Thesis for Master's Degree in Physics

Investigation of Resistive Random Access Memory for 1T1R Nanowire Array Integration

Autumn 2019 - Spring 2020

Course code: FYTM30

Division of Synchrotron Radiation

Author:
Orestes Theodoridis

Supervisor:
Dr. Karl-Magnus Persson
Co-Supervisors:
Saketh Ram Mamidala
Dr. Anders Mikkelsen

Lund University
Faculty of Science

Contents

Acknowledgements

I would first and foremost like to express my sincere gratitude to Dr. Karl-Magnus Persson for the supervising, teaching and encouragement throughout the project. I would also like to acknowledge Prof. Lars-Erik Wernersson, Saketh Ram Mamidala, and the rest of the Nanoelectronics Group at Lund University for their fascinating seminars on nanoscale electronics and teaching of the instruments used in this work. Lastly, I wish to express my deepest gratitude to Mattias Åstrand for invaluable discussions on physics, project strategies and for his support throughout this project.

Acronyms and Abbreviations

RRAM Resistive Random Access Memory

CMOS Complementary Metal Oxide Semiconductor

MOSFET Metal-Oxide-Semiconductor Field Effect Transistor

VM Volatile Memory

NVM Non-Volatile Memory

MIM Metal-Insulator-Metal

TE Top Electrode

BE Bottom Electrode

$\mathbf{HfO_2}$ Hafnium Dioxide

ITO Indium-Tin Oxide

TiN Titanium Nitride

HRS High Resistance State

LRS Low Resistance State

CBA Cross-Bar Array

ALD Atomic Layer Deposition

Abstract

As modern electronics have started to reach its physical scaling limits, novel architectures and physics is needed to meet future demands. Oxide-based Resistive Random Access Memory (RRAM) is a new emerging technology that uses filament formation and rupture in thin oxides to generate resistive switching. Structure of RRAM devices often use transistors as selector devices. In this work a one-transistor-one-RRAM (1T1R) device is characterised using pulsed measurements. The endurance of the device is extracted as well as the switching probability. Moreover, the data acquired from the device is used to simulate 1T1R nano-wire (NW) arrays by integrating a cell level small signal model with an array-level model. Analytical expressions are used to calculate parasitic capacitances. Worst-case cell analyses of the access voltage and read margin are performed for different pulse widths, resistivities and technology nodes. The average switching probabilities for a given array size are also calculated using experimental data. It turns out that the examined device showed excellent endurance, exceeding 2 million cycles. Moreover, it also showed excellent switching characteristics and resistance window. Simulations showed that high probability of switching could be achieved even for array sizes > 1000 bits. These results show that it is possible to integrate $>$ Mbit sub-arrays with low-voltage 10 ns pulses, providing a foundation for larger Gbit memory sizes, which are comparable with current DRAM and NAND technology.

1 Introduction

Throughout history, humanity has used external resources to record information in order to pass on knowledge. The external bearers of information have evolved, ranging from sequences of stone formations to ink on paper or magnetic tapes. While the technology responsible for the bearing of information has changed, the mechanism responsible for the retention of information is often referred to as *memory*. Modern memory technology has evolved both in terms of speed and size. The invention of digital memory has led to smaller and faster technologies, such as tapes, hard disk drives and the more recent solid state flash memory. As memory technology has improved, so has also the computing capability of computers. One major reason for this can be attributed to the successful scaling and integration of silicon (Si) complementary metal oxide semiconductor (CMOS) transistors, which has constituted a major part of the past 50 years of semiconductor research.

Today, on the other hand, the future of memory technology suffers from two main bottlenecks. One has to do with the scale of nanoscale devices. The size of transistor technology is approaching its physical limits. This means that the small size of a transistor leads to leakage currents or a decrease in performance from parasitic elements. The parasitic elements become significant for devices and interconnects in close proximity [1]. Another main obstacle concerns the von Neumann architecture of conventional computers. The data processing of the central processing unit and the random access memory unit are separated, giving an increase in latency and power consumption [2].

In order to tackle these bottlenecks, several novel technologies have been proposed, ranging from device architecture to material science [3]. One branch of technology in particular is called *resistive random access memory* (RRAM), which is the specific objective of study in this work. The following text is therefore concerned with the electronic characteristics and physics of RRAM technology. The study is exploratory and interpretive in nature, meaning its outcomes are not hypothetically known or estimated beforehand. A combination of device-level empirical measurements and system-level simulations are used in the analysis. Ultimately, a relation between experiment and simulations will be examined and presented.

1.1 Historical Background

In his famous paper from 1971, Leon Chua laid down the theory for a fourth fundamental circuit element with the possible property of carrying memory. He held the resistor, capacitor and inductor as the other three fundamental circuit elements. In his text, Chua argued that the properties of these elements are governed by differential equations between four main quantities: voltage v, current i, charge q and flux linkage ϕ. Since there are four quantities linking three components, Chua postulated that, by symmetry, a fourth fundamental component could exist. The differential equations that link these components are listed in table 1.

Table 1: The four fundamental circuit devices are related by differential equations between voltage v, current i, magnetic flux linkage ϕ and charge q.

Device	Property (unit)	Relationship
Resistor	Resistance (ohm)	$R = \frac{dv}{di}$
Capacitor	Capacitance (farad)	$C = \frac{dq}{dv}$
Inductor	Inductance (henry)	$L = \frac{d\phi}{di}$
Memristor	Memristance (ohm)	$M = \frac{d\phi}{dq}$

Among these relationships, the resistor, capacitor and inductor were well known. Only the last element, the memristor, was at the time Chua's paper was written not discovered. He also showed that solving the above equation for voltage as a function of time gave the relationship

$$v(t) = M(q(t))i(t), \tag{1}$$

which showed that the memristor has a charge-dependent resistance [4]. It presented a possibility for a circuit component whose resistance is dependent on the history of past voltages and currents.

While the theoretical and practical implication of Chua's paper has been up for considerable debate, his idea sparked an interest in the discovery of devices whose resistance depends on past voltages. The research into memristors has led to a variety of possible candidates. These include (among others) ferroelectric memories, phase change memories and resistive memories [1]. While there is much to be said of each technology, it is outside the scope of this work, and the rest of the text will therefore be concerned with resistive memory technology.

1.2 Current Memory Technology

Modern digital memory technology is diverse and the physical mechanisms depend on the application. Information is measured in bits and one bit of information has a binary value of a 0 or a 1. For a memory cell to store information it thus needs to be able to switch between two well defined states, as well as being read without disrupting its value. What determines how fast and how efficiently a memory cell can be switched or read depends on the cell size, materials as well as its complexity.

1.2.1 Volatile Memory

Digital memory technology can be classified into *volatile* and *non-volatile* memory (VM and NVM respectively). Volatile memory needs constant power consumption to retain its data. The two dominant VM technologies that dominate the market today is considered to be dynamic and static random access memory (DRAM and SRAM, respectively). These have different cell architectures, strengths and weaknesses. SRAM technology is a fast memory technology used often as temporary storage for central processing units (CPUs). The memory cell is in its most generic form consisted of six transistors, forming two inverters, which makes the SRAM cell capable of writing and retaining the data without refreshment, although at the cost of a large feature size due to the number of components needed. A DRAM cell, on the other hand, is built up of one transistor and one capacitor. Its small feature size makes DRAM capable of storing large amounts of data at high speed. The major drawback of this technology is its high power consumption. Due to current leakage the charge of the capacitor needs to be refreshed regularly to retain its memory [5].

1.2.2 Non-Volatile Memory

Non-volatile memory, as opposed to volatile memory, is characterised by its ability to retain data even without power consumption. Modern USB memories and solid state drives (SSD) are composed of flash memory cells, which include technology based on the NAND or NOR logic gate architectures. A flash memory cell is consisted of a single floating-gate MOSFET, which makes it possible to significantly reduce its size. It has a very fast read speed, though it suffers from a much lower write speed. Furthermore, its endurance (number of read/write cycles before failure) is much lower than for static and dynamic RAM, which hinders it from applications where many read and write operations are needed [5].

1.3 Benefits and Types of RRAM

The current digital memory landscape thus brings fast volatile memory on one hand, and energy-efficient / small non-volatile memory on the other hand. Though all of these technologies have excellent performance in terms of random access and reliability, they suffer from high power consumption and low write speed. To tackle these problems new architectures, engineering and physics are needed.

Resistive random access memory holds promise for tackling all of these drawbacks. Its simple structure makes it both fast and highly scalable, while also operating at high speed and low power. Moreover RRAM is non-volatile as well. While the details of the mechanism and architectures of RRAM will be laid out below, there are additional computational advances that can be realised and improved by RRAM.

1.3.1 Neuromorphic Computing

Neuromorphic computing (NC) draws inspiration from the brain and aims to resolve the von-Neumann limitations of sequential computing. The von-Neumann architecture is assembled to produce computations in series. The reason for this is because of the separation of the processing unit and the memory. Neuromorphic computing architecture, on the other hand, compute *in-memory*, meaning that signals travel to different nodes, as opposed to sequential inputs and outputs. This distributed way of signal processing makes it possible for NC to solve problems much faster than conventional computers. In particular, problems related to artificial neural networks, which also have nodal signal pathways, will become faster. There are neuromorphic architectures today based on current transistor technology. As such, power consumption is still a problem even though the problem-solving capacity is improved [6].

1.3.2 Hyper-Dimensional Computing

Hyper-dimensional computing is a newer development of brain-inspired computing. The central objective is to build a computational model that incorporates the size and error immunity of the brain. Data is represented as random vectors with a very high dimensionality ($\sim$ 10000 bits). Any new random vector will, on average, differ in 50 % of the bits. As a result, data with high uncertainty (10-20 % error rate) can still be classified. This, in turn, means that HD computing is capable of one-shot learning, meaning it can classify data with few examples, as opposed to neural networks that require huge amounts of data. RRAM based hyper-dimensional computing on 3D arrays have been demonstrated, capable of recognising 21 languages with 90 % and 98 % accuracy [7, 8].

1.3.3 Classification

RRAM can be classified into two types with respect to material selection. *Anion-based* RRAM is the type that is used in this thesis, where oxygen ion migration and redox-reaction is the main mechanism to form conductive filaments of oxygen vacancies. This type of RRAM has been extensively researched [1, 3, 9] and is also in most cases classified as bipolar due to the fact that a reverse voltage is needed to rupture the conductive filament.

Cation-based (CB) RRAM, on the other hand, is dominated by electro-chemical reactions and diffusion of Ag or Cu to form a conductive bridge. This type of RRAM is normally operated with voltage pulses of the same polarity, where the main mechanism for the rupture the conductive filament is Joule heating [10].

Anion and cation based RRAM have shown strengths and weaknesses of different kinds. Anion type RRAM are the most researched and have exhibited the best performance with regards to endurance and compatibility with industry CMOS technology as a high-k technology node. On the other hand, one main obstacle is the need for a forming operation (explained in section 2.1.1). This lies in contrast to CBRAM which does not need a forming operation. CBRAM also has a high resistance window between its two states, making it a viable option for multi-bit storage cells [10]. Therein lies a challenge as well, as the diffusion of Ag and Cu atoms in CBRAM makes the conductive bridge hard to control consistently. This in turn is often considered as a main contributor to the low endurance of CBRAM [1].

There is much more to be said of both these categories of RRAM and while this classification holds in general, there are cases where it is problematic. There are indeed cases where oxide-based RRAM has displayed Joule heating filament rupture [11]. Furthermore, ITO based RRAM has shown conductive bridge formation due to indium diffusion [12]. Even so, this categorisation of RRAM is still used and provides a general landscape of the technology.

2 Theory

The theory of RRAM technology is still ongoing, and much is yet to be discovered. In particular, the details regarding the switching mechanics of RRAM remain to be unfolded. The following section thus aims to provide the frameworks of the general consensus regarding anion-type oxide RRAM, which is the technology of interest in this work and has been extensively researched in Lund University. In particular the specific mechanisms at play for ITO-HfO_2-TiN RRAM will be considered. A large part of this work also investigates the system-level behaviour of RRAM. For this case the terms "RRAM device" and "RRAM cell" will at times be used interchangeably, as opposed to "RRAM array", which will denote the system composed of RRAM cells. The author apologises for any unintentional ambiguities that may arise due to this terminology.

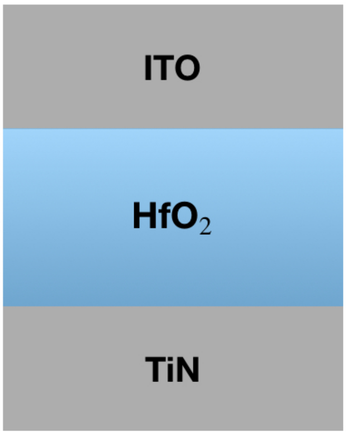

Figure 1: Schematic of a MIM RRAM cell. The dielectric HfO_2 is sandwiched between the top electrode ITO and bottom electrode TiN.

2.1 The RRAM Cell

The structure of an oxide-based cell is, at its core, identical to a capacitor. For an oxide-based RRAM a thin insulating high-k dielectric (in this work: hafnium dioxide, or HfO_2) is sandwiched between two metal electrodes (a metal-oxide-metal (MIM) structure) which is illustrated in figure 1. The top electrode (TE) signifies the input, while the output is represented by the bottom electrode (BE), which may be composed of different metals or alloys. In this work the TE is made of indium-tin-oxide (ITO) and the bottom electrode of titanium nitride (TiN).

2.1.1 Switching Mechanism and Operation Modes

The main mechanism for switching of oxide based RRAM is the formation and rupture of a conductive filament. These are formed by applying a soft breakdown voltage over the device. Initially the RRAM is at a high resistance state (HRS). To get a working RRAM cell, a filament is constructed by a forming voltage. The electric field displaces oxygen ions, which reside in the dielectric, towards the oxygen rich ITO, leaving a conductive oxygen vacancy filament. In this mode the RRAM is said to be in a low resistance state (LRS). To reset the RRAM into a HRS a reverse bias is applied which lets the oxygen in the ITO re-migrate into the filament, resulting in a rupture. A set operation can be programmed by again applying a positive voltage. This process is illustrated in figure 2a-d.

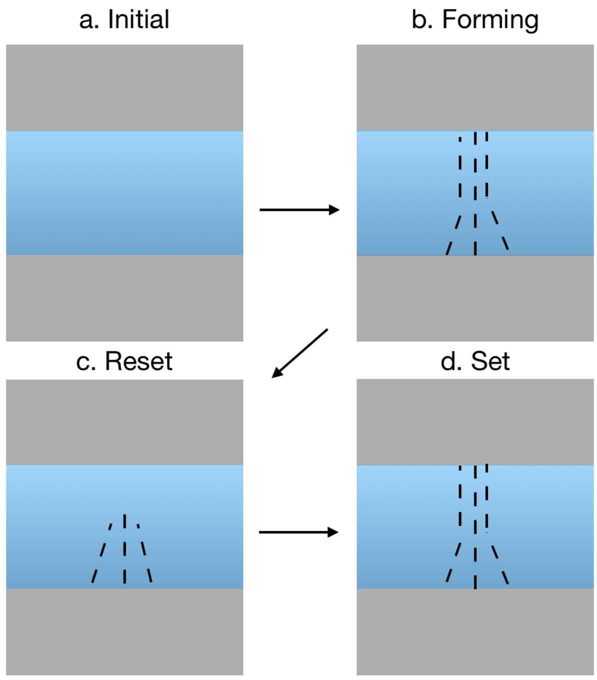

Figure 2: Schematic of the different operations of an RRAM cell. a) The RRAM is in its initial high resistance state. b) A soft breakdown forming voltage is applied which results in a conductive filament of oxygen vacancies (dashed lines). The RRAM is now in a low resistance state. c) A reverse voltage pulse ruptures the filament by ion-vacancy recombination which rests the cell into a high resistance state. d) the RRAM is now functioning and can be set to a low resistance state.

These modes can be observed by measuring the direct current (DC) versus voltage curve of the RRAM, which is illustrated in figure 3. A positive voltage is applied and the RRAM behaves as a resistor, until a critical voltage is achieved (the SET voltage V_{set}). The filament is formed and the lowered resistance allows for a greater current flow. The resetting is achieved by a negative bias where, just as for the SET operation, a critical voltage V_{reset} ruptures the filament and resets the cell into HRS. Although the curve might look different depending on the structure and material of the RRAM, this generic curve illustrates a generic RRAM switching behaviour.

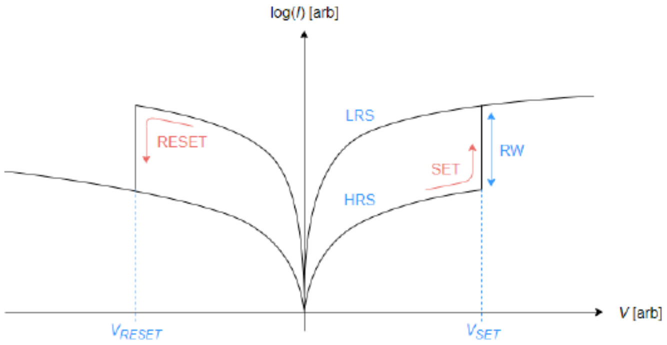

Figure 3: Generic IV characteristics of a RRAM cell. The operations involve the set which sets the RRAM from a high resistances state to a low resistance state. The reset operation resets the device. The path of the voltage sweep is given by the arrows. RW denotes the resistance window. The figure is reprinted with permission from [13]

There are a number of different quantities and properties of the RRAM that determine the SET and

RESET voltage. The materials of all three components; TE, BE and dielectric, significantly alter the behaviour of the RRAM. The oxide thickness is also an important parameter that determines how the filament is formed. The RRAM material stack chosen for this work is described in section 3.1.1.

2.2 The Crossbar Array

As mentioned above, RRAM stands as a promising candidate for novel computing architectures. In order to use RRAM in applications, arrays of devices will be used. Below follows the background regarding one of the most investigated structures for the realisation of new computation architectures, the *crossbar array* (CBA)[14, 7, 8, 15, 16]. The CBA is a square array with the rows and columns separated by RRAM cells, situated on each node. This structure is attractive because of its simplicity and similarity with current DRAM arrays [8, 5]. The architecture allows for random access of each device, meaning the time it takes to write and read information from each node is in the same time scale. Furthermore, the cells may be composed of various memory technology as well as compositions of devices, which are used to build artificial neurons [6]. Thus, an investigation of the CBA is not only important for oxide based RRAM, but for all resistive non-volatile memory technology. Nonetheless, this work will focus on CBA performance in relation to RRAM.

The CBA has different appearances and architectures. The cells may be made of a single RRAM cell as depicted in figure 1. This will be referred to as a *passive* CBA and is, in terms of engineering, an uncomplicated structure. A passive crossbar array is illustrated in figure 4. Ideally, passive CBAs would stand as an excellent structure, though the operations that are used to write and read memory bring difficulties related to leakage currents.

Active CBAs have been shown to suppress leakage currents by introducing selector devices in the cell. The selectors are coupled in series with the RRAM. Different selectors have been investigated both experimentally and by simulation, ranging from diodes to transistors [17, 18, 19, 20]. For this work, the simulation will incorporate nanowire transistors as the selector device. In particular the one-transistor-one-RRAM (1T1R) structure will be investigated. A generic array structure for a 1T1R NWRRAM array can be seen in figure 5. The circuit representations for both 1T1R and one-RRAM (1R) cells can be viewed in figure 6.

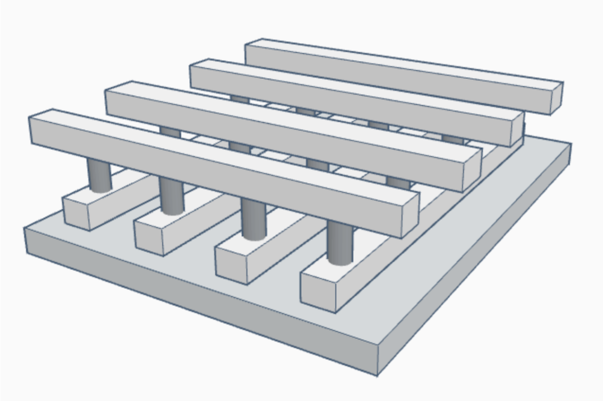

Figure 4: An illustration of a passive crossbar array. At each intersection a RRAM cell is situated. By applying a potential difference between the bottom level (word line) and the top level (bit line) the RRAM can be switched on and off.

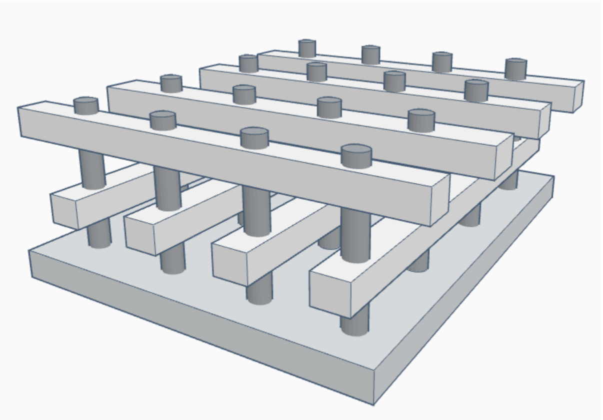

Figure 5: An illustration of an active 1T1R NW array. The NWs are attached to the substrate. The word-line is connected to the transistor gate (lower level) and the bit-line to the RRAM device (top level).

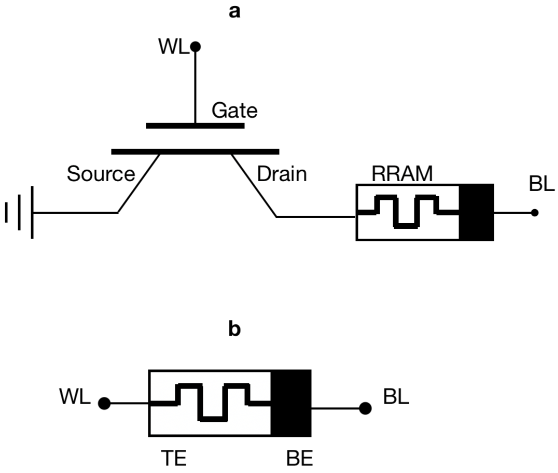

Figure 6: Circuit representation of a 1T1R and 1R cell structure. (**a**) A 1T1R structure. The WL is connected to the gate of the MOSFET and the BL to the source. The RRAM is connected in series with the drain of the transistor with the BE grounded. (**b**) A single passive 1R cell. The WL is connected to the TE and the BL to the BE

A cell is programmed by applying voltages to the word-line (WL) and bit-line (BL) of the CBA. The WL connects to a row of cells. The operation is trivial for passive CBAs but for 1T1R CBAs the WL accesses the transistor and allows for a current to flow between the source and the drain. The specific way the WL and BL are biased depends on the type of CBA, and will be explained in detail below.

3 Method

As mentioned above, this work consists of both an experimental and a simulations part. A one-transistor-one-RRAM (1T1R) structure was examined and the data was used to investigate the RRAM's performance in a simulated array environment. The methods for both parts are explained below. It should be noted that several RRAM devices and structures were investigated, with different equipment, before it was decided on which experiment to include in the work. This work accounts for one of these experiments, where pulsed voltages were applied on a 1T1R structure, which gave presentable and clear results, as well as good statistical performance. Lastly, the RRAM device in the experimental part and the simulated NW RRAM devices have different structures. Ideally the measured device should have close resemblance to the simulated one, but at the current time no such devices had been successfully processed.

3.1 Experimental Part

3.1.1 Experimental Setup and Data Acquisition

Several different RRAM stacks had been previously fabricated at Lund NanoLab. The RRAM device that is presented in this work was grown in a via opening on a Silicon on oxide-substrate with a 200 nm oxide layer. It has a TiN top electrode deposited by 200 °C atomic layer deposition (ALD), a 3 nm thick HfO_2 film also deposited by 200 °C ALD. The 20 nm thick ITO bottom electrode was deposited with sputtering with Au cover. Further details concerning the fabrication of the stack can be found in [21].

The RRAM device is connected to a FET with a coaxial wire. The FET used for this work had been previously fabricated for low-power RF-applications and consisted of a single NW [22]. For this experiment it acted as a selector, with the role of reducing current overshoot and forces the current to compliance. The circuit representation of this setup is given in figure 6a above.

The devices were situated on a probe station. The RRAM BE and the gate of the transistor were connected by probes to a B1500A Device Parameter Analyser by Keysight Technologies, which was used to apply signals and measure the current. Before probing the device, the probes were assured to be working by measuring signals to the gold cover of the RRAM sample. Care was taken to not break the probes.

Following this, a functional FET was chosen. The transfer and output characteristics of the FET can be viewed in figures 7 and 8 respectively. From figure 7 one can see the drain current as dependent on gate voltage for V_{SD} = 0.5 V (lower line) and 1 V (upper line). This shows a well functioning gate modulation of the current. Figure 8 shows the drain current as dependent on the source/drain voltage, with the gate voltage swept between -0.8 V to 0.8 V in steps of 0.4 V.

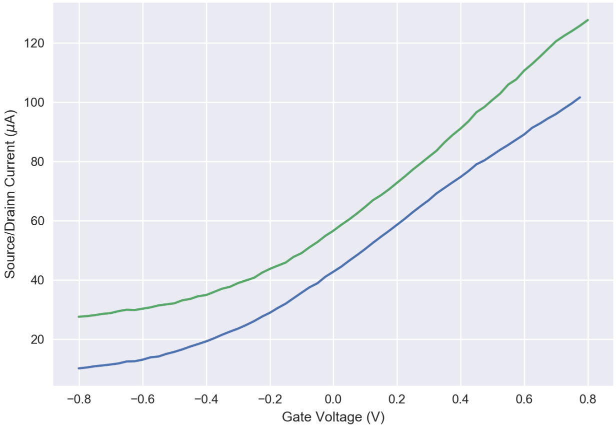

Figure 7: The transfer characteristics of a FET device used in the experiment for V_{SD} = 0.5 V (lower line) and 1 V (upper line)

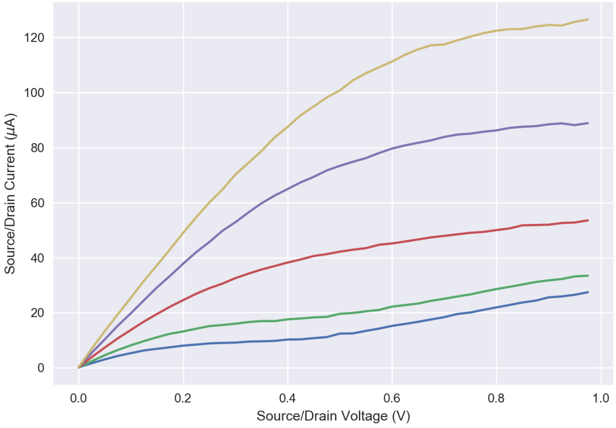

Figure 8: The output characteristics of a FET device used in the experiment. The output was measured by sweeping the gate voltage from -0.8 V (lowest line) to 0.8 V (upmost line) in steps of 0.4 V

Well defined biasing schemes are needed to operate the 1T1R device. The three different modes of operation for the setup are the set, reset and read operations. Motivated by the FET figures 7 and 8, the set operation was defined at zero gate bias to counteract current overshoot. For the read operation the gate bias was set to 0.4 V. This level of gate bias gave a larger source/drain current. For the reset operation a large gate bias was applied so as to allow for a high compliance current. For each gate pulse, the structure was biased from RRAM to the drain of the FET with different amplitudes. The set operation had positive voltages, while the reset had negative. The experiment was implemented for amplitudes 1.5 V and 1 V. The reading voltage was set to 50 mV. A summary of the bias scheme can be viewed in table 2.

Table 2: The bias scheme for the 1T1R experimental setup

Gate Voltage	Source-Drain Voltages	Operation
0.0 V	1.5/1 V	Set
0.4 V	50 mV	Read
0.8 V	-1.5/-1 V	Reset

One sequence of measurements corresponded to the operations read-set-read-reset-read. The pulses on the gate were square, while the pulses between the source and drain were triangular. The wire connecting the RRAM to the FET was long enough to give rise to capacitive currents for small pulses. To alleviate this, long pulses of 3 ms pulse width were applied. Figure 9 shows the current and gate voltage of one measuring sequence with voltage amplitude being 1.5 V. The first current spike shows a successful set event, with a sudden vertical elevation at around 7 ms. The same is true for the reset operation at 17 ms, where the first negative current spike marks the reset event.

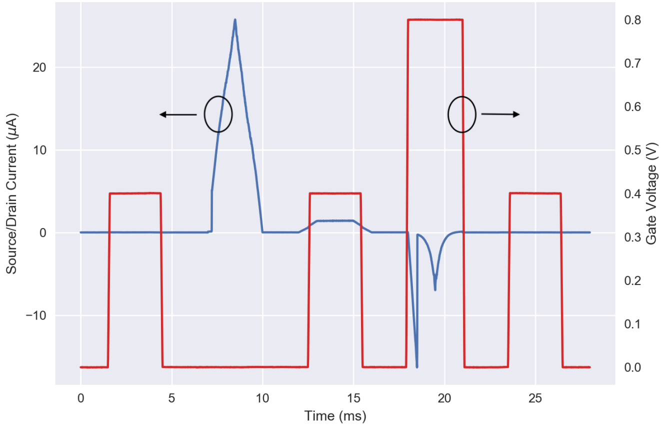

Figure 9: One cycle of measurements shows the source/drain current (left axis) and gate voltage (right axis) and their time evolution. The sequence of operations are read-set-read-reset-read. The stop voltage for this measurement was 1.5 V

3.1.2 Data Handling

The pulsing sequence described above was implemented for the setup up to two million times. Because of the high number of pulses, there was no way to measure every cycle. Because of this checkpoints were placed in logarithmic intervals to measure the status of the device. The data acquired was handled using the Python programming language. Before analysis, background noise was subtracted from the current.

In order to extract the statistics of V_{set} and V_{reset}, the switching event had to be clearly defined. The definition of the set voltage was done by looking at the derivative of the current. The reason behind this being that the steepest change should mark the middle of the switching event. While it proved successful most of the times, it marked a number of false switching events. It was found that the current had neighbouring points that were noisy enough to give off spikes of false switching events. In order to mitigate this, the data was first filtered through a Gaussian filter to smooth out the curve. Figure 10 shows the process of extracting one set event.

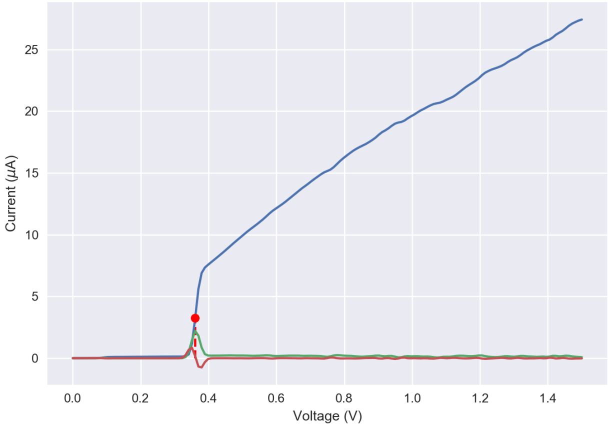

Figure 10: A set event marked by the gradient. The data is filtered and the middle of the set event was found by the derivative and second derivative of the curve.

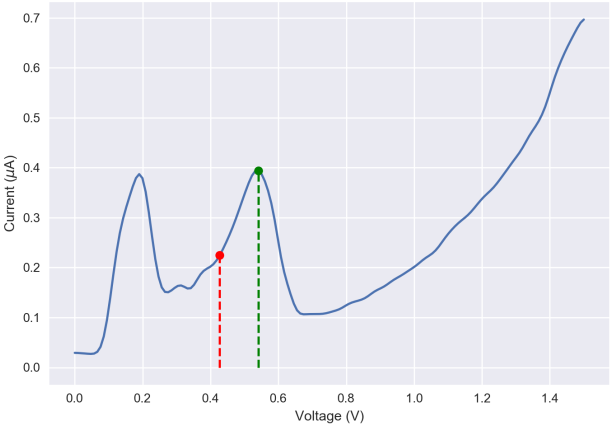

Figure 11: A reset event marked by the gradient. Two reset events are seen, yet only one is successfully marked

The reset voltage statistics were extracted differently. This was mainly because of the fact that one sweep sometimes included two reset events. One such sweep can be seen in figure 11, where two points are marked. The leftmost (red) marks a false event and the rightmost (green) marks a true second reset event. The actual reset voltage was then defined as the average of the two. Because of the wide switching window of the device, a false reset voltage was not seen as problematic, since it is in general unclear how to define a switching voltage for a double reset event.

For the purpose of calculating the LRS and HRS resistances of the RRAM, the reading current for each cycle was extracted. The structure was operated at $V_r = 20$ mV for a reading operation. From figure 8 one can see that the onset of the curve gives the on resistance of the device, R_{on}. Since the measured current applies to the whole structure which is connected in series, we get the total resistance $R_{tot} = R_{RRAM} + R_{on} = V_r/I_r$, where I_r is the reading current through the structure. A simple rearrangement gives an estimation of the resistance of the RRAM:

$$R_{RRAM} = \frac{V_r}{I_r} - R_{on}. \tag{2}$$

Once the relevant quantities had been clearly defined and extracted, the statistics were calculated. The quantities that were calculated were the mean and standard deviation of the set/reset voltage and the switching probability. The switching probability was extracted by calculating the cumulative probability distribution of the switching voltages.

3.2 Simulation Part

The simulation of the CBA can be divided into two parts. One part deals with an array without selectors. The model was incorporated from [15] with the reason of visualising the DC access voltage without selectors. The second model is a small-signal model on cell level, and is based on the framework that was in development in the Nano-electronics group in Lund University.

3.2.1 Passive Array

For the purpose of developing a simulation platform for a passive crossbar array, the approach given in [15] was incorporated, which allows for versatile biasing schemes. As such, details concerning derivation and mathematical rigour of the resulting formulas are omitted in this work and the interested reader can be referred to [15].

The CBA is represented by a matrix, which can be viewed in figure 12. The matrix is rectangular with m rows (word lines) and n columns (bit-lines), whereby the intersections containing RRAM cells are represented as matrix elements $\{i \in m, j \in n\}$. The array is operated at a supply voltage V_{app} which can be applied to the left and right of the WL plane ($V_{app,WL1}$ and $V_{app,WL2}$ respectively) or to the top and bottom of the BL plane ($V_{app,BL1}$ and $V_{app,BL2}$ respectively). The applied voltage has an access resistance, which in the model is represented by the vectors $R_{S,WL1}$, $R_{S,WL2}$, $R_{S,BL1}$ and $R_{S,BL2}$. The nodes are connected by wires with a specific line resistance R_l. Each node contains an RRAM with a specific resistance. The RRAM resistance can be represented by a resistance matrix with dimensions $m \times n$.

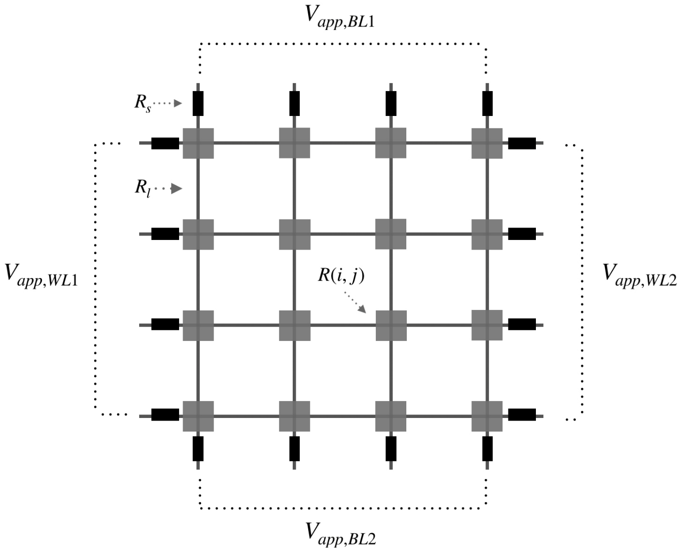

Figure 12: A schematic of a 4x4 passive CBA with corresponding parameters. The array can be biased on the WL plane ($V_{app,WL1}$ and $V_{app,WL2}$) and the BL plane($V_{app,BL1}$ and $V_{app,BL2}$). At each intersection resides an RRAM cell with resistance R_{ij}. The model also includes line resistance R_l and access resistances R_s.

This model effectively represents a 4 port network, whereby the cells can be classified into four groups. Selected devices are accessed by a selected WL and BL. The unselected devices have a common unselected WL and BL bias. Half selected devices have either common WL or BL bias with the selected devices. At each node Kirchhoff's continuity equation is assumed to hold. For an array of size $m \times n$ this means that there will be $2mn$ equations (one set for the WL plane and one set for the BL plane). The current at site (i, j) is

$$\begin{aligned} I_{WL}(i,j) &= I(i,j) + I_{WL}(i,j+1) \\ I_{BL}(i,j) &= I(i,j) + I_{BL}(i-1,j), \end{aligned} \tag{3}$$

where $I(i, j)$ is the current through the cell, $I_{WL}(i, j)$ the current in the WL plane going into node (i, j) from $(i, j-1)$ and $I_{BL}(i, j)$ the current in the BL plane going out of the node (i, j). The equations may also be written in voltage form. The resulting unknown voltage delivered on the WL and BL planes can be obtained by solving the $2mn$ equations [15]. How this is done is given by the Python code in the appendix.

Figure 13 shows one simulation run for a 32x32 bit array, with a high line resistance, for the purpose of illustration. The voltage loss is clearly demonstrated, with the best cell residing in the top left corner, and the worst cell at the lower right corner. Biasing schemes are integral to passive array operation to not switch unselected devices. For this run the selected WL was biased at $V_{app} = 1$ V while the unselected WL and the BL were biased at $V_{app}/2$. The WL was biased from the left and BL from the top. The access voltage given in the figure is defined as the voltage difference between the BL and WL planes.

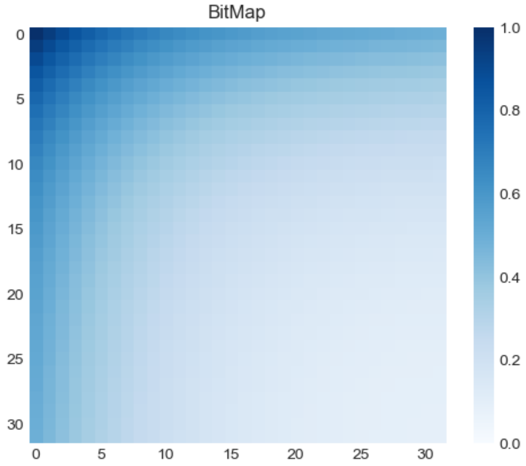

Figure 13: Bit map of a 32x32 bit passive array. Each square pixel correponds to a RRAM cell. The left y axis shows the insertion point of the BL voltage and the top x axis the insertion point of the WL voltage. The applied voltage in this run was 1 V. The map illustrates the behaviour of a passive array with effects of line resistance and sneak path currents. For this simulation run the line resistance was raised significantly for visualisation purposes.

The figure shows that the worst case bit (at the bottom rightmost corner) suffers the worst access voltage, while the top leftmost corner yields the best access voltage, as would be expected. With this model at hand, the 1T1R cells could be incorporated.

3.2.2 1T1R Array

For the purpose of investigating nanowire 1T1R arrays, an active 1T1R cell model was constructed. The model takes into account the dimensions of the NW and RRAM. The schematic of a cell and its corresponding dimensions can be viewed in figures 14 and 15.

Figure 14 shows a top-down view of the 1T1R cell. The NW is situated in the middle with a diameter d_{nw}. The total NW diameter includes several development films which are formed during the fabrication process. The interconnects have a width W and are in this work half the size of the node U.

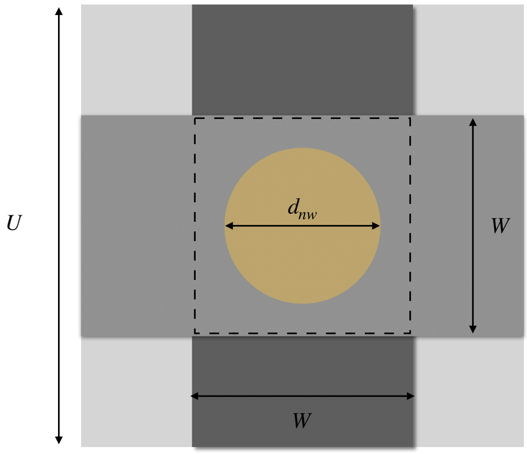

Figure 14: A top-down schematic of the NW cell, with its corresponding dimensions. U signifies the unit length of the square cell, W the width of the interconnect and d_{nw} the diameter of the nanowire. The figure is illustrative and the relative dimensions differ in the simulations.

Figure 15 shows a side-view of one NW 1T1R cell. The distances between interconnects are marked as well as a circuit schematic.

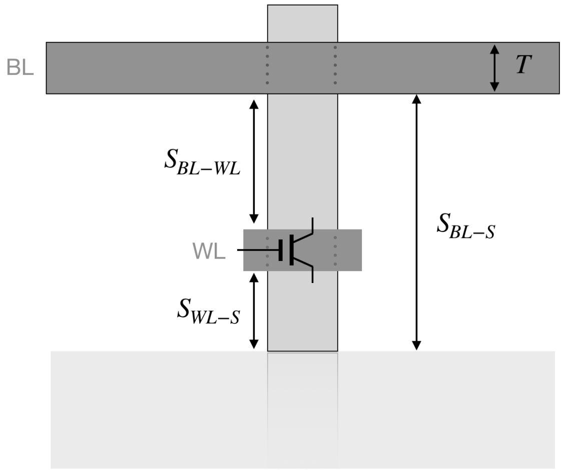

Figure 15: A sideways schematic of the NW RRAM cell. The distances between the WL, BL and source/substrate are S_{BL-WL}, S_{BL-S} and S_{WL-S}. The interconnects have thickness T

3.2.3 Small Signal Modelling of NW 1T1R cell

The 1T1R cell was modelled to incorporate signal loss, which arises due to parasitic capacitances of the cell. These losses might not be apparent for small arrays, but may give rise to large losses for larger arrays. Thus the RRAM device can be viewed as being constituted by a resistor, R_{RRAM}, and a capacitor, C_{RRAM}, connected in parallel. The resulting circuit representation can be viewed in figure 16. The parameters that were used in the simulations are shown in table 3. While the NW material is generic in this work, some material parameters were chosen in particular. The interconnects are made of copper, because of its extensive use in the industry. For the same reason the interconnects and NW are spaced by silicon dioxide.

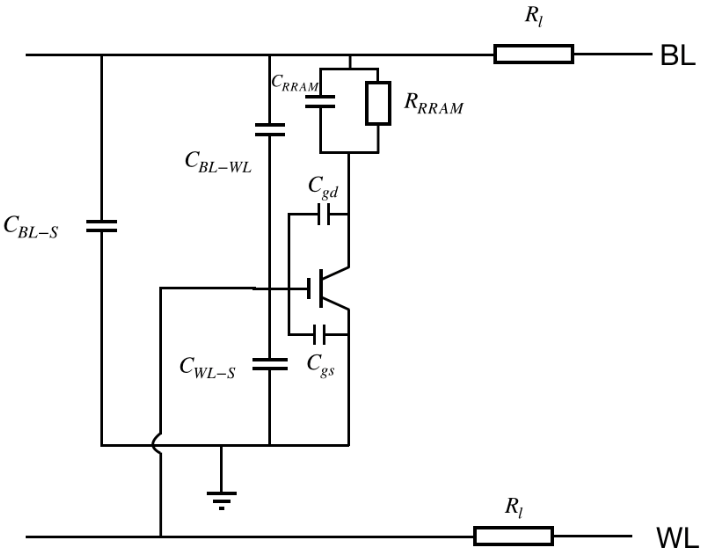

Figure 16: Circuit representation of a 1T1R NW cell without fringe and neighbouring capacitances. The RRAM is modelled as a resistor coupled in parallel with a capacitor.

Table 3: Parameters for a 1T1R cell for different node sizes

Parameter	Description	Value (Node = 35/50/100 nm)
d_{nw}	Nanowire diameter	8/12/18 nm
W	Interconnect width	17.5/25/50 nm
T	Interconnect thickness	35/50/100 nm
ρ_C	Interconnect Resistivity	$5.8 \times 10^{-8} \Omega\,\mathrm{m}$
$t_{ox-RRAM}$	RRAM oxide thickness	2.8 nm
L_{nw}	NW Length	320/480/720 nm
S_{WL-S}	WL-Substrate Distance	80/120/160 nm
Lg	FET Gate Length	30/40/50 nm

3.2.4 Capacitances

The different geometries included in a cell give rise to varying types of capacitances. Firstly the parasitic capacitances of the NWFET were modelled as coaxial capacitances:

$$C_{coax} = \frac{2\pi\epsilon_0\epsilon_{ox}}{\ln\left((r_{nw}+t_{ox})/r_{nw}\right)} \cdot t_l, \tag{4}$$

where ϵ_0 is the electric permitivity of vacuum, ϵ_{ox} the relative permitivity of the oxide, r_{nw} the radius of the NW, t_{ox} the oxide thickness and t_l the thickness of the WL.

The capacitance between the interconnects have a parallel component. For that the formula for parallel plate capacitance was used:

$$C_{par} = \frac{\epsilon_0\epsilon_{ox}A}{d}, \tag{5}$$

where A is the area of the parallel plate and d the distance between the corresponding parallel plates.

Besides these two types of capacitances, the close proximity of cells and interconnects give rise to additional parasitic elements, which occur between fringes of interconnects and neighbouring cells. While it is possible to compute the fringe capacitances numerically using a wide array of software, some of them can be extracted analytically and used without any significant computational cost. The conceptual framework of how these are inferred can be found in [23]. In this work the resulting analytical expressions are incorporated as they were presented in the article.

In figure 17 the two fringe capacitances used in this work are demonstrated between interconnects. The WL and BL have capacitive couplings with both the substrate and each other. The capacitive coupling from the BL to the substrate goes from the side of the BL electrode to the top of the substrate. The same type of coupling is modelled between the side of the BL electrode and the top of the WL. The electrodes also couple to adjacent electrodes in the same level, where the coupling that is mainly between the top of each interconnect. The formulas for the two fringe capacitances C_{s-t} and C_{t-t} are given in equations (10) and (11) in appendix A.1. For each cell C_{t-t} was calculated between all interconnects in a proximity of 2 microns. C_{s-t} was calculated both between interconnects and interconnect to substrate.

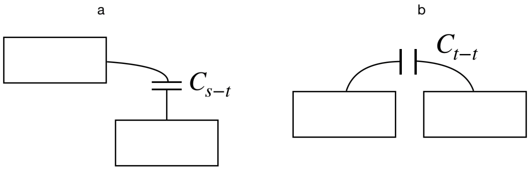

Figure 17: The fringe capacitances between interconnects. **a** shows the coupling between the side of an interconnect to the top of another. **b** shows the fringe capacitance between the top of two interconnects. In this work the couplings are also modelled between interconnects and the substrate.

The cell also has capacitive couplings to neighbouring cells. These included interconnect to interconnect couplings as well as nanowire-to-nanowire capacitance. All these capacitances were calculated using equation (5). The total capacitance from neighbouring and fringe couplings for the BL can then be expressed as:

$$\begin{aligned} C_{f,n} = &\left(\sum_{\{2\mu\mathrm{m}\}} C_{t-t} \right) \\ &+C^{bs}_{s-t} + C^{bw}_{s-t} + 2C_{BL-BL} + 4C_{NW-NW}, \end{aligned} \tag{6}$$

where the sum is within a neighbourhood of 2 µm, C^{bs}_{s-t} the fringe capacitance between the BL and the substrate, C^{wb}_{s-t} the fringe capacitance between BL and WL, C_{BL-BL} the capcitance between two BL interconnects and C_{NW-NW} the capacitance between the neighbouring nanowires.

With all the capacitances defined, the impedance for each cell could be calculated as follows:

$$Z_C = \frac{1}{i2\pi fC}, \tag{7}$$

where f is the frequency of the voltage pulse, C the capacitance responsible for the impedance and i the imaginary unit. The impedance from the WL and the BL are different. The capacitances included in the calculation of the impedance for the BL are (cf the circuit in figure 16): C_{BL-S}, C_{BL-WL}, C_{RRAM}, C_{gd} och C_{gs}. The BL also has a resistive part R_{RRAM}. The total impedance experienced by a pulse sent to the BL is ("//" denotes parallel coupling):

$$\begin{aligned} Z_{BL} = Z_{C_{BL-S}}//Z_{C_{BL-WL}}//Z_{C_{f,n}} \\ //((Z_{C_{RRAM}}//Z_{R_{RRAM}}) + Z_{C_{gd}} + Z_{C_{gs}}). \end{aligned} \tag{8}$$

3.2.5 Integration of 1T1R Cell to the Array Model

The 1T1R cell has slightly different connections than the passive arrays. The cell is a three-terminal device, while the passive array cell is a two-terminal device (cf. figure 6). In order to incorporate a three-terminal device into a two-terminal structure, the WL and BL needed to be modified. Since the biasing of the WL for the 1T1R model modulates the FET, its function is different than in the passive array model. This meant that the programming of the 1T1R array would have to be amended. To solve this discrepancy, each node resistance R_{ij} was set to $Z_{mod,i}$, where i denotes the BL row number of the array. The WL of the array then effectively acts as the substrate level.

One last obstacle was the computational cost of large arrays. One run for 32x32 array took about one second, but for intermediate arrays (64x64) one simulation ran for minutes, and for even larger arrays it would take hours and days. To be able to simulate larger arrays, the resistances R_{ij} and line resistances were first calculated for smaller sub-arrays, and later inserted for a simulation of a 32x32 array as effective resistances. This reduced the run-time scale to seconds. The code is provided in the appendix.

3.2.6 Array Performance and Figures of Merit

The performance of the array can be characterised by a number of key figures of merit.Because of the high number of cells included in an array, analysis of the worst case cell is a natural way to go. The worst case cell suffers from the worst signal losses (in this work the cell in the lower right corner). For both write and read operations the voltage of the array needs to access all the cells. This will be referred to as the access voltage. Furthermore, the resistance of a cell needs to be observed reliably in a read operation. The read margin is calculated by

$$RM = V_{LRS} - V_{HRS}, \tag{9}$$

i.e. the measured voltage difference between the cell in low resistance state and high resistance state.

Another interesting part is to relate the observed data in relation to array simulations. In particular the probability of switching a cell is of interest for application purposes, specifically in HD computing where stochastic distributions of resistances is a sought after property of the array [8]. To this end, the access voltage of the array can be mapped to the switching probability. In particular, the average switching probability over a given array size was calculated.

4 Results

4.1 Experimental Part

The pulses used on the 1T1R device were long enough to be considered as DC sweeps. For each set and reset pulse, the DC characteristics were extracted. The device was first pulsed up to one million times with a pulse amplitude of 1.5 V, and after that an additional million times with 1 V pulse amplitude. A total of 60 set and reset measurements were extracted for 1 million cycles. Figures 18 and 19 show the resulted IV plot for the device with a stop voltage of 1.5 V and 1 V respectively. The thickness of the lines are growing along with successive sweeps. The forming is not included in the figures, since the device was pulsed several times until a functioning switching was achieved. The RRAM device showed clear switching characteristics, although with a distinct spread to the switching voltages. There is more information that can be gathered from these plots.

For both runs at a region at around -0.45 V a sudden downward spike of current can be seen, which can be explained as a capacitive current due to the long wires used in the experiment. A comparison of the IV curves reveals some differences and similarities. One difference between the runs that is the apparent spread and stability of the current. For 1.5 V stopping voltage, the switching voltages have a larger spread than for 1 V stop voltage. For a measure of this difference, figures 20 and 21 provide the histogram of both runs. The mean (notated as μ) and standard deviation (σ) were calculated and are also included in the plot.

This difference can be attributed to two main reasons, both of which include the filament. One explanation has to do with the order of the measurements. As mentioned in the theory section, the filament of an ITO-based RRAM is consisted of oxygen vacancies, that are assumed to migrate due to an electric field. Because of this, the filament formation might need many pulses before a stable switching behaviour is achieved. Another possible explanation has to do with the stopping voltage. The smaller stopping voltage gives a lower interaction with the filament. This means that the rupture and resetting of the filament could be under less stress, and thus giving less spread and lower switching voltages.

Another feature that sticks out for this particular RRAM is the occurrence of two set/reset events. This quality is most visible in figure 19 but can also be noticed in figure 18. Figure 21 also strengthens this bu having a bi-modal distribution for the set voltage. While the mechanism regarding the formation and evolution of the filament lies outside the scope of this work, it can be mentioned that for this device may consist of two filaments or a filament with two main paths.

The switching probability of both runs can be viewed in figures 22 and 23, for 1.5 V and 1 V respectively. Both reset and set voltages are positive for easier comparison. For both set and reset probabilities the highest measured switching voltage is marked with vertical dashed lines. The top set voltage for 1.5 V lies at around 0.95 V, while the top reset voltage lies at around 0.87 V. For the 1 V run the top set voltage lies at 0.87 V while the top reset voltage is at 0.54 V. For both runs one observation that stands out is the clear asymmetry between the set and reset voltages. This asymmetry is larger for 1 V stop voltage than for 1.5 V. Furthermore, the cumulative probability for 1 V has a steeper form. indicating a more defined switching voltage with less spread, which is corroborated by the histograms and the IV curves.

The endurance of this setup is seen in figures 24 and 25 for stop voltages 1.5 V and 1 V respectively. From this data the mean high resistance and low resistance states was calculated and presented in table 4.

Table 4: Table showing the mean and standard deviation of both low and high resistance modes of for both 1.5 V and 1 V stop voltages

Voltage	State	Mean	Std
1.5	LRS	58 kΩ	32 kΩ
1.5	HRS	46 MΩ	43 MΩ
1	LRS	89 kΩ	16 kΩ
1	HRS	107 MΩ	164 MΩ

First and foremost the device showed remarkably good endurance, with the total number of pulses exceeding 2 million cycles. Both figures and table 4 show that the resistance window is large enough to distinguish the states of the device, with a resistance ratio of $R_{HRS}/R_{LRS} \sim 10^3$. The spread of the device is significantly large, with standard deviations reaching the same order of magnitude as the mean. For both runs the spread is smaller for the LRS than for the HRS, while for 1 V stop voltage the LRS spread is smaller along with a larger HRS spread.

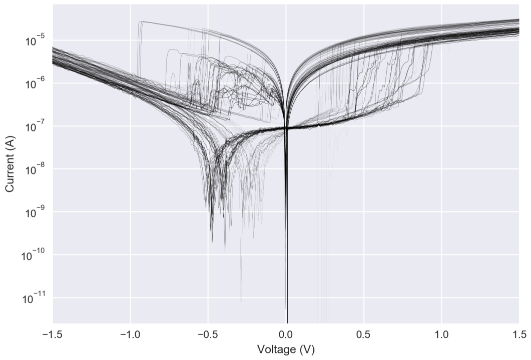

Figure 18: DC sweeps of the initial one million cycles with stop voltage 1.5 V. The number of cycles between the sweeps range from 100 to 100 000. Thicker lines indicate a later stage in the cycling process.

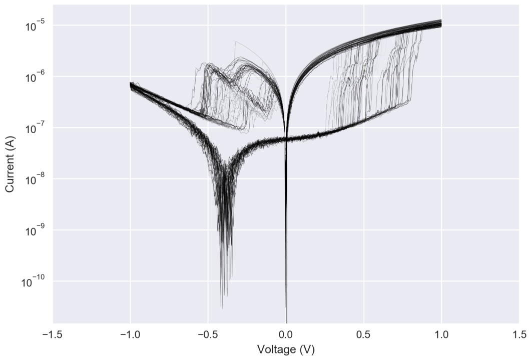

Figure 19: DC sweeps with stop voltage 1 V. The number of cycles between the sweeps range from 100 to 100 000. Thicker lines indicate a later stage in the cycling process.

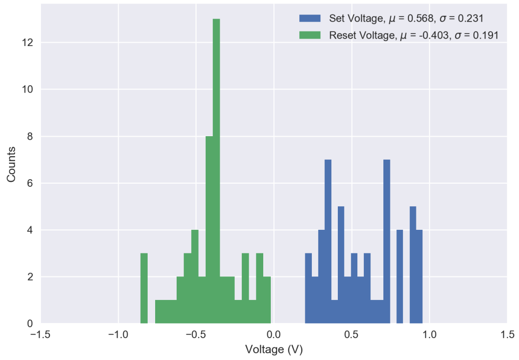

Figure 20: Histogram of the extracted set and reset voltages of the initial one million cycles for stop voltage 1.5 V. The mean set voltage was found to be $\mu = 0.568$ V with standard deviation $\sigma = 0.23$ V. The mean reset voltage was found to be -0.403 V ± 0.191 V

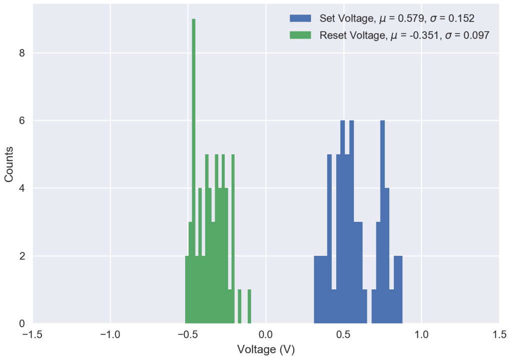

Figure 21: Histogram of the extracted set and reset voltages for stop voltage 1 V. The mean set voltage was found to be $\mu = 0.579$ V with standard deviation $\sigma = 0.152$ V. The mean reset voltage was found to be -0.351 ± 0.097 V.

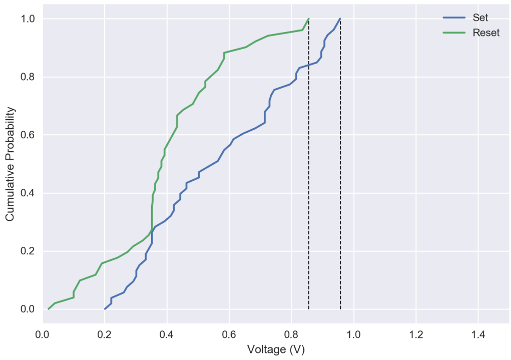

Figure 22: Cumulative switching probability of the initial one million cycles with stop voltage 1.5 V.

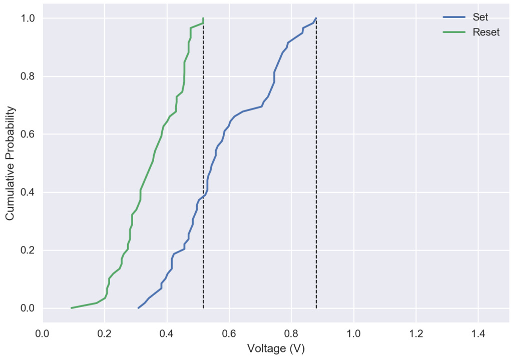

Figure 23: Cumulative switching probability with stop voltage 1 V.

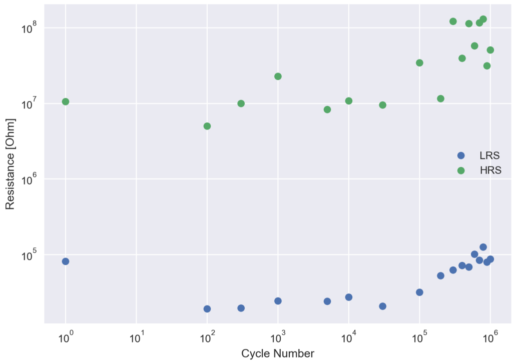

Figure 24: Endurance measurements for the initial one million cycles with stop voltage 1.5 V.

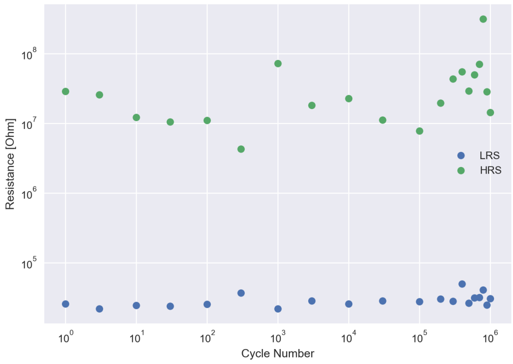

Figure 25: Endurance measurement with stop voltage 1 V.

4.2 Simulations

All the parameters for the simulations are listed in table 3, unless indicated otherwise. Worst case cell analyses for the access voltage and read margin can be seen in figures 26 - 31, with varying parameters. Figure 26 shows the access voltage versus array size for technology node 35 nm. For high-speed applications, the frequencies will reside in the GHz regime, thus the pulse width is varied from 10 ns to 150 ns. The plot shows a distinct decline in performance with a lower pulse width. This is expected, as a fast switching give rise to higher losses due to parasitic elements. For larger array sizes (> 10000 bits) the differences between the runs becomes smaller. Figure 27 shows the same type of analysis with varying resistivity of the interconnects and with the pulse width fixed at 10 ns. The resistivity is varied from the bulk resistivity of copper (16.8 Ω nm) to ten times the bulk resistivity. The plot shows a large decline in performance with higher resistivity. The difference between runs is not uniform, with the improvement being smaller for high resistivities. For array sizes > 1000 bits the access voltage decreases exponentially.

A similar analysis for the read margin can be seen in figures 28 and 29 for 35 nm technology node. In order to choose a reasonable reading voltage so as to not switch a cell unintentionally, figure 23 was considered. From this the reading voltage was chosen to be around 150 mV with positive polarization. The RM was calculated using equation (9). In figure 28 the sense resistance is varied from 100 Ω to 100 kΩ. This shows the sense resistance dependence of the read margin, and provides a sanity-check that the model works as expected. For 100 kΩ the sense resistance is in the same order of magnitude as the LRS of the RRAM, and was thus accepted as a reasonable value for the remaining simulations. Figure

28 shows the RM with varying pulse widths. For small arrays (< 1000 bits) the RM declines slowly, transitioning into an exponential decrease with array size. The RM is significantly improved for larger pulse widths.

Figures 30 and 31 show access voltage and read margin analysis, respectively, for different technology nodes. For these runs all the parameters are fixed as indicated in table 3. The 100 nm technology node shows the best performance, while the 35 technology node shows the worst. This result is expected, as arrays with tighter devices give rise to both higher resistivity of interconnects and larger parasitic elements. To get a figure of how the signal losses are distributed between the different parasitic elements the conductance (inverse of resistance) was calculated for one cell. The results are presented in table 5 for each technology node. The parasitic elements that are included are BL-to-BL, BL-to-WL, BL-to-Substrate capacitances. Most of the leakage is due to BL-BL capacitances. This leakage goes down if technology node is compared with 100 nm node. The next largest contributors of losses are the parasitic elements of the NW. These losses increase with technology node. This result is expected, as the BL-to-BL portion of the losses decrease with node size. The BL-to-substrate capacitances make up a small portion of the total losses. The losses due between BL and WL take similarly only make up a marginal portion of the total losses. This result shows that there are no significant losses due to WL parasitic elements, although it goes up for one order of magnitude from 35 nm to 100 nm technology node. This increase is not something that has been looked into at this time. A possible cause might be the relative sizes of the different parameters in use, which are slightly different for the three node sizes investigated.

The veracity of the model could be further investigated by looking at how the leakage distribution behaves with pulse width. Since the small signal model simulates signal losses, the losses due to parasitic elements should become less significant as the pulse width is increased. Figures 32 and 33 show the same parasitic elements with increasing pulse width for technology node 35 nm. The simulations show that the BL-to-BL and NW stand for the majority of the current leakage. For larger pulses the BL-to-BL parasitic capacitances become less significant, while the NW leakage become more significant, approaching 100 % for pulses > 1µs. Since the NW portion of the small signal model includes less capacitive elements and more resistive elements, this behaviour is expected. Moreover, the BL-WL and BL-S parasitic elements play a less important role even for 10 ns pulses, and decline rapidly with pulse width.

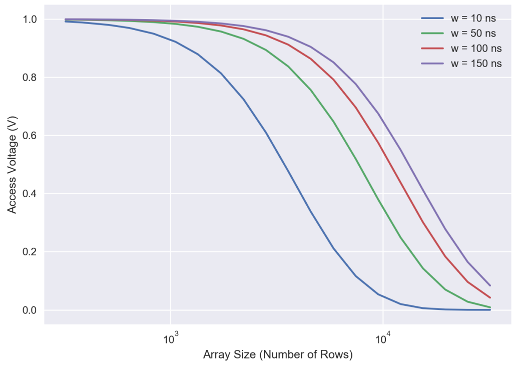

Figure 26: Write Access voltage over the worst case cell vs array size for different pulse widths. The resistivity is fixed at 5.8×10^{-8} $\Omega\,$m and the technology node is 35 nm.

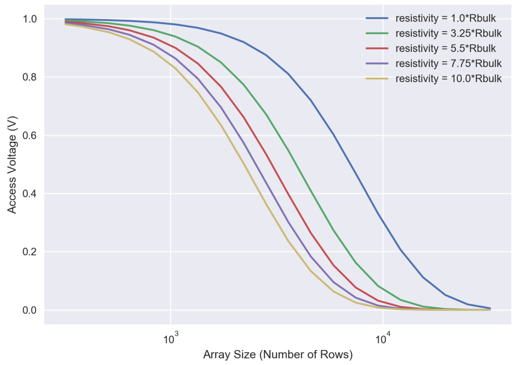

Figure 27: Access Voltage over the worst case cell vs array size for different resistivities of the interconnects. The pulse width is fixed at 10 ns and the technology node is 35 nm.

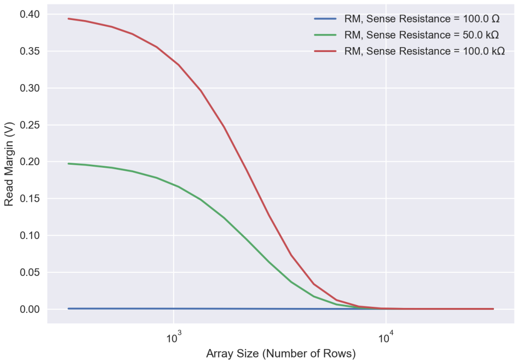

Figure 28: Read Margin analysis for the worst case cell, vs array size and for different sense resistances. The pulse width is fixed at 10 ns and resistivity $5.8{\times}10^{-8}$ $\Omega\,$m. Technology node is 35 nm.

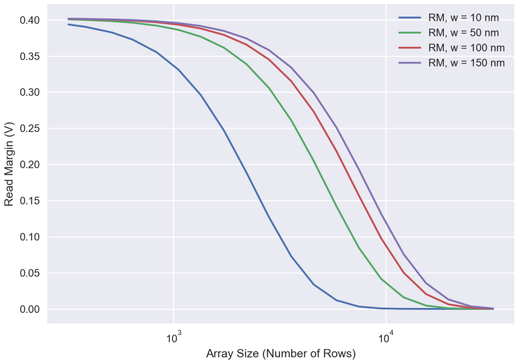

Figure 29: Read Margin analysis for the worst case cell, vs array size and for different pulse widths. The resistivity is fixed at $5.8{\times}10^{-8}$ $\Omega\,$m. Technology node is 35 nm.

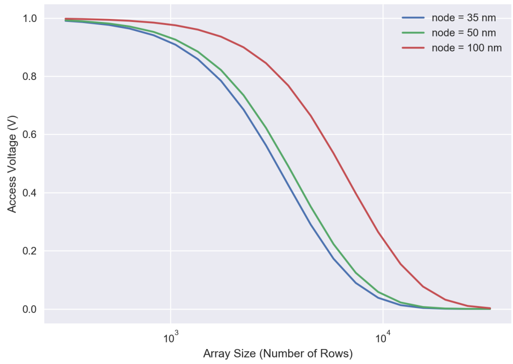

Figure 30: Access voltage for the worst case cell, vs array size and for different technology nodes. The pulse width is fixed at 10 ns and resistivity $5.8{\times}10^{-8}$ $\Omega\,$m

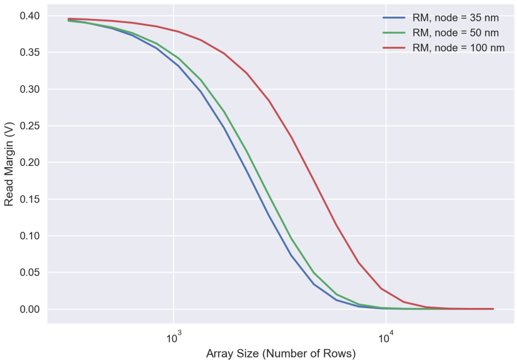

Figure 31: Read margin analysis for the worst case cell, vs array size and for different technology nodes. The pulse width is fixed at 10 ns and resistivity $5.8{\times}10^{-8}$ $\Omega\,$m

Table 5: Table showing the leakage percentage for each parasitic element of 1T1R cell. The pulse width is set at 10 ns.

Parasitic Element	35 nm Node (%)	50 nm Node (%)	100 nm Node (%)
BL to BL	91.33	90.76	85.80
NW	8.47	9.06	13.47
BL to Substrate	0.12	0.18	0.38
BL to WL	0.08	0.01	0.35

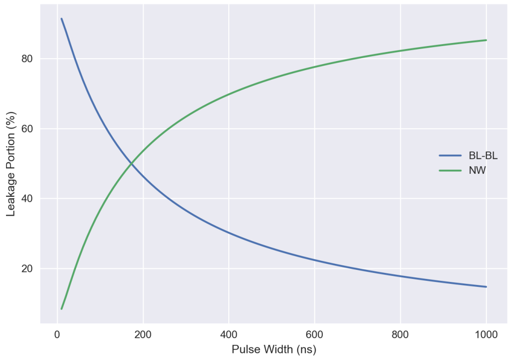

Figure 32: Leakage portion of the parasitic elements due to BL-BL couplings (top level) and NW leakage, as a function of pulse width for technology node 35 nm

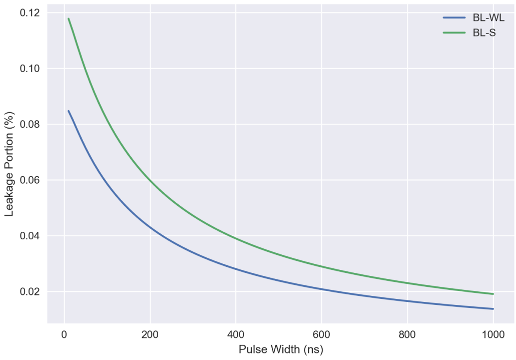

Figure 33: Leakage portion of the parasitic elements due to BL-WL and BL-Source couplings, as a function of pulse width for technology node 35 nm

The distribution of access voltage for array size 2048 bits is given in figure 34. For this size the voltage loss was large enough to be visualised. The cells furthest to the right have the largest loss of voltage, while the ones on the left have the highest. There are no significant losses that arise due to the vertical position of a cell. The access voltage was mapped to switching probabilities from the experimental part, and can be seen in figures 35-36 for array size 2048 bits. The uneven distribution between set and reset operations from experiments is, naturally, reflected in the probability maps. Reset operations can be seen as having a distinctly better performance than set operations, with a uniform probability = 1 over the whole array. To get a figure of this performance, the average switching probability, with standard deviation, for a given array size was calculated, which can be seen in figures 37 and 38 for array sizes 1024 and 2048 bits respectively. The plots are reminiscent of figure 23, which is expected. In particular the asymmetry between set and reset operations is clear. The probability of switching decreases with array size, while the spread increases with array size. The spread of the probabilities is low for large and small voltage amplitudes, while increasing for intermediate amplitudes. This is due to the fact that a larger array has a larger distribution of access voltage, and as such the probability of switching a random cell has a wider range.

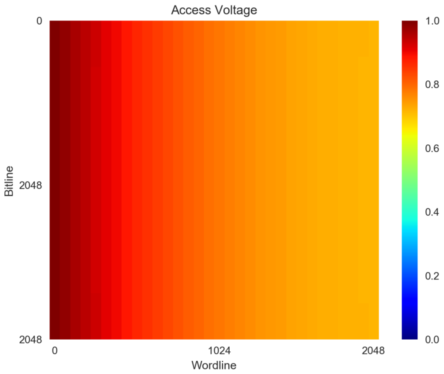

Figure 34: Access voltage bitmap for a 2048 bit array and voltage amplitude 1 V. The pulse width is fixed at 10 ns and resistivity 5.8×10^{-8} Ω m

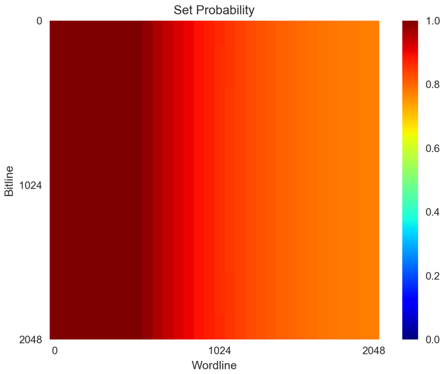

Figure 35: Set probability map for a 2048 bit array. The pulse width is fixed at 10 ns and resistivity 5.8×10^{-8} Ω m

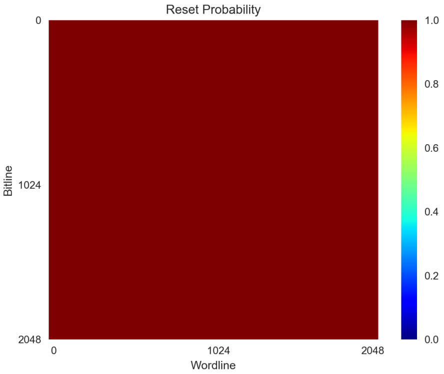

Figure 36: Reset probability map for a 2048 bit array

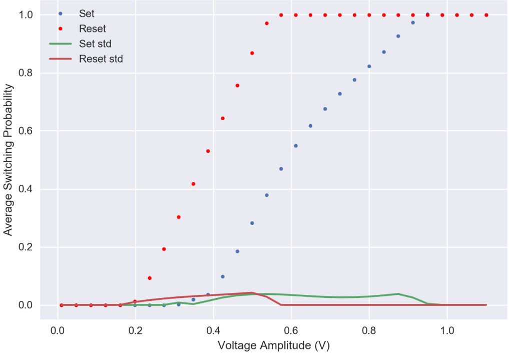

Figure 37: Average switching probability and standard deviation for array size = 1024 bit

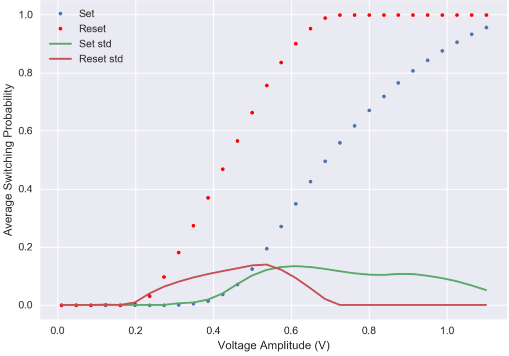

Figure 38: Average switching probability and standard deviation for array size = 2048 bit

5 Discussion

The work presented in this thesis can be separated into two distinct sections, where in the first part characterisation of RRAM devices was conducted, and in the second part, modelling and simulations were performed using realistic values and data taken from the measured RRAM devices in the first part.

5.1 Experimental Part

For the experimental part the goal was to set up a 1T1R structure that was functional enough to show switching behaviours that are distinct to RRAM, as well as good endurance. The RRAM material stack that was used had been previously developed in LNL with HfO_2 as oxide layer, TiN TE and ITO BE. The FET was also developed previously and consisted of arrays if NW FET's. Several devices were examined, but the device presented in this work yielded the best results. Both the FET, RRAM and integration of both devices showed clear characteristics, yielding a wide resistance widow and strong endurance. The device was measured for 1.5 V stop voltage and 1 V stop voltage. The RRAM had a mean LRS of 58-89 kΩ and a HRS of 46-107 MΩ, showing an excellent resistance window. The spread of resistances is lower for LRS, which is due to the compliance set by the transistor. For the HRS the spread of the resistances are within a good margin to allow for single-bit storage.

Endurance measurements were made on the structure, which exceeded 2 million switching cycles, which is a result that is impressive in itself, though it should be mentioned that several other devices of the same RRAM die showed poor endurance. While this is mainly a question of the yield of the specific RRAM stack, it is hard to generalise this result. Even so, an excellent endurance using a NW transistor and a thin high-k dielectric film is generally a hard task to achieve. Compared to DRAM and SRAM, where > 1015 endurance has been achieved, RRAM lags behind [24]. What is important, though, is that RRAM is non-volatile. Compared to contemporary non-volatile flash memory, which has up to 10^{12} endurance [24]. Although this figure is higher, RRAM is still in its infancy and holds great promise with future improvements. Furthermore, RRAM performance does not degrade with scaling, due to the small size of the filament, while flash memory is reliant on floating gates, whose performance does degrade with scaling.

There are a few other things that have been revealed in the endurance data. If we turn to figure 24, both the HRS and LRS of the cell increase close to one million cycles. The increase of LRS in particular might indicate that the device approaches switching failure. This is in comparison to figure 25 which does not show this trend. This shows that a too high switching voltage might induce switching failure. This result is in agreement with [25], where it is showed that stopping voltage is the primary reason for switching failure, as opposed to pulse widths. This failure of switching with higher stop voltages can be explained by looking to the workings of the ITO-HfO_2 -TiN composition. As the oxygen-vacancy filament is formed, oxygen ions migrate towards the oxygen rich ITO top electrode. The ITO is eventually depleted of oxygen-vacancies and reaches a saturated self-compliance of the current [26]. The larger the stop voltage becomes, the further in the oxygen ions migrate in the ITO and, as a result, they become more dispersed. In order to reverse this dispersion a larger backward bias is needed to rupture the filament. Since the same magnitude of the voltage was used in our experiment, the LRS becomes higher, which might lead to failure.

The cumulative switching probability for both set and reset operations were extracted. For this device the striking difference between set and reset operations was the asymmetry. This shows the effect of stopping voltage has on the stability of the filament. A too large stop voltage might give off unintentional switching, as figure 22 shows switching voltages as low as a few mV's. For application purposes this would be a significant parameter to consider, since it has a direct effect on the biasing.

5.2 Simulations

The structure and goals of the simulation part of the work was evolving during the project, but the initial plan for the simulations was to set up a working framework to simulate the performance of RRAM crossbar arrays using experimental data. The investigation started from simulation in ADS (no presentable models were made). While ADS was successful to investigate compact RRAM models ([27]), it was eventually decided that for the purpose of signal loss for large arrays it would be more fruitful to use Python as the framework, since the workings of filament- and conduction-mechanics were seen as irrelevant for large scale simulations.

A model for passive arrays from [15] was used specifically as a starting point, where Kirchoff's continuity equations were used to calculate the voltage propagation. The model had no selector device, and simulated the sneak-path currents that occur. This model was included in the work as a motivation for why to include selectors in the first place. The cells were modelled by developing an integrated NWRRAM small-signal model, which included both parasitic and non-parasitic capacitances. Moreover, analytical expressions for fringe capacitances were incorporated in the model.

The system level passive array model and cell-level small-signal model were joined to form a 1T1R array that was used to simulate signal losses. Within this framework a worst-case scenario analysis was done for access voltage and read margin while varying pulse width, resistivity and node size. The results show that for array sizes above 1000 bits there is a significant loss for high speed or high density arrays. This result shows that the array size is high and stands to compete with current technology array sizes [5], and shows that 1T1RNW arrays show great potential even if capacitances disturb the signal for larger arrays. On the other hand, it is worth noting that RRAM arrays for in-computing structures, which require high speed signalling, might meet new difficulties for higher density architectures, like 3D stacked arrays. On the other hand, for purposes which do not require ultra fast operations, like pure memory applications, denser and larger arrays would be less hindered by NW technology.

The access voltage was mapped to a probability map, using the measured switching probabilities. To get a quantitative figure of this distribution, the average switching probability of a given array size was calculated and it was showed that the probability decreased with array size, while the spread of switching increased with array size. This type of analysis could prove useful for application purposes that include stochastic switching of cells, which is a property that is considered important for hyper-dimensional computing applications [7, 8].

The models used in the simulations are subject to some sources of error. Most notably, the capacitances used in a run are entirely analytical and might be over-estimated. Numerical simulations concerning capacitances would give a more detailed picture, and could be used to compare the analytical results of this work for small arrays. Another uncertainty concerns the combination of the three-port 1T1R cell model and the two-port array model. The pathway of the true signal propagation is more complex than modelled in the array. Still, this shows no significant losses in the WL arises in these structures.

The project could be improved in a number of ways, which were considered but not implemented due to lack of time. One thing would be to compare the results with dedicated circuit-design simulation frameworks, like ADS. This comparison would elucidate some possible errors that the analytical expressions might give rise to. Another thing that could be improved is the RRAM device that were chosen in the experiment. To get a sounder correspondence between experiment and simulation, an actual integrated NWRRAM 1T1R cell should be measured. At the time of writing this thesis, such structures are in development at Lund NanoLab.

5.3 Summary and Outlook

This project set out to examine the workings of RRAM arrays using experimental data. The general theory of filament formation and switching mechanism for oxide-based RRAMs was given. Furthermore, the technology of crossbar-arrays and their importance for applications such as neuromorphic circuits and hyper-dimensional computing was discussed. A 1T1R device was characterised by applying voltage pulses. The devices showed excellent endurance performance, exceeding 2 million cycles. Moreover, it had a wide resistance window.

The second part of this work used the measured data to show how this type of device would behave in a nano-wire array environment. The simulations showed that even for small technology nodes large arrays ($>$ 1000 bits per row) could reliably be constructed, with good access voltages and switching probabilities.

Specifically, the results show that it is possible to integrate $>$ Mbit sub-arrays with low-voltage 10 ns pulses, providing a solid foundation for larger Gbit memory sizes, which are comparable with current DRAM and NAND technology. The advantage of RRAM will be, for one, its excellent scalability . Secondly, the non-volatility and low-voltage operation will make RRAM an ultra-low-power choice for random access applications. This stands in contrast to the constant power supply needed to refresh DRAM memory cells. As such, the outlook for the future of RRAM technology is promising.

References

[1] T.-C. Chang, K.-C. Chang, T.-M. Tsai, T.-J. Chu, and S. M. Sze, "Resistance random access memory," *Materials Today*, vol. 19, no. 5, pp. 254–264, 2016.

[2] D. Ielmini, "Brain-inspired computing with resistive switching memory (rram): Devices, synapses and neural networks," *Microelectronic Engineering*, vol. 190, pp. 44–53, 2018.

[3] N. G. Orji, M. Badaroglu, B. M. Barnes, C. Beitia, B. D. Bunday, U. Celano, R. J. Kline, M. Neisser, Y. Obeng, and A. Vladar, "Metrology for the next generation of semiconductor devices," *Nature electronics*, vol. 1, no. 10, pp. 532–547, 2018.

[4] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, 1971.

[5] J. E. Ayers, *Digital integrated circuits: analysis and design.* CRC Press, 2003.

[6] M. Suri, *Advances in Neuromorphic Hardware Exploiting Emerging Nanoscale Devices*, vol. 31. Springer, 2017.

[7] H. Li, T. F. Wu, A. Rahimi, K.-S. Li, M. Rusch, C.-H. Lin, J.-L. Hsu, M. M. Sabry, S. B. Eryilmaz, J. Sohn, *et al.*, "Hyperdimensional computing with 3d vrram in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition," in *2016 IEEE International Electron Devices Meeting (IEDM)*, pp. 16–1, IEEE, 2016.

[8] A. Rahimi, T. F. Wu, H. Li, J. M. Rabaey, H.-S. P. Wong, M. M. Shulaker, and S. Mitra, "Hyperdimensional computing nanosystem," *arXiv preprint arXiv:1811.09557*, 2018.

[9] R. Gastaldi and G. Campardo, *In Search of the Next Memory: Inside the Circuitry from the Oldest to the Emerging Non-Volatile Memories.* Springer, 2017.

[10] S. K. Vishwanath, H. Woo, and S. Jeon, "Effect of dysprosium and lutetium metal buffer layers on the resistive switching characteristics of cu–sn alloy-based conductive-bridge random access memory," *Nanotechnology*, vol. 29, p. 385207, jul 2018.

[11] M. Uenuma, Y. Ishikawa, and Y. Uraoka, "Joule heating effect in nonpolar and bipolar resistive random access memory," *Applied Physics Letters*, vol. 107, no. 7, p. 073503, 2015.

[12] D. Kumar, R. Aluguri, U. Chand, and T.-Y. Tseng, "Conductive bridge random access memory characteristics of SiCN based transparent device due to indium diffusion," *Nanotechnology*, vol. 29, p. 125202, feb 2018.

[13] M. Åstrand, "Hfo2 and ito resistive random-access memory," Master's thesis, Lund University, 2020.

[14] A. Chen, "A review of emerging non-volatile memory (nvm) technologies and applications," *Solid-State Electronics*, vol. 125, pp. 25–38, 2016.

[15] A. Chen, "A comprehensive crossbar array model with solutions for line resistance and nonlinear device characteristics," *IEEE Transactions on Electron Devices*, vol. 60, no. 4, pp. 1318–1326, 2013.

[16] J. Woo, X. Peng, and S. Yu, "Design considerations of selector device in cross-point rram array for neuromorphic computing," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4, IEEE, 2018.

[17] J. Zhou, K.-H. Kim, and W. Lu, "Crossbar rram arrays: Selector device requirements during read operation," *IEEE Transactions on Electron Devices*, vol. 61, no. 5, pp. 1369–1376, 2014.

[18] S. Kim, J. Zhou, and W. D. Lu, "Crossbar rram arrays: Selector device requirements during write operation," *IEEE Transactions on Electron Devices*, vol. 61, no. 8, pp. 2820–2826, 2014.

[19] L. Zhang, S. Cosemans, D. J. Wouters, G. Groeseneken, M. Jurczak, and B. Govoreanu, "Selector design considerations and requirements for 1 sir rram crossbar array," in *2014 IEEE 6th International Memory Workshop (IMW)*, pp. 1–4, IEEE, 2014.

[20] Y.-X. Deng, P. Huang, B. Chen, X.-L. Yang, B. Gao, L.-F. Liu, J.-F. Kang, and X.-Y. Liu, "Size analysis of multilevel rram array and optimization of device and circuit characteristics," in *2012 IEEE 11th International Conference on Solid-State and Integrated Circuit Technology*, pp. 1–3, IEEE.

[21] K.-M. Persson, M. S. Ram, M. Borg, and L.-E. Wernersson, "Investigation of reverse filament formation in ito/hfo2-based rram," in *2019 Device Research Conference (DRC)*, pp. 91–92, IEEE, 2019.

[22] K.-M. Persson, *Nanowire Transistors and RF Circuits for Low-Power Applications*. Lund University, 2014.

[23] A. Bansal, B. C. Paul, and K. Roy, "An analytical fringe capacitance model for interconnects using conformal mapping," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 12, pp. 2765–2774, 2006.

[24] T.-C. Chang, K.-C. Chang, T.-M. Tsai, T.-J. Chu, and S. M. Sze, "Resistance random access memory," *Materials Today*, vol. 19, no. 5, pp. 254–264, 2016.

[25] S. Balatti, S. Ambrogio, Z.-Q. Wang, S. Sills, A. Calderoni, N. Ramaswamy, and D. Ielmini, "Pulsed cycling operation and endurance failure of metal-oxide resistive (rram)," in *2014 IEEE International Electron Devices Meeting*, pp. 14–3, IEEE, 2014.

[26] C.-H. Pan, T.-C. Chang, T.-M. Tsai, K.-C. Chang, T.-J. Chu, C.-C. Shih, C.-Y. Lin, P.-H. Chen, H. Wu, N. Deng, *et al.*, "Ultralow power resistance random access memory device and oxygen accumulation mechanism in an indium–tin-oxide electrode," *IEEE Transactions on Electron Devices*, vol. 63, no. 12, pp. 4737–4743, 2016.

[27] Z. Jiang, Y. Wu, S. Yu, L. Yang, K. Song, Z. Karim, and H.-S. P. Wong, "A compact model for metal–oxide resistive random access memory with experiment verification," *IEEE Transactions on Electron Devices*, vol. 63, no. 5, pp. 1884–1892, 2016.

A Supplementary formulas

A.1 Fringe Capacitances

Two formulas for fringe capacitances are used in this work which have been derived in [23]. The first one goes from the side of the top interconnect to the top of the lower interconnect and is given by:

$$C_{s-t} = \frac{\epsilon_{di}}{\pi/2} \ln \left[\frac{H + T' + \sqrt{S^2 + T'^2 + 2HT'}}{S + H} \right], \tag{10}$$

where ϵ_{ox} is the relative permittivity of the dielectric (Silicon dioxide in this work), H the height difference between the bottom of the top electrode and the top of the bottom electrode, S the horizontal spacing between the electrodes and $T' = \eta T$ the effective thickness of the top electrode of thickness T, with a correction factor η. The capacitance between the top of two interconnects is given by

$$C_{t-t} = \frac{\epsilon_{di} W \alpha \left(\ln\left(1 + \frac{2W}{S}\right) + e^{-\frac{S+T}{3S}} \right)}{W \pi \alpha + (H + T) \left(\ln\left(1 + \frac{2W}{S}\right) + e^{-\frac{S+T}{3S}} \right)}, \tag{11}$$

where all the parameters are the same, W the width of the interconnect and α a correction parameter. For details concerning these formulas, the reader is referred to [23].

B Python Code

B.1 Passive Array Model

```
”””
Passive Array model returns the access voltage of each node for a RRAM crossbar array.
Can be simulated for one run, or by get_bit_map: this function goes through a
simulation for each node and returns a map of the accessed voltage for the each node as
For array sizes > 32 get_bit_map will take a significant amount of time.

Model based on:

Chen, An. ”A comprehensive crossbar array model with solutions
for line resistance and nonlinear device characteristics.” IEEE
Transactions on Electron Devices 60.4 (2013): 1318−1326.

Article can be found on: https://ieeexplore.ieee.org/abstract/document/6473873
”””
import matplotlib.pyplot as plt
import numpy as np
import time
import warnings
from progress.bar import Bar
warnings.filterwarnings(’ignore’)
plt.style.use(’seaborn−dark’)


def A_i(RS_WL1, RS_WL2, R_WL, R, i, n):
    lower_diagonal = np.ones(n−1)∗(−1)/R_WL
    upper_diagonal = np.ones(n−1)∗(−1)/R_WL
    diagonal = (1/R[i]) + (2/R_WL)
    diagonal[0] = 1/RS_WL1[i] + 1/R[i][0] + 1/R_WL
    diagonal[−1] = 1/RS_WL2[i] + 1/R[i][n−1] + 1/R_WL
    Ai = np.zeros((n, n))
    u = np.diag(upper_diagonal, k=1)
    l = np.diag(lower_diagonal, k=-1)
    d = np.diag(diagonal)
    Ai += u+l+d

    return Ai

def A_matrix(m, n, RS_WL1, RS_WL2, R_WL, R):
    A = np.zeros((m*n, m*n))
    k = 0
    for i in range(m):
        A[k:k+n, k:k+n] = A_i(RS_WL1, RS_WL2, R_WL, R, i, n)
        k += m
    return A

def B_i(R, n, i):
    diagonal = np.ones(n)*(-1)/R[i]

    Bi = np.diag(diagonal)

    return Bi

def B_matrix(m, n, R):
    B = np.zeros([m*n, m*n])
    k = 0
    for i in range(m):
        B[k:k+n, k:k+n] = B_i(R, n, i)
        k += m
    return B

def C_j(m, n,  R, j):
    Cj = np.zeros([m, n*m])
    for i in range(0, m):
        Cj[i][n*(i) + j] = 1/R[i][j]

    return Cj

def C_matrix(m, n, R):

    C = C_j(m, n, R, 0)
    for j in range(1, n):
        C_old = C
        C_add = C_j(m, n, R, j)
        C = np.concatenate((C_old, C_add), axis=0)

    return C

def D_j(m, n, RS_BL1, RS_BL2, R_BL, R, j):
    Dj = np.zeros([m, n*m])
    for i in range(m):
        if i == 0:
            Dj[i][j] = (-1/RS_BL1[j] -1/R_BL -1/R[i][j])
            Dj[i][n + j] = 1/R_BL
        elif (1<=i<=(m-2)):
            Dj[i][n*(i-1) + j] = 1/R_BL
            Dj[i][n*(i-0) + j] = (-1/R_BL -1/R[i][j] -1/R_BL)
            Dj[i][n*(i+1) + j] = 1/R_BL
        elif (i == m-1):
            Dj[i][n*(i-1) + j] = 1/R_BL
            Dj[i][n*(i-0) + j] = (-1/RS_BL2[j] -1/R[i][j] -1/R_BL)

    return Dj

def D_matrix(m, n, RS_BL1, RS_BL2, R_BL, R):
    #j = 0
    D =D_j(m, n, RS_BL1, RS_BL2, R_BL, R, 0)
    for j in range(1, n):
        D_old = D
        D_add = D_j(m, n, RS_BL1, RS_BL2, R_BL, R, j)
        D = np.concatenate((D_old, D_add), axis=0)

    return D

def E_Wi(VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2, n, i):
    E = np.zeros(n)
    E[0] = VAPP_WL1[i]/RS_WL1[i]
    E[-1] = VAPP_WL2[i]/RS_WL2[i]
    return E

def E_Bj(VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2, m, j):
    E = np.zeros(m)
    E[0] = -VAPP_BL1[j]/RS_BL1[j]
    E[-1] = -VAPP_BL2[j]/RS_BL2[j]
    return E

def E_W(VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2, m, n):
    Ew = E_Wi(VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2, n, i = 0)
    for i in range(1, m):
        E_old = Ew
        E_add = E_Wi(VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2, n, i)
        Ew = np.concatenate((E_old, E_add), axis=0)

    return Ew

def E_B(VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2, m, n):
    Eb = E_Bj(VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2, m, j = 0)
    for j in range(1, n):
        E_old = Eb
        E_add = E_Bj(VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2, m, j)
        Eb = np.concatenate((E_old, E_add), axis=0)

    return Eb

def E_matrix(m, n, VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2,
      VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2):
    EW = E_W(VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2, m, n)
    EB = E_B(VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2, m, n)

    E = np.concatenate((EW, EB), axis=0)

    return E
```

```
def Kirchhoff_matrix(m, n, R, R_WL, R_BL,
                     VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2,
                     VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2):

    A = A_matrix(m, n, RS_WL1, RS_WL2, R_WL, R)
    B = B_matrix(m, n, R)
    C = C_matrix(m, n, R)
    D = D_matrix(m, n, RS_BL1, RS_BL2, R_BL, R)

    K1 = np.concatenate((A, B), axis=1)
    K2 = np.concatenate((C, D), axis=1)
    K = np.concatenate((K1, K2), axis=0)

    return K

def Resistance_matrix(m, n, Ron, Roff,
                      random = True, On = True,
                      variability = False, sd=1):
    '''
    Returns resistance matrix (cell resistances)
    '''
    if random==True:
        R = np.random.rand(m, n)
        if variability == True:
            for i in range(m):
                for j in range(n):
                    if R[i][j] < 0.5:
                        Roff_var = np.random.normal(Roff, sd, 1)
                        R[i][j] = Roff_var
                    else:
                        Ron_var = np.random.normal(Ron, sd, 1)
                        R[i][j] = Ron_var
        else:
            np.where(R<0.5, Roff, Ron)
    else:
        R = np.ones((m, n))
        if On == True:
            R = R*Ron
        else:
            R = R*Roff
    return R

def Resistance_1T1R_Matrix(array_size,
                           SelWL,
                           SelBL,
                           Ron,
                           Roff,
                           Rp,
                           Z,
                           FEToff):
    m, n = array_size
    R = Resistance_matrix(m, n, Ron = Ron, Roff = Roff, sd=1)
    R = np.ones((m, n))*FEToff
    R[SelBL, :] = Rp
    R[SelBL, 0] = Z
    return R


def Vout(m, n, Rs, Ron, Roff, state=”LRS”):
    rs = Rs/Ron
    k = Roff/Ron

    if state == ”LRS”:
        V_out_min = 1/(1/rs + m)
        V_out_max = k/(k/rs + (m-1)+k)

        return (V_out_max, V_out_min)

    if state == ”HRS”:
        V_out_min = 1/(k/rs + k*(m - 1) + 1)
        V_out_max = 1/(k/rs + m)

        return (V_out_max, V_out_min)
    else:
        return 0


def simulate(senseresistance_normalized = 1e-5,
        line_resistance = 1e-2,
        random = True,
        bias_scheme = ”V/2”,
        SelWL = -1,
        SelBL = -1,
        Ron = 10e3,
        print_it = False,
        array_size=(10, 10)):
    Vdd = 1
    V_SWL = Vdd
    if bias_scheme == ”V/2”:
        V_UWL = Vdd/2
        V_SBL = 0
        V_UBL = Vdd/2
    elif bias_scheme == ”V/3”:
        V_UWL = Vdd/3
        V_SBL = 0
        V_UBL = 2*Vdd/3
    else:
        V_UWL = 0
        V_SBL = 0
        V_UBL = 0

    Roff = 10*Ron
    R_L = line_resistance
    R_access = 1

    M, N = array_size

    Sel_WL, Sel_BL = (SelWL, SelBL)

    R = Resistance_matrix(M, N, Ron, Roff, random = random, On = False,
                          variability=True, sd=1e-1*Ron)
```

```
R_WL = R_L  ; R_BL = R_L

VAPP_WL1 = V_UWL*np.ones(M)  ; VAPP_WL1[Sel_WL] = V_SWL
VAPP_WL2 = 0*np.ones(M)  ; #VAPP_WL2[Sel_WL] = V_SWL

VAPP_BL1 = V_UBL*np.ones(N)  ; VAPP_BL1[Sel_BL] = V_SBL
VAPP_BL2 = 0*np.ones(N)  ; #VAPP_BL2[-1] = 0

RS_WL1 = R_access*np.ones(M)  ; #RS_WL1[Sel_WL] = R_access
RS_WL2 = 1e9*np.ones(M)  ; #RS_WL2[-1] = 10000

RS_BL1 = R_access*np.ones(N)  ; #RS_BL1[Sel_BL] = R_access
RS_BL2 = 1e9*np.ones(N)  ; #RS_BL2[Sel_BL] = senseresistance_normalized*Ron

start = time.time()
K = Kirchhoff_matrix(M, N, R, R_WL, R_BL,
                     VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2,
                     VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2)
stop = time.time()
E = E_matrix(M, N, VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2,
      VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2)


start = time.time()
V = np.linalg.solve(K, E)

stop = time.time()

end = int(len(V)/2)
V_WL = V[:end]/Vdd
V_BL = V[end:]/Vdd

assert(all(V_BL != V_WL))
V_WL = np.reshape(V_WL, (M, N))
V_BL = np.reshape(V_BL, (M, N))

V_CELL = V_WL - V_BL


plt.imshow(V_WL, interpolation='none', cmap='viridis')
plt.title("Word␣Line␣Voltage")
plt.grid(False)
plt.colorbar()
plt.clim(0, 1)
plt.show()

plt.imshow(V_BL, interpolation='none', cmap='viridis')
plt.title("Bit␣Line␣Voltage")
plt.grid(False)
plt.colorbar()
plt.clim(0, 1)
plt.show()

plt.imshow(V_CELL, interpolation='none', cmap='viridis')
```

```
    plt.title("Voltage␣Difference")
    plt.grid(False)
    plt.colorbar()
    plt.clim(0, 1)
    plt.show()

    Sense_resistance = RS_BL2[Sel_BL]
    SM = Sense_Margin(M, N, Sense_resistance, Ron, Roff)
    if print_it == True:
        print("Sensing␣Margin:", SM, "V")
        print("Voltage␣over␣selected␣cell:", V_CELL[Sel_WL][Sel_BL])

    return {"Sensing␣Margin" : SM,
            "Delivered␣Voltage" : V_CELL[Sel_WL][Sel_BL]}

def get_bit_map(m, n, bias_scheme = "V/2", plot = True, random = True,
                    line_resistance = 1e-6, all_on=True):

    """
    Function that returns the access voltage of a passive crossbar array with each
    node selected.
    INPUT:
        m, n = array dimensions (should be square)
        bias_scheme: two different bias schemes: V/2 or V/3 give different results!
        plot: Boolean, returns a plot if true.
        random: Should the resistance (LRS or HRS) of each RRAM node be randomly
                generated throughout the array?
        line_resistance: the line resistance of the array
        all_on: Boolean, if true, all the resistances are in LRS.
    """

    start = time.time()

    Vdd = 1
    V_SWL = Vdd
    if bias_scheme == "V/2":
        V_UBL = Vdd/2
        V_UWL = Vdd/2
    else:
        V_UBL = 2*Vdd/3
        V_UWL = Vdd/3
    V_SBL = 0
    Ron = 10e3
    Roff = 10*Ron
    R_L = line_resistance*Ron
    R_access = 1
    M = m ; N = n
    Map = np.ones((M, N))
    bar = Bar('Progress', max = M)
    for i in range(M):
        bar.next()
        for j in range(N):

            Sel_WL = i ; Sel_BL = j

            R = Resistance_matrix(M, N, Ron, Roff, random = random, On = all_on,
                                    variability=True, sd=1e-4*Ron)

            R_WL = R_L ; R_BL = R_L

            VAPP_WL1 = V_UWL*np.ones(M) ; VAPP_WL1[Sel_WL] = V_SWL
            VAPP_WL2 = 0*np.ones(M) ; #VAPP_WL2[Sel_WL] = V_SWL

            VAPP_BL1 = V_UBL*np.ones(N) ; VAPP_BL1[Sel_BL] = V_SBL
            VAPP_BL2 = 0*np.ones(N) ; #VAPP_BL2[Sel_BL] = V_SBL

            RS_WL1 = R_access*np.ones(M) ; #RS_WL1[Sel_WL] = R_access
            RS_WL2 = 1e9*np.ones(M) ; #RS_WL2[-1] = 10000

            RS_BL1 = R_access*np.ones(N) ; #RS_BL1[Sel_BL] = R_access
            RS_BL2 = 1e9*np.ones(N) ; #RS_BL2[Sel_BL] = 0.1*Ron

            K = Kirchhoff_matrix(M, N, R, R_WL, R_BL,
                                 VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2,
                                 VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2)
            E = E_matrix(M, N, VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2,
                   VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2)

            V = np.linalg.solve(K, E)

            end = int(len(V)/2)
            V_WL = V[:end]/Vdd
            V_BL = V[end:]/Vdd

            assert(all(V_BL != V_WL))

            V_WL = np.reshape(V_WL, (M, N))
            V_BL = np.reshape(V_BL, (M, N))

            V_CELL = V_WL - V_BL
            Map[i][j] = V_CELL[i][j]

        time.sleep(1)


    bar.finish()

    if plot==True:
        plt.imshow(Map, interpolation='none', cmap='Blues')
        plt.title("BitMap")
        plt.xlabel("Bit␣Line")
        plt.ylabel("Word␣Line")
        plt.grid(False)
        plt.colorbar()
        plt.clim(0, 1)
        plt.show()
    else:
        stop = time.time()


if __name__ == "__main__":
    get_bit_map(8, 8)
```

B.2 1T1R Cell Model

```
"""
This module calculates an impedance matrix based on a small signal model a
one-transistor-one-RRAM (1T1R) nano wire cell.
It returns the total impedance matrix as well as impedances for the
corresponding constituents of the model.
"""
import os
import matplotlib.pyplot as plt
import numpy as np
import scipy as sc
import warnings
warnings.filterwarnings('ignore')
plt.style.use('seaborn')


def Ccoax(er, t_ox, radius, tline):
    e_0 = 8.85e-12
    num = 2*np.pi*e_0*er
    den = np.log((radius+t_ox)/(radius))
    return num/den * tline

def Cpar(area, distance, er = 18*8.85e-12):
    return area*er/distance

def Zcap(C, pulse_width):
    f = pulse_width2freq(pulse_width)
    return 1/(1j*2*np.pi*f*C)

def Cu_Res(tnm):
    t = tnm*1e9
    ro_bulk = 1.68e-8
    k1 = 20
    k2 = 1+0.5*1/(1+np.exp((t-50)/100))
    ro = ((k1*(1-np.sqrt(t/100)/(1+np.exp((t-200)/100)))+t)*(k2/t))**2*ro_bulk
    return ro

def C_coax_to_coax(L, S, D):
    r = D/2
    C_area = r**2*np.pi
    SA = S*D
    Sm = (SA-C_area)/D

    return Cpar(L*D, Sm)

def Cfringe(W, S):
    "C_pi_from_paper"
    e_di = 3.9*8.85e-12
    C = e_di/(np.pi/2)*np.log(1 + 2*W/S)
    return C


def Outside_Screening(W, D):
    """
    For 35 nm node the NW diameter encloses the interconnects. This function subtracts
    that are not included inthe capacitive coupling bewteen the BL and WL.
    """
    # Circle Segment Screening effect
    d = W/2
    R = D/2
    c = 2*R*np.sqrt(1 - (d/R)**2) # Segment Width
    k = c/(2*R)
    As = R**2*(np.arcsin(k) - k*np.sqrt(1 - k**2)) # Segment Area
    Asr = (R**2*np.pi-2*As-W**2)/2
    for i in range(len(R)):
        if c[i] < d[i]:
            Asr[i] = As[i]
    return Asr

def Corner_Screening(W, D):
    """
    This function removes the areas on the corner of the NW
    that are not included inthe capacitive coupling bewteen the BL and WL
    """
    d = W/2
    R = D/2
    c = 2*R*np.sqrt(1 - (d/R)**2) # Segment Width
    k = c/(2*R)
    As = R**2*(np.arcsin(k) - k*np.sqrt(1 - k**2)) # Segment Area
    Acrnr = -1*(R**2*np.pi-4*As-W**2)/4
    for i in range(len(R)):
        if d[i] > c[i]:
            Acrnr[i] = 0
    return Acrnr

def Fringe_Cap(W, S, H, T, x_to_x):
    """
    Different Fringe capacitances between interconnects depending on their dimensions
    INPUT:
        W: Width
        S: Spacing
        H: Height
        T: Thickness
        x_to_xo: Type of fringe capacitance: sideways (sw) and top combinations.
    """
    t = 3.7
    arg = (W + S - np.sqrt(S**2 + T**2 + 2*H*T))/(t*W)
    n = np.exp(arg)
    a = np.exp(-1*(H + T)/(S + W))
    b = np.exp((S + W)/(H + T))
    e_di = 3.9 * 8.85e-12

    if x_to_x == "sw-top":
        C = e_di/(np.pi/2)*np.log((H + n*T + np.sqrt(S**2 + (n*T)**2 +
                                    2*H*n*T))/(S + H))
    elif x_to_x == "top-top":
        C = e_di*W*a*(np.log(1+2*W/S)+
                        np.exp(-1*(S+T)/(3*S)))/(W*np.pi*a+(H+T)*(np.log(1+2*W/S)+
                                                                 np.exp(-1*(S+T)/(3*S))))
    elif x_to_x == "sw-sw":
        k1 = e_di*T*b*(np.log(1 + 2*T/H) + np.exp(-1*(H+W)/(3*H)))
        k2 = T*np.pi*b + (S+W)*(np.log(1+2*T/H)+np.exp(-1*(H+W)/(3*H)))

        C = k1/k2
    elif x_to_x == ”corner”:
        C = e_di/np.pi*np.sqrt((H*S)/(H**2+S**2))
    else:
        C = 1e-25
    return C

def pulse_width2freq(pulse_width):
    ”””
    Get the frequency for a corresponding sinusoidal pulse width
    ”””
    return 1/(2*pulse_width)

def Rpar(R1, R2):
    ”””
    Parallel resistance
    ”””
    return R1*R2/(R1+R2)

def Resistance_Matrix_1T1R(array_size,
                           Rp,
                           Z):
    ”””
    Function that gives the resistance matrix depending with a given array_size
    ”””

    m, n = array_size
    R = np.ones((m, n))*Rp
    ZA = np.ones(m)*Z
    R[:, 0] = ZA
    return R


def Impedance_Array(include_fringe=False,
                    plot_AV = True,
                    vary_resistivity=False,
                    cell = ”off”,
                    npoints = 10,
                    total_arraysize = 32,
                    sub_arraysize = 128,
                    subarray=False,
                    pulse_width = 10,
                    node = 35,
                    rho = 8e-8,
                    tox=2.8,
                    tg = 5,
                    resistivity=1.68e-8,
                    Lg = 30,
                    Lgs = 80):
    ”””
    The main function that gives the Impedance array using a small signal model
    for each cell
    ”””
```

```
if subarray == True:
    array_size = sub_arraysize
elif subarray == False:
    array_size = total_arraysize*sub_arraysize

e_0 = 8.85e-12 #Perimittivity in Vacuum
er_ox = 18 # HfO2 relative permittivity


Measurements = np.ones(1)
Node = node*1e-9*Measurements # Node geometry
Wp = np.array([pulse_width])*1e-9
U = Node # Unit length
W = U/2 # Width of interconnect
T = W*2 # thickness of interconnect


# Different Node Sizes

if node==35:
    dnw = 8*1e-9*Measurements # Nanowire diameter
    Lnw = 320*1e-9*Measurements # Nanowire Length
    Lm = 50*1e-9*Measurements # Length of RRAM NW
    tshell = 2e-9*Measurements # thickness of NW diameter in RRAM
    tox = tox*1e-9*Measurements # Oxide Thickness
    tg = tg*1e-9*Measurements# Gate thickness
    Lg = Lg*1e-9*Measurements # Gate Length
    S_gs = 80*1e-9*Measurements # Distance between gate and source in FET

if node==50:
    dnw = 12*1e-9*Measurements # Nanowire diameter
    Lnw = 480*1e-9*Measurements # Nanowire Length
    Lm = 75*1e-9*Measurements # Length of RRAM NW
    tshell = 3e-9*Measurements # thickness of NW diameter in RRAM
    tox = tox*1e-9*Measurements # Oxide Thickness
    tg = 7*1e-9*Measurements# Gate thickness
    Lg = 40*1e-9*Measurements # Gate Length
    S_gs = 120*1e-9*Measurements # Distance between gate and source in FET

if node==100:
    dnw = 18*1e-9*Measurements # Nanowire diameter
    Lnw = 720*1e-9*Measurements # Nanowire Length
    Lm = 100*1e-9*Measurements # Length of RRAM NW
    tshell = 4e-9*Measurements # thickness of NW diameter in RRAM
    tox = tox*1e-9*Measurements # Oxide Thickness
    tg = 10*1e-9*Measurements# Gate thickness
    Lg = 60*1e-9*Measurements # Gate Length
    S_gs = 160*1e-9*Measurements # Distance between gate and source in FET

H_Wplug = W*2 # Viaplug
S_WLBL = Lnw-S_gs-T+H_Wplug # Distance WL to BL
S_WLS = S_WLBL+S_gs # Distance WL till Substrate
dDEV = dnw+2*(tshell+tox+tg) # total diameter of RRAM + shell and everything

if node==35:
    Ascrn = Outside_Screening(W, dDEV) # NW larger than interconnects
    A_WLS = W**2 - 2*Ascrn
    A_BLS = Outside_Screening(W, dDEV - 2*tg)# NW larger than  interconnects
    A_BLS = W**2 - 2*A_BLS
    A_WLBL = Outside_Screening(W, dDEV)
else:
    A_WLS = W*U - W**2
    A_BLS = W*U - np.pi*(dDEV/2)**2
    A_WLBL = W**2 - np.pi*(dDEV/2)**2

Wscrn = A_WLS/W
Sscrn = W - Wscrn
Side_fringe = W * Fringe_Cap(W = Wscrn, S = Sscrn,
                             H = S_WLBL, T = T, x_to_x = 'sw-top')

Overlay = Cpar(A_WLBL, S_WLBL, er_ox*e_0)
C_BLS = Cpar(A_BLS, S_gs, er_ox*e_0)
C_WLBL = Overlay + 2*Side_fringe

Ascrn2 = Corner_Screening(W, dDEV)
Wscrn2 = (W**2 - 4*Ascrn2)/W
Sscrn2 = (W - Wscrn2)/2
Side_fringe2 = W * Fringe_Cap(W = Wscrn2, S = Sscrn2,
                             H = S_WLS, T = T, x_to_x = 'sw-top')
Overlay2 = Cpar(A_WLS, S_WLS, er_ox*e_0)
C_WLS = Overlay2 + Side_fringe2

NW_to_NW = C_coax_to_coax(Lm, U, dDEV) # NW to NW capacitance
Via_to_Via = Cpar(W*H_Wplug, W, er_ox*e_0)
WL_to_WL = Cpar(2*W*T, W) # WL to WL parallel plate capacitance

range_param = [int(round(2e-6/x)) for x in Node]
WL_fringeC = np.zeros((len(W), max(range_param)))
WL_to_WL_fringe = np.zeros(len(Measurements))
for i in range(len(W)):
    for j in range(range_param[i]):
        WL_fringeC[i][j] = 2*W[i]*Cfringe(W[i], (1+2*j)*W[i])
    WL_to_WL_fringe[i] = sum(WL_fringeC[i])



C_WLWL = 2*(2*NW_to_NW + Via_to_Via +
            WL_to_WL + 1.5*WL_to_WL_fringe) # Total WL to WL capacitance
R_FET_off = 20*25e6 # OP transistors
R_FET_on = 20e3 # OP transistors from nanoletters
R_RRAM_on = 17.2*1e3 # The RRAM in LRS from experiments
R_RRAM_off = 18.391*1e6 # The RRAM in HRS from experiments
C_RRAM = Ccoax(18, tox, dDEV - 2*tg, Lm) # capacitance of RRAM
C_FET = Ccoax(18, tox, dDEV -2*(tg + tshell), Lg) # Capacitance of FET
C_GS = 0.1*C_FET/2 # Gate-source Parasitic FET capacitance
C_GD = C_GS # The same for gate-drain parasitic

Z_WLWL = Zcap(C_WLWL, Wp)
Z_WLBL = Zcap(C_WLBL, Wp)
Z_WLS = Zcap(C_WLS, Wp)

ZC_RRAM = Zcap(C_RRAM, Wp) # Capacitive impedance of RRAM
```

```
ZR_RRAM = R_RRAM_on # Resistive impedance of RRAM
ZR_FET = R_FET_off # Resistive impedance of FET in off mode
ZC_GD = Zcap(C_GD, Wp) # Impedance due to parasitic gate-drain capacitance of FET


if vary_resistivity == True:
    Res = resistivity
else:
    Res = Cu_Res(W) # Resistivity of copper

R_WL = Res*2*W/(W*T) # Line Resistance of WL interconnect

C_BL = 2*C_BLS + C_GS
R_SD = abs(ZC_GD)
Z_BL = Zcap(C_BL, Wp) # Loss from MOSFET
Z_BL = Rpar(Z_BL, R_SD)
C_WL = C_WLWL + C_WLBL*C_BLS/(C_WLBL+C_BLS) + C_WLS # Total WL capacitance


Z_RRAM = ZC_RRAM*ZR_RRAM/(ZC_RRAM + ZR_RRAM) # Total RRAM impedance
Z_FET = ZR_FET*ZC_GD/(ZC_GD+ZR_FET) # Total FET impedance
Z_NW = Z_RRAM + Z_FET # Total nanowire impedance

Z_RRAM_on = ZC_RRAM*R_RRAM_on/(ZC_RRAM + R_RRAM_on)
Z_RRAM_off = ZC_RRAM*R_RRAM_off/(ZC_RRAM + R_RRAM_off)
Z_FET_on = R_FET_on*ZC_GD/(ZC_GD+R_FET_on)
Z_NW_on = Z_FET_on + Z_RRAM_on
Z_NW_off = Z_FET_on + Z_RRAM_off



Z_WL = Zcap(C_WL, Wp)
Z_TOT = Z_NW*Z_WL/(Z_NW+Z_WL)


N = npoints

Rvertical = abs(Z_BL)
Rvertical_NEW = Rvertical
Rline_new = 0
Rij_new = 0
Z = 0

if plot_AV == True:
    X = np.logspace(0, 4.5, N, base=10.0).astype(int)
else:
    X = np.ones(1)*array_size

if cell == "off":
    Zcell = abs(Z_NW_off)
else:
    Zcell = abs(Z_NW_on)
u = 0
Z0 = Zcell[u]
R = R_WL[u]
Rij = abs(Z_TOT[u])
```

```
    Rij_new = Rij

    R_BL = np.ones(array_size)
    Z = Z0
    Z2 = Z_BL
    for i in range(array_size):
        Rvertical_NEW = Rpar(Rvertical_NEW, Rvertical)
        R_BL[i] = Rvertical_NEW
        Z2 += R
        Z2 = Z2*Rvertical/(Z2+Rvertical)
    Rvertical_NEW = 0


    for i in range(array_size):
        Rline_new += R
        Rij_new = Rpar(Rij_new, Rij)
        Z = Z + R
        Z = (Z*Rij)/(Z+Rij)

    if subarray == True:
        R_Matrix = Resistance_Matrix_1T1R(array_size=(total_arraysize,
        total_arraysize), Rp=Rij_new, Z=Z)
        return([R_Matrix, Rline_new, abs(Z_NW_on),
                abs(Z_NW_off), abs(Z_TOT), abs(Z_WL),
                abs(Z_NW), abs(Z_WLWL), abs(Z_WLBL),
                abs(Z_WLS), abs(Z_BL), abs(R_SD)])


    elif subarray ==  False:
        return Z+Z2
    else:
        raise ValueError("subarray must be true or false")
```

B.3 1T1R Array Model

```
"""
Module that simulates 1T1R performance. The matrices included are A, B, C, D, Ew, Eb
and 'Kirchoff' calculate Kirchoff's continuity equations for each node of a given
array size. The function "simulate" simulates the array.

"""

# Import dependencies
import matplotlib.pyplot as plt
import numpy as np
import time
import warnings
from progress.bar import Bar
warnings.filterwarnings('ignore')
plt.style.use('seaborn-dark')


def A_i(RS_WL1, RS_WL2, R_WL, R, i, n):
    lower_diagonal = np.ones(n-1)*(-1)/R_WL
    upper_diagonal = np.ones(n-1)*(-1)/R_WL
    diagonal = (1/R[i])  + (2/R_WL)
    diagonal[0] = 1/RS_WL1[i] + 1/R[i][0] + 1/R_WL
    diagonal[-1] = 1/RS_WL2[i] + 1/R[i][n-1] + 1/R_WL

    Ai = np.zeros((n, n))
    u = np.diag(upper_diagonal, k=1)
    l = np.diag(lower_diagonal, k=-1)
    d = np.diag(diagonal)
    Ai += u+l+d

    return Ai

def A_matrix(m, n, RS_WL1, RS_WL2, R_WL, R):
    A = np.zeros((m*n, m*n))
    k = 0
    for i in range(m):
        A[k:k+n, k:k+n] = A_i(RS_WL1, RS_WL2, R_WL, R, i, n)
        k += m
    return A

def B_i(R, n, i):
    diagonal = np.ones(n)*(-1)/R[i]

    Bi = np.diag(diagonal)

    return Bi

def B_matrix(m, n, R):
    B = np.zeros([m*n, m*n])
    k = 0
    for i in range(m):
        B[k:k+n, k:k+n] = B_i(R, n, i)
        k += m
    return B

def C_j(m, n,  R, j):
    Cj = np.zeros([m, n*m])
    for i in range(0, m):
        Cj[i][n*(i) + j] = 1/R[i][j]

    return Cj

def C_matrix(m, n, R):

    C = C_j(m, n, R, 0)

    for j in range(1, n):
        C_old = C
        C_add = C_j(m, n, R, j)
        C = np.concatenate((C_old, C_add), axis=0)

    return C

def D_j(m, n, RS_BL1, RS_BL2, R_BL, R, j):
    Dj = np.zeros([m, n*m])
    for i in range(m):
        if i == 0:
            Dj[i][j] = (-1/RS_BL1[j] -1/R_BL -1/R[i][j])
            Dj[i][n + j] = 1/R_BL
        elif (1<=i<=(m-2)):
            Dj[i][n*(i-1) + j] = 1/R_BL
            Dj[i][n*(i-0) + j] = (-1/R_BL -1/R[i][j] -1/R_BL)
            Dj[i][n*(i+1) + j] = 1/R_BL
        elif (i == m-1):
            Dj[i][n*(i-1) + j] = 1/R_BL
            Dj[i][n*(i-0) + j] = (-1/RS_BL2[j] -1/R[i][j] -1/R_BL)


    return Dj

def D_matrix(m, n, RS_BL1, RS_BL2, R_BL, R):

    D =D_j(m, n, RS_BL1, RS_BL2, R_BL, R, 0)

    for j in range(1, n):
        D_old = D
        D_add = D_j(m, n, RS_BL1, RS_BL2, R_BL, R, j)
        D = np.concatenate((D_old, D_add), axis=0)

    return D

def E_Wi(VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2, n, i):
    E = np.zeros(n)
    E[0] = VAPP_WL1[i]/RS_WL1[i]
    E[-1] = VAPP_WL2[i]/RS_WL2[i]

    return E

def E_Bj(VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2, m, j):
    E = np.zeros(m)

    E[0] = -VAPP_BL1[j]/RS_BL1[j]
    E[-1] = -VAPP_BL2[j]/RS_BL2[j]

    return E

def E_W(VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2, m, n):
    Ew = E_Wi(VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2, n, i = 0)
    for i in range(1, m):
        E_old = Ew
        E_add = E_Wi(VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2, n, i)
        Ew = np.concatenate((E_old, E_add), axis=0)

    return Ew

def E_B(VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2, m, n):
    Eb = E_Bj(VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2, m, j = 0)
    for j in range(1, n):
        E_old = Eb
        E_add = E_Bj(VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2, m, j)

        Eb = np.concatenate((E_old, E_add), axis=0)
    return Eb

def E_matrix(m, n, VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2,
      VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2):
    EW = E_W(VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2, m, n)
    EB = E_B(VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2, m, n)

    E = np.concatenate((EW, EB), axis=0)

    return E

def Kirchhoff_matrix(m, n, R, R_WL, R_BL,
                     VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2,
                     VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2):

    A = A_matrix(m, n, RS_WL1, RS_WL2, R_WL, R)
    B = B_matrix(m, n, R)
    C = C_matrix(m, n, R)
    D = D_matrix(m, n, RS_BL1, RS_BL2, R_BL, R)

    K1 = np.concatenate((A, B), axis=1)
    K2 = np.concatenate((C, D), axis=1)
    K = np.concatenate((K1, K2), axis=0)

    return K

def Rpar(R1, R2):
    """
    Resistances coupled in parallel
    """
    return R1*R2/(R1+R2)



def simulate(RMatrix = np.ones([32, 32]),
        line_resistance = 10,
        access_resistance = 1,
        supply_voltage=1,
        array_size=(10, 10)):
    """
    Function that simulates an RRAM 1T1R crossbar array and gives the access voltage of
    each node. The BL acts as the line connected to the gate of the transistor.

    INPUTS:
        RMatrix: a resistance matrix representing the resistance of each node.
        line_resistance: the line resistance of the array
        access_resistance: the access resistance for each access point
                            (should be very small, about 1-10 Ohm)
        supply_voltage: the applied voltage on the array
        array_size: the dimensions of the crossbar array
    RETURNS:
        Array of the access voltage to each node.

    """
    Vdd = supply_voltage
```

```
R_L = line_resistance
R_access = access_resistance # access resistance (small)
M, N = array_size

R = RMatrix

R_WL = R_L ; R_BL = 10 # R_BL is very small due to the integration of 1T1R subarrays
                        # R_BL and R_WL should be the same

VAPP_WL1 = Vdd*np.ones(M)
VAPP_WL2 = 0*np.ones(M)

VAPP_BL1 = 0*np.ones(N)
VAPP_BL2 = 0*np.ones(N)

RS_WL1 = R_access*np.ones(M)
RS_WL2 = 1e10*np.ones(M)

RS_BL1 = R_access*np.ones(N)
RS_BL2 = R_access*np.ones(N)

# The total Matrix
K = Kirchhoff_matrix(M, N, R, R_WL, R_BL,
                     VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2,
                     VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2)

E = E_matrix(M, N, VAPP_BL1, RS_BL1, VAPP_BL2, RS_BL2,
      VAPP_WL1, RS_WL1, VAPP_WL2, RS_WL2)

# solve the 2*n*m equations
V = np.linalg.solve(K, E)


end = int(len(V)/2)
V_WL = V[:end]
V_BL = V[end:]

assert(all(V_BL != V_WL))
V_WL = np.reshape(V_WL, (M, N))
V_BL = np.reshape(V_BL, (M, N))

# The access voltage is the potential difference between the WL and BL planes.

Vaccess = V_WL - V_BL
return Vaccess
```