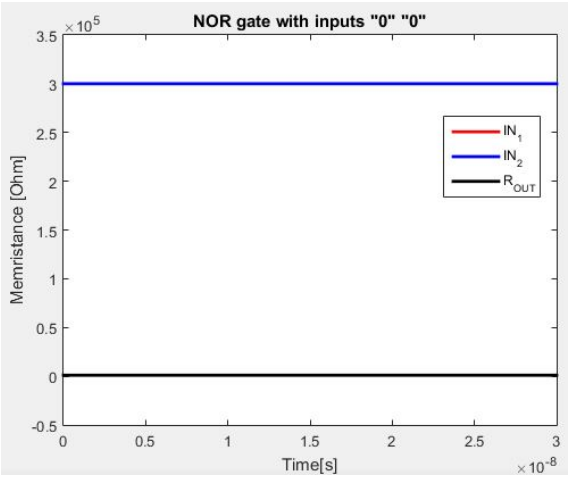


Figure 4.57: NOR gate with inputs"0" "0"

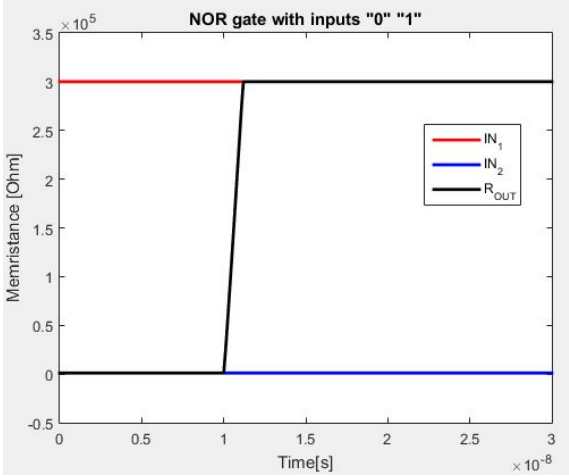


Figure 4.58: NOR gate with inputs"0" "1"

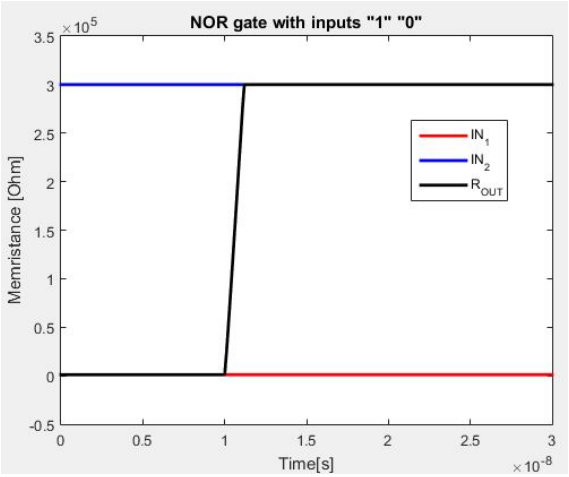


Figure 4.59: NOR gate with inputs"1" "0"

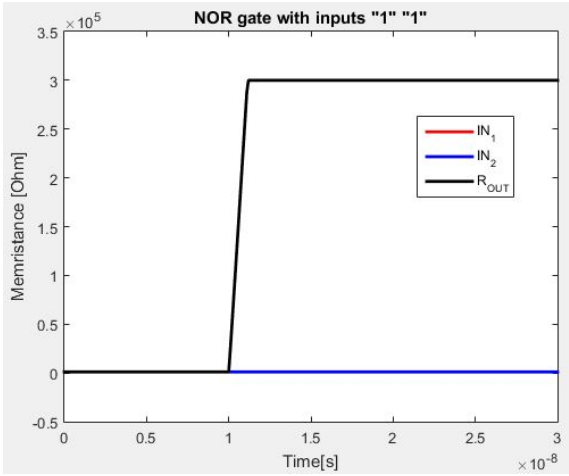


Figure 4.60: NOR gate with inputs"1" "1"

4.2.2 NAND gate

Connecting two input memristors in series with the same polarity with an output memristor in series with direct polarity results in a NAND gate, in this configuration the output memristor is initialized at 1, the corresponding circuit can be seen in figure 4.61.

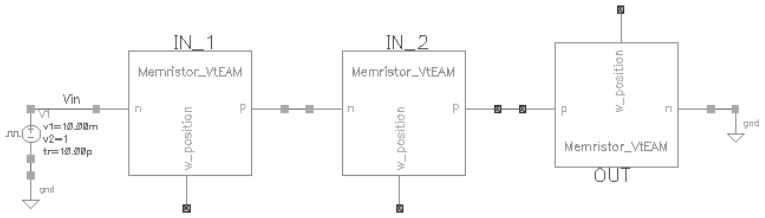


Figure 4.61: Circuit for the NAND gate

In this gate, only when both memristors have a low memristance (logical 1) will the voltage across the output memristor be high enough to change the configuration of the output memristor to 0. The figures 4.62 and4.63 and 4.64 and 4.65 show the behavior of the NAND logic gate.

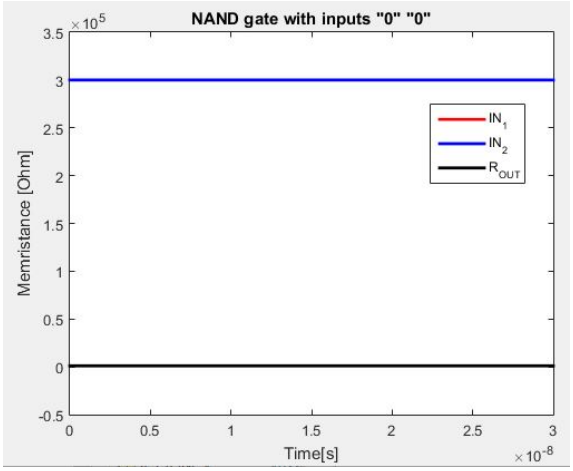


Figure 4.62: NAND gate with inputs"0" "0"

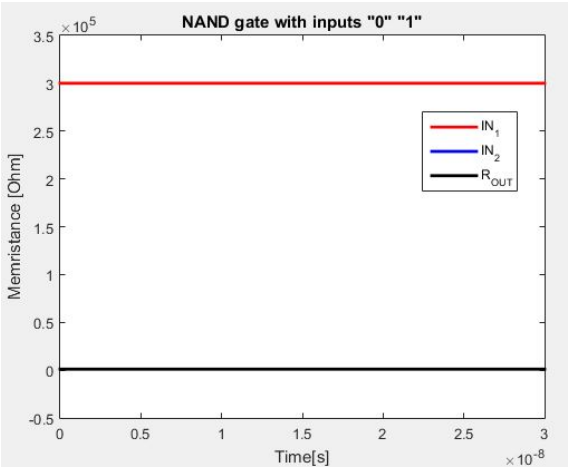


Figure 4.63: NAND gate with inputs"0" "1"

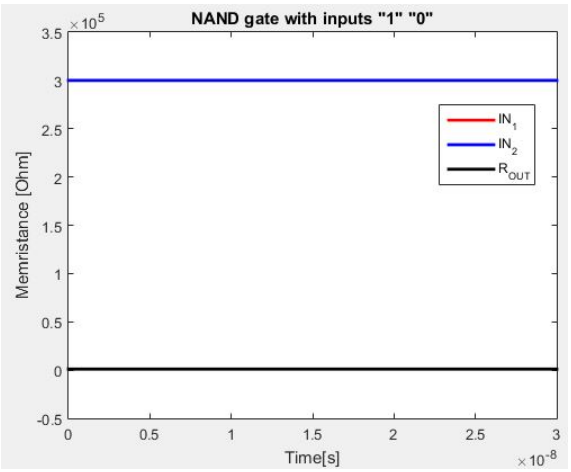


Figure 4.64: NAND gate with inputs”1” ”0”

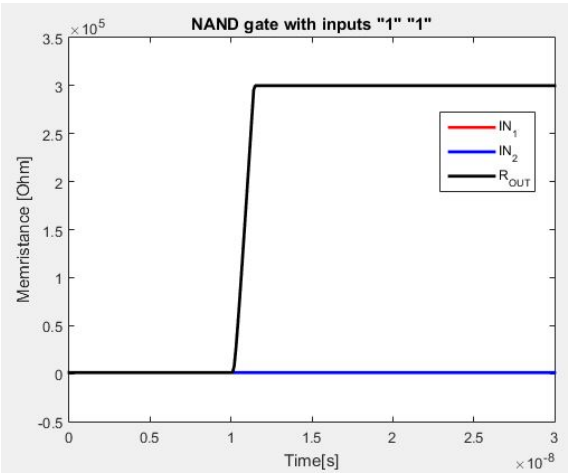


Figure 4.65: NAND gate with inputs”1” ”1”

4.2.3 OR Gate

The OR is similar to the NOR gate, the difference being the output memristor will have the same polarity as the input memristors and the initial value for the output memristor will be different, the initial value is 0, the circuit for the OR gate is shown in figure 4.66.

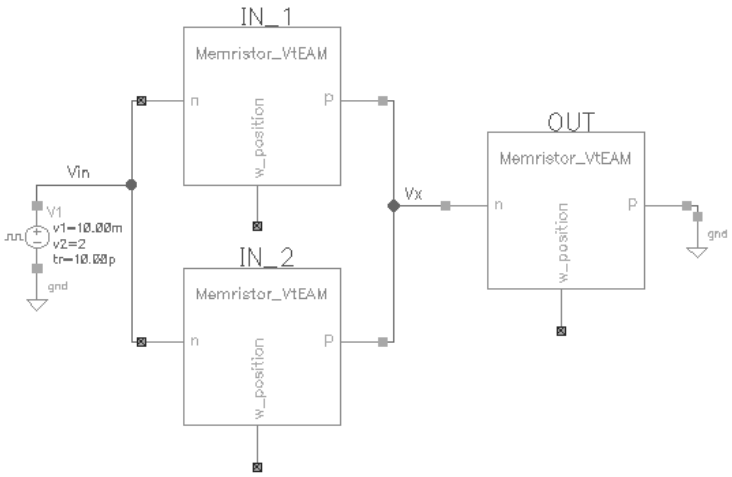


Figure 4.66: Circuit for the OR gate

In the OR gate, only when both the input memristors are ”0” will the voltage across the output memristor will not be enough to switch it to ”1”. The figures 4.67 and4.68 and 4.69 and 4.70 show the behavior of the OR logic gate.

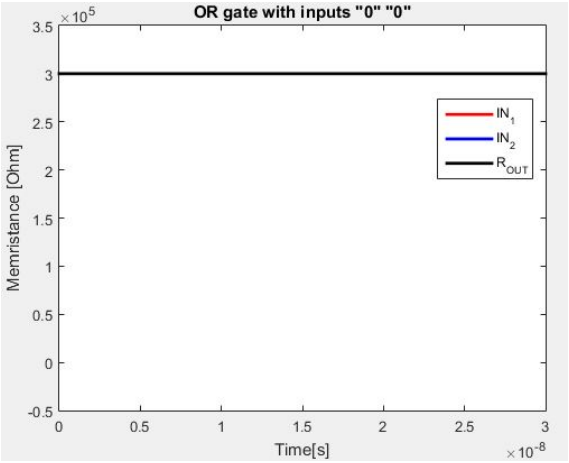


Figure 4.67: OR gate with inputs”0” ”0”

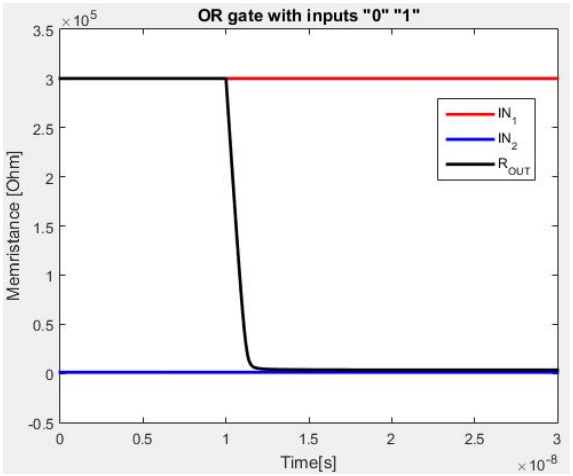


Figure 4.68: OR gate with inputs”0” ”1”

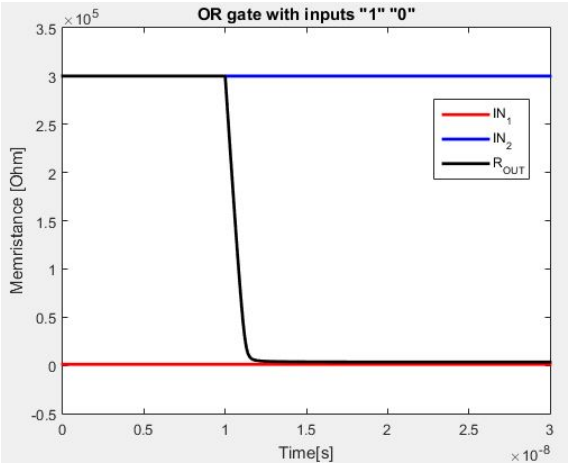


Figure 4.69: OR gate with inputs”1” ”0”

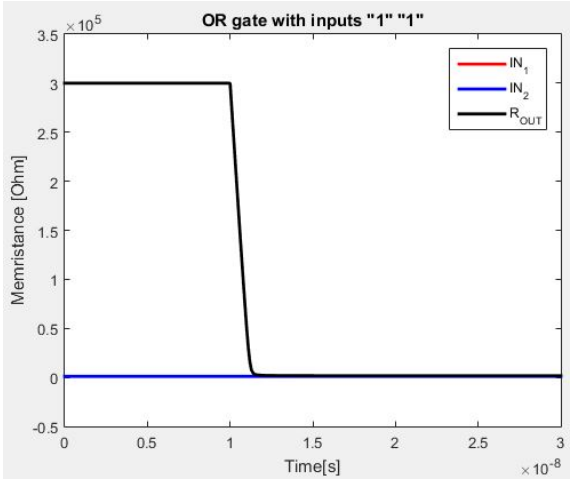


Figure 4.70: OR gate with inputs”1” ”1”

4.2.4 AND gate

The AND is similar to the NAND gate, the difference being the output memristor will have the same polarity as the input memristors and the initial value for the output memristor will be different, the initial value is 0, the circuit for the AND gate is shown in figure 4.71.

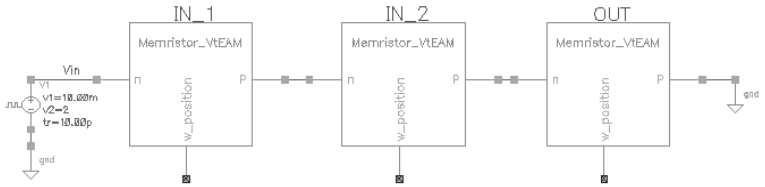


Figure 4.71: Circuit for the AND gate

In the AND gate both of the input memristos need to be at ”1” for the the voltage across the output memristor be enough to switch it to ”1”. The figures 4.72 and4.73 and

4.74 and 4.75 show the behavior of the AND logic gate.

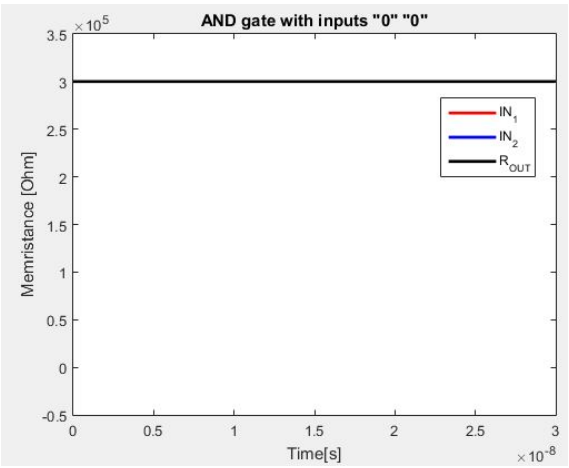


Figure 4.72: AND gate with inputs"0" "0"

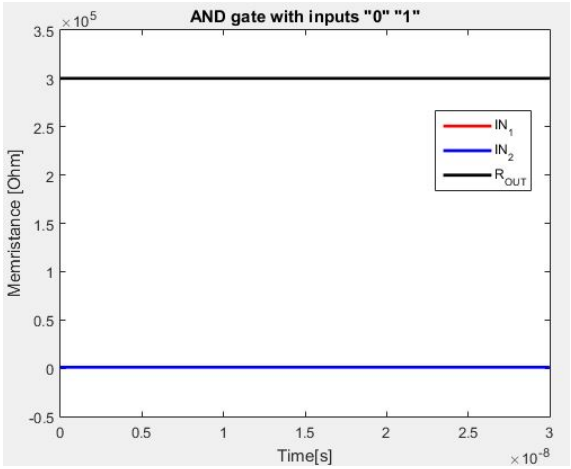


Figure 4.73: AND gate with inputs"0" "1"

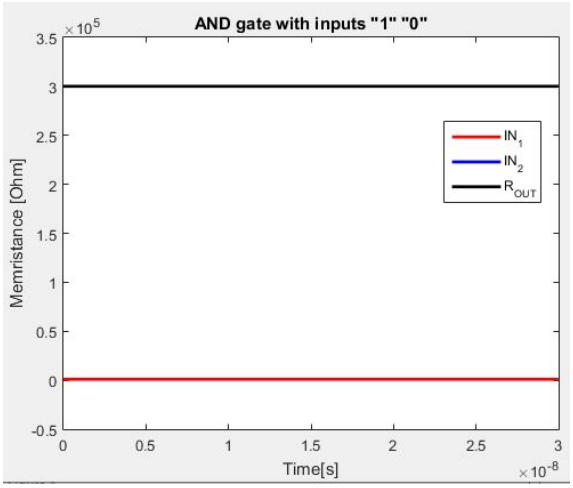


Figure 4.74: AND gate with inputs"1" "0"

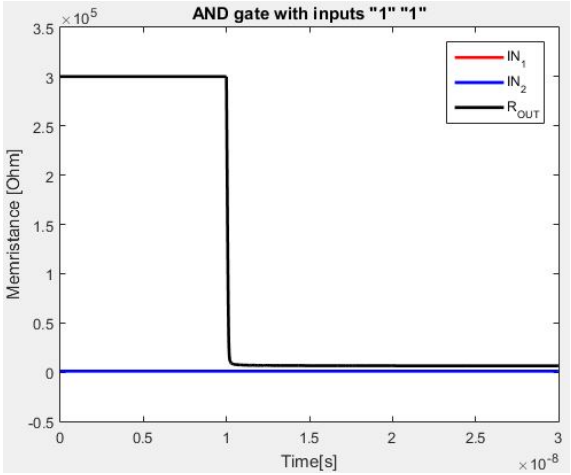


Figure 4.75: AND gate with inputs"1" "1"

4.2.5 NOT gate

The NOT gate consists of a single input memristor with reverse polarity connected in series to an output memristor with direct polarity, the initial value of the output memristor will be "1". When V_o is applied the voltage divider between both memristor will decide if the resistance of the output memristor switches configuration. Only when the input memristor is at "1" will the voltage across the output memristor will be high enough to switch it to "0". Due to convergence problems the input voltage was lowered to $0.95V$

The figure 4.76 shows the circuit for the NOT gate.

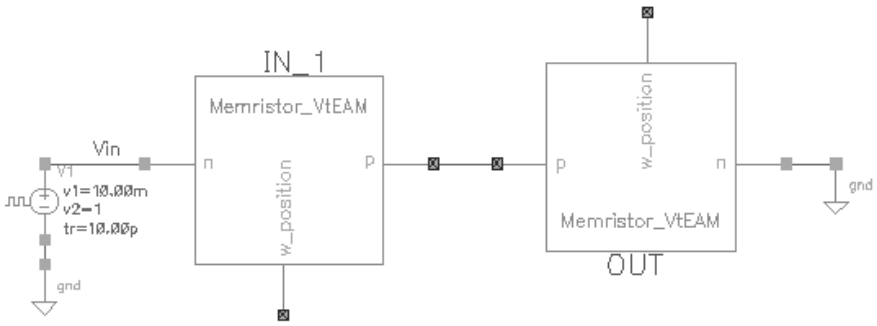


Figure 4.76: Circuit for the NOT gate

The figures 4.77 and4.78 show the behavior of the NOT logic gate.

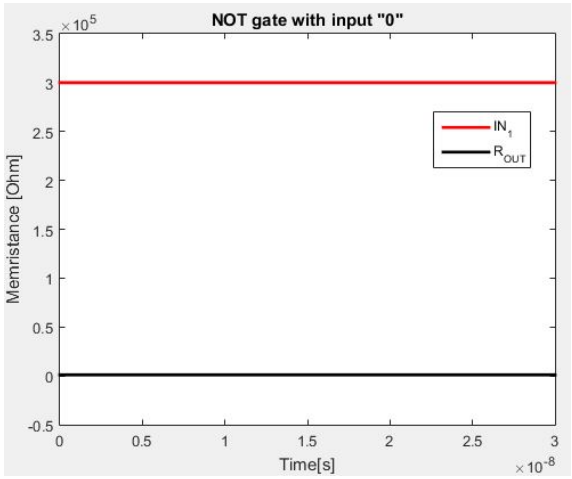


Figure 4.77: NOT gate with inputs"0"

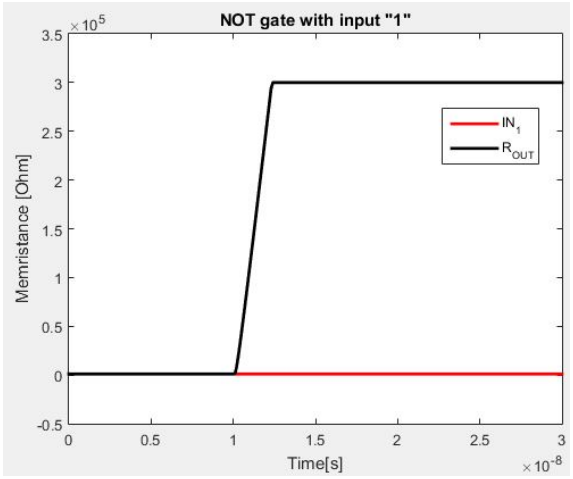


Figure 4.78: NOT gate with inputs "1"

4.3 Summary

In this chapter, more complex circuits using memristors were tested, first two memristors in parallel then in series, it was concluded that when both memristors are in parallel configuration, they behave simirlarly to two resistors, this means that when one is in a low memristance it will dominate the total memristor of the circuit. If they have the same polarity configuration they have equal memristance and the total memristance of the circuit will be half of a single memristor. If they have mixed polarity they will have a symmetrical behavior witch will result in spikes in the total memristance.

If two memristors are connected in series, the total memristance of the circuit will be the sum of each memristor, if both memristors have the same polarity, they will have the same voltage thresholds, meaning, for they to switch configurations, the applied voltage needs to be twice as much as the voltage needed to switch configurations of a single memristor. If they have mixed polarity, each memristor will have its own voltage thresholds which will result in spikes or dips in the total memristance.

Then the memristors were used to test logic gates, the ones tested were, the NAND, NOR, OR, AND and NOT gates. The logic of their behavior is, the two input memristors will define if the voltage across the output memristor will be enough to switch the configuration of the output memristor, for all the gates tested the results were acording to the expected.

Chapter 5

Conclusions

This chapter contains two sections, in the first the conclusions to this dissertation are presented then, future work that can be done to further study memristor based logic circuits.

5.1 Conclusions

In this dissertation a study was made on the various models of memristors and their use when used to build logic gates. Firstly, an introduction was made on how a memristor works and its main advantages and applications, such as compatibility with CMOS technology and use in signal processing.

Then a study on the various memristors models was made. Many different types of models exist, some with current, other with voltage as their control mechanism, some with higher accuracy in modelling the behaviour of the memristor, such as the Simmons models, are too complex. Other models such as the VTEAM and the Voukas model offer a simpler and more general model, but with lower accuracy.

Then a study was made to decide which logic and memristor model to use, the VTEAM model was chosen due to its implementation being in Verilog code, which offers more simulation tools the Spice code, also due to it being the only one that can be used to implement the logic chosen to build logic gates circuits. The logic chosen was the MAGIC logic, due to its simplicity and easier implementation.

Then, to better understand how a voltage controlled memristor, with voltage thresholds, behaves, circuits composed of two memristors were implemented and tested. Then finally the logic gates were implemented and tested, with the various starting configurations, with this it was proven that memristors can be used to build logic gates.

5.2 Future Work

Of all the different models tested to build memristor based logic gates, only the code developed in [10] was successfully used to build logic gates, in the future further work could be done in studying this circuits such as:

Further study of the verilog code: Although the logic gates built in chapter 4 were successfully tested, for certain parameters such as high values for OFF and ON resistances mistakes would occur in the simulations, further testing could be done to find the best values to test the model with, which would result in better simulation results.

Building different logic gates: In the tests run in chapter 4, all the memristors used had the same parameter values, experimenting with memristors with different values of voltage thresholds and different values of memristance, R_{OFF} and R_{ON}, could result in different circuits, such as more logic gates being discovered.

Build logic gates using SPICE codes: Using a SPICE code such as the Vorluka model, since it is voltage-controlled and it has voltage thresholds, it could be used with MAGIC logic described in section 4.2 to build logic gates in a different environment like LTSPICE, the results could then be compared with the verilog code to see which one would be the preferred to use.

Bibliography

[1] W. A. A. Adzmi A. Nasrudin. "Memristor Spice Model for Designing Analog Circuit." In: *IEEE Student COnference on Reserch and Development*. 2012, pp. 78–83.

[2] L. O. Chua. "Memristor-The Missing Circuit Element." In: *IEEE Transactions on circuit theory* CT-18.5 (Sept. 1971), pp. 507–519.

[3] D. R. S. R. S. W. Dmitri B. Strukov Gregory S. Snider. "The missing memristor found." In: *nature* 453 (May 2008), pp. 80–83.

[4] M. D. P. Hisham Abdalla. "SPICE Modeling of Memristors." In: *IEEE* (2011), pp. 1832–1835.

[5] G. C. S. Ioannis Vourkas. "A Novel Design and Modeling Paradigm for Memristor-Based Crossbar Circuits." In: *IEEE Transactions on Nanotechnology* 11.6 (Nov. 2012), pp. 1151–1159.

[6] G. C. S. Ioannis Vourkas. "Memristor-based Nanoelectronic Computing Circuits And Architectures." In: *IEEE transactions on very large scale integration* 22.10 (Oct. 2014).

[7] V. Keshmiri. "A Study of the Memristor Models and Applications." Master's thesis. Linköping University, The Institute of Technology., 2014.

[8] S. Kvatinsky, M. Ramadan, E. G. Friedman, and A. Kolodny. "VTEAM A General Model for Voltage Controlled Memristors." In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 62 (Aug. 2015), pp. 786 –790.

[9] S. e. a. Kvatinsky. "Verilog-A for Memristor Models." In: *Journal of Computational Electronics* 18 (2019).

[10] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser. "MAGIC – Memristor Aided LoGIC." In: *IEEE* 61.11 (Nov. 2014), pp. 895 –899.

[11] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C.Weiser. "Memristor-Based Material Implication (IMPLY) Logic: Design Principles and Methodologies." In: *IEEE* 22.10 (Oct. 2014).

[12] E. Lehtonen and M. Laiho. "Stateful Implication Logic with Memristors." In: *IEEE* (2009).

[13] S.-M. K. Sangho Shin Kyungmin Kim. "Compact Models for Memristors Based on Charge–Flux Constitutive Relationships." In: *IEEE* 29.4 (Apr. 2010), pp. 590–598.

[14] S. E.-R. Sherif F. Nafea Ahmed A.S. Dessouki. "Memristor Overview up to 2015." In: *ReserchGate* (July 2015).

[15] C. P. Themis Prodromakis Christofer Toumazou. "A Versatile Memristor Model With Nonlinear Dopant Kinetics." In: *IEEE Transactions on Electron Devices* (Sept. 2011).

[16] S. K. E. F. A. K. U. Weiser. "TEAM: ThrEshold Adaptive Memristor Model." In: *IEEE Transactions on Circuits and Systems* ().

[17] S. J. W. Yogesh N. Joglekar. "The elusive memristor: properties of basic electrical circuits." In: *European Journal of Physics* 30 (Jan. 2009), pp. 661–675.

[18] V. B. Zdeněk Biolek Dalibor Biolek. "SPICE Model of Memristor with Nonlinear Dopant Drift." In: *Radioengeneering* 18.2 (June 2009), pp. 210–214.

ANNEX I

ANNEX 1

Listing I.1: HP memristor code

```
* HP Memristor SPICE Model
* For Transient Analysis only
* created by Zdenek and Dalibor Biolek
**************************
* Ron, Roff - Resistance in ON / OFF States
* Rinit - Resistance at T=0
* D - Width of the thin film
* uv - Migration coefficient
* p - Parameter of the WINDOW-function
* for modeling nonlinear boundary conditions
* x - W/D Ratio, W is the actual width
* of the doped area (from 0 to D)
*
.SUBCKT memristor Plus Minus PARAMS:
+ Ron=1K Roff=100K Rinit=80K D=10N uv=10F p=1
*********************************************
* DIFFERENTIAL EQUATION MODELING *
*********************************************
Gx 0 x value={ I(Emem)*uv*Ron/D^2*f(V(x),p)}
Cx x 0 1 IC={(Roff-Rinit)/(Roff-Ron)}
Raux x 0 1T
* RESISTIVE PORT OF THE MEMRISTOR *
*****************************
Emem plus aux value={-I(Emem)*V(x)*(Roff-Ron)}
Roff aux minus {Roff}
*********************************************
*Flux computation*
*********************************************
Eflux flux 0 value={SDT(V(plus,minus))}
```

```
30 ********************************************
31 *Charge computation*
32 ********************************************
33 Echarge charge 0 value={SDT(I(Emem))}
34 ********************************************
35 * WINDOW FUNCTIONS
36 * FOR NONLINEAR DRIFT MODELING *
37 ********************************************
38 *window function, according to Joglekar
39 .func f(x,p)={1-(2*x-1)^(2*p)}
40 *proposed window function
41 ;.func f(x,i,p)={1-(x-stp(-i))^(2*p)}
42 .ENDS memristor
```

Listing I.2: Simmons memristor code

```
1  +phio=0.95 Lm=0.0998 w1=0.1261 foff=3.5e-6
2  +ioff=115e-6 aoff=1.2 fon=40e-6 ion=8.9e-6
3  +aon=1.8 b=500e-6 wc=107e-3
4  G1 plus internal value={sgn(V(x))*(1/V(dw))^2*0.0617*(V(phiI)
5  *exp(-V(B)*V(sr))-(V(phiI)+abs(V(x)))*exp(-V(B)*V(sr2)))}
6  Esr sr 0 value={sqrt(V(phiI))}
7  Esr2 sr2 0 value={sqrt(V(phiI)+abs(V(x)))}
8  Rs internal minus 215
9  Eg x 0 value={V(plus)-V(internal)}
10 Elamda Lmda 0 value={Lm/V(w)}
11 Ew2 w2 0 value={w1+V(w)-(0.9183/(2.85+4*V(Lmda)-2*abs(V(x))))}
12 EDw dw 0 value={V(w2)-w1}
13 EB B 0 value={10.246*V(dw)}
14 ER R 0 value={(V(w2)/w1)*(V(w)-w1)/(V(w)-V(w2))}
15 EphiI phiI 0 value={phio-abs(V(x))*((w1+V(w2))/(2*V(w)))
16 -1.15*V(Lmda)*V(w)*log(V(R))/V(dw)}
17 C1 w 0 1e-9 IC=1.2
18 R w 0 1e8MEG
19 Ec c 0 value={abs(V(internal)-V(minus))/215}
20 Emon1 mon1 0 value={((V(w)-aoff)/wc)-(V(c)/b)}
21 Emon2 mon2 0 value={(aon-V(w))/wc-(V(c)/b)}
22 Goff 0 w value={foff*sinh(stp(V(x))*V(c)/ioff)*exp(-exp(V(mon1))-V(w)/wc)}
23 Gon w 0 value={fon*sinh(stp(-V(x))*V(c)/ion)*exp(-exp(V(mon2))-V(w)/wc)}
24 .ENDS modelmemristor
```

Listing I.3: Vourkas memristor code

```
1 .subckt Memristor_Vorluka plus minus PARAMS:
2 *Parameters's␣vlaues
3 +rmin=100␣rmax=390␣rinit=100␣alpha=5E3␣beta=0␣gamma=0.1␣VtR=1␣VtL=-1␣yo=0.0001
4 +m=82␣fo=310␣Lo=5
5 Gr␣0␣r␣value={dr_dt(V(plus)-V(minus))*(st_f(V(plus)-V(minus))*st_f(V(r)-rmin)+
```

```
+st_f(-(V(plus)-V(minus)))*st_f(rmax-V(r)))}
Cr r 0 1 IC={rinit}
Raux r 0 1E12
*Current equation Imem=V/R(L)
Gpm plus minus value={(V(plus)-V(minus))/((fo*exp(2*L(V(r))))/L(V(r)))}
*Func. for non-linear threshold-based behavior
.func dr_dt(y)={-alpha*((y-VtL)/(gamma+abs(y-VtL)))*st_f(-y+VtL)
-beta*y*st_f(y-VtL)*st_f(-y+VtR)-alpha*((y-VtR)/(gamma+abs(y-VtR)))*st_f(y-VtR)}
*Smoothing function
.func st_f(y)={1/(exp(-y/yo)+1)}
*L(V) function
.func L(y)={Lo-Lo*m/y}
.ends Memristor_Vorluka
```

Listing I.4: Verilog memristor code

```
////////////////////////////////////////////////
// VerilogA model for memristor
//
// kerentalis@gmail.com
// Dimafliter@gmail.com
// shahar@ee.technion.ac.il
// rmisbah@tx.technion.ac.il
//
// Technion - Israel Institute of Technology
// EE Dept. December 2015
//
////////////////////////////////////////////////

`include "disciplines.vams"
`include "constants.h"

// define meter units for w parameter
nature distance
  access = Metr;
  units = "m";
  abstol = 0.01n;
endnature

discipline Distance
  potential distance;
enddiscipline

module Memristor(p, n,w_position);
  input p;//positive pin
  output n;//negative pin
  output w_position;// w-width pin

  electrical p, n,gnd;
```

```
34   Distance w_position;
35   ground gnd;
36
37    parameter real model = 0;
38 // define the model:
39 // 0 - Linear Ion Drift;
40 // 1 - Simmons Tunnel Barrier;
41 // 2 - Team model;
42 // 3 - Nonlinear Ion Drift model
43 // 4 - Vteam model;
44
45    parameter real window_type=0;
46 // define the window type:
47 // 0 - No window;
48 // 1 - Jogelkar window;
49 // 2 - Biolek window;
50 // 3 - Prodromakis window;
51 // 4 - Kvatinsky window (Team model only)
52 // 5 - Kvatinsky window2 (Vteam model only)
53
54    parameter real dt=0;
55 // user must specify dt same as max step size in
56 // transient analysis & must be at least 3 orders
57 //smaller than T period of the source
58
59    parameter real init_state=0.5;
60 // the initial state condition [0:1]
61
62
63 ////////////// Linear Ion Drift model ///////////////
64
65  //parameters definitions and default values
66     parameter real Roff = 200000;
67     parameter real Ron = 100;
68     parameter real D = 3n;
69     parameter real uv = 1e-15;
70     parameter real w_multiplied = 1e8;
71 // transformation factor for w/X width
72 // in meter units
73     parameter real p_coeff = 2;
74 // Windowing function coefficient
75
76 parameter real J = 1;
77 // for prodromakis Window function
78
79
80 parameter real p_window_noise=1e-18;
81 // provoke the w width not to get stuck at
82 // 0 or D with p window
83
```

```
84  parameter real threshhold_voltage=0;
85
86      // local variables
87      real w;
88      real dwdt;
89      real w_last;
90      real R;
91      real sign_multply;
92      real stp_multply;
93      real first_iteration;
94
95  ////////// Simmons Tunnel Barrier model //////////
96
97     //parameters definitions and default values
98  //for Simmons Tunnel Barrier model
99      parameter real c_off = 3.5e-6;
100     parameter real c_on = 40e-6;
101     parameter real i_off = 115e-6;
102     parameter real i_on = -8.9e-6;
103     parameter real x_c = 107e-12;
104     parameter real b = 500e-6;
105     parameter real a_on = 2e-9;
106     parameter real a_off = 1.2e-9;
107
108     // local variables
109     real x;
110     real dxdt;
111     real x_last;
112
113 /////////////////TEAM model////////////////////
114
115     parameter real K_on=-8e-13;
116     parameter real K_off=8e-13;
117     parameter real Alpha_on=3;
118     parameter real Alpha_off=3;
119     parameter real v_on=-1.78;
120     parameter real v_off=0.0115;
121     parameter real IV_relation=0;
122 // IV_relation=0 means linear V=IR.
123 // IV_relation=1 means nonlinear V=I*exp{..}
124     parameter real x_on=0;
125     parameter real x_off=3e-09; // equals D
126
127 // local variables
128     real lambda;
129
130 ////////////Nonlinear Ion Drift model ////////////
131
132     parameter real alpha = 2;
133     parameter real beta  = 9;
```

```
134     parameter real c     = 0.01;
135     parameter real g     = 4;
136     parameter real N     = 14;
137     parameter real q     = 13;
138     parameter real a     = 4;
139
140  analog function integer sign;
141 //Sign function for Constant edge cases
142      real arg; input arg;
143      sign = (arg >= 0 ? 1 : -1 );
144  endfunction
145
146  analog function integer stp;        //Stp function
147      real arg; input arg;
148      stp = (arg >= 0 ? 1 : 0 );
149  endfunction
150
151
152 ////////////////////// MAIN ////////////////////////
153
154  analog begin
155
156     if(first_iteration==0) begin
157         w_last=init_state*D;
158 // if this is the first iteration,
159 //start with w_init
160         x_last=init_state*D;
161 // if this is the first iteration,
162 // start with x_init
163     end
164
165 //////////////Linear Ion Drift model //////////////
166
167 if (model==0) begin // Linear Ion Drift model
168     dwdt =(uv*Ron/D)*I(p,n);
169      //change the w width only if the
170  // threshhold_voltage permits!
171        if(abs(I(p,n))<threshhold_voltage/R) begin
172             w=w_last;
173         dwdt=0;
174        end
175
176 // No window
177     if ((window_type==0)|| (window_type==4)) begin
178         w=dwdt*dt+w_last;
179     end // No window
180
181 // Jogelkar window
182      if (window_type==1) begin
183         if (sign(I(p,n))==1) begin
```

```
184             sign_multply=0;
185             if(w<p_window_noise) begin
186             sign_multply=1;
187             end
188         end
189         if (sign(I(p,n))==-1) begin
190                 sign_multply=0;
191                 if(w>(D-p_window_noise)) begin
192                     sign_multply=-1;
193                 end
194         end
195         w=dwdt*dt*(1-pow(pow(2*w/D-1,2),p_coeff))+w_last+sign_multply
196         *p_window_noise;
197     end // Jogelkar window
198 
199 // Biolek window
200     if (window_type==2) begin
201 
202         if (stp(-I(p,n))==1) begin
203             stp_multply=1;
204         end
205         if (stp(-I(p,n))==0) begin
206                 stp_multply=0;
207                 end
208         w=dwdt*dt*(1-pow(pow(w/D-stp_multply,2),p_coeff))+w_last;
209     end // Biolek window
210 
211   // Prodromakis window
212     if (window_type==3) begin
213         if (sign(I(p,n))==1) begin
214             sign_multply=0;
215             if(w<p_window_noise) begin
216             sign_multply=1;
217             end
218         end
219         if (sign(I(p,n))==-1) begin
220                 sign_multply=0;
221                 if(w>(D-p_window_noise)) begin
222                     sign_multply=-1;
223                     end
224         end
225         w=dwdt*dt*J*(1-pow(pow(w/D-0.5,2)+0.75,p_coeff))
226         +w_last+sign_multply*p_window_noise;
227     end  // Prodromakis window
228     if (w>=D) begin
229         w=D;
230                 dwdt=0;
231     end
232     if (w<=0) begin
233         w=0;
```

```
234                 dwdt=0;
235     end
236        //update the output ports(pins)
237        R=Ron*w/D+Roff*(1-w/D);
238        w_last=w;
239        Metr(w_position) <+ w*w_multiplied;
240        V(p,n) <+ (Ron*w/D+Roff*(1-w/D))*I(p,n);
241        first_iteration=1;
242
243 end // end Linear Ion Drift model
244
245 ///////// Simmons Tunnel Barrier model ///////////
246
247
248 if (model==1) begin // Simmons Tunnel Barrier model
249     if (sign(I(p,n))==1) begin
250        dxdt =c_off*sinh(I(p,n)/i_off)*exp(-exp((x_last-a_off)
251        /x_c-abs(I(p,n)/b))-x_last/x_c);
252     end
253     if (sign(I(p,n))==-1) begin
254        dxdt =c_on*sinh(I(p,n)/i_on)*exp(-exp((a_on-x_last)
255        /x_c-abs(I(p,n)/b))-x_last/x_c);
256     end
257         x=x_last+dt*dxdt;
258      if (x>=D) begin
259             x=D;
260             dxdt=0;
261      end
262      if (x<=0) begin
263             x=0;
264             dxdt=0;
265      end
266        //update the output ports(pins)
267        R=Ron*(1-x/D)+Roff*x/D;
268        x_last=x;
269        Metr(w_position) <+ x/D;
270        V(p,n) <+ (Ron*(1-x/D)+Roff*x/D)*I(p,n);
271        first_iteration=1;
272 end // end Simmons Tunnel Barrier model
273
274 ////////////////////// TEAM model /////////////////
275
276 if (model==2) begin // TEAM model
277     if (I(p,n) >= i_off) begin
278         dxdt =K_off*pow((I(p,n)/i_off-1),Alpha_off);
279     end
280     if (I(p,n) <= i_on) begin
281         dxdt =K_on*pow((I(p,n)/i_on-1),Alpha_on);
282     end
283         if ((i_on<I(p,n)) && (I(p,n)<i_off)) begin
```

```
284     dxdt=0;
285     end
286
287 // No window
288     if (window_type==0) begin
289     x=x_last+dt*dxdt;
290     end // No window
291
292     // Jogelkar window
293 if (window_type==1) begin
294   if (sign(I(p,n))==1) begin
295       sign_multply=0;
296       if(x<p_window_noise) begin
297         sign_multply=1;
298       end
299     end
300     if (sign(I(p,n))==-1) begin
301       sign_multply=0;
302       if(x>(D-p_window_noise)) begin
303         sign_multply=-1;
304       end
305     end
306         x=x_last+dt*dxdt*(1-pow(pow((2*x_last/D-1),2),p_coeff))
307         +sign_multply*p_window_noise;
308     end // Jogelkar window
309
310 // Biolek window
311     if (window_type==2) begin
312         if (stp(-I(p,n))==1) begin
313             stp_multply=1;
314         end
315         if (stp(-I(p,n))==0) begin
316             stp_multply=0;
317             end
318         x=x_last+dt*dxdt*(1-pow(pow((x_last/D-stp_multply),2),p_coeff));
319     end  // Biolek window
320
321 // Prodromakis window
322     if (window_type==3) begin
323   if (sign(I(p,n))==1) begin
324       sign_multply=0;
325       if(x<p_window_noise) begin
326         sign_multply=1;
327       end
328     end
329     if (sign(I(p,n))==-1) begin
330       sign_multply=0;
331       if(x>(D-p_window_noise)) begin
332         sign_multply=-1;
333       end
```

```
334     end
335             x=x_last+dt*dxdt*J*(1-pow((pow((x_last/D-0.5),2)+0.75),p_coeff))
336             +sign_multply*p_window_noise;
337     end  // Prodromakis window
338 
339 //Kvatinsky window
340     if (window_type==4) begin
341            if (I(p,n) >= 0) begin
342            x=x_last+dt*dxdt*exp(-exp((x_last-a_off)/x_c));
343            end
344        if (I(p,n) < 0) begin
345            x=x_last+dt*dxdt*exp(-exp((a_on-x_last)/x_c));
346        end
347         end // Kvatinsky window
348     if (x>=D) begin
349     dxdt=0;
350     x=D;
351     end
352     if (x<=0) begin
353     dxdt=0;
354     x=0;
355     end
356     lambda = ln(Roff/Ron);
357      //update the output ports(pins)
358         x_last=x;
359         Metr(w_position) <+ x/D;
360      if (IV_relation==1) begin
361         V(p,n) <+ Ron*I(p,n)*exp(lambda*(x-x_on)/(x_off-x_on));
362      end
363      else if (IV_relation==0) begin
364         V(p,n) <+ (Roff*x/D+Ron*(1-x/D))*I(p,n);
365      end
366     first_iteration=1;
367 end // end Team model
368 
369 /////////////// Nonlinear Ion Drift model ///////////////
370 
371 if (model==3) begin // Nonlinear Ion Drift model
372 
373     if (first_iteration==0) begin
374         w_last=init_state;
375     end
376     dwdt = a*pow(V(p,n),q);
377 
378 // No window
379     if ((window_type==0) || (window_type==4)) begin
380         w=w_last+dt*dwdt;
381         end // No window
382 
383   // Jogelkar window
```

```
384   if (window_type==1) begin
385   if (sign(I(p,n))==1) begin
386       sign_multply=0;
387       if(w<p_window_noise) begin
388         sign_multply=1;
389       end
390     end
391     if (sign(I(p,n))==-1) begin
392       sign_multply=0;
393       if(w>(D-p_window_noise)) begin
394         sign_multply=-1;
395       end
396     end
397         w=w_last+dt*dwdt*(1-pow(pow((2*w_last-1),2),p_coeff))
398         +sign_multply*p_window_noise;
399     end // Jogelkar window
400
401 // Biolek window
402     if (window_type==2) begin
403         if (stp(-V(p,n))==1) begin
404             stp_multply=1;
405         end
406         if (stp(-V(p,n))==0) begin
407                 stp_multply=0;
408             end
409         w=w_last+dt*dwdt*(1-pow(pow((w_last-stp_multply),2),p_coeff));
410     end // Biolek window
411
412 // Prodromakis window
413     if (window_type==3) begin
414   if (sign(I(p,n))==1) begin
415       sign_multply=0;
416       if(w<p_window_noise) begin
417         sign_multply=1;
418       end
419     end
420     if (sign(I(p,n))==-1) begin
421       sign_multply=0;
422       if(w>(D-p_window_noise)) begin
423         sign_multply=-1;
424       end
425     end
426             w=w_last+dt*dwdt*J*(1-pow((pow((w_last-0.5),2)+0.75),p_coeff))
427             +sign_multply*p_window_noise;
428     end  // Prodromakis window
429     if (w>=1) begin
430         w=1;
431         dwdt=0;
432     end
433     if (w<=0) begin
```

```
434         w=0;
435         dwdt=0;
436     end
437     //change the w width only if the
438 // threshhold_voltage permits!
439         if(abs(V(p,n))<threshhold_voltage) begin
440             w=w_last;
441         end
442       //update the output ports(pins)
443        w_last=w;
444             Metr(w_position) <+ w;
445             I(p,n) <+ pow(w,N)*beta*sinh(alpha*V(p,n))+c*(exp(g*V(p,n))-1);
446             first_iteration=1;
447 end // end Nonlinear Ion Drift model
448
449 ////////////////////// VTEAM model ///////////////////
450
451 if (model==4) begin // VTEAM model
452     if (V(p,n) >= v_off) begin
453         dxdt =K_off*pow((V(p,n)/v_off-1),Alpha_off);
454     end
455     if (V(p,n) <= v_on) begin
456         dxdt =K_on*pow((V(p,n)/v_on-1),Alpha_on);
457     end
458         if ((v_on<V(p,n)) && (V(p,n)<v_off)) begin
459     dxdt=0;
460     end
461
462 // No window
463     if (window_type==0) begin
464     x=x_last+dt*dxdt;
465     end // No window
466
467     // Jogelkar window
468 if (window_type==1) begin
469   if (sign(V(p,n))==1) begin
470       sign_multply=0;
471       if(x<p_window_noise) begin
472         sign_multply=1;
473       end
474     end
475     if (sign(V(p,n))==-1) begin
476       sign_multply=0;
477       if(x>(D-p_window_noise)) begin
478         sign_multply=-1;
479       end
480     end
481         x=x_last+dt*dxdt*(1-pow(pow((2*x_last/D-1),2),p_coeff))
482         +sign_multply*p_window_noise;
483     end // Jogelkar window
```

```
484
485 // Biolek window
486     if (window_type==2) begin
487         if (stp(-V(p,n))==1) begin
488             stp_multply=1;
489         end
490         if (stp(-V(p,n))==0) begin
491             stp_multply=0;
492             end
493         x=x_last+dt*dxdt*(1-pow(pow((x_last/D-stp_multply),2),p_coeff));
494     end  // Biolek window
495
496 // Prodromakis window
497     if (window_type==3) begin
498   if (sign(V(p,n))==1) begin
499       sign_multply=0;
500       if(x<p_window_noise) begin
501         sign_multply=1;
502       end
503     end
504     if (sign(V(p,n))==-1) begin
505       sign_multply=0;
506       if(x>(D-p_window_noise)) begin
507         sign_multply=-1;
508       end
509     end
510             x=x_last+dt*dxdt*J*(1-pow((pow((x_last/D-0.5),2)+0.75),p_coeff))
511             +sign_multply*p_window_noise;
512     end  // Prodromakis window
513
514 //Kvatinsky window2 VTEAM only
515     if (window_type==5) begin
516             if (V(p,n) >= 0) begin
517             x=x_last+dt*dxdt*exp(-exp((x_last-a_off)/x_c));
518             end
519        if (V(p,n) < 0) begin
520             x=x_last+dt*dxdt*exp(-exp((a_on-x_last)/x_c));
521        end
522         end // Kvatinsky window
523     if (x>=D) begin
524     dxdt=0;
525     x=D;
526     end
527     if (x<=0) begin
528     dxdt=0;
529     x=0;
530     end
531        lambda = ln(Roff/Ron);
532      //update the output ports(pins)
533         x_last=x;
```

```
534         Metr(w_position) <+ x/D;
535       if (IV_relation==1) begin
536         V(p,n) <+ Ron*I(p,n)*exp(lambda*(x-x_on)/(x_off-x_on));
537       end
538       else if (IV_relation==0) begin
539         V(p,n) <+ (Roff*x/D+Ron*(1-x/D))*I(p,n);
540       end
541      first_iteration=1;
542 end // end VTEAM model
543   end   // end analog
544 endmodule
```